SAC−GCL: Structure-Aware Compression for Graph Contrastive Learning

Muhammad Ifte khairul Islam
Department of Computer Science
Texas A&M University- Texarkana
Texarkana,Texas, USA
mislam1@tamut.edu

Tanvir Hossain
Department Of Computer Science
Georgia State University
Atlanta, GA, USA
thossain5@student.gsu.edu

Esra Akbas
Department Of Computer Science
Georgia State University
Atlanta, Georgia, USA
eakbas1@gsu.edu

arXiv:2610.37898v1 [cs.LG] 27 Oct 2026

***Abstract*—Graph Contrastive Learning (GCL) has achieved remarkable success in self-supervised representation learning by maximizing agreement between augmented views of a graph. Although augmented view generation is the central step in GCL, existing models rely on stochastic augmentation of the input graph, such as edge perturbation and node masking, which lack structural awareness and frequently destroy critical topology or introduce semantically inconsistent views. While recent adaptive methods improve upon random perturbations, they primarily operate on localized node or edge neighborhoods and fail to capture higher-order, macro-level structural organization crucial for graph classification. To address this limitation, we propose Structure-aware Compression for Graph Contrastive Learning (SAC−GCL), a novel framework that employs structural decomposition and graph compression as a principled view generation strategy. Moving beyond random augmentations, SAC−GCL leverages structural decomposition techniques to extract cohesive subgraphs and condense them into supernodes, constructing coarsened graph views that preserve local density and capture higher-level structural patterns while reducing noise and redundancy. We leverage multiple cohesive subgraph models to derive complementary views of the graph, each capturing distinct structural properties. To capture multi-resolution information, a hierarchical view encoder processes these compressed views by initializing super-node features with shared-weight cohesive subgraph embeddings. Our results from extensive experiments on six benchmark datasets demonstrate that structure-aware compression provides a powerful alternative to random perturbations, yielding more robust and discriminative graph representations for graph classification tasks.**

***Index Terms*—Graph Contrastive Learning, Graph Compression, Graph Neural Networks, Graph Classification, Graph Decomposition, Cohesive Subgraphs**

I. Introduction

Recently, graph contrastive learning (GCL) has emerged as a powerful self-supervised paradigm for graph representation learning, demonstrating state-of-the-art performance across both node-level and graph-level downstream tasks. GCL aims to maximize the agreement between representations of semantically consistent views of the same graph while distinguishing them from representations of other graphs [1]–[4]. Recent models typically generate multiple graph views through stochastic augmentation strategies like random node dropping [2], edge deletion [5], and attribute masking [3], implicitly assuming that semantic information is preserved under stochastic corruption. While these augmentations provide diversity and encourage the model to learn invariances, they often lack structural awareness and may disrupt crucial structural information or introduce semantically inconsistent views that inject noise into the contrastive learning objective.

This limitation has motivated the need for more structure-aware augmentation in graph contrastive learning [6], [7]. These methods aim to generate more informative views by formulating augmentation as a learnable process, leveraging model feedback or extracting local subgraphs. Although these adaptive schemes offer significant improvements over purely stochastic baselines by preserving local node centralities, they remain bound to local topological operations (e.g., node- or edge-level modifications). Consequently, they lack an explicit mechanism to capture, preserve, and leverage the higher-level structural organization inherent in complex networks. This drawback is particularly pronounced in graph-level classification tasks, where global topology, macro-scale connectivity, and multi-scale substructures dictate overall graph functionality and semantic identity.

On the other hand, graphs are organized into structurally dense regions that encode important semantic and functional information [8]. In particular, cohesive subgraphs such as k-cores and k-trusses [9], [10] isolate densely connected and semantically meaningful regions that are robust to perturbations, making them well-suited for constructing stable and informative graph views. Unlike local ego-nets or isolated subgraphs, these cohesive structures capture higher-order topological interactions and retain the fundamental structural backbone of the network.

In this work, we propose **S**tructure-**a**ware **C**ompression for **G**raph **C**ontrastive **L**earning (SAC−GCL), a novel framework for graph-level representation learning. Unlike random augmentations or locality-preserving subgraph sampling, SAC−GCL introduces a structural decomposition and graph compression pipeline that explicitly preserves fine-grained connectivity while abstracting macro-level graph topology. Specifically, our framework extracts dense cohesive subgraphs and compresses them into higher-order super-nodes, constructing coarsened graph views that capture both internal local density and external inter-community relationships. By abstracting dense regions into super-nodes, these compressed

views provide multi-scale structural representations that effectively filter out local topological noise and redundancy without losing fundamental connectivity patterns.

While our framework is generic and can accommodate various cohesive subgraph extraction algorithms, in this work we utilize k-core and k-truss decompositions due to their efficiency and complementary properties. Specifically, k-core captures node-level connectivity by enforcing minimum degree constraints, whereas k-truss captures edge-level cohesion through triangle-based reinforcement. Together, they enable the model to represent complementary aspects of structural organization in a scalable manner.

To learn effective representations, proposed structure-preserving compressed views are encoded hierarchically with dedicated view encoders and optimized through a joint objective that combines supervised classification with contrastive consistency across views. In the view encoder, we first generate subgraph embeddings via shared-weights GNN applied across all extracted cohesive subgraphs. These embeddings are then utilized as the initial features for the corresponding super-nodes in the compressed graph while learning the compressed graph embeddings. By jointly encoding both original and compressed views, the framework captures complementary structural information across resolutions. Overall, `SAC-GCL` produces robust, structure-aware representations that overcome the limitations of random augmentation-based methods. Our main contributions are summarized as follows:

- We propose a novel contrastive learning framework for graphs that employs graph compression and structural decomposition as a deterministic, structure-aware view generation strategy, moving beyond random perturbation-based augmentations.
- We introduce a hierarchical encoding approach combining local (cohesive subgraphs) and global (super-node aggregation) information. With a joint objective combining supervised and contrastive loss, our model generates robust, structure-aware graph representations.
- We perform extensive experiments on benchmark graph classification datasets, demonstrating that `SAC-GCL` consistently outperforms state-of-the-art baselines while providing enhanced robustness against topological noise.

