UnifiedGT: Towards a Universal Framework of Transformers in Large-Scale Graph Learning

Junhong Lin[1], Xiaojie Guo[2], Shuaicheng Zhang[3], Dawei Zhou[3], Yada Zhu[2], and Julian Shun[1]
[1]*MIT CSAIL*, junhong@mit.edu, jshun@mit.edu
[2]*IBM Research*, Xiaojie.Guo@ibm.com, yzhu@us.ibm.com
[3]*Virginia Tech*, zshuai8@vt.edu, zhoud@vt.edu

Abstract—**Graph learning plays a pivotal role in many high-impact application domains. Despite significant advances in this field, there is currently no single solution that effectively handles (1) data heterogeneity, (2) long-range dependencies, (3) graph heterophily, and (4) scalability to large graphs, all at the same time. Classical graph neural networks (GNNs) and graph transformers (GTs) address some but not all of these issues, often suffering from limitations such as quadratic computation complexity and/or suboptimal generalization performance in realistic applications. This paper introduces UnifiedGT, a novel framework that systematically addresses all of these challenges by automatically providing a graph transformer architecture with multiple components via neural architecture search. UnifiedGT consists of five major components: (1) graph sampling, (2) structural prior injection, (3) graph attention, (4) local/global information mixing, and (5) type-specific feedforward networks (FFNs). This modular approach enables the efficient processing of large-scale graphs and effective management of heterogeneity and heterophily while capturing long-range dependencies. We demonstrate the versatility of UnifiedGT through comprehensive experiments on several benchmark datasets, revealing insights such as the efficacy of graph sampling for GTs, the importance of explicit graph structure injection via attention masking, and the synergistic effect of local/global information mixing via a combination of global attention with local message passing. Furthermore, we formulate these design choices into a search space, where an optimal combination can be discovered for a particular dataset via neural architecture search. Notably, UnifiedGT improves generalization performance on various graph datasets, outperforming state-of-the-art GT models by a margin of about 3.7% on average. The framework is available on Github (https://github.com/junhongmit/H2GB) and PyPI (https://pypi.org/project/H2GB/), and documentation can be found at https://junhongmit.github.io/H2GB/.**

I. Introduction

Graph neural networks (GNNs) have become an important paradigm for learning in structured data in many high-impact domains, including biology [1], chemistry [2, 3], and finance [4]. However, GNNs have been shown to suffer from over-smoothing and over-squashing [5]. Graph transformers (GTs) have emerged and have shown great potential in alleviating these limitations. While GTs are effective, many of them are designed with the implicit assumption that their graph inputs are small, homogeneous (containing a single node and edge type), and homophilic (neighbors often belong to the same class or have similar features).

Real-world graphs frequently present challenges for graph learning due to (1) data heterogeneity, (2) heterophily, (3) long-range dependencies, and (4) large sizes, each of which can significantly impede the performance of traditional graph learning approaches. As an example of data heterogeneity and scalability issues, the `OGB-LSC` dataset [6] includes `MAG240M`, a citation network that not only has diverse node/edge types (such as author, paper, and institution nodes) but is also large, containing over 240 million nodes and 3 billion edges. GTs designed for small homogeneous graphs tend to either perform poorly on such graphs or are not able to process them at all due to scalability issues. Moreover, many graphs exhibit graph heterophily, where connected nodes have different features or labels. In this case, naive message passing from neighbors can provide noisy information, which often impedes graph learning. Additionally, well-known issues like over-smoothing and over-squashing can arise when addressing long-range dependencies in real-world graph learning.

TABLE I: Comparison with several state-of-the-art GTs on handling the four graph properties in graph learning. "✓": supported; "✗": not supported.

	Data Heterogeneity	Graph Heterophily	Long-Range Dependencies	Scalability on Large Graphs
GraphTrans [15]	✗	✗	✓	✗
Gophormer [16]	✗	✗	✓	✓
Graphormer [17]	✗	✗	✓	✗
GraphGPS [14]	✗	✗	✓	✗
NAGphormer [18]	✗	✗	✓	✓
GOAT [19]	✗	✓	✓	✓
HGT [9]	✓	✗	✗	✓
UnifiedGT (ours)	✓	✓	✓	✓

Advances in graph learning research have begun to address some of these challenges, with models optimized for heterogeneous graphs [7, 8, 9] and heterophilic graphs [10, 11, 12]. Most graph transformers have shown great potential in overcoming issues with long-range dependencies [13, 14]. However, a holistic solution that simultaneously tackles data heterogeneity, graph heterophily, and long-range dependencies and can also scale to large graphs does not currently exist. We summarize existing GTs and what they support in Table I. This observation leads to our research question: ***Given a particular graph domain or task, how can we define a graph transformer solution that can effectively handle all of these challenges at once?*** Such a solution should be able to dynamically adjust its components to handle the different properties of large real-world graphs.

