Advances in Continuous and Discrete Models

RESEARCH **Open Access**

EHG: efficient heterogeneous graph transformer for multiclass node classification

Man Wang[1*], Shouqiang Liu[1] and Zhen Deng[2*]

*Correspondence:
wangman@scnu.edu.cn;
dengzhencn83@smu.edu.cn
[1]School of Artificial Intelligence, South China Normal University, Foshan, 528000, China
[2]Neurology Department of Nanfang Hospital, Southern Medical University, Guangzhou, 510000, China

Abstract

Graph neural networks empowered by the Transformer's self-attention mechanism have arisen as a preferred solution for many graph classification and prediction tasks. Despite their efficacy, these networks are often hampered by their quadratic computational complexity and large model size, which pose significant challenges during graph training and inference. In this study, we present an innovative approach to heterogeneous graph transformation that adeptly navigates these limitations by capturing the rich diversity and semantic depth of graphs with various node and edge types. Our method, which streamlines the key–value interaction to a straightforward linear layer operation, maintains the same level of ranking accuracy while significantly reducing computational overhead and accelerating model training. We introduce the "EHG" model, a testament to our approach's efficacy, showcasing remarkable performance in multiclass node classification on heterogeneous graphs. Our model's evaluation on the DBLP, ACM, OGBN-MAG, and OAG datasets reveals its superiority over existing heterogeneous graph models under identical hyperparameter configurations. Notably, our model achieves a reduction of approximately 25% in parameter count and nearly 20% savings in training time compared to the leading heterogeneous graph-transformer models.

Keywords: Graph Neural Networks; Representation Learning; Graph Attention; Heterogeneous Graph; Graph Analysis; Graph Transformer; Knowledge Graph

1 Introduction

In the real world, network architectures frequently manifest as large-scale heterogeneous graphs, encompassing social networks, recommendation systems, academic graphs, and knowledge graphs. These heterogeneous graphs are comprised of a diverse array of nodes and edges, where (node-type, edge-type, node-type) triplets signify underlying semantic relationships, also known as metarelations. A case in point is the triplet ("Author", "writes", "Paper") found within the Open Graph Benchmark – Microsoft Academic Graph (OGBN-MAG) [1], which articulates distinct semantic information and epitomizes the most succinct metapath.

Over the past decade, there has been a surge in research dedicated to the exploration of heterogeneous graphs, with a particular emphasis on graph representation. Traditional nondeep-learning methodologies, such as DeepWalk [2], metapath2vec [3], PathSim [4], and ESim [5], predominantly focus on characterizing node features by measuring similarities between nodes or extracting structural insights through the navigation of metapaths. Conversely, deep-learning strategies leverage the principles of message passing and aggregation [6] within Graph Neural Networks (GNNs) [7], integrating advanced techniques like graph convolution, LSTM, attention mechanisms, and the Transformer. These have culminated in the creation of diverse architectural frameworks, including GCN [8], GAT [9], HAN [10], HetGNN [11], and HGT [12]. In the context of large-scale heterogeneous graphs, the conventional approach of loading the entire graph into memory for batch processing has been supplanted by more sophisticated subgraph sampling techniques. Innovations such as HGSampling [12], GraphSAGE [13], and FastGCN [14] have emerged to address these challenges. While Graph Neural Networks (GNNs) fortified with attention mechanisms have demonstrated remarkable success across a spectrum of downstream applications, they are not without their challenges, facing three issues:

Issue 1: They restrict their scope to n-hop local message passing among adjacent nodes and edges by uniformly distributing weights across a single level, and concurrently aggregate metapath semantic information on a different level, neglecting the broader global context of the entire graph.

Issue 2: The performance of the model is critically tied to the efficacy of the customized metapaths, which can lead to unstable or inadequate training process. Moreover, the selection of an inappropriate metapath might trigger gradient vanishing as a result of the recurrent cycles within the path.

Issue 3: The Graph Transformer's matrix multiplication operations for the Q (query), K (key), and V (value) matrices are computationally costly, often necessitating substantial memory and computational resources, particularly during the model-training phase.

Drawing inspiration from the advances in Vision Transformers (ViTs), specifically linear self-attention [15], separable self-attention [16], and efficient additive attention [17], we have developed an efficient transformer-based architecture designed for node-classification tasks within heterogeneous graphs. The essence of the separable self-attention [15, 16] approach entails the computation of context scores relative to a latent token. These scores are then used to reweight the input tokens and produce a context vector that encodes global information. This adjustment results in reducing the complexity of multiheaded self-attention (MHA) in transformers from $O\left(n^2\right)$ to $O(n)$, with n being the token count. In the realm of graph applications, input tokens can be viewed as representations of nodes. To enhance inference efficiency, the separable self-attention mechanism has supplanted the costly MHA dot product operations with simpler element-wise summation and element-wise multiplication operations. Similarly, efficient additive attention [17] introduces a hybrid model that maintains linear complexity, only focusing on element-wise interactions between queries and keys with tunable attention weights, rather than the traditional dot-product approach, thereby enhancing inference speed. In our framework, the input tokens are interpreted as graph embeddings, with context scores acting as global graph signals. On the one hand, we streamlined the interaction among Q (query), K (key), and V (value) by dispensing with the value computation and incorporating a learnable triplet tensor to extract global graph context, transcending the limitations of localized messaging. On the other hand, we employ element-wise operations to model query–key interactions instead of dot products, enhancing both training and inference speed.

Table 1 Time-complexity analysis of GAT, MLP, HAN, HGT, and our approach EHG

	Feature Projection	Neighbor aggregation	Semantic fusion	Total
GAT	$O\left(nd^2\right)$	$O(ed)$	/	$O\left(nd^2+ed\right)$
MLP	$O\left(nd^2\right)$		/	$O\left(nd^2\right)$
GCN	$O\left(nd^2\right)$	$O\left(ne^2\right)$	/	$O\left(nd^2+ne^2\right)$
HAN	$O\left(nd^2\right)$	$O(nke_md)$	$O\left(nd^2k\right)$	$O\left(nd^2k+nde_m\right)$
HGT	$O\left(nd^2\right)$	$O(ne_1d)$ $O(ne_1d)$		$O\left(n^2d^2e_1^2+nd^2\right)$
EHG	$O\left(nd^2\right)$	$O(ne_1d)$	$O(1)$	$O\left(nd^2+nde_1\right)$

Our approach, termed EHG, markedly reduces the time complexity on heterogeneous graphs from $O\left(n^2\right)$ to $O(n)$ [7, 18, 19], as delineated in Table 1. For the baseline models presented below, we presume a single-layered structure with a single attention head. Here, n represents the number of target-type nodes, d denotes the dimensions of both the input and hidden vectors, e quantifies the graph's edges, e_1 denotes the average number of one-hop neighbors, and e_m indicates the average number of neighbors within metapath graphs on the HAN model, with k representing the total number of metapaths.

The EHG model not only significantly boosts the efficiency of model training and inference but also marginally enhances the ranking accuracy of classifying nodes within heterogeneous graphs. It adeptly learns and adjusts the significance of various metarelations by substituting the V matrix with edge-type specific tensors. This innovation empowers the model to generate node embeddings that more effectively encapsulate the intricate global context and semantic richness inherent in heterogeneous graphs. After that, the model can be fine tuned to produce accurate node embeddings through backpropagation, customized to meet the specific demands of various downstream tasks.

