Union Subgraph Neural Networks

Jiaxing Xu*, Aihu Zhang*, Qingtian Bian, Vijay Prakash Dwivedi, Yiping Ke

Nanyang Technological University, Singapore
{jiaxing003,zhan0547,bian0027,vijaypra001}@e.ntu.edu.sg, ypke@ntu.edu.sg

Abstract

Graph Neural Networks (GNNs) are widely used for graph representation learning in many application domains. The expressiveness of vanilla GNNs is upper-bounded by 1-dimensional Weisfeiler-Leman (1-WL) test as they operate on rooted subtrees through iterative message passing. In this paper, we empower GNNs by injecting neighbor-connectivity information extracted from a new type of substructure. We first investigate different kinds of connectivities existing in a local neighborhood and identify a substructure called union subgraph, which is able to capture the complete picture of the 1-hop neighborhood of an edge. We then design a shortest-path-based substructure descriptor that possesses three nice properties and can effectively encode the high-order connectivities in union subgraphs. By infusing the encoded neighbor connectivities, we propose a novel model, namely Union Subgraph Neural Network (UnionSNN), which is proven to be strictly more powerful than 1-WL in distinguishing non-isomorphic graphs. Additionally, the local encoding from union subgraphs can also be injected into arbitrary message-passing neural networks (MPNNs) and Transformer-based models as a plugin. Extensive experiments on 18 benchmarks of both graph-level and node-level tasks demonstrate that UnionSNN outperforms state-of-the-art baseline models, with competitive computational efficiency. The injection of our local encoding to existing models is able to boost the performance by up to 11.09%. Our code is available at https://github.com/AngusMonroe/UnionSNN.

1 Introduction

With the ubiquity of graph-structured data emerging from various modern applications, Graph Neural Networks (GNNs) have gained increasing attention from both researchers and practitioners. GNNs have been applied to many application domains, including quantum chemistry (Duvenaud et al. 2015; Dai, Dai, and Song 2016; Masters et al. 2022), social science (Ying et al. 2018; Fan et al. 2019; Shiao et al. 2022), transportation (Peng et al. 2020; Derrow-Pinion et al. 2021) and neuroscience (Ahmedt-Aristizabal et al. 2021; Wang et al. 2021), and have attained promising results on graph classification (Ying et al. 2018; Masters et al. 2022), node classification (Shi et al. 2020) and link prediction (Zhang and Chen 2018; Suresh et al. 2023) tasks.

Most GNNs are limited in terms of their expressive power. Xu et al., (Xu et al. 2018) show that GNNs are at most as powerful as 1-dimentional Weisfeiler-Leman (1-WL) test (Weisfeiler and Leman 1968) in distinguishing non-isomorphic graph structures. This is because a vanilla GNN essentially operates on a subtree rooted at each node in its message passing, *i.e.*, it treats every neighbor of the node equally in its message aggregation. In this regard, it overlooks any discrepancy that may exist in the connectivities between neighbors. To address this limitation, efforts have been devoted to incorporating local substructure information to GNNs. Several studies attempt to encode such local information through induced subgraphs (Zhao et al. 2021), overlap subgraphs (Wijesinghe and Wang 2021) and spatial encoding (Bouritsas et al. 2022), to enhance GNNs' expressiveness. But the local structures they choose are not able to capture the complete picture of the 1-hop neighborhood of an edge. Some others incorporate shortest path information to edges in message passing via distance encoding (Li et al. 2020), adaptive breath/depth functions (Liu et al. 2019) and affinity matrix (Wan et al. 2021) to control the message from neighbors at different distances. However, the descriptor used to encode the substructure may overlook some connectivities between neighbors. Furthermore, some of the above models also suffer from high computational cost due to the incorporation of certain substructures.

In this paper, we aim to develop a model that overcomes the above drawbacks and yet is able to empower GNNs' expressiveness. (1) We define a new type of substructures named union subgraphs, each capturing the entire closed neighborhood w.r.t. an edge. (2) We design an effective substructure descriptor that encodes high-order connectivities and it is easy to incorporate with arbitrary message-passing neural network (MPNN) or Transformer-based models. (3) We propose a new model, namely Union Subgraph Neural Network (UnionSNN), which is strictly more expressive than the vanilla GNNs (1-WL) in theory and also computationally efficient in practice. Our contributions are summarized as follows:

- We investigate different types of connectivities existing in the local neighborhood and identify the substructure, named "union subgraph", that is able to capture the com-

*These authors contributed equally.

plete 1-hop neighborhood.

- We abstract three desirable properties for a good substructure descriptor and design a shortest-path-based descriptor that possesses all the properties with high-order connectivities encoded.
- We propose a new model, UnionSNN, which incorporates the information extracted from union subgraphs into the message passing. We also show how our local encoding can be flexibly injected into any arbitrary MPNNs and Transformer-based models. We theoretically prove that UnionSNN is more expressive than 1-WL. We also show that UnionSNN is stronger than 3-WL in some cases.
- We perform extensive experiments on both graph-level and node-level tasks. UnionSNN consistently outperforms baseline models on 18 benchmark datasets, with competitive efficiency. The injection of our local encoding is able to boost the performance of base models by up to 11.09%, which justifies the effectiveness of our proposed union subgraph and substructure descriptor in capturing local information.

2 Related Work

2.1 Substructure-Enhanced GNNs

In recent years, several GNN architectures have been designed to enhance their expressiveness by encoding local substructures. GraphSNN (Wijesinghe and Wang 2021) brings the information of overlap subgraphs into the message passing scheme as a structural coefficient. However, the overlap subgraph and the substructure descriptor used by GraphSNN are not powerful enough to distinguish all non-isomorphic substructures in the 1-hop neighborhood. Zhao et al. (Zhao et al. 2021) encode the induced subgraph for each node and inject it into node representations. (Bouritsas et al. 2022) introduces structural biases in the aggregation function to break the symmetry in message passing. For these two methods, the neighborhood under consideration should be pre-defined, and the subgraph matching is extremely expensive ($O(n^k)$ for k-tuple substructure) when the substructure gets large. Similarly, a line of research (Bodnar et al. 2021; Thiede, Zhou, and Kondor 2021; Horn et al. 2021) develops new WL aggregation schemes to take into account substructures like cycles or cliques. Despite these enhancements, performing cycle counting is very time-consuming. Other Transformer-based methods (Dwivedi and Bresson 2020; Kreuzer et al. 2021; Wu et al. 2021; Mialon et al. 2021) incorporate local structural information via positional encoding (Lim et al. 2022; Zhang et al. 2023). Graphormer (Ying et al. 2021) combines the node degree and the shortest path information for spatial encoding, while other works (Li et al. 2020; Dwivedi et al. 2021) employ random walk based encodings that can encode k-hop neighborhood information of a node. However, these positional encodings only consider relative distances from the center node and ignore high-order connectivities between the neighbors.

2.2 Path-Related GNNs

A significant amount of works have focused on the application of shortest paths and their related techniques to GNNs. Li et al. (Li et al. 2020) present a distance encoding module to augment node features and control the receptive field of message passing. GeniePath (Liu et al. 2019) proposes an adaptive breath function to learn the importance of different-sized neighborhoods and an adaptive depth function to extract and filter signals from neighbors within different distances. PathGNN (Tang et al. 2020) imitates how the Bellman-Ford algorithm solves the shortest path problem in generating weights when updating node features. SPN (Abboud, Dimitrov, and Ceylan 2022) designs a scheme, in which the representation of a node is propagated to each node in its shortest path neighborhood. Some recent works adapt the concept of curvature from differential geometry to reflect the connectivity between nodes and the possible bottleneck effects. CurvGN (Ye et al. 2019) reflects how easily information flows between two nodes by graph curvature information, and exploits curvature to reweigh different channels of messages. Topping et al. (Topping et al. 2021) propose Balanced Forman curvature that better reflects the edges having bottleneck effects, and alleviate the over-squashing problem of GNNs by rewiring graphs. SNALS (Wan et al. 2021) utilizes an affinity matrix based on shortest paths to encode the structural information of hyperedges. Our method is different from these existing methods by introducing a shortest-path-based substructure descriptor for distinguishing non-isomorphic substructures.

