CSCN: The Crossed Subtree Convolutional Network

Zongjun Han[1], **Lu Bai**[1]*, **Lixin Cui**[2], **Ming Li**[3], **Hangyuan Du**[4*]
[1]School of Artificial Intelligence, Beijing Normal University, Beijing, China.
[2]School of Information, Central University of Finance and Economics, Beijing, China.
[3]Zhejiang Key Laboratory of Intelligent Education Technology and Application,
Zhejiang Normal University, Jinhua, China.
[5]School of Computer and Information Technology, Shanxi University, Taiyuan, China.
bailu@bnu.edu.cn, duhangyuan@sxu.edu.cn

Abstract

Message Passing Neural Networks (MPNNs) commonly suffer from over-squashing and over-smoothing issues, that limit their performance on graph-level tasks. To overcome the shortcoming, in this paper we propose a **C**rossed **S**ubtree **C**onvolutional **N**etwork (CSCN) for graph classification. Specifically, CSCN first introduces an ordered subtree alignment strategy to decompose the original graph into an ordered sequence of structured subtrees. This sequence preserves rich local connectivity patterns and reduces dependence on input node indices. Based on these subtrees, we design a crossed subtree convolution operation. Specifically, it utilizes a sliding window to enable information interactions across different subtrees while simultaneously aggregating node features within each subtree. This design captures local information effectively while also modeling long-range dependencies. In this way, CSCN avoids deep message passing over the global graph topology and thus alleviates both over-squashing and over-smoothing. Experimental results on multiple graph classification benchmarks show that the proposed CSCN achieves competitive performance.

1 Introduction

Graphs are general data structures with strong expressive power, enabling the effective representation of irregular data in non-Euclidean spaces. They are widely applied in fields such as social networks, traffic systems, protein interaction networks [1–4], etc. In recent years, Message Passing Neural Networks (MPNNs) have become the most popular Graph Neural Networks (GNNs) for graph data analysis. These models update node representations by aggregating the information from neighboring nodes based on the graph structure. After n message passing layers, each node can aggregate node features from its n-hop neighbors. This allows effective capture of local neighbor information. Therefore, MPNNs have achieved strong performance on various node-level tasks.

However, graph-level tasks typically require models to capture long-range dependencies beyond local neighbors [5]. To achieve this, standard MPNNs often employ many message passing layers to expand the receptive field of a node. Nevertheless, the MPNNs with many layers commonly introduce issues such as over-squashing and over-smoothing [6, 7]. This is because the receptive field of a node may expand rapidly when the number of layers increases, forcing a large amount of information from distant nodes to be compressed into fixed-dimensional node representations. This compression can lead to information loss, a phenomenon referred to as over-squashing. Moreover, during multiple message passing layers, certain nodes are aggregated multiple times, causing discriminative information to be gradually overwhelmed by redundant information. As a result, node representations

*Corresponding Authors: Lu Bai and Hangyuan Du.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

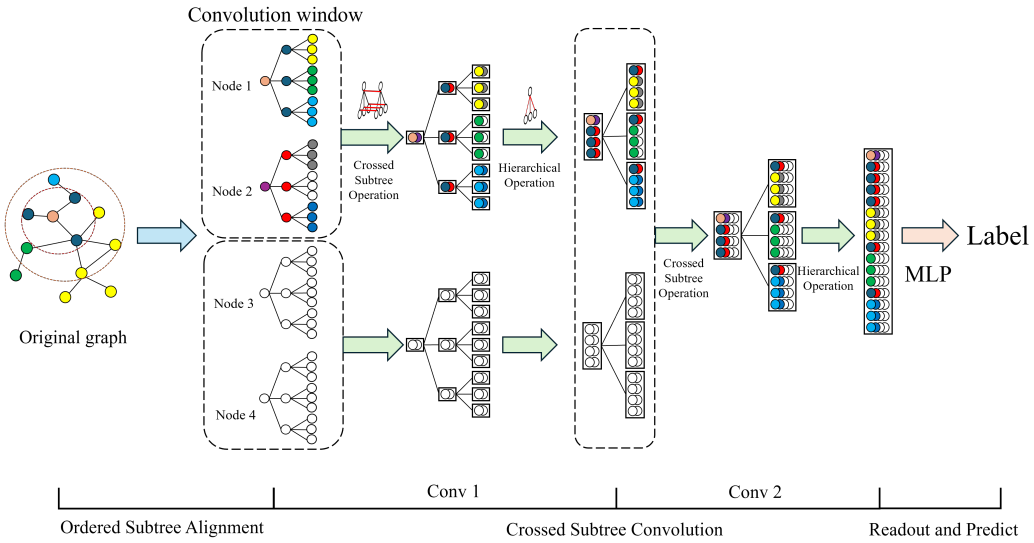


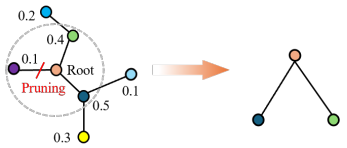
Figure 1: The overall framework of the Crossed Subtree Convolutional Network. For clarity, only four nodes are depicted.

tend to become increasingly similar, giving rise to the over-smoothing phenomenon. These issues significantly limit the performance of MPNNs on graph-level tasks.

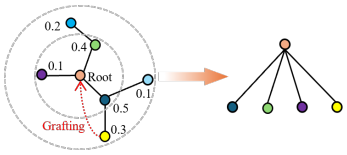
On the other hand, recent studies have explored alternative frameworks beyond the aforementioned MPNNs. Some methods aim to transform graphs into more structured representations for subsequent learning. For instance, several works [8–14] have constructed subgraphs to leverage richer structural patterns. This strategy partially breaks the symmetries induced by the local aggregation functions, thereby achieving performance superior to the standard MPNNs. Despite their effectiveness, these methods often overlook information interactions across different subgraphs, leading to insufficient capture of long-range dependencies. As a result, their performance on graph-level tasks remains limited.

To address the aforementioned challenges, we propose a Crossed Subtree Convolutional Network (CSCN) for graph classification. The CSCN provides a novel convolutional framework to integrate both local and global information. First, utilizing global topological information, we decompose an original graph into multiple structured k-layer m-ary subtrees. Each subtree is rooted at a node in the graph, and all subtrees are organized into an ordered sequence. Subsequently, we design a crossed subtree convolution operation to capture the information from this ordered subtree sequence. The node-rooted subtrees provide controllable receptive fields, enabling the CSCN to avoid the exponentially expanding receptive fields induced by deep message passing over the global graph topology in standard MPNNs. The crossed subtree convolution consists of a crossed subtree operation and a hierarchical operation. The crossed subtree operation slides a window over the ordered subtree sequence to enable information interactions across separated subtrees, thereby capturing long-range dependencies. Meanwhile, the hierarchical operation captures local information within each subtree. This design prevents information from increasingly large neighborhoods from being forced into fixed-dimensional representations, thus alleviating over-squashing. It also reduces repeated neighborhood mixing that tends to homogenize node representations, thereby alleviating over-smoothing. The main contributions are summarized as follows.

- **Ordered Subtree Alignment**. We propose a graph decomposition strategy based on the topological information of the original graph. It transforms irregular graphs into ordered sequences of structured subtrees. This strategy preserves rich local connectivity patterns while helping reduce dependence on input node indices.
- **Crossed Subtree Convolution**. We introduce a novel crossed subtree convolution that consists of a crossed subtree operation and a hierarchical operation. The crossed subtree operation enables information interactions across different subtrees via a sliding window over the ordered subtree sequence, while the hierarchical operation captures local information within each subtree. This design effectively integrates both global and local information.



(a) Graph pruning, when $|\mathcal{N}_1(v_i)| > m$.



(b) Graph grafting, when $|\mathcal{N}_1(v_i)| < m$.

Figure 2: Illustrations of graph pruning and grafting strategies. (a) Example with $m = 2$ and $|\mathcal{N}_1(v_i)| = 3$. (b) Example with $m = 4$ and $|\mathcal{N}_1(v_i)| = 3$.

- **Competitive Graph Classification Performance**. We validate the effectiveness of our model on standard graph classification benchmarks. Extensive experiments demonstrate that our method achieves competitive performance.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 presents the details of the proposed CSCN. Section 4 presents the experimental settings and results. Section 5 concludes the paper.

2 Related Works

In this section, we briefly review some related works that motivate our method, including message passing neural networks and subgraphs in graph representation learning.

Message Passing Neural Networks. MPNNs constitute a paradigm in graph representation learning. Models such as GCN [15], GraphSAGE [16], GAT [17], and GIN [18] improve neighborhood aggregation from different perspectives and have achieved strong performance on various graph learning tasks. In standard MPNNs, each layer updates node representations by aggregating the information from local neighbors, so that deeper models can progressively enlarge the receptive field. However, for graph-level tasks that require capturing dependencies beyond local neighborhoods, this paradigm often becomes less effective as depth increases. In particular, the progressive expansion of receptive fields may force the information from large neighborhoods into fixed-dimensional representations, while repeated neighborhood mixing may reduce the distinguishability of node features. To address these limitations, representative approaches such as DBGNN [19], GTC [20], LightGNN [21], and Flow2GNN [22] have been proposed to alleviate over-squashing and over-smoothing. However, these methods generally remain within the message passing framework over the original graph topology, with limited mechanisms for explicitly controlling receptive field growth.