The key contributions of our work are as follows:

• Through an indepth analysis of self-attention mechanisms within heterogeneous graphs, we have uncovered crucial insights. These insights not only highlight the redundancy of the V matrix but also establish the efficacy of employing element-wise operations for interactions between the Q and K matrices in graph transformers.

• Armed with these insights, we have developed the EHG model, which reduces the time complexity from quadratic to linear. EHG innovatively leverages learnable semantic tensors to autonomously generate metarelation importance scores and capture graph global context information, eliminating the reliance on predefined metapaths.

• Our extensive experiments on four diverse and widely adopted datasets confirm EHG's superiority over existing state-of-the-art models. It achieves superior ranking accuracy and expedites training, marking a significant advancement in the field.

2 Preliminaries and related work

In this section, we delve into the fundamental concepts and theoretical frameworks of heterogeneous graphs, complemented by an overview of the latest advancements in heterogeneous graph neural networks. Subsequently, in the forthcoming section, we analyze and highlight the differences between our approach and existing methodologies, explaining why EHG outperforms them in the multiclass node-classification task.

2.1 Heterogeneous graph data mining

Definition 2.1 *Heterogeneous graphs.* A heterogeneous graph is a directed graph denoted as $G=(\mathcal{V},\mathcal{E},\mathcal{A},\mathcal{R})$, consists of a node set $\mathcal{V}$ and an edge set $\mathcal{E}$, where each $\upsilon\in\mathcal{V}$ is associated with a node-type mapping function $\tau(\upsilon):\mathcal{V}\to\mathcal{A}$ and each $e\in\mathcal{E}$ is associated with

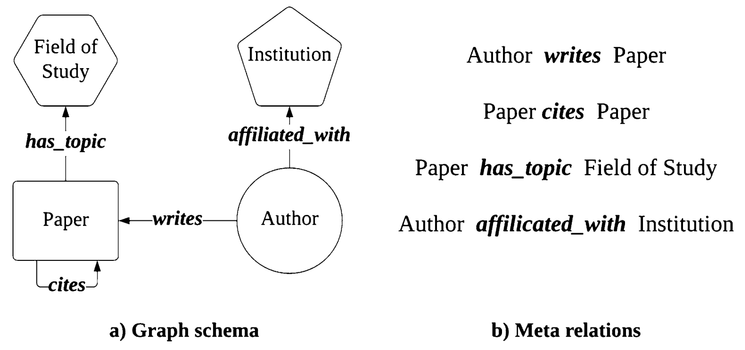

Figure 1 The schema and metarelations of OGBN-MAG (Open Graph Benchmark – Microsoft Academic Graph). As an example of a heterogeneous graph, the academic network OGBN-MAG exhibits two main characteristics: a) the assignment of unique types to each node and edge, which together constitute the graph's schema; b) the one-hop edges that link source nodes to target nodes, embodying metarelations and transmitting the semantic information of the graph

an edge-type mapping function $\phi(e) : \mathcal{E} \to \mathcal{R}$. $\mathcal{A}$ and $\mathcal{R}$ denote the sets of predefined node types and edge types, where $|\mathcal{A}| + |\mathcal{R}| > 2$.

Definition 2.2 *Metarelation and Metapath.* For an edge $e = (s, t)$ linked from source node s to target node s, its metarelation is denoted as $\langle \tau(s), \phi(e), \tau(t) \rangle$. A metapath is defined as a path that describes a composite metarelation between different nodes.

Example As shown in Fig. 1, two papers can be linked via the metarelation "Papers-Cites-Papers", while authors and fields of study are connected through the metapath "Authors-Writes-Papers-has_topic-Fields of Study." Different metapaths capture distinct semantic meanings and uncover relationships between various nodes and edges.

2.2 Graph attention and graph transformer

The Graph Attention Networks (GATs) integrate an attention mechanism into the framework of Graph Convolutional Networks (GCNs), endowing nodes with the capability to selectively aggregate information from neighboring nodes. In essence, the features of each neighbor are processed via a shared, learnable matrix, followed by a normalization step that yields unique attention scores. These scores are then multiplied with the target node's features from the previous layer, thereby aggregating its representation at the current level. To elucidate this concept, we will initially explore the scenario of single-head attention:

$$h_i^{(l+1)} = \sigma\Big(\sum_{j \in N(i)} a_{ij}^{(l)} \boldsymbol{W}^{(l)} h_i^{(l)}\Big), \tag{1}$$

where $h_i^{(l+1)}$ represents the embedding of node i in the $l+1$ layer, while $N(i)$ means all the neighbor nodes of node i. $a_{ij}^{(l)}$ is the attention score between node i and its neighboring node j after normalization. To obtain the normalized attention score $a_{ij}^{(l)}$, GAT [9] compute them across all choices of j using the Softmax function:

$$a_{ij}^{(l)} = Softmax_{j \in N(i)}\left(LeakyRelu\left(a^{(l)}\left(\boldsymbol{W}^{(l)} h_i^{(l)} \| \boldsymbol{W}^{(l)} h_j^{(l)}\right)\right)\right). \tag{2}$$

Here, $a^{(l)}$ is a learnable vector, and $W^{(l)}$ is a learnable matrix. Softmax normalization is applied to each node pair after the concatenation operation and LeakyReLU [20] activation. $\|$ is the contraction operation. In the original GAT paper, $W^{(l)}$ and $a^{(l)}$ actually share the same weights across all layers.

In multihead attention calculation, K represents the number of attention heads, and k refers to the kth attention head. the output features of each attention head are concatenated after nonlinear activation before reaching the final prediction layer:

$$h_i^{(l)} = \|_{k=1}^{K} \sigma \left(\sum_{j \in N(i)} a_{ij}^{k} \boldsymbol{W}^{\boldsymbol{k}} h_j^{(l-1)} \right). \tag{3}$$

Next, all the neighbor node embeddings are summed up via attention transformation, averaged by the number of attention heads, and then a nonlinear activation is applied in the last layer for downstream classification task:

$$h_i^{(l+1)} = \sigma \left(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in N(i)} a_{ij}^{k} \boldsymbol{W}^{\boldsymbol{k}} h_j^{(l)} \right). \tag{4}$$

The aforementioned equations succinctly delineate the three pivotal stages inherent in graph-attention mechanisms: initially, the computation of the attention score $a_{ij}^{(l)}$ for each pair of neighboring nodes; subsequently, the aggregation of attention scores and neighboring node embeddings; and ultimately, the activation of an aggregated message from the previous stage. GATs predominantly consider the structural information of graphs, often neglecting the influences emanating from different node and edge types. It can engender substantial redundant computation, especially in graphs where neighborhoods often highly overlap. Consequently, while GATs excel in node-classification tasks within homogeneous graphs, they often fall short in adequately capturing the semantic information in heterogeneous graphs.

In order to integrate semantic information, the heterogeneous graph attention network (HAN) [10] introduces a metapath-based hierarchical model. This model learns node-level and semantic-level attention in two separate phases. Nevertheless, the selection of metapaths is a manual process, and improper selection may adversely affect the classification outcomes. Moreover, the model's complexity is linear to the number of metapaths; consequently, an increase in metapaths leads to a substantial rise in training time.

Heterogeneous Graph Transformer (HGT) [12], inspired by the architecture design of Transformer [21], avoids the need for customized metapaths and being scalable to Web-scale graphs. It projects the target node into a Query vector and source node into a Key vector, with their dot product being utilized to compute attention. HGT introduces matrices that cater to diverse edge types and employs adaptive scaling tensors tailored for metarelations, empowering the model to autonomously learn and dynamically adjust the attention importance of various edge and node types.

