Cross-attentive Cohesive Subgraph Embedding to Mitigate Long-range Dependencies in GNNs

Tanvir Hossain[1] (✉), Muhammad Ifte Khairul Islam[1], Lilia Chebbah[1], Charles Fanning[2], and Esra Akbas[1]

[1] Department of Computer Science, Georgia State University, Atlanta, GA 30302, USA

[2] Department of Data Science and Analytics, Kennesaw State University, 1000 Chastain Road, Kennesaw, GA 30144, USA

{thossain5, mislam29, lchebbah1}@student.gsu.edu, cfannin8@students.kennesaw.edu, eakbas1@gsu.edu

Abstract. Graph neural networks (GNNs) have achieved strong performance across various real-world domains. Nevertheless, they suffer from oversquashing, where long-range information is distorted as it is compressed through limited message-passing pathways. This bottleneck limits their ability to capture essential global context and decreases their performance, particularly in dense and heterophilic regions of graphs. To address this issue, we propose a novel graph learning framework that enriches node embeddings via cross-attentive cohesive subgraph representations to mitigate the impact of excessive long-range dependencies. This framework enhances the node representation by emphasizing cohesive structure in long-range information but removing noisy or irrelevant connections. Additionally, It preserves essential global context without overloading the narrow bottlenecked channels, thereby enhancing message aggregation across vertices. Extensive experiments on multiple benchmark datasets demonstrate that our model achieves consistent improvements in classification accuracy over standard baseline methods.

Keywords: Oversquashing · Graph Decomposition · Subgraph Pooling · Cross-attention · Graph Representation Learning

1 Introduction

Oversquashing [2] is a critical limitation of GNNs that limits their applications in diverse graph domains. It occurs when GNNs capture local signals effectively but struggle to model long-range dependencies. For growing size of the n-hop neighborhood with respect to path distance leads to excessive message propagation through limited edges and narrow channels. Therefore, meaningful multi-hop information becomes bottlenecked, compressed or lost, thereby leading to oversquashing problem of GNNs.

Many studies have been proposed to understand and quantify oversquashing using node sensitivity [29], effective resistance [8], and commute time [14]. The

most common approach to alleviate this issue is rewiring, adding edges to reduce bottlenecks, but often causes high complexity [21,29,28,12]. Moreover, the majority of these methods rely on computationally intensive techniques, such as spectral decomposition [21] and optimal transport [28].

Real-world networks often preserve critical information within tightly connected dense regions. In presence of weak ties among regions, uniform message aggregation across these neighborhoods causes oversquashing. Dense regions dominate message passing, forcing long-range dependencies through narrow bottlenecks. Graph Decomposition offers a potential remedy by isolating structurally significant regions. However, the nondeterministic nature of traditional algorithms [9,22] often results in suboptimal graph partitions. To address this issue, cohesion-aware graph decomposition operates hierarchically, preserving subgraphs' semantic concerning the original network. Besides, these algorithms can be executed in parallel [10,13], causing negligible decomposition overhead.

Furthermore, applying GNNs to these subgraphs allows capturing considerable task-relevant information for vertices in downstream graph analytics. We conduct a pilot study (in section 2.2) that illustrates cohesion-sensitive graph partition shortens long-range dependencies while preserving homophily within subgraphs. Furthermore, pooling over these subgraphs enhances homophilic characteristics. Preserving homophily facilitates effective representation learning in graph-based models [17]. Nonetheless, focusing solely on the densely connected regions may compromise long-range connectivity. Cross-attention [31] addresses this limitation by enabling interactions between distant nodes, connecting relevant representations even when they are positionally unaligned.

Our Work. We propose a novel graph learning framework, that utilizes **C**ross-**a**ttentive **Co**hesive **S**ubgraph **E**mbeddings (`CaCoSE`) to alleviate oversquashing issue in GNNs. This `CaCoSE` framework begins by a leveraging k-core [27] algorithm to compute the edges' cohesiveness scores based on their membership in k-level regions. Using these scores, it constructs edge-induced subgraphs guided by structural closure and feed them into GNNs to learn node representations.

After getting nodes representations within each subgraph, it applies graph pooling, followed by cross-subgraph attention mechanism. By graph pooling, `CaCoSE` selectively filters out noisy or irrelevant connections while preserving task-relevant structures and critical global information. Moreover, the attention module acts a learnable interfaces between vertices to capture their mutual relations that provides more expressivity in representations.

Finally, the enriched subgraphs' representations are combined with the embeddings of nodes present in those subgraphs to capture long range global information. This enhances message aggregation and maintains essential local and global connectivity in representations, even in large-scale heterophilic networks. Throughout the paper, we refer to the `CaCoSE` framework as our model for simplicity. The contributions of our model is as follows:

- The pilot study opens up a new viewpoint to understand the reasons for oversquashing prior to the experiment. Closure-guided k-core decomposition benefits cohesive awareness among vertices, which removes noisy pathways

and provides essential locality for GNNs operations. Moreover, pooling enhances network's homophily to attain meaningful subgraph representations.
– The cross-subgraph attention mechanism encodes essential global relations among all subgraphs. Merging node representations with enriched subgraph embeddings maintains both local and global connectivity, helping to alleviate the oversquahing problem in GNNs.
– In an extensive experiment on multiple datasets, `CaCoSE` outperforms the standard baselines in both node **(NC)** and graph classification **(GC)** tasks.

2 Methodology

In this section, we first describe the primary components of `CaCoSE`. Then, we present a pilot study that motivates the design choices of our model. Finally, we provide details of `CaCoSE` architecture and describe the representation learning procedure to address the oversquashing problem in GNNs.

2.1 Preliminaries

Oversquashing. One of the major drawbacks of GNNs that occurs when information is severely bottlenecked due to its failure in useful message passing. According to [29], when a node t that is connected with another node s by an r-hop distance, the Jacobian operation $\delta h_t^{(r+1)}/\delta x_s$ denotes the change of the feature vector x_s's impact on the $(r+1)st$ layer's output of $h_t^{(r+1)}$. The quantification is observed from the absolute value $|\delta h_t^{(r+1)}/\delta x_s|$ of the Jacobian, where a negligible value indicates limited information propagation or oversquashing. In addition, as presented in [8], oversquashing can be associated with effective resistance between two node pairs. As low as the effective resistance is, the two nodes have more influence on each other during GNNs' operations.

Problem Statement. Our study addresses the challenge of oversquashing by introducing a closure-aware cohesive subgraph decomposition framework along with graph pooling and a cross-subgraph attention mechanism. The primary objective is to enhance graph component features' expressivity in downstream analytics tasks. In a formal sense, a graph is denoted as $G = (V, E, X)$, where V is the vertex set, E is the edge set and $X \in \mathbb{R}^{N_v \times d}$ represents the initial feature vector of the vertices. The number of nodes is $|V| = N_v$ and d denotes the dimension of the nodes' features. The graph will be partitioned into cohesion-centric subgraphs, such as $S = \{S_1, S_2, ..S_{k_{max}}\}$ where each $S_k \subseteq G$ corresponds to cohesion level $k = \{1, 2, .., k_{max}\}$. We apply the proposed `CaCoSE` model to these subgraphs. For node classification, the model trained on $D_v = (G, \mathbb{X}, Y_v)$ to learn the mapping $f_v : \mathbb{X} \rightarrow Y_v$. Besides, for graph classification, it is trained on $D_G = (\mathbb{G}, Y_G)$ to learn $f_G : \mathbb{G} \rightarrow Y_G$. In both cases, the learned functions leverage the reduced graph complexity achieved through the `CaCoSE` framework.

