ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240759

Hop-Based Heterogeneous Graph Transformer

Zixuan Yang[a], Xiao Wang[b], Yanhua Yu[a,*], Yuling Wang[c], Kangkang Lu[a], Zirui Guo[a], Xiting Qin[a], Yunshan Ma[d] and Tat-Seng Chua[d]

[a]Beijing University of Posts and Telecommunications
[b]Beijing University of Aeronautics and Astronautics
[c]School of Cyberspace, Hangzhou Dianzi University
[d]National University of Singapore

Abstract. The Graph Transformer (GT) has shown significant ability in processing graph-structured data, addressing limitations in graph neural networks, such as over-smoothing and over-squashing. However, the implementation of GT in real-world heterogeneous graphs (HGs) with complex topology continues to present numerous challenges. Firstly, a challenge arises in designing a tokenizer that is compatible with heterogeneity. Secondly, the complexity of the transformer hampers the acquisition of high-order neighbor information in HGs. In this paper, we propose a novel **H**op-based **H**eterogeneous **G**raph Trans**former** (H^2Gormer) framework, paving a promising path for HGs to benefit from the capabilities of Transformers. We propose a Heterogeneous Hop-based Token Generation module to obtain high-order information in a flexible way. Specifically, to enrich the fine-grained heterogeneous semantics of each token, we propose a tailored multi-relational encoder to encode the hop-based neighbors. In this way, the resulting token embeddings are input to the Hop-based Transformer to obtain node representations, which are then combined with position embeddings to obtain the final encoding. Extensive experiments on four datasets are conducted to demonstrate the effectiveness of H^2Gormer.

1 Introduction

In recent years, there has been a significant surge of interest in graph neural networks (GNNs) due to their superior performance in handling various graph representation learning tasks. Early GNNs were initially developed for homogeneous graphs [13, 9], characterized by a single type of node and edge. However, heterogeneous graphs (HGs), prevalent and versatile in real-world scenarios and encountered in fields such as biological networks [15] and citation networks [26], offer a more realistic representation of complex relationships with richer semantics.

Recently, numerous Heterogeneous Graph Neural Networks (HGNNs) have been proposed to cope with challenges posed by heterogeneity [27, 8]. These HGNNs primarily build upon Message Passing Neural Networks (MPNNs), explicitly propagating information according to the input heterogeneous graph or pre-designed meta-paths. Consequently, current HGNNs also inherit various limitations of MPNNs that have already been proven to exist, such as over-smoothing [28], over-squashing [23], and under-reaching [21]. Taking inspiration from the success of the transformer in Natural Language Processing (NLP) [25], the Graph Transformer (GT) has recently been introduced to address the above issues in homogeneous/heterogeneous graphs. Unlike the conventional use of words as tokens in NLP, GT typically applies nodes or graph substructures as tokens [34, 7]. Then, a global attention mechanism is employed to learn potential connections between tokens, even when they are not connected in the input graph.

The key to extending transformers to HGs lies in how to design the tokenizer of the transformer. Roughly speaking, current GTs in HG mainly fall into the following two groups based on the style of the tokenizer: (a) Utilizing different nodes as tokens involves sampling the surrounding neighbors of the target node and employing the representations of these neighbors as input tokens [18]. (b) Utilizing high-level information of the target node as tokens. For example, SeHGNN [32] aggregates messages according to meta-paths, using information from different meta-paths as input tokens. MetaHGT [16] extracts hypergraphs based on meta-paths and employs the nodes and edges in the hypergraph as tokens. While promising, they also face significant challenges that cannot be ignored.

One challenge lies in designing a tokenizer that is compatible with heterogeneity and can effectively express heterogeneous semantics. Current heterogeneous graph GT [18] treats nodes as token information, employing additional modules to assess the heterogeneous information between nodes, but overlooks the connection between node features and heterogeneous information. Considering that tokens serve as the fundamental semantic units in the transformer computation process, there is an urgent need to incorporate advanced semantic information into tokens, e.g., the informative propagation results of initial features on HG structures.

Another challenge is that modeling the high-order neighbors in heterogeneous GTs often entries prohibitive increases in model complexity. The transformer structure imposes a substantial computational burden in contrast to traditional GNNs, attributable to its self-attention mechanism that computes relationships between any two tokens. In the context of HG scenarios, different hops of neighbors often represent distinct semantic information, extracting heterogeneous semantics based on high-order neighbors is indispensable for comprehending complex graph structures. However, the introduction of high-order neighbor information in heterogeneous GTs often requires additional domain knowledge to design complex meta-path or stacking more message passing layers, which often increases the model complexity [32]. Therefore, existing methods possess a limited receptive field and cannot flexibly incorporate

* Corresponding Author. Email: yuyanhua@bupt.edu.cn.

high-order neighbor information. In this paper, we propose the **H**op-based **H**eterogeneous **G**raph Trans**former** (H^2Gormer), which enables HGs to benefit from the understanding and expressive capabilities of transformers. Specifically, we propose a Heterogeneous Hop-based Token Generation module to consider the various hops of a node as distinct tokens, enabling the direct utilization of k-hop neighbor information during the generation of each token. Subsequently, we propose a Multi-Relational Encoder with a tailored attention mechanism to aggregate same-hop neighbor information into a token embedding. This introduces fine-grained heterogeneous semantics into tokens, i.e., the informative multi-relational propagation results in different levels of hop-based neighbors. To this end, the resulting token embeddings are input to the Hop-based Transformer to extract the hop-level relationships, which are then combined with position embeddings to obtain the final representations. In summary, our key contributions are as follows:

- We investigate how to effectively integrate transformers into HGNNs to capture richer heterogeneous semantics.
- We propose H^2Gormer, a new heterogeneous graph transformer model that can flexibly exploit high-order neighbor information and incorporate informative hop-based heterogeneous semantics into tokens.
- Extensive experiments on four datasets demonstrate that H^2Gormer achieves superior performance compared to state-of-the-art baselines in node classification.