Subgraphs in Graph Representation Learning. Subgraph-based representations have been widely studied to enhance the expressive power of GNNs. Some methods decompose the original graph into multiple subgraphs and leverage standard GNN architectures to extract structural patterns from them. Examples include K-Reconstruction GNNs [23], ID-GNN [24], DySubC [25], and GSN [10]. Other methods explicitly replace node representations with subgraph representations. NGNN [26] extracts node-rooted subgraphs and uses their embeddings as substitutes for node features, UnionSNN [27] aggregates union subgraphs over node pairs, and GNN-AK [28] improves local structure modeling through induced subgraphs. Among tree-based variants, QS-CNN [14] and GTR [29] also organize graphs into tree-like substructures: QS-CNN constructs node-rooted subtrees and applies fixed convolution kernels to capture intra-subtree features, while GTR models dependencies based on graph skeleton trees. However, these methods either encode substructures in isolation or rely on a single tree-like global structure, limiting information interactions across substructures. In contrast, CSCN organizes graphs into a PageRank-ordered sequence of aligned subtrees and explicitly models cross-subtree interactions through a sliding convolution window. This design facilitates the modeling of both local connectivity patterns and long-range dependencies without message passing over the global graph topology.

3 Crossed Subtree Convolutional Network

In this section, we detail the proposed Crossed Subtree Convolutional Network (CSCN). As illustrated in Figure 1, the framework operates in three sequential stages including ordered subtree alignment,

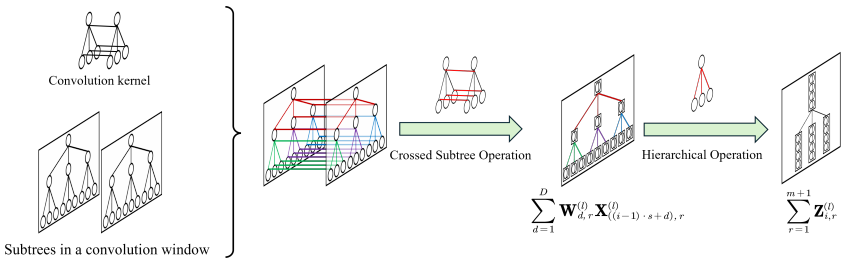


Figure 3: Illustration of the crossed subtree convolution within a convolution window, utilizing a window size of 2.

crossed subtree convolution, and graph readout and prediction. We first introduce the preliminaries and then describe each step in detail.

3.1 Preliminaries

Let $G(V, E)$ denote a graph, where $V = \{v_1, v_2, \ldots, v_N\}$ denotes the set of N nodes and $E = \{e_1, e_2, \ldots, e_M\}$ denotes the set of M edges. Each node $v_i \in V$ is associated with a feature vector $\boldsymbol{x}_{v_i} \in \mathbb{R}^{c_0}$, where c_0 denotes the dimension of node features. The set of n-hop neighbors of node v_i is denoted by $\mathcal{N}_n(v_i)$, and $|\mathcal{N}_n(v_i)|$ represents the number of its n-hop neighbors. Given a graph G, it is associated with a class label $y \in \mathcal{Y}$, where $\mathcal{Y}$ denotes the set of possible classes. Our objective is to learn a graph-level representation $\mathbf{H}$ for G and utilize it as input to a Multilayer Perceptron (MLP) to predict the class label.

3.2 Ordered Subtree Alignment

Standard Message Passing Neural Networks (MPNNs) typically rely on iterative message passing over the global graph topology. As noted in Section 1, this paradigm often suffers from over-squashing and over-smoothing. Motivated by recent subgraph-based approaches [14, 30], we propose decomposing the graph into an ordered sequence of structured subtrees.

PageRank captures global topological information, effectively reflecting the importance of nodes within a graph. Consequently, we utilize PageRank scores to construct the ordered subtree sequence. First, we compute the PageRank scores for all nodes and arrange them into an ordered sequence based on these scores. This ordering reduces dependence on input node indices in subsequent stages. For each node $v_i \in V$, we then construct a k-layer m-ary subtree rooted at v_i, where $k, m > 1$. The root v_i constitutes the first layer of the subtree. The subtree is expanded layer-wise. For a node at the t-th layer ($t < k$), its 1-hop neighbors are selected as children to form the $(t+1)$-th layer. This expansion continues until the subtree reaches a depth of k. Since the number of neighbors rarely matches the predefined number of children m, we employ graph pruning and grafting strategies to enforce a uniform m-ary structure, as shown in Figure 2. The strategies are described as follows:

Graph Pruning. When the number of 1-hop neighbors exceeds the predefined number of children (i.e., $|\mathcal{N}_1(v_i)| > m$), we sort the neighbors in $\mathcal{N}_1(v_i)$ by their PageRank scores and retain the top-m nodes as children.

Graph Grafting. When the number of 1-hop neighbors is insufficient (i.e., $|\mathcal{N}_1(v_i)| < m$), we iteratively expand the search scope to neighbors at larger hops. In each iteration, neighbors are sorted by PageRank scores and sequentially added as children of v_i until the count reaches m.

These operations construct the subtree representation without modifying the original graph. A neighbor excluded from one expansion remains the root of its own subtree, but the omitted local relation is not necessarily preserved. Grafted links between parent and child nodes encode context from multiple hops rather than introduce new edges into the original graph.

By leveraging global topological information, this step decomposes the graph into an ordered subtree sequence. The node ordering helps reduce dependence on input node indices, while the pruning and grafting strategies ensure the construction of structurally aligned subtrees. This alignment establishes a fixed receptive field for each node, enabling the model to capture rich local connectivity patterns effectively.

3.3 Crossed Subtree Convolution

To effectively capture information from the ordered subtree sequence and obtain graph-level representation, we design a crossed subtree convolution. The crossed subtree convolution employs a convolution window that slides over the ordered subtree sequence. As shown in Figure 3, within each convolution window, the convolution kernel performs the convolution covering each local unit (i.e., each parent node and its m children) and the corresponding local unit in other subtrees. The convolution kernel operates along two complementary dimensions, i.e., crossed subtree operation and hierarchical operation. As a result, multiple k-layer m-ary subtrees are mapped to a new $(k-1)$-layer m-ary subtree as the output of the convolution window.

Formally, let $\mathbf{X}_j^{(l)} \in \mathbb{R}^{(m+1)\times C^{(l)}\times P^{(l)}}$ denote the representation of the j-th subtree at the l-th $(0 < l < k)$ convolution layer. Here, $(m+1)$ denotes the size of a local unit, i.e., the number of nodes in a local unit. $C^{(l)}$ denotes the node feature dimension and $P^{(l)}$ is the number of local units in the subtree at the l-th convolution layer, i.e., the number of parent nodes in the subtree. The entire subtree sequence at the l-th convolution layer is denoted by $\mathbf{X}^{(l)} \in \mathbb{R}^{N^{(l)}\times(m+1)\times C^{(l)}\times P^{(l)}}$, where $N^{(l)}$ is the number of subtrees. We further define a learnable convolution kernel $\mathbf{W}^{(l)} \in \mathbb{R}^{D\times(m+1)\times C^{(l+1)}\times C^{(l)}}$, where D represents the size of the convolution window sliding along the ordered subtree sequence. The specific operations are defined below.

Crossed Subtree Operation. The crossed subtree operation captures global dependencies through the information interactions across the ordered subtree sequence. As the convolution window slides over the sequence with a stride of s, it covers D subtrees. For the i-th window at the l-th convolution layer, the operation is formulated as

$$\mathbf{Z}_{i,r}^{(l)} = \sum_{d=1}^{D} \mathbf{W}_{d,r}^{(l)} \mathbf{X}_{((i-1)\cdot s+d),\, r}^{(l)}, \tag{1}$$

where $r \in \{1, \ldots, m+1\}$ indexes the node position within a local unit. $\mathbf{W}_{d,r}^{(l)} \in \mathbb{R}^{C^{(l+1)}\times C^{(l)}}$ is the weight matrix assigned to the r-th node position of the local unit within the d-th subtree of the convolution window, and $\mathbf{X}_{((i-1)\cdot s+d),\, r}^{(l)} \in \mathbb{R}^{C^{(l)}\times P^{(l)}}$ is the feature matrix of the r-th node position in each local unit of the subtree corresponding to the current window position. $\mathbf{Z}_{i,r}^{(l)} \in \mathbb{R}^{C^{(l+1)}\times P^{(l)}}$ represents the feature matrix obtained by aggregating information from the D neighboring subtrees at the same node position r. The crossed subtree operation facilitates information interactions among nodes at the identical structural position within each convolution window.

Hierarchical Operation. While the crossed subtree operation handles the information interactions across different subtrees, the hierarchical operation focuses on capturing the local information within each subtree. Specifically, it aggregates the features $\mathbf{Z}_{i,r}^{(l)}$ across all node positions r in the local unit. The operation is formulated as

$$\text{Aggregate}_i^{(l)} = \sum_{r=1}^{m+1} \mathbf{Z}_{i,r}^{(l)}. \tag{2}$$

This hierarchical operation aggregates features within local units, allowing the model to progressively learn higher-level features from the leaves up to the root.