Graph Neural Network. Graph neural networks (GNNs) embedded structural information in the graph via message propagation into node and graph representations. In graph convolution network (GCN), message propagation rule is defined as:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}\Theta^{(l)}) \tag{1}$$

where, $\tilde{A} = A + I_{N_v}$ denotes the adjacency matrix with the self-connections, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ degree matrix and $\Theta \in \mathbb{R}^{d\times h}$ learnable weight matrix for h-dimensional embeddings. $\sigma(.)$ indicates the $ReLU(x) = max(0, x)$ activation. $H^{(l)} \in \mathbb{R}^{N_v \times h}$ denotes the l^{th} layer's nodes' embeddings matrix for h-dimension where $H^{(0)} = X$.

Self-Attention Graph Pooling. Graph pooling captures the entire graph's representation through compression. SAGPool [24] facilitates self-attentive graph learning through selecting impactful neighbors of vertices. It utilizes the (GCN) to measure the self-attention scores.

$$Attn_v = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta_{att}) \tag{2}$$

Where $Attn_v \in \mathbb{R}^{N_v \times 1}$ is calculated from $\tilde{A} = A + I_N$, X and a learnable parameters' matrix Θ_{att}. Then utilizing the pooling ratio (PR) it selects the top-k node indices, $idx = topk(Attn_v, \lceil (PR)N_v \rceil)$. Next, only considers the top-k vertices and their connections, $Attn_{mask} = Attn_v[idx]$. After the computation of *top*-k node indexes, the graph pooling of nodes' features is measured as $X_{out} = X_{idx,:} \odot Attn_{mask}$ and $A_{out} = A_{idx,idx}$. Finally, through a READOUT function the representation of the entire graph is computed as $Z_G = READOUT(X_{out})$. Here, READOUT is the global pooling function: sum, mean, or other advanced learnable aggregator.

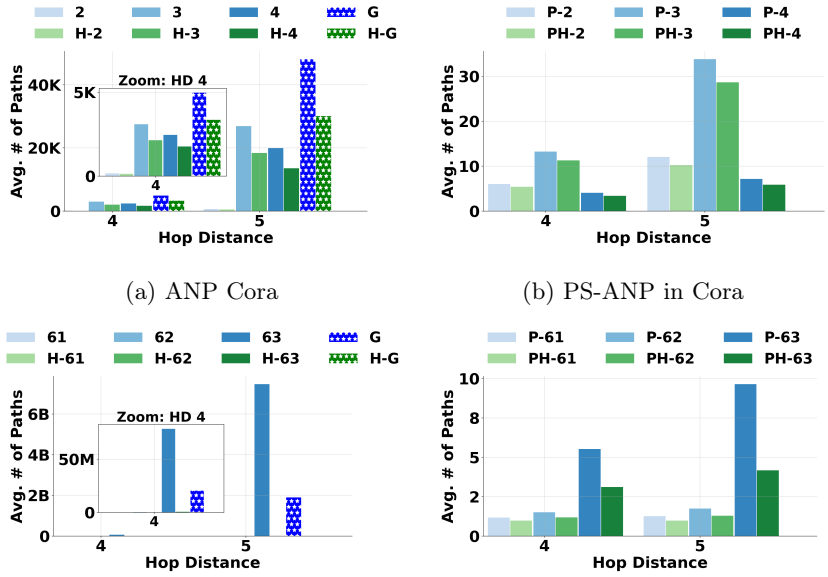
k-core Decomposition. Cohesive subgraph decomposition is instrumental to determine the intended graph regions. The decomposition is performed by iteratively peeling away nodes whose degrees fall below the specified threshold: nodes with degree less than k are removed.

In particular, if the graph has no isolated node, the original graph G can be denoted as 1-core. As indicated by hierarchy, $G_{k_{max}} \subseteq \ldots G_3 \subseteq G_2 \subseteq G_1$, where, $k \in \{1, 2, \ldots k_{max}\}$. A k-core subgraph is defined as -

Definition 1. *(k-**core**): For a given $k \geq 1$, the k-core subgraph is represented from the graph G when each node in the subgraph belongs to the same (k) or more neighbors such that $|N(v)| \geq k$, where $N(v)$ represents the number of neighbors of node $v \in G_k$.*

2.2 Pilot Study

Oversquashing occurs because GNNs struggle to learn long-range dependencies due to the bottleneck effects in graph structure [2]. Cohesion-aware graph de-



(a) ANP Cora

(b) PS-ANP in Cora

(c) ANP in Chameleon

(d) PS-ANP in Chameleon

Fig. 1: **Pilot Study**. Average number (#) of paths (ANP) per node for $n \in \{4, 5\}$ hop distances (HD) in Cora and Chameleon (Chm) datasets. Blue bars denote the original graph (G), its cores (k) and their (PS)-pooled subgraphs ($P - k$); green bars present their homophilic counterparts ($H - G$, $H - k$ and $PH - k$). From right to left, the deep (blue & green) bar pairs (Figs. 1(a) and 1(c)) with hatch ($'*'$) present the ANP of original graph and its homophilic subgraph respectively; in other cases, more deeper bar color denotes more denser subgraph. K - thousand and, M|B - m|billion.

composition not only reduces burdens on bottlenecked channels, but also preserves essential structural properties. However, in heterophilic networks, selecting task-relevant neighbors remains challenging due to cross-class mixing. This pilot study demonstrates that applying k-core decomposition, followed by SAGPooling, preserves essential class-consistency among vertices even in heterophilic graphs. That enables GNNs to aggregate more reliable signals, producing stable node representations.

Definition 2. *(**Edge score**): In the context of k-**core**, for a graph $G = (V, E)$ and $k \geq 1$, an edge $(u, v) \in E$ can appear in multiple k-core subgraphs. The score of an edge is assigned as*

$$C(u, v) = max\{k \mid (u, v) \in G_k\} \tag{3}$$

$C(u, v)$ is computed for the highest k-valued subgraph where (u, v) exists.

Fig. 1 presents a pilot study on Cora (homophilic) and Chameleon (heterophilic) datasets to examine how well subgraph decomposition preserves assortativity and disassortativity. According to definitions 1 and 2, we first compute edge scores and extract edge-induced dense subgraphs for top-three $k-$core values. Form Cora $k \in \{2, 3, 4\}$ and from Chameleon $k \in \{61, 62, 63\}$. Next, we compute the average number of paths (ANP) for vertices at hop distances $n \in \{4, 5\}$ across original graphs, subgraphs and their homophilic counterparts $H = \{(u, v) \in E : y_u = y_v\}$. The measures are presented in bar graphs.

