UNION SUBGRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Neural Networks (GNNs) are widely used for graph representation learning in many application domains. The expressiveness of GNNs is upper-bounded by 1-dimensional Weisfeiler-Lehman (1-WL) test as they operate on rooted subtrees in message passing. In this paper, we empower GNNs by injecting neighbor-connectivity information extracted from a new type of substructures. 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. Our extensive experiments on both graph-level and node-level tasks demonstrate that UnionSNN outperforms state-of-the-art baseline models, with competitive computational efficiency.

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 et al., 2016), social sciences (Ying et al., 2018; Fan et al., 2019), and brain networks (Ahmedt-Aristizabal et al., 2021). GNNs also attained promising results on the tasks of graph classification (Xu et al., 2018; Ying et al., 2018), node classification (Kipf & Welling, 2016; Hamilton et al., 2017), link prediction (Zhang & Chen, 2018), etc.

Recently, a limitation in the expressiveness of GNNs has been identified. Xu et al. (2018) shows that GNNs are at most as powerful as 1-dimentional Weisfeiler-Lehman (1-WL) test (Weisfeiler & 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 local substructure information, such as induced subgraph (Zhao et al., 2021), overlap subgraph (Wijesinghe & Wang, 2021) and spatial encoding (Bouritsas et al., 2022), to enhance GNNs' expressiveness. But the local structure 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), affinity matrix (Wan et al., 2021), etc., to control the message from different distance of neighbors. However, the substructure descriptor used to encode the substructure may overlook some connectivities between neighbors. Furthermore, some of above models would 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. (3) We propose a new model, namely Union Subgraph Neural Network (UnionSNN), which is strictly more expressive than 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 complete neighborhood.
- We abstract three desired 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 injects the information extracted from union subgraphs in message passing. We theoretically prove that UnionSNN is more expressive than 1-WL. We also show that UnionSNN at least stronger than 3-WL on some cases.
- We perform extensive experiments on both graph-level and node-level tasks. The results justify the superiority of UnionSNN over state-of-the-art baselines in terms of both effectiveness and efficiency.

2 Related Work

2.1 Substructure-enhanced GNNs

In recent years, several novel GNN architectures have been designed to enhance their expressiveness by encoding local substructures. GraphSNN (Wijesinghe & Wang, 2021) brings the information of overlap subgraphs into the massage passing scheme as a structural coefficient. However, the overlap subgraph and the substructure descriptor used by GraphSNN are is not powerful enough to distinguish all non-isomorphic substructures in the 1-hop neighborhood. Zhao et al. (2021) encode the induced subgraph for each node and inject it into node representations. Graph Substructure Network (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. Other transformer-based methods incorporate local structural information via positional encoding. Graphormer (Ying et al., 2021) combines the node degree and shortest path information for spatial encoding. Dwivedi et al. (2021) introduce the random-walk matrix and Graph Laplacian matrix to add the canonical positional information into nodes. 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 other shortest-path-based techniques to GNNs. Li et al. (2020) presents a distance encoding 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 et al., 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. (2021) propose Balanced Forman curvature that better reflects the edges having bottleneck effects, and alleviates 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 in the message passing of MPNNs.

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 Massage Passing Neural Networks

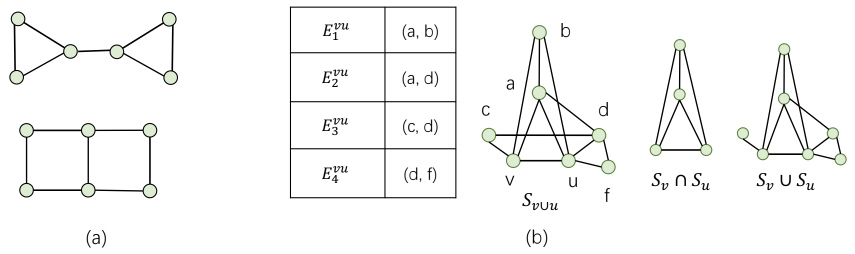
We represent a graph as $G = (V, E, X)$, where $V = \{v_1, ..., v_n\}$ is the set of nodes, $E \in V \times V$ is the set of edges, and $X = \{\mathbf{x}_v \mid v \in V\}$ is the set of node features. The set of neighbors of node v is denoted by $\mathcal{N}(v) = \{u \in V \mid (v, u) \in E\}$. As defined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

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

where $h_v^{(l)}$ is the representation of node v at the l-th layer, $\mathbf{h}_v^{(0)} = \mathbf{x}_v$, $\text{AGG}(\cdot)$ and $\text{MSG}(\cdot)$ 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 could not be distinguished by MPNN (and also 1-WL). Such an example is shown in Figure 1(a). A simple fix to this 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?**



E_1^{vu}	(a, b)
E_2^{vu}	(a, d)
E_3^{vu}	(c, d)
E_4^{vu}	(d, f)

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.