In this paper, we first break down the design space of state-of-the-art GTs and introduce a modular GT framework, which we call UNIFIEDGT, that synergizes the strengths of existing models while addressing their limitations. This framework uses a graph transformer backbone, enhanced with multiple components, each tailored to overcome specific challenges in graph learning. UNIFIEDGT consists of five major components: (1) graph sampling, (2) structural prior injection, (3) graph attention, (4) local/global information mixing, and (5) type-specific feedforward networks (FFNs).

Second, we summarize the existing designs for each component and extend them to solve the graph learning challenges listed above. Specifically, to tackle the data heterogeneity, we propose a novel graph attention component, *cross-type heterogeneous attention*, enabling UNIFIEDGT to handle both homogeneous and heterogeneous graphs.[1] Furthermore, we design a *cross-type k-hop neighbor attention masking* method, which allows simultaneous communication among k-hop neighbors of various node types. To address the graph heterophily challenge, we introduce a local/global information mixing module by incorporating local message passing with a long-range attention module to allow information exchange across distant homophilic nodes. To enable scalability to large graphs, we employ graph sampling to enable mini-batch training. Furthermore, we formulate these design choices into a search space, where an optimal combination can be discovered for a particular dataset through a neural architecture search.

Third, we show through comprehensive experiments on several benchmarks that UNIFIEDGT achieves superior performance over existing state-of-the-art GT models by around 3.7% on average and draw the following insights: (1) explicitly accounting for data heterogeneity can usually improve downstream task performance on heterogeneous datasets; (2) local/global information mixing enabled by a composition of a GNN and a GT greatly boosts performance on heterophilic graphs; (3) neighbor nodes within several hops can usually provide enough meaningful context in large-scale graph learning, which justifies the use of graph sampling; and (4) explicit graph structure injection through direct neighbor attention masking is significantly more effective than implicit injection through graph encoding.

In summary, our technical contributions are as follows:

1) **Findings:** Through comprehensive experiments, we provide several important findings to better understand the four challenges in current large-scale graph learning: (1) data heterogeneity, (2) graph heterophily, (3) long-range dependencies, and (4) scalability to large graphs.
2) **Framework:** We design a modular graph transformer framework, UNIFIEDGT, as an easy-to-use library with configurable architecture components. To the best of our knowledge, UNIFIEDGT is the first universal solution that can handle all four of the challenges mentioned above at the same time. Additionally, we formulate the design choices into a search space so that optimal component combinations for a given graph domain can be found through a neural architecture search.
3) **Evaluation:** We show that UNIFIEDGT surpasses the state-of-the-art GTs on a wide range of benchmarks, showcasing the potential of a universal graph learning architecture.

[1]Homogeneous graphs are a special case of heterogeneous graphs with a single node and edge type.

II. GRAPH TRANSFORMER EXPLORATION

In this section, we first present preliminaries on heterogeneous graphs and GTs. Then, we present observations from our exploration study of GTs on large real-world graphs.

A. Preliminaries

Definition 1 (**Heterogeneous Graphs**). A heterogeneous graph is a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R})$, where each node $v \in \mathcal{V}$ and edge $e \in \mathcal{E}$ has a type given by $\tau(v) : \mathcal{V} \to \mathcal{A}$ and $\phi(e) : \mathcal{E} \to \mathcal{R}$. Here, $\mathcal{A}$ and $\mathcal{R}$ are the set of node and edge types, respectively.

Definition 2 (**Graph Transformers (GTs)**). A transformer is a stack of alternating blocks of multi-head attention (MHA) modules and fully connected feed-forward networks (FFNs). Let $\mathcal{G}$ be a graph with node feature matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]^T \in \mathbb{R}^{n\times d}$, where d is the hidden dimension and $\boldsymbol{x}_i \in \mathbb{R}^d$ is the node feature of node v_i. In each layer l ($l > 0$), given the hidden feature matrix $\boldsymbol{H}^{(l-1)} \in \mathbb{R}^{n\times d}$, where $\boldsymbol{H}^{(0)} = \boldsymbol{X}$, the MHA module first linearly projects the input $\boldsymbol{H}^{(l-1)}$ to the query, key, and value spaces. This is computed using Equation (1), where the projection using weight matrices $\boldsymbol{W}_Q^{(h,l)}, \boldsymbol{W}_K^{(h,l)}$, and $\boldsymbol{W}_V^{(h,l)} \in \mathbb{R}^{d\times d_h}$ results in the matrices $\boldsymbol{Q}^{(h,l)}$, $\boldsymbol{K}^{(h,l)}$, and $\boldsymbol{V}^{(h,l)}$, representing the query, key and value spaces, respectively.