2 Related Work

Graph Transformer. The transformer has revolutionized graph representation learning by treating nodes as tokens in a fully-connected graph, overcoming limitations (e.g.,over-smoothing, over-squashing, under-reaching) of traditional GNNs. A noteworthy issue is how to introduce the graph structure information into transformers. Some works incorporate Laplacian eigenvectors [7] or the nodes' random walk matrices [19] as positional encodings (PE). Others like [31] introduce inductive bias through an auxiliary message passing mechanism. Another concern is the computational complexity of extending transformers to large graphs. Some works modify the calculation of attention coefficients to reduce computational costs [29, 30], while others preprocess the graph data before calculating attention scores [5, 14]; the former adopts a subgraph sampling approach, and the latter computes attention coefficients only with cluster centers after graph clustering using KNN. NAGformer [6] designed a novel tokenizer to alleviate the time complexity problem of GT on large graphs. These models are primarily designed for homogeneous graphs and may not perform optimally on HGs. SeHGNN extends the GT to HGs which uses transformers to capture the relationships between different hop meta-path representations of nodes. However, meta-paths cannot be extracted automatically and require manual design.

Heterogeneous Graph Neural Network. HGNNs aim to leverage the rich semantic information alongside structure details in HGs to enhance representation learning, primarily dividing into two categories: meta-path-based and meta-path-free approaches. Meta-path-based methods utilize designed or extracted meta-paths to assist the message passing mechanism. For instance, HAN proposes an attention mechanism based on various meta-paths to learn semantic information [27] and MAGNN enhances HAN by introducing the missing node content features and intermediate nodes along meta-paths [8]. Meta-path-free HGNN models do not rely on meta-path but capture heterogeneity through carefully designed modules. RGCN [20] and e-RGCN [22] extend GCN by introducing relation-based graph convolution operations to handle different types of edges. To make multi-relational models more solid (with a theoretical foundation), EMR-GNN [28] and HALO [1] design a new perspective of optimization objectives and derive a new HGNN architecture. To address issues such as over-smoothing and over-squashing in current HGNNs, HINormer [18] introduces GT to learn the relationships between target nodes and their surrounding nodes. However, these models are difficult to utilize high-order neighbor information.

3 Proposed Method

In this section, we present a detailed description of our method, H^2Gormer, with its overall framework illustrated in Figure 1. Firstly, we categorize the neighbors of target nodes into different collections based on different hops. We introduce a multi-relational encoder with a tailored attention mechanism to aggregate neighbors in each collection and obtain token embeddings. Next, the hop-based transformer learns the semantic connections between tokens and integrates them into a node embedding. Concurrently, a graph-based position encoder is employed to capture the structural information of the nodes. Finally, the resulting node representation is fed through a predictor to predict the node labels. Subsequent sections will provide detailed explanations of the components mentioned above.

3.1 Notations

Consider a HG $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. Each edge $(v_i, v_j) \in \mathcal{E}$ has the relation type $r(i,j) \in \mathcal{R}$; each node v has its type $\phi(v)$ and node features $\mathbf{x}_v \in \mathbb{R}^{d_0}$, where r is relation mapping function and ϕ is node type mapping function. We use $\mathcal{T}_v = \{\phi(v) : v \in \mathcal{V}\}$ denote the set of node types and $\mathcal{R} = \{r(i,j) : (v_i, v_j) \in \mathcal{E}\}$ denote the set of relation types. If $|\mathcal{T}_v| = |\mathcal{R}| = 1$, the graph is homogeneous; otherwise, it is heterogeneous.

3.2 Heterogeneous Hop-based Token Generation

To empower Heterogeneous Graphs (HGs) with the expressive capabilities of transformers, a fundamental task is to design a tokenizer that is compatible with heterogeneity and capable of capturing heterogeneous semantics. Therefore, we introduce the Heterogeneous Hop-based Token Generation module to extract high-order heterogeneous neighbors and encode the rich semantics they contribute into tokens.

Hop-based Neighbors Generation. Considering a target node u, we categorize its neighbor set V into multiple collections based on the length of the shortest path from $v \in V$ to u, measured by $d(u, v)$:

$$\mathcal{S}_k(u) = \{v \in V : d(u, v) = k\}, \tag{1}$$

where $\mathcal{S}_k(u)$ represents the set of nodes for which the shortest path to u is k, and $k \in \{0, 1, ..., N\}$. Specially, we define $\mathcal{S}_0(u) = \{u\}$ which denotes the node itself.

Node Feature Projection. As features corresponding to different types of nodes reside in distinct feature spaces within Heterogeneous Graphs (HGs), we initially project node features of different types