Overall crossed subtree convolution. By substituting the definition of the crossed subtree operation into the above aggregation and applying a non-linear activation function, we obtain the unified convolution formula. The l-th crossed subtree convolution layer for the i-th convolution window is calculated as

$$\mathbf{H}_i^{(l)} = \sigma\left(\sum_{d=1}^{D}\sum_{r=1}^{m+1} \mathbf{W}_{d,r}^{(l)} \mathbf{X}_{((i-1)\cdot s+d),\, r}^{(l)} + \mathbf{b}^{(l)}\right), \tag{3}$$

where $\mathbf{b}^{(l)} \in \mathbb{R}^{C^{(l+1)}\times P^{(l)}}$ is the bias term, $\sigma(\cdot)$ is the activation function, and $\mathbf{H}_i^{(l)} \in \mathbb{R}^{C^{(l+1)}\times P^{(l)}}$ is the output feature matrix. In addition, when $l < k-1$, the output serves as the input for the next convolution after reshaping

$$\mathbf{X}_i^{(l+1)} = \text{Reshape}\left(\mathbf{H}_i^{(l)}\right) \in \mathbb{R}^{(m+1)\times C^{(l+1)}\times P^{(l+1)}}, \tag{4}$$

where $P^{(l+1)} = (P^{(l)} - 1)/m$ denotes the number of local units at the $(l+1)$-th convolution layer. The subtree sequence at the $(l+1)$-th convolution layer is denoted by $\mathbf{X}^{(l+1)} \in \mathbb{R}^{N^{(l+1)} \times (m+1) \times C^{(l+1)} \times P^{(l+1)}}$. The number of subtrees $N^{(l+1)}$ is calculated as

$$N^{(l+1)} = \frac{N^{(l)} - D}{s} + 1. \tag{5}$$

The crossed subtree convolution enables effective information capture from the ordered subtree sequence. On the one hand, since the sequence is ordered based on PageRank scores, adjacent subtrees in the sequence correspond to nodes with similar global topological importance. The crossed subtree operation allows the model to capture long-range dependencies between nodes that may be distant in the original graph but topologically similar. On the other hand, the hierarchical operation aggregates features within local units of each subtree and progressively propagates information toward parent nodes. It avoids message passing over the global graph topology and alleviates the over-squashing and over-smoothing issues commonly observed in deep MPNNs.

3.4 Readout and Prediction

Following the $(k-1)$-th crossed subtree convolution layer, each graph is represented by a sequence of subtree roots. To handle graphs of varying sizes, we apply global sum pooling to generate the final graph-level representation as

$$\mathbf{H} = \sum_{i=1}^{N^{(k-1)}} \mathbf{H}_i^{(k-1)}. \tag{6}$$

The resulting graph-level representation $\mathbf{H}$ is fed into an MLP to obtain the classification predictions. Let $\text{MLP}(\mathbf{H}) \in \mathbb{R}^{|\mathcal{Y}|}$ denote the predicted probability distribution, where $|\mathcal{Y}|$ is the number of possible classes. The model is trained using the loss as below

$$\mathcal{L} = -\sum_{c=1}^{|\mathcal{Y}|} y_c \log\left([\text{MLP}(\mathbf{H})]_c\right) \tag{7}$$

where y_c is the binary indicator (0 or 1) if class c is the correct classification for the input graph, and $[\text{MLP}(\mathbf{H})]_c$ is the predicted probability for class c.

The proposed Crossed Subtree Convolutional Network first decomposes each graph into an ordered subtree sequence. This strategy preserves rich local connectivity patterns while reducing dependence on input node indices. It further enables information interactions across different subtrees through a sliding convolution window, thereby integrating both local and global information. Finally, a global sum pooling produces a graph-level representation for classification. By capturing both local information and long-range dependencies, CSCN provides an effective framework for graph classification.

3.5 Discussion of CSCN

To better understand the efficacy of CSCN, we analyze its information aggregation behavior and contrast it with standard message passing.

Proposition 3.1. *Consider a standard MPNN with 1-hop message passing on a graph with maximum degree $\Delta > 2$. As the number of message passing layers l increases, the receptive field of each node grows at most exponentially with l.*

Proposition 3.2. *In CSCN, the receptive field of a convolution window along the ordered subtree sequence grows at most linearly with the number of crossed subtree convolution layers l when $s = 1$.*

The proof of Proposition 3.1 and Proposition 3.2 can be found in Appendix A.1 and Appendix A.2, respectively. Proposition 3.1 shows that stacking 1-hop message passing layers causes the node receptive field to grow at most exponentially. This forces information from a large neighborhood into fixed-dimensional representations and aggravates over-squashing. Moreover, repeated neighborhood aggregation tends to homogenize node representations, aggravating over-smoothing. In contrast, Proposition 3.2 indicates that CSCN expands the receptive field in a controlled manner. Specifically, ordered subtree alignment decomposes a graph into an ordered sequence of structurally aligned

Table 1: Classification accuracy (% $\pm$ standard error) for comparisons with deep learning methods.

	MUTAG	TOX21	PROTEINS	D&D	NCI1	FRANK.
GRAPHSAGE	80.38±10.98	88.36±0.15	74.87±3.38	73.10±4.34	63.94±2.52	52.95±4.01
GAT	77.56±10.49	90.35±0.71	74.34±2.09	74.25±3.76	78.07±1.94	62.85±1.90
GIN	86.23±8.17	91.74±0.95	72.86±4.14	70.29±2.96	82.29±1.98	66.50±2.37
UGFORMER	75.66±8.67	88.06±0.50	70.17±5.42	75.51±3.52	68.84±1.54	56.13±2.51
UNIONSNN	87.31±5.29	91.76±0.85	75.02±2.50	77.00±2.37	82.34±1.93	67.83±1.99
DROPGIN	84.09±8.42	91.66±0.92	73.39±4.90	71.05±0.68	82.19±1.39	66.91±2.16
NESTEDGIN	86.23±8.82	91.42±1.18	68.55±3.22	70.04±4.32	82.04±2.23	67.07±1.46
CSCN	**90.44±3.40**	**92.81±0.59**	**75.57±0.40**	**78.69±1.42**	**82.87±2.18**	**74.01±0.77**

	NCI109	IMDB_B	IMDB_M	RED_B_2K	RED_M_5K	RED_M_12K
GIN	**80.95±1.87**	-	-	89.90±1.90	56.10±1.70	-
GRAPHSAGE	65.46±1.12	-	-	84.30±1.90	50.00±1.30	39.10±0.60
SAGPOOL	74.06±0.78	72.16±0.88	49.47±0.56	84.70±4.40	46.29±5.61	42.30±3.70
DIFFPOOL	61.98±1.98	73.14±0.70	51.31±0.72	82.12±1.06	-	-
ABDPOOL	71.78±1.35	70.58±0.71	50.63±1.47	82.75±0.82	-	-
SCHPOOL	75.91±0.20	74.45±0.28	**53.02±0.15**	85.82±0.24	-	-
CSCN	80.71±1.92	**74.50±0.95**	51.53±0.56	**90.90±0.62**	**56.11±0.87**	**46.71±1.24**

subtrees, where each subtree has a fixed-size receptive field. Crossed subtree convolution then enables information interactions across different subtrees via a sliding window over the ordered subtree sequence. As a result, the receptive field of a convolution window grows at most linearly with the number of crossed subtree convolution layers, allowing CSCN to incorporate broader context without relying on deep message passing over the global graph topology. This controlled expansion helps alleviate both over-squashing and over-smoothing.

To summarize, the advantages of CSCN mainly come from (1) establishing a controllable receptive field via ordered subtree alignment, and (2) enabling information interactions across subtrees through crossed subtree convolution, without message passing over the global graph topology.

4 Experiments

In this section, we evaluate our Crossed Subtree Convolutional Network through extensive experiments. We first describe the public datasets and experimental settings, followed by a discussion of the results. Finally, we conduct ablation studies and analyze the computational complexity.

4.1 Experiment Settings

Datasets. We evaluate the proposed CSCN on 18 publicly available graph classification benchmarks, covering three domains. The bioinformatics domain includes MUTAG, Tox21, PTC_MR, PROTEINS, D&D, NCI1, NCI109, FRANKENSTEIN (FRANK.), and PPIs [31], together with ogbg-molhiv and ogbg-molbbbp [32]. The social network domain includes IMDB_B, IMDB_M, RED_B_2K, RED_M_5K, and RED_M_12K [31]. The computer vision domain includes SHOCK and BAR31 [33]. Collectively, these benchmarks span a broad range of graph sizes and densities. Dataset statistics are summarized in Appendix B.

Alternative Methods for Comparisons. We select a diverse set of baselines for comparison, covering both deep learning methods and graph kernel approaches. The deep learning baselines fall into four categories: (1) standard MPNNs, including GraphSAGE [16], GAT [17], and GIN [18]; (2) graph pooling methods, including DiffPool [34], SAGPool [35], ABDPool [36], and SCHPool [37]; (3) subgraph-based GNNs, including DropGIN [38], NestedGIN [26], and UnionSNN [27]; and (4) the Transformer-based method UGformer [39]. The graph kernel baselines fall into three categories: (1) classical substructure kernels, including SPGK [40], RWGK [41], GCGK [42], and WLSK [43]; (2) information-theoretic kernels, including JSSK [44], JTQK [45], and QJSK [46]; and (3) matching- and embedding-based kernels, including PMGK [47], EDBMK [48], and AEGK [49].