3 Local Substructures to Empower MPNNs

In this section, we first introduce MPNNs. We then investigate what kind of local substructures are beneficial to improve the expressiveness of MPNNs.

3.1 Message Passing Neural Networks

We represent a graph as $G = (V, E, X)$, where $V = \{v_1, ..., v_n\}$ is the set of nodes, $E \in V \times V$ is the set of edges, and $X = \{\mathbf{x}_v \mid v \in V\}$ is the set of node features. The set of neighbors of node v is denoted by $\mathcal{N}(v) = \{u \in V \mid (v, u) \in E\}$. The l-th layer of an MPNN (Xu et al. 2018) can be written as:

$$\mathbf{h}_v^{(l)} = \mathrm{AGG}^{(l-1)}(\mathbf{h}_v^{(l-1)}, \mathrm{MSG}^{(l-1)}(\{\mathbf{h}_u^{(l-1)}, u \in \mathcal{N}(v)\})), \quad (1)$$

where $\mathbf{h}_v^{(l)}$ is the representation of node v at the l-th layer, $\mathbf{h}_v^{(0)} = \mathbf{x}_v$, $\mathrm{AGG}(\cdot)$ and $\mathrm{MSG}(\cdot)$ denote the aggregation and message functions, respectively.

3.2 Local Substructures to Improve MPNNs

According to Eq. (1), MPNN updates the representation of a node isotropously at each layer and ignores the structural connections between the neighbors of the node. Essentially, the local substructure utilized in the message passing of MPNN is a subtree rooted at the node. Consequently, if two non-isomorphic graphs have the same set of rooted subtrees, they cannot be distinguished by MPNN (and also 1-WL). Such an example is shown in Figure 1(a). A simple fix to this

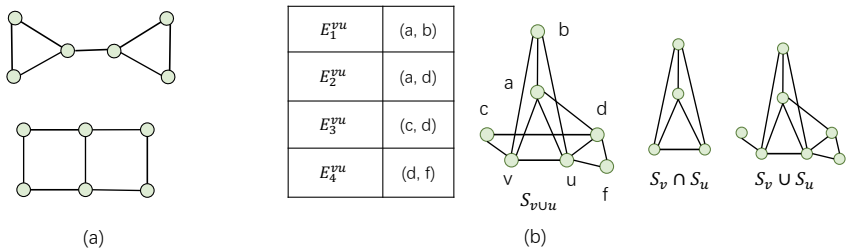


Figure 1: (a) A pair of non-isomorphic graphs not distinguishable by 1-WL; (b) An example of various local substructures for two adjacent nodes v and u.

problem is to encode the local structural information about each neighbor, based on which neighbors are treated unequally in the message passing. One natural question arises: **which substructure shall we choose to characterize the 1-hop local information?**

To answer the above question, we consider two adjacent nodes v and u, and discuss different types of edges that may exist in their neighbor sets, $\mathcal{N}(v)$ and $\mathcal{N}(u)$. We define the closed neighbor set of node v as $\tilde{\mathcal{N}}(v) = \mathcal{N}(v) \cup \{v\}$. The induced subgraph of $\tilde{\mathcal{N}}(v)$ is denoted by S_v, which defines the closed neighborhood of v. The common closed neighbor set of v and u is $\mathcal{N}_{vu} = \tilde{\mathcal{N}}(v) \cap \tilde{\mathcal{N}}(u)$ and the exclusive neighbor set of v w.r.t u is defined as $\mathcal{N}_v^{-u} = \tilde{\mathcal{N}}(v) - \mathcal{N}_{vu}$. As shown in Figure 1(b), there are four types of edges in the closed neighborhood of $\{v, u\}$:

- $E_1^{vu} \in \mathcal{N}_{vu} \times \mathcal{N}_{vu}$: edges between the common closed neighbors of v and u, such as (a, b);
- $E_2^{vu} \in (\mathcal{N}_{vu} \times \mathcal{N}_v^{-u}) \cup (\mathcal{N}_{vu} \times \mathcal{N}_u^{-v})$: edges between a common closed neighbor of v and u and an exclusive neighbor of v/u, such as (a, d);
- $E_3^{vu} \in \mathcal{N}_v^{-u} \times \mathcal{N}_u^{-v}$: edges between two exclusive neighbors of v and u from different sides, such as (c, d);
- $E_4^{vu} \in (\mathcal{N}_v^{-u} \times \mathcal{N}_v^{-u}) \cup (\mathcal{N}_u^{-v} \times \mathcal{N}_u^{-v})$: edges between two exclusive neighbors of v or u from the same side, such as (d, f).

We now discuss three different local substructures, each capturing a different set of edges.

Overlap Subgraph(Wijesinghe and Wang 2021). The overlap subgraph of two adjacent nodes v and u is defined as $S_{v\cap u} = S_v \cap S_u$. The overlap subgraph contains only edges in E_1^{vu}.

Union Minus Subgraph. The union minus subgraph of two adjacent nodes v, u is defined as $S_{v\cup u}^{-} = S_v \cup S_u$. The union minus subgraph consists of edges in E_1^{vu}, E_2^{vu} and E_4^{vu}.

Union Subgraph. The union subgraph of two adjacent nodes v and u, denoted as $S_{v\cup u}$, is defined as the induced subgraph of $\tilde{\mathcal{N}}(v) \cup \tilde{\mathcal{N}}(u)$. The union subgraph contains all four types of edges mentioned above.

It is obvious that union subgraph captures the whole picture of the 1-hop neighborhood of two adjacent nodes. This subgraph captures all types of connectivities within the neighborhood, providing an ideal local substructure for enhancing the expressive power of MPNNs. Note that we restrict the discussion to the 1-hop neighborhood because we aim to develop a model based on the MPNN scheme, in which a single layer of aggregation is performed on the 1-hop neighbors.

3.3 Union Isomorphism

We now proceed to define the isomorphic relationship between the neighborhoods of two nodes i and j based on union subgraphs. The definition follows that of overlap isomorphism in (Wijesinghe and Wang 2021).

Overlap Isomorphism. S_i and S_j are overlap-isomorphic, denoted as $S_i \simeq_{overlap} S_j$, if there exists a bijective mapping g: $\tilde{\mathcal{N}}(i) \rightarrow \tilde{\mathcal{N}}(j)$ such that $g(i) = j$, and for any $v \in \mathcal{N}(i)$ and $g(v) = u$, $S_{i\cap v}$ and $S_{j\cap u}$ are isomorphic (ordinary graph isomorphic).

Union Isomorphism. S_i and S_j are union-isomorphic, denoted as $S_i \simeq_{union} S_j$, if there exists a bijective mapping g: $\tilde{\mathcal{N}}(i) \rightarrow \tilde{\mathcal{N}}(j)$ such that $g(i) = j$, and for any $v \in \mathcal{N}(i)$ and $g(v) = u$, $S_{i\cup v}$ and $S_{j\cup u}$ are isomorphic (ordinary graph isomorphic).

As shown in Figure 2, we have two subgraphs G_1 and G_2. These two subgraphs are overlap-isomorphic. The bijective mapping g is: $g(k_1) = g(k_2), \forall k \in \{v, a, b, c, d, e, f, u, h\}$. Take a pair of corresponding overlap subgraphs as an example: $S_{v_1\cap u_1}$ and $S_{v_2\cap u_2}$. They are based on two red edges (v_1, u_1), and (v_2, u_2) and the rest of edges in the overlap subgraphs are colored in blue. It is easy to see that $S_{v_1\cap u_1}$ and $S_{v_2\cap u_2}$ are isomorphic (ordinary one). In this ordinary graph isomorphism, its bijective mapping between the nodes is not required to be the same as g. It could be the same or different, as long as the ordinary graph isomorphism holds between this pair of overlap subgraphs. In this example, any pair of corresponding overlap subgraphs defined under g are isomorphic (ordinary one). In fact, all overlap subgraphs have the same structure in this example. In this sense, the concept of "overlap isomorphism" does not look at the neighborhood based on a single edge, but captures the neighborhoods of all edges and thus the "overlap" connectivities of the two subgraphs G_1 and G_2. As for the union subgraph concept we define, the union subgraph of nodes v_1 and u_1 ($S_{v_1\cup u_1}$) is the same as G_1, and the union subgraph of nodes v_2 and u_2 ($S_{v_2\cup u_2}$) is the same as G_2. Therefore, $S_{v_1\cup u_1}$ and $S_{v_2\cup u_2}$ are not union-isomorphic. This means that union isomorphism is able to distinguish the local structures centered at v_1 and v_2 but overlap isomorphism cannot.