$$\boldsymbol{Q}^{(h,l)} = \boldsymbol{H}^{(l-1)}\boldsymbol{W}_Q^{(h,l)},\ \boldsymbol{K}^{(h,l)} = \boldsymbol{H}^{(l-1)}\boldsymbol{W}_K^{(h,l)},\ \boldsymbol{V}^{(h,l)} = \boldsymbol{H}^{(l-1)}\boldsymbol{W}_V^{(h,l)}. \tag{1}$$

Then, multiple attention heads are used to compute attention score on all pairs of nodes through the scaled dot product, as shown in Equation (2), where the softmax function is applied row-wise, $\boldsymbol{W}_O^{(l)} \in \mathbb{R}^{d\times d}$ is a learnable weight matrix, d_h denotes the feature dimension of the matrices $\boldsymbol{Q}^{(h,l)}$ and $\boldsymbol{K}^{(h,l)}$, $h = 1$ to H denotes the index of different attention heads, and $\|$ denotes the concatenation operator.

$$\mathrm{MHA}\left(\boldsymbol{H}^{(l-1)}\right) = \big\|_{h\in[1,H]} \left(\mathrm{softmax}\left(\frac{\boldsymbol{Q}^{(h,l)}(\boldsymbol{K}^{(h,l)})^T}{\sqrt{d_h}}\right)\boldsymbol{V}^{(h,l)}\right)\boldsymbol{W}_O^{(l)}. \tag{2}$$

The multi-head attention module $\mathrm{MHA}(\boldsymbol{H}^{(l-1)})$ concatenates several attention heads together. By combining the result with additional residual connections and normalization, the transformer layer updates features $\boldsymbol{H}^{(l-1)}$ as follows:

$$\begin{aligned}\hat{\boldsymbol{H}}^{(l)} &= \mathrm{MHA}\left(\boldsymbol{H}^{(l-1)}\right) + \boldsymbol{H}^{(l-1)}\\ \boldsymbol{H}^{(l)} &= \mathrm{FFN}\left(\hat{\boldsymbol{H}}^{(l)}\right) + \hat{\boldsymbol{H}}^{(l)} = \left[\sigma\left(\hat{\boldsymbol{H}}^{(l)}\boldsymbol{W}_1^{(l)}\right)\boldsymbol{W}_2^{(l)}\right] + \hat{\boldsymbol{H}}^{(l)},\end{aligned}$$

where σ refers to the activation function, and $\boldsymbol{W}_1^{(l)} \in \mathbb{R}^{d\times d_f}$ and $\boldsymbol{W}_2^{(l)} \in \mathbb{R}^{d_f\times d}$ are trainable parameters in the feedforward network (FFN) layer. The final output $\boldsymbol{H}^{(L)} \in \mathbb{R}^{n\times d}$ can be used as the updated node representation for downstream tasks.

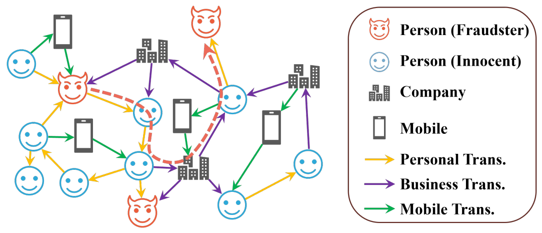

Fig. 1: Data heterogeneity (various node/edge types), graph heterophily (edges between fraudsters and innocent people), and long-range dependencies (highlighted by a red dashed arrow) can all exist in real-world graphs.

Definition 3 (**Graph Encoding (GE)**). GTs compute attention over all pairs of nodes (which are also called fully-connected GT, or FullGT) and do not inherently capture the underlying graph structure. Graph encoding was introduced to incorporate the graph topology into the attention mechanism, assigning each node a representation that reflects its structural role within the graph. Nodes that are closer in the graph, i.e., connected by shorter paths, are assigned more similar encoding values, while distant nodes have more dissimilar encodings. This encoding helps GTs focus their attention on structurally relevant nodes.