Table 2: Classification accuracy (% $\pm$ standard error) for comparisons with graph kernel methods.

	MUTAG	PTC_MR	D&D	PPIs	BAR31	SHOCK
JSSK	83.77±0.74	56.94±0.43	76.32±0.46	45.04±0.88	52.76±0.47	37.66±0.80
EDBMK	86.35	56.79	78.19	84.13	70.08	33.24
SPGK	83.38±0.81	55.52±0.46	78.45±0.26	59.04±0.44	55.73±0.44	37.88±0.93
WLSK	82.88±0.57	58.26±0.47	79.78±0.36	88.09±0.41	58.53±0.53	36.40±1.00
GCGK	82.04±0.39	55.41±0.59	74.70±0.30	46.61±0.47	22.96±0.65	26.63±0.63
JTQK	85.50±0.55	58.50±0.39	**79.89±0.32**	88.47±0.47	60.56±0.35	37.73±0.42
QJSK	82.72±0.44	56.70±0.49	77.68±0.31	65.61±0.77	30.80±0.61	40.60±0.92
RWGK	80.77±0.72	55.91±0.37	71.70±0.47	-	-	-
PMGK	86.67±0.60	60.22±0.86	77.78±0.48	-	-	-
AEGK	88.55±0.43	59.38±0.36	77.60±0.47	84.47±0.56	**73.66±0.57**	46.26±0.74
CSCN	**90.44±3.40**	**64.79±2.38**	78.69±1.42	**90.41±3.83**	64.33±11.01	**53.33±8.13**

	PROTEINS	IMDB_B	IMDB_M	RED_B_2K	RED_M_5K	RED_M_12K
SPGK	75.10±0.50	71.26±1.04	51.33±0.57	84.20±0.70	51.10±0.20	36.30±0.90
WLSK	73.52±0.43	71.88±0.77	49.50±0.49	76.56±0.30	50.54±0.16	39.03±0.29
JTQK	72.86±0.41	72.45±0.81	50.33±0.49	77.60±0.35	51.78±0.19	40.10±0.16
RWGK	74.20±0.40	67.94±0.77	46.72±0.30	72.73±0.39	-	-
PMGK	-	68.53±0.61	45.75±0.66	82.70±0.68	-	-
AEGK	74.08±0.32	-	-	88.28±0.20	51.16±0.17	40.67±0.77
CSCN	**75.57±0.40**	**74.50±0.95**	**51.53±0.56**	**90.90±0.62**	**56.11±0.87**	**46.71±1.24**

Implementation Details. In our experiments[2], we set the number of output channels for each layer to 128. The convolution window size is set to 3, covering three subtrees per window, with a stride of 1 along the ordered subtree sequence. We employ grid search to determine the optimal subtree layers (i.e., k) and the branching factor of the subtree (i.e., m) for each dataset. The search space for k is $\{2, 3, 4\}$, and for m is $\{2, 3, 4, 5\}$. An analysis of these hyperparameters is provided in Appendix C. The final classifier is an MLP consisting of two linear layers with a hidden dimension of 128. The ReLU activation function is applied. For the input node features, we use node labels when available and node degrees otherwise. All weights are randomly initialized from a normal distribution with mean 0 and variance 0.01. The model is optimized using the Adam optimizer. For ogbg-molhiv and ogbg-molbbbp, we follow the official OGB scaffold splits and report the average ROC-AUC and standard error over 10 runs. For other datasets, we report the average classification accuracy and standard error via 10-fold cross-validation.

4.2 Classification Performance

Tables 1 and 3 present the comparisons with deep learning methods, reporting classification accuracy in Table 1 and ROC-AUC in Table 3, both accompanied by standard errors. Table 2 further reports the classification accuracy and standard error for comparisons with graph kernel methods. The results of all baseline methods are taken from their original papers, with standard errors included when available. Overall, CSCN consistently achieves competitive performance, outperforming deep learning baselines on 11 out of 14 datasets and graph kernel baselines on 10 out of 12 datasets. These results demonstrate that CSCN offers a unified and effective framework that generalizes well across diverse graph classification domains.

The consistent improvement of CSCN across diverse domains stems from its ability to alleviate over-squashing and over-smoothing. Ordered subtree alignment establishes a controllable receptive field, and crossed subtree convolution enables information interactions across subtrees without deep message passing over the global graph topology. The receptive field analysis in Section 3 (Propositions 3.1 and 3.2) supports that CSCN avoids the exponential receptive field growth that aggravates over-squashing in standard MPNNs. We further evaluate the effects of CSCN on oversquashing and oversmoothing through two complementary empirical studies. The sensitivity analysis in Appendix D shows that CSCN better retains normalized sensitivity to distant node features in the original graph, providing empirical evidence for the alleviation of oversquashing. The MAD analysis in Appendix E measures the Mean Average Distance of representations at each layer and shows that CSCN pre-

[2]The source code is available at `https://github.com/ZJ-Han/CSCN`.

Table 3: Graph classification results (ROC-AUC ± standard error) on ogbg-molhiv and ogbg-molbbbp datasets.

Model	OGBG-MOLHIV	OGBG-MOLBBBP
GraphSAGE	70.37±0.42	60.78±2.43
GIN	70.60±2.56	64.10±1.05
GAT	70.60±1.78	63.30±0.53
UnionSNN	74.44±1.21	**68.28±1.47**
CSCN	**78.11±0.49**	66.57±0.68

Table 4: Ablation study on convolution components.

Variant	MUTAG	Tox21	NCI1	FRANK.	RED_M_12K
CSCN	**90.44**	**92.81**	**82.87**	**74.01**	**46.71**
w/o hier. op	89.39	92.07	80.56	73.08	46.63
w/o cross. op	86.17	91.99	81.80	73.11	45.34

Table 5: Ablation study on subtree construction strategies.

Variant	MUTAG	Tox21	NCI1	FRANK.	RED_M_12K
CSCN	**90.44**	**92.81**	**82.87**	**74.01**	**46.71**
Random pruning	85.70	92.01	77.42	72.17	45.39
Zero grafting	89.39	90.94	81.58	73.81	46.53

Table 6: Ablation study on node ordering methods.

Ordering	MUTAG	Tox21	NCI1	FRANK.	RED_M_12K
PageRank	**90.44**	**92.81**	**82.87**	**74.01**	**46.71**
Degree	84.62	92.18	81.48	73.28	46.55
Entropy	86.73	91.92	79.32	72.91	45.93
Random	83.60	89.92	71.48	68.80	44.19

serves more distinguishable representations than the MPNN baselines, supporting its effectiveness in alleviating oversmoothing.

4.3 Ablation Studies

To better understand the contribution of each component in CSCN, we conduct ablation studies on five representative datasets: MUTAG, Tox21, NCI1, FRANKENSTEIN, and RED_M_12K.

Convolution Components. As shown in Table 4, we evaluate two variants: removing the hierarchical operation (w/o hierarchical op) and removing the crossed subtree operation (w/o crossed subtree op). Both variants lead to performance drops across all datasets. These results verify that the two operations are complementary and both contribute to the effectiveness of CSCN.

Subtree Construction Strategies. As shown in Table 5, we compare two variants: replacing PageRank-based pruning with random pruning, and replacing multi-hop grafting with zero padding (zero grafting). Both of them lead to notable drops across all datasets. This suggests that expanding to multi-hop neighbors preserves richer local connectivity patterns than simply padding with zeros. Appendix F further compares degree- and entropy-based pruning and grafting.

Node Ordering Methods. As shown in Table 6, we compare PageRank ordering with three alternatives: degree-based ordering, entropy-based ordering, and random ordering. Degree-based and entropy-based orderings perform better than random but still fall behind PageRank, suggesting that PageRank captures more informative global topological information that is beneficial for constructing the ordered subtree sequence. Appendix G additionally evaluates stability under 100 random node relabelings per test graph.

4.4 Complexity Analysis

We analyze the computational complexity of each stage. In the ordered subtree alignment stage, PageRank computation costs $\mathcal{O}(N + M)$, where N is the number of nodes and M is the number of edges. Sorting nodes by PageRank scores costs $\mathcal{O}(N \log N)$. For subtree construction, each of the N subtrees contains $S_k = \frac{m^k - 1}{m-1}$ nodes. Since k and m are small constants, $S_k = \mathcal{O}(1)$. At each non-leaf node, neighbor sorting costs $\mathcal{O}(\Delta \log \Delta)$, where Δ is the maximum node degree. The total cost of this stage is $\mathcal{O}(N \cdot \Delta \log \Delta)$. In the crossed subtree convolution stage, $k - 1$ layers of convolution are performed. At the l-th layer, the number of convolution windows is $\frac{N^{(l)} - D}{s} + 1$. Each window performs $D \times (m + 1)$ matrix multiplications, each costing $\mathcal{O}(C^{(l+1)} \cdot C^{(l)} \cdot P^{(l)})$. Since D, m, and s are small constants, and $P^{(l)}$ decreases geometrically across layers, the total cost of this stage is $\mathcal{O}(N \cdot C^2)$, where C denotes the feature dimension. In the readout and prediction stage, global sum pooling costs $\mathcal{O}(N^{(k-1)} \cdot C)$ and the MLP costs $\mathcal{O}(C \cdot |\mathcal{Y}|)$. The total cost of this

stage is $\mathcal{O}(N \cdot C)$. Therefore, the overall complexity of CSCN is $\mathcal{O}(N \cdot \Delta \log \Delta + N \cdot C^2)$, which is linear with respect to the number of nodes N. We compare the complexity of CSCN with several representative baselines in Appendix H.

