HEGformer: Representation Learning on Heterogeneous Graph Neural Network with Efficient Transformer

Tianzuo Qin
Department of Computer Science
The University of Hong Kong
Hong Kong, China
u3009723@connect.hku.hk

Junwei Su
Department of Computer Science
The University of Hong Kong
Hong Kong, China
junweisu@connect.hku.hk

Chuan Wu
Department of Computer Science
The University of Hong Kong
Hong Kong, China.
cwu@cs.hku.hk

***Abstract*—Heterogeneous Graph Neural Networks (HGNNs) have shown remarkable success in learning from real-world graph data by capturing complex heterogeneous characteristics like various semantic and relational information across different node and edge types.**

However, as graphs grow in size and complexity, existing models struggle with scalability. Current graph transformers are often constrained by sampling algorithms, limiting their receptive fields and preventing them from fully leveraging the transformer's capability. Simply expanding the receptive field (i.e., long-range dependencies) leads to quadratic computational costs, resulting in time and memory constraints. In addition, although some transformer architectures are trying to overcome the quadratic computational complexity, few of them have considered the relational graph structure or heterogeneity.

To address these challenges, we proposed HEGformer, an innovative integration of Graph Transformers and Heterogeneous Graphs. We propose two techniques: Relational Local Sensitive Hashing (R-LSH) and Relational-aware Enhanced Attention (REA). R-LSH extends the traditional Local Sensitive Hashing approach by incorporating relational information between different node types, significantly reducing the quadratic computational cost commonly associated with vanilla Graph Transformers while maintaining relational integrity. REA leverages node-type information to create global relational contexts, enhancing attention mechanisms to be more type-aware and computationally efficient. Our methods demonstrate improved scalability and competitive performance on graph-based tasks of various sizes, offering a powerful model for future research in large-scale, complex graph analytics.

***Index Terms*—Graph Transformer, Heterogeneous Graph Neural Networks, Graph Representation Learning**

I. Introduction

Heterogeneous Graph Neural Networks (HGNNs) [2], [3], [5], [6], [48], [49] address the complexity of diverse node and edge types, reflecting intricate semantic relationships found in real-world scenarios. A heterogeneous graph, also known as a heterogeneous information network, is a graph that contains different types of nodes and edges, each type representing various entities and their interactions. This is in contrast to homogeneous graphs, where all nodes and edges are of the same type. Heterogeneous graphs are important because they more accurately represent complex systems, such as social networks, biological networks, and knowledge graphs, where multiple types of entities and relationships coexist. HGNN models emphasize heterogeneous graph representation learning, offering a promising framework for embedding nodes in lower-dimensional spaces while preserving their rich structural attributes. By capturing the diversity of node and edge types, HGNNs can model the complex and multi-faceted relationships present in real-world data. However, HGNNs face challenges in maintaining the information of heterogeneous nodes and edges while ensuring efficient computation. This balance is crucial for leveraging the full potential of HGNNs in applications such as recommendation systems, fraud detection, and drug discovery.

Traditional message-passing GNNs, in both homogeneous and heterogeneous scenario, suffer from over-smoothing [32], [33], where repeated aggregation of node features leads to indistinguishable node representations, and over-squashing [34], where distant node information is inadequately propagated. These issues are particularly pronounced in heterogeneous graphs, where the diversity of node and edge types adds another layer of complexity. The introduction of the Transformer architecture into the graph domain could address these issues by leveraging its global attention mechanism to better capture intricate relationships and interactions between different types of nodes and edges [30].

There are many works done in the Graph Transformer area, but most of them focus on homogeneous cases rather than heterogeneous ones. Hence, the power of the Graph Transformer in heterogeneity still has great potential, especially when scaling to large graphs [17]. The computational and storage demands, inherently quadratic with respect to the number of nodes, pose significant barriers to practical applications [2], [13], [17]. Efforts to introduce heterogeneous graph structures within Transformers have made some progress but have not yet fully capitalized on the unique characteristics of graph data, often resulting in sub-optimal outcomes. For example,

This work was supported in part by grants from Hong Kong RGC under the contracts HKU 17207621, 17203522 and C7004-22G (CRF).

the Heterogeneous Graph Transformer (HGT) [3] is often regarded as a transformer-based model for HGNNs, but its core mechanism primarily relies on message-passing. The performance gains attributed to HGT are largely due to its sampling method, HGSampling, rather than a comprehensive utilization of the capabilities of the transformer architecture.

By combining HGNNs with Transformer models, we aim to leverage the strengths of both approaches. HGNNs provide a robust framework for handling the diversity of node and edge types, while Transformers offer powerful global attention mechanisms to effectively capture complex relational information. Recent advancements [2], [3], [21], [39]–[41] in Graph Transformer models suggest a potential evolution from traditional message-passing architectures, showing impressive results across various graph tasks. Nevertheless, the interplay between the Transformer's complexities and the demands of managing large-scale, heterogeneity-rich HGNNs remains a significant challenge. This integrated approach holds the potential to significantly advance the field of graph analytics by addressing scalability and performance issues in heterogeneous graph settings. There are two major challenges need to be overcome to realize this potential.