For MHA calculation, source node s and target node t are connected by a directed edge e, each source–target node pair is normalized to obtain the attention score before aggregation as shown in equations (5) and (6):

$$Attention\,(s,e,t) = \text{Softmax}_{\forall s \in N(t)} \left(\|_{k=1}^{K} ATT - head^{k}\,(s,e,t) \right) \tag{5}$$

$$ATT-head^{k}(s,e,t)=\left(K^{k}(s)\,\boldsymbol{W}^{\boldsymbol{ATT}}_{\boldsymbol{\phi(e)}}Q^{k}(t)^{T}\right)\cdot\frac{\mu_{\langle\tau(s),\phi(e),\tau(t)\rangle}}{\sqrt{d}} \tag{6}$$

$$K^{k}(s)=K-Linear^{k}_{\tau(s)}\left(h^{(l)}_{s}\right) \tag{7}$$

$$Q^{k}(t)=Q-Linear^{k}_{\tau(t)}\left(h^{(l)}_{t}\right), \tag{8}$$

where $\tau(\cdot)$ represents the node type, and $\phi(\cdot)$ represents the edge type. $K^{k}(s)$ and $Q^{k}(t)$ denote the kth head key and query vectors after linear transformation. The functions $Q-Linear^{k}_{\tau(t)}$ and $K-Linear^{k}_{\tau(s)}$ are linear projections $R^{d}\rightarrow R^{\frac{d}{K}}$, where $\frac{d}{K}$ is the vector dimension per head. Unlike the direct dot product between Q vectors and K vectors in a vanilla transformer, HGT introduces a distinct $\boldsymbol{W}^{\boldsymbol{ATT}}_{\boldsymbol{\phi(e)}}\in R^{\frac{d}{K}\times\frac{d}{K}}$ matrix for each edge type, accounting for the influence of different edge types, and providing more accurate semantic information from various node and edge types. Additionally, since each metarelations contributes differently to the target node, a tensor $\mu_{\langle\tau(s),\phi(e),\tau(t)\rangle}$ is introduced to further scale the attention.

For message passing, HGT calculates a pair of nodes multihead message information as follows:

$$\text{Message}_{(s,e,t)}=\|_{k=1}^{K}\,V^{k}(s)\,\boldsymbol{W}^{\boldsymbol{MSG}}_{\boldsymbol{\phi(e)}} \tag{9}$$

$$V^{k}(s)=\text{V}-\text{Linear}^{k}_{\tau(s)}\left(h^{(l)}_{s}\right). \tag{10}$$

A source node s of $\tau(s)$-type is projected into the kth message vector with a linear projection $V-Linear^{k}_{\tau(s)}:R^{d}\rightarrow R^{\frac{d}{K}}$ to generate the V vectors. Subsequently, the matrix $\boldsymbol{W}^{\boldsymbol{MSG}}_{\boldsymbol{\phi(e)}}$ is applied to incorporate the edge dependency. After contracting all message heads, each node pair $\text{Message}_{(s,e,t)}$ is formed.

The final step involves aggregating the previous calculated attention and message information, then mapping the vector of node t into its type-specific distribution using projection $A-Linear_{\tau(t)}$, followed by a residual connection [22]. Each node embedding $h^{(l)}_{t}$ can then be fed into various models to conduct downstream heterogeneous network tasks, such as node classification and link prediction:

$$\overline{h^{(l)}_{t}}=\oplus_{\forall s\in N(t)}\left(Attention(s,e,t)\cdot Message(s,e,t)\right) \tag{11}$$

$$h^{(t)}_{t}=A-Linear_{\tau(t)}\left(\sigma\left(\overline{h^{(l)}_{t}}\right)\right)+h^{(l-1)}_{t}. \tag{12}$$

Both HGT and HAN outperform GAT in node-classification tasks on heterogeneous graphs, with HGT having a particular advantage in handling more complex and large-scale heterogeneous graphs.

3 The proposed architecture

Recent studies have confirmed the efficacy of attention mechanisms in addressing a spectrum of challenges in graph-node classification and link prediction. Typical graph transformers encode relevance scores for the contextual information of input features, relying on the interactions among the trio of attention components (Q, K, V). However, this approach exacts a high computational toll, with complexity increasing significantly, especially when dealing with large heterogeneous graphs and complex metarelations that

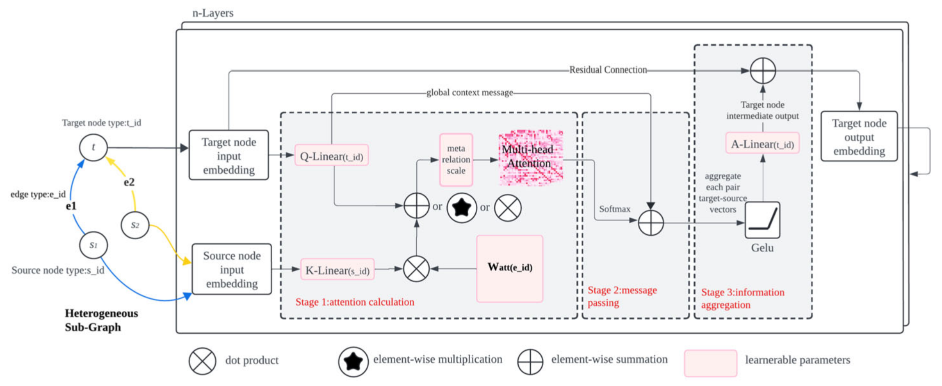

Figure 2 The Overall Architecture of an Efficient Heterogeneous Graph Transformer. A heterogeneous subgraph is sampled with target node and its source nodes, where each node or edge is tagged with its corresponding type identifier before being sent to the type-specific linear layer. The subsequent operations are orchestrated as follows: 1) The computation of attention scores based on the Q (Query) and K (Key) matrices, 2) The multiplication of context messages to facilitate message passing, and 3) The aggregation of the vectors from each target–source pair into one, which is subsequently activated and sent to the next layer

involve numerous node and edge types. Research on efficient adaptive attention [11] has shown that the removal of key–value interactions does not compromise performance; instead, a focused and effective encoding of query–key interactions, incorporated by a linear projection layer, is sufficient to learn the interrelationships among embeddings. Therefore, we integrate an adaptive mechanism into heterogeneous graphs, as depicted in Fig. 2, which consists of three stages: attention calculation, message passing, and information aggregation.