II. Preliminaries and Related Work

In this section, we begin by introducing the preliminaries and problem formulation. We then give the related work about GNN and graph contrastive learning.

A. Preliminaries and Problem Formulation

We denote a graph as $G(V, A, X)$ where V is the node-set, $A \in \mathbb{R}^{\mathbb{N}\times\mathbb{N}}$ is the adjacency matrix, and $X \in \mathbb{R}^{N\times d}$ is the feature matrix with d dimensional node features and N is the number of nodes in the graph. We represent a graph collection as $(\mathcal{G}, Y)$ where $\mathcal{G} = \{G_0, G_1, ..., G_n\}$ with G_i's are graphs and Y is the set of the graph labels. In this paper, we work on the graph classification problem, where the goal is to learn a function $f : \mathcal{G} \to Y$ to predict graph labels with a graph neural network in an end-to-end way.

*1) **Graph Neural Network**:* Graph Neural Networks (GNNs) have been widely used for static graphs in tasks such as node classification, link prediction, and graph classification. Among them, the Graph Convolutional Network (GCN) [11] is one of the most widely adopted architectures. A GCN is a multilayer neural network that processes graph data by combining the features of each node with those of its neighbors while propagating information through edges.

Given an input graph $G = (V, A, X)$, where $X \in \mathbb{R}^{N\times d}$ is the node feature matrix, we set the initial hidden representation as $H^{(0)} = X$. A GCN layer updates node representations as:

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

where $\tilde{A} = A + I$ is the adjacency matrix with self-loops, $\tilde{D}$ is the degree matrix of $\tilde{A}$, $H^{(l)}$ denotes the feature matrix at the l-th layer, $\Theta^{(l)}$ are trainable parameters, and σ is a nonlinear activation function. Intuitively, each GCN layer aggregates information from one-hop neighbors, so an L-layer GCN captures information up to L hops away. In our framework, GCN is used as the default instantiation of the message-passing operator $\text{GNN}^{(l)}(\cdot)$ in both levels of the view encoder.

For graph classification, the GCN is typically trained with a cross-entropy loss that compares the predicted labels with the ground-truth labels.

*2) **Contrastive Learning**:* Contrastive learning (CL) has become one of the most popular approaches for unsupervised representation learning, which learns through comparisons among different samples. This comparison is typically conducted between positive pairs of "similar" inputs and negative pairs of "dissimilar" inputs. Contrastive learning in the graph domain aims to learn a node or graph representation for a given input graph by maximizing the similarity between the different views of the input graph in their latent space via contrastive loss. In general, there are three parts to graph contrastive learning: i) View construction, ii) View encoder, and iii) Contrastive loss.

- **View Construction:** A view is represented as graph data, denoted as $v_i = (A_i, X_i)$, where $A_i \in \mathbb{R}^{\mathbb{N}\times\mathbb{N}}$ and $X_i \in \mathbb{R}^{N\times d}$. While it is common to get 2 views, there are also some models that create more than 2 views [12], [13]. In practical terms, view augmentation approaches involve techniques like node dropping [14], edge perturbation [4], attribution masking [2], and subgraph sampling [15].
- **View Encoder:** The view encoder maps the node representation or graph representation from different views of input graphs. As encoders for learning representations from views, any GNN model can be used based on the problem and type of graphs.
- **Contrastive loss:** After learning the representations for different views using the view encoder, the contrastive learning model is optimized by a contrastive loss. The

objective of the contrastive loss function is to maximize the similarity among different view representations of the same graph and minimize the similarity for view representations of other graphs. In general, the contrastive loss can be described as follows:

$$L_{contra} = -\log\left(\frac{sim(Z_{v_i}, Z_{v_j})}{\sum_{j'=1}^{N} sim(Z_{v_i}, Z_{v_{j'}})}\right) \quad (1)$$

where the $sim(\cdot)$ function measures the similarity between two representations. We can use any similarity function, like cosine similarity and mutual information (MI) in the contrastive loss. Z_{v_i} and Z_{v_j} are the ith view and jth view representations of node v.

B. Related Works

Graph neural networks (GNNs) have become the dominant approach for graph representation learning due to their ability to capture both node attributes and structural dependencies. Early models such as Graph Convolutional Networks (GCN) [11] and Graph Attention Networks (GAT) [16] demonstrated the effectiveness of message passing in aggregating neighborhood information. Subsequent advances, including pooling and spectral techniques, have improved their ability to capture higher-order dependencies, which have proven particularly effective for graph classification in domains such as chemistry, bioinformatics, and social networks. However, these models generally rely on large-scale labeled data, which can be scarce in practice. This limitation has motivated the development of self-supervised approaches.

To address labeled data scarcity, Graph contrastive learning (GCL) models learn node and graph representations by maximizing agreement between augmented views of a graph [1], [17]–[21]. They relies heavily on view generation to construct correlated, self-supervised contrastive pairs that force the encoder to capture invariant semantic representations [17]. Initial models in GCL such as GRACE [4] and GraphCL [2] apply random augmentations such as node droping, edge removal and attribute masking to generate diverse graph views. While it is common to get two views, some models also create more than two views [12], [13]. While computationally efficient, they apply uniform perturbations across the entire graph and these perturbations are largely structure-agnostica, which disrupt important connectivity patterns and unintentionally degrading the cohesive regions of the graph [22].

To address this issue, CI-GCL [23] introduces a community-invariant augmentation strategy that preserves high-level community structure during contrastive view generation. It further unifies topology and feature augmentation through a spectral constraint, improving robustness against different types of graph noise. Recent research also explores other structure-aware augmentation strategies, such as motif-driven or cohesive subgraph-based methods (e.g., MicroGraph [15], MHGCL [24]). These methods aim to retain community-level or motif-level structure, but they still generate augmented views through perturbation or heuristic augmentation. More recent paradigms utilize learnable view generators or data-driven

