Community-Aware Graph Transformer: Preserving Community Semantics for Effective Global Aggregation

Yutai Duan[1], Jie Liu[1]✉, Jianhua Wu[1], and Jialin Liu[1]

College of Artificial Intelligence, Nankai University,Tianjin 300350, CHINA
{ytduan, wujianhua, 2120220573}@mail.nankai.edu.cn jliu@nankai.edu.cn

Abstract. Graph Transformers (GTs) address the locality limitation of traditional GNNs, which aggregate only local neighbor information, by leveraging global attention. However, they suffer from two significant issues: neglecting community structures and information over-squeezing. In this paper, we first identify these two problems and propose a Community-Aware Graph Transformer (CoGT) to solve them. CoGT introduces a novel node-community-global hierarchical aggregation framework. This design preserves community-level semantics while reducing the volume of aggregated information, alleviating the over-squeezing problem. CoGT first employs a two-stage positional encoding to identify latent communities and enhance semantic consistency. Then, a hierarchical and parallel transformer computation method based on community representations facilitates global information interaction. Furthermore, we enable community-wise parallel attention computation, improving computational efficiency. Experimental results demonstrate that CoGT outperforms existing methods across multiple real-world datasets.

Keywords: Graph transformer · Graph representation learning

1 Introduction

The core objective of Graph Representation Learning (GRL) [27] is to capture both topological structure and semantic information by modeling node interactions. Traditional Graph Neural Networks (GNNs) [24], limited by local neighborhood aggregation, struggle to uncover long-range dependencies [3, 23]. Graph Transformers (GTs) [22, 26], using global attention mechanisms [16], enable interactions across all nodes, significantly enhancing node representations. This global aggregation can overcome local neighborhood limitations and identify distant nodes with potential benefits [25]. However, many studies have focused on improving the efficiency of GTs while ignoring the problems of global aggregation in GRL: the loss of community structure semantics and information over-squeezing.

Loss of Community Structure Semantics (Section 2.1): Graph data often exhibit a hierarchical community structure [14], an inherent property of

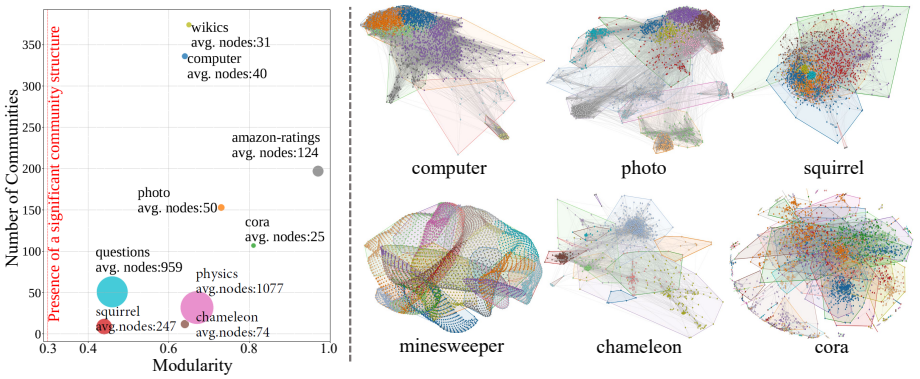
graph data. Therefore, to describe a graph from a global rather than a local perspective, community-based semantics best capture its nature [7]. GTs introduce global perspectives into node representations via global information exchange. However, existing GTs use homogeneous global attention that overlooks hierarchical community semantics, undermining the graph's global structure. This leads to semantic inconsistencies across hierarchical levels, where core and peripheral nodes are treated uniformly, failing to capture their distinct roles. Moreover, this indiscriminate aggregation of large volumes of node features introduces an additional challenge: information over-squeezing, which diminishes the quality of node representations by blending informative signals with excessive noise.

Information Over-Squeezing (Section 2.2): Global attention aggregates information from all nodes simultaneously. This may lead to over-squeezing, where high-weight noise mixes with low-weight signals, resulting in the loss of important information. This disrupts the hierarchical structure of the central node, flattening the influence of all nodes on it. This is similar to over-squashing in GNN [18], which is essentially the problem of weakening or losing important signals caused by receiving too much information. From a structural entropy perspective [13], this is equivalent to using a global structure to represent each node, which increases the structural entropy of node representations. As a result, node features learned in a high-entropy space are suboptimal. While global aggregation uncovers latent information, a balance must be struck between global aggregation and information over-squeezing.

To this end, we propose a Community-Aware Graph Transformer (CoGT), which features two key designs: Learnable Two-stage Positional Encoding (TiCoding) and Hierarchical and Parallel Transformer Computation (HPTC). CoGT achieves hierarchical, semantics-aware global aggregation by identifying latent communities in graphs. Specifically, TiCoding first injects global positional information into nodes via a local structure encoder and leverages the result to detect potential communities. Each identified community is assigned a unique encoding to model community semantics explicitly. HPTC extracts community representations and computes the hybrid attention that integrates information from central nodes, intra-community nodes, and inter-community representations to update node embeddings. We also crafted a parallelized attention computation process to improve efficiency. This design not only explicitly models community semantics but also replaces large-scale node features with community representations for aggregating, alleviating the over-squeezing issue.

The contributions of this paper can be summarized as follows:

- We propose CoGT that mitigates community semantics loss and information over-squeezing via latent community discovery and hierarchical aggregation with parallel and efficient attention computation.
- We design TiCoding for effective multi-semantic positional encoding and introduce HPTC for efficient hierarchical aggregation, enabling community-aware modeling while alleviating over-squeezing.
- We conduct an empirical study revealing community semantics loss and over-squeezing as limitations in existing GTs.



(a) Number, density, and size of communities (b) Visualization of communities in real-world datasets

Fig. 1. The community structure is prominently present in graphs. (a) Statistics of community-related metrics. Modularity reflects the density of communities within a graph to some extent, while the size of each point represents the average community size. (b) Visualization for different communities in datasets by coloring the nodes according to their respective communities.

– Extensive experiments show that CoGT achieves superior performance and by explicitly modeling community-level semantics.