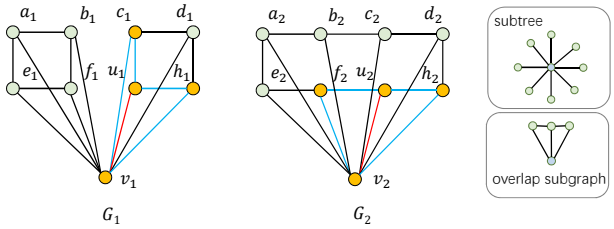


Figure 2: An example that the bijective mapping between the nodes in the subgraphs is not the same as g.

Theorem 1. *If $S_i \simeq_{union} S_j$, then $S_i \simeq_{overlap} S_j$; but not vice versa.*

Theorem 1 states that union-isomorphism is stronger than overlap-isomorphism. The detailed proofs of all theorems are available in our arXiv version (Xu et al. 2023).

4 UnionSNN

4.1 Design of Substructure Descriptor Function

Let $\mathcal{U} = \{S_{v\cup u} | (v,u) \in E\}$ be the set of union subgraphs in G. In order to fuse the information of union subgraphs in message passing, we need to define a function $f(\cdot)$ to describe the structural information of each $S_{v\cup u} \in \mathcal{U}$. Ideally, given two union subgraphs centered at node v, $S_{v\cup u} = (V_{v\cup u}, E_{v\cup u})$ and $S_{v\cup u'} = (V_{v\cup u'}, E_{v\cup u'})$, we want $f(S_{v\cup u}) = f(S_{v\cup u'})$ iff $S_{v\cup u}$ and $S_{v\cup u'}$ are isomorphic. We abstract the following properties of a good substructure descriptor function $f(\cdot)$:

- **Size Awareness**. $f(S_{v\cup u}) \neq f(S_{v\cup u'})$ if $|V_{v\cup u}| \neq |V_{v\cup u'}|$ or $|E_{v\cup u}| \neq |E_{v\cup u'}|$;
- **Connectivity Awareness**. $f(S_{v\cup u}) \neq f(S_{v\cup u'})$ if $|V_{v\cup u}| = |V_{v\cup u'}|$ and $|E_{v\cup u}| = |E_{v\cup u'}|$ but $S_{v\cup u}$ and $S_{v\cup u'}$ are not isomorphic;
- **Isomorphic Invariance**. $f(S_{v\cup u}) = f(S_{v\cup u'})$ if $S_{v\cup u}$ and $S_{v\cup u'}$ are isomorphic.

Figure 3 illustrates the properties. Herein, we design $f(\cdot)$ as a function that transforms $S_{v\cup u}$ to a path matrix $\boldsymbol{P}^{vu} \in \mathbb{R}^{|V_{v\cup u}|\times|V_{v\cup u}|}$ such that each entry:

$$\boldsymbol{P}^{vu}_{ij} = \mathrm{PathLen}(i, j, S_{v\cup u}), i, j \in V_{v\cup u}, \tag{2}$$

where $\mathrm{PathLen}(\cdot)$ denotes the length of the shortest path between i and j in $S_{v\cup u}$. We choose the path matrix over the adjacency matrix or the Laplacian matrix as it explicitly encodes high-order connectivities between the neighbors. In addition, with a fixed order of nodes, we can get a unique $\boldsymbol{P}^{vu}$ for a given $S_{v\cup u}$, and vice versa. We formulate it in Theorem 2.

Theorem 2. *With a fixed order of nodes in the path matrix, we can obtain a unique path matrix $\boldsymbol{P}^{vu}$ for a given union subgraph $S_{v\cup u}$, and vice versa.*

It is obvious that our proposed $f(\cdot)$ satisfies the above-mentioned three properties, with a node permutation applied in the isomorphic case.

Discussion on Other Substructure Descriptor Functions. In the literature, some other functions have also been proposed to describe graph substructures. (1) Edge Betweenness (Brandes 2001) is defined by the number of shortest paths between any pair of nodes in a (sub)graph G that pass through an edge. When applying the edge betweenness to (v,u) in $S_{v\cup u}$, the metric would remain the same on two different union subgraphs, one with an edge in E_4^{vu} and one without. This shows that edge betweenness does not satisfy Size Awareness; (2) Wijesinghe and Wang (Wijesinghe and Wang 2021) puts forward a substructure descriptor as a function of the number of nodes and edges. This descriptor fails to distinguish non-isomorphic subgraphs with the same size, and thus does not satisfy Connectivity Awareness; (3) Discrete Graph Curvature, e.g., Olliveier Ricci curvature (Ollivier 2009; Lin, Lu, and Yau 2011), has been introduced to MPNNs in recent years (Ye et al. 2019). Ricci curvature first computes for each node a probability vector of length $|V|$ that characterizes a uniform propagation distribution in the neighborhood. It then defines the curvature of two adjacent nodes as the Wasserstein distance of their corresponding probability vectors. Similar to edge betweenness, curvature doesn't take into account the edges in E_4^{vu} in its computation and thus does not satisfy Size Awareness either.

4.2 Network Design

For the path matrix of an edge (v,u) to be used in message passing, we need to further encode it as a scalar. We perform Singular Value Decomposition (SVD) (Horn and Johnson 2012) on the path matrix and extract the singular values: $\boldsymbol{P} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*$. The sum of the singular values of $\boldsymbol{P}^{vu}$, denoted as $a^{vu} = \mathrm{sum}(\boldsymbol{\Sigma}^{vu})$, is used as the local structural coefficient of the edge $(v,u) \in E$. Note that since the local structure never changes in message passing, we can compute the structural coefficients in preprocessing before the training starts. A nice property of this structural coefficient is that, it is **permutation invariant** thanks to the use of SVD and the sum operator. With arbitrary order of nodes, the computed a^{vu} remains the same, which removes the condition required by Theorem 2.

UnionSNN. We now present our model, namely Union Subgraph Neural Network (UnionSNN), which utilizes union-subgraph-based structural coefficients to incorporate local substructures in message passing. For each vertex $v \in V$, the node representation at the l-th layer is generated by:

$$\boldsymbol{h}_v^{(l)} = \mathrm{MLP}_1{}^{(l-1)}((1+\epsilon^{(l-1)})\boldsymbol{h}_v^{(l-1)} + \sum_{u\in\mathcal{N}(v)} \mathrm{Trans}^{(l-1)}(\tilde{a}^{vu})\boldsymbol{h}_u^{(l-1)}), \tag{3}$$

where $\epsilon^{(l-1)}$ is a learnable scalar parameter and $\tilde{a}^{vu} = \frac{a^{vu}}{\sum_{u\in\mathcal{N}(v)} a^{vu}}$. $\mathrm{MLP}_1(\cdot)$ denotes a multilayer perceptron (MLP) with a non-linear function ReLU. To transform the weight $\tilde{a}^{vu}$ to align with the multi-channel representation $\boldsymbol{h}_u^{(l-1)}$, we follow (Ye et al. 2019) and apply a transformation function $\mathrm{Trans}(\cdot)$ for better expressiveness and easier training:

$$\mathrm{Trans}(a) = \mathrm{softmax}(\mathrm{MLP}_2(a)), \tag{4}$$

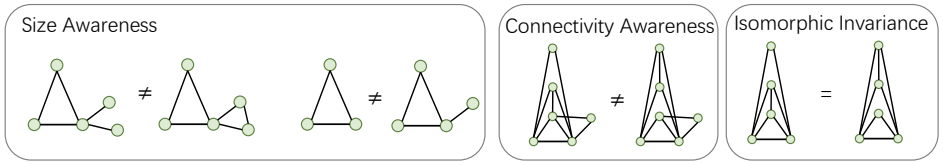


Figure 3: Three properties that a good substructure descriptor function $f(\cdot)$ should exhibit.

where MLP_2 denotes an MLP with ReLU and a channel-wise softmax function $\mathrm{softmax}(\cdot)$ normalizes the outputs of MLP separately on each channel.