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 & 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 and 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 the union subgraph captures the whole picture of the neighborhood of two adjacent nodes. It covers all kinds of connectivities in the neighborhood, which serves as a perfect local substructure for improving the expressiveness of MPNNs. Note that we restrict the discussion to the 1-hop neighborhood because we aim to develop a model based on the MPNN architecture, in which a single layer of convolution is performed on the 1-hop neighbors.

3.3 Union Isomorphism

In this subsection, we define the isomorphic relationship between the neighborhoods of two nodes based on the concept of union subgraphs. The definition follows that of overlap isomorphism in Wijesinghe & 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: $\tilde{\mathcal{N}}(i) \to \tilde{\mathcal{N}}(j)$ 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: $\tilde{\mathcal{N}}(i) \to \tilde{\mathcal{N}}(j)$ 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).

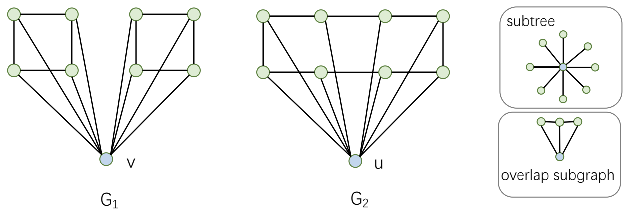


Figure 2: G_1 and G_2 are overlap-isomorphic (have the same overlap subgraphs), but not union-isomorphic. Any union subgraph containing v are G_1 itself. Same for u. The detailed discussion is provided in Appendix A.

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 proofs of all theorems in this paper are provided in Appendix B. Figure 2 shows a pair of non-isomorphic graphs that are distinguishable under union-isomorphism but not overlap-isomorphism or 1-WL (subtree). The union subgraph of v and one of its arbitrary neighbors is G_1 itself (same for node u). Since G_1 and G_2 are not isomorphic, they are not union-isomorphic either.

4 UnionSNN

In this section, we first discuss how to design our substructure descriptor so that it well captures the structural information in union subgraphs with several desired properties. We then present our model UnionSNN, which effectively incorporates the information encoded by the substructure descriptor to MPNNs. Finally, we show that UnionSNN has a stronger expressiveness than 1-WL and is superior to GraphSNN in its design.

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.

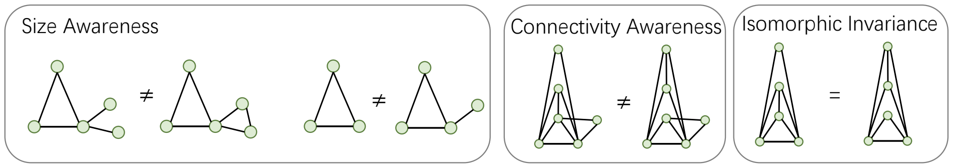


Figure 3: Three properties that a good substructure descriptor function $f(\cdot)$ should exhibit.

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:

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

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 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 & Wang (2021) put 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 et al., 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. We detail the definitions of these substructure descriptor functions in Appendix C.

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 choose to perform Singular Value Decomposition (SVD) (Horn & Johnson, 2012) on the path matrix and extract the singular values:

$$\boldsymbol{P} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*. \tag{3}$$

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 an arbitrary order of nodes, the computed a_{vu} remains the same, which removes the condition required by Theorem 2.

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:

$$\boldsymbol{h}_v^{(l)} = \mathrm{MLP}_1{}^{(l-1)}\left(\left(1+\epsilon^{(l-1)}\right)\boldsymbol{h}_v^{(l-1)} + \sum_{u\in\mathcal{N}(v)} \mathrm{Trans}^{(l-1)}(\tilde{a}^{vu})\boldsymbol{h}_u^{(l-1)}\right), \tag{4}$$

where $\epsilon^{(l-1)}$ is a learnable scalar parameter and $\tilde{a}^{vu} = \frac{a^{vu}}{\sum_{u\in\mathcal{N}(u)} a^{vu}}$. $\mathrm{MLP}_1(\cdot)$ 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:

$$\mathrm{Trans}(a) = \mathrm{softmax}(\mathrm{MLP}_2(a)), \tag{5}$$

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.