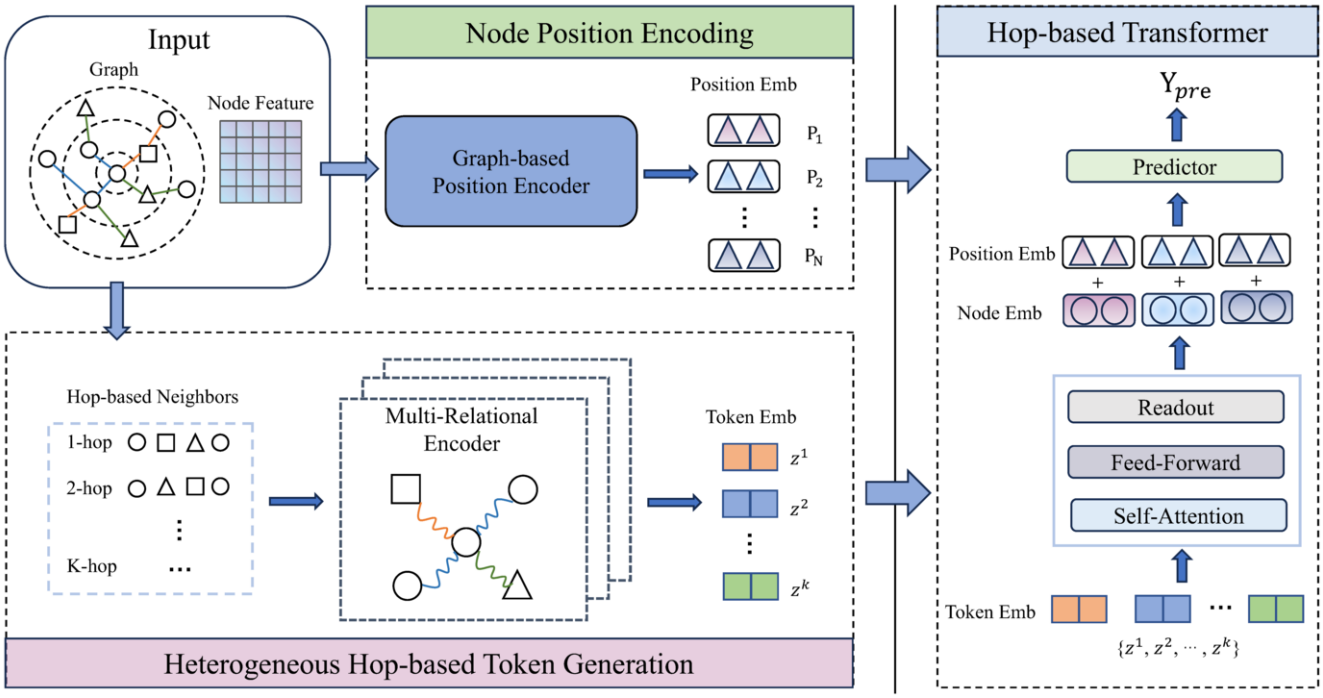


Figure 1. The overall framework of H^{2}Gormer.

into a shared feature space before proceeding with subsequent aggregation. In practice, we employ a simple linear layer for feature projection:

$$\mathbf{h}_u = \mathbf{W}_{\phi(u)}\mathbf{x}_u + \mathbf{b}_{\phi(u)}, \tag{2}$$

where $\mathbf{W}_{\phi(u)} \in \mathbb{R}^{d\times d_0}$ is a learnable matrix associated with the type of node u and $\mathbf{b}_{\phi(u)}$ is an optional bias. In this case, nodes of different types can be manipulated in the same feature space.

Token Embedding Generation. After acquiring the hop-based neighborhood structure and initial node features, we aggregate information from the same-hop neighbors of node u to obtain a hop-level representation. This process leads to obtaining token embeddings that encapsulate nuanced hop information at a fine-grained level:

$$\mathbf{z}_u^k = \sum_{v\in\mathcal{S}_k(u)} f(\mathbf{h}_u, \mathbf{h}_v), \tag{3}$$

where $\mathbf{z}_u^k$ represents k-th hop representation of node u, i.e., the token of the subsequent transformer structure. $f(\cdot)$ serves as the aggregate function. We approach the aggregation of nodes within each hop of the collection independently, allowing each node the flexibility to directly utilize information from its high-order neighbors. In practice, we propose a multi-relational encoder with heterogeneous information as the aggregate function $f(\cdot)$, which will be introduced in the following subsection.

Multi-relational Encoder. While we have proposed the independent encoding of fine-grained information for each hop, it is crucial to note that within the k-th hop neighbor set $\mathcal{S}_k$, multiple relations exist between each pair of nodes. Therefore, when aggregating neighbors at the same hop, the design of the aggregation function in Eq. (3), incorporating heterogeneous multiple relations, becomes crucial for enhancing the model's capacity to capture richer relation-level heterogeneity.

To accomplish this, we propose extending the attention mechanism by incorporating heterogeneous information about the relations between nodes. This involves calculating the attention coefficients between node u and v as follows:

$$\begin{aligned}\mathrm{Attn}(\mathbf{h}_u, \mathbf{h}_v) &= \mathbf{a}^\top \sigma((\mathbf{W}_Q\hat{\mathbf{h}}_u + \mathbf{W}_K\hat{\mathbf{h}}_v) + \mathbf{W}\mathbf{e}_{r(u,v)}),\\ \alpha_{uv} &= \frac{\exp(\mathrm{Attn}(\mathbf{h}_u, \mathbf{h}_v))}{\sum_{v'\in\mathcal{S}_k(u)}\exp(\mathrm{Attn}(\mathbf{h}_u, \mathbf{h}_{v'}))},\end{aligned} \tag{4}$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}$ are learnable weight matrix, $\mathbf{a}$ is a learnable vector and $\mathbf{e}_{r(u,v)}$ is learnable edge embedding related to node types. $\hat{\mathbf{h}}$ represents the features extracted from $\mathbf{h}$ through a feature extractor g_m; in this instance, we set g_m to be a Graph-based encoder [17]. σ is a non-linear activation function which is ReLU as default.

