HOGT: High-Order Graph Transformers

Anonymous Author(s)
Affiliation
Address
email

Abstract

Inspired by the success of transformers on natural language processing (NLP) and computer vision (CV) tasks, graph transformers (GTs) have recently been proposed to boost the performance of graph learning. However, the attention mechanisms used in existing GTs face certain limitations in capturing crucial topological information or scaling to large graphs, due to their quadratic complexity. To address these limitations, in this paper, we propose a high-order information propagation strategy within the transformer architecture to simultaneously learn the local, long-range, and higher-order relationships of the graph. We first propose a flexible sampling method to extract communities from the graph, and create new community nodes, especially a learnable community sampling method with reinforcement learning. We then propose a three-step message-passing strategy dubbed *HOGT* to capture the local and higher-order information in the communities and propagate long-range dependency information between the community nodes to finally obtain comprehensive node representations. Note that as structural information has been flexibly integrated into our designed community-based message-passing scheme, HOGT discards the positional encoding which was thought to be important for GT. We theoretically demonstrate that GTs with effective substructures can achieve an approximate global attention. HOGT can be viewed as a unified framework, taking many existing graph models as its special cases. We empirically show that HOGT achieves highly competitive results consistently across node and graph classification tasks. The official code will be released.

1 Introduction

Learning from graph-structured data, such as social networks, biological networks, and brain networks, is critical for many real-world applications. Graph Neural Networks (GNNs) [42, 64, 25, 28] are one type of mainstream architecture that adopts a local Message-Passing (MP) scheme where the information is propagated and aggregated between the connected nodes. However, traditional GNNs suffer from the over-smoothing [54], over-squashing [62], and limited expressiveness [73] problems because of this neighbourhood-dependent message passing strategy.

The transformer architecture [63] has recently attracted great attention for graph learning as its global attention mechanism provides a potential solution to the above problems. In contrast to traditional GNNs, Graph Transformers (GTs) [43, 49, 75] enable information to pass between any two nodes, regardless of the original graph connections. When applying transformers on graphs, the key is to properly incorporate graph structural information. This motivates several studies [43, 17, 75] to focus on constituting good positional encoding or attention bias to integrate graph structure. However, Muller et al.[52] showed that current graph transformers still suffer from limited expressivity, and no clear expressivity hierarchy exists for commonly used positional or structural encodings. Moreover, when developing GTs on real graph tasks, especially for node classification, existing models [6, 43, 55] suffer from high computational complexity due to dense connections. In

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

conclusion, the current GT models not only fail to fully capture useful topological information (e.g., intrinsic local structure, implicit higher-order correlations) of the graph but also cannot effectively propagate long-range information.

Inspired by the successful use of patches in the vision domain, some recent works [23, 80] have incorporated patch/substructure representations into GTs. While the introduced substructures can benefit graph representation in some cases, we can see the limitation of the existing works [44, 87, 86]in achieving flexible and suitable substructures for different graphs and theoretically demonstration for success of GTs with substructures. Therefore, it is nontrival to develop new scheme to effectively capture the complex structural relationships in the graph for different graph and data types, while also providing theoretical support.

In this work, we develop a powerful architecture that can effectively propagate comprehensive information with the flexible sampling method and term it as *HOGT*. To better capture the intricate relationships within a graph, we group graph nodes into multiple communities where all nodes within the same community share similar properties (semantic or information). Notably, we design a learnable community sampling method based on reinforcement learning (RL). When encoding closer graph nodes into the same community, the challenge is how to capture the local high-order information in the community and propagate it globally for effective and comprehensive representation learning. To tackle this challenge, we introduce a new node to represent each community which serves as the bridge to allow the graph node information to propagate and aggregate along these introduced nodes to establish global connections among all nodes. The generated communities can encode more complex structural information as a substitute for positional encoding.

Based on community-structured data, we adopt a three-step message-passing strategy: 1) Graph Node-to-Community Node (***G2C*-MP**); 2) Community Node-to-Community Node (***C2C*-ATTN**); and 3) Community Node-to-Graph node (***C2G*-MP**). In the first step, within each community, the information of each node is propagated and aggregated to its corresponding community node to capture local high-order information. Then, based on the community-level representations of the community nodes, we apply a self-attention mechanism between them to allow each community node to capture long-range information from other communities. Finally, we update the representations of the graph nodes by aggregating information from their respective communities. We can see that the community nodes effectively connect to almost all nodes in the graph.

Our proposed HOGT is a general framework and several other existing graph models can be viewed as special cases. At the level of message-passing strategy: if removing Community Node-to-Community Node (***C2C*-ATTN**), the framework simplifies to a Message-Passing Architecture. At the level of community generation: if we view the whole graph as a community, our model simplifies to a GT model [70], which takes a special token to connect with all other nodes to achieve global information, representing the lower bound of HOGT; if we view each node as a community, our model essentially becomes the vanilla transformer, representing the upper bound of HOGT. In comparison to the existing graph models, the advantage of our proposed HOGT in processing various graph information and graph types is shown in Table 1.

Our proposed framework demonstrates its versatility by accommodating various graph types (graph and hypergraph), data types (homophily and heterophily), data scales (same-scale and large-scale), and different graph tasks. We mainly evaluate HOGT on node classification tasks in which GT models have a performance gap, and also extend HOGT for graph classification. We find improvements in accuracy on almost all datasets, especially on heterophilic datasets. In summary, our main contributions are as follows:

- We propose a flexible sampling method with a followed three-step message-passing framework in GTs to capture comprehensive information to achieve high expressiveness for graph representation learning.
- We unify message-passing and GTs by constructing communities and introducing new community nodes. We demonstrate that our model can approximate any other message-passing model and theoretically show that the three-step message-passing with newly introduced community node can achieve global attention as general transformers do.
- We conduct extensive experiments on benchmark datasets to demonstrate the effectiveness of the proposed method for node and graph classification. The experimental results also verify the effectiveness of higher-order representations.

Table 1: A summary of the capabilities of different graph models in processing graph information and graph types. GNN is the vanilla graph neural network, HGNN is a hypergraph-based neural network, and GT is the general Graph Transformer.

Model	Local Information	Global Information	Higher-Order Information	Graph	Hypergraph
GNN	✓	✗	✗	✓	✗
HGNN	✓	✗	✓	✓	✓
GT	✓	✓	✗	✓	✗
HOGT (ours)	✓	✓	✓	✓	✓

2 Related Work

Graph Transformers. Recently, the transformer architecture has been successfully applied to the graph domain, showing competitive or even superior performance on many tasks when compared to GNNs. The standard transformer was first extended to graphs [17], with four special designs including the position encoding for nodes in a graph. Subsequently, many other GTs [59, 77, 8, 70, 36, 7, 53, 43] and applications of GTs [72, 89, 88, 5, 46] have been developed — Rampasek et al. [58] and Min et al. [50] provide a more detailed introduction and review of different GTs. However, the above methods are mostly designed for graph-level tasks, as they impose great time and memory constraints due to the self-attention layer. Therefore, several works [81, 14, 27, 55, 68, 48] have been proposed to make graph transformers more scalable and efficient, but they still suffer from various challenges such as missing long-range and higher-order information or noise aggregation.

Graph Transformer Utilizing Substructures. Due to the exponentially increasing scale of graph data, researchers have attempted to utilize substructures to scale up graph representation learning through methods such as subgraph learning [40], and graph condensation [65, 82, 84, 39, 35, 21]. In terms of Graph Transformers (GTs), substructures [78] (such as hierarchical structure, clusters, communities, and subgraphs) has been utilized for both graph and node classifications. For graph classification tasks, some methods [23, 80] segmented the graph into patches or subgraphs and used the substructural representations to learn topological high-level information. For node classification, researchers studied on extracting substructures and using these substructural representations to reduce the quadratic complexity of global self-attention while capturing global information of the graph. Specifically, Coarformer [44] and HSGT [87] employed coarsening techniques to obtain a coarser graph with fewer nodes to capture long-range information. With obtained clusters, CoBFormer [71] introduced the inter-cluster and intra-cluster Transformers to extract local information and long-range dependent information from distant nodes. AnchorGT [86] and AGFormer [38] selected several topologically important nodes as anchors, allowing information to propagate over a large receptive field, where the anchor nodes can be viewed as a substructure of the original graph. Furthermore, VCR-Graphormer [22] transfers global and long-range information by establishing multiple virtual connections using personalized PageRank. While the introduced substructures can benefit graph representation in some cases, we can see the limitation of the existing works in achieving flexible and suitable substructures for different graphs. Specifically, the anchor nodes in [86, 38] are derived from the original graph nodes and therefore cannot introduce additional information, such as higher-order information. Similarly, the supernodes in the coarse graphs in [44, 87] which used to propagate high-level information are also constrained by the original graph structure. Moreover, it is not trivial to provide theoretical support for Graph Transformers (GTs) with substructures in capturing global attention while ensuring adaptability across various graph datasets. In this work, we demonstrate that the proposed HOGT offers a general and theoretically grounded framework. It shows significant advantages in capturing comprehensive information through flexible community sampling methods and proves its versatility by effectively accommodating various graph datasets.

More related works about Higher-Order Representation Learning and Virtual Node in Message-Passing can be found in Appendix A.1.

3 High-order Graph Transformer

Overview. As illustrated in Figure 1, our proposed HOGT framework is designed to effectively aggregate and propagate all levels of information for comprehensive graph representation learning. By dividing the whole graph into several communities and introducing a representative node for

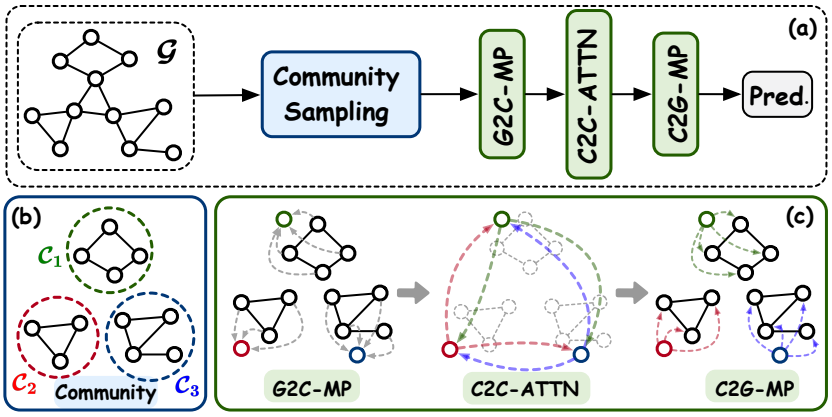

Figure 1: (a) The overall HOGT framework. (b) HOGT first adopts a community sampling method to obtain multiple communities. (c) Then, it propagates and aggregates information following a three-step operation: 1) ***G2C*-MP** which aggregates the high-order information of a community into the community node; 2) ***C2C*-MP** which propagates community-level information in a self-attention mechanism; 3) ***C2G*-MP** which gathers the updated community-level information for node representation.

each community, we achieve the local higher-order representation of each community and adopt community-level attention to effectively propagate the long-range dependent information. In the following, first we introduce the notations and provide a background on transformer architecture, and then we describe in detail each component of the architecture. The complexity analysis of HOGT can be found in Appendix A.2.