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. We further provide a special case to show some graphs can be distinguished by UnionSNN but not by 3-WL or GraphSNN in Appendix D.

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.

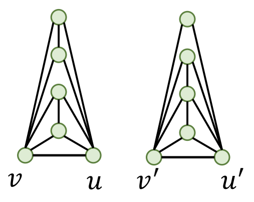


Figure 4: Example of two non-isomorphic subgraphs with the same structural coefficient in GraphSNN

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? **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 three tasks: graph classification, graph regression and node classification.

Datasets. For graph classification, we use twelve 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)). These datasets cover various graph sizes and densities. The statistics of datasets are summarized in Appendix E.

Baseline Models. We select various GNN models as baselines, including (1) classical MPNNs such as GCN (Kipf & Welling, 2016), GIN (Xu et al., 2018), GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2017); (2) WL-based GNNs such as 3WL-GNN (Maron et al., 2019); (3) transformer-based methods such as UGformer (Nguyen et al., 2019); (4) state-of-the-art graph pooling methods such as MEWISPool (Nouranizadeh et al., 2021); (5) methods introducing structural information by shortest paths or curvature, such as GeniePath (Liu et al., 2019), CurvGN (Ye et al., 2019) and NestedGIN (Zhang & Li, 2021); (6) GNNs with positional encoding, such as GatedGCN-LSPE (Dwivedi et al., 2021); (7) GraphSNN (Wijesinghe & Wang, 2021). Model implementation details are provided in Appendix F.

5.1 Performance on Different Graph Tasks

	MUTAG	PROTEINS	ENZYMES	DD	FRANK	Tox21	NCI1	NCI109
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
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
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
UnionGGCN(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
UnionSAGE(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
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
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
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
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
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 1: Graph Classification Results (Average Accuracy ± Standard Deviation) over 10-fold-CV. The best result is highlighted in **bold**. The winner between a base model with and without our structural coefficient injected is underlined. UnionGGCN is short for UnionGatedGCN and UnionSAGE is short for UnionGraphSAGE.

Graph Classification. We report the results on 8 TUDatasets in Table 1 and the results on 2 OGB datasets in Table 9 (Appendix G). Our UnionSNN outperforms all baselines in 8 out of 10 datasets. We further apply our structural coefficient as a plugin component to two MPNNs: UnionGCN for GCN and UnionGIN for GIN. The results show that our structural coefficient is able to boost up the performance of the base model in almost all cases, with an improvement of up to 6.15%. The graph regression results are shown in Table 10 (Appendix G.2).

Node Classification. We report the results of node classification in Table 2. Our model achieves the best performance on all 5 datasets. When injecting our structural coefficient to existing models such as GCN, GIN, and GraphSNN, their performance improves in almost all cases.

	Cora	Citeseer	PubMed	Computer	Photo
GCN	72.56 ± 4.41	58.30 ± 6.32	74.44 ± 0.71	84.58 ± 3.02	91.71 ± 0.55
UnionGCN (ours)	<u>74.48</u> ± 0.42	<u>59.02</u> ± 3.64	<u>74.82</u> ± 1.10	<u>**88.84**</u> ± 0.27	<u>92.33</u> ± 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)	<u>75.90</u> ± 0.80	<u>63.66</u> ± 1.75	<u>76.78</u> ± 1.02	<u>86.81</u> ± 2.12	<u>92.28</u> ± 0.19
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
GraphSNN	75.44 ± 0.73	64.68 ± 2.72	76.76 ± 0.54	84.11 ± 0.57	<u>90.82</u> ± 0.30
UnionGraphSNN (ours)	<u>75.58</u> ± 0.49	<u>**65.22**</u> ± 1.12	<u>76.92</u> ± 0.56	<u>84.58</u> ± 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 2: Node Classification Results (Average Accuracy ± Standard Deviation) over 10 runs. The best result is highlighted in **bold**. The winner between a base model with and without our structural coefficient injected is <u>underlined</u>.

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 are 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 3. We can see that the best results are achieved by using union-related substructures: 4 on union subgraphs and 2 on union minus subgraphs.

	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 3: Ablation study on local substructure. The best result is highlighted in **bold**.

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. As shown in Table 4, our UnionSNN is a clear winner: it achieves the best result on 5 out of 6 datasets. More experiments based on GCN and GIN are provided in Appendix G.3. These experiments demonstrate that the path matrix better captures substructure information.

	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 4: Ablation study on substructure descriptor. The best result is highlighted in **bold**.