In this case, we aggregate the neighbors of node u at the same hop based on the attention coefficients to update its representation:

$$\mathbf{z}_u^k = \sigma\Big(\sum_{v\in\mathcal{S}_k(u)} \alpha_{uv}\mathbf{W}_V\hat{\mathbf{h}}_v\Big), \tag{5}$$

where $\mathbf{W}_V$ denotes learnable weight matrix. Furthermore, we implement a multi-head attention mechanism by assigning distinct weight parameters to different heads, aiming to stabilize the learning process and enhance the model's capacity. This implies that Eq. (3) with the function $f(\cdot)$ becomes:

$$\mathbf{z}_u^k = \sum_{h=1}^{N_h} \mathbf{W}_O^h \sigma\Big(\sum_{v\in\mathcal{S}_k(u)} \alpha_{uv}^h \mathbf{W}_V^h\hat{\mathbf{h}}_v\Big), \tag{6}$$

where $\mathbf{W}_O^h, \mathbf{W}_V^h$ is learnable weight matrix for each head h and N_h is the number of head. To prevent the model from having an excessive number of learnable parameters, which could elevate model

complexity and the risk of overfitting, we employ a shared multi-relational encoder $f(\cdot)$ to aggregate information from different hop neighbor sets.

In summary, through the introduction of attention mechanisms within different hops, incorporating edge information during the aggregation of same-hop neighbors, we achieve improved hop-level token representations by integrating the relation-enhanced HG information.

Explain the Multi-relational Encoder from Optimization Objective. To offer a more in-depth mathematical explanation of the underlying optimization objective of our designed multi-relational encoder, i.e., clarifying the information captured in the learned token embeddings from the optimization point of view, we represent the proposed multi-relational encoder as a graph denoising process with a defined optimization objective. Notably, the literature [33] defines the graph energy function as:

$$\|\mathbf{Z}-\mathbf{H}\|_F^2+\lambda\sum_{u,v\in\mathcal{E}}\|\mathbf{z}_u-\mathbf{z}_v\|_2^2, \tag{7}$$

where λ is a parameter, the first term is called the feature fitting term, while the second term, referred to as the graph smoothness term, aims to encourage similarity in representations of connected nodes. For a target node u, considering the nodes in k-hop neighbor sets $\mathcal{S}_k(u)$, where $k\in\{1,2,...,K\}$.

Lemma 1. *Following the computation of attention coefficients α_{uv} for node pairs, the aggregation process defined in Eq.(5) can be seen as an equivalent to a gradient descent procedure optimizing the following energy function:*

$$\|\mathbf{z}_u^k-\mathbf{h}_u\|_F^2+\lambda\sum_{u,v\in\mathcal{S}_k(u)}\alpha_{uv}\|\mathbf{z}_u^k-\mathbf{z}_v^k\|_2^2. \tag{8}$$

Proof: For given the attention score $\alpha_{u,v}$, we denote the energy function as $\ell(\mathbf{z})$. Here, we consider the gradient descent process from layer $l-1$ to l. The gradient of $\ell(\mathbf{z_u})$ w.r.t. $\mathbf{z_u}$ is

$$\nabla=2(\mathbf{z}_u^{l-1}-h_u)+2\lambda\sum\alpha_{u,v}(\mathbf{z}_u^{l-1}-\mathbf{z}_v^{l-1}). \tag{9}$$

When the step size of gradient descent is τ, the calculation process of $\mathbf{z}_u^l$ is as follows:

$$\begin{aligned}&\mathbf{z}_u^l=\mathbf{z}_u^{l-1}-\tau\nabla &(10)\\ =\;&\mathbf{z}_u^{l-1}-2\tau(\mathbf{z}_u^{l-1}-h_u)-2\tau\lambda\sum\alpha_{u,v}(\mathbf{z}_u^{l-1}-\mathbf{z}_v^{l-1}) &(11)\\ =\;&(1-2\tau-2\tau\lambda)\mathbf{z}_u^{l-1}+2\tau\mathbf{h}_u+2\tau\lambda\sum\alpha_{u,v}\mathbf{z}_v^{l-1}. &(12)\end{aligned}$$

Our gradient descent step is only one step, which means $z^{l-1}=h$. Afterword, our calculation is:

$$\mathbf{z}_u^l=(1-2\tau\lambda)\mathbf{h}_u+2\lambda\sum\alpha_{u,v}\mathbf{h}_v. \tag{13}$$

When we set $\tau=\frac{1}{2\lambda}$ and $\lambda=0.5$, we get the calculation formula for the multi-relational encoder considering one hop $\mathbf{z}_u^k$:

$$\mathbf{z}_u^k=\sum_{v\in\mathcal{S}_k(u)}\alpha_{uv}\mathbf{h}_v. \tag{14}$$

Non-linear activation functions can be introduced empirically. In this way, the multi-relational encoder can be viewed as the gradient descent process of an energy function.

When k approaches the diameter of the graph, the energy function becomes:

$$\sum_{k=1}^{K}\|\mathbf{z}_u^k-\mathbf{h}_u\|_F^2+\lambda\sum_{u,v}\alpha_{uv}\|\mathbf{z}_u-\mathbf{z}_v\|_2^2, \tag{15}$$

which can be seen as a graph smoothness criteria process for dense graphs. Therefore, our multi-relational encoder can be viewed as a node representation updating process based on a graph optimization objective.

3.3 Hop-based Transformer

There is a consensus that treating nodes as tokens in large HGs significantly increases computational complexity. This is attributed to the necessity of token augmentation to introduce additional inductive bias. In our proposed Hop-based Transformer, we shorten the token sequence corresponding to each node by considering each hop of neighbors as a token. This approach enables nodes to integrate with high-order neighbor information without requiring excessive layer stacking.