2 Empirical Investigation

2.1 Community Structures Prevalent in Graphs

To highlight the importance of explicitly modeling community-level semantics, we analyzed key community-related metrics from common graph datasets, including the number of communities N_C, the average number of nodes in communities N_{avg}, and modularity [14] $Q = \frac{1}{2e}\sum_{i,j}\left(A_{ij} - \frac{d_i d_j}{2e}\right)\delta(c_i, c_j)$, where e is the edge numbers, A_{ij} is the adjacency matrix, d_i is the degree of node v_i, and $\delta(c_i, c_j)$ equals 1 if v_i and v_j are in the same community and 0 otherwise. The results are shown in Fig. 1(a). We use the Louvain algorithm [14] for community detection on common graph datasets and count the number of communities. We set the community granularity to 1.2, ensuring that the algorithm identifies larger communities, rather than smaller, loosely connected ones. Based on the community partitioning results, we calculated modularity Q to quantify the extent of the community structure. We also visualized the communities in datasets by coloring the nodes according to their respective communities in Fig. 1(b).

The results in Fig. 1(a) show that: (1) Common graph datasets typically exhibit dense community structures, with each graph containing tens to hundreds of communities. (2) Generally, when the modularity $Q > 0.3$, the community structure can be considered significant. All datasets exceed this threshold, with

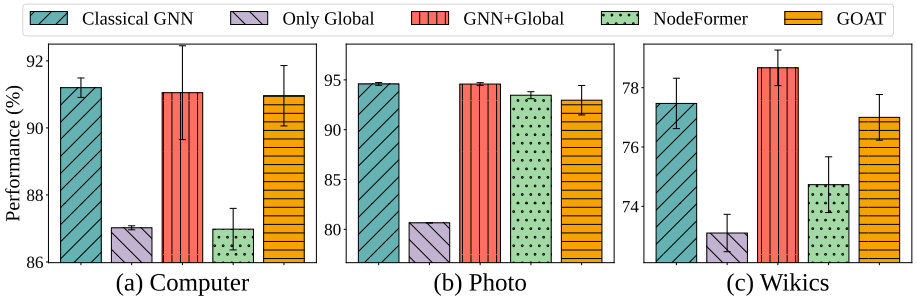


Fig. 2. A set of comparative experiments to demonstrate the issue of information over-squeezing in GTs. Global node aggregation alone performs significantly worse than GNNs, emphasizing important neighbors, indicating that global aggregation suffers from the information over-squeezing problem.

most having Q values greater than 0.6, indicating that these graphs feature significant community structures. (3) The calculation shows that $N_C \times N_{avg} \geq$ total number of nodes N, with some datasets approaching $2N$. This indicates that the community structure covers nearly all nodes in the graph, with some nodes playing roles in multiple communities, which supports the discovery of latent information through global aggregation. (4) From the visualizations, we can easily observe significant community structures, with dense connections generally present within the communities.

In summary, we conclude that community-level semantics are a crucial and often overlooked level of semantic granularity. However, existing methods often overlook modeling the community structure, resulting in the loss of community-level semantics during the global aggregation process.

2.2 Information Over-Squeezing in GTs.

We conducted empirical studies to validate that the use of simple global aggregation leads to the issue of information over-squeezing. We selected the best-performing model from GCN, GAT, and GraphSAGE as the baseline performance of classical GNNs. We compared it with models using only transformer-based global aggregation, classical GNNs with global aggregation, and two prominent GTs. The results are shown in Fig. 2.

Our findings are as follows: (1) Information Over-Squeezing Issue: Using only GNNs significantly outperforms global aggregation alone. Compared to GNNs, purely global aggregation injects excessive information into each node, making their structural representations overly similar. This leads to a low signal-to-noise ratio, making node differentiation challenging. (2) In contrast, GNNs emphasize crucial information through local aggregation while inherently preserving some community structures, resulting in more effective representations. (3) When naively combining GNNs with global aggregation, performance degradation is observed in two out of three datasets, indicating that global aggregation

exacerbates the over-squeezing issue. This phenomenon is also evident in the performance of state-of-the-art GT models, such as NodeFormer and GOAT in the figure, suggesting that directly integrating global aggregation suffers from severe information compression. (4) However, the performance improvement on the enhanced WikiCS dataset suggests that global aggregation can be beneficial by identifying latent yet useful node relationships.

Based on these observations, we need to find a balance between global aggregation and information over-squeezing, ensuring that the benefits of global aggregation are leveraged while avoiding issues like over-squeezing.

3 Related Work

3.1 Graph Representation Learning

The goal of GRL is to integrate node or graph-level features with supervision signals through an end-to-end training process [27], generating discriminative low-dimensional representations. Traditional approaches primarily include matrix factorization, e.g., Laplacian Eigenmaps [1], and random walk-based shallow models, e.g., DeepWalk [15], Node2Vec [8]. While these methods effectively capture the statistical properties of graph structures, they suffer from a strong dependence on feature engineering and lack adaptive parameters, making it difficult to jointly optimize representations for downstream tasks.

The emergence of GNNs [24] marked a significant shift toward deep learning-based approaches in GRL. Message-passing GNNs [9], e.g., GCN [11] and GAT [19], aggregate neighborhood features iteratively to capture local structural patterns. Specifically, GCNs [3] employ spectral convolution operations to establish a paradigm for information propagation, whereas GATs [20] introduce attention mechanisms to enhance the interpretability of the aggregation process. However, classical GNNs are fundamentally constrained by their local aggregation assumption, limiting their ability to model long-range and latent dependencies [6]. This limitation is particularly pronounced in domains with complex topological structures, such as social networks and molecular graphs. To overcome the bottleneck of local neighborhood aggregation, Transformer architectures have been introduced into GRL, leading to the development of GTs [16, 22]. GTs leverage global attention mechanisms to compute attention between any pair of nodes [26, 2], effectively capturing long-range and latent dependencies in graph structures.