Firstly, deploying Graph Transformers in heterogeneous graphs presents significant computational and scalability challenges. The diverse node and edge types in these graphs amplify the complexity of capturing long-range dependencies and intricate relational patterns. Traditional attention mechanisms in Graph Transformers, despite their power, suffer from quadratic time complexity relative to the number of nodes, making them impractical for large-scale heterogeneous graphs. Some recent works, such as HINormer [2], claim to address this issue, but they often rely on sampling methods to constrain the receptive field to a constant size, which can limit their ability to fully capture the rich and diverse relationships in heterogeneous graphs. Hence, if we want to expand the "vision" of the transformer, we need try to reduce the quadratic computation overhead. Local Sensitive Hashing (LSH), known for its efficiency in approximating nearest-neighbor searches, which might provide a potential solution to this complexity. However, applying LSH within Graph Transformers is challenging. The primary concern is ensuring the hashing mechanism preserves the rich relational information inherent in heterogeneous graphs while reducing computational overhead. Specifically, the challenge lies in designing LSH functions that can effectively hash nodes with diverse and complex feature representations without losing crucial information about node types and inter-node relationships. Addressing this issue is essential to fully harness the potential of LSH in improving the scalability and efficiency of heterogeneous Graph Transformers.

Secondly, while Graph Transformers is skilled in modeling sequential data, their application to heterogeneous graphs faces challenges, particularly in maintaining and enhancing relational integrity across diverse node types. The conventional attention mechanism tends to oversimplify the complex relationships in heterogeneous graphs, often diluting critical relational information. To address these limitations, we propose Relational-aware Enhanced Attention (REA). REA aims to enrich the attention mechanism by incorporating node-type-specific contexts, preserving and amplifying relational integrity within heterogeneous graphs. The challenge lies in designing attention heads that are not only aware of the node types but also capable of dynamically adjusting their focus based on type-specific relational contexts. This requires an approach that ensures the enhanced attention mechanism does not introduce prohibitive computational complexity while significantly improving the model's performance in capturing rich, type-aware relational patterns inherent in heterogeneous graphs.

To tackle these two challenges, we propose HEGformer, an efficient Graph Transformer that utilizes Local Sensitive Hashing (LSH) and Relational-aware Enhanced Attention (REA) to improve the performance of GNNs in heterogeneous settings. Specifically, we use LSH to approximate the nearest "semantic" neighbor searches efficiently by hashing nodes into buckets based on their feature representations. Nodes that are hashed into the same bucket are considered neighbors, which significantly reduces the computational complexity associated with traditional attention mechanisms. By leveraging LSH, we manage large-scale heterogeneous graphs more effectively, ensuring that similar semantic neighbors are processed together, thereby maintaining proximity and relational integrity essential for effective graph analytics.

REA aims to enhance the attention mechanism in Graph Transformers by incorporating node-type-specific contexts. Unlike conventional attention mechanisms, which may oversimplify complex relationships in heterogeneous graphs, REA dynamically adjusts its focus based on the type-specific relational contexts of nodes. This approach preserves and amplifies the intricate interactions between different node types, ensuring that the model captures the rich relational information crucial for understanding and leveraging heterogeneous graphs.

In summary, our work addresses the significant challenges posed by the scalability and computational complexity of Graph Neural Networks (GNNs) when applied to heterogeneous graphs. By introducing Local Sensitive Hashing, we significantly reduce the computational overhead of attention mechanisms, enabling efficient handling of large-scale heterogeneous graph data. Additionally, our Relational-aware Enhanced Attention mechanism ensures the preservation and amplification of complex relational information by incorporating node-type-specific contexts into the attention process. For the validation of our design, we perform substantial experiments on several heterogeneous GNN benchmark datasets, demonstrating the performance and efficiency of HEGformer compared to other models.

II. Related Works

A. Heteogeneous Graph Neural Networks

Heterogeneous Graph Neural Networks (HGNNs) have emerged as a powerful extension of Graph Neural Networks