First, to calculate the source–target node pair attention, we use a simplified metarelation scale-learnable tensor $R_{\langle\phi(e),k\rangle}$, which is initialized as a tensor of ones for each edge type per attention head. Additionally, we introduce a global query tensor $q_{\phi(\cdot)}$ to scale the query matrix across each dimension. To simplify the explanation, we treat the Key vector of source node s after linear projection and reshaping to $R^{k\times\frac{d}{K}}$ as the K matrix, and the Query vector of target node t after linear projection and reshaping to $R^{k\times\frac{d}{K}}$ as the Q matrix, respectively. Next, the key matrix K is multiplied by $\boldsymbol{W}^{ATT}_{\boldsymbol{\phi(e)}}$, followed by an element-wise addition, multiplication or dot product with the Q matrix. The result is then multiplied by $R_{\langle\phi(e),k\rangle}$, which adjusts the attention scores according to specific edge type. The changes during training of $R_{\langle\phi(e),k\rangle}$ reveal the edge-type importance and implicit global semantic information. Under certain conditions, we have observed that element-wise operations on the Q and K matrices outperform dot product. Equation (13), (14), (15), and (16) illustrate the complete mathematical attention calculation process:

$$ATT-head_{EHG-sum}(s,e,t) = \left(K\boldsymbol{W}^{ATT}_{\boldsymbol{\phi(e)}} + Q\boldsymbol{q}_{\boldsymbol{\phi}(\cdot)}\right)\frac{R_{\langle\phi(e),k\rangle}}{\sqrt{d}} \tag{13}$$

$$ATT-head_{EHG-mul}(s,e,t) = \left(K\boldsymbol{W}^{ATT}_{\boldsymbol{\phi(e)}} \otimes Q\boldsymbol{q}_{\boldsymbol{\phi}(\cdot)}\right)\frac{R_{\langle\phi(e),k\rangle}}{\sqrt{d}} \tag{14}$$

$$ATT-head_{EHG-dot}(s,e,t) = \left(K\boldsymbol{W}^{ATT}_{\boldsymbol{\phi(e)}} \odot Q\boldsymbol{q}_{\boldsymbol{\phi}(\cdot)}\right)\frac{R_{\langle\phi(e),k\rangle}}{\sqrt{d}} \tag{15}$$

$$Attention_{EHG}(s,e,t) = Softmax_{\forall s\in N(t)}\left(ATT-head_{EHG}(s,e,t)\right). \tag{16}$$

Second, we entirely remove the V-liner layer and edge-type specific matrix $\boldsymbol{W}^{\boldsymbol{MSG}}_{\boldsymbol{\phi(e)}}$ from the original HGT. Instead, we directly use Q from the attention stage, rather than V, as the context message. Q serves as the target-node value the model is searching for, which is added to the attention output after applying Softmax function. The rationale behind this approach is that Q already integrates information from target-node type, while the attention calculation has already combined the source-node type and edge-node type. By doing so, shared weights are strengthened from the perspective of the graph's global metarelations and global query embeddings, rather than being limited to local query–key–value interactions. This retains the benefits of the self-attention mechanism and allows us to capture information from heterogeneous edges and node types. Take the EHG-sum approach for instance:

$$\overline{h_t^{(l)}} = \oplus_{\forall s\in N(t)} \left(Q + \text{Softmax}_{\forall s\in N(t)} \left(\left(K \boldsymbol{W}^{\boldsymbol{ATT}}_{\boldsymbol{\phi(e)}} + Q \right) \frac{R_{\langle \phi(e),k \rangle}}{\sqrt{d}} \right) \right). \tag{17}$$

Third, we apply a linear transformation $A-Linear_{\tau(t)}$ layer to the query–key interactions to learn the hidden representation of the nodes. We use the GELU (Gaussian Error Linear Unit) [23] activation function because it outperforms RELU in transformer models by better fitting the data distribution. The final output embedding vector of the target node can be described as (18). For downstream tasks like the node classification, it is used to generate prediction results:

$$h_t^{(l)} = A - Linear_{\tau(t)} \left(Gelu \left(\overline{h_t^{(l)}} \right) \right) + h_t^{(l-1)}. \tag{18}$$

Compared to HGT, our proposed method directly uses the global query Q matrix derived from the attention phase as the message for passing, instead of introducing an additional learnable V matrix. This streamlined strategy achieves a dual objective: it minimizes the number of parameters and bolsters computational efficiency. The embeddings for the target nodes are generated through the aggregation of attention scores and messages from all associated source nodes. These refined node embeddings can subsequently be deployed in a variety of downstream applications, including node classification and link prediction. Algorithm 1 provides a comprehensive breakdown of the EHG's overall process.

4 Analysis and comparison with different self-attention modules

In this section, we analyze the relationship between vision transformers and graph transformers, and explain why EHG maintains efficiency on heterogeneous graphs by streamlining the V matrix and employing element-wise operations exclusively for interactions between the Q and K matrices.

The attention mechanism has quadratic complexity due to the dot-product operation between the Q and K matrices, followed by a Softmax normalization for each token. In the original transformer model developed for the Natural Language Processing (NLP) domain [21], each word in a sequence of n words must attend to every other word to understand its context, resulting in $n \times n$ pairwise interactions. Positional encoding further enables the transformer model to capture both local and global positional relationships within the sequence.

Vision transformers (ViTs), which are built on self-attention mechanisms, can effectively model interactions between input tokens and have demonstrated remarkable success across various vision tasks. Although ViTs outperform convolutional neural networks

Algorithm 1 The overall process of EHG

```
Input: The heterogeneous graph $G = (V, E, A, R)$; the node-type numbers $|A|$; the
edge-type numbers $|R|$; the node features $\{h_i, \forall i \in V\}$; the number of attention heads $K$;
output feature dimension $d$.
Parameters: Feature projection linear layers: $K - Linear^k_{\tau(\cdot)}$, $\mathrm{Q} - \mathrm{Linear}^k_{\tau(\cdot)}$, and
$\mathrm{A} - \mathrm{Linear}^k_{\tau(\cdot)}$; global query tensor $q_{\phi(\cdot)}$; edge-type-based matrix $W^{ATT}_{\phi(\cdot)}$; edge-type-based
scale-learnable tensor $R_{\langle\phi(\cdot),k\rangle}$.
Output: The final embeddings $\{Z_i, \forall i \in V\}$.
1: Initialize $K, Q, A$ linear layers for each node type
2: Initialize tensor $R_{\langle\phi(\cdot),k\rangle}$ with shape $\left(|R|, K, \frac{d}{K}\right)$
3: Initialize tensor $q_{\phi(\cdot)}$ with shape $\left(|R|, \frac{d}{K}\right)$
4: Initialize random and uniform matrix $W^{ATT}_{\phi(\cdot)}$ for each head and each edge type with
shape $\left(|R|, K, \frac{d}{K}, \frac{d}{K}\right)$
# attention-score calculation
5: For each (source-node type, edge-node type, target node type) in $G$ by edge type do:
6:     $G_{subgraph} \leftarrow G$ sampled by (source-node type, edge-node type, target-node type)
7:     $\mathbf{K}$ matrix $\leftarrow K - Linear^k_{\tau(s)}$. reshape $\left(K, \frac{d}{K}\right)$
8:     $\mathbf{Q}$ matrix $\leftarrow Q - Linear^k_{\tau(t)}$. reshape $\left(K, \frac{d}{K}\right)$
9:     $\mathbf{K}$ matrix $\leftarrow \mathbf{K} \odot W^{ATT}_{\phi(e)}$
10:    $\mathbf{Q}$ matrix $\leftarrow \mathbf{Q} \times q_{\phi(\cdot)}$
11:    attention_score $\leftarrow \mathbf{Q} \odot \mathbf{K} \times R_{\langle\phi(\cdot),k\rangle} / \sqrt{\frac{d}{K}}$
12:    attention_score $\leftarrow$ Softmax attention_score by subgraph edge type
# message passing
13: For each edge type do:
14:    Message $\mathbf{M}$ from edges that point to target node $t \leftarrow \mathbf{Q}+$ attention_score
15:    update target node feature $h_t$ by mean value of $\mathbf{M}$
# information aggregation
16: For each node type do:
17:    $h_t \leftarrow A - Linear^k_{\tau(t)}(t)$
18: Concatenate the learned node features from all attention heads
19: Apply activation function GELU and transfer out to next level until the end of
forward propagation
20: Calculate Crossentropy
21: Back propagation and update parameters in EHG
22: return the output presentation of nodes
```

(CNNs) in capturing global features, deploying them on resource-constrained mobile devices for real-time applications remains challenging due to their quadratic complexity. This challenge has led to the development of efficient alternatives, such as linear self-attention [15], separable self-attention [16], and efficient additive attention [17].