3.2 Transformer Models

Transformer models [5] were proposed for natural language processing, where its key innovation lies in eliminating the sequential dependencies inherent in traditional recurrent neural networks. Instead, it leveraged self-attention mechanisms to model global interactions between sequence elements. The success of ViT [10] in computer vision, surpassing the performance of Convolutional Neural Networks, highlighted the generalizability of attention mechanisms across different modalities. This realization has driven the extension of Transformers to GRL.

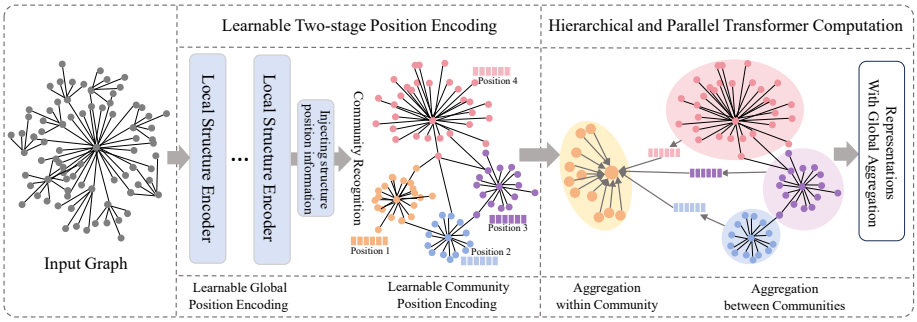


Fig. 3. The CoGT framework.

GTs treat all nodes in the graph as interactive attention units [21], allowing direct computation of attention weights between any pair of nodes [4]. Current research efforts focus on enhancing the scalability of Transformer models for large-scale graphs [12, 17, 22]. However, existing approaches often overlook two critical challenges associated with global attention aggregation in GRL: (1) excessive reliance on node-level interactions may weaken the semantic representation at the community level; (2) global attention can lead to an "Information Over-Squeezing" effect, increasing noise interference and diminishing the effectiveness of crucial information.

4 Methodology

4.1 Overview

The goal of CoGT is to explicitly model community-level semantics during the global aggregation step through hierarchical aggregation. By using community-level features instead of a large number of node features in attention-based aggregation, CoGT effectively mitigates the issue of information over-squeezing.

The CoGT framework is shown in Fig. 3. To model a community-level semantic in the global aggregation process, we first apply a two-stage positional encoding to all nodes in the graph. In the first stage, a local structure encoder is used to encode the position of each node. Based on this encoding, nodes are then communityed, and each community is assigned a community-level positional encoding to distinguish different communities. Next, the communityed feature matrix is passed to the HPTC layer for attention-based computation. This layer not only enables global aggregation but also mitigates excessive information squeezing. Furthermore, its design inherently supports parallel computation at the community level, significantly enhancing computational efficiency.

4.2 Two-stage Positional Encoding

For a graph $\mathcal{G}(E, X)$ with edge set E and node features X, where the total number of nodes is N, we first apply a local structure encoder to inject positional

information into each node's representation:

$$\mathbf{X} = \text{LocalStructureEncoder}(\mathcal{G}(E, X)), \tag{1}$$

where $\mathbf{X} \in \mathbb{R}^{N \times F}$ denotes the encoded feature matrix, and F is the hidden dimension. We adopt graph convolutional networks as the local structure encoder.

To ensure a differentiable community assignment process and prevent excessive community sizes that may cause memory overflow and semantic imbalance, we design a classification-based assignment strategy with a constraint on the "average community size $N_m = \lceil \frac{N}{M} \rceil$", where M is the number of communities. We formulate node assignment as a classification problem and apply the `softmax` function to normalize the probability $Z'_{i,j}$ of assigning node i to community j:

$$\mathbf{Z} = \mathbf{X}\mathbf{W}_{\text{C}}, \quad Z'_{i,j} = \frac{\exp(\mathbf{Z}_{i,j})}{\sum_{k=1}^{M} \exp(\mathbf{Z}_{i,k})}, \tag{2}$$

where $\mathbf{W}_{\text{C}} \in \mathbb{R}^{F \times M}$ is a learnable matrix, and $\mathbf{Z}$ is the assignment score matrix. We enforce a fixed number of nodes, N_m, per community by sorting nodes based on assignment probabilities and selecting the top N_m nodes. Then, we assign a learnable community position embedding $\mathbf{E}_m \in \mathbb{R}^F$ to each community C_m to distinguish its semantics and enhance intra-community node consistency:

$$\mathbf{X}'_i = \mathbf{X}_i + \mathbf{E}_m (i \in C_m), \tag{3}$$

Based on node-community affiliations, we obtain the reorganized feature matrix $\mathbf{X}' \in \mathbb{R}^{M \times N_m \times F}$, which incorporates twofold positional semantics. Here, $\mathbf{X}'[m, :, :]$ represents the features of all nodes in C_m.

4.3 Hierarchical and Parallel Transformer Computation

We employ hierarchical aggregation to achieve global feature interaction while mitigating the information over-squeezing problem. First, we compute the query, key, and value matrices for each community C_m. We directly feed $\mathbf{X}'$ into linear layers to compute the $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ in parallel, which reduces memory peak usage because the tensor size of each community is much smaller than the original $\mathbf{X}$:

$$\mathbf{Q}_m = \mathbf{W}_Q \mathbf{X}'_m, \quad \mathbf{K}_m = \mathbf{W}_K \mathbf{X}'_m, \quad \mathbf{V}_m = \mathbf{W}_V \mathbf{X}'_m, \tag{4}$$

where $\mathbf{W}_{(\cdot)}$ are learnable weights, F is the hidden dimension, $\mathbf{X}'_m$ is $\mathbf{X}'[m, :, :]$.

To mitigate the information over-squeezing issue caused by the $O(N^2)$ level of feature interactions among all nodes globally, we introduce community-level features to replace node-level interactions outside the community. Nodes within the same community interact via node features, while nodes from different communities engage through a set of community-level features. Since community features have higher semantic consistency and are much fewer in number compared to node features, they effectively prevent information over-squeezing and high noise within node features, thereby improving the efficiency of feature interactions.