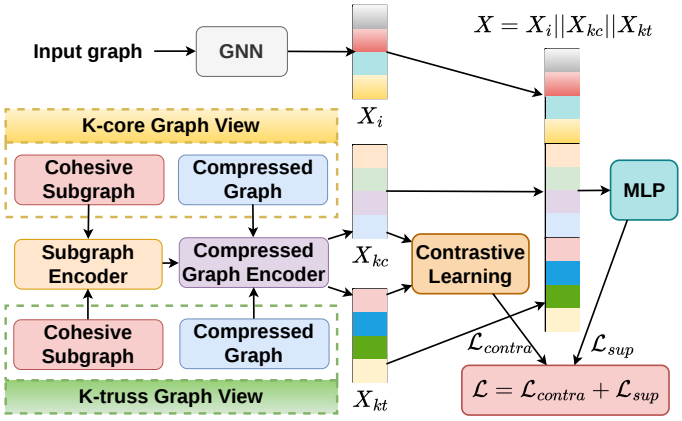


Fig. 1: Model Architecture: Graph Contrastive Learning with Structure-Aware Graph Compression

optimization to discover optimal views automatically [25]. While effective at capturing complex graph characteristics, generative and optimization-driven pipelines introduce significant computational overhead, require delicate joint optimization schedules, and remain susceptible to mode collapse or unstable training dynamics.

Graph compression and sparsification have been extensively studied in graph mining as a way to reduce complexity while preserving essential structures. Early works primarily utilized structural pruning to lower computational bottlenecks and mitigate message-passing pathologies. For example, TADA [26] utilizes topology-aware graph sparsification to augment the graph for the oversmoothing problem in deep Graph Neural Networks (GCNs). In parallel, recent self-supervised learning frameworks incorporate graph pooling and coarsening algorithms to hierarchically condense graph topology, generating multiscale views that capture multi-granularity representation [6], [7]. Unlike random perturbation-based or purely locality-preserving subgraph views, our approach leverages structural decomposition and targeted compression as a unified, structure-aware view generation pipeline. By systematically decomposing the graph into fundamental structural primitives (e.g., spectral components or dense subgraphs) and compressing redundant topology, our method preserves both global connectivity and fine-grained neighborhood semantics.

III. Methodology

In this section, we present our proposed framework, Structure-aware Compression for Graph Contrastive Learning (`SAC-GCL`). `SAC-GCL` aims to enhance graph representations, particularly for graph classification tasks. The overall architecture is illustrated in Figure 1 and comprises three main components: (i) **Graph View Generation via Cohesive Subgraph Compression**, (ii) **Structure-Aware View Encoder**, and (iii) **Contrastive Learning**.

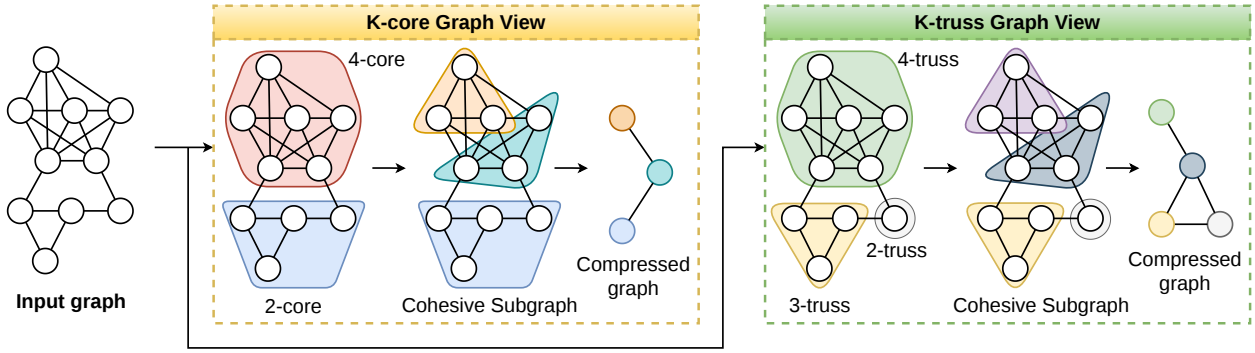


Fig. 2: Graph View Generation via Cohesive Subgraph Compression. Input graphs are decomposed into cohesive k-core and k-truss subgraphs with a size threshold. Extracted cohesive subgraphs are contracted into super-nodes to construct compressed graph views

.

A. *Graph View Generation via Cohesive Subgraph Compression*

Unlike random augmentations, SAC−GCL leverages structural decomposition and compression methods to generate structurally distinct graph views from the original input graph G. Graph decomposition methods extract cohesive subgraphs that preserve fine-grained connectivity patterns. Figure 2 illustrate the overview of graph view generation via cohesive subgraph compression. While our framework is generic and can accommodate any method for extracting cohesive subgraphs, in this work, we utilize k-core and k-truss decompositions. These two decompositions capture fundamentally different notions of cohesiveness within a network: k-core identifies dense regions based on *vertex degree*, while k-truss identifies cohesive structures based on *triangle participation*. By leveraging both, we obtain complementary structural perspectives that enrich the subsequent contrastive learning process.

Next, we introduce a compression step that condenses cohesive subgraphs into supernodes, constructing compressed graphs as views that capture both local and global structural patterns. These compressed views, derived from cohesive subgraphs, provide higher order structural abstractions while preserving local density within supernodes.

k-core decomposition: The k-core of a graph represents a subgraph where every node has a degree of at least k. Unlike node-level measures that quantify connectivity based on individual node degree, the k-core captures the collective connectivity of a group of nodes by identifying densely connected substructures within the graph. Formally, given an undirected graph $G = (V, E)$ and an integer $k \geq 1$, the K-core is the maximal induced subgraph $G_k = (V_k, E_k)$ such that:

$$\forall v \in V_k, \quad \deg_{G_k}(v) \geq k.$$

Intuitively, the k-core decomposition iteratively removes nodes with degree less than k, along with their incident edges, until no such nodes remain. Higher values of k produce more tightly connected subgraphs, representing the structural backbone of the network. The k-core hierarchy $G_k \subseteq G_{k-1} \subseteq \cdots \subseteq G_1$ reveals connectivity layers of increasing cohesiveness. This property makes k-core decomposition a natural choice for view generation: it emphasizes progressively denser regions, providing an effective basis for forming compressed graph representations.

k-truss decomposition: While k-core is based solely on the degree of nodes, k-truss considers *triangle-based edge support*, capturing higher-order connectivity patterns in the graph. The support of an edge $e = (u, v) \in E$ in G is defined as:

$$sup_G(e) = |\{w \in V : (u, w) \in E \ \wedge \ (v, w) \in E\}|,$$

which counts the number of triangles in which the edge participates. A subgraph $S = (V_S, E_S)$ is a **k-truss** if

$$\forall e \in E_S, \quad sup_G(e) \geq k - 2$$

where $k \geq 2$. The k-truss hierarchy is nested, i.e., $G_k \subseteq G_{k-1}$, and increasing k yields progressively more cohesive triangle-rich regions. This property is particularly useful for contrastive view generation because triangle-rich subgraphs tend to correspond to community-like structures in real-world networks, which are structurally different from degree-based k-core regions.