According to Figs. 1(a) and 1(c), the subgraphs in both datasets retain the homophilic and heterophilic ($y_u \neq y_v$) properties of their original graphs. While this preservation is often sufficient for homophilic networks to maintain task-relevant features, it becomes more challenging for heterophilic graphs like Chameleon. To extend the study, we apply SAGPool to each subgraph and repeat the ANP evaluation on pooled subgraphs (PS-ANP). Interestingly, in the Chameleon dataset (Fig. 1(d)), the pooled subgraphs exhibit a higher ratio of homophilic paths to the total average number of paths per node compared to the initial evaluation. That facilitates effective representation learning in graph models [17]. Besides, it increases the ratio for the homophilic dataset.

2.3 Model Architecture

In Figure 2, the model architecture is comprised of four modules. **Module (1)** works as the graph pre-processor, while **module (2)** trains the decomposed subgraphs with GNNs. On the other hand, **module (3)** functions as a feature post-processor, whereas **module (4)** operates for the final evaluation.

(1) Cohesive Subgraph Decomposition. Density-informed subgraphs, called cohesive subgraphs, are crucial for effective graph structure learning where each vertex gains sufficient connectivity in a particular region. Hence, learning the representation of vertices with these cohesive regions can capture adequate proximal insight for them. The k-core is one of the popular algorithms to obtain cohesion-focused subgraphs [11].

Theorem 1. *($CaEF$: **Closure-aware Edge Filtration**): Let G_k denote the k-core of $G = (V, E)$. For an edge $e = (u, v) \in G_k$, where $N(u)$ and $N(v)$ are the neighbor set of node u and v, its triadic support is defined as*

$$S(u, v) = |N(u) \cap N(v)| \tag{4}$$

For $k \geq \delta$, if $S(u, v) = 0$ then (u, v) is removed from G_k and reassigned to previous core $C(u, v) = (k - 1)$, where δ denotes the edge filtering threshold.

In this module, to partition a graph into cohesive subgraphs, `CaCoSE` employs the k-core decomposition algorithm in [6]. Initially, it calculates the edges coreness following the definition (2). For example, edge (v_4, v_6) is a part of both the 1-core and the 2-core subgraphs. As $(2 > 1)$, the coreness score $C(v_4, v_6) = 2$. Similarly, $(v_6, v_7) \in G_1 \cap G_2$, hence, $C(v_6, v_7) = 2$.

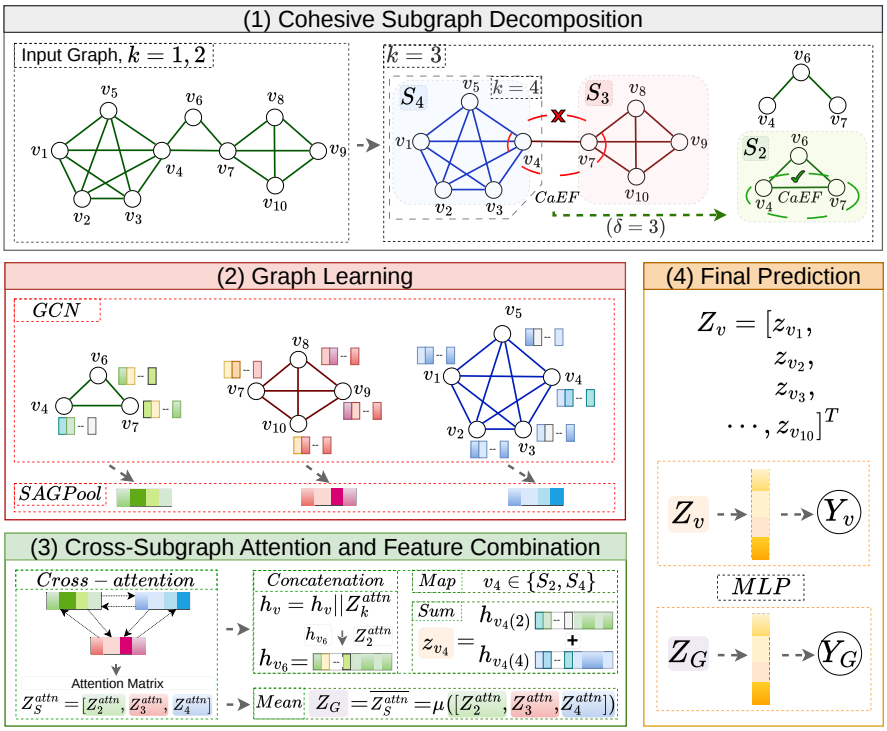


Fig. 2: Architecture of `CaCoSE` ($\delta = 3$).

Simultaneously, we examine the existence of narrow edges in k-core subgraphs. According to theorem (1), such edges are removed from k-core subgraph and reassigned to the $(k-1)$-core subgraph. For example, in Figure 2 when $k = 3$, the edge (v_4, v_7) has no support. It may act as a noisy channel during neighborhood aggregation in GNNs. Hence, for $\delta = 3$, (v_4, v_7) is pruned from G_3 and assigned to G_2, then $C(v_4, v_7) = 2$. After assigning scores to each of the edges, the graph is decomposed into subgraphs corresponding to the same edge-score groups -

$$S = \{S_k\}_{k=1}^{k_{\max}}, \quad S_k = (V_S, E_S), \\ E_S = \{(u, v) \in E \mid C(u, v) = k\} \tag{5}$$

For example, in Fig. 2, the final set of decomposed subgraphs is $S = \{S_2, S_3, S_4\}$. Note that $S_k \neq G_k$, e.g $S_3 \neq G_3$. Although many nodes overlap across different subgraphs, they consistently carry the same weighted edges within each subgraph. Hence, these score-based partitions avoid structural inconsistencies and preserve meaningful subgraphs for graph operations.

(2) Graph Learning. In this part, `CaCoSE`, particularly aims to embed the topological information of all decomposed subgraphs into node representations.

For this purpose, it applies the GCN to each subgraph and obtains the representation of each vertex $h_v \in H_k$ separately.

$$H_k = \mathrm{GCN}_k(S_k) \tag{6}$$

$H_k \in \mathbb{R}^{N_{v_S} \times h}$ presents the node feature matrix, where $N_{v_S} = |S_k|$ denotes the number of vertices in the subgraph. Nevertheless, Just learning within subgraph get local cohesive information but loss global information. To get the global information of each subgraph, `CaCoSE` attains subgraph embeddings via employing the self-attention graph pooling ($SAGPool$).

$$Z_k = \mathrm{SAGPool}_k(S_k, H_k) \tag{7}$$

This pooling encodes essential global structural information of the subgraphs through relevant neighborhood selection, $Z_k \in \mathbb{R}^{d_S}$, where, d_S presents the pooling dimension. It effectively filters out candidate nodes' task-irrelevant or noisy neighbors even from highly heterophilic networks. Combining this subgraph embeddings with node embeddings provides crucial global information to vertices.

(3) Cross-Subgraph Attention & Feature Combination. Cohesion-centric partitioning reduces bottleneck loads but causes loss of long-range dependency among vertices. Additionally, the subgraphs' embeddings via pooling capture only local structural information while overlooking other subgraphs. Hence, a way to recover global connectivity across partitions is required. The attention mechanism [31] enables the modeling of long-range dependencies through sequential learning. Toward this goal, cross-subgraph attention facilitates communication and information propagation among distinct sequence subgraphs.