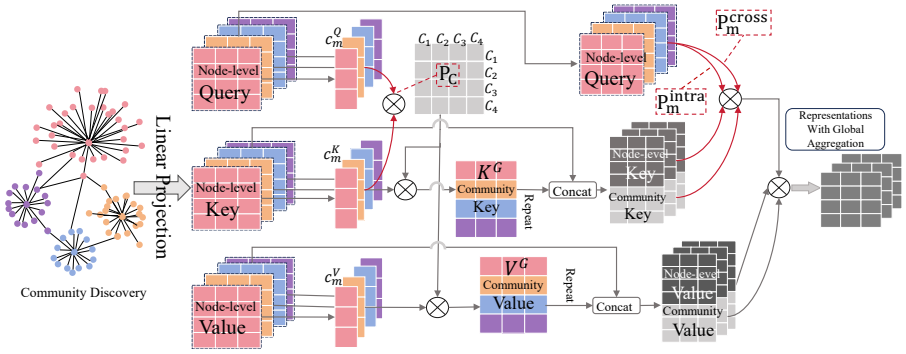


Fig. 4. HPTC leverages community features to enable parallel attention computation at the community level. This design not only incorporates community semantics but also enhances computational efficiency.

We compute the community-level query ($\mathbf{C}^Q$), key ($\mathbf{C}^K$), and value ($\mathbf{C}^V$) matrices, which are obtained by applying an aggregation function to the features of each community, followed by concatenation:

$$\mathbf{c}_m^Q = \mathrm{Agg}(\mathbf{Q}_m), \quad \mathbf{c}_m^K = \mathrm{Agg}(\mathbf{K}_m), \quad \mathbf{c}_m^V = \mathrm{Agg}(\mathbf{V}_m), \tag{5}$$

$$\mathbf{C}^Q = \Big\|_{m=1}^{M}(\mathbf{c}_m^Q), \quad \mathbf{C}^K = \Big\|_{m=1}^{M}(\mathbf{c}_m^K), \quad \mathbf{C}^V = \Big\|_{m=1}^{M}(\mathbf{c}_m^V), \tag{6}$$

where $\mathbf{c}_m^Q, \mathbf{c}_m^K, \mathbf{c}_m^V$ is the query, key, and value for community m. $\mathrm{Agg}(\cdot)$ denotes to the aggregation operation, and $\Big\|_{m=1}^{M}$ denotes concatenation. To improve computational efficiency, we employ simple aggregation methods such as mean pooling.

Next, we refine the community-level key and value representations using community-level attention, allowing the community features to be optimized and enriched in semantic information. The community-level attention matrix $\mathbf{P}_\mathrm{C} \in \mathbb{R}^{M\times M}$ is computed as:

$$\mathbf{P}_\mathrm{C} = \sigma\left(\frac{\mathbf{C}^Q {\mathbf{C}^K}^\top}{\sqrt{F}}\right), \quad \mathbf{K}^G = \mathbf{P}_\mathrm{C}\mathbf{C}^K, \quad \mathbf{V}^G = \mathbf{P}_\mathrm{C}\mathbf{C}^V. \tag{7}$$

Here, σ denotes the activation function, and we use tanh as the activation function in this work. The use of `tanh` for activation is to maintain computational efficiency at the $O(N)$ level, rather than the common N-level of `softmax`. Furthermore, positive or negative attentional values can also be considered to aggregate or depress the features of corresponding communities.

To establish global semantic dependencies for each node, we compute intra-community and cross-community attention as follows:

$$\mathbf{P}_m^{\mathrm{intra}} = \sigma\left(\frac{\mathbf{Q}_m {\mathbf{K}_m}^\top}{\sqrt{F}}\right), \quad \mathbf{P}_m^{\mathrm{cross}} = \sigma\left(\frac{\mathbf{Q}_m {\mathbf{K}^G}^\top}{\sqrt{F}}\right), \tag{8}$$

where $\mathbf{P}_m^{\text{intra}} \in \mathbb{R}^{N_m \times N_m}$ captures interactions between nodes within the same community, and $\mathbf{P}_m^{\text{cross}} \in \mathbb{R}^{N_m \times M}$ models interactions between nodes in community m and all communities globally. Notably, both $\mathbf{P}_m^{\text{intra}}$ and $\mathbf{P}_m^{\text{cross}}$ can be efficiently computed in a single step by concatenating $\mathbf{K}_m$ and $\mathbf{K}^G$. Moreover, since we reshape the feature matrix into 3 dimensions, $\mathbf{P}_m^{\text{intra}}$ and $\mathbf{P}_m^{\text{cross}}$ computation for M communities can be performed in parallel simultaneously.

Finally, we apply the attention scores to both node-level and community-level value matrices to generate the final node representations for C_m:

$$\mathbf{H}_m = [\mathbf{P}_m^{\text{cross}} \| \mathbf{P}_m^{\text{cross}}] \cdot [\mathbf{V}_m \| \mathbf{V}^G], \tag{9}$$

where $[\cdot\|\cdot]$ denotes a concatenation operation along columns, and each row of $\mathbf{H}_m \in \mathbb{R}^{N_m \times F}$ corresponds to the representation of a node in community m. These representations incorporate information from other nodes within the same community as well as global information from all M communities. Finally, we restore the original order of the feature matrix through the node indexes generated by the community assignment. It is worth noting that the computation of all M community representation matrices is performed in parallel, significantly reducing both computational time and peak memory usage.

5 Theoretical Justification

5.1 Global Feature Interaction via Cross-Community Attention

Although the model avoids explicit attention computation for all node pairs through community partitioning, the hierarchical aggregation mechanism still enables information interaction between any two nodes in the input graph. For any two nodes $v_i \in C_m$ and $v_j \in C_n$:

If v_i and v_j are in the same community (i.e., $m = n$), their interaction is directly modeled by node-level attention. The weight of v_i on v_j is defined as:

$$\mathbf{P}_m^{\text{intra}}(i,j) = \sigma\left(\frac{\mathbf{Q}_m[i,:]\mathbf{K}_m[j,:]^\top}{\sqrt{F}}\right), \tag{10}$$

where $\mathbf{P}_m^{\text{intra}}(i,j)$ directly encode the semantic similarity between v_i and v_j. The local aggregation result for node v_i is $\mathbf{h}_i^{\text{local}} = \sum_{j \in C_m} \mathbf{P}_m^{\text{intra}}(i,j)\mathbf{V}_m[j,:]$, explicitly capturing fine-grained associations within the community.