Graph transformers, which closely follow the original transformer architecture [21], take as input a graph with nodes and edges representing relationships between the nodes [24]. However, similar to ViTs, they encounter efficiency challenges, particularly when handling large-scale, complex heterogeneous graphs. During the preprocessed graph loading phase and local neighbor message-passing phase, these challenges often result in out-of-memory

(OOM) issues or excessively long processing times [25]. Here, we conduct a comprehensive analysis and comparison of our proposed method with several transformer models, as well as the Heterogeneous Graph Transformer (HGT) model, to highlight its advantages and efficacy as shown in Fig. 3:

Our EHG model is based on the recently introduced Effective Additive Attention mechanism, which produces more robust contextual representations, as demonstrated by its performance on computer vision tasks. We combine this mechanism with the HGT model to effectively represent heterogeneous graph nodes. Since graph features tend to become smooth through global node-type and edge-type aggregation at the subgraph level, encoding query–key interactions by incorporating a linear projection layer—without the need for value–matrix interactions—is sufficient to capture the relationship between target-node features and source-node features. Furthermore, we introduce learnable global query vectors, denoted as $q_{\phi(\cdot)}$, to fine tune node features by dividing the feature dimension by the number of attention heads. These global query vectors also capture global edge-type features, which help produce the global query matrix and extract global node features.

In summary, the EHG model is carefully designed to efficiently encode both local and global representations in the attention calculation, and then propagate the normalized message through one-hop metarelations. This design enables EHG to achieve both efficiency and effectiveness in handling complex heterogeneous graphs.

5 Evaluation

5.1 Datasets

We employ four datasets of varying sizes to perform node-classification tasks. The statistics for each dataset are listed in Table 2. The DBLP dataset contains 26,128 nodes and 239,566 edges, with authors labeled into 4 categories. The ACM dataset includes 30,003 nodes and 160,686 edges, where papers are classified based on the conference they were published in, resulting in 14 classes. The OGBN-MAG dataset consists of 1,939,743 nodes and 21,111,007 edges, with papers labeled according to their venue, which results in 349 classes. The OAG-Venue dataset contains 1,116,162 nodes and 13,985,692 edges, where papers are also labeled based on their venue, classifying them into 2506 classes. It is important to note that DBLP and ACM are connected heterogeneous graphs, while OGB-MAG and OAG-Venue are not. A connected heterogeneous graph means that each edge has a corresponding reverse edge, ensuring that any node is reachable from any other node. Among these datasets, DBLP and ACM are small heterogeneous graphs, while OGBN-MAG and OAG-Venue are large heterogeneous graphs.

5.2 Baselines

We compare the proposed EHG with a suite of established heterogeneous graph neural network-based approaches baselines, such as GAT, MLP, RGCN, HAN, and HGT, to verify its effectiveness. The EHG model is further delineated into EHG-sum, EHG-mul, and EHG-dot, which correspond to the application of element-wise addition, element-wise multiplication, and dot product in the attention-head calculation, respectively, as delineated in Equations (13), (14), and (15).

Graph Attention Networks (GAT) [9], which adopt multihead additive attention on neighbors and assigns different importances to nodes of the same neighborhood.

Relational Graph Convolutional Networks (RGCN) [26], which simply average the neighbor's embedding followed by linear projection.

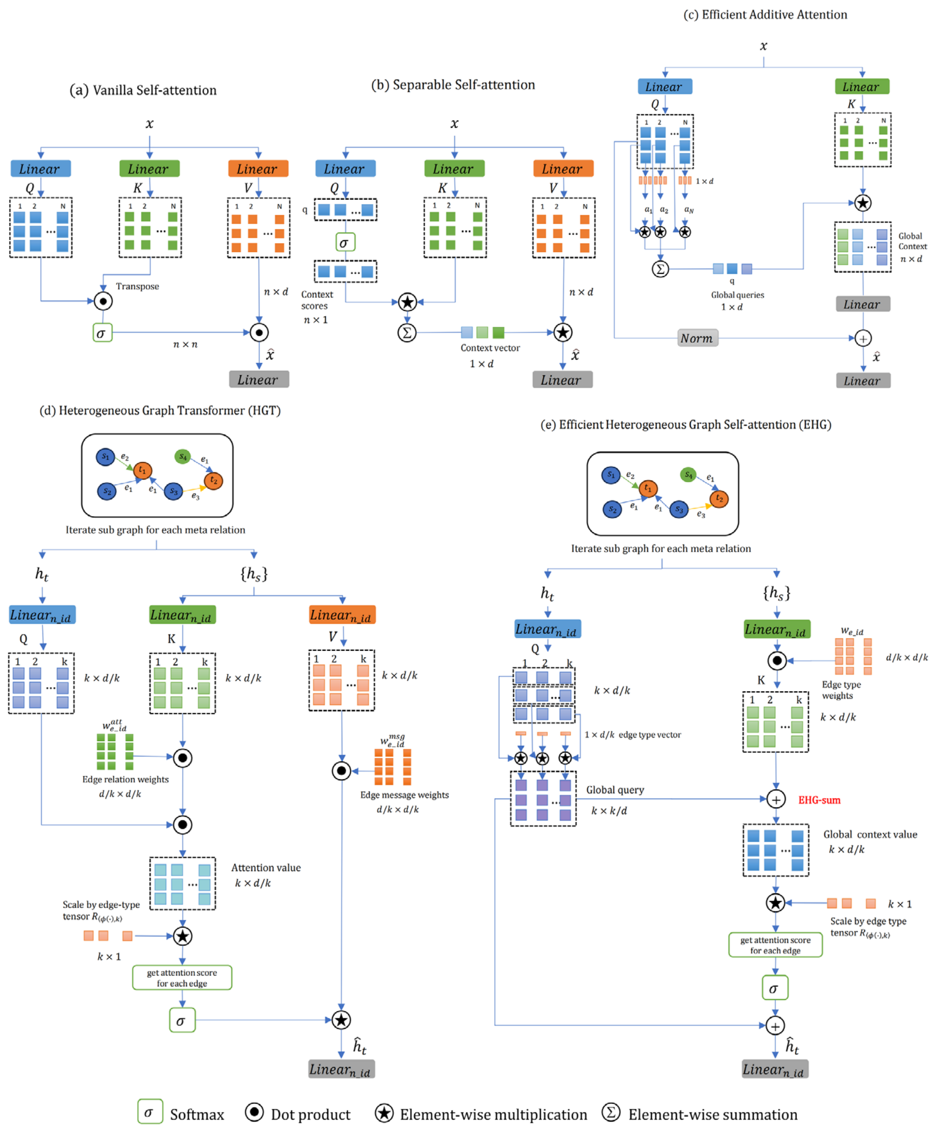

Figure 3 Comparison with different transformer models. (a) is the typical transformer self-attention module. (b) is the separable self-attention that uses element-wise operations to compute the context vector from the interactions of Q and K matrices. The context vector is then multiplied by the V matrix to produce the final output. (c) is efficient additive self-attention where the query matrix is multiplied by learnable weights and pooled to produce global queries. Then, the matrix K is element-wise multiplied by the broadcasted global queries to generate the global context representation. (d) The HGT self-attention module projects the d-dimension node features into k-head Q, K, and V matrices. The K matrix is then multiplied by edge-type-specific tensor weights and then dot product with the Q matrix to compute attention scores for each node pair. After applying Softmax normalization, these scores are used to weight the contributions of each edge pointing to the target node t. Finally, the V matrices of source nodes are aggregated to compute the output representation for node t. (e) The proposed EHG self-attention mechanism operates differently. The query matrix is first multiplied by learnable edge-type weights $q_{\phi(\cdot)}$ and pooled to generate a global query matrix. The matrix K is then element-wise calculated with the broadcasted global query matrix to produce a global context matrix. Messages from source nodes are integrated into this global context matrix, which is subsequently added directly to the query matrix to construct the representation of the target node t

Heterogeneous Graph Attention Networks (HAN) [10] design hierarchical attentions to aggregate neighbor information via different metapaths.