In this stage, `CaCoSE` employs cohesion-sensitive subgraph attention to update the subgraph embeddings. Each subgraph embedding is processed through cross-attention to encode mutual information across regions. Attention [31] help in capturing essential awareness among entities. The pooled subgraphs' embeddings are represented as, $Z_S = \left[Z_1, Z_2 \dots Z_{k_{max}}\right]^\top \in \mathbb{R}^{N_S \times d_S}$, where $N_S = |S|$ denotes the number of subgraphs and d_S as feature dimension. From the subgraphs feature matrix, the query, key, and value matrices are computed as $Q = Z_S W_Q,\ K = Z_S W_K,\ V = Z_S W_V$. Here, W_Q, W_K, W_V are learnable weight matrices. Then, the attention scores are measured and subgraphs representations are updated as:

$$Attn_S = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_S}}\right), \quad Z_S^{attn} = Attn_S V \tag{8}$$

$Z_S^{attn} = \left[Z_1^{attn}, Z_2^{attn} \cdots Z_{k_{max}}^{attn}\right]^T \in \mathbb{R}^{N_S \times d_S}$ presents the updated representation of a subgraph, incorporating other subgraphs' attentions where $Attn_S \in \mathbb{R}^{N_S \times N_S}$ denotes subgraphs' attention matrix. Such as, in Fig 2, `CaCoSE` computed the cross-attentive matrix as, $Z_S^{attn} = \left[Z_2^{attn}, Z_3^{attn}, Z_4^{attn}\right]^T$. It is noteworthy that the procedure of the attention mechanism seems like complete graph

learning. Meanwhile, cohesion-sensitive partitions produce a smaller number of subgraphs; therefore, cross-subgraph attention adds only negligible computational overhead. The final graph representation is derived by taking the mean (μ) of the cross-attentive subgraphs embeddings, preserving inter-subgraph relational information.

$$Z_G = \overline{Z_S^{attn}} = \mu(\left[Z_1^{attn}, Z_2^{attn} \cdots Z_{k_{max}}^{attn}\right]) \tag{9}$$

`CaCoSE` obtains informative nodes' embeddings by concatenating ($\|$) each subgraph's attentive features with its vertices representations.

$$h_v = h_v \parallel Z_k^{attn}, \quad v \in S_k \tag{10}$$

If a node belongs to multiple subgraphs, a mapping (Map) function matches its occurrences, and all of its representations are summed. Such as, for $v \in \{S_l, S_m, S_n\}$ and $S_l, S_m, S_n \subseteq S$, the final node representations is computed as:

$$z_v = \sum_{k \in \{l,m,n\}} h_{v(k)} \tag{11}$$

For instance, in Fig 2, node v_4 is part of both S_2 and S_4. Hence, the final representations is summed (Sum) as: $z_{v_4} = h_{v_4(2)} + h_{v_4(4)}$.

(4) Final Prediction. Decomposition reduces excessive information processing through bottlenecked channels in graph. Subsequently, selection-based learning provides compact and meaningful subgraph representations. Following that, cross-subgraph attention alleviates long-hop dependency by modeling interactions among vertices across different subgraphs. Finally, the processed nodes' ($Z_v = \left[z_{v_1}, z_{v_2} \cdots z_{v_{N_v}}\right]^T$) and graphs' ($Z_G$) representations are passed through a multilayer perceptron (MLP) to produce final predictions Y_v (NC) and Y_G (GC). Experimental results validate the model's ability in mitigating oversquashing and enhance performance on downstream inference. The detailed examination of the `CaCoSE` algorithm, its complexity, theoretical justification, and scalability analysis are available in Appendix A.

3 Related Works

Graph Decompositions. Numerous graph decomposition algorithms are applied to improve GNNs' effectiveness. At the earlier stage of GNNs, spectral clustering [35,7] methods were much more popular along with the hierarchical [34] and modularity-based [30] clustering models. However, due to expensive eigenvalue decomposition and clustering assignment steps, these methods encounter relatively higher time complexity.

Cohesion-sensitive Decompositions. These algorithms are mostly applied to determine the interconnectedness between nodes in multiple network regions for solving different domain problems: graph compression [1], high-performance

computing [26], analyzing social networks [11], etc. Only a few methods utilize these algorithms to enhance the effectiveness of GNNs. TGS [20] utilizes the k-truss algorithm for edge scoring and sparsifies noisy edges to mitigate oversmoothing in GNNs. Another model, CTAug [33], utilizes k-truss and k-core algorithms to provide cohesive subgraph awareness and improve graph contrastive learning (GCL). Nonetheless, due to repetitive edge filtering for sparsification and GCL's resource-expensive nature, both models require longer runtime.

Oversquashing. Prior work [2] illustrates that due to bottlenecks in the graph, long-range neighborhood signals are distorted, which downgrades GNNs' performance in downstream graph learning tasks. SDRF [29] proposed a curvature-based graph rewriting to detect edges responsible for the information bottleneck. FoSR [21] applies systematic edge addition operations in graphs in accordance with spectral expansion, while BORF [28] demonstrates the impacts of curvature signs for oversquashing and oversmoothing. However, due to dependency on intermediate layer output and subgraph matching with optimal transport, these models show inconsistency in large-scale graphs' operations. Another approach, GTR [8], uses effective resistance and a repeated edge addition technique to reduce the impact of oversquashing in GNN. However, this method focuses on the entire graph's information for edge rewiring, which increases complexity in executing larger graphs. Recently, GOKU [25] utilizes spectrum-preserving graph rewiring, first densifying the network and subsequently applying structure-aware graph sparsification. In contrast, LASER [5] adopts locality aware sequential rewiring while considering multiple graph snapshots. GraphViT [19] leverages the Vision Transformers combined with MLP mixing mechanism to model long-range dependencies in graphs. On the other hand, LRGB [15] addresses the limitations of benchmark GNNs and Transformers in analyzing long-range vertex interactions. However, both approaches primarily focus on relatively ordered and structurally regular datasets. In contrast, our work concentrates on reducing oversquashing in complex, irregular real-world and social networks.

4 Experiment Results

This section covers `CaCoSE`'s experiment outcomes' information. First, it discusses the datasets and baselines. Then, it describes the experimental setup and, finally, presents the model's results compared to other baselines with various experiments. **Note that, Prior works [15,2] experiment on synthetic datasets (tree neighbors-match, ring transfer, etc.) are unsuitable for k-core due to their fixed structure. Hence, our experiment mostly focuses on real-world complex networks.**

4.1 Experimental Settings

Datasets and Baselines. Our `CaCoSE` model is evaluated on 8 datasets (D_v) for the node classification task. Where five datasets are homophilic: **Cora**, **C**ite**Seer**, **Co**Author **CS**, **Am**azon **Comp**uters and **Am**azon **P**hot**o**s. Besides,

Table 1: Node Classification Accuracy Comparison. Highest accuracy is **bolded**, second-best is underlined. OOM: Out of Memory in execution.