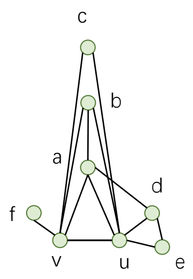
Path matrix encoding method. We test three methods that transform a path matrix to a scalar: (1) sum of all elements in the path matrix (matrix sum); (2) maximum eigenvalue of the path matrix (eigen max); (3) sum of all singular values of the matrix (svd sum) used by UnionSNN in Section 4.2. Table 5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

5.3 Case Study

	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 5: Ablation study on path matrix encoding method. The best result is highlighted in **bold**.

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 5 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 with 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 massages from different neighbors could be properly adjusted when passing to the center node. This also explains the effectiveness of UnionSNN in the 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 5: 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.

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 6. 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. This is because UnionSNN does not need to pad the adjacency matrix and the feature matrix of each graph to the maximum graph size in the dataset, as what GraphSNN does. The analysis of preprocessing computational cost are provided in Appendix G.4.

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

6 Conclusions

In this paper, we propose UnionSNN, a model that is strictly more powerful than 1-WL in distinguishing non-isomorphic graphs. UnionSNN is empowered by an effective shortest-path-based substructure descriptor applied to union subgraphs. Our experimental results show that UnionSNN outperforms state-of-the-art baselines on both graph-level and node-level classification tasks, without sacrificing its computational efficiency. The strength of union subgraphs in capturing neighbor connectivities and benefiting message passing is demonstrated. When applying union subgraphs to existing models, their performance is improved by up to 6.15%.

REFERENCES

Ralph Abboud, Radoslav Dimitrov, and İsmail İlkan Ceylan. Shortest path networks for graph property prediction. *arXiv preprint arXiv:2206.01003*, 2022.

David Ahmedt-Aristizabal, Mohammad Ali Armin, Simon Denman, Clinton Fookes, and Lars Petersson. Graph-based deep learning for medical diagnosis and analysis: Past, present and future. *Sensors*, 2021.

Muhammet Balcilar, Pierre Héroux, Benoit Gauzere, Pascal Vasseur, Sébastien Adam, and Paul Honeine. Breaking the limits of message passing graph neural networks. In *International Conference on Machine Learning*, pp. 599–608. PMLR, 2021.

Cristian Bodnar, Fabrizio Frasca, Yuguang Wang, Nina Otter, Guido F Montufar, Pietro Lio, and Michael Bronstein. Weisfeiler and lehman go topological: Message passing simplicial networks. In *International Conference on Machine Learning*, pp. 1026–1037. PMLR, 2021.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos P Zafeiriou, and Michael Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

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

Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. *international conference on machine learning*, 2016.

David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy D. Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. *neural information processing systems*, 2015.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. *arXiv preprint arXiv:2110.07875*, 2021.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. *the web conference*, 2019.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Roger A Horn and Charles R Johnson. *Matrix analysis*. Cambridge university press, 2012.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Yinan Huang, Xingang Peng, Jianzhu Ma, and Muhan Zhang. Boosting the cycle counting power of graph neural networks with i˜2-gnns. *arXiv preprint arXiv:2210.13978*, 2022.

Kristian Kersting, Nils M. Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. Benchmark data sets for graph kernels, 2016. URL `http://graphkernels.cs.tu-dortmund.de`.

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

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

Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 33:4465–4478, 2020.

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

Ziqi Liu, Chaochao Chen, Longfei Li, Jun Zhou, Xiaolong Li, Le Song, and Yuan Qi. Geniepath: Graph neural networks with adaptive receptive paths. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4424–4431, 2019.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *Advances in neural information processing systems*, 32, 2019.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pp. 43–52, 2015.

Dai Quoc Nguyen, Tu Dinh Nguyen, and Dinh Phung. Universal graph transformer self-attention networks. *arXiv preprint arXiv:1909.11855*, 2019.

Amirhossein Nouranizadeh, Mohammadjavad Matinkia, Mohammad Rahmati, and Reza Safabakhsh. Maximum entropy weighted independent set pooling for graph neural networks. *arXiv preprint arXiv:2107.01410*, 2021.

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

Pál András Papp and Roger Wattenhofer. A theoretical comparison of graph neural network extensions. *arXiv preprint arXiv:2201.12884*, 2022.

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

Hao Tang, Zhiao Huang, Jiayuan Gu, Bao-Liang Lu, and Hao Su. Towards scale-invariant graph-related problem solving by iterative homogeneous gnns. *Advances in Neural Information Processing Systems*, 33:15811–15822, 2020.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*, 2021.