Table 2 Summary of the datasets used in our experiment

Dataset	DBLP	ACM	OGBN-MAG	OAG-Venue
#nodes	26,128	30,003	1,939,743	1,116,162
*edges	239,566	160,686	21,111,007	13,985,692
edge types	6	6	8	15
node types	4	3	4	5
target type	#author	#paper	#paper	#paper
classes	4	14	349	3506

Heterogeneous Graph Transformer (HGT) [5] introduces a node-type and edge-type dependent attention mechanism. Its transformer convolution architecture can incorporate information from high-order neighbors of different types through message passing across layers.

5.3 Experiment settings and results

We used a single NVIDIA RTX 3090 GPU with 24 GB of VRAM for training on the DBLP and ACM datasets, a single NVIDIA A800 GPU with 80 GB of VRAM for training on the OGBN-MAG and OAG-Venue datasets. The ratio of the training, validation, and test sets was set to 6:2:2. For ACM and DBLP, the input dimension was set to 256, with 16 attention heads and 4 layers. Due to memory constraints per epoch for larger graphs, the input dimension for OGBN-MAG and OAG-Venue were set to 128, with 4 attention heads and 2 layers. The input features we used were not pretrained but rather randomly generated vectors, which theoretically should better represent the actual performance learned by the model itself. For parameter setting, in order to compare performance fairly among these models, we set all the hyperparameters with the same learning rate, steps, and Adam optimizer.

5.4 Performance and time-consumption analysis

The performance of our approaches and baselines are shown in Table 3 and Table 4. In Table 3, we observed Accuracy, Macro-F1, Micro-F1, NDCG, and MRR metrics. Table 4 provides time-consumption metrics, such as model size, inference latency on GPU (in milliseconds), inference latency on CPU (in milliseconds), and training time (in seconds) across datasets of varying sizes. Fig. 4 shows the accuracy and NDCG scores relative to the average time consumption per training epoch for these models. All model parameter sizes on different datasets are computed under consistent configurations of input-feature dimensions, network layers, and attention head settings. The results for our model variant are highlighted in bold. Given the high variance often observed in graph-structured data, we repeat the experiments 10 times and report the average values of the metrics. Based on these results, we draw the following conclusions:

• EHG demonstrates superior overall effectiveness across all baselines and datasets, regardless of graph size. It also effectively balances time constraints with model performance, achieving an optimal tradeoff in terms of cost-effectiveness. Compared to HGT, EHG reduces the number of parameters by approximately 25% and decreases training time by around 20%, without compromising the accuracy of model rankings. These improvements are largely attributed to the removal of the value matrix (V) and the efficiency gained from performing element-wise operations instead of dot products.

• While EHG-dot slightly outperforms EHG-sum in terms of accuracy, its training time is longer due to the greater computational cost of the dot-product operation compared

Table 3 Performance on four datasets of varying sizes

Dataset	DBLP			ACM					OGBN-MAG		OAG-Venue	
	Accuracy	Macro-F1	Micro-F1	NDCG@5	MRR	Accuracy	Macro-F1	Micro-F1	NDCG@15	MRR	NDCG@100	MRR
MLP	0.2954	0.2058	0.2954	0.2392	0.2871	0.1277	0.0281	0.1277	0.0813	0.0632	0.2594	0.1235
GAT	0.8674	0.8579	0.8674	0.3814	0.3421	0.1367	0.0302	0.1367	0.1514	0.1139	OOM	OOM
RGCN	0.9212	0.9148	0.9212	0.6525	0.5933	0.3878	0.3419	0.3878	0.152	0.1134	OOM	OOM
HAN	0.8855	0.8777	0.8855	0.7339	0.6718	0.4788	0.4045	0.4788	0.1577	0.1161	0.6376	0.6011
HGT	0.9217	0.9163	0.9217	0.733	0.6728	0.477	0.4435	0.477	0.3152	0.2124	0.6856	0.6306
EHG-sum	**0.9404**	**0.935**	**0.9404**	**0.7465**	**0.6848**	**0.492**	**0.4658**	**0.492**	**0.3679**	**0.2665**	0.6816	0.6298
EHG-mul	0.9251	0.9178	0.9251	0.7288	0.6654	0.4799	0.4557	0.4799	**0.3682**	**0.267**	0.6792	0.6268
EHG-dot	**0.9432**	**0.9391**	**0.9432**	**0.7441**	**0.6816**	**0.4935**	**0.4651**	**0.4935**	0.3572	0.2559	**0.6861**	**0.6332**

Table 4 Time Consumption on four datasets of varying sizes

Model	Params (M)	Training Time(s)	CPU Latency (ms)	GPU Latency (ms)
(a) DBLP Time Consumption				
MLP	0.07	0.39	0.22	0.52
GAT	0.07	2.49	3.6	4.03
RGCN	0.10	3.3	4.36	5.03
HAN	0.40	28.65	29.87	41.98
HGT	1.23	30.39	29.2	30.71
EHG-sum	0.91	20.46	26.78	27.64
EHG-mul	0.91	21.77	27.69	29.17
EHG-dot	0.91	24.09	28.98	29.14
(b) ACM Time Consumption				
MLP	0.20	0.49	0.18	0.57
GAT	2.38	12.93	21.22	22.09
RGCN	0.42	3.79	4.26	6.45
HAN	3.23	169.84	37.56	265.56
HGT	3.56	30.27	29.11	30.25
EHG-sum	2.67	22.82	33.45	32.57
EHG-mul	2.67	23.57	29.54	28.82
EHG-dot	2.67	24.18	27.32	29.71
(c) OGBN-MAG Time Consumption				
MLP	0.11	32.13	465.96	42.99
GAT	0.25	1063.18	20,549.68	1244.66
RGCN	0.24	89.89	3052.63	178.97
HAN	0.31	184.49	9650.65	164.86
HGT	0.71	1615.79	10,100.99	1776.06
EHG-sum	0.54	1396.21	8672.86	1801.28
EHG-mul	0.54	1403.19	8600.81	1797.48
EHG-dot	0.54	1439.01	8662.91	1755.44
(d) OAG-Venue Time Consumption				
MLP	0.53	86.34	1158.65	37.96
GAT	OOM	OOM	OOM	OOM
RGCN	OOM	OOM	OOM	OOM
HAN	1.93	170.5	7532.97	256.76
HGT	1.44	739.6	10,609.21	1209.67
EHG-sum	1.16	606.97	10,512.85	1177.86
EHG-mul	1.16	615.57	10,474.49	1171.3
EHG-dot	1.16	615.84	10,440.39	1142.37

to the sum operation. However, no distinct improvement is observed in GPU inference time or CPU inference time across the different EHG models. This phenomenon can be attributed to the fact that GPU inference latency is heavily contingent upon the architecture of the hardware platform. The high degree of parallelism inherent in GPU matrix operations mitigates any significant variation in inference time.