As a Plugin to Empower Other Models. In addition to a standalone UnionSNN network, our union-subgraph-based structural coefficients could also be incorporated into other GNNs in a flexible and yet effective manner. For arbitrary MPNNs as in Eq. (1), we can plugin our structural coefficients via an element-wise multiplication:

$$\mathbf{h}_v^{(l)} = \mathrm{AGG}^{(l-1)}(\mathbf{h}_v^{(l-1)}, \mathrm{MSG}^{(l-1)}(\{\mathrm{Trans}^{(l-1)}(\tilde{a}^{vu})\mathbf{h}_u^{(l-1)}, u \in \mathcal{N}(v)\})). \tag{5}$$

For transformer-based models, inspired by the spatial encoding in Graphormer (Ying et al. 2021), we inject our structural coefficients into the attention matrix as a bias term:

$$A_{vu} = \frac{(h_v W_Q)\,(h_u W_K)^T}{\sqrt{d}} + \mathrm{Trans}(\tilde{a}^{vu}), \tag{6}$$

where the definition of $\mathrm{Trans}(\cdot)$ is the same as Eq. (4) and shared across all layers, $h_v, h_u \in \mathbb{R}^{1\times d}$ are the node representations of v and u, $W_Q, W_K \in \mathbb{R}^{d\times d}$ are the parameter matrices, and d is the hidden dimension of h_v and h_u.

Design Comparisons with GraphSNN. Our UnionSNN is similar to GraphSNN in the sense that both improve the expressiveness of MPNNs (and 1-WL) by injecting the information of local substructures. However, UnionSNN is superior to GraphSNN in the following aspects. (1) Union subgraphs in UnionSNN are stronger than overlap subgraphs in GraphSNN, as ensured by Theorem 1. (2) The shortest-path-based substructure descriptor designed in UnionSNN is more powerful than that in GraphSNN: the latter fails to possess the property of Connectivity Awareness (as elaborated in Section 4.1). An example of two non-isomorphic subgraphs $S_{v\cap u}$ and $S_{v'\cap u'}$ is shown in Figure 4. They have the same structural coefficients in GraphSNN. (3) The aggregation function in UnionSNN works on adjacent nodes in the input graph, while that in GraphSNN utilizes the structural coefficients on all pairs of nodes (regardless of their adjacency). Consequently, GraphSNN requires to pad the adjacency matrix and feature matrix of each graph to the maximum graph size, which significantly increases the computational complexity. The advantages of UnionSNN over GraphSNN are also evidenced by experimental results in Section 5.4.

Time and Space Complexities. Given that the path matrices have been precomputed, our time and space complexities for model training are in line with those of GIN and GraphSNN. Denote m as the number of edges in a graph, f and d as the dimensions of input and output feature vectors, and k as the number of layers. Time and space complexities of UnionSNN are $\mathcal{O}(kmfd)$ and $\mathcal{O}(m)$, respectively.

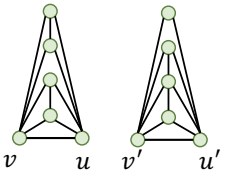


Figure 4: Example of two non-isomorphic subgraphs with the same structural coefficient in GraphSNN

4.3 Expressive Power of UnionSNN

We formalize the following theorem to show that UnionSNN is more powerful than 1-WL test in terms of expressive power.

Theorem 3. *UnionSNN is more expressive than 1-WL in testing non-isomorphic graphs.*

The stronger expressiveness of UnionSNN over 1-WL is credited to its use of union subgraphs, with an effective encoding of local neighborhood connectivities via the shortest-path-based design of structural coefficients.

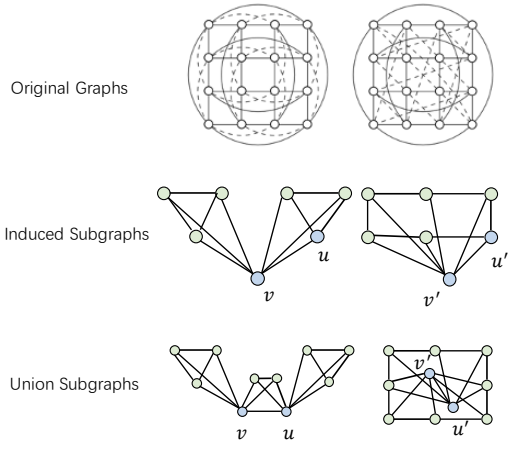


Figure 5: These two graphs can be distinguished by UnionSNN but not by 3-WL.

The Connection with Higher-Order WL Tests. As discussed in (Papp and Wattenhofer 2022), high-order WL tests

Dataset	Graph #	Class #	Avg Node #	Avg Edge #	Metric
MUTAG	188	2	17.93	19.79	Accuracy
PROTEINS	1113	2	39.06	72.82	Accuracy
ENZYMES	600	6	32.63	62.14	Accuracy
DD	1178	2	284.32	715.66	Accuracy
FRANKENSTEIN	4337	2	16.90	17.88	Accuracy
Tox21	8169	2	18.09	18.50	Accuracy
NCI1	4110	2	29.87	32.30	Accuracy
NCI109	4127	2	29.68	32.13	Accuracy
OGBG-MOLHIV	41127	2	25.50	27.50	ROC-AUC
OGBG-MOLBBBP	2039	2	24.06	25.95	ROC-AUC
ZINC10k	12000	-	23.16	49.83	MAE
ZINC-full	249456	-	23.16	49.83	MAE
4CYCLES	20000	2	36.00	61.71	Accuracy
6CYCLES	20000	2	56.00	87.84	Accuracy

Table 1: Statistics of Graph Classification/Regression Datasets.

are concepts of global comparisons over two graphs. Therefore, it is arguable if the WL hierarchy is a suitable tool to measure the expressiveness of GNN extensions as the latter focus on locality. Nonetheless, there exist some graph structures on which the 3-WL test is not stronger than UnionSNN, i.e., some graphs can be distinguished by UnionSNN but not by 3-WL. As an example, UnionSNN can distinguish the 4x4 Rook's graph and the Shrikhande graph (as shown in Figure 5, and cited from (Huang et al. 2022)), while 3-WL cannot, which suggests that UnionSNN is stronger than 3-WL on such graphs. The induced graphs of arbitrary nodes v and v' in the two graphs cannot be distinguished by 1-WL. Thus the original graphs cannot be distinguished by 3-WL. However, the numbers of 3-cycles and 4-cycles in their union subgraphs are different, and UnionSNN is able to reflect the number of 3-cycles and 4-cycles with the consideration of E_3^{vu} and E_4^{vu} edges in union subgraphs. In addition, the range of the union subgraph of an edge is inherently limited to 3-hop neighbors of each end node of the edge. As a result, one should expect that UnionSNN's power is theoretically upper-bounded by 4-WL. Furthermore, UnionSNN has limitations in distinguishing cycles with length larger than 6, as this would require information further than 3 hops.

5 Experimental Study

In this section, we evaluate the effectiveness of our proposed model under various settings and aim to answer the following research questions: **RQ1.** Can UnionSNN outperform existing MPNNs and transformer-based models? **RQ2.** Can other GNNs benefit from our structural coefficient? **RQ3.** How do different components affect the performance of UnionSNN? **RQ4.** Is our runtime competitive with other substructure descriptors? We conduct experiments on four tasks: graph classification, graph regression, node classification and cycle detection. When we use UnionSNN to plugin other models we use the prefix term "Union-", such as UnionGCN. The implementation details of our experiments are available in our arXiv version (Xu et al. 2023).