Subgraph extraction and refinement: For each decomposition method (k-core and k-truss), we compute subgraphs over a range of k values, $k \in \{1, ..., k_{max}\}$ for $k-$ core and $k \in \{2, ..., k_{max}\}$ for k-truss, to capture structural diversity. However, a direct decomposition can yield overlapping subgraphs, as nodes or edges may appear in multiple k-levels. To avoid redundancy, we assign each node or edge to the highest-k subgraph in which it appears, thereby preserving its most cohesive structural role. This strategy

ensures that the extracted subgraphs are mutually exclusive and collectively cover the original graph's cohesive regions. Specifically, each node v is assigned a core number $k(v)$, defined as the largest k for which v belongs to the k-core of the graph. Nodes with the same core number are then grouped, and each connected component induced by these nodes defines a shell. Thus, a shell consists of a maximally connected set of nodes that share the same core number. We apply the same principle to k-truss decomposition. In both cases, each shell is treated as a cohesive subgraph, and the resulting shells are mutually exclusive while collectively representing the graph's cohesive structure.

While decomposition generates cohesive subgraphs, in practice, some subgraphs can be significantly large. Compressing these large subgraphs into a single compressed nodes may yield a very small compressed graph and thus information loss. To prevent excessive information loss within large cohesive regions, we impose a maximum size threshold λ for any subgraph. If a cohesive subgraph exceeds this threshold, we split it into smaller connected subgraphs. To preserve the local connectivity of nodes within subgraphs, instead of a random split, we apply a Breadth-First Search (BFS) partitioning procedure. Specifically, for a large subgraph, we choose a random node as a seed, run BFS over the subgraph, and split the subgraph into smaller connected components. The process repeats until all resulting subgraphs contain at most λ nodes. This ensures that each super-node in the compressed graph represents a localized portion of the original community, thereby retaining meaningful local structure while capturing global structure in a compressed graph.

Compressed graph construction: We construct a compressed graph in which each subgraph is represented as a super-node. Nodes in a subgraph are combined into a super-node. This reduces the graph size by filtering low-level details while preserving high-level structural relationships between the cohesive components. Two super-nodes i and j are connected if any of their constituent nodes are adjacent in the original graph, and the edge weight is defined as

$$A_{\text{super}}(i,j) = \frac{|E_{ij}|}{\min(|E_i|, |E_j|)},$$

where E_{ij} denotes the set of edges between subgraphs i and j, and $|E_i|$, $|E_j|$ are the number of edges internal to subgraphs i and j, respectively. This normalization by the smaller of the two internal edge counts ensures that the connection strength between super-nodes reflects their relative interconnectivity rather than absolute size. The compressed graph preserves the connectivity between cohesive subgraphs while removing the fine-grained node-level details within each shell.

Compressed graphs enable the model to focus on relationships between higher-order structures (e.g., dense cores) rather than individual nodes, which can help reduce noise and improve representation quality of graphs. Also, by generating compressed views from two different decomposition strategies, we obtain complementary abstractions of the same underlying graph, which is critical.

Importantly, the k-core and k-truss decompositions capture different notions of structural cohesiveness and therefore produce different sets of cohesive subgraphs. Applying the above compression procedure independently to each decomposition yields two compressed graphs, denoted by $G^c\mathrm{core}$ and $G^c\mathrm{truss}$. Although both compressed graphs are derived from the same original graph G, their super-nodes and connectivity patterns differ because they are based on different structural properties. We therefore treat $G^c\mathrm{core}$ and $G^c\mathrm{truss}$ as two complementary structural views of the original graph. These views provide different but related representations of the graph's cohesive organization, which are subsequently used in a contrastive learning framework to encourage the model to learn representations that are consistent across the two structural perspectives while retaining view-specific structural information.

B. Structure-Aware View Encoder

The view encoder generates informative embeddings from each compressed graph view using a two-level GNN strategy: *subgraph-level encoding* and *compressed-graph encoding*. The subgraph-level stage captures fine-grained local patterns within cohesive subgraphs, while the compressed-graph stage captures higher-order structural relationships between these subgraphs. We perform subgraph encoding first so that each cohesive subgraph is summarized into a meaningful vector representation, allowing the compressed-graph GNN to focus on inter-subgraph interactions rather than relearning local details, resulting in robust, structure-aware graph representations.

Subgraph-level Encoding Let a cohesive subgraph be denoted as $G_s = (\mathcal{V}_s, \mathcal{E}_s, X_s)$, where $\mathcal{V}_s$ and $\mathcal{E}_s$ are its nodes and edges and X_s is the initial features of the nodes in G_s. We apply a dedicated subgraph-level GNN (distinct for each view) with shared weights across all subgraphs within that view. The GNN updates node embeddings through message passing:

$$\mathbf{h}_v^{(l+1)} = \text{GNN}^{(l)}\Big(\mathbf{h}_v^{(l)}, \{\mathbf{h}_u^{(l)} : u \in \mathcal{N}_s(v)\}\Big),$$

where $\mathbf{h}_v^{(0)} = \mathbf{x}_v$ After the final layer, we apply a readout function to obtain a single subgraph embedding:

$$\mathbf{z}_s = \text{Readout}\Big(\{\mathbf{h}_v^{(L_s)} : v \in \mathcal{V}_s\}\Big),$$

The embedding $\mathbf{z}_s$ captures the local structural patterns within each subgraph.