• GAT, RGCN, and HAN can become inefficient when handling large datasets, often encountering out-of-memory (OOM) issues and excessively long training times. This inefficiency arises primarily from the complexity of large graphs and the need to maintain separate attention vectors for each edge at every layer. As the number of layers, edges, or edge types increases, memory consumption grows significantly, resulting in OOM errors during training.

• The performance and training time of HAN are strongly influenced by the selection of metapaths. In the context of complex graphs, the disconnected nature of the graph limits the availability of metapaths, resulting in reduced parameter efficiency and overall performance on large datasets. In our experiment, only one metapath is available for training on the OAG-Venue and OGBN-MAG datasets, which leads to insufficient training.

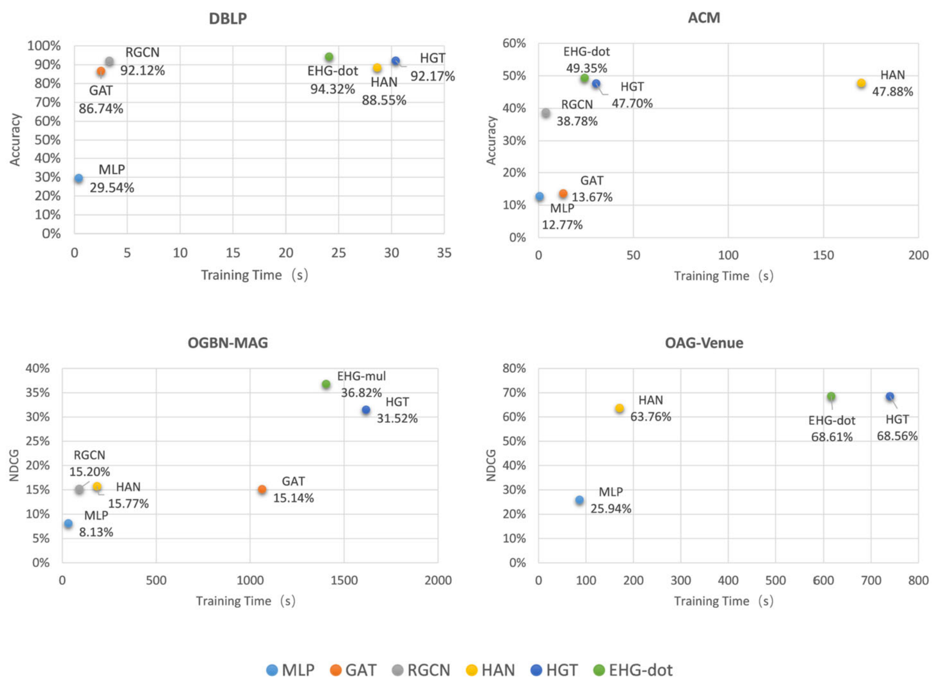

Figure 4 Performance and training time of different methods on small and large datasets

• Performance differences among these baselines are minimal on smaller datasets, however, models with more complex architectures, such as EHG, HGT, and HAN, exhibit a distinct advantage over GAT and RGCN when applied to larger datasets.

In conclusion, EHG computation complexity is lower, its element-wise operations directly manipulate the features within each dimension of the nodes, allowing the model to capture more localized and contextual information. In contrast, dot-product attention tends to emphasize the overall relationships. Depending on the requirement to accentuate either the global graph structure or localized details, the dot-product approach or element-wise operations may be more suitable, respectively. EHG integrates both local and global information, enhancing its overall performance. With an escalation in the number of classes and the expansion of graph scale, there is a discernible decrease in NDCG and MRR metrics, potentially due to oversmoothing—a well-known challenge in graph neural networks. This downward trend could also be related to the heterogeneous graph dysconnectivity in the OGBN-MAG and OAG-Venue datasets, which constrains the available metapath options in HAN. However, EHG retains significant merit in revealing the importance of metapaths and being applied to large-scale heterogeneous graphs.

5.5 Attention visualization and interpretability analysis

In EHG, the variation in weights $\boldsymbol{W}_{\boldsymbol{\phi(e)}}^{\boldsymbol{ATT}}$ across different attention heads within each layer provides insight into the influence exerted by diverse edge types. Figure 5 shows the heat map of the average attention matrix for each edge type in the final layer. We observe a highly discrete distribution rather than clustering, which indicates that our model demonstrates strong generalization and effectively captures global contextual information.

Furthermore, scaling tensors $R_{\langle\phi(e),k\rangle}$ are automatically learned, revealing which metarelations are important. By visualizing the average score of $R_{\langle\phi(e),k\rangle}$ in each layer, we can infer

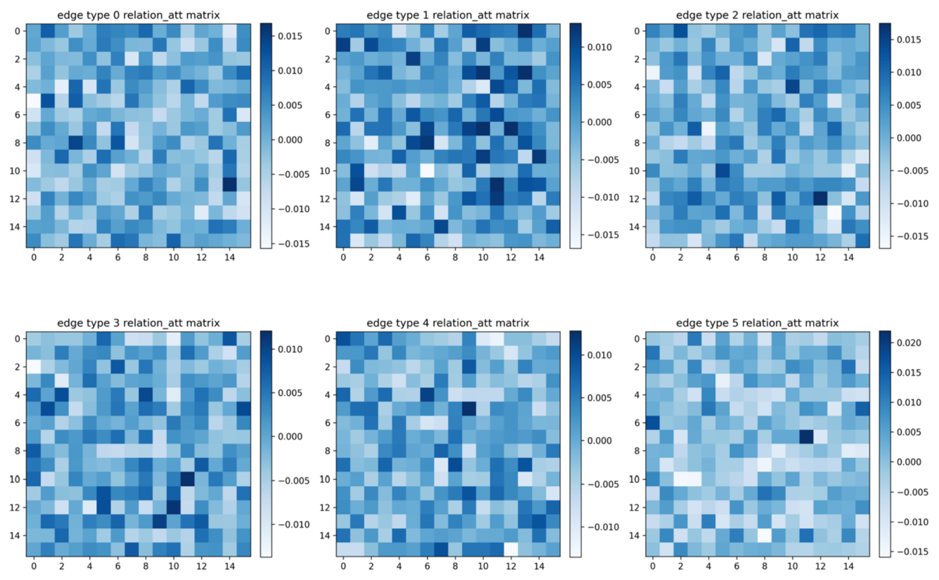

Figure 5 Averaging attention heat map for each edge type in ACM

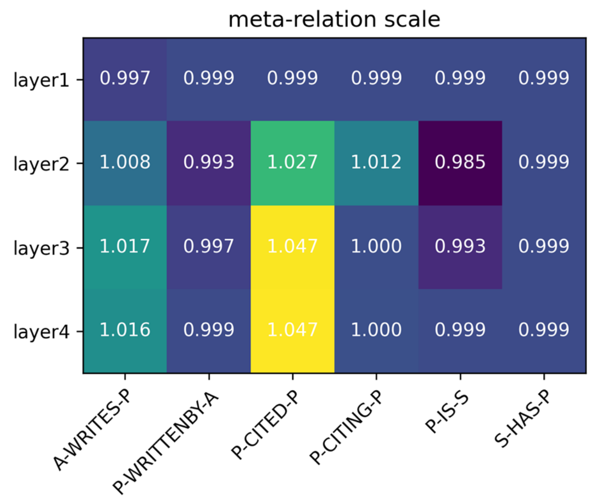

Figure 6 Metarelation score visualization across different layers

that EHG tends to implicitly learn to construct important metapaths. For example, in the ACM dataset, the 'Papers-Cited-Papers' and 'Author-Writes-Papers' paths have a greater impact on the downstream node-classification task, as shown in Fig. 6.