Notation. Given an graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with node set $\mathcal{V}$ and edge set $\mathcal{E}$. Suppose there are N nodes in $\mathcal{V}$, the set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ define the connections among the N nodes, $(v_i, v_j) \in \mathcal{E}$ denotes the edge between node v_i and node v_j. The graph topology is presented by the adjacency matrix $\mathbf{A}$, where $\mathbf{A}_{ij} = 1$ if there exists an edge (v_i, v_j), $\mathbf{A}_{ij} = 0$ otherwise. We denote $\mathbf{X} \in \mathbb{R}^{N \times d}$ the node features, where each node i has $x \in \mathbb{R}^d$. Let y_i denote the label of node i, in this work, we focus on the node classification task which aims to predict the labels of the unknown nodes in the graph.

3.1 Community Sampling

Effectively utilizing the structural information of the graph is the key challenge for graph representation learning. We note that data correlations in practice can be complex and are often beyond pairwise, for example, a community of friends shares their common interest in basketball in a social network. To encode these higher-order correlations, we consider extracting meaningful communities from the whole graph. Here, a community is introduced to collect multiple vertices sharing similar properties (semantic or information), similar to how a hyperedge connects multiple objects.

We design the community sampling method tailored to different graph types: 1). For hypergraphs, we intuitively view each hyperedge as a community; 2). For regular graphs, we explore a learnable sampling method that employs reinforcement learning to determine the optimal communities.

Learnable sampling. For graph $\mathcal{G} = \{\mathbf{A}, \mathbf{X}\}$, we learn a GNN-based encoder and obtain the hidden representation of N nodes: $\mathbf{H} = \left[\boldsymbol{h}_1^\top, \ldots, \boldsymbol{h}_N^\top\right]^\top \in \mathbb{R}^{N \times d}$. Then, we employ a trainable projection vector $\mathbf{p}$ to project all node features to 1D. Given node v_i with feature $\boldsymbol{h}_i$, its scale projection on $\mathbf{p}$ is $y_i = \langle_i \mathbf{p}/\|\mathbf{p}\|$. Here, y_i measures how much information of node v_i can be retained when projected to the direction of $\mathbf{p}$. After that, we adopt top-k sampling to select kN nodes, here $k \in (0, 1]$. For each selected node i, we generate a community $\tilde{\mathcal{V}}_i$ with its neighbors. To find the optimal k in top-k sampling, we present a reinforcement learning (RL) algorithm to update the sampling ratio k adaptively. We model the updating process of k as a finite horizon Markov Decision Process (MDP) and adopt Q-learning [67, 61] to learn the MDP. In the experiments, we also apply

two general sampling approaches: random walk [76] and spectral clustering [11]. More details of these three sampling methods can be found in Appendix A.3.

3.2 Model Design

Operating on communities, HOGT leverages the following three steps to obtain local, high-order, and long-range information. While achieving expressive representation with reduced computational cost, the proposed community-based method can be viewed as a new structural encoding strategy.

(1) Graph Node-to-Community Node (*G2C*-MP). To capture the higher-order information in the community, we introduce a representative community node (CN) for each community, and connect it with other nodes in the community. The use of an additional node (virtual node) that connects to all input graph nodes, has been observed to improve GNNs [26, 31, 70] and has been justified theoretically [4]. Instead of aggregating the whole information in the graph (as $\mathrm{READOUT}$) [70], we introduce the additional node for each community to capture the higher-order structural information of a graph and support the global information propagation as bridges.

Assume there are m communities $\left\{\tilde{\mathcal{V}}_1, \ldots, \tilde{\mathcal{V}}_m\right\}$, we then have m community nodes $\overline{V} = \{\overline{v}_1, \ldots, \overline{v}_m\}$. We initialize the community node feature $\overline{x}_i$ with a d-dimensional random vector. Note that the number of community nodes is significantly smaller than the number of graph nodes. For each community $\tilde{\mathcal{V}}_i$, the community representation can be obtained by the community node acting as the query $\overline{\boldsymbol{q}}_i$ with $\overline{\boldsymbol{q}}_i = \overline{x}_i \mathbf{W}^Q$: $\boldsymbol{h}_i^c = \mathrm{softmax}\left(\alpha \overline{\boldsymbol{q}}_i \mathbf{K}_{\tilde{\mathcal{V}}_i}^\top\right) \mathbf{V}_{\tilde{\mathcal{V}}_i}$, where α is a constant scalar ($\alpha = \frac{1}{\sqrt{d'}}$), $\mathbf{K}_{\tilde{\mathcal{V}}_i}$ and $\mathbf{V}_{\tilde{\mathcal{V}}_i}$ are the key and value matrices of $\overline{v}_i$'s community. The community node aggregates the community-level information.

(2) Community Node-to-Community Node (*C2C*-ATTN). To maintain the benefit of global attention in the transformer architecture, we enable information propagation between any two communities. Viewing each community node as a token, we adopt self-attention to refine the community-level representations: $\mathrm{Attn}(\mathbf{H}^c) = \mathrm{softmax}(\frac{\mathbf{Q}^c \mathbf{K}^{c\top}}{\sqrt{d'}})\mathbf{V}^c$, where $\mathbf{Q}^c = \mathbf{H}^c \mathbf{W}^Q$, $\mathbf{K}^c = \mathbf{H}^c \mathbf{W}^K$, $\mathbf{V}^c = \mathbf{H}^c \mathbf{W}^V$, with $\mathbf{H}^c = \left[\boldsymbol{h}_1^{c\top}, \ldots, \boldsymbol{h}_m^{c\top}\right]^\top \in \mathbb{R}^{m \times d'}$. By propagating information between communities, we obtain the updated community representation $\mathbf{H}^{c'}$. The information passing from community to community helps to: 1) enhance the relationship of communities, and 2) capture the long-range dependency at the community level.

(3) Community Node-to-Graph Node (*C2G*-MP). To finally obtain the representation of each node, we aggregate the community representations to update node features. We define the query vector of graph node v_i as $\boldsymbol{q}_i$, while the key and value matrices from introduced community nodes are $\mathbf{K}^{c'} \in \mathbb{R}^{m \times d'}$ and $\mathbf{V}^{c'} \in \mathbb{R}^{m \times d'}$, respectively. For graph node v_i, its representation can be enhanced with community-level representations as: $\boldsymbol{h}_i = \mathrm{softmax}\left(\alpha \boldsymbol{q}_i \mathbf{K}_{V(i)}^{c'\top}\right) \mathbf{V}_{V(i)}^{c'}$, where $\mathbf{K}_{V(i)}^{c'}$ and $\mathbf{V}_{V(i)}^{c'}$ are the key and value matrices of v_i's communities.

Considering the importance of neighbors, it is also necessary to maintain local message-passing [81] for the local-dependency graph data. Thus, the representation of graph node v_i can be updated as follows:

$$\boldsymbol{h}_i = \mathrm{softmax}\left(\alpha \boldsymbol{q}_i \mathbf{K}_{V(i)}^\top\right) \mathbf{V}_{V(i)}, \tag{1}$$

where $\mathbf{K}_{V(i)} = \begin{bmatrix} \mathbf{K}_{V(i)}^{c'} \\ \mathbf{K}_{\mathcal{N}(i)} \end{bmatrix}$ is the combination of $\mathbf{K}_{V(i)}^{c'}$ and $\mathbf{K}_{\mathcal{N}(i)}$, and $\mathbf{V}_{V(i)}$ is the combination of $\mathbf{V}_{V(i)}^{c'}$ and $\mathbf{V}_{\mathcal{N}(i)}$, where $\mathbf{K}_{\mathcal{N}(i)}$, $\mathbf{V}_{\mathcal{N}(i)}$ are the key and value matrices of neighboring nodes of v_i, respectively.

Implementation Details of HOGT We have presented the individual attention mechanism in line with general transformers. HOGT adopts multi-head attention (MHA) followed by feed-forward blocks (FFN) and layer normalization ($\mathrm{LN}(\cdot)$) as:

$$\boldsymbol{h}'^{(l)} = \mathrm{LN}\left(\mathrm{MHA}\left(\boldsymbol{h}^{(l-1)}\right)\right) + \boldsymbol{h}^{(l-1)}; \boldsymbol{h}^{(l)} = \mathrm{LN}\left(\mathrm{FNN}\left(\boldsymbol{h}'^{(l)}\right)\right) + \boldsymbol{h}'^{(l)}. \tag{2}$$

The positional encoding is an important component in transformers, and in the graph domain, researchers have integrated the positional information into GTs by random walk positional encoding [17], or Laplacian positional encoding [18]. In HOGT, the proposed community-based method can be viewed as a new structural encoding strategy.

4 Theoretical Analysis

Here, we analyze several properties of HOGT including 1) the lower bound of HOGT, 2) the upper bound of HOGT, and 3) a general case of HOGT. We show that HOGT is a powerful model that can approximate the GT model and achieve global attention, i.e., unifying MP and GT with the community and newly introduced community nodes. We also analyze the role of community nodes in capturing the high-order representation versus the function of hyperedges in hypergraph convolutional networks in Appendix A.4.

Viewing the Whole Graph as a Community. In this case, GT can be simplified by Message-Passing Neural Networks (MPNN) with an additional node that connects to all graph nodes. This forms the lower bound of HOGT (number of communities $m = 1$). It has been demonstrated by Cai et al. [4] that MPNN with a virtual node can approximate a self-attention layer arbitrarily well.

Viewing Each Node as a Community. In this case, HOGT is the standard transformer. Specifically, the three-step MP in HOGT is reduced to one step: Community Node-to-Community Node. Since a node is a community, HOGT is equivalent to propagating information between any two nodes.

Multiple Communities With Multiple Nodes. In the general case, there are multiple communities with each containing multiple nodes. In this case, we demonstrate the power of HOGT by showing that the information passing from graph nodes to community nodes back to graph nodes can approximate global attention arbitrarily well.

Definition 4.1. A full self-attention layer is defined as:

$$\boldsymbol{x}_i^{(l+1)} = \sum_{j=1}^{n} \frac{\phi\left(\boldsymbol{q}_i\right)^T \phi\left(\boldsymbol{k}_j\right)}{\sum_{k=1}^{n} \phi\left(\boldsymbol{q}_i\right)^T \phi\left(\boldsymbol{k}_k\right)} \cdot \boldsymbol{v}_j = \frac{\left(\phi\left(\boldsymbol{q}_i\right)^T \sum_{j=1}^{n} \phi\left(\boldsymbol{k}_j\right) \otimes \boldsymbol{v}_j\right)^T}{\phi\left(\boldsymbol{q}_i\right)^T \sum_{k=1}^{n} \phi\left(\boldsymbol{k}_k\right)}, \tag{3}$$

where $\phi(\cdot)$ is a low-dimensional feature map with random transformation, $\boldsymbol{q}_i$, $\boldsymbol{k}_i$, $\boldsymbol{v}_i$ are the query, key, and value vector, respectively.

Proposition 4.1. *The $\sum_{k=1}^{n} \phi\left(\boldsymbol{k}_k\right)$ and $\sum_{j=1}^{n} \phi\left(\boldsymbol{k}_j\right) \otimes \boldsymbol{v}_j$ can be approximated by the virtual node, and shared for all graph nodes, using only $\mathcal{O}(1)$ layers of MPNNs.*

Proposition 4.1 asserts that Message-Passing Neural Networks with community nodes (MPNN+CN) can function as the self-attention layer. Based on Proposition 4.1, we derive the following theorem for our three-step message-passing framework.