Inspired by [6], We specifically consider the combination of a node's heterogeneous hop-based token embeddings as a sequence:

$$\mathbf{Z}_u=[\mathbf{z}_u^0;\mathbf{z}_u^1;\mathbf{z}_u^2;...;\mathbf{z}_u^K], \tag{16}$$

where $\mathbf{Z}_u\in\mathbb{R}^{(K+1)\times d}$ incorporates the K-hop information of node u. Then, we feed $\mathbf{Z}_u$ into a standard transformer module with a multi-head self-attention (MSA) layer and a feed-forward neural network (FFN). Formally, the MSA involves projecting the input $\mathbf{Z}_u$ into a query ($\mathbf{Q}$), key ($\mathbf{K}$), and value ($\mathbf{V}$) through three distinct linear transformations, each with different parameter matrices. Then, we apply the scaled dot-product attention mechanism:

$$\mathrm{MSA}(\mathbf{Z}_u)=\mathrm{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d'}})\mathbf{V}, \tag{17}$$

where $\sqrt{d'}$ is the scaling factor and the softmax is applied row-wise. After this, the output of the MSA is fed into the FFN layer, each followed by a residual connection [10] and a normalization layer [2] (LN), denoted as:

$$\tilde{\mathbf{Z}}_u=\mathrm{MSA}(\mathrm{LN}(\mathbf{Z}_u))+\mathbf{Z}_u \tag{18}$$

$$\hat{\mathbf{Z}}_u=\mathrm{FFN}(\mathrm{LN}(\tilde{\mathbf{Z}}_u))+\tilde{\mathbf{Z}}_u. \tag{19}$$

After L layers we get $\hat{\mathbf{Z}}_u^{(L)}$, then we use a readout function to aggregate information from different hops to one embedding:

$$\mathbf{h}_u^{out}=\mathrm{READOUT}(\hat{\mathbf{Z}}_u^{(L)}). \tag{20}$$

In practice, we implement this simply with a mean readout function which is a non-parametric function as follows:

$$\mathbf{h}_u^{out}=\hat{\mathbf{Z}}_u^{(L)}[0:]+\frac{1}{K}\sum_{k=1}^{K}\hat{\mathbf{Z}}_u^{(L)}[k:], \tag{21}$$

where $\mathbf{Z}_u^{(L)}[k:]$ denotes the k-th row of $\mathbf{Z}_u^{(L)}$. Based on the heterogeneous hop-based transformer, we can flexibly acquire high-order neighbor information which is a challenge for traditional message-passing methods, and can capture the rich hop-level semantic correlations of a node's neighbors.

3.4 Node Position Encoding

Empirically, the local information of a graph can enable the model to learn better representations. To incorporate more local-based graph structure information, also known as graph inductive bias, into the model, we utilize a graph-based position encoder as follows:

$$\mathbf{p}_u = g_\theta(\mathbf{H})[u:\,], \tag{22}$$

where $g_\theta(\cdot)$ is a simple and efficient GNN which is the same as g_m and $\mathbf{H} = \{\mathbf{h}_u\}_{u=1}^N$ represents initial node features and u is one row of $\mathbf{H}$ denoting the embedding of node u. In this case, we obtain the position embedding of node u, i.e., $\mathbf{p}_u$, which represents the node's local information. Subsequently, we integrate the position embedding with the representation obtained from the heterogeneous hop-based transformer:

$$\mathbf{z}'_u = \mathbf{h}_u^{out} + \mathbf{p}_u, \tag{23}$$

where $\mathbf{z}'_u$ is the final representation of node u and can be used in downstream tasks. In this way, we simultaneously learn each node's high-order neighbor information and local neighbor information. The overall architecture of H^{2}Gormer is shown in Algorithm 1.

Algorithm 1 H^{2}Gormer

Input: Heterogeneous Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{R})$ and node features $\{\mathbf{x}_u : u \in \mathcal{V}\}$.
Output: Node representation $\mathbf{Z}'$.

```
for u ∈ V do
    Heterogeneous feature projection using Eq.(2).
    Calculate node position embedding using Eq.(22).
    for k=1...K do
        for v ∈ S_k(u) do
            Calculate attention coefficient using Eq.(4).
        end for
        Calculate token vector z_u^k using Eq.(5) or Eq.(6).
    end for
    Concat Z_u^(0) = [z_u^0; z_u^1; z_u^2; ...; z_u^K].
    Calculate z'_u using Eq.(19)-Eq.(23).
end for
return Z' = {z'_u}_{u=1}^N to downstream task
```

3.5 Training Objective

In this paper, our downstream task is semi-supervised node classification on HG. Formally, we use a predictor to predict the node's label based on its final representation $\mathbf{z}'_u$:

$$\hat{\mathbf{y}}_u = \mathrm{h}_\theta(\mathbf{z}'_u), \tag{24}$$

where h_θ is MLPs as a predictor with the learnable parameters θ. The overall training loss function is as follows:

$$\mathcal{L} = \sum_{u \in \mathcal{V}} \mathcal{D}(\hat{\mathbf{y}}_u, \mathbf{y}^*_u), \tag{25}$$

where $\mathcal{V}$ is the whole node set, and $\mathcal{D}$ is a discriminator function of Cross-Entropy, $\hat{\mathbf{y}}_u$ and $\mathbf{y}^*_u$ are the ground-truth and predicted label of node u. Backpropagation is used to optimize parameters with the guide of labeled data.

Datasets	Nodes	Node Types	Edges	Edge Types	Target	Classes
DBLP	26,128	4	239,566	6	author	4
ACM	10,942	4	547,872	8	paper	3
Freebase	43,854	4	151,034	6	movie	3
MUTAG	23,644	1	74,227	23	molecule	2

Table 1. Statistics of multi-relational datasets.

4 Experiment

4.1 Experimental Settings

Datasets. To test the performance of our model, we use four real-world heterogeneous graph datasets from different domains, including two academic citation datasets (DBLP [8] and ACM [17]), a knowledge graph dataset (Freebase [3]), and a complex molecules dataset (MUTAG [20]). The statistical information of the datasets is in Table 1. More information is as follows.