5 Conclusion

In this paper, we have introduced the Crossed Subtree Convolutional Network (CSCN), a novel framework for graph classification that addresses the over-squashing and over-smoothing issues in standard MPNNs. By decomposing a graph into an ordered sequence of aligned subtrees and applying crossed subtree convolution, our CSCN integrates hierarchical feature aggregation within each subtree with information interactions across subtrees, enabling effective modeling of long-range dependencies while capturing local information. Extensive experiments on multiple benchmark datasets show that CSCN achieves competitive performance. In future work, we will explore alternative subtree construction strategies to further improve structural fidelity and robustness, and develop more effective graph-level readout functions as well as stronger MLP designs for downstream prediction.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China under Grants 62676036, 62576371, 62576198 and 62622615, and the Open Project Foundation of Key Laboratory of Computation Intelligence and Chinese Information Processing of Ministry of Education and Key Laboratory of Data Intelligence and Cognitive Computing of Shanxi Province. The authors declare no competing interests.

References

[1] A. K. Jaiswal, S. Liu, T. Chen, Y. Ding, and Z. Wang, "Graph ladling: Shockingly simple parallel GNN training without intermediate communication," in *Proceedings of ICML*, vol. 202, 2023, pp. 14 679–14 690.

[2] L. Bai, L. Cui, M. Li, Y. Wang, and E. Hancock, "Qbmk: quantum-based matching kernels for un-attributed graphs," in *Proceedings of ICML*, 2024.

[3] L. Li, L. Xiang, G. Wang, Y. G. Wang, and K. Xia, "Path complex neural network for molecular property prediction," in *ICML 2024 Workshop on Geometry-grounded Representation Learning and Generative Modeling*, 2024.

[4] B. Armgaan, M. Dalmia, S. Medya, and S. Ranu, "Graphtrail: Translating gnn predictions into human-interpretable logical rules," *Proceedings of NeurIPS*, vol. 37, pp. 123 443–123 470, 2024.

[5] J. Liang, Z. Du, J. Liang, K. Yao, and F. Cao, "Long and short-range dependency graph structure learning framework on point cloud," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 12, pp. 14 975–14 989, 2023.

[6] F. Di Giovanni, L. Giusti, F. Barbero, G. Luise, P. Lio, and M. M. Bronstein, "On over-squashing in message passing neural networks: The impact of width, depth, and topology," in *Proceedings of ICML*, 2023, pp. 7865–7885.

[7] D. Shen, C. Qin, Q. Zhang, H. Zhu, and H. Xiong, "Handling over-smoothing and over-squashing in graph convolution with maximization operation," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[8] C. Qian, G. Rattan, F. Geerts, M. Niepert, and C. Morris, "Ordered subgraph aggregation networks," in *Advances in Neural Information Processing Systems*, vol. 35, 2022, pp. 21 030–21 045.

[9] L. Bai, L. Cui, Y. Jiao, L. Rossi, and E. R. Hancock, "Learning backtrackless aligned-spatial graph convolutional networks for graph classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 2, pp. 783–798, 2020.

[10] G. Bouritsas, F. Frasca, S. Zafeiriou, and M. M. Bronstein, "Improving graph neural network expressivity via subgraph isomorphism counting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 657–668, 2023.

[11] F. Frasca, B. Bevilacqua, M. Bronstein, and H. Maron, "Understanding and extending subgraph gnns by rethinking their symmetries," in *Proceedings of NeurIPS*, vol. 35, 2022, pp. 31 376–31 390.

[12] C. Zhou, X. Wang, and M. Zhang, "From relational pooling to subgraph gnns: A universal framework for more expressive graph neural networks," in *Proceedings of ICML*, 2023, pp. 42 742–42 768.

[13] D. Zeng, W. Liu, W. Chen, L. Zhou, M. Zhang, and H. Qu, "Substructure aware graph neural networks," in *Proceedings of AAAI*, vol. 37, no. 9, 2023, pp. 11 129–11 137.

[14] Z. Zhang, D. Chen, J. Wang, L. Bai, and E. R. Hancock, "Quantum-based subgraph convolutional neural networks," *Pattern Recognition*, vol. 88, pp. 38–49, 2019.

[15] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of ICLR*, 2017.

[16] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Proceedings of NeurIPS*, vol. 30, 2017.

[17] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio *et al.*, "Graph attention networks," *Proceedings of ICLR*, 2018.

[18] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *Proceedings of ICLR*, 2019.

[19] C. Nauck, R. Gorantla, M. Lindner, K. Schürholt, A. S. Mey, and F. Hellmann, "Dirac–bianconi graph neural networks-enabling long-range graph predictions," in *Proceedings of ICML*, 2024.

[20] Y. Sun, D. Zhu, Y. Wang, Y. Fu, and Z. Tian, "Gtc: Gnn-transformer co-contrastive learning for self-supervised heterogeneous graph representation," *Neural Networks*, vol. 181, p. 106645, 2025.

[21] G. Chen, L. Xia, and C. Huang, "Lightgnn: Simple graph neural network for recommendation," in *Proceedings of WSDM*, 2025, pp. 549–558.

[22] C. Huang, Y. Wang, Y. Jiang, M. Li, X. Huang, S. Wang, S. Pan, and C. Zhou, "Flow2gnn: Flexible two-way flow message passing for enhancing gnns beyond homophily," *IEEE Transactions on Cybernetics*, 2024.

[23] L. Cotta, C. Morris, and B. Ribeiro, "Reconstruction for powerful graph representations," *Proceedings of NeurIPS*, vol. 34, pp. 1713–1726, 2021.

[24] J. You, J. M. Gomes-Selman, R. Ying, and J. Leskovec, "Identity-aware graph neural networks," in *Proceedings of AAAI*, vol. 35, no. 12, 2021, pp. 10 737–10 745.

[25] K.-J. Chen, L. Liu, L. Jiang, and J. Chen, "Self-supervised dynamic graph representation learning via temporal subgraph contrast," *ACM Transactions on Knowledge Discovery from Data*, vol. 18, no. 1, pp. 1–20, 2023.

[26] M. Zhang and P. Li, "Nested graph neural networks," in *Proceedings of NeurIPS*, vol. 34, 2021, pp. 15 734–15 747.

[27] J. Xu, A. Zhang, Q. Bian, V. P. Dwivedi, and Y. Ke, "Union subgraph neural networks," in *Proceedings of AAAI*, vol. 38, no. 14, 2024, pp. 16 173–16 183.

[28] L. Zhao, W. Jin, L. Akoglu, and N. Shah, "From stars to subgraphs: Uplifting any GNN with local structure awareness," in *Proceedings of ICLR*, 2022.

[29] Z. Huang, Y. Wang, C. Li, and H. He, "Growing like a tree: Finding trunks from graph skeleton trees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 5, pp. 2838–2851, 2023.

[30] X. Qin, L. Bai, L. Cui, M. Li, H. Du, Y. Wang, and E. Hancock, "Ha-scn: Learning hierarchical aligned subtree convolutional networks for graph classification," in *Proceedings of IJCAI*, 2025, pp. 3245–3253.

[31] K. Kersting, N. M. Kriege, C. Morris, P. Mutzel, and M. Neumann, "Benchmark data sets for graph kernels," 2016.

[32] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *Proceedings of NeurIPS*, vol. 33, pp. 22 118–22 133, 2020.

[33] S. Biasotti, S. Marini, M. Mortara, G. Patane, M. Spagnuolo, and B. Falcidieno, "3d shape matching through topological structures," in *Proceedings of DGCI*, 2003, pp. 194–203.

[34] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proceedings of NeurIPS*, vol. 31, 2018.

[35] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *Proceedings of ICML*, vol. 97, 2019, pp. 3734–3743.

[36] Y. Liu, L. Cui, Y. Wang, and L. Bai, "Abdpool: Attention-based differentiable pooling," in *Proceedings of ICPR*, 2022, pp. 3021–3026.

[37] Z. Zhao, L. Bai, M. Li, L. Cui, H. Du, Y. Wang, and E. R. Hancock, "An end-to-end simple clustering hierarchical pooling operation for graph learning based on top-k node selection," in *Proceedings of IJCAI*, 2025, pp. 3670–3679.

[38] P. A. Papp, K. Martinkus, L. Faber, and R. Wattenhofer, "Dropgnn: Random dropouts increase the expressiveness of graph neural networks," in *Proceedings of NeurIPS*, vol. 34, 2021, pp. 21 997–22 009.

[39] D. Q. Nguyen, T. D. Nguyen, and D. Phung, "Universal graph transformer self-attention networks," in *Companion Proceedings of the Web Conference 2022*, 2022, p. 193–196.

[40] K. M. Borgwardt and H.-P. Kriegel, "Shortest-path kernels on graphs," in *Proceedings of ICDM*, 2005, p. 8.

[41] H. Kashima, K. Tsuda, and A. Inokuchi, "Marginalized kernels between labeled graphs," in *Proceedings of ICML*, 2003, pp. 321–328.