M / D_v	Cora	C.Seer	Co. CS	Am. Cmp	Am. Pto	Texas	Cham.	Squir.
GCN	84.24	69.10	88.86	87.77	92.12	52.36	63.32	48.79
GAT	**85.00**	67.94	87.13	88.41	92.01	52.37	61.70	46.18
SAGE	83.60	67.76	88.42	87.22	91.58	56.05	62.64	47.15
LRGB	71.02	57.58	59.50	70.34	74.04	57.36	46.93	30.49
BORF	83.68	67.08	90.52	89.79	91.93	54.21	60.24	OOM
FOSR	83.66	67.24	90.44	89.82	91.83	52.37	60.31	40.19
SDRF	84.04	67.42	90.56	86.79	91.71	52.63	60.74	43.02
GTR	84.07	69.15	88.87	85.37	91.76	52.37	63.78	48.67
DR	44.35	24.90	72.50	71.43	79.86	**69.45**	27.87	23.85
LASER	75.77	64.26	76.90	38.63	67.36	32.63	41.90	25.87
GOKU	82.66	66.08	90.04	OOM	92.51	37.10	65.90	46.74
`CaCoSE`	**85.00**	**69.42**	**90.73**	**90.43**	**92.98**	54.47	**68.99**	**58.86**

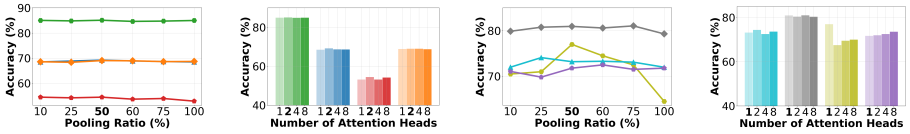
Table 2: Comparison of different methods on graph classification benchmarks.

M/ D_G	MUTAG	IMDB-B	IMDB-M	RDT-B	COLLAB	PROTEINS
SDRF	74.53	62.90	41.53	85.40	70.22	66.88
FOSR	75.89	60.40	37.33	83.25	69.85	66.70
BORF	64.00	60.82	38.20	84.92	OOM	68.41
GTR	76.00	70.20	45.33	**89.65**	68.02	71.52
DR	71.00	53.60	35.60	73.60	55.54	**72.23**
LASER	68.00	67.30	45.40	81.70	71.32	71.07
GOKU	74.50	64.80	42.40	80.55	69.02	70.80
`CaCoSE`	**76.99**	**73.20**	**49.14**	85.70	**80.95**	71.79

three datasets are heterophilic: **Cham**eleon, **Squir**rel and **Texas**. In case of graph classification tasks we experiment on six different datasets where four from social network domain: **IMDB-B**INARY, **IMDB-M**ULTI, **COLLAB** and **R**ED**DIT-B**INARY. Other two from the biomedical domian: **MUTAG** and **PROT**EINS.

Our model is compared with the 11 standard methods for the node classification **(NC)** tasks: GCN [23], GAT [32], SAGE [18], LRGB [15], BORF [28], FOSR [21], SDRF [29], GTR [8], DR [3], LASER [5] and GOKU [25]. For graph classification **(GC)**, we compare `CaCoSE` with seven methods (BORF, FOSR, SDRF, GTR, DR, LASER and GOKU), excluding the first four baselines.

Implementation Details. We evaluate our model by running baselines' codes for a fair comparison. Except for Cora (1208, 500, 500) and CiteSeer (1812, 500, 500), we split other datasets into 48%, 32%, and 20% for training, validation, and testing, respectively. In the case of graph classification (GC) for all datasets, the ratio is (80% : 10% : 10%). Each model runs for up to 250 iterations (NC) and 100 iterations (GC), with early stopping after 50 and 25 consecutive epochs without validation improvement, respectively. Nodes' features are initialized with $1-hot$ encoding and hidden dimension in GNNs is set as 128. For a learning



(a) NC (PR) (b) NC (NH) (c) GC (PR) (d) GC (NH)

Fig. 3: **Sensitivity Analysis**. Varying Pooling Ratios (PR) and Numbers of Heads (NH). `CaCoSE`'s settings for Node Classification (PR = 50% and NH = 2) and for Graph Classification (PR = 50% and NH = 1) are highlighted in bold. For NC datasets, ■ Cora ■ CiteSeer ■ Texas ■ Chameleon. For GC datasets ■ IMDB-B ■ COLLAB ■ MUTAG ■ PROTEINS.

Table 3: Performance comparison across different values of δ. In `CaCoSE`, $\delta = 3$. Bold fonts indicate better accuracy than `CaCoSE`.

D_v / δ	**3**	4	5	6	7	D_G / δ	**3**	4	5	6	7
Cora	85.00	84.88	**85.22**	84.96	85.10	RDT-B	85.70	83.50	83.20	83.15	81.80
C.Seer	69.42	69.12	68.90	68.96	69.00	IMDB-B	73.20	71.60	**73.40**	72.70	**73.79**
Cham.	68.99	68.73	68.46	68.58	68.93	PROT.	71.79	**72.77**	**72.05**	71.52	**72.41**

rate of $2.5e-3$, `CaCoSE` utilizes l_2 regularization with a weight decay of $1e-4$. The pooling ratio is set as 0.5 in both cases while the number of heads are set as 2 (NC) and 1 (GC). Besides, we set the $CaEF$ threshold δ as 3 for both tasks. Finally, we split each dataset using 10 different seeds and report the mean accuracy as the final result.

4.2 Result Analysis

Node Classification and Graph Classification. Table 1 demonstrates the comparison of our model with the baselines in the accuracy metric. In most cases, `CaCoSE` achieves a superior gain $\left(\frac{acc(\texttt{CaCoSE})-acc(baseline)}{acc(baseline)} * 100\right)$ in (%) over other methods. Particularly on the dense heterophilic: Chameleon and Squirrel datasets it shows performance gains of 4.69% and 20.63%, respectively, over the nearest performing baselines GOKU and GCN.

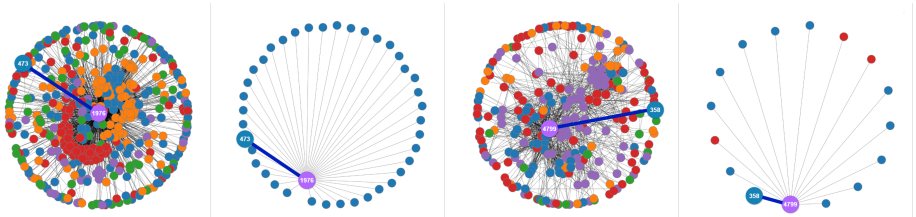
Table 2 presents the performance of our model in comparison to standard oversquashing addressing baselines. The `CaCoSE` achieves better or almost similar performance over the baselines. Notably, on the IMDB-BINARY and COLLAB datasets, it surpasses the other baselines with substantial gains of 4.27% and 13.50%, respectively, where the closest scoring baselines are GTR and LASER.