B. *Exploration Study*

Most existing GTs are designed for learning on small homogeneous graphs [20, 21, 17, 15, 14]. However, there are many real-world graphs that are not homogeneous. Such graphs can exhibit data heterogeneity, heterophily, and long-range dependencies. An example graph with all of these properties is presented in Figure 1. This example shows a financial transaction graph where different node types (person, company, etc.) and edge types (personal, business transaction, etc.) exist, making the graph heterogeneous. The class labels of fraudsters differ from those of their neighbors, making the graph heterophilic. Furthermore, fraudsters usually use long chains of transactions when committing fraud to evade detection. Therefore, graph learning solutions need to take into account long-range dependencies in order to discover such fraudulent transactions. Furthermore, these real-world graphs can also be much larger than what most prior papers on GTs have used (existing GTs have only been applied on graphs with thousands of nodes [14, 15]). To bridge the gap in the literature on GTs, we perform an empirical study to explore the power of GTs on large real-world graphs that are heterogeneous and heterophilic. Based on our comprehensive experimental study, we present a few interesting observations in this section.

1) Property 1: Long-Range Dependencies: Real-world graph learning can involve long-range dependencies that require information exchange among distant nodes, as illustrated in Figure 1. The ability of GTs to allow direct communication among arbitrary nodes in a graph has shown great potential in alleviating the well-known problems caused by over-smoothing and over-squashing in traditional graph learning models that repeatedly aggregate local information [5].

TABLE II: Average accuracy (%) and standard deviation of several models on the `ogbn-mag` dataset, calculated over 5 runs with different random seeds. The graph contains the paper, author, venue, and field of study nodes. Homo.: discard the node/edge types and treat the graph as homogeneous. Hetero.: treat the graph as heterogeneous and apply the corresponding relational model, e.g., relational GCN (R-GCN). The best results are highlighted in **bold**.

	GCN	GraphSAGE	HGT [9]
Paper nodes only	35.72 $\pm$ 0.41	35.36 $\pm$ 0.31	36.13 $\pm$ 0.43
Whole graph (Homo.)	44.90 $\pm$ 0.58	44.49 $\pm$ 0.20	45.83 $\pm$ 0.56
Whole graph (Hetero.)	**46.93 $\pm$ 0.46**	**50.94 $\pm$ 0.44**	**50.23 $\pm$ 0.48**

Therefore, this paper focuses on building a graph learning framework based on GTs.

2) Property 2: Data Heterogeneity: Many real-world graphs contain heterogeneous information, e.g., different types of nodes and edges in a financial network illustrated in Figure 1. However, most GTs are designed for homogeneous graphs and are not optimized for such heterogeneity. Ignoring the node and edge types often results in information loss and performance degradation. As a motivating example, Table II shows a comparison of classification accuracy for several graph learning models on the popular heterogeneous graph benchmark `ogbn-mag` when accounting for data heterogeneity differently. Specifically, the models trained on a reduced homogeneous graph, where only paper nodes and citation relations are retained to form a homogeneous graph [22, 6, 23], or on the entire graph without type differentiation, perform significantly worse than the models trained on the whole graph with heterogeneity information. This highlights the intrinsic value of incorporating data heterogeneity into graph learning.

Observation #1: Explicitly accounting for heterogeneity can usually benefit learning on heterogeneous graphs.

3) Property 3: Graph Heterophily: Many of the existing GT models assume graph homophily, i.e., locally connected nodes are similar in features and labels. However, recent research highlights the limitations of such graph models in dealing with heterophilic graphs, where nodes with similar features or common connections have different classes [24]. Specifically, *edge homophily* [10] is the fraction of edges that connect nodes with the same label. A low homophily value indicates that the graph has a high heterophily. `snap-patents` [12], a large-scale heterophilic graph with edge homophily 0.07, is one such example. Prior works [10, 11, 25] suggest that identifying distant homophilic nodes from the higher-order neighbors is important for task performance. The attention mechanism of GTs allows for direct communication among distant nodes, which can potentially improve task performance on heterophilic graphs. We, therefore, perform an experimental study of different ways to tackle graph heterophily in GTs on the large-scale heterophilic `snap-patents` graph, the results of which are shown in Table III.

We observe that directly applying local message passing using a conventional GCN yields 46.82% accuracy, and using global attention with a fully connected GT (FullGT) yields

TABLE III: Average accuracy (%) and standard deviation of various models on the large-scale heterophilic `snap-patents` [12] graph containing ∼3 million nodes and ∼13 million edges. Results of GT-based models that use graph encoding are marked in **bold** while the other results do not use graph encoding.