[42] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *Artificial intelligence and statistics*, 2009, pp. 488–495.

[43] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels." *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.

[44] L. Bai and E. R. Hancock, "Fast depth-based subgraph kernels for unattributed graphs," *Pattern Recognition*, vol. 50, pp. 233–245, 2016.

[45] L. Bai, L. Rossi, H. Bunke, and E. R. Hancock, "Attributed graph kernels using the jensen-tsallis q-differences," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2014, pp. 99–114.

[46] L. Bai, L. Rossi, A. Torsello, and E. R. Hancock, "A quantum jensen–shannon graph kernel for unattributed graphs," *Pattern Recognition*, vol. 48, no. 2, pp. 344–355, 2015.

[47] G. Nikolentzos, P. Meladianos, and M. Vazirgiannis, "Matching node embeddings for graph similarity," in *Proceedings of AAAI*, vol. 31, no. 1, 2017.

[48] L. Xu, L. Bai, X. Jiang, M. Tan, D. Zhang, and B. Luo, "Deep rényi entropy graph kernel," *Pattern Recognition*, vol. 111, p. 107668, 2021.

[49] L. Bai, L. Cui, M. Li, P. Ren, Y. Wang, L. Zhang, P. S. Yu, and E. R. Hancock, "Aegk: Aligned entropic graph kernels through continuous-time quantum walks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 37, no. 3, pp. 1064–1078, 2025.

A Proof of theorems

A.1 Proof of Proposition 3.1

Proposition 3.1. *Consider a standard MPNN with 1-hop message passing on a graph with maximum degree $\Delta > 2$. As the number of message passing layers l increases, the receptive field of each node grows at most exponentially with l.*

Proof. In standard MPNNs, message passing is performed over the global graph topology, and each layer expands the receptive field by at most one hop. Therefore, the receptive field of a node v at layer l exactly corresponds to its l-hop neighborhood. The receptive field $\mathcal{R}^{(l)}_{\text{MPNN}}(v)$ is defined as

$$\mathcal{R}^{(l)}_{\text{MPNN}}(v) = \{u \in V \mid d_G(u, v) \leq l\}, \tag{8}$$

where $d_G(u, v)$ denotes the shortest path distance between node u and node v in graph G.

Let Δ denote the maximum degree. For $\Delta > 2$, the size of the receptive field is bounded by

$$|\mathcal{R}^{(l)}_{\text{MPNN}}(v)| \leq 1 + \sum_{\ell=1}^{l} \Delta(\Delta-1)^{\ell-1} = 1 + \Delta \cdot \frac{(\Delta-1)^l - 1}{\Delta - 2} = \mathcal{O}((\Delta-1)^l). \tag{9}$$

Therefore, the number of nodes in the receptive field of a standard MPNN grows at most exponentially with the number of message passing layers l. □

A.2 Proof of Proposition 3.2

Proposition 3.2. *In CSCN, the receptive field of a convolution window along the ordered subtree sequence grows at most linearly with the number of crossed subtree convolution layers l when $s = 1$.*

Proof. We analyze the receptive field growth induced by crossed subtree convolution along the ordered subtree sequence. At the l-th crossed subtree convolution, the i-th convolution window aggregates the D inputs $\mathbf{X}^{(l-1)}_{(i-1)s+1}, \ldots, \mathbf{X}^{(l-1)}_{(i-1)s+D}$.

Let $\mathcal{S}^{(l)}(i)$ denote the indices of the input subtrees that can influence $\mathbf{H}^{(l)}_i$ after l crossed subtree convolution layers. Clearly, $\mathcal{S}^{(0)}(i) = \{i\}$. For $l \geq 1$, since $\mathbf{H}^{(l)}_i$ aggregates $\mathbf{X}^{(l-1)}_{(i-1)s+1}, \ldots, \mathbf{X}^{(l-1)}_{(i-1)s+D}$, the receptive field indices satisfy

$$\mathcal{S}^{(l)}(i) = \bigcup_{d=1}^{D} \mathcal{S}^{(l-1)}\big((i-1)s + d\big). \tag{10}$$

Eq. (10) shows that $\mathcal{S}^{(l)}(i)$ is a union of D consecutive sets from layer $(l-1)$. By induction, each $\mathcal{S}^{(l)}(i)$ forms a contiguous interval on the ordered subtree sequence.

Let $R_l = \max_i |\mathcal{S}^{(l)}(i)|$ denote the maximum number of input subtrees covered by any convolution window after l crossed subtree convolution layers. We have $R_0 = 1$. For $l \geq 1$, consider the D consecutive indices $(i-1)s+1, \ldots, (i-1)s+D$ at layer $(l-1)$. Shifting the index by 1 at layer $(l-1)$ shifts the window start by s at layer $(l-2)$, and tracing this back for $(l-1)$ layers yields a shift of at most $s^{\,l-1}$ positions on the original sequence. Therefore, relative to the first interval $\mathcal{S}^{(l-1)}((i-1)s+1)$, taking the union of D consecutive intervals can enlarge the covered range by at most $(D-1)s^{\,l-1}$. There are

$$R_l \leq R_{l-1} + (D-1)s^{\,l-1}. \tag{11}$$

Unrolling this recursion with $R_0 = 1$ gives

$$|\mathcal{S}^{(l)}(i)| \leq R_l \leq 1 + (D-1)\sum_{t=0}^{l-1} s^t = \begin{cases} 1 + l(D-1), & s = 1, \\ 1 + (D-1)\frac{s^l - 1}{s-1}, & s > 1. \end{cases} \tag{12}$$

Finally, each input $\mathbf{X}_j^{(0)}$ corresponds to a k-layer m-ary subtree. The number of nodes in the subtree is bounded by $S_k = 1 + m + \cdots + m^{k-1} = \frac{m^k - 1}{m-1}$. Therefore, the number of original graph nodes that can influence $\mathbf{H}_i^{(l)}$ satisfies

$$|\mathcal{R}_{\mathrm{CSCN}}^{(l)}(i)| \leq |\mathcal{S}^{(l)}(i)|\, S_k \leq \left(1 + (D-1)\sum_{t=0}^{l-1} s^t\right) S_k. \tag{13}$$

In particular, when $s = 1$,

$$|\mathcal{R}_{\mathrm{CSCN}}^{(l)}(i)| \leq (1 + l(D-1))\, S_k = \mathcal{O}(l). \tag{14}$$

Thus, the receptive field grows at most linearly with l, giving CSCN a controllable growth rate that is independent of the graph degree. □

B Dataset Statistics

All datasets used in this paper are publicly available. Here we report their statistics, including the average number of nodes, the average number of edges, the number of graphs, the number of graph classes, and the domain each dataset belongs to (bioinformatics, social networks, or computer vision). The detailed statistics of all datasets are summarized in Table 7.

Table 7: Statistical information of the benchmark datasets. FRANK. denotes FRANKENSTEIN; MOLHIV and MOLBBBP denote ogbg-molhiv and ogbg-molbbbp, respectively.

DATASETS	MUTAG	TOX21	PTC_MR	PROTEINS	D&D	NCI1	NCI109
AVG NODES	17.93	18.57	14.29	39.06	284.32	29.87	29.68
AVG EDGES	19.79	18.50	14.69	72.82	715.66	32.30	32.13
GRAPHS	188	8169	344	1113	1178	4110	4127
CLASSES	2	2	2	2	2	2	2
DOMAIN	BIO	BIO	BIO	BIO	BIO	BIO	BIO

DATASETS	FRANK.	PPIS	MOLHIV	MOLBBBP	IMDB_B	IMDB_M
AVG NODES	16.90	93.64	25.50	24.06	19.77	13.00
AVG EDGES	17.88	531.50	27.50	25.95	96.53	65.94
GRAPHS	4337	219	41127	2039	1000	1500
CLASSES	2	5	2	2	2	3
DOMAIN	BIO	BIO	BIO	BIO	SN	SN

DATASETS	RED_B_2K	RED_M_5K	RED_M_12K	SHOCK	BAR31
AVG NODES	429.63	508.52	391.41	13.16	95.43
AVG EDGES	497.75	594.87	456.89	12.17	94.59
GRAPHS	2000	4999	11929	150	300
CLASSES	2	5	11	10	15
DOMAIN	SN	SN	SN	CV	CV

C Hyperparameter Analysis

In CSCN, the subtree layer k and the branching factor m determine the structure of the aligned subtrees produced by ordered subtree alignment. We discuss their impacts below.

Subtree layer k. The subtree layer k controls the height of each node-rooted subtree and determines the number of crossed subtree convolution layers. When k is too small, each subtree covers only a limited neighborhood, and the model mainly captures highly local information. In this case, information interactions across subtrees become insufficient, leading to weaker modeling of long-range dependencies. When k is too large, the subtree receptive field expands substantially and more distant information is aggregated into fixed-dimensional representations aggravating the over-squashing.

Branching factor m. The branching factor m specifies the number of children per parent node, affecting the model's capacity to capture local connectivity patterns. When m is too large, grafting is invoked more frequently to satisfy the fixed m-ary constraint, bringing in higher-hop nodes to fill missing child slots. While this increases coverage, it may also inject less relevant distant nodes and thus add noise to local patterns. In contrast, when m is too small, pruning becomes more aggressive, discarding informative 1-hop neighbors and thereby losing important local structural details.