Sensitivity Analysis. We analyze the impact of the pooling ratio and number of heads on model performance. As shown in Fig. 3(a), the accuracy curve for `CaCoSE`'s exhibits minimal fluctuation on four (NC) datasets for lower to higher pooling ratios. In contrast on GC datasets, in Fig. 3(c) MUTAG's curve

displays an uneven trend, while other three demonstrate a moderate movement. Regarding the number of heads, Fig. 3(b) illustrates that, our model experiences small-scale performance changes on the Texas and CiteSeer datasets with nearly identical on other two NC datasets. For the GC datasets (in Fig. 3(d)), there is a little fluctuation in the performance on IMDB-BINARY and COLLAB while PROTEINS exhibits a gradual increase in performance.

Furthermore, in Table 3, we analyze `CaCoSE`'s performance by changing the $CaEF$ threshold δ across three NC datasets and three GC datasets. Generally, models performance is observed to decrease as the value of δ increase. However, fluctuates on some datasets, specially on IMDB-B and PROTEINS.

Analysis on Bridges in Heterophilic Networks. In experiments, `CaCoSE` shows outstanding performance on heterophilic networks. To determine the underlying reason, we analyze the bridge edges that act as the narrow channels in graphs. For each bridge edge, we examine the 2-hop neighborhood of its endpoint nodes and color the vertices by class label. The same analysis is repeated on the corresponding edges in the decomposed subgraphs. In many cases, the neighborhoods around decomposed heterophilic bridges reveal latent homophily.



(a) $\text{Chm}_{(1976,473)}$ (b) $\text{Chm-k}_{1\,(1976,473)}$ (c) $\text{Sqr}_{(4799,358)}$ (d) $\text{Sqr-k}_{1\,(4799,358)}$

Fig. 4: **Snippets of Bridge Analysis**. Edge $(1976, 473)$ in Chameleon (Chm) and $(4799, 358)$ in Squirrel (Sqr) datasets. k_1 denotes the subgraph (S_1) and ▬ presents the Bridge Edges.

Figure 4, illustrates the $2-hop$ surroundings of bridge edges in the both original graph (Figs. 4(a) and 4(c)) and the edge-induced partitioned subgraph (Figs. 4(b) and 4(d)). In the original graphs, bridge edges show strong heterophilic characteristics, causing cross-class contamination in GNN's aggregation and reducing representational expressivity. After k-core decomposition, the surroundings of bridge edges remain heterophilic and resemble star like pattern where hub node differs from its neighbors. However, most client nodes share the same class label. This allow a $2-layer$ GNN to effectively capture the homophilic structure. Hence, the partition preserves latent homophily in those subgraphs that enhances model's performance.

Ablation Study. In this analysis, we experiment with our model by altering its components. Figure 5 illustrates the performance gains of the k-core decomposition in `CaCoSE` compared to alternative partition methods on six different datasets. We use four NC datasets: two citations networks (Cora, CiteSeer)

	Cora	CiteSeer
Lv	0.520	0.813
M-4	0.283	0.813
M-8	0.473	1.107
Hi-4	0.461	0.872
Hi-8	0.854	0.173
Rw-4	-0.071	0.115
Rw-8	-0.188	0.478

(a)

	Chameleon	Squirrel
Lv	7.864	8.498
M-4	8.271	8.960
M-8	6.994	7.291
Hi-4	9.213	6.303
Hi-8	7.915	5.333
Rw-4	19.670	19.756
Rw-8	14.735	18.264

(b)

	IMDB-B	MUTAG
Lv	13.885	5.466
M-4	26.957	1.316
M-8	40.115	-3.157
Hi-4	15.525	31.607
Hi-8	16.242	17.542
Rw-4	14.780	6.945
Rw-8	18.123	13.237

(c)

Fig. 5: **Performance gain (in %)−`CaCoSE` vs other Decomp**ositions: Illustrate on six datasets (4 NC and 2 GC). Louvain (Lv), Metis (M), Hierarchical (Hi), and Random-Walk (Rw). Except for Louvain, the other methods are annotated with the number of partitions, i.e, ($M-4$) presents Metis with 4-partitions. Green and red shades present the positive and negative gains respectively.

Table 4: Ablation Study and Performance Comparison across Datasets. Accuracy values exceeding `CaCoSE` are indicated in bold.

Change	Component	Cora	C.Seer	Cham.	PROT.	IMDB-B
Attention Mechanism	without cross-Attention	84.42	68.38	**69.23**	**71.87**	73.00
CaEF	without CaEF	84.96	68.41	68.73	65.89	73.10
SAGPool	TopkPool	84.94	69.20	65.62	65.09	72.90
	DMon	**85.10**	69.37	65.82	**74.55**	67.50
	GMT	84.70	**69.74**	65.27	56.90	60.71
GCN	GAT	83.31	68.86	65.49	61.33	71.70
	SAGE	83.92	69.12	60.04	63.84	**73.90**

and two heterophilic networks (Chameleon, Squirrel). Besides, two GC datasets: IMDB-BINARY and MUTAG. With a few exceptions, our model consistently achieves higher accuracy than other decompositions.

Table 4 presents an ablation study on five datasets (three NC and two GC) to asses the contribution of different modules. First, we remove the attention mechanism, followed by the exclusion of the closure component ($CaEF$). In most cases, the fall in accuracy highlights the significance of these components in `CaCoSE`. Next, we replace SAGPool with alternative pooling methods, including TopKPool [16], DMonPool [30] and GMT [4]. With only a few exceptions, our model consistently outperforms these variants. Finally, we substitute the GCN backbone with GAT and GraphSAGE. Although GraphSAGE enables us to achieve higher accuracy on the IMDB-BINARY dataset, it performs worse on other datasets. Regarding GAT, our model consistently yields superior performance across all datasets.

5 Conclusion

In a nutshell, our `CaCoSE` model successfully demonstrates its efficacy that facilitates graph representation learning to overcome long-range dependency in

GNNs. Through closure-aware cohesive graph decomposition, it provides essential locality to the vertices. Besides, the SAGPool filters out noisy connections in networks and the attention mechanism provides crucial global information across subgraphs. Extensive experiments on benchmark datasets illustrate its consistency over the standard graph learning models. We expect this technique will open up some new research directions: subgraph-wise graph rewiring, regional graph super-resolution and parallel subgraph learning to develop robust graph representation learning models.