Theorem 4.1. *The combination of Message-Passing and self-attention: Message-Passing with an introduced new node followed by a self-attention aggregation followed by another Message-Passing can approximate self-attention arbitrarily well.*

We briefly show how the approximation error can be bounded in Proposition 4.1 and provide the proof of Theorem 4.1 in Appendix A.5.

5 Experiments

In this section, we evaluate the effectiveness of HOGT in node classification tasks, a scenario where GTs have yet to demonstrate state-of-the-art performance. We compare HOGT with standard GCN-based models (graph and hypergraph-based), heterophilic-graph based models, and GT-based models. We also apply HOGT on graph classification tasks to further demonstrate its superiority in Appendix A.8. Then, we evaluate the components of HOGT, including community sampling; structural encoding; the necessity of message-passing between the communities, the local connections in the graph. The detailed experiment settings can be found in Appendix A.6.

Table 2: Node classification results on different datasets (mean accuracy (%) and standard deviation over 10 different runs). **Red**: the best performance per dataset. **Blue**: the second best performance per dataset. OOM denotes out-of-memory.

	Cora	Citeseer	Pubmed	ogbn-arxiv
GCN-based methods				
GCN [42]	$86.92_{\pm 1.33}$	$76.13_{\pm 1.51}$	$87.01_{\pm 0.62}$	$70.40_{\pm 0.10}$
APPNP [25]	$87.75_{\pm 1.30}$	$76.53_{\pm 1.61}$	$86.52_{\pm 0.61}$	$70.20_{\pm 0.16}$
GAT [64]	$87.34_{\pm 1.14}$	$75.75_{\pm 1.86}$	$85.37_{\pm 0.56}$	$67.56_{\pm 0.12}$
HGNN [19]	$86.88_{\pm 1.22}$	$75.87_{\pm 1.47}$	$84.71_{\pm 0.56}$	OOM
Graph Transformer-based methods				
SAN [43]	$81.91_{\pm 3.42}$	$69.63_{\pm 3.76}$	$81.79_{\pm 0.98}$	$69.17_{\pm 0.15}$
Graphormer [75]	$67.71_{\pm 0.78}$	$73.30_{\pm 1.21}$	OOM	OOM
LiteGT [6]	$80.62_{\pm 2.69}$	$69.09_{\pm 2.03}$	$85.45_{\pm 0.69}$	OOM
UniMP [60]	$84.18_{\pm 1.39}$	$75.00_{\pm 1.59}$	$88.56_{\pm 0.32}$	**73.19**$_{\pm 0.18}$
ANS-GT [79]	$86.71_{\pm 1.45}$	$74.57_{\pm 1.51}$	**89.76**$_{\pm 0.46}$	–
NodeFormer [69]	$86.00_{\pm 1.59}$	$76.70_{\pm 1.70}$	$88.76_{\pm 0.50}$	–
Gapformer [48]	$87.37_{\pm 0.76}$	$76.21_{\pm 1.47}$	$88.98_{\pm 0.46}$	$71.90_{\pm 0.19}$
HOGT (randomwalk)	**88.11**$_{\pm 1.05}$	**76.74**$_{\pm 1.47}$	$89.20_{\pm 1.34}$	$71.38_{\pm 0.14}$
HOGT (clustering)	$88.09_{\pm 1.34}$	$76.35_{\pm 1.47}$	$88.96_{\pm 0.49}$	$71.10_{\pm 0.72}$
HOGT (learnable)	**88.53**$_{\pm 1.26}$	**77.59**$_{\pm 0.94}$	**89.52**$_{\pm 0.55}$	**72.02**$_{\pm 0.25}$

Datasets We experiment on a range of graph benchmarks: (1) homophilic graph datasets (Cora, Citeseer, Pubmed, and ogbn-arxiv) [56, 33], (2) heterophilic graph datasets (Cornell, Texas, Wisconsin, Actor, roman-empire, and amazon-ratings) [85, 57], and (3) hypergraph datasets (Co-authorship Cora, DBLP, and News20) [83, 74, 12], involving diverse domains and sizes (roman-empire, amazon-ratings, Co-authorship DBLP, and ogbn-arxiv are large-scale datasets). The details of these datasets are provided in Appendix A.7.

5.1 Main Results

Performance on Homophilic Graphs. The homophilic datasets are graphs with high **Homo.** (indicating the proportion of edges connecting nodes with the same label [85]). The prediction accuracies for node classification tasks are reported in Table 2 (more comparative results can be found in Appendix A.8). It can be observed that our proposed HOGT method achieves the state-of-the-art or a competitive performance on most of the datasets, regardless of the sampling method.

Compared with GCN-based methods, HOGT performs better on graphs with more nodes (*e.g.*, Pubmed), specifically, HOGT improves upon popular GNN methods-APPNP by a margin of 3% on Pubmed. This is likely because, based on local message-passing, GCN methods only capture local structural information. By contrast, HOGT enables the learning of more informative representations, including community- and global-level information, which represents a significant advantage.

Compared to GT-based methods, there is an obvious advantage for HOGT on small-scale datasets (*e.g.*, Cora and Citeseer) with higher **Homo.**, i.e., where the local-neighborhood information is more important. Thus, the vanilla global attention on the whole graph adopted in existing GTs (such as Graphormer) leads to massive unrelated information aggregation. Gapformer, a special case of HOGT with one community, also achieves good performance.

In terms of efficiency, HOGT can be easily applied to large-scale graphs, ogbn-arxiv, while some other GT methods cannot due to their poor scalability. Particularly, Graphformer and LiteGT encountered out-of-memory errors, even on small graphs. This highlights the need for a GT that can scale effectively to large-scale graphs.

Performance on Heterophilic Graphs. These heterophilic datasets are of low **Homo.**, thus can be viewed as long-range dependency datasets. From the results in Table 3 (more comparative results can be found in Appendix A.8), we can observe that specially designed heterophily-based methods can generally achieve improved performance, but not on large-scale datasets (roman-empire, amazon-ratings). Except for Gapformer, most GT-based models demonstrate a poor performance, which

Table 3: Node classification results on heterophilic datasets (mean accuracy (%) and standard deviation over 10 different runs). **Red**: the best performance per dataset. **Blue**: the second best performance per dataset.

	Cornell	Texas	Wisconsin	Actor	roman-empire	amazon-ratings
Heterophily-based methods						
MLP [45]	$71.62_{\pm 5.57}$	$77.83_{\pm 5.24}$	$82.15_{\pm 6.93}$	$33.26_{\pm 0.91}$	$64.45_{\pm 0.61}$	$42.44_{\pm 0.70}$
MixHop [1]	$76.48_{\pm 2.97}$	$83.24_{\pm 4.48}$	$85.48_{\pm 3.06}$	$34.92_{\pm 0.91}$	$82.90_{\pm 0.57}$	$51.35_{\pm 0.38}$
H2GCN [85]	$75.40_{\pm 4.09}$	$79.73_{\pm 3.25}$	$77.57_{\pm 4.11}$	$36.18_{\pm 0.45}$	$60.11_{\pm 0.52}$	$36.47_{\pm 0.23}$
FAGCN [2]	$67.56_{\pm 5.26}$	$75.67_{\pm 4.68}$	$75.29_{\pm 3.06}$	$32.13_{\pm 1.33}$	$65.22_{\pm 0.56}$	$44.12_{\pm 0.30}$
GPRGNN [13]	$76.76_{\pm 2.16}$	$81.08_{\pm 4.35}$	$82.66_{\pm 5.62}$	$35.30_{\pm 0.80}$	$64.85_{\pm 0.27}$	$44.88_{\pm 0.34}$
Graph Transformer-based methods						
SAN [43]	$50.85_{\pm 8.54}$	$60.17_{\pm 6.66}$	$51.37_{\pm 3.08}$	$27.12_{\pm 2.59}$	OOM	OOM
UniMP [60]	$66.48_{\pm 12.5}$	$73.51_{\pm 8.44}$	$79.60_{\pm 5.41}$	$35.15_{\pm 0.84}$	-	-
NAGphormer [9]	$56.22_{\pm 8.08}$	$63.51_{\pm 6.53}$	$62.55_{\pm 6.22}$	$34.33_{\pm 0.94}$	$76.12_{\pm 0.22}$	$49.44_{\pm 0.54}$
Gapformer [48]	$77.57_{\pm 3.43}$	$80.27_{\pm 4.01}$	$83.53_{\pm 3.42}$	$36.90_{\pm 0.82}$	$87.65_{\pm 0.47}$	$46.38_{\pm 0.58}$
HOGT (randomwalk)	**$79.46_{\pm 2.16}$**	**$83.44_{\pm 1.87}$**	**$87.25_{\pm 2.67}$**	**$38.11_{\pm 0.87}$**	**$88.74_{\pm 0.52}$**	**$53.94_{\pm 0.43}$**
HOGT (clustering)	$78.65_{\pm 2.82}$	**$82.63_{\pm 4.97}$**	**$86.47_{\pm 2.97}$**	$37.44_{\pm 0.68}$	$88.47_{\pm 0.53}$	$53.59_{\pm 0.59}$
HOGT (learnable)	**$79.73_{\pm 3.25}$**	$81.62_{\pm 4.49}$	$85.10_{\pm 2.00}$	**$38.62_{\pm 1.02}$**	**$88.94_{\pm 0.52}$**	**$54.32_{\pm 0.44}$**

Table 4: Analysis of positional encoding on different datasets (mean accuracy (%) and standard deviation over 10 different runs).

Community Sampling	Model	Cora	Citeseer	Cornell	Texas	Wisconsin
Spectral Clustering	HOGT(lpe)	87.79±1.33	75.87±1.75	71.35±4.05	77.30±7.37	81.96±3.26
	HOGT(rwpe)	87.52±1.53	75.65±1.78	73.78±3.83	78.38±4.01	84.71±2.11
	HOGT(w/o pe)	88.09±1.34	76.35±1.47	78.65±2.82	82.63±4.97	86.47±2.97

implies that GTs fail to propagate and aggregate useful information. By contrast, our HOGT method can be easily extended to heterophilic graph datasets. Specifically, for 4 small-scale datasets, HOGT improves upon the popular heterophily-based GNN method GPRGNN by margins of 2.7%, 4.6%, 2.8% (absolute differences) on Cornell, Wisconsin, and Actor. Compared to Gapformer, HOGT achieves performance gains of 2.1%, 3.1%, 3.7% on Cornell, Texas, and Wisconsin, respectively. On the 2 large-scale heterophilic datasets (roman-empire and amazon-ratings), HGT is significantly better than previous models. We further evaluate the effectiveness of HGT by t-test in Appendix A.8 and find that the improvements of HGT over baselines are all statistically significant (p-value$\ll$0.05).

More experimental results in Appendix 12 show that HOGT achieves better performance than popular hypergraph methods HGNN [19] and HGNN+ [24] across all hypergraph datasets. Compared to traditional HGCN methods, HOGT can propagate higher-order information more flexibly based on attention architecture.