	Local	Global		Local+Global	
MLP	GCN	FullGT	2-HopGT	GCN+FullGT	GCN+2-HopGT
30.69 $\pm$ 0.22	46.82 $\pm$ 0.12	33.78 $\pm$ 0.42	51.97 $\pm$ 0.29	52.80 $\pm$ 0.40	55.09 $\pm$ 0.76
		42.94 $\pm$ 0.23	**54.56 $\pm$ 0.52**	**57.30 $\pm$ 0.58**	**57.90 $\pm$ 0.56**

42.94% accuracy. In contrast, we see that simply mixing the local and global information by combining the outputs from the GCN and FullGT significantly improves performance (GCN+FullGT, 57.30% accuracy).

Observation #2: Local/global information mixing by combining local GNNs and global attention via GTs is highly beneficial for learning on heterophilic graphs.

4) Property 4: Scalability to Large Graphs: While GTs can propagate information across long distances, they also lead to a quadratic time complexity due to the attention score computation between all pairs of nodes. This limits the applicability of GTs to large-scale graphs. However, we find that when dealing with large-scale graphs with millions of nodes, it is not necessary for a single node to directly communicate with all of the other nodes, as this could lead to severe attention dilution [26]. To address this scalability challenge, we focus on restricting GTs to operate within a k-hop neighborhood, where k is a tunable parameter. This allows us to adapt the model to different dataset scales: on large datasets, a smaller k can help mitigate attention dilution, while on smaller datasets, a larger k can help capture long-range dependencies. For example, we find that using a 2-hop neighborhood (2-HopGT) yields better results than FullGT on the `snap-patents` dataset as seen in Table III. Lastly, consistent with observation #2, we observe that we can achieve better performance by again mixing local information with the 2-hop neighbors via GCN+2-HopGT (57.90% accuracy).

Observation #3: For large-scale graph learning, attending to neighbor nodes within several hops can provide enough information.

Motivated by this observation, we can use a graph sampling strategy to significantly reduce the number of nodes that the GT needs to attend to, thereby improving runtime efficiency. Furthermore, although GTs commonly apply graph encoding to incorporate the graph topology into its attention mechanism, allowing the GT to attend to all nodes in a large-scale graph can still lead to attending to irrelevant nodes with high similarity despite the injected graph encoding. We find that the attention masking technique (introduced in Section III-B, Equation (3)) used in 2-HopGT to restrict pairwise dot product calculations to certain pairs of nodes (i.e., nodes within two hops of each other), significantly improves downstream task performance by 18.19%. From Table III, we observe that adding graph encoding improves FullGT's accuracy from 33.78% to 42.94%. Similarly, adding graph encoding to 2-HopGT further improves its accuracy from 51.97% to 54.56%. Notably, 2-HopGT without graph encoding (51.97% accuracy) still outperforms FullGT with graph encoding (42.94% accuracy) by a considerable margin, demonstrating that the attention masking strategy alone provides substantial benefits.

Observation #4: Explicit graph structure injection via attention masking can be more useful than the common implicit injection through graph encoding.

III. Unified Graph Transformer (UnifiedGT)

In this section, we introduce our general unified graph transformer framework, UnifiedGT (Figure 2). UnifiedGT consists of five modular ingredients: (1) graph sampling, (2) structural prior injection, (3) graph attention, (4) local/global information mixing, and (5) type-specific feedforward networks (FFNs). Many of the existing GTs can be implemented in this framework with various designs of each modular ingredient, as shown in Table IV. While some of the modules are based on existing work, we also introduce several new methods, e.g., cross-type k-hop neighbor attention mask, cross-type heterogeneous attention, and type-specific FFNs. Furthermore, our main contribution is enabling many existing GT-based methods as well as new ones that we come up with to be expressed within the same framework.

A. Graph Sampling

Graph sampling and minibatch training are crucial for processing large-scale graphs. However, since existing GTs mainly focus on small graphs (up to thousands of nodes), graph sampling is rarely used. To extend GTs to large graphs, we provide three standard sampling options: ***neighbor sampling*** [27], ***GraphSAINT sampling*** [28], and ***HGSampling*** [9].

B. Structural Prior

Capturing the graph structure is important for graph learning. The structural prior can be injected in two ways:

1) Graph Encoding (GE): Existing GTs commonly use graph encoding to differentiate the position of the nodes and store structural information by encoding the node proximity. However, they all focus on homogeneous graphs and cannot capture heterogeneous information. To extend GTs to heterogeneous graphs, we provide the following GE options: ***Node2Vec embedding*** [29], ***Metapath2Vec embedding*** [30], and ***knowledge graph embedding*** [31, 32, 33], ***random-walk graph encoding (RWGE)*** [34], and ***shortest path distance graph encoding (SPD-GE)*** [17].