Datasets. For graph classification, we use 10 benchmark datasets. Eight of them were selected from the TUDataset (Kersting et al. 2016), including MUTAG, PROTEINS, ENZYMES, DD, FRANKENSTEIN (denoted as FRANK in our tables), Tox21, NCI1 and NCI109. The other two datasets OGBG-MOLHIV and OGBG-MOLBBBP were selected from Open Graph Benchmark (Hu et al. 2020). For graph regression, we conduct experiments on ZINC10k and ZINC-full datasets (Dwivedi et al. 2020). For node classification, we test on five datasets, including citation networks (Cora, Citeseer, and PubMed (Sen et al. 2008)) and Amazon co-purchase networks (Computer and Photo (McAuley et al. 2015)). For cycle detection, we conduct experiments on the detection of cycles of lengths 4 and 6, implemented as a graph classification task (Kersting et al. 2016). These datasets cover various graph sizes and densities. Statistics of the datasets used are summarized in Tables 1 and 2.

	Node #	Edge #	Class #	Feature #	Training #
Cora	2708	5429	7	1433	140
Citeseer	3327	4732	6	3703	120
PubMed	19717	44338	3	500	60
Computer	13381	259159	10	7667	1338
Photo	7487	126530	8	745	749

Table 2: Statistics of Node Classification Datasets.

Baseline Models. We select various GNN models as baselines, including (1) classical MPNNs such as GCN (Kipf and Welling 2016), GIN (Xu et al. 2018), GraphSAGE (Hamilton, Ying, and Leskovec 2017), GAT (Veličković et al. 2017), GatedGCN (Bresson and Laurent 2017); (2) WL-based GNNs such as 3WL-GNN (Maron et al. 2019); (3) transformer-based methods such as UGformer (Nguyen, Nguyen, and Phung 2019), Graphormer (Ying et al. 2021) and GPS (Rampášek et al. 2022); (4) state-of-the-art graph pooling methods such as MEWISPool (Nouranizadeh et al. 2021); (5) methods that introduce structural information by shortest paths or curvature, such as GeniePath (Liu et al. 2019), CurvGN (Ye et al. 2019), and NestedGIN (Zhang and Li 2021); (6) GNN with subgraph aggregation method, such as DropGIN (Papp et al. 2021); (7) GNNs with positional encoding, such as GatedGCN-LSPE (Dwivedi et al. 2021); (8) GraphSNN (Wijesinghe and Wang 2021).

	MUTAG	PROTEINS	ENZYMES	DD	FRANK	Tox21	NCI1	NCI109
GAT	77.56 ± 10.49	74.34 ± 2.09	67.67 ± 3.74	74.25 ± 3.76	62.85 ± 1.90	90.35 ± 0.71	78.07 ± 1.94	74.34 ± 2.18
3WL-GNN	84.06 ± 6.62	60.18 ± 6.35	54.17 ± 6.25	74.84 ± 2.63	58.68 ± 1.93	90.31 ± 1.33	78.39 ± 1.54	77.97 ± 2.22
UGformer	75.66 ± 8.67	70.17 ± 5.42	64.57 ± 4.53	75.51 ± 3.52	56.13 ± 2.51	88.06 ± 0.50	68.84 ± 1.54	66.37 ± 2.74
MEWISPool	84.73 ± 4.73	68.10 ± 3.97	53.66 ± 6.07	76.03 ± 2.59	64.63 ± 2.83	88.13 ± 0.05	74.21 ± 3.26	75.30 ± 1.45
CurvGN	87.25 ± 6.28	75.73 ± 2.87	56.50 ± 7.13	72.16 ± 1.88	61.89 ± 2.41	90.87 ± 0.38	79.32 ± 1.65	77.30 ± 1.78
NestedGIN	86.23 ± 8.82	68.55 ± 3.22	54.67 ± 9.99	70.04 ± 4.32	67.07 ± 1.46	91.42 ± 1.18	82.04 ± 2.23	79.94 ± 1.59
DropGIN	84.09 ± 8.42	73.39 ± 4.90	67.50 ± 6.42	71.05 ± 2.68	66.91 ± 2.16	91.66 ± 0.92	82.19 ± 1.39	81.13 ± 0.43
GatedGCN-LSPE	88.33 ± 3.88	73.94 ± 2.72	64.50 ± 5.92	76.74 ± 2.04	67.74 ± 2.65	91.71 ± 0.71	80.75 ± 1.67	80.13 ± 2.33
GraphSNN	84.04 ± 4.09	71.78 ± 4.11	67.67 ± 3.74	76.03 ± 2.59	67.17 ± 2.25	**92.24** ± 0.59	70.87 ± 2.78	70.11 ± 1.86
GCN	77.13 ± 5.24	73.89 ± 2.85	64.33 ± 5.83	72.16 ± 2.83	58.80 ± 1.06	90.10 ± 0.77	79.73 ± 0.95	75.91 ± 1.53
UnionGCN (ours)	81.87 ± 3.81	75.02 ± 2.50	64.67 ± 7.14	69.69 ± 4.18	61.72 ± 1.76	91.63 ± 0.72	80.41 ± 1.94	79.50 ± 1.82
GatedGCN	77.11 ± 10.05	76.18 ± 3.12	66.83 ± 5.08	72.58 ± 3.04	61.40 ± 1.92	90.83 ± 0.96	80.32 ± 2.07	78.19 ± 2.39
UnionGatedGCN(ours)	77.14 ± 8.14	**76.91** ± 3.06	67.83 ± 6.87	72.50 ± 2.22	61.47 ± 2.54	91.31 ± 0.89	80.95 ± 2.11	78.21 ± 2.58
GraphSAGE	80.38 ± 10.98	74.87 ± 3.38	52.50 ± 5.69	73.10 ± 4.34	52.95 ± 4.01	88.36 ± 0.15	63.94 ± 2.52	65.46 ± 1.12
UnionGraphSAGE(ours)	83.04 ± 8.70	74.57 ± 2.35	58.32 ± 2.64	73.85 ± 4.46	56.75 ± 3.85	88.59 ± 0.12	69.36 ± 1.64	69.87 ± 1.04
GIN	86.23 ± 8.17	72.86 ± 4.14	65.83 ± 5.93	70.29 ± 2.96	66.50 ± 2.37	91.74 ± 0.95	82.29 ± 1.77	80.95 ± 1.87
UnionGIN (ours)	**88.86** ± 4.33	73.22 ± 3.90	67.83 ± 6.10	70.47 ± 4.98	**68.02** ± 1.47	91.74 ± 0.74	82.29 ± 1.98	**82.24** ± 1.24
UnionSNN (ours)	87.31 ± 5.29	75.02 ± 2.50	**68.17** ± 5.70	**77.00** ± 2.37	67.83 ± 1.99	91.76 ± 0.85	**82.34** ± 1.93	81.61 ± 1.78

Table 3: Graph classification results (average accuracy ± standard deviation) over 10-fold-CV. The first and second best results on each dataset are highlighted in bold and underlined. The winner between a base model with and without our structural coefficient injected is highlighted in gray background. The same applies to all tables.

model	OGBG-MOLHIV	OGBG-MOLBBBP
GraphSAGE	70.37 ± 0.42	60.78 ± 2.43
GCN	73.49 ± 1.99	64.04 ± 0.43
GIN	70.60 ± 2.56	64.10 ± 1.05
GAT	70.60 ± 1.78	63.30 ± 0.53
CurvGN	73.17 ± 0.89	66.51 ± 0.80
GraphSNN	73.05 ± 0.40	62.84 ± 0.36
UnionSNN (ours)	**74.44** ± 1.21	**68.28** ± 1.47

Table 4: Graph classification results (average ROC-AUC ± standard deviation) on OGBG-MOLHIV and OGBG-MOLBBBP datasets. The best result is highlighted in bold.