Compressed Graph Encoding The subgraph embeddings $\mathbf{z}_s$ serve as the initial features for the super-nodes: as $\mathbf{h}_i^{(0)} =$

$\mathbf{z}_{s_i}, \quad i \in \mathcal{V}_c$. We then apply a second GNN over the compressed graph:

$$\mathbf{h}_i^{(l+1)} = \text{GNN}^{(l)}\Big(\mathbf{h}_i^{(l)}, \{\mathbf{h}_j^{(l)} : j \in \mathcal{N}_c(i)\}\Big), \quad l = 0, \ldots, L_c - 1,$$

where $\mathcal{N}_c(i)$ denotes the neighbors of super-node i in the compressed graph, and L_c is the number of layers.

This **two-level encoding** preserves both local subgraph information and global higher-order structure, resulting in **robust, structure-aware graph representations**. In both subgraph and compressed graph encoding layers, as a readout function, we apply mean and max pooling and concatenate their outputs:

$$Z = \frac{1}{N}\sum_{i=1}^{N} h_i \,\|\, \max_{1 \le i \le N} h_i,$$

where h_i denotes the embedding of node i and $\|$ represents concatenation. Mean pooling summarizes the global distribution of features, while max pooling captures salient structural signals. Applying this readout to the set of subgraph embeddings gives the subgraph-derived graph embedding Z_{sub}. Applying it to super-node embeddings gives the compressed-graph embedding Z_{comp}. As the final embedding for a view, we concatenate these into one as $Z_{\text{final}} = Z_{\text{sub}} \,\|\, Z_{\text{comp}}$.

The same pipeline is applied independently to the k-core and k-truss compressed graphs, yielding the two final view embeddings Z_{core} and Z_{truss}.

C. *Contrastive Learning*

In the final stage, we align the two view representations while also optimizing for graph classification. We adopt a cross-view contrastive objective: for each graph, the embeddings from the k-core and k-truss views form a positive pair, while negatives are taken from *all other graphs in the batch*, but always across views (i.e., $Z_{\text{core}}^{(i)}$ is contrasted against $Z_{\text{truss}}^{(j)}$ for $j \neq i$). This cross-view negative sampling, following GraphCL [2], encourages the model to learn view-invariant representations across structurally different compressions. Instantiating Eq. (1) with cross-view positive pairs, the contrastive loss for graph i is defined as:

$$\mathcal{L}_{\text{contra}}^{(i)} = -\log\left(\frac{\text{sim}(Z_{\text{core}}^{(i)}, Z_{\text{truss}}^{(i)})}{\sum_{j=1}^{N}\text{sim}(Z_{\text{core}}^{(i)}, Z_{\text{truss}}^{(j)})}\right), \tag{2}$$

where $\text{sim}(\cdot)$ is a similarity function such as cosine similarity, $Z_{\text{core}}^{(i)}$ and $Z_{\text{truss}}^{(i)}$ are the paired embeddings of the same graph from the two views, and $Z_{\text{truss}}^{(j)}$ represents embeddings of other graphs in the batch.

The embeddings from the two views, Z_{core} and Z_{truss}, are concatenated to form a joint representation as $Z_{\text{joint}} = Z_{\text{core}} \,\|\, Z_{\text{truss}}$ and then Z_{joint} is fed into a multi-layer perceptron (MLP) for graph classification. As the supervised loss, cross-entropy (CE) loss is used as

$$\mathcal{L}_{\text{sup}} = -\sum Y \log \hat{Y}, \quad \hat{Y} = \text{MLP}(Z_{\text{joint}}). \tag{3}$$

The overall training objective combines supervised and contrastive losses given in Equations 2 and 3:

$$\mathcal{L} = \alpha\mathcal{L}_{\text{sup}} + (1-\alpha)\mathcal{L}_{\text{contra}},$$

where α balances label supervision and cross-view consistency.

D. *Theoretical Insight on Compression-Based View Alignment*

Random perturbation–based view generation in graph contrastive learning often disrupts global connectivity by removing nodes or edges arbitrarily. In contrast, our compression-based strategy constructs structurally coherent graph views by deterministically grouping cohesive regions (via k-core and k-truss) and representing their interconnections at a higher level of abstraction. Although intra-community edges are collapsed during compression, the local structural context is largely preserved through the subgraph splitting mechanism applied before compression. Aligning multiple compressed views—each derived from a distinct decomposition such as k-core and k-truss—encourages the model to learn rep- representations that are invariant to redundant internal structure yet sensitive to the graph's global organization.

a) Mutual Information Perspective.: Let $G = (V, E, X)$ denote the original graph and f_c a compression mapping that produces a compressed view $G' = f_c(G)$. The mutual information (MI) between G and G' is defined as:

$$I(G; G') = H(G) - H(G|G'),$$

where $H(\cdot)$ denotes Shannon entropy. Compression differs from random perturbation because it removes redundant edges systematically rather than randomly, thus reducing entropy loss. When large dense regions are collapsed into super-nodes, the conditional entropy term $H(G|G')$ is dominated by information within the intra-subgraph structures:

$$H(G|G') \le H(G_{\text{intra}}),$$

where G_{intra} denotes the set of edges and nodes internal to compressed subgraphs. Since these regions are highly cohesive and internally redundant, the lost information is relatively small compared with random edge deletions that destroy inter-community links.

b) Proposition (Informal).: For a graph G with modular or hierarchical organization, let $f_c^{(1)}$ and $f_c^{(2)}$ denote two complementary compression mappings (e.g., k-core and k-truss). If both mappings preserve inter-community connectivity and each subgraph is locally constrained by the size limit λ, then

$$I(G_c^{(1)}; G_c^{(2)}) \ge I(G_{\text{pert}_1}; G_{\text{pert}_2}),$$

where G_{pert_1} and G_{pert_2} are randomly perturbed views. Thus, compression-based contrastive learning retains more informative and topologically faithful relationships between graph views.

IV. Experiments

A. *Datasets*

We use six publicly available datasets summarized in Table I, all of which are commonly used for benchmarking graph classification models [27]. These datasets include four social network datasets (IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY, and COLLAB) and two biological or chemical datasets (D&D and PROTEIN). Each dataset differs in terms of graph size, number of nodes and edges, and class distribution, providing a diverse evaluation scenario.
The datasets vary significantly in size and domain. For instance, REDDIT-BINARY and COLLAB contain large graphs representing user interactions, while D&D and PROTEIN contain more complex molecular structures with rich biological features. This diversity ensures a robust evaluation of our model across different types of graph data.