If the nodes belong to different communities (i.e., $m \neq n$), their interaction is indirectly achieved through community-level features. The node features of community C_m are aggregated into a community feature $\mathbf{c}_m^V$, which is further weighted by the inter-community attention matrix $\mathbf{P}_{\text{C}} \in \mathbb{R}^{M \times M}$ to form the updated global feature $\mathbf{V}^G$. The elements of $\mathbf{P}_{\text{C}}$ are computed as:

$$\mathbf{P}_{\text{C}}(m,n) = \sigma\left(\frac{\mathbf{c}_m^Q \mathbf{c}_n^{K\top}}{\sqrt{F}}\right), \tag{11}$$

where $\mathbf{P}_{\mathrm{C}}(m,n)$ represents the attention weight of community C_m on C_k. Each row $\mathbf{v}_n^G \in \mathbb{R}^F$ of $\mathbf{V}^G$ denotes the higher-order representation of community C_n in the global semantic space. The interaction between cross-community nodes v_i and v_j is achieved through the cross-community attention matrix $\mathbf{P}_m^{\text{cross}} \in \mathbb{R}^{N_m \times M}$, whose elements are defined as:

$$\mathbf{P}_m^{\text{cross}}(i,n) = \sigma\left(\frac{\mathbf{Q}_m[i,:]\mathbf{K}^G[n,:]^\top}{\sqrt{F}}\right), \tag{12}$$

where $\mathbf{P}_m^{\text{cross}}(i,k)$ represents the weight of $v_i \in C_m$ on community C_k. The global aggregation result for v_i is $\mathbf{h}_i^{\text{global}} \propto \sum_{k=1}^{M} \mathbf{P}_m^{\text{cross}}(i,k)\mathbf{V}^G[k,:]$. Notably, the information of node $v_j \in C_n$ is transmitted to v_i through its community's $\mathbf{V}^G[k,:]$, i.e., $\mathbf{h}_i^{\text{global}} \propto \mathbf{P}_m^{\text{cross}}(i,n)\mathbf{V}^G[n,:]$. Since $\mathbf{V}^G[n,:]$ is the updated $\mathbf{c}_n^V$, this process indirectly establishes cross-community dependencies.

From the perspective of gradient propagation, regardless of whether nodes belong to the same community, their gradients can be backpropagated through local or global paths. For $v_j \in C_n$, its gradient contribution to $v_i \in C_m$ is:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{X}_j} = \begin{cases} \frac{\partial \mathcal{L}}{\partial \mathbf{h}_i^{\text{local}}} \cdot \frac{\partial \mathbf{P}_m^{\text{intra}}(i,j)}{\partial \mathbf{X}_j} & m = n \\ \frac{\partial \mathcal{L}}{\partial \mathbf{h}_i^{\text{global}}} \cdot \frac{\partial \mathbf{P}_m^{\text{cross}}(i,n)}{\partial \mathbf{K}^G[n,:]} \cdot \frac{\partial \mathbf{K}^G[n,:]}{\partial \mathbf{c}_n^K} \cdot \frac{\partial \mathbf{c}_n^K}{\partial \mathbf{X}_j} & m \neq n \end{cases}. \tag{13}$$

This indicates that the hierarchical aggregation mechanism still enables information interaction between any two nodes in the input graph.

5.2 Reducing Structural Entropy via Community Segmentation

This study introduces structural entropy to quantify the structural information encoded in node representations. Traditional graph transformers employ global attention to enable full-node interactions, effectively leveraging graph structures but leading to high structural entropy. The fully connected interaction mechanism makes nodes highly susceptible to noise, particularly in large-scale graphs, where irrelevant interactions dilute critical semantic information.

To address this issue, we propose a hierarchical propagation strategy based on community partitioning. First, fine-grained intra-community attention is performed to capture local dependencies while avoiding inter-community noise. Then, global information exchange is achieved through the aggregation of community representations, which filters noise while preserving essential semantics. This strategy optimizes the information propagation path from $N-1$ to $N_m - 1 + N$, ensuring $N_m - 1 + M \ll N - 1$, significantly reducing structural entropy. In conventional methods, structural entropy is determined by interactions between all node pairs, formulated as:

$$\mathcal{H}_{\text{global}} \propto -\sum_{\text{all nodes } N} p \log p, \tag{14}$$

Table 1. Statistics of the datasets used in this paper.

Dataset	Type	# Nodes	# Edges	# Features	Classes	Metric
Amazon-Computer	Homophily	13,752	245,861	767	10	Accuracy
Amazon-Photo	Homophily	7,650	119,081	745	8	Accuracy
Coauthor-CS	Homophily	18,333	81,894	6,805	15	Accuracy
Cora	Homophily	2,708	5,278	1,433	7	Accuracy
WikiCS	Homophily	11,701	216,123	300	10	Accuracy
ogbn-proteins (large)	Homophily	132,534	39,561,252	8	2	ROC-AUC
Squirrel	Heterophily	2,223	46,998	2,089	5	Accuracy
Chameleon	Heterophily	2,277	31,421	2,325	5	Accuracy
Amazon-Ratings	Heterophily	24,492	183,831	300	5	ROC-AUC
Minesweeper	Heterophily	10,000	39,402	6	2	ROC-AUC
Questions	Heterophily	48,921	118,540	301	2	ROC-AUC
pokec (large)	Heterophily	1,632,803	30,622,564	65	2	Accuracy

where p represents the dependency strength between any two nodes, resulting in a high entropy value. In contrast, our method decomposes entropy into local interaction entropy (within communities) and global propagation entropy (between condensed community representations):

$$\mathcal{H}_{\text{new}} \propto -\sum_{\text{intra-community } N_m} p \log p - \sum_{\text{inter-community } M} q \log q, \tag{15}$$

where p represents the dependency strength between two nodes in one community, analogous to $\mathbf{P}^{\text{intra}}$, and q represents the dependency strength between community features, analogous to $\mathbf{P}^{\text{cross}}$. Consequently, CoGT updates node representations using only intra-community node information and the coarse-grained, sparse linkage structure between communities. This approach results in significantly lower structural entropy compared to methods that perform interactions at the global node level, mitigating the issue of information over-squeezing.