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

Changlin Wan, Muhan Zhang, Wei Hao, Sha Cao, Pan Li, and Chi Zhang. Principled hyperedge prediction with structural spectral features and neural networks. *arXiv preprint arXiv:2106.04292*, 2021.

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

Asiri Wijesinghe and Qing Wang. A new perspective on" how graph neural networks go beyond weisfeiler-lehman?". In *International Conference on Learning Representations*, 2021.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

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

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

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

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

Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From stars to subgraphs: Uplifting any gnn with local structure awareness. *arXiv preprint arXiv:2110.03753*, 2021.

A An Example for Our Isomorphism Concepts

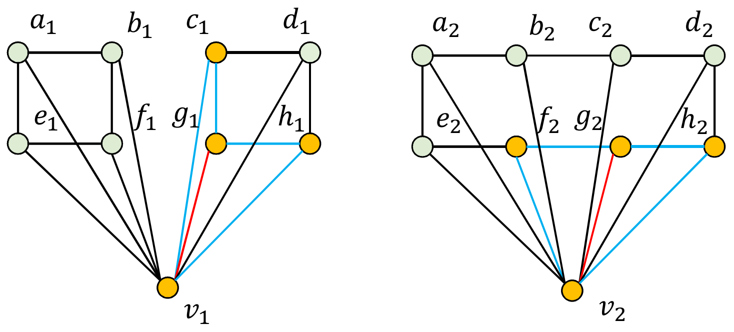


Figure 6: An example that the bijective mapping between the nodes in the subgraphs is not required to be the same as g.

We provide an example for our subgraph and isomorphism concepts and how they are related. As shown in Figure 6, we have two subgraphs S_{v_1} and S_{v_2}. These two subgraphs are overlap-isomorphic. The bijective mapping g is: $g(k_1) = g(k_2), \forall k \in \{v, a, b, c, d, e, f, g, h\}$. Take a pair of corresponding overlap subgraphs as an example: $S_{v_1 \cap g_1}$ and $S_{v_2 \cap g_2}$. They are based on two red edges (v_1, g_1), and (v_2, g_2) and the rest of edges in the overlap subgraphs are colored in blue. It is easy to see that $S_{v_1 \cap g_1}$ and $S_{v_2 \cap g_2}$ are isomorphic (ordinary one). In this ordinary graph isomorphism, its bijective mapping between the nodes is not required to be the same as g, just as in the case of $S_{v_1 \cap g_1}$ and $S_{v_2 \cap g_2}$. 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 S_{v_1} and S_{v_2}.

B Proofs of Theorems

B.1 Proof of Theorem 1

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

Proof. By the definition of union isomorphism, if $S_i \simeq_{union} S_j$, then there exists a bijective mapping $g : \tilde{\mathcal{N}}(i) \to \tilde{\mathcal{N}}(j)$ 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. Since $g(i) = j$ and $g(v) = u$, according to the definition of graph isomorphism, $v_1 \in \tilde{\mathcal{N}}(i) \cap \tilde{\mathcal{N}}(v)$ if and only if $g(v_1) \in \tilde{\mathcal{N}}(j) \cap \tilde{\mathcal{N}}(u)$. Therefore, $g(\cdot)$ is a bijective mapping from $\tilde{\mathcal{N}}(i) \cap \tilde{\mathcal{N}}(v)$ to $\tilde{\mathcal{N}}(j) \cap \tilde{\mathcal{N}}(u)$, and $v_1, v_2 \in \tilde{\mathcal{N}}(i) \cap \tilde{\mathcal{N}}(v)$ are adjacent if and only if $g(v_1), g(v_2) \in \tilde{\mathcal{N}}(j) \cap \tilde{\mathcal{N}}(u)$ are adjacent. This proves that $S_{i \cap v}$ and $S_{j \cap u}$ are isomorphic, and thus $S_i \simeq_{overlap} S_j$. On the contrary, it is possible that $S_i \simeq_{overlap} S_j$ but $S_i \not\simeq_{union} S_j$, as shown by graphs G_1 and G_2 in Figure 2. □

B.2 Proof of 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.*

Proof. We prove the theorem in two steps.

Step 1: We prove that we get a unique $\boldsymbol{P}^{vu}$ from a given $S_{v \cup u}$. With the node order fixed, the rows and the columns of $\boldsymbol{P}^{vu}$ are fixed. Since $\boldsymbol{P}^{vu}$ stores the lengths of all-pair shortest paths, the matrix is unique given an input union subgraph.