2) Attention Masking: To address the limitation where unrestricted attention can lead to attending to irrelevant nodes despite the encoded graph structure, we propose a novel design for learning heterogeneous graph structure, ***cross-type k-hop neighbor attention masking***, which is formulated in Equation (3) [35]:

$$\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{K}^T/\sqrt{d_k} + \boldsymbol{B}, \text{ where } \boldsymbol{B}_{ij} = \begin{cases} b_m & \text{if } \boldsymbol{A}^m_{ij} > 0, m \leq k, \\ -\infty & \text{otherwise.} \end{cases} \tag{3}$$

Here $\boldsymbol{S}$ denotes the attention score matrix, $\boldsymbol{A}$ denotes the adjacency matrix and m denotes the smallest positive integer

TABLE IV: Different components of the state-of-the-art GT methods and components supported by our UNIFIEDGT framework. We list various representative existing GT methods that can be expressed by our framework. We omit the multi-head attention for simplicity. $\sigma(\cdot)$ denotes the softmax function. $\boldsymbol{A}$ is the adjacency matrix.

Graph Type	Methods	Sampling/ ***Tokenization***	Graph Encoding	Attention	Masking	FFN
Homo.	GraphTrans [15]	—	$\boldsymbol{H}=\boldsymbol{X} \parallel \mathrm{GNN}(\boldsymbol{X},\boldsymbol{A})$	$\sigma\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}}\right)\boldsymbol{V}$	—	$\boldsymbol{H}=\mathrm{ReLU}(\boldsymbol{H}\boldsymbol{W}^{(1)})\boldsymbol{W}^{(2)}$
	Gophormer [16]	Neighbor	—	$\sigma\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}}+\phi_{ij}\boldsymbol{b}\right)\boldsymbol{V}$	$\boldsymbol{A}_{ij}^m$	$\boldsymbol{H}=\mathrm{ReLU}(\boldsymbol{H}\boldsymbol{W}^{(1)})\boldsymbol{W}^{(2)}$
	Graphormer [17]	—	$\boldsymbol{H}=\boldsymbol{X}+\boldsymbol{Z}_{\mathrm{deg}}$	$\sigma\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}}+\boldsymbol{b}_{D(i,j)}\right)\boldsymbol{V}$	—	$\boldsymbol{H}=\mathrm{ReLU}(\boldsymbol{H}\boldsymbol{W}^{(1)})\boldsymbol{W}^{(2)}$
	GraphGPS [14]	—	$\boldsymbol{H}=\boldsymbol{X} \parallel \boldsymbol{P}$	$\sigma\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}}\right)\boldsymbol{V}+\mathrm{GNN}(\boldsymbol{X},\boldsymbol{A})$	—	$\boldsymbol{H}=\mathrm{ReLU}(\boldsymbol{H}\boldsymbol{W}^{(1)})\boldsymbol{W}^{(2)}$
	NAGphormer [18]	***Hop2Seq***	—	$\sigma\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}}\right)\boldsymbol{V}$	—	$\boldsymbol{H}=\mathrm{ReLU}(\boldsymbol{H}\boldsymbol{W}^{(1)})\boldsymbol{W}^{(2)}$
	GOAT [19]	Neighbor	$\boldsymbol{H}=\boldsymbol{X} \parallel \boldsymbol{P}_{\mathrm{node2vec}}$	$\sigma\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}}+\boldsymbol{b}_{D(i,j)}\right)\boldsymbol{V}$	—	$\boldsymbol{H}=\mathrm{ReLU}(\boldsymbol{H}\boldsymbol{W}^{(1)})\boldsymbol{W}^{(2)}$
Hetero.	HGT [9]	HGSampling	—	$\sigma\left(\frac{\mu\boldsymbol{Q}_{\tau(t)}\boldsymbol{W}_{\phi(e)}^{\mathrm{ATT}}\boldsymbol{K}_{\tau(s)}^T}{\sqrt{d}}\right)\boldsymbol{V}_{\tau(s)}\boldsymbol{W}_{\phi(e)}^{\mathrm{MSG}}$	$\boldsymbol{A}_{ij}^{\phi(e)}$	—
Hetero.	UNIFIEDGT (ours)	Neighbor; GraphSAINT; HGSampling	$\boldsymbol{H}_\tau=\boldsymbol{W}_\tau\boldsymbol{X}_\tau \parallel \boldsymbol{P}_\tau$	$\sigma\left(\frac{\boldsymbol{q}_i\boldsymbol{W}_{\phi(e)}^{\mathrm{edge}}\boldsymbol{k}_j^T}{\sqrt{d_k}}\right)\boldsymbol{v}_j$	$\boldsymbol{A}_{ij}^m$	$\boldsymbol{H}_\tau=\mathrm{ReLU}(\boldsymbol{H}_\tau\boldsymbol{W}_\tau^{(1)})\boldsymbol{W}_\tau^{(2)}$