5.2 Efficiency and Scalability

We evaluate the scalability of the proposed HOGT on other two large-scale datasets including ogbn-proteins and ogbn-products. From Table 5, we can observe that HOGT outperforms all the baselines on these large graphs. The Table 6 reports the training time per epoch, inference time, and GPU memory costs for Cora and ogbn-proteins. Since it is common practice to use a fixed number of training epochs for model training on these datasets, we report the training time per epoch to compare training efficiency. We observe that HOGT is orders of magnitude faster than Graphormer, LiteGT, and Polynormer. Compared to SGFormer, HOGT achieves a balance between performance and efficiency.

5.3 Ablation Studies

Here, we conduct a set of ablation studies to test different configurations of HOGT. The effect of self-attention between communities and local information can be found in Appendix A.8.

Evaluation with Community Sampling. Here, we compare the performance of HOGT with three different community sampling methods, i.e., learnable, random walk and spectral clustering. The results have been included in Tables 2 and 3. It shows that HOGT with proposed learnable sampling slightly outperforms random walk while random walk sampling slightly outperforms spectral clustering. Intuitively, 1) random walk sampling constrains the nodes in a community with k-hop walk length, while spectral clustering separates the graph from a global view. Thus, random walk sampling captures more local structural information than spectral clustering method; 2) Spectral clustering method is more sensitive to data types (homogeneous or heterogeneous) than random walk, as it focuses on global connections. A larger difference between HOGT (randomwalk) and HOGT (clustering) is observed on heterophilic datasets compared to homophilic datasets. It is important to note that the results of HOGT (clustering) on large-scale heterophilic datasets (roman-empire and amazon-ratings) are reported with a single community. Increasing the number of communities will result in a significant performance decrease. While our proposed learnable method can actively select optimal communities, HOGT (learnable) can achieve improved performance. We further analyze the effect of the number of communities on two unlearnable sampling methods in Appendix A.8.

Table 5: Node classification results on large-scale datasets (mean accuracy (%) and standard deviation over 3 different runs).

Model	**ogbn-proteins**	**ogbn-products**
Graphormer	OOM	OOM
SAN	OOM	OOM
ANS-GT	74.67 ± 0.65	80.64 ± 0.29
HSGT	78.13 ± 0.25	81.15 ± 0.13
SGFormer	79.53 ± 0.38	81.61 ± 0.26
Polynormer	78.20 ± 0.44	82.97 ± 0.28
HOGT	80.39 ± 0.64	83.48 ± 0.32

Effect of Position Encoding. Based on Spectral Clustering, we test the role of positional encoding for the proposed HOGT. We compare two popular positional encoding methods including Laplacian-based (lpe) and random walk positional encoding (rwpe) to HOGT without any positional encoding (w/o pe). It can be seen from Table 4 that the gap in performance is minor with or without positional encoding on homophilic datasets (Cora and Citeseer). While without positional encoding, HOGT achieves obvious better performance on heterophilic datasets, such as, Cornell, Texas, and Wisconsin. The positional encoding methods (such as lpe) usually encode the original graph connections, thus, integrating positional encoding will lead to a negative effect for these heterophilic datasets which contain massive noisy information in graph structure. A detailed analysis of the failure of positional encoding can be found in Appendix A.8. Compared to popular positional encoding methods, community sampling in HOGT are able to integrate structural information in a more flexible and effective way.

Table 6: Efficiency comparison of HOGT and graph Transformer competitors w.r.t. training time per epoch, inference time and GPU memory (GB) cost on a A100. The missing results are caused by out-of-memory.

Method	Cora			ogbn-proteins		
	Tr (ms)	Inf (ms)	Mem (MB)	Tr (s)	Inf (s)	Mem (MB)
Graphormer	90.58	71.26	359.25	-	-	-
LiteGT	15.57	5.77	227.69	-	-	-
polynormer	218.23	5.13	264.06	1.60	0.127	6429.06
SGFormer	3.66	1.42	50.87	1.26	0.098	228.19
HOGT	7.40	2.69	109.22	1.12	0.087	1284.32

6 Conclusion

In this paper, we introduced a higher-order message-passing strategy within the Transformer architecture to learn long-range, higher-order relationships for graph representation. Initially, we extract communities from the entire graph and introduce a new node for each community. Subsequently, leveraging community-structured data, we adopt a three-step message-passing scheme to aggregate information from the graph node to the community node, propagate information between community nodes and send the community-level information back to the graph nodes. The introduced nodes act like hyperedges in a hypergraph to effectively propagate information to other graph nodes. We theoretically demonstrate the powerful expressiveness of HOGT and empirically show the effectiveness of HOGT across diverse datasets on node classification. In the future, we will consider designing more flexible community sampling methods for different data types.

References

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *ICML*, 2019.

[2] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *AAAI*. AAAI Press, 2021.

[3] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations*, 2022.

[4] Chen Cai, Truong Son Hy, Rose Yu, and Yusu Wang. On the connection between MPNN and graph transformer. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 3408–3430. PMLR, 2023.

[5] Weishan Cai, Wenjun Ma, Jieyu Zhan, and Yuncheng Jiang. Entity alignment with reliable path reasoning and relation-aware heterogeneous graph transformer. In *IJCAI*, 2022.

[6] Cong Chen, Chaofan Tao, and Ngai Wong. Litegt: Efficient and lightweight graph transformers. 2021.

[7] Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. Structure-aware transformer for graph representation learning. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research, 2022.

[8] Jianwen Chen, Shuangjia Zheng, Ying Song, Jiahua Rao, and Yuedong Yang. Learning attributed graph representation with communicative message passing transformer. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2242–2248, 8 2021.

[9] Jinsong Chen, Kai-Xin Gao, Gaichao Li, and Kun He. Nagphormer: Neighborhood aggregation graph transformer for node classification in large graphs. *ArXiv*, abs/2206.04910, 2022.

[10] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pages 1725–1735. PMLR, 2020.

[11] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 257–266, 2019.

[12] Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. You are allset: A multiset function framework for hypergraph neural networks. *arXiv preprint arXiv:2106.13264*, 2021.

[13] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021.

[14] Krzysztof Choromanski, Han Lin, Haoxian Chen, Tianyi Zhang, Arijit Sehanobish, Valerii Likhosherstov, Jack Parker-Holder, Tamas Sarlos, Adrian Weller, and Thomas Weingarten. From block-toeplitz matrices to differential equations on graphs: towards a general theory for scalable masked transformers. In *International Conference on Machine Learning*, pages 3962–3983. PMLR, 2022.

[15] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[17] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021.

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

[19] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3558–3565, 2019.

[20] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *Int. Conf. Learn. Representations Workshop Representation Learn. Graphs Manifolds*, 2019.

[21] Matthias Fey, Jan-Gin Yuen, and Frank Weichert. Hierarchical inter-message passing for learning on molecular graphs. *arXiv preprint arXiv:2006.12179*, 2020.

[22] Dongqi Fu, Zhigang Hua, Yan Xie, Jin Fang, Si Zhang, Kaan Sancak, Hao Wu, Andrey Malevich, Jingrui He, and Bo Long. Vcr-graphormer: A mini-batch graph transformer via virtual connections. *arXiv preprint arXiv:2403.16030*, 2024.

[23] Han Gao, Xu Han, Jiaoyang Huang, Jian-Xun Wang, and Liping Liu. Patchgt: Transformer over non-trainable clusters for learning graph representations. In *Learning on Graphs Conference*, pages 27–1. PMLR, 2022.

[24] Yue Gao, Yifan Feng, Shuyi Ji, and Rongrong Ji. Hgnn+: General hypergraph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3181–3199, 2022.

[25] Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Combining neural networks with personalized pagerank for classification on graphs. In *International Conference on Learning Representations*, 2019.

[26] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272, 2017.

[27] Lingbing Guo, Qian Zhang, and Huajun Chen. Unleashing the power of transformer for graphs. *ArXiv*, abs/2202.10581, 2022.

[28] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances Neural Inf. Process. Syst.*, volume 30, 2017.

[29] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. *Advances in Neural Information Processing Systems*, 34:15908–15919, 2021.

[30] Xiaoxin He, Bryan Hooi, Thomas Laurent, Adam Perold, Yann LeCun, and Xavier Bresson. A generalization of vit/mlp-mixer to graphs. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 12724–12745. PMLR, 2023.

[31] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. OGB-LSC: A large-scale challenge for machine learning on graphs. In Joaquin Vanschoren and Sai-Kit Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021.

[32] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*, 2021.

[33] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv:2005.00687*, 2020.

[34] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[35] Zengfeng Huang, Shengzhong Zhang, Chong Xi, Tang Liu, and Min Zhou. Scaling up graph neural networks via graph coarsening. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 675–684, 2021.

[36] Md Shamim Hussain, Mohammed J. Zaki, and Dharmashankar Subramanian. Global self-attention as a replacement for graph convolution. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 655–665, 2022.

[37] EunJeong Hwang, Veronika Thost, Shib Sankar Dasgupta, and Tengfei Ma. An analysis of virtual nodes in graph neural networks for link prediction. In *The First Learning on Graphs Conference*, 2022.

[38] Bo Jiang, Fei Xu, Ziyan Zhang, Jin Tang, and Feiping Nie. Agformer: Efficient graph representation with anchor-graph transformer. *arXiv preprint arXiv:2305.07521*, 2023.

[39] Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. Graph condensation for graph neural networks. *arXiv preprint arXiv:2110.07580*, 2021.

[40] Dongkwan Kim and Alice Oh. Translating subgraphs to nodes makes simple gnns strong and efficient for subgraph representation learning. In *Forty-first International Conference on Machine Learning*.

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

[42] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proc. 5th Int. Conf. Learn. Representations*, 2017.

[43] Devin Kreuzer, Dominique Beaini, William L. Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. In *Advances in Neural Information Processing Systems*, 2021.

[44] Weirui Kuang, Z WANG, Yaliang Li, Zhewei Wei, and Bolin Ding. Coarformer: Transformer for large graph via graph coarsening, 2022. In *URL https://openreview. net/forum*, 2021.

[45] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[46] Chaoliu Li, Lianghao Xia, Xubin Ren, Yaowen Ye, Yong Xu, and Chao Huang. Graph transformer for recommendation. In *SIGIR*, pages 1680–1689, 2023.

[47] Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. Gated graph sequence neural networks. In *Proceedings of ICLR'16*, 2016.

[48] Chuang Liu, Yibing Zhan, Xueqi Ma, Liang Ding, Dapeng Tao, Jia Wu, and Wenbin Hu. Gapformer: Graph transformer with graph pooling for node classification. In *IJCAI*, pages 2196–2205, 2023.

[49] Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. Graphit: Encoding graph structure in transformers, 2021.

[50] Erxue Min, Runfa Chen, Yatao Bian, Tingyang Xu, Kangfei Zhao, Wenbing Huang, Peilin Zhao, Junzhou Huang, Sophia Ananiadou, and Yu Rong. Transformer for graphs: An overview from architecture perspective. *arXiv preprint arXiv:2202.08455*, 2022.

[51] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv:2007.08663*, 2020.

[52] Luis Müller, Mikhail Galkin, Christopher Morris, and Ladislav Rampásek. Attending to graph transformers. *CoRR*, abs/2302.04181, 2023.

[53] Dai Quoc Nguyen, Tu Dinh Nguyen, and Dinh Phung. Universal graph transformer self-attention networks. In *Companion Proceedings of the Web Conference 2022*, WWW '22, page 193–196, 2022.