6 Experiments

6.1 Experimental Setup

For the datasets, we selected several commonly used homophilic and heterophilic graphs, including large graphs. The specific data set statistics are shown in Table 1. The training, validation, and test splits for each dataset follow the official partitioning method provided for that dataset. Except for the ogbn-proteins, Amazon-Ratings, Questions, and Minesweeper datasets, which are evaluated using the ROC-AUC metric, all other datasets are evaluated using accuracy. This setting follows the evaluation protocol used in previous works. ROC-AUC (Area Under the Receiver Operating Characteristic Curve) measures a model's ability to distinguish between positive and negative classes and is particularly useful in imbalanced classification scenarios. In the preprocessing stage, we first convert the graph into an undirected graph and then add self-loops.

Table 2. Node classification results on homophilic datasets. The top **first**, **second**, and **third** results are highlighted.

Model	Computer	Photo	CS	Cora	WikiCS	ogbn-proteins
GCN	89.65 ± 0.52	92.70 ± 0.20	92.92 ± 0.12	81.60 ± 0.40	77.47 ± 0.85	72.51 ± 0.35
GraphSAGE	91.20 ± 0.29	94.59 ± 0.14	93.91 ± 0.13	82.68 ± 0.47	74.77 ± 0.95	77.68 ± 0.20
GAT	90.78 ± 0.13	93.87 ± 0.11	93.61 ± 0.14	83.00 ± 0.70	76.91 ± 0.82	72.02 ± 0.44
GraphGPS	91.19 ± 0.54	95.06 ± 0.13	93.93 ± 0.12	82.84 ± 1.03	78.66 ± 0.49	76.83 ± 0.26
NAGphormer	91.22 ± 0.14	**95.49** ± 0.11	**95.75** ± 0.09	82.12 ± 1.18	77.16 ± 0.72	73.61 ± 0.33
Exphormer	**91.47** ± 0.17	95.35 ± 0.22	94.93 ± 0.01	82.77 ± 1.38	78.54 ± 0.49	74.58 ± 0.26
NodeFormer	86.98 ± 0.62	93.46 ± 0.35	**95.64** ± 0.22	82.20 ± 0.90	74.73 ± 0.94	**77.45** ± 1.15
GOAT	90.96 ± 0.90	92.96 ± 1.48	94.21 ± 0.38	**83.18** ± 1.27	77.00 ± 0.77	74.18 ± 0.37
Polynormer	**93.68** ± 0.21	**96.46** ± 0.26	95.53 ± 0.16	**83.25** ± 0.93	**80.10** ± 0.67	**78.97** ± 0.47
CoGT	**94.01** ± 0.28	**96.56** ± 0.31	**95.93** ± 0.20	**83.98** ± 0.76	**81.23** ± 0.24	**81.33** ± 0.37

Table 3. Node classification results on heterophilic datasets. The top **first**, **second**, and **third** results are highlighted.

Model	amazon-ratings	minesweeper	squirrel	chameleon	questions	pokec
GCN	48.70 ± 0.63	89.75 ± 0.52	38.67 ± 1.84	41.31 ± 3.05	76.09 ± 1.27	75.45 ± 0.17
GraphSAGE	53.63 ± 0.39	93.51 ± 0.57	36.09 ± 1.99	37.77 ± 4.14	76.44 ± 0.62	75.63 ± 0.38
GAT	52.70 ± 0.62	**93.91** ± 0.35	35.62 ± 2.06	39.21 ± 3.08	**76.79** ± 0.71	72.23 ± 0.18
GraphGPS	53.10 ± 0.42	90.63 ± 0.67	39.67 ± 2.84	40.79 ± 4.03	71.73 ± 1.47	OOM
NAGphormer	51.26 ± 0.72	84.19 ± 0.66	39.99 ± 3.90	**44.39** ± 3.93	68.17 ± 1.53	**76.59** ± 0.25
Exphormer	**53.51** ± 0.46	90.74 ± 0.53	**40.41** ± 2.42	**42.06** ± 2.44	73.94 ± 1.06	OOM
NodeFormer	43.86 ± 0.35	86.71 ± 0.88	38.52 ± 1.57	34.73 ± 4.14	74.27 ± 1.46	71.00 ± 1.30
Polynormer	**54.81** ± 0.49	**97.46** ± 0.36	**41.97** ± 2.14	41.97 ± 3.14	**78.92** ± 0.89	**86.10** ± 0.05
CoGT	**55.31** ± 0.54	**97.52** ± 0.37	**45.10** ± 1.28	**45.38** ± 3.39	**77.98** ± 1.03	**86.14** ± 0.05

We compare CoGT with both classic GNNs and recent GTs. The GNN baselines include GCN [11], GraphSAGE [9], and GAT [19], which rely on local message passing. The GTs include GraphGPS [16], NAGphormer [2], Exphormer [17], NodeFormer [21], GOAT [12], and Polynormer [4].

We ran each experiment ten times with different random seeds and reported the average results. Regarding hyperparameter settings, the type for the local structure encoder is selected from {GCN, GAT, GraphSAGE}, with the number of layers ranging from [2, 12]. The learning rate is chosen from {0.01, 0.005, 0.001, 0.0005}, and weight decay is selected from {0, 5e-4, 5e-5}. For the number of communities, we select values from [0, 70] for medium-sized graphs. For large graphs, to ensure a fair comparison, we use a batch size of 10,000 for ogbn-proteins and a batch size of 550,000 for pokec, setting the number of communities to 15 and 425, respectively. Finally, the number of layers for CoGT is chosen from {1, 2}.