Step 2: We prove that we can recover a unique $S_{v \cup u}$ from a given $\boldsymbol{P}^{vu}$. The node set of $S_{v \cup u}$ can be recovered from the row (or column) indices of $\boldsymbol{P}^{vu}$. The edge set of $S_{v \cup u}$ can be recovered from the entries in $\boldsymbol{P}^{vu}$ with the value of "1". Both the node set and the edge set can be uniquely constructed from $\boldsymbol{P}^{vu}$, and thus $S_{v \cup u}$ is unique. □

B.3 Proof of Theorem 3

The proof of Theorem 3 follows a similar flow to that of Theorem 4 in (Wijesinghe & Wang, 2021).

We first define the concept of subtree isomorphism. Let h_v denote the node feature of a node $v \in V$.

Subtree Isomorphism. S_i and S_j are subtree-isomorphic, denoted as $S_i \simeq_{subtree} S_j$, if there exists a bijective mapping g: $\tilde{\mathcal{N}}(i) \to \tilde{\mathcal{N}}(j)$ such that $g(i) = j$, and for any $v \in \mathcal{N}(i)$ and $g(v) = u$, $h_v = h_u$.

We assume that $\mathcal{H}, \mathcal{A}$ and $\mathcal{W}$ are three countable sets that $\mathcal{H}$ is the node feature space, $\mathcal{A}$ is the structural coefficient space, and $\mathcal{W} = \{a^{ij}h_i | a^{ij} \in \mathcal{A}, h_i \in \mathcal{H}\}$. Suppose H and W are two multisets that contain elements from $\mathcal{H}$ and $\mathcal{W}$, respectively, and $|H| = |W|$. In order to prove Theorem 3, we need to use Lemmas 1 and 2 in Wijesinghe & Wang (2021). To be self-contained, we repeat the lemmas here and refer the readers to Appendix C of Wijesinghe & Wang (2021) for the proofs.

Lemma 1. *There exists a function f s.t. $\pi(H, W) = \sum_{h \in H, w \in W} f(h, w)$ is unique for any distinct pair of multisets (H, W).*

Lemma 2. *There exists a function f s.t. $\pi'(h_v, H, W) = \gamma f(h_v, |H| h_v) + \sum_{h \in H, w \in W} f(h, w)$ is unique for any distinct (h_v, H, W), where $h_v \in H$, $|H| h_v \in W$, and γ can be an irrational number.*

From the lemmas above, we can now prove Theorem 3:

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

Proof. By Theorem 3 in (Wijesinghe & Wang, 2021), if a GNN M satisfies the two following conditions, then M is strictly more powerful than 1-WL in distinguishing non-isomorphic graphs.

1. M can distinguish at least one pair of neighborhood subgraphs S_i and S_j such that S_i and S_j are subtree-isomorphic, but they are not isomorphic, and $\{\{\tilde{a}^{iv} | v \in \mathcal{N}(i)\}\} \neq \{\{\tilde{a}^{ju} | u \in \mathcal{N}(j)\}\}$, where $\tilde{a}^{vu}$ is the normalised value of a^{vu};
2. The aggregation scheme $\Phi(h_v^{(t)}, \{\{h_u^{(t)} | u \in \mathcal{N}(v)\}\}, \{\{ (\tilde{a}^{uv}, h_u^{(t)}) | u \in \mathcal{N}(v)\}\})$ is injective.

For condition 1, the pair of graphs in Figure 2 satisfies the condition, and can be distinguished by UnionSNN as they are not union-isomorphic.

For condition 2, by Lemmas 1 and 2 and the fact that the MLP $\text{Trans}(\cdot)$ is a universal approximator and can model and learn the required functions, we conclude that UnionSNN satisfies this condition.

Therefore, UnionSNN is more expressive than 1-WL in testing non-isomorphic graphs. □

C Definitions of three other substructure descriptor functions

Edge Betweenness. The betweenness centrality of an edge $e \in E$ in a graph $G = (V, E, X)$ is defined as the sum of the fraction of all-pair shortest paths that pass through e:

$$c_G(e) = \sum_{v,u \in V} \frac{\sigma(v, u, G|e)}{\sigma(v, u, G)}, \tag{6}$$

where $\sigma(v, u, G)$ is the number of shortest paths between v and u, and $\sigma(v, u, G|e)$ is the number of those paths passing through edge e.

Node/Edge Counting. GraphSNN (Wijesinghe & Wang, 2021) defines a structural coefficient based on the number of nodes and edges in the (sub)graph. When applied to the union subgraph, we have

$$\omega(S_{v\cup u}) = \frac{|E_{v\cup u}|}{|V_{v\cup u}| \cdot |V_{v\cup u} - 1|} |V_{v\cup u}|^{\lambda}, \tag{7}$$

where $\lambda = 1$ for node classification and $\lambda = 2$ for graph classification.