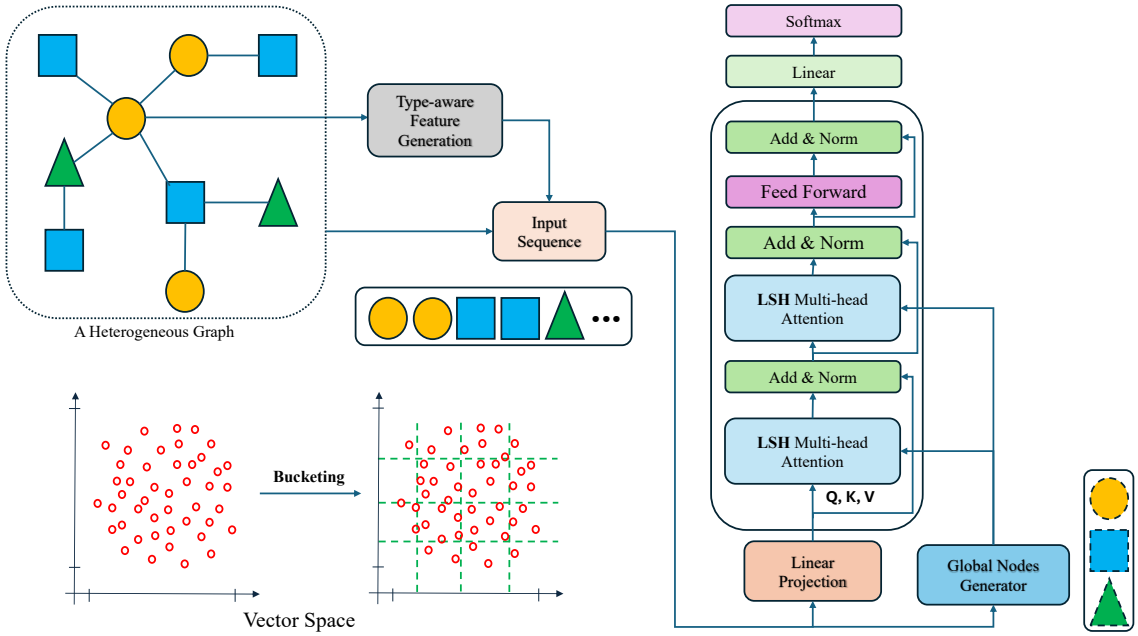

Fig. 1: Overview of HEGformer

(GNNs) to handle graphs with diverse types of nodes and edges. Traditional GNNs, such as Graph Convolutional Networks (GCNs) [42] and Graph Attention Networks (GATs) [43], primarily focus on homogeneous graphs where all nodes and edges are of a single type. However, many real-world graphs, such as bibliographic networks, social networks, and knowledge graphs [44]–[46], are inherently heterogeneous. This necessitates the development of specialized models capable of leveraging the rich semantic information embedded in such structures.

A heterogeneous graph (or network) $\mathcal{G}$ is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R})$, where $\mathcal{V}$ represents the set of nodes, $\mathcal{E}$ represents the set of edges, $\mathcal{T}$ represents the set of node types, and $\mathcal{R}$ represents the set of relation types (or edge types). Unlike homogeneous graphs, where each node and edge are of the same type, in a heterogeneous graph, nodes and edges can belong to multiple types.

Heterogeneous Graph Neural Networks (HGNNs) extend traditional GNNs to handle heterogeneous graphs. The primary challenge for HGNNs is to encode information from various types of nodes and edges while maintaining the graph's structural and semantic integrity.

Formally, given a node v in $\mathcal{G}$, the objective of an HGNN is to learn a function $f : \mathcal{V} \rightarrow \mathbb{R}^d$, which maps each node to a d-dimensional vector by aggregating information from its neighbors and considering node and relation types. The aggregation function is typically heterogeneity-aware, taking into account both the structural information and type information.

Many early works are focusing on HGNN [2], [3], [6], [21], [48]. For example, Heterogeneous Graph Attention Networks (HAN) [6], extends the attention mechanism to heterogeneous graphs by introducing node-level and semantic-level attention layers. HAN effectively captures the importance of different types of neighbors and meta-paths, providing a significant improvement in performance over traditional GNNs. Heterogeneous Graph Transformer (HGT) [3] incorporates type-specific parameters for nodes and edges, enabling the model to learn the complex interactions among various types of nodes and edges. The use of multi-head attention in HGT further enhances its ability to capture diverse relational patterns within the graph.

Despite their advancements, these models often struggle with scalability issues when applied to large-scale heterogeneous graphs. The computational cost of attention mechanisms in HGNNs scales quadratically with the number of nodes, posing significant challenges for real-world applications.

B. Graph Transformer

The Graph Transformer is a powerful model that extends the Transformer architecture to graph-structured data. It leverages the self-attention mechanism to capture the complex dependencies between nodes in a graph. Unlike traditional graph neural networks, which rely on fixed aggregation functions, the Graph Transformer dynamically computes the importance of neighboring nodes, allowing it to adaptively learn the relationships in the graph.

The core component of the Graph Transformer is the self-attention mechanism, which computes attention scores between nodes to capture their dependencies. Given a set of node feature vectors $\{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_N\}$, the self-attention mechanism first projects these features into three different spaces: queries ($\mathbf{Q}$), keys ($\mathbf{K}$), and values ($\mathbf{V}$). This is achieved using learned weight matrices $\mathbf{W}_Q$, $\mathbf{W}_K$, and $\mathbf{W}_V$:

$$\mathbf{Q} = \mathbf{H}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{H}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{H}\mathbf{W}_V, \tag{1}$$

where $\mathbf{H} \in \mathbb{R}^{N \times d}$ is the matrix of input node features, and $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d_k}$ are the weight matrices.

The attention scores between nodes are computed as the scaled dot product of the queries and keys:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}, \tag{2}$$

where d_k is the dimensionality of the queries and keys. The softmax function ensures that the attention scores sum to one.