To verify the sensitivity of k and m, we report the classification accuracy on MUTAG, Tox21, NCI1, and FRANKENSTEIN with different k and m in Table 8. As shown in the results, different combinations of k and m lead to varying classification performance, with the optimal combination achieving the best accuracy. Nevertheless, even with suboptimal combinations, the accuracy of CSCN fluctuates only marginally, demonstrating the robustness of our model.

Table 8: Classification accuracy with different k and m.

k	m	MUTAG	Tox21	NCI1	FRANK.
2	2	88.89	91.26	78.73	72.82
2	3	88.33	91.41	80.95	73.39
2	4	87.78	91.70	81.97	73.35
2	5	87.25	91.75	81.53	73.46
3	2	89.33	92.36	80.97	73.99
3	3	**90.44**	92.63	**82.87**	**74.01**
3	4	88.30	92.80	81.17	72.38
3	5	89.39	**92.81**	79.76	72.61
4	2	89.42	92.28	82.09	73.48
4	3	87.31	92.47	79.90	72.47
4	4	88.30	92.64	79.25	71.71
4	5	88.33	92.77	78.76	71.55

D Over-squashing Analysis

To complement the over-squashing analysis, we examine how strongly nodes at different distances influence the representation at an anchor position. Following the Jacobian perspective on oversquashing [6], we define

$$J(u,v) = \left\| \frac{\partial \mathbf{z}_u}{\partial \mathbf{x}_v} \right\|_F, \tag{15}$$

where $\mathbf{x}_v$ is the original input feature vector of node v and $\mathbf{z}_u$ is the representation associated with anchor u immediately before graph readout. For GIN, GAT, and GraphSAGE, $\mathbf{z}_u$ is the final node representation at u. For CSCN, we associate each final window with the middle original subtree root in its receptive interval, treat this root as anchor u, and use the corresponding window representation as $\mathbf{z}_u$. All distances are computed as shortest path distances in the original graph.

For each run r, we first average the sensitivity over nodes at distance d from each anchor and then give equal weight to all sampled anchors,

$$S_r(d) = \frac{1}{|\mathcal{A}_r|} \sum_{u \in \mathcal{A}_r} \frac{1}{|\mathcal{V}_u(d)|} \sum_{v \in \mathcal{V}_u(d)} J(u,v), \qquad \mathcal{V}_u(d) = \{v : \mathrm{dist}_G(u,v) = d\}, \tag{16}$$

where $\mathcal{A}_r$ is the set of sampled anchors. We normalize each run by its sensitivity at distance one, $\widetilde{S}_r(d) = S_r(d)/S_r(1)$, and report the mean $\widetilde{S}(d)$ over ten independent runs with different random seeds. This normalization measures the relative change in sensitivity with distance.

For each dataset and run, we use the same 24 test graphs and four valid anchors per graph across all models, giving 96 anchors per model. To avoid computing the full Jacobian, we estimate its Frobenius norm using 16 Rademacher probes with entries taking values $+1$ or -1. All models use a hidden dimension of 32. GIN, GAT, and GraphSAGE use five layers, each operating on immediate neighbors, whereas CSCN uses $k = 3$. Since the architectures use different propagation units, we compare sensitivity at the same distances in the original graph rather than matching their layer counts.

As shown in Table 9, normalized sensitivity decreases with distance for all models, but generally decreases more slowly for CSCN. CSCN also retains the highest normalized sensitivity at $d = 5$ on all five datasets. These results indicate that CSCN better maintains sensitivity to distant node features relative to its sensitivity at distance one, providing empirical evidence that CSCN alleviates oversquashing compared with the MPNN baselines.

Table 9: Jacobian sensitivity normalized by the value at $d = 1$ and averaged over ten independent runs. Bold indicates the highest mean within each dataset and distance, excluding the normalized $d = 1$ column.

Dataset	Model	$d = 1$	$d = 2$	$d = 3$	$d = 4$	$d = 5$
MUTAG	CSCN	1.0000	**0.9508**	**0.8208**	**0.6817**	**0.5828**
	GIN	1.0000	0.6488	0.3595	0.1155	0.0271
	GAT	1.0000	0.5770	0.2559	0.0807	0.0214
	GraphSAGE	1.0000	0.4569	0.1829	0.0449	0.0102
Tox21	CSCN	1.0000	**0.9468**	**0.8289**	**0.6043**	**0.4859**
	GIN	1.0000	0.7320	0.4533	0.1780	0.0453
	GAT	1.0000	0.7419	0.4257	0.1988	0.0619
	GraphSAGE	1.0000	0.4811	0.1950	0.0483	0.0083
NCI1	CSCN	1.0000	**0.8958**	**0.7177**	**0.4790**	**0.3863**
	GIN	1.0000	0.6978	0.4755	0.1600	0.0589
	GAT	1.0000	0.6927	0.4345	0.1629	0.0621
	GraphSAGE	1.0000	0.6384	0.2976	0.0850	0.0159
FRANK.	CSCN	1.0000	**0.9616**	**0.8528**	**0.6256**	**0.5216**
	GIN	1.0000	0.6695	0.4713	0.1446	0.0687
	GAT	1.0000	0.6108	0.3052	0.0990	0.0255
	GraphSAGE	1.0000	0.5197	0.2020	0.0482	0.0075
RED_M_12K	CSCN	1.0000	0.3504	**0.1511**	**0.0460**	**0.0327**
	GIN	1.0000	0.2677	0.0726	0.0114	0.0017
	GAT	1.0000	**0.3999**	0.1464	0.0270	0.0037
	GraphSAGE	1.0000	0.2707	0.0560	0.0072	0.0006

E Over-smoothing Analysis

To verify that CSCN alleviates over-smoothing, we measure the Mean Average Distance (MAD) of node representations at each layer. A higher MAD indicates more distinguishable node representations, while a lower MAD suggests that representations are converging. We compare CSCN with three MPNN baselines (GIN, GAT, and GraphSAGE) on four datasets. All models use randomly initialized parameters. The results are reported in Table 10.

Across all four datasets, CSCN consistently maintains substantially higher MAD values than the three MPNN baselines at every layer. The MAD values of GIN, GAT, and GraphSAGE generally decrease as the number of layers increases, confirming the over-smoothing phenomenon in standard message passing. In contrast, the MAD of CSCN decreases more slowly and remains at a relatively high level. Beyond the absolute MAD values, CSCN also exhibits a slower relative decline across layers. For example, on MUTAG, CSCN retains 53% of its initial MAD at Layer 5, compared with 39% for GraphSAGE. On TOX21 and FRANKENSTEIN, the MAD of CSCN even slightly increases at deeper layers, suggesting that crossed subtree convolution introduces diverse information across subtrees and counteracts representation homogenization.

These results show that CSCN preserves more distinguishable node representations than standard MPNNs. Two factors contribute to this behavior. First, the hierarchical operation aggregates features within local units of each subtree and avoids repeated neighborhood aggregation over the entire graph. Second, the crossed subtree operation enables information interactions across different subtrees, bringing in diverse signals that help maintain representation distinguishability. Together with the receptive field analysis in Section 3, these results confirm that CSCN effectively alleviates both over-smoothing and over-squashing.

Table 10: MAD values of node representations at each layer on four datasets. Higher values indicate more distinguishable representations.

Dataset	Model	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
MUTAG	CSCN	**0.2441**	**0.1890**	**0.1791**	**0.1359**	**0.1296**
	GIN	0.0818	0.0874	0.0495	0.0245	0.0111
	GAT	0.0746	0.0298	0.0446	0.0191	0.0128
	GraphSAGE	0.1588	0.1405	0.1166	0.0807	0.0626
TOX21	CSCN	**0.1922**	**0.1748**	**0.1764**	**0.1927**	**0.1669**
	GIN	0.0285	0.0156	0.0067	0.0072	0.0084
	GAT	0.0077	0.0059	0.0041	0.0027	0.0042
	GraphSAGE	0.0256	0.0307	0.0279	0.0223	0.0262
NCI1	CSCN	**0.3087**	**0.2594**	**0.2340**	**0.2353**	**0.2092**
	GIN	0.0308	0.0101	0.0038	0.0011	0.0017
	GAT	0.0106	0.0077	0.0034	0.0029	0.0022
	GraphSAGE	0.0560	0.0477	0.0497	0.0474	0.0327
FRANK.	CSCN	**0.3112**	**0.2630**	**0.2329**	**0.2491**	**0.2342**
	GIN	0.0863	0.0489	0.0187	0.0080	0.0101
	GAT	0.0684	0.0418	0.0418	0.0233	0.0289
	GraphSAGE	0.1686	0.1912	0.1391	0.1032	0.0855

F Supplementary Analysis of Subtree Construction

Construction statistics. Table 11 reports construction statistics for $m = 3$, using the first expansion of each original node as a root as the unit of analysis. Pruning and grafting are the percentages of root expansions with more than m and fewer than m immediate neighbors, respectively. Retained PR is computed only over root expansions that trigger pruning. Across these expansions in all graphs of the dataset, we accumulate the PageRank mass of the retained m immediate neighbors with the highest scores and that of all candidate immediate neighbors. Retained PR is the ratio of the retained total to the candidate total, expressed as a percentage. The columns for two and three hops report the proportions of all grafted child positions at each distance from the parent in the original graph.