[54] Hoang NT and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *CoRR*, abs/1905.09550, 2019.

[55] Jinyoung Park, Seongjun Yun, Hyeon ju Park, Jaewoo Kang, Jisu Jeong, KyungHyun Kim, Jung-Woo Ha, and Hyunwoo J. Kim. Deformable graph transformer. *ArXiv*, abs/2206.14337, 2022.

[56] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations*, 2020.

[57] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of gnns under heterophily: Are we really making progress? In *ICLR*, 2023.

[58] Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a General, Powerful, Scalable Graph Transformer. *arXiv:2205.12454*, 2022.

[59] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying WEI, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. In *Advances in Neural Information Processing Systems*, volume 33, pages 12559–12571, 2020.

[60] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.

[61] Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Yuanxing Ning, Philip S. Yu, and Lifang He. Sugar: Subgraph neural network with reinforcement pooling and self-supervised mutual information mechanism. In *Proc. 30th Int. Conf. World Wide Web*, page 2081–2091, 2021.

[62] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[64] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *Proc. 6th Int. Conf. Learn. Representations*, 2018.

[65] Lin Wang, Wenqi Fan, Jiatong Li, Yao Ma, and Qing Li. Fast graph condensation with structure-based neural tangent kernel. In *Proceedings of the ACM on Web Conference 2024*, pages 4439–4448, 2024.

[66] Peihao Wang, Shenghao Yang, Yunyu Liu, Zhangyang Wang, and Pan Li. Equivariant hypergraph diffusion neural operators. *arXiv preprint arXiv:2207.06680*, 2022.

[67] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[68] Qitian Wu, Chenxiao Yang, Wentao Zhao, Yixuan He, David Wipf, and Junchi Yan. Difformer: Scalable (graph) transformers induced by energy constrained diffusion. In *ICLR*, 2023.

[69] Qitian Wu, Wentao Zhao, Zenan Li, David P. Wipf, and Junchi Yan. Nodeformer: A scalable graph structure learning transformer for node classification. In *NeurIPS*, 2022.

[70] Zhanghao Wu, Paras Jain, Matthew Wright, Azalia Mirhoseini, Joseph E Gonzalez, and Ion Stoica. Representing long-range context for graph neural networks with global attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[71] Yujie Xing, Xiao Wang, Yibo Li, Hai Huang, and Chuan Shi. Less is more: on the over-globalizing problem in graph transformers. *arXiv preprint arXiv:2405.01102*, 2024.

[72] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. Graph contextualized self-attention network for session-based recommendation. In *IJCAI*, 2019.

[73] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proc. 7th Int. Conf. Learn. Representations*, 2019.

[74] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. Hypergcn: A new method for training graph convolutional networks on hypergraphs. *Advances in neural information processing systems*, 32, 2019.

[75] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[76] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

[77] Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*, 2020.

[78] Zaixi Zhang, Qi Liu, Qingyong Hu, and Chee-Kong Lee. Hierarchical graph transformer with adaptive node sampling. *Advances in Neural Information Processing Systems*, 35:21171–21183, 2022.

[79] Zaixi Zhang, Qi Liu, Qingyong Hu, and Chee-Kong Lee. Hierarchical graph transformer with adaptive node sampling. *arXiv preprint arXiv:2210.03930*, 2022.

[80] Haiteng Zhao, Shuming Ma, Dongdong Zhang, Zhi-Hong Deng, and Furu Wei. Are more layers beneficial to graph transformers? *arXiv preprint arXiv:2303.00579*, 2023.

[81] Jianan Zhao, Chaozhuo Li, Qianlong Wen, Yiqi Wang, Yuming Liu, Hao Sun, Xing Xie, and Yanfang Ye. Gophormer: Ego-graph transformer for node classification. *arXiv preprint arXiv:2110.13094*, 2021.

[82] Xin Zheng, Miao Zhang, Chunyang Chen, Quoc Viet Hung Nguyen, Xingquan Zhu, and Shirui Pan. Structure-free graph condensation: From large-scale graphs to condensed graph-free data. *Advances in Neural Information Processing Systems*, 36, 2024.

[83] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *NeurIPS*, pages 1601–1608, 2006.

[84] Houquan Zhou, Shenghua Liu, Danai Koutra, Huawei Shen, and Xueqi Cheng. A provable framework of learning graph embeddings via summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 4946–4953, 2023.

[85] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020.

[86] Wenhao Zhu, Guojie Song, Liang Wang, and Shaoguo Liu. Anchorgt: Efficient and flexible attention architecture for scalable graph transformers. *arXiv preprint arXiv:2405.03481*, 2024.

[87] Wenhao Zhu, Tianyu Wen, Guojie Song, Xiaojun Ma, and Liang Wang. Hierarchical transformer for scalable graph learning. *arXiv preprint arXiv:2305.02866*, 2023.

[88] Yangfu Zhu, Linmei Hu, Xinkai Ge, Wanrong Peng, and Bin Wu. Contrastive graph transformer network for personality detection. In *IJCAI*, 2022.

[89] Yiran Zhu, Xing Xu, Fumin Shen, Yanli Ji, Lianli Gao, and Heng Tao Shen. Posegtac: Graph transformer encoder-decoder with atrous convolution for 3d human pose estimation. In *IJCAI*, 2021.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly outline the main contributions and scope of the paper, aligning with the claims made throughout the manuscript.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The limitations are discussed in Conclusion.

   Guidelines:

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

3. **Theory Assumptions and Proofs**

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

   Answer: [Yes]

Justification: The proof is provided in Appendix.

Guidelines:

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

4. **Experimental Result Reproducibility**

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

   Answer: [Yes]

   Justification: We provide all information to reproduce the results.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We provide the source of the data.

Guidelines:

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

6. **Experimental Setting/Details**

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

Answer: [Yes]

Justification: We provide detail experimental settings.

Guidelines:

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

7. **Experiment Statistical Significance**

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

Answer: [Yes]

Justification: We provide error bars.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: We provide the compute resources.

   Guidelines:

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

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: The paper adheres to all ethical guidelines set forth by NeurIPS.

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [Yes]

    Justification: We discuss potential societal impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: the paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We cite the original papers.

    Guidelines:

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

13. **New Assets**

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

Answer: [NA]

Justification: The paper does not release new assets.

Guidelines:

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

14. **Crowdsourcing and Research with Human Subjects**

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

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

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

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

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

A Appendix

A.1 More Related Works

Higher-Order Representation learning. In computer vision, it is a common approach to divide the whole image into multiple local patches. Vision Transformers (ViTs) [16] then generate the image

representation by aggregating high-level representations from these patches rather than individual pixels. Following the transformer architecture, Han et al. [29] further subdivide each local patch into smaller patches. This innovative approach enables the model to capture more detailed representations, thus enhancing feature representations. The high-order, or high-level representations derived from local patches, which often share similar content, play a critical role in learning visual representations. In the graph domain, several studies [19, 66] also consider encoding higher-order correlations for graph representation learning. Typically, the hypergraph structure with a series of hyperedges is introduced to model the complex higher-order relationship. Within the context of GTs, some recent studies [23, 80] have attempted to extract substructures, treat them as patches, and utilize the substructural representations for graph classification tasks. As graphs continue to grow rapidly in size, the relationships among nodes become increasingly complex. Therefore, exploring and exploiting higher-order representations is essential for graph representation learning.

Virtual Node in Message-Passing. The introduction of a virtual node expands the graph by adding an extra node that facilitates information exchange among all pairs of nodes. Its effectiveness in improving performance has been observed in various tasks [32]. Recently, there has been a significant focus on studying its theoretical properties. Hwang et al. [37] analyzed the virtual node's role in the context of link prediction. They found that virtual nodes can help to add expressiveness of the learned link representation and decrease under-reaching and over-smoothing. Cai et al. [4] demonstrated the power of message-passing with a virtual node, showing that it can approximate an arbitrary self-attention layer within GTs. While the function of virtual node as READOUT has been explored in existing GNNs, the community nodes in our HOGT have a slightly different function. In addition to aggregate information like the READOUT, they act as bridges connecting the entire graph to propagate long-range dependent information, while also saving computational costs, as the number of communities is significantly smaller than the number of graph nodes.

A.2 Complexisty Analysis of HOGT

We analyze the complexity of HOGT. The computational complexity of the first step Graph Node-to-community Node is $\mathcal{O}(mN)$. Since m is the number of community and usually much smaller than the number of graph nodes N, the computational complexity can be simplified as $\mathcal{O}(N)$. Moreover, the computational complexity of the second step community Node-to-community Node is $\mathcal{O}\left(m^2\right)$, it is a self-attention. The final step community Node-to-community Node is $\mathcal{O}(N)$. Therefore, the overall complexity of HOGT is $\mathcal{O}(m^2 + N)$.

A.3 The community sampling methods

Random walk sampling. To preserve the graph structural information as well as local or long-range connectivity, random walk sampling is a simple but effective approach. We consider a regular random walk sampler with m root nodes selected uniformly at random and each walker goes k hops. As such, we can obtain the communities $\left\{\tilde{\mathcal{V}}_1, \ldots, \tilde{\mathcal{V}}_m\right\}$. Each community $\tilde{\mathcal{V}}_i$ has $k+1$ nodes which are k-hop neighbours.

Spectral clustering. Spectral clustering methods segment the graph by minimum cuts such that the number of within-cluster links is much higher than between-cluster links in order to better capture good community structure. However, these spectral clustering methods can just obtain non-overlapping clusters. As we aim to achieve more communication between communities, we extend each cluster with its 1-hop neighbourhood [30]. Thus, we can obtain m communities $\left\{\tilde{\mathcal{V}}_1, \ldots, \tilde{\mathcal{V}}_m\right\}$, where $\tilde{\mathcal{V}}_i \leftarrow \tilde{\mathcal{V}}_i \cup \left\{\mathcal{N}_1(j) \mid j \in \tilde{\mathcal{V}}_i\right\}$.

Learnable sampling. For regular graphs, we explore a learnable method that employs reinforcement learning to determine the optimal number of clusters.

Given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with node set $\mathcal{V}$ and edge set $\mathcal{E}$. Suppose there are N nodes in $\mathcal{V}$. The graph topology is presented by the adjacency matrix $\mathbf{A}$. First, we learn a GNN-based encoder:

$$\mathbf{H}_\ell = \mathrm{GNN}\left(\mathbf{A}, \mathbf{H}_{\ell-1}\right), \ell = 1, \ldots, L, \tag{4}$$

and obtain the representation of N nodes $\mathbf{H} = \left[\boldsymbol{h}_1^\top, \ldots, \boldsymbol{h}_N^\top\right]^\top \in \mathbb{R}^{N\times d}$. Then, we employ a trainable projection vector $\mathbf{p}$ to project all node features to 1D. Given node v_i with feature $\boldsymbol{h}_i$, the scale projection of x_i on $\mathbf{p}$ is $y_i = \boldsymbol{h}_i\mathbf{p}/\|\mathbf{p}\|$. Here, y_i measures how much information of node v_i can be retained when projected to the direction of $\mathbf{p}$. After that, we adopt top-k sampling to select kN nodes, here $k \in (0, 1]$. For each selected node i, we generate a community $\tilde{\mathcal{V}}_i$ with its neighbors.