Ollivier Ricci Curvature. Given a node $v \in V$ in a graph $G = (V, E, X)$, a probability vector of v is defined as:

$$\mu_v^{\alpha} : u \mapsto \begin{cases} \alpha, u = v \\ \frac{1-\alpha}{d_v}, u \in \mathcal{N}(v), \\ 0, \text{ otherwise} \end{cases} \tag{8}$$

where $\alpha \in [0, 1)$ is a hyperparameter. Following Ye et al. (2019), we use $\alpha = 0.5$ in all our experiments. The α-Ricci curvature κ_{vu}^{α} on an edge $(v, u) \in E$ is defined by:

$$\kappa_{vu}^{\alpha} = 1 - \frac{\text{Wass}(\mu_v^{\alpha}, \mu_u^{\alpha})}{d(v, u)}, \tag{9}$$

where Wass$(\cdot)$ denotes the Wasserstein distance and $d(\cdot)$ denotes the shortest path length between v and u. The Wasserstein distance can be estimated by the optimal transportation distance, which can be solved by the following linear programming:

$$\begin{aligned} &\min_{M} \sum_{i\in\tilde{\mathcal{N}}(v), j\in\tilde{\mathcal{N}}(u)} d(i,j)\, M(i,j) \\ \text{s.t.} &\sum_{j\in\tilde{\mathcal{N}}(u)} M(i,j) = \mu_v^{\alpha}(i), \forall i \in \tilde{\mathcal{N}}(v); \\ &\sum_{i\in\tilde{\mathcal{N}}(v)} M(i,j) = \mu_u^{\alpha}(j), \forall j \in \tilde{\mathcal{N}}(u). \end{aligned} \tag{10}$$

D The Connection with Higher-order WL Tests

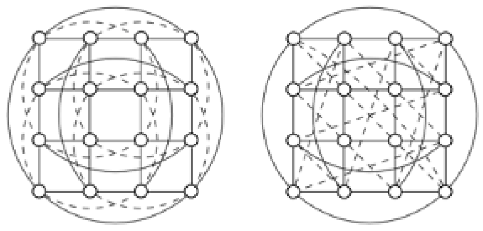


Figure 7: These two graphs can be distinguished by UnionSNN but 3-WL will fail.

As discussed in Papp & Wattenhofer (2022), high-order WL tests 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 such that the 3-WL test and GraphSNN are not stronger than UnionSNN, i.e., some graphs can be distinguished by UnionSNN but not by 3-WL and GraphSNN. Similar analyses have been performed in existing works, such as Balcilar et al. (2021) and Bodnar et al. (2021). As an example, UnionSNN can distinguish the 4x4 Rook's graph and the Shrikhande graph (as shown in Figure 7, which is cited from Huang et al. (2022)), while 3-WL and GraphSNN cannot, which suggests that UnionSNN is stronger than 3-WL and GraphSNN on such graphs. This could be explained by the fact that the number of 4-cycles in these two graphs are different, and UnionSNN is able to reflect the number of 4-cycles with the consideration of E_3^{vu} edges in union subgraphs.

E Datasets Statistics

Statistics of the datasets used are summarized in Tables 7 and 8.

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

Table 7: Statistics of Graph Classification/Regression Datasets.

	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 8: Statistics of Node Classification Datasets.

F Implementation Details

The settings of our experiments mainly follow those in Dwivedi et al. (2020). For graph classification on TUDataset, we split each one into 8:1:1 for training, validation and test, respectively. We evaluate each model with the same random seed for 10-fold cross-validation and report the average accuracy. As for graph classification on OGB datasets, we use the scaffold splits run 10 times with different random seeds and report the average ROC-AUC due to the class imbalance issue. For graph regression, we use the data split of ZINC10k and ZINC-full datasets from Dwivedi et al. (2020). We keep the total parameters of all GNNs less than 100K and 500K on ZINC10k and ZINC-full datasets, respectively, by adjust the hidden dimensions. For node classification, we use the standard split in the original paper and report the average accuracy for 10 runs with different random seeds. We choose the best values of the learning rate from {0.02, 0.01, 0.005, 0.001}, weight decay from {0.005, 0.001, 0.0005}, hidden dimension from {16, 64, 128, 256} and dropout from {0, 0.2, 0.4, 0.6, 0.8}. The whole network is trained in an end-to-end manner using the Adam optimizer (Kingma & Ba, 2014). We use the early stopping criterion, i.e., we stop the training once there is no further improvement on the validation loss during 25 epochs. All experiments were conducted in a Linux server with Intel(R) Core(TM) i9-10940X CPU (3.30GHz), GeForce GTX 3090 GPU, and 125GB RAM.