B. *Baseline:*

1) Graph Pooling Methods: We compare against several hierarchical graph pooling methods, including gPool [28], SAGPool [29], MinCutPool [30], DiffPool [31], ASAP [32], and MPool [33]. These methods learn graph-level representations by progressively reducing the graph size while preserving task-relevant structural information.

- *gPool*: selects important nodes by projecting node features onto a learnable vector and retaining the top-ranked nodes at each pooling layer.
- *SAGPool*: computes node importance scores using graph convolution, allowing the pooling operation to consider both node features and graph topology.
- *MinCutPool*: learns a soft assignment of nodes to clusters by optimizing a normalized min-cut objective together with an orthogonality regularization term.
- *DiffPool*: uses a GNN-based assignment function to group nodes into clusters, forming hierarchical graph representations in an end-to-end manner.
- *ASAP*: forms local clusters using attention-based neighborhood scoring and retains the most informative clusters for graph-level representation learning.

TABLE I: Dataset statistics. $|G|$ is the number of graphs, V_{avg} and E_{avg} denote the average number of nodes and edges per graph, and $|C|$ is the number of classes.

Dataset	$\|G\|$	V_{avg}	E_{avg}	$\|C\|$
D&D	1178	284.32	715.66	2
PROTEIN	1113	39.06	72.82	2
IMDB-BINARY	1000	19.77	96.53	2
IMDB-MULTI	1500	13.00	65.94	3
REDDIT-BINARY	2000	429.63	497.75	2
COLLAB	5000	74.49	2457.78	3

- *MPool*: combines multiple pooling operations in parallel and integrates their outputs to capture complementary structural information.

2) Graph Contrastive Learning Methods: We also compare with representative graph contrastive and self-supervised learning methods, including InfoGraph [1], GraphCL [2], S2GAE [18] CTAug [19], and CI-GCL [23].

- *InfoGraph*: maximizes the mutual information between graph-level representations and representations of local patches, such as nodes and substructures.
- *GraphCL*: generates two augmented views of each graph using operations such as node dropping, edge perturbation, attribute masking, and subgraph sampling, and then aligns their representations with a contrastive objective.
- *S2GAE*: follows a reconstruction-based self-supervised strategy, where part of the graph structure is masked and the model is trained to recover the missing edges.
- *CTAug*: generates augmentations with awareness of cohesive subgraph structures, making it closely related to our method. However, it still relies on stochastic augmentation rather than deterministic compression.
- *CI-GCL*: preserves high-level community structure during graph contrastive learning by using community-invariant topology and feature augmentation. Unlike CI-GCL, which learns augmentation strategies over the original graph, our method explicitly constructs compressed graph views through structural decomposition and supernode-based coarsening.

C. *Experimental Setup*

For each dataset, we perform 10 independent random splits using different seed values. The data is divided into 80% for training, 10% for validation, and 10% for testing. We implement our model using PyTorch and PyTorch Geometric. The model is optimized using the Adam optimizer [34]. Node embedding dimensionality is fixed to 128 across all datasets.

We tune hyperparameters using grid search over the following ranges: learning rate {1e-2, 5e-2, 1e-3, 5e-3, 1e-4, 5e-4}, weight decay {1e-2, 1e-3, 1e-4, 1e-5}, and pooling ratio {1/2, 1/4}. Each model is trained for a maximum of 100K epochs, with early stopping triggered if the validation loss does not improve over 50 consecutive epochs.

For fair comparison with baseline methods, we use GCN as the message-passing backbone in the main experiments, as it is the most commonly adopted architecture in prior work. Unless otherwise specified, we set the subgraph size parameter to $\lambda = 15$ and the contrastive loss weight to $\alpha = 0.6$, as these values yield the best performance in our ablation studies.

D. *Results*

Table II presents the graph classification accuracy of `SAC-GCL` and the baseline methods on six benchmark datasets. Overall, `SAC-GCL` obtains the highest accuracy on five out of six datasets, including IMDB-BINARY,

TABLE II: Graph Classification Performance (Accuracy %). The best results are highlighted in **bold**, the second-best results are underlined.

Method	Model	IMDB-B	IMDB-M	REDDIT-B	COLLAB	D&D	PROTEIN
Supervised Learning	gPool	73.40	50.27	74.70	77.58	75.0	71.9
	SAGPool	73.00	49.43	84.66	70.10	75.7	72.5
	MinCutPool	70.78	52.8	75.67	69.91	76.7	76.3
	DiffPool	68.40	45.62	66.65	74.83	66.9	68.2
	ASAP	72.74	-	-	78.95	76.9	74.2
	MPool	74.20	50.26	84.10	83.62	81.2	79.30
Contrastive Learning	InfoGraph	71.34	47.93	82.50	72.36	72.85	74.09
	GraphCL	71.48	48.11	89.53	72.36	78.62	74.32
	S2GAE	75.76	51.79	87.83	81.02	-	76.37
	CTAug	76.60	51.12	**92.28**	81.72	-	74.10
	CI-GCL	73.85	-	90.80	74.40	79.63	76.50
	SAC-GCL	**78.81**	**52.88**	89.65	**86.70**	**83.43**	**81.42**

IMDB-MULTI, COLLAB, D&D, and PROTEIN. These results indicate that structure-aware graph compression can provide effective contrastive views for learning graph-level representations.

Compared with supervised graph pooling methods, SAC−GCL achieves consistent improvements across the benchmark datasets. On IMDB-BINARY, SAC−GCL obtains 78.81%, compared with 74.20% from MPool. On COLLAB and PROTEIN, SAC−GCL achieves 86.70% and 81.42%, respectively, while MPool obtains 83.62% and 79.30%. These improvements suggest that the proposed compression strategy differs from conventional pooling methods. Rather than only reducing the number of nodes, SAC−GCL constructs compressed views from cohesive subgraphs, allowing the model to preserve local structural information while capturing relationships among higher-level graph components.