To find the optimal k in top-k sampling, we present a reinforcement learning (RL) algorithm to update the sampling ratio k adaptively. We model the updating process of k as a finite horizon Markov Decision Process (MDP). Formally, the state, action, transition, reward and termination of the MDP are defined as follows:

State. The state s_e at epoch e is represented by the indices of selected nodes with pooling ratio k:

Action. RL agent updates k by taking action a_e based on reward. We define the action a_e as add or minus a fixed value $\Delta k \in [0, 1]$ from k.

Transition. After updating k, we use top-k sampling to select a new set of nodes and corresponding communities in the next epoch.

Reward. Due to the black-box nature of GTs, it is hard to sense its state and cumulative reward. So we define a discrete reward function $\text{reward}\left(s_e, a_e\right)$ for each s_e at a_e directly based on the classification results:

$$\text{reward}\left(s_e, a_e\right) = \begin{cases} +1, & \text{if } acc_e > acc_{e-1} \\ 0, & \text{if } acc_e = \text{acc}_{e-1} \\ -1, & \text{if } acc_e < acc_{e-1}, \end{cases} \tag{5}$$

where acc_e is the classification accuracy at epoch e. Eq. (4) indicates if the classification accuracy with a_e is higher than the previous epoch, the reward for a_e is positive, and vice versa.

Termination. If the change of k among 10 consecutive epochs is no more than Δk, the RL algorithm will stop and k will remain fixed during the next training process. This means that RL finds the optimal threshold that can retain the most striking nodes. The terminal condition is formulated as:

$$\text{Range } \left(\{k_{e-10}, \cdots, k_e\}\right) \leq \Delta k. \tag{6}$$

We adopt Q-learning [67, 61] to learn the MDP. Q-learning is an off-policy RL algorithm that seeks to find the best action to take given the current state. It fits the Bellman optimality equation as follows:

$$Q^*\left(s_e, a_e\right) = \text{reward}\left(s_e, a_e\right) + \gamma \arg\max_{a'} Q^*\left(s_{e+1}, a'\right), \tag{7}$$

where $\gamma \in [0, 1]$ is a discount factor of future reward. We adopt a ε-greedy policy with an explore probability ε:

$$\pi\left(a_e \mid s_e; Q^*\right) = \begin{cases} \text{random action}, & \text{w.p. } \varepsilon \\ \arg\max_{a_e} Q^*\left(s_e, a\right), & \text{otherwise} \end{cases} \tag{8}$$

This means that the RL agent explores new states by selecting an action at random with probability ε instead of selecting actions based on the max future reward. We train the RL agent and node classification model jointly in an end-to-end manner.

A.4 Connection between Community Node and Hyperedge

We analyze the role of community nodes in capturing the high-order representation in HOGT versus the function of hyperedges in hypergraph convolutional networks.

Encode complex relationship. To encode the high-order correlations in the complicated graph, in hypergraph convolutional networks (HGCN), the hyperedges are introduced to connect multiple nodes. In this work, we introduce a community node for each community which contains multiple nodes sharing similar properties (semantic or information). Like the hyperedge, the community node connects with every node in its community.

High-Order Message-Passing. Following the message-passing scheme, HGCN first propagates and aggregates information along hyperedge e^h to obtain the hyperedge presentation $\boldsymbol{a}_{e^h}$, then updates

the node representation by aggregating the hyperedge representations. Formally, the layer-wise message-passing is defined as:

$$\boldsymbol{a}_{e^h}^{(k)} = \text{Aggregate}^{(k)}\left(\left\{\boldsymbol{z}_u^{(k-1)} : u \in e^h\right\}\right), \boldsymbol{z}_v^{(k)} = \text{Update}^{(k)}\left(\left\{\boldsymbol{a}_{e^h}^{(k)} : v \in e^h\right\}\right), \tag{9}$$

where $\boldsymbol{z}_v^{(k)}$ is the feature vector of node v at the k^{th} layer. The hypergraph-based convomutional networks design $\text{Aggregate}^{(k)}(\cdot)$ and $\text{Combine}^{(k)}(\cdot)$ operations based on hypergraph structure.

For example, in a spectral-based hypergraph convolutional network, the convolutional operation is defined as:

$$\boldsymbol{\Delta} = \mathbf{D}_v^{-1/2}\mathbf{S}\mathbf{W}\mathbf{D}_e^{-1}\mathbf{S}^\top\mathbf{D}_v^{-1/2}, \boldsymbol{h}^{(k)} = \sigma\left(\boldsymbol{\Delta}\boldsymbol{Z}^{(k-1)}\boldsymbol{\Theta}^{(k)}\right), \tag{10}$$

where the diagonal matrices $\mathbf{D}_v$ and $\mathbf{D}_e$ denote the vertex and hyperedge degrees, respectively. $\mathbf{W}$ indicate the relationship of hyperedges, the incidence matrix $\mathbf{S}$ denote the correlations of nodes and hyperedges with $S(v,e) = \begin{cases} 1, & \text{if } v \in e \\ 0, & \text{if } v \notin e \end{cases}$, Θ^k is the weights of k^{th} layer. Based on the hyperedge operation, we can refine the message-passing in Eq. 10 into three steps: node-to-hyperedge, hyperedge-to-hyperedge, hyperedge-to-node with the approximate presentation:

$$\boldsymbol{a}_{e^h}^{(k)} = \mathbf{S}^\top\boldsymbol{z}^{(k-1)}, \boldsymbol{a}_{e^h}^{(k)} = \mathbf{W}\boldsymbol{a}_{e^h}^{(k)}, \boldsymbol{z}^{(k)} = \mathbf{S}\boldsymbol{a}_{e^h}^{(k)}. \tag{11}$$

We can see that the three-step message-passing in HGCN is equivalent to the three-step operation in HOGT. In HGCN, the relationship of hyperedges usually can be ignored, i.e., $\mathbf{W} = \mathbf{I}$. In HOGT, the framework can also be simplified to two steps without Community Node-to-Community Node. From a high level, graph convolutional neural networks can be viewed as special cases of hypergraph convolutional networks. In comparison, our proposed HOGT framework can be simplified to other existing GT models.

A.5 Proof

Proof of Proposition 4.1 Here, we briefly show how the approximation error can be bounded in Proposition 4.1. The complete proof can be found in [4].

Proof. We first make the following assumptions on the feature space $\mathcal{X} \subset \mathbb{R}^{n\times d}$ and the regularity of layer $\mathbf{L}$.

Assumption 1. $\forall i \in [n]$, $\boldsymbol{x}_i \in \mathcal{X}_i$, $\|\boldsymbol{x}_i\| < C_1$. This implies $\mathcal{X}$ is compact.

Assumption 2. $\|\boldsymbol{W}_Q\| < C_2$, $\|\boldsymbol{W}_K\| < C_2$, $\|\boldsymbol{W}_V\| < C_2$ for target layer $\mathbf{L}$. Combined with Assumption 1 on $\mathcal{X}$, this means the unnormalized attention $\alpha'(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i{}^T\boldsymbol{W}_Q(\boldsymbol{W}_K)^T\boldsymbol{x}_j$ is both upper and lower bounded, which further implies $\sum_j e^{\alpha'(\boldsymbol{x}_i,\boldsymbol{x}_j)}$ be both upper bounded and lower bounded.

Under Assumptions 1 and 2, MPNN+CN of $\mathcal{O}(1)$ width and $\mathcal{O}(1)$ depth can approximate $\boldsymbol{L}_{\text{Performer}}$ and $\boldsymbol{L}_{\text{Linear-Transformer}}$ arbitrarily well. Specifically, ϕ can be approximated arbitrarily well by MLP with $\mathcal{O}(1)$ width and $\mathcal{O}(1)$ depth [15], $\phi(\boldsymbol{q}_i)$, $\sum_{j=1}^n \phi(\boldsymbol{k}_j)\otimes\boldsymbol{v}_j$ lies in a compact domain (n is fixed) as ϕ is continuous, $\phi(\boldsymbol{q}_i)^T\sum_{k=1}^n\phi(\boldsymbol{k}_k)$ is uniformly lower bounded by a positive number for any node features in $\mathcal{X}$. In Proposition 4.1, we consider Linear Transformer for convenience. □

Proof of Theorem 4.1 The "full" self-attention can be approximated following: 1) Message-Passing Neural Networks with community nodes (MPNN+CN) can act as the self-attention layer, and 2) Under our three-step message-passing framework, the combination of MPNN+CN with the self-attention can achieve the approximated full self-attention in graph. While point 1) has been validated in Proposition 1, we mainly demonstrated point 2).

Proof. In the process of Graph Node-to-Community Node (***G2C*-MP**), the message-passing in a community is powerful to update community node (cn) by aggregate the information fom graph nodes (gn) as:

$$h_i^{(k)} = \tau_{j\in\mathcal{C}(i)}\phi_{\mathrm{gn-cn}}^{(k)}\left(h_i^{(k-1)}, x_j^{(k-1)}, e_{j,i}\right), \tag{12}$$

where ϕ is message function, and τ is aggregation function, $\mathcal{C}(i)$ is the graph nodes in the community i. Based on **Proposition 4.1**, the message-passing with a new introduced node that connected to every nodes in the community can be approximated by the following aggregation function τ:

$$h_i^{(k)} = \tau_{j\in\mathcal{C}(i)}\phi_{\boldsymbol{G2C-\mathrm{MP}}}^{(k)}\left(\cdot, \{\boldsymbol{x}_i\}_i\right) = \left[\sum_{j=1}^{|\mathcal{C}|}\phi\left(\boldsymbol{k}_j\right),\ f\left(\sum_{j=1}^{|\mathcal{C}|}\phi\left(\boldsymbol{k}_j\right)\otimes\boldsymbol{v}_j\right)\right], \tag{13}$$

where $f(\cdot)$ flattens a 2D matrix to a 1D vector in raster order, $\boldsymbol{k}_j = \boldsymbol{W}_K^{(k)}\boldsymbol{x}_j^{(k)}$, and $\boldsymbol{v}_j = \boldsymbol{W}_V^{(k)}\boldsymbol{x}_j^{(k)}$.

Then, in the process of Community Node-to-Community Node (***C2C*-ATTN**), a self-attention mechanism ($\gamma_{\boldsymbol{C2C-\mathrm{ATTN}}}$) is adopted to propagate information between any two community nodes. The updated community nodes can be represented as:

$$\overline{h}_i^k = \gamma_{\boldsymbol{C2C-\mathrm{ATTN}}}\left(\left[\sum_{j=1}^{m}\phi\left(\boldsymbol{k}_j\right),\ f\left(\sum_{j=1}^{m}\phi\left(\boldsymbol{k}_j\right)\otimes\boldsymbol{v}_j\right)\right]\right), \tag{14}$$

where m is the number of communities, $\boldsymbol{k}_j = \boldsymbol{W}_K^{(k)}h_j^{(k)}$, and $\boldsymbol{v}_j = \boldsymbol{W}_V^{(k)}h_j^{(k)}$.