G More experimental results

G.1 Graph Classification on OGB datasets.

We conduct experiments on OGBG-MOLHIV and OGBG-MOLBBBP datasets. As shown in Table 9, UnionSNN outperforms all 6 baseline methods.

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

G.2 Graph Regression

We conduct experiments on ZINC10k and ZINC-full datasets, as shown in Table 10, UnionSNN outperforms all 7 baseline methods, and the performance of MPNNs with our structural coefficient (UnionGCN, UnionGIN and UnionGraphSAGE) are also beyond their counterparts. We are not reporting the result of CurvGN on ZINC-full since the preprocessing of computes Ricci Curvature would take more than 100 hours.

	ZINC10k	ZINC-full
GCN	0.4652	0.1159
UnionGCN	<u>0.4211</u>	<u>0.0879</u>
GIN	0.3745	0.1496
UnionGIN	<u>0.3607</u>	<u>0.1343</u>
GraphSAGE	0.4468	0.1219
UnionGraphSAGE	<u>0.4404</u>	<u>0.1134</u>
GAT	0.4869	0.1112
GatedGCN	0.4425	0.0919
CurvGN	0.4512	-
3WLGNN	0.3616	0.0964
UnionSNN	**0.3566**	**0.0843**

Table 10: Graph regression results (test MAE) on ZINC10k and ZINC-full datasets. The best result is highlighted in **bold**. The winner between a base model with and without our structural coefficient injected is <u>underlined</u>.

G.3 Substructure Descriptor Ablation Study on Other MPNN variants

In this subsection, we tried to plugin different substructure descriptors into GCN and GIN, and the results are reported in Table 11 and Table 12. The best results are highlighted in bold and the second-best results are highlighted in italic. The experiments show that the path matrix (UnionGCN and UnionGIN) can always get the top 2 results comparing to other substructure descriptors. This demonstrates the effectiveness of the substructure descriptor in UnionSNN.

	MUTAG	PROTEINS	ENZYMES	NCI109
GCN	77.13 ± 5.24	73.89 ± 2.85	64.33 ± 5.83	75.91 ± 1.53
BetGCN	76.61 ± 8.86	73.23 ± 6.45	63.04 ± 8.70	76.12 ± 1.35
CountGCN	78.56 ± 4.74	**75.02** ± 4.49	**64.67** ± 7.14	77.84 ± 1.36
CurvatureGCN	**84.59** ± 7.30	73.94 ± 6.01	61.32 ± 2.64	78.35 ± 1.74
LapGCN	79.13 ± 6.83	74.24 ± 5.52	64.57 ± 2.35	**80.33** ± 1.59
UnionGCN	*81.87* ± 3.81	**75.02** ± 2.50	**64.67** ± 7.14	*79.50* ± 1.82

Table 11: Ablation study on substructure descriptor based on GCN. The best result is highlighted in **bold**and the second-best results are highlighted in *italic*.

G.4 Efficiency Analysis for Preprocessing

	MUTAG	PROTEINS	ENZYMES	NCI109
GIN	86.23 ± 8.17	72.86 ± 4.14	65.83 ± 5.93	80.95 ± 1.87
BetGIN	86.70 ± 5.42	*73.31* ± 3.20	66.83 ± 7.94	81.42 ± 1.80
CountGIN	87.25 ± 4.78	**73.49** ± 3.91	66.67 ± 4.53	*81.83* ± 1.62
CurvatureGIN	86.62 ± 5.40	62.86 ± 8.65	64.67 ± 5.72	80.96 ± 2.29
LapGIN	*87.81* ± 5.24	71.87 ± 4.13	**68.50** ± 7.97	81.54 ± 2.38
UnionGIN	**88.86** ± 4.33	73.22 ± 3.90	*67.83* ± 6.10	**82.24** ± 1.24

Table 12: Ablation study on substructure descriptor based on GIN. The best result is highlighted in **bold**and the second-best results are highlighted in *italic*.

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

Table 13: Time cost (seconds) for computing structural coefficients in preprocessing.

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, and Ricci curvature (Curvature) in CurvGN. As shown in Table 13, the preprocessing time of UnionSNN is comparable to that of other models. This demonstrates that our proposed structural coefficient derived by SVD on the path matrix is able to improve classification performance without significantly sacrificing the preprocessing cost.