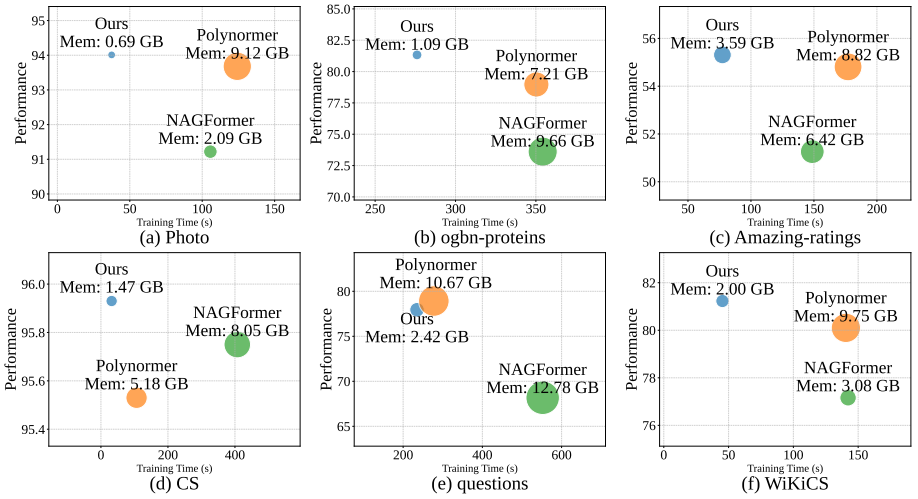


Fig. 5. Comparison of CoGT and SOTA GTs in terms of performance, efficiency, and memory consumption. The size of each point represents memory consumption.

6.2 Experimental Results

Tables 1 and 2 present the comprehensive performance of CoGT across a wide range of datasets. The selected datasets include diverse types, such as homophilic and heterophilic graphs, as well as large graphs containing a single component.

CoGT achieves outstanding performance in various scenarios and achieves the best results on multiple datasets. The key difference between CoGT and previous GTs is that it explicitly models community-level semantics during the global aggregation process. Our design provides the transformer layer with features that incorporate structural semantics, which are crucial characteristics of graph data. We believe that the performance gains of CoGT primarily stem from this step. By utilizing community-level features in the attention calculation, CoGT avoids the problem of excessive feature noise that arises from introducing node features of $O(N)$ complexity. Moreover, community-level features are derived from a community process with positional encodings, where the node semantics within each community are more consistent, allowing the central nodes to effectively aggregate valuable information while filtering out noise.

Notably, CoGT also achieves the best performance on two large-scale datasets, indicating that global aggregation using community features remains effective even for large graphs. This also confirms the existence of the information oversqueezing problem in the global aggregation process.

6.3 Efficiency Analysis

In the design of CoGT, we employ several strategies to enhance the overall model efficiency. These include utilizing a community-structured parallel atten-

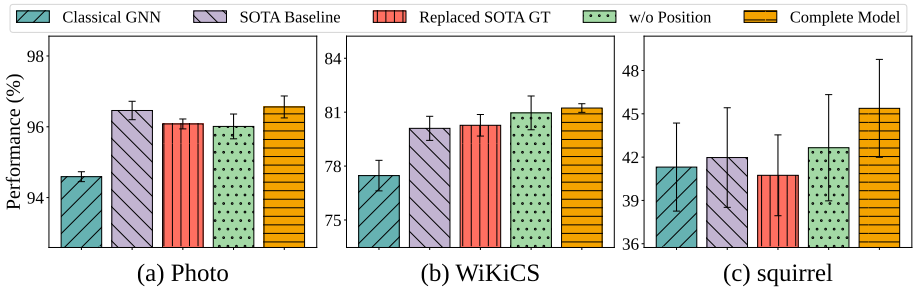


Fig. 6. The results of ablation studies. The results show that removing the proposed components leads to varying degrees of performance degradation.

tion mechanism within the transformer layer, replacing the quadratic-complexity Softmax function with the linearly-complex ReLU function, and incorporating efficient average pooling aggregation.

In terms of complexity, to prevent any community from growing too large during the aggregation process and further increasing memory consumption, we adopted a strategy where each community contains an average number of nodes. As a result, the number of nodes in a community scales with the size of the graph, leading to quadratic complexity. To address this issue, we merge the community features during computation, aggregating the features of M communities into K through block averaging, thereby achieving linear complexity.

The comparison between CoGT and SOTA baseline methods in terms of efficiency is shown in Fig.5. From the figure, it is evident that CoGT not only achieves superior performance but also demonstrates advantages in terms of peak memory usage and training time. This is attributed to the parallel attention computation, which significantly boosts model efficiency. By decomposing the multiplication of large feature matrices into parallel multiplications of smaller matrices, the required peak memory usage is greatly reduced.

6.4 Ablation Study

Fig. 6 presents the results of experiments validating the effectiveness of various components of CoGT. We replaced the HTPC layer with the SOTA method, Polynormer, to assess the effectiveness of the proposed HTPC layer. We also removed the community positional encoding to verify the importance of this step. The results show that replacing CoGT with other GT layers led to a performance decline, indicating that explicitly modeling community semantics is effective. Furthermore, removing the positional encoding also resulted in a performance drop, demonstrating that community positional encoding is beneficial. We believe this encoding highlights the semantic distinctions between communities, which is important for attention modeling.

Additionally, we conducted experiments to explore the impact of community count on model performance, with the results shown in Fig.7. It can be observed

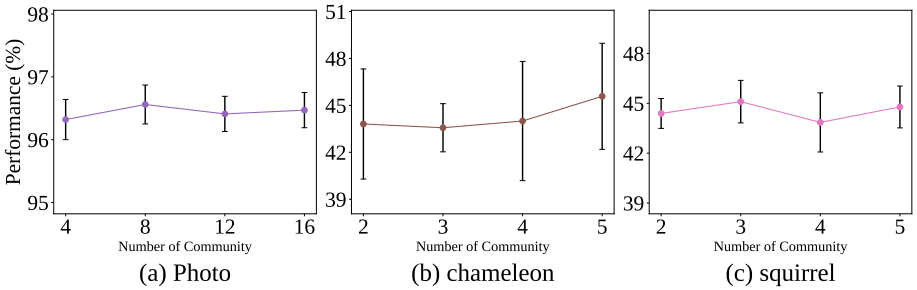


Fig. 7. The results of proposed CoGT under different community numbers. The results indicate that the optimal number of communities varies across datasets, suggesting that different datasets require different levels of semantic granularity in community representation.

that the optimal number of communities varies across different datasets. We interpret this result as indicating that the communities with clear semantic distinctions differ across datasets, similar to how universities have varying numbers and types of academic departments. From an empirical perspective, we hypothesize that the hyperparameter for community count is proportional to the number of nodes and the number of label categories.