References

1. Akbas, E., Zhao, P.: Truss-based community search: a truss-equivalence based indexing approach. Proceedings of the VLDB Endowment **10**(11), 1298–1309 (2017)
2. Alon, U., Yahav, E.: On the bottleneck of graph neural networks and its practical implications. arXiv preprint arXiv:2006.05205 (2020)
3. Attali, H., Buscaldi, D., Pernelle, N.: Delaunay graph: Addressing over-squashing and over-smoothing using delaunay triangulation. In: Forty-first International Conference on Machine Learning (2024)
4. Bacciu, D., Conte, A., Grossi, R., Landolfi, F., Marino, A.: K-plex cover pooling for graph neural networks. Data Mining and Knowledge Discovery **35**(5), 2200–2220 (2021)
5. Barbero, F., Velingker, A., Saberi, A., Bronstein, M., Di Giovanni, F.: Locality-aware graph-rewiring in gnns. arXiv preprint arXiv:2310.01668 (2023)
6. Batagelj, V., Zaversnik, M.: An o (m) algorithm for cores decomposition of networks. arXiv preprint cs/0310049 (2003)
7. Bianchi, F.M., Grattarola, D., Alippi, C.: Spectral clustering with graph neural networks for graph pooling. In: International conference on machine learning. pp. 874–883. PMLR (2020)
8. Black, M., Wan, Z., Nayyeri, A., Wang, Y.: Understanding oversquashing in gnns through the lens of effective resistance. In: International Conference on Machine Learning. pp. 2528–2547. PMLR (2023)
9. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of statistical mechanics: theory and experiment **2008**(10), P10008 (2008)
10. Chen, C., Qian, J., Luo, H., Li, Y., Wang, X.: Parallel higher-order truss decomposition. arXiv preprint arXiv:2411.06405 (2024)
11. Chen, H., Conte, A., Grossi, R., Loukides, G., Pissis, S.P., Sweering, M.: On breaking truss-based and core-based communities. ACM Transactions on Knowledge Discovery from Data **18**(6), 1–43 (2024)
12. Chen, R., et al.: Redundancy-free message passing for graph neural networks. Advances in Neural Information Processing Systems **35**, 4316–4327 (2022)
13. Chu, D., Zhang, F., Zhang, W., Lin, X., Zhang, Y.: Hierarchical core decomposition in parallel: From construction to subgraph search. In: 2022 IEEE 38th International Conference on Data Engineering (ICDE). pp. 1138–1151. IEEE (2022)
14. Di Giovanni, F., Giusti, L., et al.: On over-squashing in message passing neural networks: The impact of width, depth, and topology. In: International conference on machine learning. pp. 7865–7885. PMLR (2023)
15. Dwivedi, V.P., Rampášek, et al.: Long range graph benchmark. Advances in Neural Information Processing Systems **35**, 22326–22340 (2022)

16. Gao, H., Ji, S.: Graph u-nets. In: international conference on machine learning. pp. 2083–2092. PMLR (2019)
17. Gu, M., Yang, G., Zhou, et al.: Homophily-enhanced structure learning for graph clustering. In: Proceedings of the 32nd ACM international conference on information and knowledge management. pp. 577–586 (2023)
18. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in neural information processing systems **30** (2017)
19. He, X., Hooi, B., Laurent, T., Perold, A., LeCun, Y., Bresson, X.: A generalization of vit/mlp-mixer to graphs. In: International conference on machine learning. pp. 12724–12745. PMLR (2023)
20. Hossain, T., Saifuddin, K.M., et al.: Tackling oversmoothing in gnn via graph sparsification. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 161–179. Springer (2024)
21. Karhadkar, K., Banerjee, P.K., Montúfar, G.: Fosr: First-order spectral rewiring for addressing oversquashing in gnns. arXiv preprint arXiv:2210.11790 (2022)
22. Karypis, G., Kumar, V.: Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices (1997)
23. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
24. Lee, J., Lee, I., Kang, J.: Self-attention graph pooling. In: International conference on machine learning. pp. 3734–3743. PMLR (2019)
25. Liang, L., Bu, F., Song, Z., Xu, Z., Pan, S., Shin, K.: Mitigating over-squashing in graph neural networks by spectrum-preserving sparsification. arXiv preprint arXiv:2506.16110 (2025)
26. Liu, Q.C., et al.: Parallel k-core decomposition with batched updates and asynchronous reads. In: Proceedings of the 29th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming. pp. 286–300 (2024)
27. Malliaros, F.D., Giatsidis, C., Papadopoulos, A.N., Vazirgiannis, M.: The core decomposition of networks: Theory, algorithms and applications. The VLDB Journal **29**(1), 61–92 (2020)
28. Nguyen, K., Hieu, N.M., Nguyen, V.D., Ho, N., Osher, S., Nguyen, T.M.: Revisiting over-smoothing and over-squashing using ollivier-ricci curvature. In: International Conference on Machine Learning. pp. 25956–25979. PMLR (2023)
29. Topping, J., Di Giovanni, F., Chamberlain, B.P., Dong, X., Bronstein, M.M.: Understanding over-squashing and bottlenecks on graphs via curvature. arXiv preprint arXiv:2111.14522 (2021)
30. Tsitsulin, A., Palowitch, J., Perozzi, B., Müller, E.: Graph clustering with graph neural networks. Journal of Machine Learning Research **24**(127), 1–21 (2023)
31. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
32. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
33. Wu, Y., Wang, L., et al.: Graph contrastive learning with cohesive subgraph awareness. In: Proceedings of the ACM Web Conference 2024. pp. 629–640 (2024)
34. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., Leskovec, J.: Hierarchical graph representation learning with differentiable pooling. Advances in neural information processing systems **31** (2018)
35. Zhang, X., Liu, H., Wu, X.M., Zhang, X., Liu, X.: Spectral embedding network for attributed graph clustering. Neural Networks **142**, 388–396 (2021)

A Appendix

A.1 Proof through Ollivier-Ricci Curvature

Olliver-Ricci curvature [28]—denoted by $\kappa(u, v)$—is a measure of the local similarity of the neighborhoods of two vertices, typically by means of random walk measures centered at those vertices. This value is positive or negative when the overlap between the probability distributions of random walkers between $N(u)$ and $N(v)$ is large or small, respectively.

Let $G = (V_G, E_G)$ be a simple graph with shortest path distance $d(u, v)$ for all $u, v \in V_G$. Let $\Pi(\mu_u, \mu_v)$ be the family of joint probability distributions of μ_u and μ_v, and take μ_u to be the 1-step random walk measure given by

$$\mu_u(v) = \begin{cases} \frac{1}{deg(u)} & \text{if } v \in N(u) \\ 0 & \text{otherwise.} \end{cases}$$

Then define the L^1-Wasserstein distance as

$$W_1(\mu_u, \mu_v) = \inf_{\pi \in \Pi(\mu_u, \mu_v)} \left(\sum_{(p,q) \in V_G^2} \pi(p, q) d(p, q) \right),$$

which measures the minimum distance random walks between u and v must take to meet each other. The Ollivier-Ricci curvature is then

$$\kappa(u, v) = 1 - \frac{W_1(\mu_u, \mu_v)}{d(u, v)}.$$

This carries the interpretation that whenever two random walkers are unlikely to meet, we have $\kappa(u, v) < 0$.

Theorem 2. *For all* $(u, v) \in E_G$, *if* $N(u) \cap N(v) = \emptyset$, *then* $\kappa(u, v) \leq 0$

Proof. For any $\pi \in \Pi(\mu_u, \mu_v)$, consider the case of $\pi(p, q) > 0$ since $\pi(p, q) = 0$ contributes 0 to $\sum_{(p,q) \in V_G^2} \pi(p, q) d(p, q)$. Then $p \in N(u)$ and $q \in N(v)$ since $\sum_{q \in V_G} \pi(p, q) = \mu_u(p)$ and $\sum_{p \in V_G} \pi(p, q) = \mu_v(q)$ give $p \notin N(u) \Rightarrow \sum_{q \in V_G} \pi(p, q) = 0$ and $q \notin N(v) \Rightarrow \sum_{q \in V_G} \pi(p, q) = 0$, respectively. But $N(u) \cap N(v) = \emptyset$, so $p \neq q$ for all $(p, q) \in V_G^2$ with $\pi(p, q) > 0$. Thus, $d(p, q) \geq 1$. It follows that

$$\sum_{(p,q) \in V_G^2} \pi(p, q) d(p, q) \geq \sum_{(p,q) \in V_G^2} \pi(p, q) \cdot 1 = 1.$$

Moreover, $W_1(\mu_u, \mu_v) \geq 1$. Since $(u, v) \in E_G$, we have $d(u, v) = 1$. Hence,

$$\kappa(u, v) = 1 - W_1(\mu_u, \mu_v) \leq 0.$$

Thus, Theorem 2 shows that the edges removed by the $CaEF$ in algorithm have non-positive Ollivier-Ricci curvature; in other words, they correspond with bottlenecks. This approach can then be said to mitigate oversquashing by removing the message passing pathways associated with this class of bottlenecks.