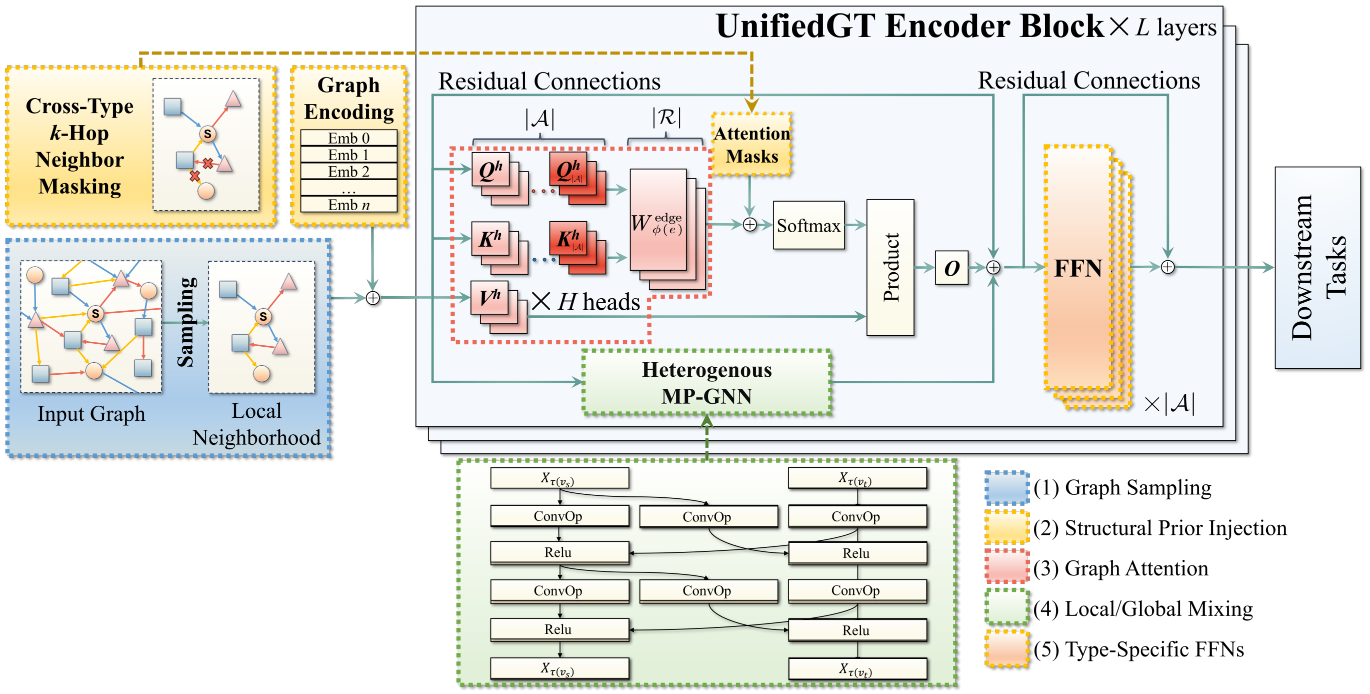

Fig. 2: Overall UNIFIEDGT framework. $|\mathcal{A}|$ is the number of node types and $|\mathcal{R}|$ is the number of edge types.

such that $\boldsymbol{A}_{ij}^m > 0$. The element $\boldsymbol{A}_{ij}^m$ in the m-order adjacency matrix captures the connectivity between nodes v_i and v_j, connected via a path of length m, where v_i and v_j can be of different types. $\boldsymbol{B}_{ij}$ denotes the attention bias added between nodes v_i and v_j, which can be used to inject structural priors [20, 36]. The added bias is passed through the softmax function to generate an attention score (see Figure 2 and Section II-A, Equation (2)). When $-\infty$ is added to $\boldsymbol{S}_{ij}$, it acts as an attention mask, zeroing out the attention score between node v_i and v_j. Our *cross-type k-hop neighbor attention masking* restricts attention to k-hop neighbors with a learnable bias b, allowing the model to learn the importance of each hop. This attention mask is *cross-type* because it allows a single node to attend to its neighbors regardless of their types.