Finally, the updated community node sends its message back to graph nodes in its community. Each graph node v_i applies the update function γ_{gn}:

$$x_i^{(k)} = \gamma_{\mathrm{gn}}^{(k)}\left(x_i^{(k-1)}, \tau_{j\in\mathcal{V}_{(i)}}\phi_{\mathrm{cn-gn}}^{(k)}\left(x_i^{(k-1)}, \overline{h}_j^{(k-1)}, e_{j,i}\right)\right), \tag{15}$$

where $\mathcal{V}_{(i)}$ is the the community set of graph node i. Based on **Proposition 4.1**, the message-passing in the step Community Node-to-Graph Node (***C2G*-MP**) can be formulated as:

$$x_i^{(k)} = \gamma_{\boldsymbol{C2G-\mathrm{MP}}}\left(\boldsymbol{x}_i, \left[\sum_{j=1}^{|\mathcal{V}_{(i)}|}\phi\left(\boldsymbol{k}_j\right), f\left(\sum_{j=1}^{|\mathcal{V}_{(i)}|}\phi\left(\boldsymbol{k}_j\right)\otimes\boldsymbol{v}_j\right)\right]\right) \tag{16}$$

where $\boldsymbol{k}_j = \boldsymbol{W}_K^{(k)}\overline{h}_j^{(k)}$, and $\boldsymbol{v}_j = \boldsymbol{W}_V^{(k)}\overline{h}_j^{(k)}$.

Following the three-step architecture, the information of a graph node can be propagated to any other nodes by the community nodes as the bridges. And the representations of graph nodes can be approximated as:

$$x_i^k = \frac{\left(\phi\left(\boldsymbol{q}_i\right)\sum_{j=1}^{n}\phi\left(\boldsymbol{k}_j\right)\otimes\boldsymbol{v}_j\right)^T}{\phi\left(\boldsymbol{q}_i\right)^T\sum_{k=1}^{n}\phi\left(\boldsymbol{k}_k\right)}, \tag{17}$$

where n is the number of graph nodes, $\boldsymbol{q}_i = \boldsymbol{W}_Q^{(k)}x_i^{(k)}$, $\boldsymbol{k}_j = \boldsymbol{W}_K^{(k)}x_j^{(k)}$, and $\boldsymbol{v}_j = \boldsymbol{W}_V^{(k)}x_j^{(k)}$. Therefore, the combination of Message-Passing with a new node followed by a self-attention followed by another Message-Passing can approximate self-attention arbitrarily well.

□

A.6 Experimental part

Settings. For Cora, Citeseer, and Pubmed datasets, we follow the same experimental procedure, such as features and data splits in [56]. For heterophilic graph datasets (Cornell, Texas, Wisconsin, and Actor), we adopt the same dataset splits used by [85]. For roman-empire and amazon-ratings, we follow the settings in [57]. For hypergraphs, we adopt the same setting as [74, 12]. For other datasets, we randomly split them into 60%/20%/20% as training/validation/test sets following [79, 48]. The dataset obgn-arxiv can be downloaded from Open Graph Benchmark (OGB) [34] [1], hypergraph

[1] OGB: `https://ogb.stanford.edu/docs/nodeprop/#ogbn-arxiv`

datasets from [2], roman-empire and amazon-ratings from [3], all the other graph datasets from PyTorch Geometric (PyG) [20] [4]

For the general sampling methods-random walk [76] and spectral clustering [11], we set the number of communities to 1 (the whole graph as a community) and $1\%, 10\%, 20\%, 50\%$ of the number of nodes in the graph. For the proposed learnable sampling method, the optimal number of communities can be actively learned. The training utilizes Adam optimizer [41] for GNN methods, while Adamw is adopted for all Graph Transformer-based models. Each method runs for 200 epochs on all datasets, with the test accuracy reported based on the epoch that achieves the highest validation accuracy. We set 3 layers HOGT for ogbn-arxiv, 5 layers for roman-empire and amazon-ratings, and 2 layers for other datasets. We search model hyper-parameters including walk length of random walk, hidden dimension, and dropout. The results of HOGT are averaged over 10 runs with random weight initializations. Furthermore, all the experiments are conducted on a Linux server equipped with NVIDIA A100.

A.7 Dataset Statistic.

Table 7: Statistics of graph benchmark datasets.

	Cora	Citeseer	Pubmed	ogbn-arxiv	Cornell	Texas	Wisconsin	Actor	roman-empire	amazon-ratings
# Nodes	2,708	3,327	19,717	169,343	183	183	251	7,600	22,662	24,492
# Edges	5,429	4,732	44,338	1,166,343	280	195	466	26,752	32,927	93,050
Homo.	0.83	0.72	0.79	0.63	0.30	0.11	0.21	0.22	0.05	0.38

Table 8: Statistics of hypergraph benchmark datasets.

	Coauthorship-Cora	Coathorship-DBLP	News20
# Nodes	2,708	41,302	16,342
# Hyperedges	1,072	22,363	100
# Classes	7	6	4

[2]DHG: `https://deephypergraph.readthedocs.io/en/latest/index.html`

[3]DGL: `https://docs.dgl.ai/`

[4]PyG: `https://github.com/pyg-team/pytorch_geometric`

A.8 More Results and Explanations.

Table 9: Node classification results on different datasets (mean accuracy (%) and standard deviation over 10 different runs). **Red**: the best performance per dataset. **Blue**: the second best performance per dataset. OOM denotes out-of-memory.

	Cora	Citeseer	Pubmed	ogbn-arxiv
GCN-based methods				
GCN [42]	$86.92_{\pm 1.33}$	$76.13_{\pm 1.51}$	$87.01_{\pm 0.62}$	$70.40_{\pm 0.10}$
APPNP [25]	$87.75_{\pm 1.30}$	**$76.53_{\pm 1.61}$**	$86.52_{\pm 0.61}$	$70.20_{\pm 0.16}$
GCNII [10]	$86.08_{\pm 2.18}$	$74.75_{\pm 1.76}$	$85.98_{\pm 0.61}$	$69.78_{\pm 0.16}$
GAT [64]	$87.34_{\pm 1.14}$	$75.75_{\pm 1.86}$	$85.37_{\pm 0.56}$	$67.56_{\pm 0.12}$
GATv2 [3]	$87.25_{\pm 0.89}$	$75.72_{\pm 1.30}$	$85.75_{\pm 0.55}$	$68.84_{\pm 0.13}$
HGNN [19]	$86.88_{\pm 1.22}$	$75.87_{\pm 1.47}$	$84.71_{\pm 0.56}$	OOM
HGNN+ [24]	$83.22_{\pm 0.91}$	$74.71_{\pm 1.64}$	$83.77_{\pm 0.65}$	–
Graph Transformer-based methods				
SAN [43]	$81.91_{\pm 3.42}$	$69.63_{\pm 3.76}$	$81.79_{\pm 0.98}$	$69.17_{\pm 0.15}$
Graphormer [75]	$67.71_{\pm 0.78}$	$73.30_{\pm 1.21}$	OOM	OOM
LiteGT [6]	$80.62_{\pm 2.69}$	$69.09_{\pm 2.03}$	$85.45_{\pm 0.69}$	OOM
UniMP [60]	$84.18_{\pm 1.39}$	$75.00_{\pm 1.59}$	$88.56_{\pm 0.32}$	**$73.19_{\pm 0.18}$**
ANS-GT [79]	$86.71_{\pm 1.45}$	$74.57_{\pm 1.51}$	**$89.76_{\pm 0.46}$**	–
NodeFormer [69]	$86.00_{\pm 1.59}$	$76.70_{\pm 1.70}$	$88.76_{\pm 0.50}$	–
Gapformer [48]	$87.37_{\pm 0.76}$	$76.21_{\pm 1.47}$	$88.98_{\pm 0.46}$	**$71.90_{\pm 0.19}$**
HOGT (randomwalk)	**$88.11_{\pm 1.05}$**	**$76.74_{\pm 1.47}$**	$89.20_{\pm 1.34}$	$71.38_{\pm 0.14}$
HOGT (clustering)	$88.09_{\pm 1.34}$	$76.35_{\pm 1.47}$	$88.96_{\pm 0.49}$	$71.10_{\pm 0.72}$
HOGT (learnable)	**$88.53_{\pm 1.26}$**	**$77.59_{\pm 0.94}$**	**$89.52_{\pm 0.55}$**	**$72.02_{\pm 0.25}$**

Table 10: The p-values of the t-test between the performances of different methods.

Model	Cornell	Actor	roman-empire
Mixhop/HGT	0.026	5.67e-07	8.36e-13
GPRGNN/HGT	0.016	1.36e-06	7.82e-27
Gapformer/HGT	0.037	0.006	0.0009

Table 12: Node classification results on hypergraph datasets (mean accuracy (%) and standard deviation over 5 different runs). The complexity of information propagation can be found for different models. The number of nodes, edges, and communities are $|E|$, N, and m, respectively.

Model	Coauthor-Cora	Coauthor-DBLP	News20	Complexity
GCN	64.42±0.68	81.35±0.18	76.82±0.48	$\mathcal{O}(\|E\|)$
HGNN	61.18±0.62	82.66±1.05	81.06±1.03	$\mathcal{O}(N^2)$
HGNN+	60.40±0.77	82.86±0.85	81.24±0.75	$\mathcal{O}(N^2)$
HOGT (ours)	**68.82±1.34**	**85.82 ±0.70**	**81.32±0.80**	$\mathcal{O}(m^2+N)$

Table 11: Node classification results on heterophilic datasets (mean accuracy (%) and standard deviation over 10 different runs). **Red**: the best performance per dataset. **Blue**: the second best performance per dataset.