The multiplication of the query (Q) and key (K) matrices results in a computational complexity of $O(N^2)$, where N is the number of nodes in the graph. This quadratic complexity poses significant challenges when scaling graph transformers to large datasets, particularly in heterogeneous graphs where the diversity of node and edge types adds another layer of complexity.

Current approaches for applying graph transformers to heterogeneous graphs have not yet proposed effective methods for making these models work efficiently on large datasets. Most existing methods [2], [3] either rely heavily on sampling techniques or introduce additional preprocessing steps that limit their scalability. These approaches often fail to fully leverage the potential of the transformer architecture, resulting in unsatisfactory efficiency when dealing with large-scale heterogeneous graphs. Despite the advancements in graph neural networks, there remains a substantial gap in effectively scaling graph transformers to handle the rich relational information and diverse interactions present in extensive heterogeneous datasets.

III. Proposed Method: HEGformer

A. Overview of HEGformer

Fig. 1 provides an overview of the HEGformer architecture, which is designed to efficiently process heterogeneous graphs using a combination of Local Sensitive Hashing (LSH) and Relational-aware Enhanced Attention (REA). The process begins with type-aware feature generation, where each node in the heterogeneous graph is encoded with its type and relational information from its neighbors. These features are then organized into an input sequence for the Transformer model. Within the Transformer layers, LSH approximates nearest neighbor searches by hashing nodes into buckets based on their aggregated features, which include their own features and structural information from their neighbors. This bucketing process ensures that similar nodes are grouped together, reducing the computational complexity of the attention mechanism. A Global Nodes Generator component aggregates information across different types, maintaining relational integrity and enhancing the attention mechanism.

Algorithm 1 Relational Local Sensitive Hashing (R-LSH)

Input: Graph $G = (V, E)$ with node features X, number of hash functions k, hash function family $\mathcal{H}$
Output: Buckets $\mathcal{B}$ containing grouped nodes

Initialize an empty list of buckets $\mathcal{B}$
for *each node* $v \in V$ **do**
  Step 1: Compute Aggregated Features
  Aggregate features of node v and its neighbors:
  $x'_v = \text{Aggregate}(\{x_u : u \in \mathcal{N}(v)\} \cup \{x_v\})$
  Step 2: Compute Hash Key
  Compute hash key $\mathbf{h}(v) = (h_1(x'_v), h_2(x'_v), \ldots, h_k(x'_v))$, where $h_i \in \mathcal{H}$
  Step 3: Assign to Buckets
  if *bucket* $B_{\mathbf{h}(v)}$ *does not exist* **then**
    Create bucket $B_{\mathbf{h}(v)}$
  end
  Assign node v to bucket $B_{\mathbf{h}(v)}$
end
return Buckets $\mathcal{B}$ containing grouped nodes

B. Local Sensitive Hashing

Local Sensitive Hashing (LSH) is a technique used to approximate nearest neighbor search in high-dimensional spaces by hashing input items into buckets such that similar items are more likely to be in the same bucket [1]. The primary idea behind LSH is to use a family of hash functions to map similar data points to the same buckets with high probability, thereby reducing the search space for nearest neighbors and significantly lowering computational complexity compared to exhaustive searches.

In the context of Graph Neural Networks (GNNs), LSH can enhance the efficiency of the attention mechanism, which traditionally suffers from quadratic time complexity relative to the number of nodes, making it impractical for large-scale graphs. By employing LSH, we limit attention calculations to a subset of nodes that are likely to be similar, referred to as "semantic neighbors." This method ensures that nodes with similar features and structural contexts are processed together, thereby reducing computational complexity while preserving the quality of the attention mechanism. Empirical evidence supports the effectiveness of LSH in reducing computational complexity while maintaining performance. For example, Sketch-GNN [35] demonstrated that hashing could improve the quality of selecting similiar neighbors used for efficient convolution-based GNN training, showing scalability and competitive performance on large-graph benchmarks, which illustrates how hash algorithm can be applied to reduce computational overhead while maintaining high performance in graph-based tasks.

Hence, the use of LSH in this paper is motivated by the need to address the scalability issues associated with applying Transformer models to large and heterogeneous graphs. Heterogeneous graphs, with their diverse node and edge types, amplify the complexity of capturing long-range dependencies and intricate relational patterns. Integrating LSH allows for

more effective management of large-scale graphs by ensuring that similar nodes are processed together, thus maintaining proximity and relational integrity essential for effective graph analytics. In our method, we call it R-LSH.

After R-LSH operation, nodes are assigned to buckets based on their hash signatures. By integrating type-aware node features, the R-LSH method can better preserve the semantic relationships in heterogeneous graphs, ensuring that similar and related nodes are more likely to be hashed into the same bucket. The details of R-LSH is shown in Algorithm 1.

R-LSH can reduce the complexity of the original attention computation from $O(N^2)$ to $O(N \log N)$. Within each bucket, the algorithm computes a global node representation by aggregating the features of all nodes in the bucket. This global node provides a type-specific context that complements the local interactions within the bucket. For each node in the bucket, the attention scores are calculated for its neighbors and the global node. The node's feature representation is then updated based on these attention scores, incorporating both local and global contextual information.