- **DBLP** is a computer science bibliography website. There are four types of nodes including 4057 authors, 14328 papers, 7723 terms, and 20 publication venues. The authors are categorized into four research directions, namely Database, Data Mining, Artificial Intelligence, and Information Retrieval.
- **ACM** is also a citation network. It contains 4019 papers, 7167 authors, and 60 subjects. The papers are divided into three classes according to the conference they published. Only paper's nodes have original attributes which are bag-of-words representations of their keywords, others do not have attributes.
- **Freebase** is a movie-related network. It contains 3492 movies, 33401 actors, 2502 directors, and 4459 writers.
- **MUTAG** describes the interactions between molecules. The nodes are divided into two classes as isMutagenic or not. We remove relations that were used to create entity labels, i.e., isMutagenic.

Baselines. To assess the model's performance, we choose nine state-of-the-art baselines, including:
(1) Homogeneous GNNs: GCN [13], GAT [25].
(2) Multi-relational GNNs: RGCN [20], EMR-GNN [28].
(3) HGNNs: HAN [27], HGT [11], SimpleHGN [17].
4) Heterogeneous GT: SeHGNN [32], HINormer [18].

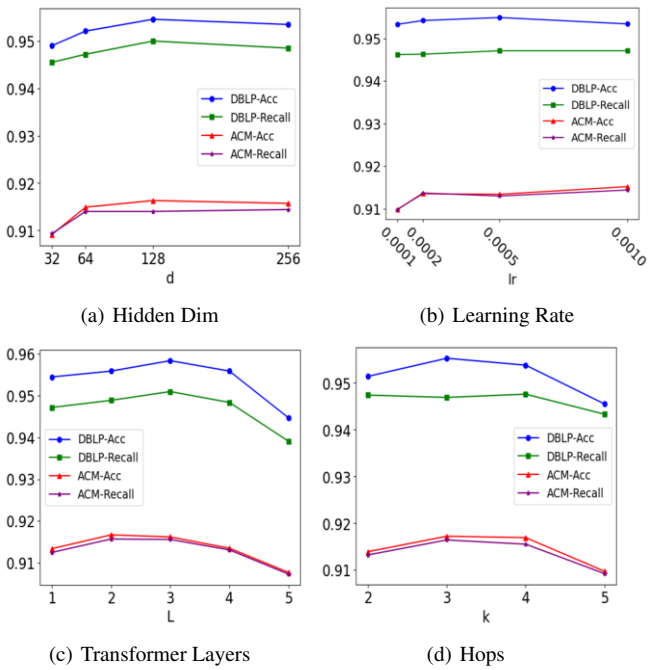
Parameter Settings. For the feature transformation layers and the predictor, we uniformly employ multi-layer MLPs across all datasets. For the hidden dimension of embeddings, we tune it from {64, 128, 256} for all datesets. For learning rate, we tune it from {1e-4, 2e-4, 5e-4, 1e-3}. For the number of transformer layers, we tune it from {2, 3, 4, 5}. For K of hop number, we tune it from {2, 3, 4, 5}. To simplify, we only use a single attention head in a hop-based transformer encoder for all datasets. For the dropout rate, we tune it from {0, 0.1, 0.2, 0.3, 0.4, 0.5}. We perform five iterations across all datasets using a consistent dataset division in every experiment. We implement our model based on PyTorch and use Adam optimizer [12].

4.2 Node Classification Results

Table 2 summarizes the performance of all methods in the semi-supervised node classification task. We utilize Accuracy and Recall, with standard deviation over five runs, as evaluation metrics. Some

Methods	DBLP		ACM		Freebase		MUTAG	
	Acc (%)	Recall (%)	Acc (%)	Recall (%)	Acc (%)	Recall (%)	Acc (%)	Recall (%)
GCN	90.39±0.38	89.49±0.52	89.58±1.47	89.47±1.49	68.35±0.11	62.32±0.21	72.35±2.17	63.28±2.95
GAT	91.97±0.40	91.25±0.58	88.99±1.58	88.89±1.56	67.32±0.77	58.26±2.13	70.74±2.13	63.01±3.79
RGCN	90.08±0.60	88.56±0.76	89.79±0.62	89.71±0.59	61.64±0.09	61.55±0.06	71.32±2.11	61.97±3.52
EMR-GNN	93.54±0.50	92.39±0.78	90.87±0.11	90.84±0.13	66.90±0.12	61.94± 4.49	74.26±0.78	**64.19±1.08**
HAN	91.73±0.61	91.15±0.72	88.51±0.35	88.50±0.30	62.34±2.13	59.13±1.91	-	-
SeHGNN	94.72±0.15	94.19±0.14	91.41±0.31	91.30±0.22	67.52±0.67	62.50±2.92	-	-
HGT	94.30±0.75	92.20±1.02	87.39±1.04	87.34±1.02	64.01±1.32	58.28±1.06	-	-
SimpleHGN	93.99±0.42	93.50±0.40	90.82±0.40	90.62±0.43	67.64±0.87	60.80±3.02	67.15±0.91	62.34±4.15
HINormer	94.87±0.17	94.32±0.13	90.12±1.20	89.54±1.23	68.53±0.28	62.80±0.81	72.71±1.61	61.25±1.17
Ours	**95.33±0.48**	**94.62±0.62**	**91.50±0.15**	**91.35±0.24**	**69.02±0.41**	**63.32±0.24**	**74.41±1.34**	63.39±2.23

Table 2. The mean and standard deviation of classification Accuracy and Recall over five different runs on four datasets.