Overall, pruning occurs in only 0.03% to 10.59% of root expansions. Among the expansions that trigger pruning, the selected neighbors retain more than 80% of the total candidate PageRank mass. Although grafting is frequent, 95.21% to 99.55% of grafted positions are filled from neighborhoods at a distance of two hops. Consistently, the random pruning and zero grafting controls in Table 5 reduce average classification accuracy across the five datasets by 2.83 and 0.92 percentage points, respectively. Our design does not assume that discarded neighbors are irrelevant or that grafted nodes are equivalent to immediate neighbors. Instead, PageRank pruning gives priority to candidates with greater global topological importance, while grafting fills empty positions mainly with nearby context from multiple hops. Together, these construction statistics and ablation results support the effectiveness of the reconstruction strategy in combining selected local relations with nearby context for graph classification.

Table 11: Root expansion statistics for $m = 3$. All entries are percentages. The hop columns describe grafted child positions rather than all subtree edges.

Dataset	Pruning	Retained PR	Grafting	2 hops	3 hops
MUTAG	0.03%	81.77%	59.80%	99.55%	0.45%
Tox21	2.84%	82.81%	71.31%	95.21%	4.79%
NCI1	2.20%	83.49%	64.99%	95.62%	4.38%
FRANK.	1.58%	83.25%	67.26%	95.71%	4.29%
RED_M_12K	10.59%	83.61%	82.54%	96.13%	3.87%

Alternative selection rules. In addition to the random pruning and zero grafting controls in Table 5, we evaluate degree and entropy scores for candidate selection during pruning and grafting. For pruning, when a node has more than m immediate neighbors, we retain the m neighbors with the highest scores. For grafting, when fewer than m immediate neighbors are available, candidates from neighborhoods beyond one hop are ranked using the corresponding score to fill the remaining child positions. Table 12 reports the resulting classification accuracies.

Compared with random pruning, degree and entropy pruning each improve accuracy on four of the five datasets, but both remain below PageRank pruning on all five. For grafting, PageRank achieves the highest accuracy on four datasets, while entropy performs slightly better on RED_M_12K, reaching 46.81% compared with 46.71% for PageRank. Overall, PageRank provides the most consistent performance across the five datasets, supporting its use for candidate selection during both pruning and grafting.

Table 12: Classification accuracy with alternative pruning and grafting scores. All values are percentages. Bold indicates the highest reported mean in each column.

Variant	MUTAG	Tox21	NCI1	FRANK.	RED_M_12K
CSCN with PageRank	**90.44**	**92.81**	**82.87**	**74.01**	46.71
Degree pruning	89.44	92.58	79.88	70.69	45.62
Entropy pruning	87.28	92.40	77.08	72.60	46.46
Degree grafting	89.39	92.23	80.83	71.58	45.73
Entropy grafting	87.81	91.36	79.17	72.11	**46.81**

G Prediction Stability under Node Relabeling

PageRank scores remain unchanged for corresponding nodes under relabeling. When the scores are distinct, sorting yields a consistent relative ordering. However, tied or numerically indistinguishable scores may be ordered differently if node indices are used to resolve ties. This can lead to different subtree constructions after relabeling.

To assess the practical impact of this ambiguity, we generate 100 random node relabelings for each test graph and reconstruct the corresponding subtrees while keeping the trained model fixed. We measure prediction agreement as the percentage of relabelings that preserve the original predicted class. This metric reflects prediction consistency rather than classification accuracy. As shown in Table 13, PageRank achieves the highest prediction agreement among the evaluated ordering strategies on all five datasets.

Table 13: Prediction agreement under 100 random node relabelings per test graph. Higher values indicate greater stability.

Dataset	PageRank	Random	Degree	Entropy
MUTAG	**100.00%**	89.45%	88.79%	92.18%
Tox21	**99.26%**	95.68%	94.31%	93.15%
NCI1	**98.66%**	80.67%	81.70%	78.55%
FRANK.	**90.87%**	79.47%	77.18%	75.17%
RED_M_12K	**99.96%**	98.20%	84.48%	88.55%

H Complexity Comparison

Table 14 compares the full-model complexity of CSCN with several representative baselines. The comparison accounts for the end-to-end cost, covering structural preprocessing, message passing, and feature transformation. To ensure a fair comparison, all methods are analyzed under the same analytical framework, in which edge-wise aggregation and node-wise linear transformation are kept as separate terms wherever they appear. Following the notation in Section 3, N and M denote the numbers of nodes and edges, L the number of GNN layers, C the hidden dimension, and Δ the maximum node degree. Three additional symbols are used only in this section: K denotes the number

of clusters per pooling layer in DiffPool, N_{sub} the maximum rooted subgraph size in NestedGIN, and N_{union} is the maximum union subgraph size in UnionSNN.

Table 14: Full-model complexity comparison with representative baselines.

Method	Full-model Complexity
GIN	$\mathcal{O}\big(L(MC + NC^2)\big)$
DiffPool	$\mathcal{O}\big(L(MC + NC^2) + N^2(C + K)\big)$
SCHPool	$\mathcal{O}\big(L(MC + NC^2)\big)$
NestedGIN	$\mathcal{O}\big(NLN_{\text{sub}}(\Delta C + C^2) + L(MC + NC^2)\big)$
UnionSNN	$\mathcal{O}\big(N_{\text{union}}^3 M + L(MC + NC^2)\big)$
CSCN	$\mathcal{O}\big(N\Delta \log \Delta + NC^2\big)$

I Additional Information

I.1 Limitations

Despite the competitive performance of CSCN, several limitations remain, which also motivate our future work. First, the ordered subtree alignment relies on heuristic graph pruning and graph grafting guided by PageRank scores. Graph pruning may discard potentially informative 1-hop neighbors when $|\mathcal{N}_1(v_i)| > m$, while graph grafting may introduce higher-hop neighbors that are only weakly related to the root, injecting noise into local connectivity patterns. Moreover, the uniform k-layer m-ary structure forces every node-rooted subtree into a fixed template, which may not faithfully reflect the heterogeneous local topology of real-world graphs. These observations suggest the need for more flexible subtree construction strategies that better preserve structural fidelity and robustness. Second, the graph-level representation is obtained by a global sum pooling over the roots of the final ordered subtree sequence. Although effective, this readout treats all subtree roots equally and may overlook the varying importance of different subtrees, limiting the discriminative power of the graph-level representation. More effective graph-level readout functions are therefore desirable. Third, the final classifier is implemented as a two-layer MLP, whose limited capacity may become a bottleneck on datasets with complex class boundaries. Designing stronger MLP architectures tailored to the structured output of crossed subtree convolution is a promising direction. In addition, the subtree layer k and the branching factor m are dataset-dependent hyperparameters that currently require grid search, which we view as an orthogonal limitation worth addressing in future work.

I.2 Compute Resources

All experiments are conducted on a single server equipped with two Intel Xeon Silver 4310 CPUs and four NVIDIA RTX 3090 GPUs. Our model is implemented in PyTorch. Each individual training run is performed on a single GPU; multiple GPUs are used only to parallelize independent runs across different datasets, hyperparameter configurations, and cross-validation folds.

I.3 Impact Statements

This paper proposes the CSCN for graph classification, which decomposes a graph into an ordered sequence of structured subtrees and applies crossed subtree convolution to enable information interactions across subtrees, thereby alleviating over-squashing and over-smoothing in standard MPNNs. The proposed method is evaluated on publicly available benchmark datasets and is not tied to any specific deployment scenario. As the work is primarily methodological, we do not identify direct positive or negative societal impacts in the current study.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper proposes CSCN as a graph classification framework built around ordered subtree alignment and crossed subtree convolution, aiming to integrate local subtree information and cross-subtree interactions for effective graph classification.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We provide a discussion of the limitations of the work in Appendix I.1.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The paper provides the main assumptions and derivations for its theoretical analyses. Specifically, it states the assumptions for the receptive field analysis of standard MPNNs and CSCN in Appendix A.1 and A.2. The paper also provides a computational complexity analysis of CSCN in Section 4.4.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides sufficient information to reproduce the main experimental results. It describes each step of CSCN in detail, including ordered subtree alignment, crossed subtree convolution, graph-level readout, and prediction, with the corresponding formulations. We provide the implementation details in Section 4.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility.

In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The paper provides open access to the source code in Section 4.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the main training and test details needed to understand the experimental results. It reports the benchmark datasets, compared baselines, evaluation metrics, hyperparameter search space for the subtree depth k and branching factor m, node feature settings, weight initialization strategy, Adam optimizer, and MLP configuration. We provide the implementation details in Section 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports the main experimental results with standard errors in Section 4.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide the details of compute resources in the Appendix I.2

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics. The paper proposes a graph classification model and evaluates it on publicly available benchmark datasets. It does not involve human subjects, private or sensitive personal data, or high-risk deployment scenarios.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We provide a discussion of both potential positive societal impacts and negative societal impacts of the work in Appendix I.3.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release data or models that have a high risk for misuse. The work proposes a graph classification method and evaluates it on publicly available benchmark datasets, without involving pre-trained generative models, scraped datasets, or sensitive data.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper uses publicly available benchmark datasets, and the original sources of these datasets are properly credited through citations.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper does not contain new assets.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing experiments or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing experiments or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this paper does not involve LLMs as important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.