	ZINC10k	ZINC-full
GCN	0.3800 ± 0.0171	0.1152 ± 0.0010
UnionGCN (ours)	0.2811 ± 0.0050	0.0877 ± 0.0003
GIN	0.5260 ± 0.0365	0.1552 ± 0.0079
UnionGIN (ours)	0.4625 ± 0.0222	0.1334 ± 0.0013
GraphSAGE	0.3980 ± 0.0290	0.1205 ± 0.0034
UnionSAGE (ours)	0.3768 ± 0.0041	0.1146 ± 0.0017
Graphormer	0.1269 ± 0.0083	0.0309 ± 0.0031
Unionormer (ours)	0.1241 ± 0.0056	0.0252 ± 0.0026
GPS	0.0740 ± 0.0022	0.0262 ± 0.0025
UnionGPS (ours)	**0.0681** ± 0.0013	**0.0236** ± 0.0017

Table 5: Graph regression results (average test MAE ± standard deviation) on ZINC10k and ZINC-full datasets.

Model	4-CYCLES	6-CYCLES
GraphSAGE	50.00	50.00
CGN	84.33	63.56
GraphSNN	67.10	66.04
GCN	50.00	50.00
UnionGCN (ours)	94.31	**95.59**
GIN	96.61	90.06
UnionGIN (ours)	97.15	92.63
UnionSNN (ours)	**99.11**	95.00

Table 6: Cycle detection results (average accuracy ± standard deviation).

5.1 Performance on Different Graph Tasks

Graph-Level Tasks. For graph classification, we report the results on 8 TUDatasets in Table 3 and the results on 2 OGB datasets in Table 4. Our UnionSNN outperforms all baselines in 7 out of 10 datasets (by comparing UnionSNN with all baselines without "ours"). We further apply our structural coefficient as a plugin component to four MPNNs: GCN, GatedGCN, GraphSAGE and GIN. The results show that our structural coefficient is able to boost the performance of the base model in almost all cases, with an improvement of up to 11.09% (Achieved when plugging in our local encoding into GraphSAGE on the ENZYMES dataset: (58.32% - 52.50%) / 52.50% = 11.09%.). For graph regression, we report the mean absolute error (MAE) on ZINC10k and ZINC-full. Table 5 shows the performance of MPNNs with our structural coefficient (UnionGCN, UnionGIN, UnionGatedGCN and UnionGraphSAGE) dramatically beat their counterparts. Additionally, when injecting our structural coefficient into Transformer-based models, Unionormer and UnionGPS further improve Graphormer and GPS.

Detecting cycles requires more than 1-WL expressivity (Morris et al. 2019). To further demonstrate the effectiveness of UnionSNN, we conduct experiments on the detection of cycles of lengths 4 and 6. Table 6 shows the superiority of UnionSNN over 5 baselines for detecting cycles. Note that GCN makes a random guess on the cycle detection with an accuracy of 50%. By incorporating our structural coefficient to GCN, we are able to remarkably boost the accuracy to around 95%. The large gap proves that our structural coefficient is highly effective in capturing local information.

Node-Level Tasks. We report the results of node classification in Table 7. UnionSNN outperforms all baselines on all 5

	Cora	Citeseer	PubMed	Computer	Photo
GraphSAGE	70.60 ± 0.64	55.02 ± 3.40	70.36 ± 4.29	80.30 ± 1.30	89.16 ± 1.03
GAT	74.82 ± 1.95	63.82 ± 2.81	74.02 ± 1.11	85.94 ± 2.35	91.86 ± 0.47
GeniePath	72.16 ± 2.69	57.40 ± 2.16	70.96 ± 2.06	82.68 ± 0.45	89.98 ± 1.14
CurvGN	74.06 ± 1.54	62.08 ± 0.85	74.54 ± 1.61	86.30 ± 0.70	92.50 ± 0.50
GCN	72.56 ± 4.41	58.30 ± 6.32	74.44 ± 0.71	84.58 ± 3.02	91.71 ± 0.55
UnionGCN (ours)	74.48 ± 0.42	59.02 ± 3.64	74.82 ± 1.10	**88.84** ± 0.27	92.33 ± 0.53
GIN	75.86 ± 1.09	63.10 ± 2.24	76.62 ± 0.64	86.26 ± 0.56	92.11 ± 0.32
UnionGIN (ours)	75.90 ± 0.80	63.66 ± 1.75	76.78 ± 1.02	86.81 ± 2.12	92.28 ± 0.19
GraphSNN	75.44 ± 0.73	64.68 ± 2.72	76.76 ± 0.54	84.11 ± 0.57	90.82 ± 0.30
UnionGraphSNN (ours)	75.58 ± 0.49	**65.22** ± 1.12	76.92 ± 0.56	84.58 ± 0.46	90.60 ± 0.58
UnionSNN (ours)	**76.86** ± 1.58	65.02 ± 1.02	**77.06** ± 1.07	87.76 ± 0.36	**92.92** ± 0.38

Table 7: Node classification results (average accuracy ± standard deviation) over 10 runs.

	MUTAG	PROTEINS	ENZYMES	DD	NCI1	NCI109
overlap	85.70 ± 7.40	71.33 ± 5.35	65.00 ± 5.63	73.43 ± 4.07	73.58 ± 1.73	72.96 ± 2.01
minus	**87.31** ± 5.29	68.70 ± 3.61	65.33 ± 4.58	74.79 ± 4.63	80.66 ± 1.90	78.70 ± 2.48
union	**87.31** ± 5.29	**75.02** ± 2.50	**68.17** ± 5.70	**77.00** ± 2.37	**82.34** ± 1.93	**81.61** ± 1.78

Table 8: Ablation study on local substructure.

	MUTAG	PROTEINS	ENZYMES	DD	NCI1	NCI109
BetSNN	80.94 ± 6.60	69.44 ± 6.15	65.00 ± 5.63	70.20 ± 5.15	74.91 ± 2.48	73.70 ± 1.87
CountSNN	84.65 ± 6.76	70.79 ± 5.07	66.50 ± 6.77	74.36 ± 7.21	81.74 ± 2.35	79.80 ± 1.67
CurvSNN	85.15 ± 7.35	72.77 ± 4.42	67.17 ± 6.54	75.88 ± 3.24	81.34 ± 2.27	80.64 ± 1.85
LapSNN	**89.39** ± 5.24	68.32 ± 3.49	66.17 ± 4.15	76.31 ± 2.85	81.39 ± 2.08	81.34 ± 2.93
UnionSNN	87.31 ± 5.29	**75.02** ± 2.50	**68.17** ± 5.70	**77.00** ± 2.37	**82.34** ± 1.93	**81.61** ± 1.78

Table 9: Ablation study on substructure descriptor.

	MUTAG	PROTEINS	ENZYMES	DD	NCI1	NCI109
matrix sum	**88.89** ± 7.19	71.32 ± 5.48	65.17 ± 6.43	70.71 ± 4.07	80.37 ± 2.73	79.84 ± 1.89
eigen max	86.73 ± 5.84	71.78 ± 3.24	67.67 ± 6.88	74.29 ± 3.26	81.37 ± 2.08	79.23 ± 2.01
svd sum	87.31 ± 5.29	**75.02** ± 2.50	**68.17** ± 5.70	**77.00** ± 2.37	**82.34** ± 1.93	**81.61** ± 1.78

Table 10: Ablation study on path matrix encoding method.

datasets. Again, injecting our structural coefficients to GCN, GIN, and GraphSNN achieves performance improvement over base models in almost all cases. Remarkably, we find that integrating our structural coefficients into base models can effectively reduce the standard deviations (stds) of the classification accuracy of these models in most cases, and some are with a large margin. For instance, the std of classification accuracy of GCN on Cora is reduced from 4.41 to 0.42 after injecting our structural coefficients.

5.2 Ablation Study

In this subsection, we validate empirically the design choices made in different components of our model: (1) the local substructure; (2) the substructure descriptor; (3) the encoding method from a path matrix to a scalar. All experiments were conducted on 6 graph classification datasets.