Compared with graph contrastive learning baselines, SAC−GCL also demonstrates competitive performance. On IMDB-BINARY and IMDB-MULTI, SAC−GCL achieves 78.81% and 52.88%, improving over CTAug by 2.20 and 1.76 percentage points, respectively. Since these datasets represent social interaction graphs, the results suggest that deterministic compressed views based on k-core and k-truss decompositions can capture useful structural patterns for graph classification. On COLLAB, SAC−GCL reaches 86.70%, outperforming the contrastive baselines reported in the table, including S2GAE, CTAug, and CI-GCL. This result indicates that modeling interactions among compressed cohesive regions can be beneficial for larger interaction networks.

On the biological datasets D&D and PROTEIN, SAC−GCL achieves 83.43% and 81.42%, respectively. These results are higher than the reported pooling and contrastive baselines, suggesting that structure-aware compression is also effective for biological graph classification. In particular, the result on PROTEIN indicates that preserving cohesive substructures before graph-level representation learning can help retain informative structural patterns that may be reduced by direct pooling or random augmentation.

For REDDIT-BINARY, CTAug obtains the highest accuracy with 92.28%, while SAC−GCL achieves 89.65%. Although SAC−GCL does not produce the best result on this dataset, it remains comparable to other contrastive baselines such as GraphCL and CI-GCL and slightly improves over GraphCL. This observation suggests that cohesive-subgraph-aware stochastic augmentation may be particularly effective for this dataset, while the proposed deterministic compression framework provides more consistent performance across different graph domains. Overall, the results show that combining complementary k-core and k-truss compressed views within a contrastive learning framework improves graph representation learning across a diverse set of datasets.

E. Ablation Studies

To better understand the contributions of different components and design choices in our framework, we conduct a series of ablation studies. Below we present the results for three aspects: component analysis, the effect of the contrastive weight, and the effect of subgraph size.

We compare the performance of the full model (SAC−GCL) with three ablated variants: SAC−GCL$_{sup}$ (trained with only supervised loss), SAC−GCL$_{core}$ (using only the k-core encoder), and SAC−GCL$_{truss}$ (using only the k-truss encoder) in Table III. Across all datasets, the full model consistently outperforms its variants. The results highlight the complementary benefits of using both graph compression views (k-core and k-truss) as well as the importance of the contrastive objective. Removing either the compression view or the contrastive loss reduces performance, confirming that each component plays a critical role in the success of our framework.

We further evaluate the influence of the loss weight parameter α, which balances supervised and contrastive learning objectives, in Figure 3. The results indicate that performance peaks when $\alpha = 0.6$. Both smaller and larger values of α reduce accuracy, suggesting that an appropriate trade-off between the two objectives is critical for optimal representation learning. When the contrastive weight is too small, the model

TABLE III: Performance comparisons with the main model (SAC−GCL), the model with supervised loss (SAC−GCL$_{sup}$), the model with k-core view encoder (SAC−GCL$_{core}$), and the model with k-truss view encoder (SAC−GCL$_{truss}$).

Dataset	SAC−GCL	SAC−GCL$_{sup}$	SAC−GCL$_{core}$	SAC−GCL$_{truss}$
IMDB-B	**78.81**	75.10	74.95	75.07
IMDB-M	**52.88**	50.26	50.15	49.88
REDDIT-B	**89.65**	88.20	86.32	87.54
COLLAB	**86.70**	84.25	83.65	83.79
D&D	**83.43**	80.85	79.54	80.32
PROTEIN	**81.42**	77.63	75.77	75.56

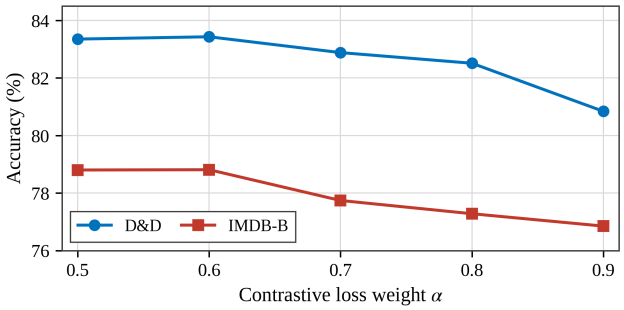


Fig. 3: Performance of SAC−GCL for different values of the contrastive loss weight α on D&D and IMDB-B.

cannot fully benefit from the structural alignment between views; when it is too large, the model becomes dominated by contrastive learning and underutilizes task supervision.

Finally, we study the role of subgraph size in compression views. The results in Figure 4 show that the intermediate values of λ provide the best performance, with $\lambda = 15$ delivering the highest precision in both D&D (83.43%) and IMDB-B (78.81%). When λ is too small (e.g., 10), the model captures less structural information and the performance decreases slightly. As λ increases further, the compressed graphs become too small, leading to severe information loss. This effect is particularly evident when $\lambda = 25$ or $\lambda = 30$, where performance drops sharply to 75.62% and 64.75% on D&D, and 72.36% and 58.54% on IMDB-B, respectively. These results confirm that, while graph compression is beneficial, excessive compression caused by large subgraph sizes removes important structural patterns and harms classification accuracy.

V. Conclusion

In this paper, we propose Structure-aware Compression for Graph Contrastive Learning (SAC−GCL), a framework that employs k-core and k-truss decompositions to generate structure-aware graph views for contrastive learning. By constructing compressed graphs that capture both local cohesiveness and higher-order global structures, SAC−GCL provides a principled alternative to random perturbation–based augmentations. Extensive experiments on six benchmark datasets

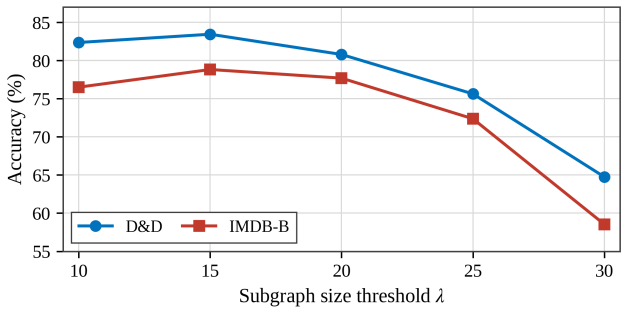


Fig. 4: Performance of SAC−GCL for different subgraph size thresholds λ on D&D and IMDB-B.

demonstrated that our method outperforms state-of-the-art pooling and contrastive baselines on five of the six datasets, confirming the effectiveness of compression-driven view generation. Ablation studies further highlighted the complementary benefits of k-core and k-truss decompositions and the importance of balancing supervised and contrastive objectives. Overall, our findings establish graph compression as a powerful strategy for enhancing self-supervised learning in graphs. Future work will extend this framework to dynamic and heterogeneous graphs, as well as explore integration with generative objectives to further improve scalability and representation quality.