(a) Hidden Dim (b) Learning Rate

(c) Transformer Layers (d) Hops

Figure 2. Analysis of parameters on DBLP and ACM.

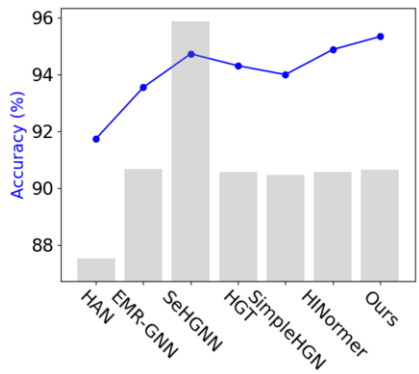


Figure 3. Parameters comparison. The height of the bars represents the relative quantity of model parameters. The x-axis represents different methods.

methods do not design specific meta-paths for the MUTAG dataset (where nodes only have one type and a high number of relations), so we do not reproduce the corresponding results for these methods. From the obtained results, we have the following observations:

- Our proposed H^{2}Gormer achieves state-of-the-art (SOTA) performance in seven out of eight indicators across all datasets. This demonstrates the promising effectiveness of our model on HGs. In particular, compared to HINormer, our proposed H^{2}Gormer outperforms it over Accuracy and Recall by 0.46% and 0.3% on DBLP. This is attributed to the efficacy of our multi-relational encoder and hop-based transformer, which better captures the connections between node feature information and heterogeneous information into embedding.
- Among all the baselines, Heterogeneous GTs, namely SeHGNN and HINormer, consistently achieve relatively good performance, generally outperforming HG models without a transformer structure. This demonstrates the effectiveness of the transformer structure in modeling complex heterogeneous semantics.

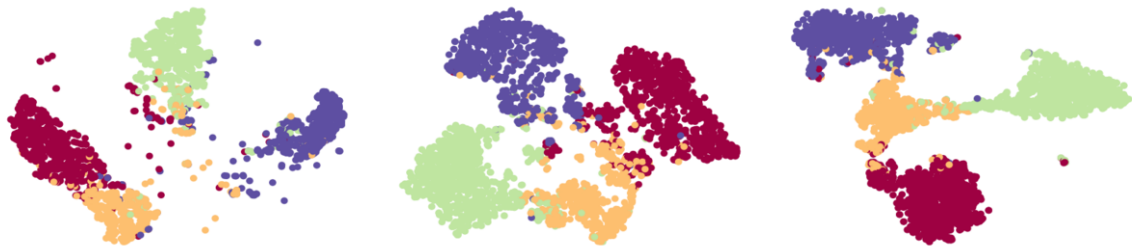
4.3 *Model Analysis*

Ablation Study. To verify the effectiveness of the components of our proposed model, we conduct ablation studies on all datasets. The result is shown in Table 3. We test three variants of the model: (a) Without heterogeneous information (HI). We remove the learnable edge features from Eq.(4), using standard GATv2 attention [4] instead. (b) Without semantic information (SI). After learning different hop information of the nodes, we remove the hop-based transformer encoder and directly obtained the final representation of the node through the readout function. (c) Without local structure information (LI). We remove the node position encoding of the model. From the results, we can observe:

- Without HI, there is a significant decline in the model's performance, reflecting the model's inability to consider heterogeneous information in the attention module, which is particularly evident in datasets with multiple types of relations. This also demonstrates the effectiveness of our proposed multi-relational encoder.

Methods	DBLP		ACM		Freebase		MUTAG	
	Acc (%)	Recall (%)	Acc (%)	Recall (%)	Acc (%)	Recall (%)	Acc (%)	Recall (%)
Ours	95.33±0.48	94.62±0.62	91.50±0.15	91.35±0.24	69.02±0.41	63.32±0.24	74.41±1.34	63.39±2.23
without HI	94.73±0.57	94.37±0.49	90.99±0.52	90.83±0.65	68.33±0.24	61.69±0.16	72.43±1.41	62.67±1.62
without SI	94.57±0.24	94.19±0.29	90.86±0.40	90.73±0.41	68.59±0.48	62.37±0.33	72.79±0.85	63.49±1.82
without LI	95.05±0.43	94.49±0.42	90.65±0.22	90.44±0.11	68.32±0.15	61.45±0.18	71.32±1.85	61.52±1.64

Table 3. Ablation study for our model.



(a) HGT (b) HINormer (c) Ours

Figure 4. Visualization of the learned node embeddings on DBLP dataset.

- Without SI, the model fails to capture the semantic information of different hops, leading to a decline in performance in all datasets. This suggests that capturing information from distant nodes and learning the semantic information between multi-hop neighbors can enhance the capability of HG representation learning.
- Consistent with empirical conjecture, incorporating local position information can improve the model's performance. So without LI, the model's performance declines due to the lack of local information of the node. Finally, the complete model achieves the best results.

Parameters Study. In addition to the model's performance, we also explore the sensitivity of some important parameters of our model such as hidden dim d, learning rate lr, the number of transformer layers L, and the hops of neighbor k on DBLP and ACM datasets, with results shown in the Figure 2. From the results, we can observe that:

- We observe that for the dimension d of the hidden layers, larger is generally better, as a greater dimension can help the model capture more information.
- For DBLP and ACM, as transformer layers L gradually increase, the model's performance progressively improves. However, if L continues to increase, performance begins to decline, indicating that too few layers may not capture the relations of different hops, while too many layers may lead to overfitting and diminish model performance.
- For hop number k, similar to empirical thinking, insufficient hop count fails to introduce enough neighbor information, while too many hops may introduce noise. The choice of k is dependent on the dataset.
- Additionally, we calculated the total number of parameters between our proposed model and several baselines, with the results shown in Figure 3. We observed that our model has advantages on both the parameters quantity and the model effect.