Local Substructure. We test three types of local substructures defined in Section 3.2: overlap subgraphs, union minus subgraphs and union subgraphs. They are denoted as "overlap", "minus", and "union" respectively in Table 8. The best results are consistently achieved by using union subgraphs.

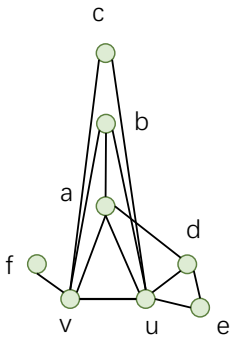
Substructure Descriptor. We compare our substructure descriptor with four existing ones discussed in Section 4.1. We replace the substructure descriptor in UnionSNN with edge betweenness, node/edge counting, Ricci curvature, and Laplacian matrix (other components unchanged), and obtain four variants, namely BetSNN, CountSNN, CurvSNN, and LapSNN. Table 9 shows our UnionSNN is a clear winner: it achieves the best result on 5 out of 6 datasets. This experiment demonstrates that our path matrix better captures structural information.

Path Matrix Encoding Method. We test three methods that transform a path matrix to a scalar: (1) sum of all elements in the path matrix (matrix sum); (2) maximum eigenvalue of the path matrix (eigen max); (3) sum of all singular values of the matrix (svd sum) used by UnionSNN in Section 4.2. Table 10 shows that the encoding method "svd sum" performs the best on 5 out of 6 datasets.

5.3 Case Study

In this subsection, we investigate how the proposed structural coefficient a^{vu} reflects local connectivities. We work on an example union subgraph $S_{v \cup u}$ in Figure 6 and modify its nodes/edges to study how the coefficient a^{vu} varies with

the local structural change. We have the following observations: (1) with the set of nodes unchanged, deleting an edge increases a^{vu}; (2) deleting a node (and its incident edges) decreases a^{vu}; (3) the four types of edges in the closed neighborhood (Section 3.2) have different effects to a^{vu}: $E_1^{vu} < E_2^{vu} < E_3^{vu} < E_4^{vu}$ (by comparing -ab, -ad, -de, and +df). These observations indicate that a smaller coefficient will be assigned to an edge with a denser local substructure. This matches our expectation that the coefficient should be small for an edge in a highly connected neighborhood. The rationale is, such edges are less important in message passing as the information between their two incident nodes can flow through more paths. By using the coefficients that well capture local connectivities, the messages from different neighbors could be properly adjusted when passing to the center node. This also explains the effectiveness of UnionSNN in performance experiments.



Original a^{vu}	23.20

Modify node	a^{vu}
-a	20.60
-b	20.02
-c	19.51
-d	19.54
-e	18.91
-f	17.50

Modify edge	a^{vu}
-ab	23.66
+bc	22.71
-ab+bc	23.16
-ad	23.70
-de	23.81
+df	21.30
-vc	24.35
-uc	24.84
-vb	24.28

Figure 6: Structural coefficient analysis.

5.4 Efficiency Analysis

In this subsection, we conduct experiments on PROTEINS, DD and FRANKENSTEIN datasets, which cover various number of graphs and graph sizes.

Preprocessing Computational Cost. UnionSNN computes structural coefficients in preprocessing. We compare its preprocessing time with the time needed in baseline models for pre-computing their substructure descriptors, including edge betweenness (Betweenness) in BetSNN, node/edge counting (Count_ne) in GraphSNN, Ricci curvature (Curvature) in CurvGN, and counting cycles (Count_cycle) in (Bodnar et al. 2021). As shown in Table 11, the preprocessing time of UnionSNN is comparable to that of other models. This demonstrates that our proposed structural coefficient is able to improve performance without significantly sacrificing efficiency. Our computational cost could be further reduced by pre-computing in the input graph all node pairs with a distance of 1, 2, or 3 (all possible distances in our union subgraphs). The local path matrix could be computed efficiently by simple checking. This can avoid repeated shortest path computations in each local neighborhood.

Runtime Computational Cost We conduct an experiment to compare the total runtime cost of UnionSNN with those in other MPNNs. The results are reported in Table 12. Although UnionSNN runs slightly slower than GCN and GIN, it runs over 4.56 times faster than WL-based MPNN (3WL-GNN) and is comparable to MPNN with positional encoding (GatedGCN-LSPE). Compared with GraphSNN, UnionSNN runs significantly faster: the efficiency improvement approaches an order of magnitude on datasets with large graphs, e.g., DD. It is because UnionSNN does not need to pad the adjacency matrix and feature matrix of each graph to the maximum graph size in the dataset, as GraphSNN does.

	PROTEINS	DD	FRANK
Betweenness	51	749	47
Count_ne	44	669	41
Curvature	304	1295	1442
Count_cycle	2286	-	86385
Ours	73	1226	59

Table 11: Time cost (seconds) for computing structural coefficients in preprocessing. We are not reporting the time of counting cycles on DD since it took more than 100 hours.

	PROTEINS	DD	FRANK
GCN	0.67	1.51	1.65
GIN	0.53	1.81	2.01
3WL-GNN	6.06	75.25	19.31
GatedGCN-LSPE	1.33	3.65	2.55
GraphSNN	4.05	30.45	5.76
UnionSNN	1.31	3.61	2.46

Table 12: Time cost (hours) for a single run with 10-fold-CV, including training, validation, test (excluding preprocessing).

6 Conclusions

We present UnionSNN, a model that outperforms 1-WL in distinguishing non-isomorphic graphs. UnionSNN utilizes an effective shortest-path-based substructure descriptor applied to union subgraphs, making it more powerful than previous models. Our experiments demonstrate that UnionSNN surpasses state-of-the-art baselines in both graph-level and node-level classification tasks while maintaining its computational efficiency. The use of union subgraphs enhances the model's ability to capture neighbor connectivities and facilitates message passing. Additionally, when applied to existing MPNNs and Transformer-based models, UnionSNN improves their performance by up to 11.09%.

Acknowledgments

This research/project is supported by the National Research Foundation, Singapore under its Industry Alignment Fund – Pre-positioning (IAF-PP) Funding Initiative, and the Ministry of Education, Singapore under its MOE Academic Research Fund Tier 2 (STEM RIE2025 Award MOE-T2EP20220-0006). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore, and the Ministry of Education, Singapore.

References

Abboud, R.; Dimitrov, R.; and Ceylan, İ. İ. 2022. Shortest Path Networks for Graph Property Prediction. *arXiv preprint arXiv:2206.01003*.

Ahmedt-Aristizabal, D.; Armin, M. A.; Denman, S.; Fookes, C.; and Petersson, L. 2021. Graph-based deep learning for medical diagnosis and analysis: past, present and future. *Sensors*, 21(14): 4758.

Bodnar, C.; Frasca, F.; Wang, Y.; Otter, N.; Montufar, G. F.; Lio, P.; and Bronstein, M. 2021. Weisfeiler and lehman go topological: Message passing simplicial networks. In *International Conference on Machine Learning*, 1026–1037. PMLR.

Bouritsas, G.; Frasca, F.; Zafeiriou, S. P.; and Bronstein, M. 2022. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Brandes, U. 2001. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2): 163–177.

Bresson, X.; and Laurent, T. 2017. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*.

Dai, H.; Dai, B.; and Song, L. 2016. Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning*, 2702–2711. PMLR.

Derrow-Pinion, A.; She, J.; Wong, D.; Lange, O.; Hester, T.; Perez, L.; Nunkesser, M.; Lee, S.; Guo, X.; Wiltshire, B.; et al. 2021. Eta prediction with graph neural networks in google maps. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 3767–3776.

Duvenaud, D. K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; and Adams, R. P. 2015. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28.

Dwivedi, V. P.; and Bresson, X. 2020. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*.

Dwivedi, V. P.; Joshi, C. K.; Luu, A. T.; Laurent, T.; Bengio, Y.; and Bresson, X. 2020. Benchmarking Graph Neural Networks. *arXiv preprint arXiv:2003.00982*.