7 Conclusion

In this paper, we first identify two key challenges that may limit the expressiveness of global attention mechanisms in GRL: (1) the absence of community-level semantics and (2) the issue of information over-squeezing. To address these issues, we propose CoGT, a novel GT variant that explicitly models community-level semantics by identifying latent communities within the graph. This design not only enhances higher-order semantic representation but also reduces the volume of aggregated information, alleviating the over-squeezing effect. Furthermore, we provide theoretical justifications to support the effectiveness of CoGT's design. Extensive experiments demonstrate that CoGT outperforms existing methods in performance while maintaining efficiency.

Currently, CoGT is not yet capable of automatically adapting the number of communities to different datasets; this value still needs to be set empirically. Therefore, a direction for future work is to develop a version of CoGT that supports adaptive community number selection. In addition, the current design of CoGT primarily targets node-level tasks, and it could be further extended to support edge-level and graph-level tasks in the future.

8 Acknowledgments

This research is supported by the National Natural Science Foundation of China under the grant No. 62376129, National Key Research and Development Program

of China under the grant No. 2023YFF0725003, Tianjin Science and Technology Major Project under the grant No. 24ZXZSSS00420, Tianjin Natural Science Foundation under the grant No. 24JCYBJC01950, and Tiankai Higher Education Science and Technology Park Enterprise R&D Special Project under the grant No. 23YFZXYC00029.

References

1. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural computation **15**(6), 1373–1396 (2003)
2. Chen, J., Gao, K., Li, G., He, K.: Nagphormer: A tokenized graph transformer for node classification in large graphs. In: The Eleventh International Conference on Learning Representations
3. Chen, M., Wei, Z., Huang, Z., Ding, B., Li, Y.: Simple and deep graph convolutional networks. In: International conference on machine learning. pp. 1725–1735. PMLR (2020)
4. Deng, C., Yue, Z., Zhang, Z.: Polynormer: Polynomial-expressive graph transformer in linear time. In: The Twelfth International Conference on Learning Representations
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers). pp. 4171–4186 (2019)
6. Dwivedi, V.P., Rampášek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A.T., Beaini, D.: Long range graph benchmark. Advances in Neural Information Processing Systems **35**, 22326–22340 (2022)
7. Girvan, M., Newman, M.E.: Community structure in social and biological networks. Proceedings of the national academy of sciences **99**(12), 7821–7826 (2002)
8. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 855–864 (2016)
9. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in neural information processing systems **30** (2017)
10. Han, K., Wang, Y., Chen, H., Chen, X., Guo, J., Liu, Z., Tang, Y., Xiao, A., Xu, C., Xu, Y., et al.: A survey on vision transformer. IEEE transactions on pattern analysis and machine intelligence **45**(1), 87–110 (2022)
11. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (2017)
12. Kong, K., Chen, J., Kirchenbauer, J., Ni, R., Bruss, C.B., Goldstein, T.: Goat: A global transformer on large-scale graphs. In: International Conference on Machine Learning. pp. 17375–17390. PMLR (2023)
13. Li, A., Pan, Y.: Structural information and dynamical complexity of networks. IEEE Transactions on Information Theory **62**(6), 3290–3339 (2016)
14. Newman, M.E., Girvan, M.: Finding and evaluating community structure in networks. Physical review E **69**(2), 026113 (2004)
15. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 701–710 (2014)

16. Rampášek, L., Galkin, M., Dwivedi, V.P., Luu, A.T., Wolf, G., Beaini, D.: Recipe for a general, powerful, scalable graph transformer. Advances in Neural Information Processing Systems **35**, 14501–14515 (2022)
17. Shirzad, H., Velingker, A., Venkatachalam, B., Sutherland, D.J., Sinop, A.K.: Exphormer: Sparse transformers for graphs. In: International Conference on Machine Learning. pp. 31613–31632. PMLR (2023)
18. Topping, J., Di Giovanni, F., Chamberlain, B.P., Dong, X., Bronstein, M.M.: Understanding over-squashing and bottlenecks on graphs via curvature. In: International Conference on Learning Representations
19. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018)
20. Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., Yu, P.S.: Heterogeneous graph attention network. In: The world wide web conference. pp. 2022–2032 (2019)
21. Wu, Q., Zhao, W., Li, Z., Wipf, D., Yan, J.: Nodeformer: A scalable graph structure learning transformer for node classification. In: Advances in Neural Information Processing Systems (2022)
22. Wu, Q., Zhao, W., Yang, C., Zhang, H., Nie, F., Jiang, H., Bian, Y., Yan, J.: Sgformer: Simplifying and empowering transformers for large-graph representations. Advances in Neural Information Processing Systems **36**, 64753–64773 (2023)
23. Wu, Z., Jain, P., Wright, M., Mirhoseini, A., Gonzalez, J.E., Stoica, I.: Representing long-range context for graph neural networks with global attention. Advances in neural information processing systems **34**, 13266–13279 (2021)
24. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems **32**(1), 4–24 (2020)
25. Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., Liu, T.Y.: Do transformers really perform badly for graph representation? Advances in neural information processing systems **34**, 28877–28888 (2021)
26. Yun, S., Jeong, M., Kim, R., Kang, J., Kim, H.J.: Graph transformer networks. Advances in neural information processing systems **32** (2019)
27. Zhang, Z., Cui, P., Zhu, W.: Deep learning on graphs: A survey. IEEE Transactions on Knowledge and Data Engineering **34**(1), 249–270 (2020)