Visualization. For a more intuitive display and comparison, we visualize a contrast between our method and two other meta-path-free methods (HGT and HINormer) on DBLP; the result is shown in Figure 4. We reduce the dimensionality of the learned node embeddings using t-SNE [24] for visualization and color nodes with ground truth labels. In comparison, our proposed H^{2}Gormer outperforms the aforementioned techniques by demonstrating superior cluster formation with well-defined boundaries, signifying an advancement in embedding quality.

5 Conclusion

In this work, we propose H^{2}Gormer, a novel heterogeneous graph transformer model that includes a heterogeneous hop-based transformer and a multi-relational encoder to exploit high-order and heterogeneous information. Heterogeneous hop-based transformer constructs a sequence for each node's multi-hop neighbors, utilizing information from neighbors at different hops, and then learns the semantic information through a hop-based transformer. Furthermore, we enhance the model's ability to represent heterogeneity by using the multi-relational encoder to utilize node information and edge information. Experiments demonstrate the effectiveness of our proposed model.

Acknowledgements

The research was supported in part by the National Natural Science Foundation of China (No. 62172052, U22B2019).

References

[1] H. Ahn, Y. Yang, Q. Gan, T. Moon, and D. P. Wipf. Descent steps of a relation-aware energy produce heterogeneous graph neural networks. *Advances in Neural Information Processing Systems*, 35:38436–38448, 2022.

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

[4] S. Brody, U. Alon, and E. Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

[5] D. Chen, L. O'Bray, and K. Borgwardt. Structure-aware transformer for graph representation learning. In *International Conference on Machine Learning*, pages 3469–3489. PMLR, 2022.

[6] J. Chen, K. Gao, G. Li, and K. He. Nagphormer: A tokenized graph transformer for node classification in large graphs. In *The Eleventh International Conference on Learning Representations*, 2022.

[7] V. P. Dwivedi and X. Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

[8] X. Fu, J. Zhang, Z. Meng, and I. King. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference 2020*, pages 2331–2341, 2020.

[9] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Z. Hu, Y. Dong, K. Wang, and Y. Sun. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*, pages 2704–2710, 2020.

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[14] K. Kong, J. Chen, J. Kirchenbauer, R. Ni, C. B. Bruss, and T. Goldstein. Goat: A global transformer on large-scale graphs. In *International Conference on Machine Learning*, pages 17375–17390. PMLR, 2023.

[15] M. La Rosa, A. Fiannaca, L. La Paglia, and A. Urso. A graph neural network approach for the analysis of sirna-target biological networks. *International Journal of Molecular Sciences*, 23(22):14211, 2022.

[16] J. Liu, L. Song, G. Wang, and X. Shang. Meta-hgt: Metapath-aware hypergraph transformer for heterogeneous information network embedding. *Neural Networks*, 157:65–76, 2023.

[17] Q. Lv, M. Ding, Q. Liu, et al. Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1150–1160, 2021.

[18] Q. Mao, Z. Liu, C. Liu, and J. Sun. Hinormer: Representation learning on heterogeneous information networks with graph transformer. In *Proceedings of the ACM Web Conference 2023*, pages 599–610, 2023.

[19] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

[20] M. Schlichtkrull, T. N. Kipf, P. Bloem, et al. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.

[21] Q. Sun, J. Li, H. Yuan, X. Fu, H. Peng, C. Ji, Q. Li, and P. S. Yu. Position-aware structure learning for graph topology-imbalance by relieving under-reaching and over-squashing. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 1848–1857, 2022.

[22] T. Thanapalasingam, L. van Berkel, P. Bloem, and P. Groth. Relational graph convolutional networks: a closer look. *PeerJ Computer Science*, 8:e1073, 2022.

[23] J. Topping, F. Di Giovanni, B. P. Chamberlain, X. Dong, and M. M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*, 2021.

[24] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[26] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[27] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu. Heterogeneous graph attention network. In *The world wide web conference*, pages 2022–2032, 2019.

[28] Y. Wang, H. Xu, Y. Yu, M. Zhang, Z. Li, Y. Yang, and W. Wu. Ensemble multi-relational graph neural networks. *arXiv preprint arXiv:2205.12076*, 2022.

[29] Q. Wu, C. Yang, W. Zhao, Y. He, D. Wipf, and J. Yan. Difformer: Scalable (graph) transformers induced by energy constrained diffusion. *arXiv preprint arXiv:2301.09474*, 2023.

[30] Q. Wu, W. Zhao, C. Yang, H. Zhang, F. Nie, H. Jiang, Y. Bian, and J. Yan. Simplifying and empowering transformers for large-graph representations. *arXiv preprint arXiv:2306.10759*, 2023.

[31] Z. Wu, P. Jain, M. Wright, A. Mirhoseini, J. E. Gonzalez, and I. Stoica. Representing long-range context for graph neural networks with global attention. *Advances in Neural Information Processing Systems*, 34:13266–13279, 2021.

[32] X. Yang, M. Yan, S. Pan, X. Ye, and D. Fan. Simple and efficient heterogeneous graph neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 10816–10824, 2023.

[33] Y. Yang, T. Liu, Y. Wang, J. Zhou, Q. Gan, Z. Wei, Z. Zhang, Z. Huang, and D. Wipf. Graph neural networks inspired by classical iterative algorithms. In *International Conference on Machine Learning*, pages 11773–11783. PMLR, 2021.

[34] J. Zhao, C. Li, Q. Wen, Y. Wang, Y. Liu, H. Sun, X. Xie, and Y. Ye. Gophormer: Ego-graph transformer for node classification. *arXiv preprint arXiv:2110.13094*, 2021.