By leveraging R-LSH to manage computational complexity and REA to preserve and enhance relational information, HEGformer is well-suited for large-scale heterogeneous graphs.

In summary, this process significantly reduces the computational complexity of subsequent operations, such as attention mechanisms, by focusing calculations on smaller, more relevant subsets of nodes, thereby improving efficiency and scalability in heterogeneous graph analytics.

C. Relational-aware Enhanced Attention

Building upon the foundation of Relational Local Sensitive Hashing (R-LSH), which efficiently groups similar nodes by leveraging both their features and structural context, we introduce the Relational-aware Enhanced Attention (REA) mechanism. R-LSH ensures that nodes with similar features and relational contexts are hashed into the same buckets, facilitating efficient and accurate neighbor searches. However, while R-LSH significantly reduces the computational overhead by focusing on smaller subsets of similar nodes, it does not fully address the complexity of capturing the intricate relationships and rich semantic information inherent in heterogeneous graphs.

To bridge this gap, the REA mechanism is essential as it enhances the model's ability to comprehend and utilize the diverse types of interactions within the graph. Traditional attention mechanisms often fail to capture the broader context and complex dependencies that exist between different types of nodes and edges in heterogeneous graphs. REA addresses this by integrating both local and global contextual information into the attention process. This dual integration is crucial for preserving the relational integrity and ensuring that the attention mechanism can dynamically adjust to the varying significance of different node types and relationships.

The REA mechanism addresses this need by integrating both local and global context into the attention computation. In heterogeneous graphs, nodes and edges belong to various types, each contributing unique semantic information. Traditional attention mechanisms focus primarily on pairwise interactions between nodes, often missing the broader context provided by the heterogeneous structure. REA enhances the standard attention mechanism by incorporating type-specific global nodes, which aggregate the features of selected nodes of a particular type. This dual attention mechanism ensures that the updated feature representation of each node captures both its immediate neighborhood and the broader type-specific characteristics, thus preserving the rich relational context.

Given a heterogeneous graph $G = (V, E)$ with a node set V and an edge set E, each node $v \in V$ is associated with a node type t_v from a set of node types T_v. We propose to group nodes by their types and represent each group with a global node that captures the collective characteristics of that type.

Let $V_t \subseteq V$ be the set of nodes of type t. We define a global node g_t for each node type t, which serves as the representative of its respective group.

The core of REA lies in its attention mechanism, which not only considers the pairwise attention between individual nodes but also integrates the global context provided by the type-specific global nodes.

For each global node g_t, we compute its representation as an aggregation of the features of selected nodes of the same type:

$$x_{g_t} = \text{Aggregate}(\{x_v^* : v \in V_t\}) \tag{3}$$

The attention scores are computed not only based on the pairwise interactions between nodes but also considering the global nodes. For a node v of type t_v, its updated representation x'_v is computed as:

$$x'_v = \sum_{u \in LSH(v)} \alpha_{uv} W x_u + \sum_{t \in T_v} \beta_{vt} W x_{g_t} \tag{4}$$

where:

- α_{uv} is the attention score between nodes u and v in R-LSH setting.
- β_{vt} is the relational-aware attention score between node v and global node g_t, reflecting the importance of the global context of type t to node v.

The relational-aware attention scores β_{vt} are computed as:

$$\beta_{vt} = \text{softmax}\left(\frac{(x_v W_q)(x_{g_{t_v}} W_k)^T}{\sqrt{d_k}}\right) \tag{5}$$

where W_q and W_k are learnable weight matrices for query and key transformations, respectively, and d_k is the scaling factor. Given that the number of global nodes is much smaller than the nodes in the original graph, computing β_{vt} won't harm the efficiency of the whole model.

The REA mechanism, by integrating type-specific global nodes, allows for a more comprehensive representation of each node. The global nodes act as aggregators of type-specific features, capturing the broader context that traditional

pairwise attention mechanisms often miss. This inclusion of global context is crucial for heterogeneous graphs where different node types contribute unique and significant semantic information.

The process begins with the aggregation of node features within each type group to form the global nodes. These global nodes then participate in the attention mechanism, providing a type-specific context that complements the local, pairwise interactions. This dual-level attention, combining local neighborhood information with type-specific global context, ensures that the feature representations of nodes are both detailed and contextually enriched.

By incorporating these enhancements, REA not only addresses the limitations of traditional attention mechanisms but also ensures that the computational complexity remains manageable. The integration of LSH reduces the number of nodes involved in the attention computation, while REA ensures that the nodes' feature representations are both accurate and contextually relevant. This combined approach of LSH and REA significantly advances the capability of Graph Transformers to handle large-scale heterogeneous graphs, preserving both computational efficiency and the rich relational context inherent in such graphs.