Dwivedi, V. P.; Luu, A. T.; Laurent, T.; Bengio, Y.; and Bresson, X. 2021. Graph neural networks with learnable structural and positional representations. *arXiv preprint arXiv:2110.07875*.

Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, E.; Tang, J.; and Yin, D. 2019. Graph neural networks for social recommendation. In *The world wide web conference*, 417–426.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Horn, M.; De Brouwer, E.; Moor, M.; Moreau, Y.; Rieck, B.; and Borgwardt, K. 2021. Topological graph neural networks. *arXiv preprint arXiv:2102.07835*.

Horn, R. A.; and Johnson, C. R. 2012. *Matrix analysis*. Cambridge university press.

Hu, W.; Fey, M.; Zitnik, M.; Dong, Y.; Ren, H.; Liu, B.; Catasta, M.; and Leskovec, J. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33: 22118–22133.

Huang, Y.; Peng, X.; Ma, J.; and Zhang, M. 2022. Boosting the Cycle Counting Power of Graph Neural Networks with I^2-GNNs. *arXiv preprint arXiv:2210.13978*.

Kersting, K.; Kriege, N. M.; Morris, C.; Mutzel, P.; and Neumann, M. 2016. Benchmark Data Sets for Graph Kernels.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kreuzer, D.; Beaini, D.; Hamilton, W.; Létourneau, V.; and Tossou, P. 2021. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34: 21618–21629.

Li, P.; Wang, Y.; Wang, H.; and Leskovec, J. 2020. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 33: 4465–4478.

Lim, D.; Robinson, J.; Zhao, L.; Smidt, T.; Sra, S.; Maron, H.; and Jegelka, S. 2022. Sign and basis invariant networks for spectral graph representation learning. *arXiv preprint arXiv:2202.13013*.

Lin, Y.; Lu, L.; and Yau, S.-T. 2011. Ricci curvature of graphs. *Tohoku Mathematical Journal, Second Series*, 63(4): 605–627.

Liu, Z.; Chen, C.; Li, L.; Zhou, J.; Li, X.; Song, L.; and Qi, Y. 2019. Geniepath: Graph neural networks with adaptive receptive paths. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 4424–4431.

Maron, H.; Ben-Hamu, H.; Serviansky, H.; and Lipman, Y. 2019. Provably powerful graph networks. *Advances in neural information processing systems*, 32.

Masters, D.; Dean, J.; Klaser, K.; Li, Z.; Maddrell-Mander, S.; Sanders, A.; Helal, H.; Beker, D.; Rampášek, L.; and Beaini, D. 2022. GPS++: An Optimised Hybrid MPNN/Transformer for Molecular Property Prediction. *arXiv preprint arXiv:2212.02229*.

McAuley, J.; Targett, C.; Shi, Q.; and Van Den Hengel, A. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 43–52.

Mialon, G.; Chen, D.; Selosse, M.; and Mairal, J. 2021. Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667*.

Morris, C.; Ritzert, M.; Fey, M.; Hamilton, W. L.; Lenssen, J. E.; Rattan, G.; and Grohe, M. 2019. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 4602–4609.

Nguyen, D. Q.; Nguyen, T. D.; and Phung, D. 2019. Universal Graph Transformer Self-Attention Networks. *arXiv preprint arXiv:1909.11855*.

Nouranizadeh, A.; Matinkia, M.; Rahmati, M.; and Safabakhsh, R. 2021. Maximum Entropy Weighted Independent Set Pooling for Graph Neural Networks. *arXiv preprint arXiv:2107.01410*.

Ollivier, Y. 2009. Ricci curvature of Markov chains on metric spaces. *Journal of Functional Analysis*, 256(3): 810–864.

Papp, P. A.; Martinkus, K.; Faber, L.; and Wattenhofer, R. 2021. Dropgnn: random dropouts increase the expressiveness of graph neural networks. *Advances in Neural Information Processing Systems*, 34: 21997–22009.

Papp, P. A.; and Wattenhofer, R. 2022. A Theoretical Comparison of Graph Neural Network Extensions. *arXiv preprint arXiv:2201.12884*.

Peng, H.; Wang, H.; Du, B.; Bhuiyan, M. Z. A.; Ma, H.; Liu, J.; Wang, L.; Yang, Z.; Du, L.; Wang, S.; et al. 2020. Spatial temporal incidence dynamic graph neural networks for traffic flow forecasting. *Information Sciences*, 521: 277–290.

Rampášek, L.; Galkin, M.; Dwivedi, V. P.; Luu, A. T.; Wolf, G.; and Beaini, D. 2022. Recipe for a General, Powerful, Scalable Graph Transformer. *arXiv preprint arXiv:2205.12454*.

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

Shi, Y.; Huang, Z.; Feng, S.; Zhong, H.; Wang, W.; and Sun, Y. 2020. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*.

Shiao, W.; Guo, Z.; Zhao, T.; Papalexakis, E. E.; Liu, Y.; and Shah, N. 2022. Link Prediction with Non-Contrastive Learning. *arXiv preprint arXiv:2211.14394*.

Suresh, S.; Shrivastava, M.; Mukherjee, A.; Neville, J.; and Li, P. 2023. Expressive and Efficient Representation Learning for Ranking Links in Temporal Graphs. In *Proceedings of the ACM Web Conference 2023*, 567–577.

Tang, H.; Huang, Z.; Gu, J.; Lu, B.-L.; and Su, H. 2020. Towards scale-invariant graph-related problem solving by iterative homogeneous gnns. *Advances in Neural Information Processing Systems*, 33: 15811–15822.

Thiede, E.; Zhou, W.; and Kondor, R. 2021. Autobahn: Automorphism-based graph neural nets. *Advances in Neural Information Processing Systems*, 34: 29922–29934.

Topping, J.; Di Giovanni, F.; Chamberlain, B. P.; Dong, X.; and Bronstein, M. M. 2021. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wan, C.; Zhang, M.; Hao, W.; Cao, S.; Li, P.; and Zhang, C. 2021. Principled hyperedge prediction with structural spectral features and neural networks. *arXiv preprint arXiv:2106.04292*.

Wang, J.; Ma, A.; Chang, Y.; Gong, J.; Jiang, Y.; Qi, R.; Wang, C.; Fu, H.; Ma, Q.; and Xu, D. 2021. scGNN is a novel graph neural network framework for single-cell RNA-Seq analyses. *Nature communications*, 12(1): 1882.

Weisfeiler, B.; and Leman, A. 1968. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series*, 2(9): 12–16.

Wijesinghe, A.; and Wang, Q. 2021. A New Perspective on" How Graph Neural Networks Go Beyond Weisfeiler-Lehman?". In *International Conference on Learning Representations*.

Wu, Z.; Jain, P.; Wright, M.; Mirhoseini, A.; Gonzalez, J. E.; and Stoica, I. 2021. Representing long-range context for graph neural networks with global attention. *Advances in Neural Information Processing Systems*, 34: 13266–13279.

Xu, J.; Zhang, A.; Bian, Q.; Dwivedi, V. P.; and Ke, Y. 2023. Union Subgraph Neural Networks. *arXiv preprint arXiv:2305.15747*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Ye, Z.; Liu, K. S.; Ma, T.; Gao, J.; and Chen, C. 2019. Curvature graph network. In *International Conference on Learning Representations*.

Ying, C.; Cai, T.; Luo, S.; Zheng, S.; Ke, G.; He, D.; Shen, Y.; and Liu, T.-Y. 2021. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34: 28877–28888.

Ying, Z.; You, J.; Morris, C.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31.

Zhang, B.; Luo, S.; Wang, L.; and He, D. 2023. Rethinking the expressive power of gnns via graph biconnectivity. *arXiv preprint arXiv:2301.09505*.

Zhang, M.; and Chen, Y. 2018. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31.

Zhang, M.; and Li, P. 2021. Nested graph neural networks. *Advances in Neural Information Processing Systems*, 34: 15734–15747.

Zhao, L.; Jin, W.; Akoglu, L.; and Shah, N. 2021. From stars to subgraphs: Uplifting any GNN with local structure awareness. *arXiv preprint arXiv:2110.03753*.