6 Discussion and conclusion

In this study, we present EHG, an efficient heterogeneous graph neural network model. It reduces training time while maintaining robust multiclass classification performance. This improvement is primarily due to a nearly 25% reduction in parameter size, resulting from the simplification of the V-linear transformation and message passing. During the computation of query–key interactions in each attention head, element-wise operations show a slight efficiency enhancement over the dot product, enabling the EHG model to capture both local and global contextual relationships effectively.

Further solidifying the capabilities of the EHG model, we conducted a thorough analysis and visualization of the attention matrices corresponding to various edge types and the scaling scores associated with metarelations. The experiment results show that EHG is capable of revealing the diversity of node and edge types within heterogeneous graphs. It adeptly establishes implicit metarelations replete with semantic information, and achieves strong generalization. Importantly, the EHG model stands out in its ability to generate node embeddings for vast knowledge graphs, including those in the biomedical domain [27]—biomedical graphs being quintessential heterogeneous graphs [19]. The insights gleaned from EHG regarding node classification and predictions are expected to significantly deepen our comprehension of intricate biological systems. Additionally, our findings indicate that EHG is more effective in connected graphs as opposed to disconnected ones, a testament to the pivotal role connectivity plays in distilling meaningful semantic insights. These observations will guide our future research direction.

Acknowledgements
Not applicable.

Author contributions
Man Wang contributed to the conception and implementation of the study and wrote the manuscript. Shouqiang Liu contributed to reviewing and editing. Zhen Deng worked on reviewing and guiding. All of the authors read the manuscript and approved the final manuscript.

Funding
This work is supported by the Basic and Applied Basic Research Foundation of Guangdong Province, China (2022A1515012525); Basic and Applied Basic Research Foundation of Guangdong Province, China (2021A1515011027).

Data availability
The datasets used and analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 9 October 2024 Accepted: 12 January 2025 Published online: 04 February 2025

References
1. Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., Leskovec, J.: Open graph benchmark: datasets for machine learning on graphs. In: Advances in Neural Information Processing Systems, pp. 22118–22133. Curran Associates, Red Hook (2020)
2. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710. Assoc. Comput. Mach., New York (2014)
3. Dong, Y., Chawla, N.V., Swami, A.: metapath2vec: scalable representation learning for heterogeneous networks. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 135–144. Assoc. Comput. Mach., New York (2017)
4. Sun, Y., Han, J., Yan, X., Yu, P.S., Wu, T.: PathSim: meta path-based top-K similarity search in heterogeneous information networks. Proc. VLDB Endow. **4**, 992–1003 (2011). https://doi.org/10.14778/3402707.3402736
5. Shang, J., Qu, M., Liu, J., Kaplan, L.M., Han, J., Peng, J.: Meta-Path Guided Embedding for Similarity Search in Large-Scale Heterogeneous Information Networks. CoRR. (2016). arXiv:1610.09769
6. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in Neural Information Processing Systems. Curran Associates, Red Hook (2016)
7. Zhang, Z., Cui, P., Zhu, W.: Deep learning on graphs: a survey. IEEE Trans. Knowl. Data Eng. **34**, 249–270 (2022). https://doi.org/10.1109/TKDE.2020.2981333
8. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings (2017). OpenReview.net
9. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph Attention Networks (2018). http://arxiv.org/abs/1710.10903

10. Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., Yu, P.S.: Heterogeneous graph attention network. In: The World Wide Web Conference, pp. 2022–2032. Assoc. Comput. Mach., New York (2019)
11. Zhang, C., Song, D., Huang, C., Swami, A., Chawla, N.V.: Heterogeneous graph neural network. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 793–803. Assoc. Comput. Mach., New York (2019)
12. Hu, Z., Dong, Y., Wang, K., Sun, Y.: Heterogeneous graph transformer. In: Proceedings of the Web Conference 2020, pp. 2704–2710. Assoc. Comput. Mach., New York (2020)
13. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 1025–1035. Curran Associates, Red Hook (2017)
14. Chen, J., Ma, T., Xiao, C.: FastGCN: fast learning with graph convolutional networks via importance sampling. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings (2018). OpenReview.net
15. Wang, S., Li, B.Z., Khabsa, M., Fang, H., Ma, H.: Linformer: Self-Attention with Linear Complexity. CoRR. (2020). arXiv:2006.04768
16. Mehta, S., Rastegari, M.: Separable Self-attention for Mobile Vision Transformers (2022). http://arxiv.org/abs/2206.02680
17. Shaker, A., Maaz, M., Rasheed, H., Khan, S., Yang, M.-H., Khan, F.S.: SwiftFormer: Efficient Additive Attention for Transformer-based Real-time Mobile Vision Applications (2023). http://arxiv.org/abs/2303.15446
18. Wu, Q., Yang, C., Zhao, W., He, Y., Wipf, D., Yan, J.: DIFFormer: Scalable (Graph) Transformers Induced by Energy Constrained Diffusion (2023). http://arxiv.org/abs/2301.09474
19. Yao, J., Sun, W., Jian, Z., Wu, Q., Wang, X.: Effective knowledge graph embeddings based on multidirectional semantics relations for polypharmacy side effects prediction. Bioinformatics **38**, 2315–2322 (2022). https://doi.org/10.1093/bioinformatics/btac094
20. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of the 30th International Conference on Machine Learning, vol. 28, pp. 3 (2013). References - Scientific Research Publishing. https://www.scirp.org/reference/referencespapers?referenceid=2747334
21. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems. Curran Associates, Red Hook (2017)
22. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, pp. 770–778. IEEE Comput. Soc., New York (2016)
23. Hendrycks, D., Gimpel, K.: Gaussian Error Linear Units (GELUs) (2016). https://ui.adsabs.harvard.edu/abs/2016arXiv160608415H
24. Dwivedi, V.P., Bresson, X.: A Generalization of Transformer Networks to Graphs (2021). http://arxiv.org/abs/2012.09699
25. Hu, J., Hooi, B., He, B.: Efficient heterogeneous graph learning via random projection. IEEE Trans. Knowl. Data Eng. **36**(12), 8093–8107 (2024). https://doi.org/10.1109/TKDE.2024.3434956
26. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Gangemi, A., Navigli, R., Vidal, M.-E., Hitzler, P., Troncy, R., Hollink, L., Tordai, A., Alam, M. (eds.) The Semantic Web, pp. 593–607. Springer, Cham (2018)
27. Su, X., Hu, P., You, Z.-H., Yu, P.S., Hu, L.: Dual-channel learning framework for drug-drug interaction prediction via relation-aware heterogeneous graph transformer. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, pp. 249–256 (2024). https://doi.org/10.1609/aaai.v38i1.27777

Publisher's Note