TABLE I: Dataset Overview

Name	# Nodes	# Node Types	# Edges	# Edge Types	Target	# Classes
DBLP	26,128	4	239,566	6	Author	4
IMDB	21,420	4	86,642	6	Movie	5
Freebase	43,854	4	151,034	6	Movie	3
AMiner	55,783	3	153,676	4	Paper	4

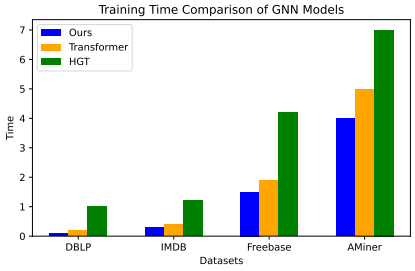

Fig. 2: Efficiency Comparison

IV. Experiment

A. Baselines

To evaluate the performance of our proposed model, we compare it against several state-of-the-art methods for heterogeneous graph neural networks. The following selected baselines are applied in our experiments:

- **RGCN** [5]: RGCN extends the Graph Convolutional Network (GCN) to handle multi-relational data by incorporating different types of relations into the model's layers. This method effectively captures the structural information present in heterogeneous graphs.
- **HGT** [3]: HGT adapts the Transformer architecture to heterogeneous graphs by introducing type-specific transformation matrices and attention mechanisms. It effectively models the diverse relationships in heterogeneous graphs through its attention-based architecture.
- **Simple-HGN** [26]: Simple-HGN simplifies the heterogeneous graph neural network by using a single GNN layer for each node type, followed by a fusion layer to combine the information from different types. This model aims to reduce complexity while maintaining high performance.

B. Environment Setup

To evaluate the performance of our proposed Heterogeneous Graph Neural Network (HGNN) with Relational Local Sensitive Hashing (Relational LSH) and Relational-aware Enhanced Attention (REA), we conducted experiments on several benchmark datasets shown in Table I.

In terms of computational resources, the experiments were executed on a state-of-the-art system with AWS EC2 g4dn.metal, providing ample power to handle the demanding computational requirements of large-scale graph processing. We pitted our proposed model against leading models in the graph neural network space, measuring performance using accuracy as the key metric for the node classification task. This benchmarking was instrumental in evaluating the proposed model's capability to accurately infer node labels while effectively capturing the complex interplay of heterogeneous relationships within each dataset. The experimental findings are expected to offer insights into the scalability and precision of graph neural networks, particularly in diverse and expansive graph-based applications. The result is shown in Table II. Note that we have utilized the baseline results reported in [26] and [2].

C. Performance Analysis

HEGformer's ability to maintain competitive performance across diverse datasets, including DBLP, IMDB, Freebase, and AMiner, highlights its robustness and adaptability. The R-LSH mechanism effectively reduces computational complexity by grouping similar nodes based on their features and structural context, enabling efficient and accurate neighbor searches. This efficiency is particularly beneficial in large-scale graph scenarios, where traditional methods may struggle with scalability. Specifically, we found that only several (less than 5) hashings provided a substantial boost in accuracy and robustness. However, further increasing the number of hashings beyond three did not yield any significant improvements and, in some cases, even led to a decrease in performance.

Furthermore, the REA mechanism enhances the attention computation by integrating both local and global contexts. By incorporating type-specific global nodes, REA ensures that the updated feature representation of each node captures both its immediate neighborhood and the broader type-specific characteristics. This dual attention mechanism preserves the rich relational context inherent in heterogeneous graphs, leading to more accurate and meaningful node embeddings.

TABLE II: Overall Performance Evaluation

Methods	DBLP		IMDB		Freebase		AMiner	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
GCN	91.47 ± 0.34	90.84 ± 0.32	64.82 ± 0.64	57.88 ± 1.18	68.34 ± 1.58	59.81 ± 3.04	85.75 ± 0.41	75.74 ± 1.10
GAT	93.39 ± 0.30	93.83 ± 0.27	64.86 ± 0.43	58.94 ± 1.35	69.04 ± 0.58	59.28 ± 2.56	84.92 ± 0.68	74.32 ± 0.95
RGCN	92.07 ± 0.50	91.52 ± 0.50	62.95 ± 0.15	58.85 ± 0.26	60.82 ± 1.23	59.08 ± 1.44	81.58 ± 1.44	62.53 ± 2.31
HetGNN	92.33 ± 0.41	91.76 ± 0.43	51.16 ± 0.65	48.25 ± 0.67	62.99 ± 2.31	58.44 ± 1.99	72.34 ± 1.42	55.42 ± 1.45
HAN	92.05 ± 0.62	91.67 ± 0.49	64.63 ± 0.58	57.74 ± 0.96	61.42 ± 3.56	57.05 ± 2.06	81.90 ± 1.51	64.67 ± 2.21
MAGNN	93.76 ± 0.45	93.28 ± 0.51	64.67 ± 1.67	56.49 ± 3.20	64.43 ± 0.73	58.18 ± 3.87	82.64 ± 1.59	68.60 ± 2.04
HGT	93.49 ± 0.25	93.01 ± 0.23	67.20 ± 0.57	63.00 ± 1.19	66.43 ± 1.88	60.03 ± 2.21	85.74 ± 1.24	74.98 ± 1.61
SimpleHGN	94.46 ± 0.22	94.01 ± 0.24	67.36 ± 0.57	63.53 ± 1.36	67.49 ± 0.97	62.49 ± 1.69	86.44 ± 0.48	75.73 ± 0.97
HEGformer	94.10 ± 0.56	93.88 ± 0.45	66.33 ± 1.76	62.88 ± 1.92	67.82 ± 1.77	61.01 ± 1.85	85.44 ± 1.23	72.95 ± 1.56

Overall, the consistent competitive performance of HEGformer across multiple datasets underscores its potential as a powerful tool for heterogeneous graph analytics. Its innovative mechanisms not only improve computational efficiency but also ensure the preservation and enhancement of relational information, making it a significant advancement in the field of Graph Neural Networks.