	Cornell	Texas	Wisconsin	Actor	roman-empire	amazon-ratings
GCN-based methods						
GCN [42]	$45.67_{\pm 7.96}$	$60.81_{\pm 8.03}$	$52.55_{\pm 4.27}$	$28.73_{\pm 1.17}$	$73.69_{\pm 0.74}$	$48.70_{\pm 0.63}$
APPNP [25]	$41.35_{\pm 7.15}$	$61.62_{\pm 5.37}$	$55.29_{\pm 3.90}$	$29.42_{\pm 0.81}$	$72.73_{\pm 0.44}$	$45.62_{\pm 0.52}$
GAT [64]	$47.02_{\pm 7.66}$	$62.16_{\pm 4.52}$	$57.45_{\pm 3.51}$	$28.33_{\pm 1.13}$	$80.87_{\pm 0.30}$	$49.09_{\pm 0.63}$
GATv2 [3]	$50.27_{\pm 8.97}$	$60.54_{\pm 4.55}$	$52.74_{\pm 3.96}$	$28.79_{\pm 1.47}$	$80.99_{\pm 0.98}$	$44.00_{\pm 0.67}$
Heterophily-based methods						
MLP [45]	$71.62_{\pm 5.57}$	$77.83_{\pm 5.24}$	$82.15_{\pm 6.93}$	$33.26_{\pm 0.91}$	$64.45_{\pm 0.61}$	$42.44_{\pm 0.70}$
MixHop [1]	$76.48_{\pm 2.97}$	$83.24_{\pm 4.48}$	$85.48_{\pm 3.06}$	$34.92_{\pm 0.91}$	$82.90_{\pm 0.57}$	$51.35_{\pm 0.38}$
H2GCN [85]	$75.40_{\pm 4.09}$	$79.73_{\pm 3.25}$	$77.57_{\pm 4.11}$	$36.18_{\pm 0.45}$	$60.11_{\pm 0.52}$	$36.47_{\pm 0.23}$
FAGCN [2]	$67.56_{\pm 5.26}$	$75.67_{\pm 4.68}$	$75.29_{\pm 3.06}$	$32.13_{\pm 1.33}$	$65.22_{\pm 0.56}$	$44.12_{\pm 0.30}$
GPRGNN [13]	$76.76_{\pm 2.16}$	$81.08_{\pm 4.35}$	$82.66_{\pm 5.62}$	$35.30_{\pm 0.80}$	$64.85_{\pm 0.27}$	$44.88_{\pm 0.34}$
Graph Transformer-based methods						
SAN [43]	$50.85_{\pm 8.54}$	$60.17_{\pm 6.66}$	$51.37_{\pm 3.08}$	$27.12_{\pm 2.59}$	OOM	OOM
UniMP [60]	$66.48_{\pm 12.5}$	$73.51_{\pm 8.44}$	$79.60_{\pm 5.41}$	$35.15_{\pm 0.84}$	-	-
NAGphormer [9]	$56.22_{\pm 8.08}$	$63.51_{\pm 6.53}$	$62.55_{\pm 6.22}$	$34.33_{\pm 0.94}$	$76.12_{\pm 0.22}$	$49.44_{\pm 0.54}$
Gapformer [48]	$77.57_{\pm 3.43}$	$80.27_{\pm 4.01}$	$83.53_{\pm 3.42}$	$36.90_{\pm 0.82}$	$87.65_{\pm 0.47}$	$46.38_{\pm 0.58}$
HOGT (randomwalk)	**79.46**$_{\pm 2.16}$	**83.44**$_{\pm 1.87}$	**87.25**$_{\pm 2.67}$	**38.11**$_{\pm 0.87}$	**88.74**$_{\pm 0.52}$	**53.94**$_{\pm 0.43}$
HOGT (clustering)	$78.65_{\pm 2.82}$	**82.63**$_{\pm 4.97}$	**86.47**$_{\pm 2.97}$	$37.44_{\pm 0.68}$	$88.47_{\pm 0.53}$	$53.59_{\pm 0.59}$
HOGT (learnable)	**79.73**$_{\pm 3.25}$	$81.62_{\pm 4.49}$	$85.10_{\pm 2.00}$	**38.62**$_{\pm 1.02}$	**88.94**$_{\pm 0.52}$	**54.32**$_{\pm 0.44}$

Performance on Hypergraphs. Theoretically, both hypergraph convolutional networks (HGCN) and our HOGT can learn high-order correlations in complex datasets. Here, based on hypergraph structure, we generate a community for each hyperedge. According to the results in Table 12, HOGT achieves better performance than popular hypergraph methods HGNN [19] and HGNN+ [24] across all hypergraph datasets. Compared to traditional HGCN methods, HOGT can propagate higher-order information more flexibly based on attention architecture.

The fail of positional encoding on heterophilic datasets. To better explain this phenomenon, we first show how positional encoding is related to the theoretical properties of GTs, e.g., their expressive power in capturing graph structure.

The implementation of PE, i.e., concatenated with input features, tends to influence the attention scores, producing an attention bias. Considering that $\mathbf{Q} \in \mathbb{R}^{n\times d}$, $\mathbf{K} \in \mathbb{R}^{n\times d}$, and $\mathbf{P} \in \mathbb{R}^{n\times d'}$ represent the query, key, and PE vectors, respectively, the attention score $\mathbf{S} \in \mathbb{R}^{n\times n}$ is calculated as:

$$\mathbf{S} = \mathbf{Q}\mathbf{K}^\top. \tag{18}$$

After concatenating the PE vector, the refined attention score $\mathbf{S}'$ is calculated as:

$$\begin{aligned}\mathbf{S}' &= [\mathbf{Q},\mathbf{P}] \times [\mathbf{K},\mathbf{P}]^\top \\ &= [\mathbf{Q},\mathbf{P}] \times \begin{bmatrix}\mathbf{K}^\top \\ \mathbf{P}^\top\end{bmatrix} \\ &= \mathbf{Q}\mathbf{K}^\top + \mathbf{P}\mathbf{P}^\top,\end{aligned} \tag{19}$$

where $[\mathbf{Q}, \mathbf{P}]$ denotes the concatenation of the query vector $\mathbf{Q}$ with the PE vector, and $[\mathbf{K}, \mathbf{P}]$ denotes the concatenation of the key vector $\mathbf{K}$ with the PE vector. The $\mathbf{P}\mathbf{P}^\top$ term can be interpreted as an attention bias.

Inappropriate positional encoding can affect the attention matrix, leading to a negative impact on performance. Muller et al. [52] clarified that no clear expressivity hierarchy exists for the popular positional or structural encodings, including Laplacian PE and RandomWalk PE. In other words, the critical aspects of existing PEs in GT haven't been demonstrated theoretically and empirically.

From Table 4 in the paper, the performance gap is minor with or without positional encoding methods on homophilic datasets (Cora and Citeseer). Without positional encoding, HGT demonstrates a better performance on heterophilic datasets, such as Cornell, Texas, and Wisconsin. This implies that existing positional encoding methods cannot accurately capture the structural information from heterophilic datasets, which is consistent with the above analysis. This motivates researchers to design more suitable positional encoding methods for different datasets or explore alternative approaches to encode the graph structural information like our HGT framework.

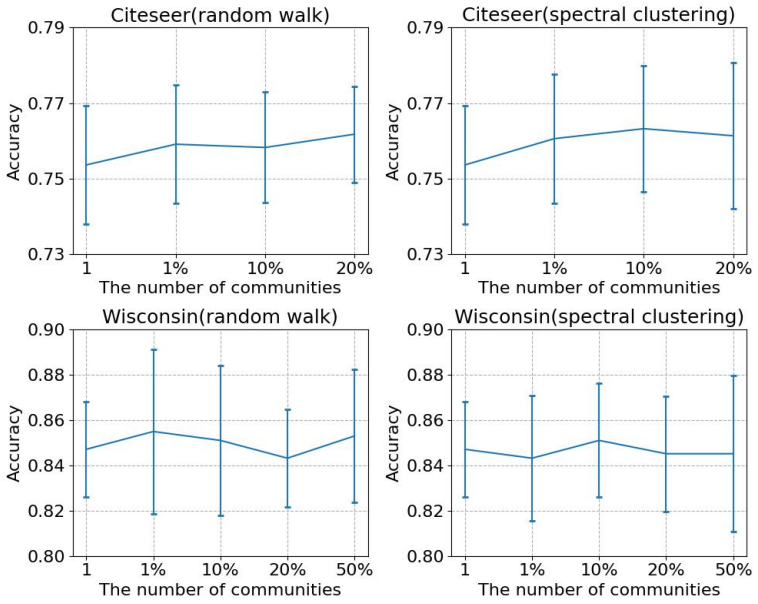

Figure 2: The ablation study on the number of communities. We set the number of communities to 1 (the whole graph as a community) and $1\%, 10\%, 20\%, 50\%$ of the number of graph nodes.

Effect of the Number of Community. We analyze the effect of the number of communities with the two unlearnable sampling methods for HOGT. From the results in Figure 2, we see that increasing the number of communities in the early stage can enhance the performance of HOGT (randomwalk) on Cora. This is because HOGT encodes more local higher-order information with more communities extracted by random walk. As the number of communities increases, we can observe a decreasing trend followed by an increase for HOGT with the spectral clustering method on Cora. This illustrates that there likely exist some important substructures in the graph. We also note the stable performance of HOGT on Wisconsin with different numbers of communities for both methods. While Wisconsin is a small-scale dataset, the global information can be well encoded by introducing a community.

Effect of Self-Attention Between Communities As we analyzed in Appendix A.4, if dropping out the second step (***C2C*-ATTN**), in terms of message-passing, HOGT behaves similarly to popular hypergraph-based neural networks. In this case, we are not taking into account the relationships between communities and we can see that in Table 13, HOGT (w/o ***C2C*-ATTN**)) exhibits a performance degradation compared to HOGT on datasets which have complex structure (like more nodes and edges). Without ***C2C*-ATTN**, the node representation is still limited in the local neighbourhood,

Table 13: Abalation study of different components of HOGT on different datasets (mean accuracy (%) and standard deviation over 10 different runs).

Community Sampling	Model	Cora	Citeseer	Cornell	Texas	Wisconsin
Random Walk	HOGT(w/o ***C2C*-ATTN**))	87.73±0.96	74.94±1.64	77.57±3.21	80.54±3.59	85.89±2.60
	HOGT	88.11±1.05	76.74±1.47	76.49±2.72	80.00±4.22	87.25±2.67
Random Walk	HOGT(w/o local)	83.04±1.48	74.47±2.10	76.49±2.72	82.70±4.86	83.44±1.87
	HOGT(w local)	88.11±1.05	76.74±1.47	70.27±2.34	74.90±2.78	78.19±2.67

i.e., community. Propagating information between communities can help the node finally capture the higher-order long-range dependency in the whole graph.

Effect of Local Information for Different Datasets Given one of the major advantages of Transformer is capturing the long-range dependency in objects, we examine the importance of local information for some of the benchmarks. From Table 13, we note that it can improve the performance if we consider the local neighbours in the third step (***G2V*-MP**) for Cora and Citeseer as they are small-scale datasets with high **Homo.**. In contrast, it is more beneficial to disregard the original graph connections for Cornell, Texas, and Wisconsin with low **Homo.**.

Performance on Graph Classification We utilize several commonly-used real-world datasets from TU database [51] to evaluate the performance of HOGT on graph classification task. **NCI1** consists of 4,110 molecule graphs from TUDataset, which represent two balanced subsets of datasets for chemical compounds screened for activity against non-small cell lung cancer and ovarian cancer cell lines, respectively. **PROTEINS** consists of 1,113 protein graphs from TUDataset, where each graph corresponds to a protein molecule, nodes represent amino acids, and edges capture the interactions between amino acids. From Table 14, we can observe that HOGT can achieve state-of-the-art performance on all datasets. Compared to GT models like GraphGPS, HOGT can encode more comprehensive information in the graph.

Table 14: Experimental results on two datasets (the mean accuracy (**Acc.**) and standard deviation over 10 different runs).

	NCI1	PROTEINS
GCN-based methods		
GCN [42]	$79.68_{\pm 2.05}$	$71.7_{\pm 4.7}$
GAT [64]	$79.88_{\pm 0.88}$	$72.0_{\pm 3.3}$
GIN [73]	$81.7_{\pm 1.7}$	$73.76_{\pm 4.61}$
GatedGCN [47]	$81.17_{\pm 0.79}$	$74.65_{\pm 1.13}$
Graph Transformer-based methods		
GT [17]	$80.15_{\pm 2.04}$	$73.94_{\pm 3.78}$
SAN [43]	$80.50_{\pm 1.30}$	$74.11_{\pm 3.07}$
Graphormer [75]	$81.44_{\pm 0.57}$	$75.29_{\pm 3.10}$
GraphTrans [70]	$82.60_{\pm 1.20}$	$75.18_{\pm 3.36}$
SAT [7]	$80.69_{\pm 1.55}$	$73.32_{\pm 2.36}$
GraphGPS [58]	$84.21_{\pm 2.25}$	$75.77_{\pm 2.19}$
GT(a whole graph as a community)	$\mathbf{84.67}_{\pm \mathbf{1.32}}$	$\mathbf{76.78}_{\pm \mathbf{1.84}}$