References

[1] F.-Y. Sun, J. Hoffman, V. Verma, and J. Tang, "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," in *ICLR*, 2019.

[2] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *NeurIPS* (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds.), vol. 33, pp. 5812–5823, Curran Associates, Inc., 2020.

[3] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," in *ICML*, pp. 3451–3461, 2020.

[4] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Deep graph contrastive representation learning," *arXiv preprint arXiv:2006.04131*, 2020.

[5] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep Graph Infomax," in *International Conference on Learning Representations*, 2019.

[6] W. Ju, Y. Gu, Z. Mao, Z. Qiao, Y. Qin, X. Luo, H. Xiong, and M. Zhang, "Gps: Graph contrastive learning via multi-scale augmented views from adversarial pooling," *Science China Information Sciences*, vol. 68, no. 1, 2025.

[7] Z. Du, T. Wang, Z. Yang, J. Chen, Z. Duan, and S. Zhao, "Graph contrastive learning via coarsening: A time and memory efficient approach," *Knowledge-Based Systems*, vol. 317, p. 113398, 2025.

[8] Y. Liu, T. Safavi, A. Dighe, and D. Koutra, "Graph summarization methods and applications: A survey," *ACM computing surveys (CSUR)*, vol. 51, no. 3, pp. 1–34, 2018.

[9] E. Akbas and P. Zhao, "Truss-based community search: a truss-equivalence based indexing approach," *Proceedings of the VLDB Endowment*, vol. 10, no. 11, pp. 1298–1309, 2017.

[10] W. Khaouid, M. Barsky, V. Srinivasan, and A. Thomo, "K-core decomposition of large networks on a single pc," *Proceedings of the VLDB Endowment*, vol. 9, no. 1, pp. 13–23, 2015.

[11] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," In Proceedings of the ICLR, 2017.

[12] J. Chen and G. Kou, "Attribute and structure preserving graph contrastive learning," in *AAAI Conference*, vol. 37, pp. 7024–7032, 2023.

[13] U. Fang, J. Li, N. Akhtar, M. Li, and Y. Jia, "Gomic: Multi-view image clustering via self-supervised contrastive heterogeneous graph co-learning," *World Wide Web*, vol. 26, no. 4, pp. 1667–1683, 2023.

[14] J. Zeng and P. Xie, "Contrastive self-supervised learning for graph classification," in *AAAI conference*, vol. 35, pp. 10824–10832, 2021.

[15] S. Zhang, Z. Hu, A. Subramonian, and Y. Sun, "Motif-driven contrastive learning of graph representations," *IEEE TKDE*, vol. 36, no. 8, pp. 4063–4075, 2024.

[16] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," *ICLR*, 2018.

[17] W. Ju, Y. Wang, Y. Qin, Z. Mao, Z. Xiao, J. Luo, J. Yang, Y. Gu, D. Wang, Q. Long, *et al.*, "Towards graph contrastive learning: A survey and beyond," *arXiv preprint arXiv:2405.11868*, 2024.

[18] Q. Tan, N. Liu, X. Huang, S.-H. Choi, L. Li, R. Chen, and X. Hu, "S2gae: Self-supervised graph autoencoders are generalizable learners with graph masking," in *ACM WSDM*, pp. 787–795, 2023.

[19] Y. Wu, L. Wang, X. Han, and H.-J. Ye, "Graph contrastive learning with cohesive subgraph awareness," in *Proceedings of the ACM Web Conference*, 2024.

[20] Y. Lin, G. Lyu, H. Cai, D.-B. Wang, H. Wang, and Z. Yang, "Simplified graph contrastive learning model without augmentation," *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[21] N. Hossain, A. Al Thaki, M. Mamun-Or-Rashid, and M. M. Khan, "Graph contrastive learning: A comprehensive review of methodologies, applications, and future directions," *IEEE Access*, 2026.

[22] P. Qin, Y. Lu, W. Chen, D. Li, and G. Feng, "Aagcn: An adaptive data augmentation for graph contrastive learning," *Pattern Recognition*, vol. 163, p. 111471, 2025.

[23] S. Tan, D. Li, R. Jiang, Y. Zhang, and M. Okumura, "Community-invariant graph contrastive learning," in *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, JMLR.org, 2024.

[24] C. Li, Y. Guo, Z. Tang, W. Li, Y. Chen, J. Cao, Y. Zhang, and J. Jiang, "Mhgcl: Combining motif and homogeneity in graph contrastive learning," *WWW*, vol. 28, no. 4, 2025.

[25] X. Zhang, Q. Tan, X. Huang, and B. Li, "Graph contrastive learning with personalized augmentation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 11, pp. 6305–6316, 2024.

[26] Y. Lai, X. Lin, R. Yang, and H. Wang, "Efficient topology-aware data augmentation for high-degree graph neural networks," in *ACM SIGKDD*, pp. 1463–1473, 2024.

[27] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," *arXiv preprint arXiv:2007.08663*, 2020.

[28] H. Gao and S. Ji, "Graph u-nets," in *international conference on machine learning*, pp. 2083–2092, PMLR, 2019.

[29] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *International Conference on Machine Learning*, vol. 97, 2019.

[30] F. M. Bianchi, D. Grattarola, and C. Alippi, "Spectral clustering with graph neural networks for graph pooling," in *International Conference on Machine Learning*, pp. 874–883, PMLR, 2020.

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

[32] E. Ranjan, S. Sanyal, and P. Talukdar, "Asap: Adaptive structure aware pooling for learning hierarchical graph representations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 5470–5477, 2020.

[33] M. I. K. Islam, M. Khanov, and E. Akbas, "Mpool: Motif-based graph pooling," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 105–117, Springer, 2023.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd ICLR*, 2015.