D. Efficiency Analysis

The efficiency of our proposed model was evaluated against HGT and Transformer on several benchmark datasets. Our model not only demonstrated the performance in terms of accuracy but also showed significant improvements in computational efficiency.

Our proposed HEGformer architecture demonstrates significantly faster computational performance compared to the Heterogeneous Graph Transformer (HGT), primarily due to the integration of Relational Local Sensitive Hashing (R-LSH) and Relational-aware Enhanced Attention (REA) mechanisms. HGT, while effective in capturing heterogeneous relationships, relies heavily on customized meta paths and extensive attention computations across all nodes and edges, resulting in high computational overhead. The attention mechanism in HGT scales quadratically with the number of nodes, making it computationally expensive and less feasible for large-scale graphs.

In contrast, HEGformer employs R-LSH to efficiently partition the graph into buckets of similar nodes, drastically reducing the number of nodes involved in the attention calculation. By focusing the attention mechanism on smaller, semantically similar subsets of nodes, HEGformer reduces the computational complexity from $O(N^2)$ to $O(N \log N)$. This reduction is achieved because LSH ensures that only nodes with a high probability of being similar are grouped together, thus limiting the attention computations to within these smaller groups.

Additionally, the REA mechanism enhances the efficiency by integrating both local and global context into the attention computation, ensuring that the attention scores are computed more effectively and accurately without the need for extensive pairwise comparisons. This dual-level attention mechanism not only preserves the rich relational context but also maintains computational efficiency.

V. Conclusion

In conclusion, the experiments conducted in this study have demonstrated that our proposed graph transformer model achieves competitive performance on various node classification tasks while significantly improving the computational efficiency of processing large-scale graph data. The efficiency test results highlight our model's faster speed across diverse datasets, a critical advantage for practical applications in graph analytics. Our model's ability to swiftly navigate the intricate web of heterogeneous relationships within graphs while maintaining high accuracy attests to its robustness and the effectiveness of the underlying architecture. These qualities make it an excellent candidate for real-time graph processing tasks, which are becoming increasingly prevalent in today's data-driven landscape. Future work will focus on further optimizing the model's architecture for even greater speed and scalability, exploring its applicability in other graph-related domains, and extending its reach into more challenging graph-analytic scenarios.

References

[1] Kitaev, Nikita, Łukasz Kaiser, and Anselm Levskaya. "Reformer: The efficient transformer." arXiv preprint arXiv:2001.04451 (2020).

[2] Mao, Qiheng, et al. "Hinormer: Representation learning on heterogeneous information networks with graph transformer." Proceedings of the ACM Web Conference 2023. 2023.

[3] Hu, Ziniu, et al. "Heterogeneous graph transformer." Proceedings of the web conference 2020. 2020.

[4] Gabrielsson, Rickard Brüel, Mikhail Yurochkin, and Justin Solomon. "Rewiring with Positional Encodings for GNNs." (2022).

[5] Schlichtkrull, Michael, et al. "Modeling relational data with graph convolutional networks." The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15. Springer International Publishing, 2018.

[6] Wang, Xiao, et al. "Heterogeneous graph attention network." The world wide web conference. 2019.

[7] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

[8] Zhao, Jianan, et al. "Gophormer: Ego-graph transformer for node classification." arXiv preprint arXiv:2110.13094 (2021).

[9] Zhu, Jiong, et al. "Beyond homophily in graph neural networks: Current limitations and effective designs." Advances in neural information processing systems 33 (2020): 7793-7804.

[10] Yeh, Pei-Kai, Hsi-Wen Chen, and Ming-Syan Chen. "Random walk conformer: Learning graph representation from long and short range." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 37. No. 9. 2023.

[11] Sun, Jie, et al. "Legion: Automatically Pushing the Envelope of Multi-GPU System for Billion-ScaleGNN Training." 2023 USENIX Annual Technical Conference (USENIX ATC 23). 2023.

[12] Chen, Jinsong, et al. "NAGphormer: A tokenized graph transformer for node classification in large graphs." arXiv preprint arXiv:2206.04910 (2022).

[13] Zhang, Zaixi, et al. "Hierarchical graph transformer with adaptive node sampling." Advances in Neural Information Processing Systems 35 (2022): 21171-21183.

[14] Yang, Xiaocheng, et al. "Simple and efficient heterogeneous graph neural network." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 37. No. 9. 2023.

[15] Nguyen, Dai Quoc, Tu Dinh Nguyen, and Dinh Phung. "A self-attention network based node embedding model." Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part III. Springer International Publishing, 2021.