Algorithm 1: CaCoS Algorithm

Input: A graph $G = (V, E)$
Output: Vertex embeddings Z_v and graph embedding Z_G
`/* Measure edge score with the k-core algorithm */`
1 $C(u, v) = \max\{k \mid (u, v) \in G_k\}$
`/* Apply CaEF */`
2 **if** $S(u, v) = 0$ **then**
3 $\quad C(u, v) = C(u, v) - 1$
4 $S_k = \{(u, v) \in E : C(u, v) = k\}$
`/* Extract subgraphs */`
5 $S = \{\text{Edge-Subgraph}(S_k) : k \in 1, \ldots, K_{\max}\}$
6 **foreach** $S_k = (V_S, E_S) \in S$ **do**
7 $\quad H_k = \text{GCN}_k(S_k)$
8 $\quad Z_k = \text{SAGPool}(S_k, H_k)$
9 $Z_{S_k}^{attn} = \text{Attention}(Z_{S_k})$
`/* Apply attention among subgraphs */`
10 $Z_G = \text{Mean}\left([Z_{S_1}^{attn}, Z_{S_2}^{attn}, \ldots, Z_{S_k}^{attn}]\right)$
11 $Z_v = \text{zeros}(N_v, d_v + d_S)$
12 **foreach** (H_k, Z_k^{attn}) *associated with* S_k **do**
13 $\quad$ **foreach** $h_v \in H_k$ **do**
14 $\quad\quad h_v = h_v \parallel Z_k^{attn}$
$\quad\quad$ `/* Concatenation operation */`
15 $\quad\quad z_v = z_v + h_v$
$\quad\quad$ `/* Map` v `from` H_k `to` Z_v `*/`
16 **return** Z_G, Z_v

A.2 Representation Learning by CaCoSE

Algorithm 1 presents the execution of the CaCoSE model for k-core decomposition. In lines (1-3), at first, the input graph is decomposed with the k-core algorithm, where nodes achieve their coreness score. Next, CaCoSE explores all the edges and assigns the coreness scores as the edge weight. Additionally, it utilizes edge filtering ($CaEF$) to update specific narrow channels edge scores. In the next stage (lines 4-8), the CaCoSE extracts edge-induced subgraphs from the input graph based on the coreness scores. Note that, the k-core algorithm [27] iteratively removes the nodes with degree less than k as k increases. Thus, in each node removal stage, the $(k-1)$ value is assigned to the removed nodes.

Then, on the partitioned subgraphs, it concurrently applies GCN for node representation learning as well as $SAGPool$ to encode the entire subgraph's information. In the next step (line 9), CaCoSE utilizes the attention mechanism across the subgraphs' embeddings to capture the mutual information among them. Next, it takes the average of all the subgraph embeddings that represent the final representation of the graph (line 10). In the context of node embeddings (lines 11-15), each subgraph's cross-attentive embeddings are concatenated with its node representations, while the shared nodes are mapped and their combined features are summed up in the vertices' global feature matrix.

Complexity. In worst case, the time complexities of k-core, SAGPool, and cross attention are $O(V+E)$, $O(V^2)$, and $O(k^2d)$. Hence the overall complexity of `CaCoSE` is $O(V^2+V+E+k^2d)$. Although looks complex, `CaCoSE` benefits from parallelizable k-core decompositions and efficient GPU execution of SAGPool and cross-attention. The detailed algorithm is omitted due to space constraints.

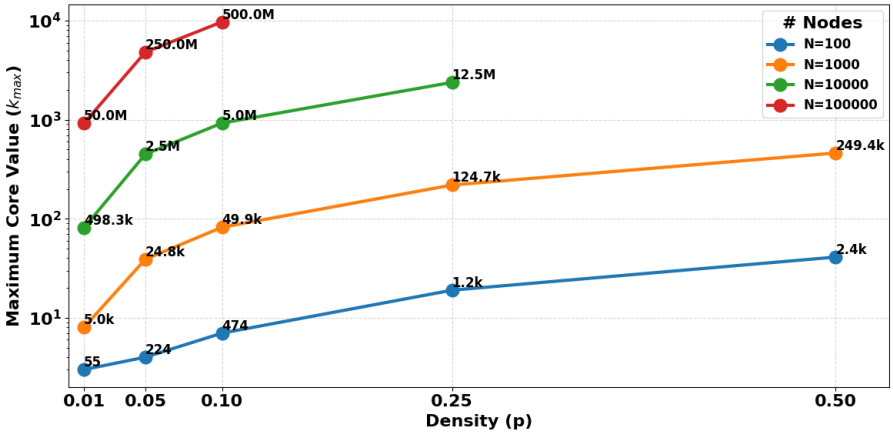


Fig. 6: **Scalability Test.** This figure presents the maximum core value K_{max} along the y-axis for networks generated with various combinations of graph size ($\#Nodes$) and density (p). The value of K_{max} is corresponds to the maximum number of subgraphs that can be extracted through k-core decomposition.

Scalability. In this experiment, we apply the Erdős–Rényi graph generator to generate random networks of varying scale. During the generation process, we construct graphs with vertex count of 10^2, 10^3, 10^4 and 10^5. For each graph size ($\#Nodes$) we employ the edge creation probability $p \in \{0.01, 0.05, 0.10, 0.25, 0.50\}$. As the graph size increases, the number of possible edges-and consequently the total generated edges-grows significantly.

For each generated graph, we compute the maximum core number (K_{max})m representing the highest number of subgraphs that can be extracted through the k-core decomposition. Figure 6 plots K_{max} against the graph's density p (x-axis) for networks having different scales (distinguished by colors and markers). Besides, each plot is annotated with the number of edges corresponding to each combination of graph size and density.

The result depicts that even in extremely large and densely connected networks, the maximum core value remains below 10^4. In practical cases the value of K_{max} is substantially smaller, this indicates that the number of extractable cohesive subgraphs via k-core decomposition is limited in realistic settings.

Since, the real-world graphs are sparser and the resulting subgraphs sizes are smaller, it implies that the computational overhead introduced by the attention mechanism in our framework is negligible. It is worth noting that, the generation

process was constrained by hardware capacity. For graphs with 10^4 vertices we limit the density $p \leq 0.25$, and for graph size with 10^5, we consider $p \leq 0.10$.