[16] Peng, Jingshu, Yanyan Shen, and Lei Chen. "Graphangel: Adaptive and structure-aware sampling on graph neural networks." 2021 IEEE International Conference on Data Mining (ICDM). IEEE, 2021.

[17] Rampášek, Ladislav, et al. "Recipe for a general, powerful, scalable graph transformer." Advances in Neural Information Processing Systems 35 (2022): 14501-14515.

[18] Jin, Di, et al. "Raw-gnn: Random walk aggregation based graph neural network." arXiv preprint arXiv:2206.13953 (2022).

[19] Zhang, Shichang, et al. "A Survey on Graph Neural Network Acceleration: Algorithms, Systems, and Customized Hardware." arXiv preprint arXiv:2306.14052 (2023).

[20] Zhang, Qingru, et al. "A biased graph neural network sampler with near-optimal regret." Advances in Neural Information Processing Systems 34 (2021): 8833-8844.

[21] Yun, Seongjun, et al. "Graph transformer networks." Advances in neural information processing systems 32 (2019).

[22] Wei, Yuecen, et al. "Heterogeneous graph neural network for privacy-preserving recommendation." 2022 IEEE International Conference on Data Mining (ICDM). IEEE, 2022.

[23] Liu, Ziqi, et al. "Bandit samplers for training graph neural networks." Advances in Neural Information Processing Systems 33 (2020): 6878-6888.

[24] Yin, Haoteng, et al. "Algorithm and system co-design for efficient subgraph-based graph representation learning." arXiv preprint arXiv:2202.13538 (2022).

[25] Gui, Yuntao, et al. "HGL: accelerating heterogeneous GNN training with holistic representation and optimization." SC22: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2022.

[26] Lv, Qingsong, et al. "Are we really making much progress? revisiting, benchmarking and refining heterogeneous graph neural networks." Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining. 2021.

[27] Huan, Chengying, et al. "T-gcn: A sampling based streaming graph neural network system with hybrid architecture." Proceedings of the International Conference on Parallel Architectures and Compilation Techniques. 2022.

[28] Yoon, Minji, et al. "Performance-adaptive sampling strategy towards fast and accurate graph neural networks." Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 2021.

[29] Wang, Minjie, et al. "Deep graph library: A graph-centric, highly-performant package for graph neural networks." arXiv preprint arXiv:1909.01315 (2019).

[30] Ying, Chengxuan, et al. "Do transformers really perform badly for graph representation?." Advances in neural information processing systems 34 (2021): 28877-28888.

[31] Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." Advances in neural information processing systems 30 (2017).

[32] Giraldo, Jhony H., et al. "On the trade-off between over-smoothing and over-squashing in deep graph neural networks." Proceedings of the 32nd ACM International Conference on Information and Knowledge Management. 2023.

[33] Rusch, T. Konstantin, Michael M. Bronstein, and Siddhartha Mishra. "A survey on oversmoothing in graph neural networks." arXiv preprint arXiv:2303.10993 (2023).

[34] Alon, Uri, and Eran Yahav. "On the bottleneck of graph neural networks and its practical implications." arXiv preprint arXiv:2006.05205 (2020).

[35] Ding, Mucong, et al. "Sketch-GNN: Scalable graph neural networks with sublinear training complexity." Advances in Neural Information Processing Systems 35 (2022): 2930-2943.

[36] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

[37] Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." arXiv preprint arXiv:2010.11929 (2020).

[38] Yao, Shaowei, Tianming Wang, and Xiaojun Wan. "Heterogeneous graph transformer for graph-to-sequence learning." Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 2020.

[39] Mei, Xin, et al. "Relation-aware heterogeneous graph transformer based drug repurposing." Expert Systems with Applications 190 (2022): 116165.

[40] Chen, Mengru, et al. "Heterogeneous graph contrastive learning for recommendation." Proceedings of the sixteenth ACM international conference on web search and data mining. 2023.

[41] Fan, Yujie, et al. "Heterogeneous temporal graph transformer: An intelligent system for evolving android malware detection." Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 2021.

[42] Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907 (2016).

[43] Veličković, Petar, et al. "Graph attention networks." arXiv preprint arXiv:1710.10903 (2017).

[44] Keramatfar, Abdalsamad, Mohadeseh Rafiee, and Hossein Amirkhani. "Graph Neural Networks: a bibliometrics overview." Machine Learning with Applications 10 (2022): 100401.

[45] Guo, Zhiwei, and Heng Wang. "A deep graph neural network-based mechanism for social recommendations." IEEE Transactions on Industrial Informatics 17.4 (2020): 2776-2783.

[46] Yasunaga, Michihiro, et al. "QA-GNN: Reasoning with language models and knowledge graphs for question answering." arXiv preprint arXiv:2104.06378 (2021).

[47] Zhang, Chuxu, et al. "Heterogeneous graph neural network." Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. 2019.

[48] Fu, Xinyu, et al. "Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding." Proceedings of the web conference 2020. 2020.

[49] Lu, Zhiyuan, et al. "Heterogeneous Graph Transformer with Poly-Tokenization."