C. Attention Calculation

Existing GTs have predominantly been applied to homogeneous graphs, where the attention is calculated by pairwise dot products on the projected key and query vectors, assuming both vectors fall into the same projection space [35]. We call this *plain graph attention*. To generalize GTs to heterogeneous graphs, the diverse semantics of nodes and edges need to be considered. UNIFIEDGT introduces the design of ***cross-type heterogeneous attention***, which utilizes the node and edge types, as shown in Equation (4) and Table IV.

$$\mathrm{MHA}\left(\boldsymbol{H}^{(l-1)}\right) = \mathop{\Big\Vert}_{h\in[1,H]}\left(\sigma\left(\frac{\boldsymbol{q}_i^{(h,l)}\boldsymbol{W}_{\phi(e)}^{\mathrm{edge}}(\boldsymbol{k}_j^{(h,l)})^T}{\sqrt{d_k}}\right)\boldsymbol{v}_j^{(h,l)}\right)\boldsymbol{W}_O^{(l)} \tag{4}$$

It performs type-dependent key-, query-, and value-projections to model the complex connections between node types. Concretely, the projection matrices $\boldsymbol{W}_{K_{\tau(v)}}$, $\boldsymbol{W}_{Q_{\tau(v)}}$, and $\boldsymbol{W}_{V_{\tau(v)}}$ are applied on node v with type $\tau(v)$, to generate the corresponding $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ matrices, as shown in the following equation, where $\boldsymbol{O} \in \{\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}\}$:

$$\boldsymbol{O} = \left[\boldsymbol{o}_1^T, \ldots, \boldsymbol{o}_n^T\right]^T = \left[\left(\boldsymbol{h}_1 \boldsymbol{W}_{O_{\tau(1)}}\right)^T, \ldots, \left(\boldsymbol{h}_n \boldsymbol{W}_{O_{\tau(n)}}\right)^T\right]^T$$

Furthermore, to incorporate edge type information, we design an edge-type dependent transformation $\boldsymbol{W}^{\text{edge}}_{\phi(e)}$ to allow modeling of diverse relations. Note that typical heterogeneous GNNs only pass messages through specified meta-relations [7, 9], while our proposed heterogeneous attention mechanism enables a single node to communicate with other nodes of different types simultaneously. Our new attention mechanism generalizes to both heterogeneous and homogeneous GTs. For homogeneous graphs, it can represent existing GTs by using a single set of $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ projection matrices across all nodes.

D. Local/Global Information Mixing

We have observed that local/global information mixing improves performance in heterophilic graph learning in Section II-B3. Inspired by the success of applying a GNN as an auxiliary module in GTs [15, 14], we incorporate the GNN into our UNIFIEDGT framework to help mix in local information. There are many choices of GNN that can be incorporated, such as ***GCN***, ***GraphSAGE***, and ***GAT***. The selected GNN will be automatically transformed into a corresponding relational GNN to handle data heterogeneity. According to the relative position between GNN layers and transformer layers, there are several composition schemes:

- **Prefixed GNN (by adding a GNN embedding to the graph encoding module)**, the most frequently adopted method (e.g., GraphTrans [15]), performs a few layers of message passing using a GNN and concatenates the results with the original node features. It allows the GNN to aggregate information from multi-hop neighbors via multilayer message passing and, therefore, brings structural information into the node embedding.
- **Parallel GNN** performs message passing using a GNN and applies a transformer in parallel, summing together the output (e.g., GraphGPS [14]). It lets the GNN focus on local information while the long-range attention module in the transformer accounts for the long-range homophilic neighbors that have a similar node embedding.
- **Prefixed+Parallel GNN**. Due to the modular design of our framework, we can apply both a prefixed and a parallel connection, which has not been done in existing GTs.

E. Type-Specific Feed-Forward Networks (FFNs)

FFNs are a typical component of transformer architectures [35] that can help the model capture more complex patterns and relationships in graphs. We introduce ***type-specific FFNs***, which apply dedicated FFN layers to each node type $\tau(v)$, as shown in Equation (5). This approach is designed to accommodate data heterogeneity, allowing the modeling of richer relationships in the semantic spaces of individual types.

$$\boldsymbol{H}_{\tau(v)} = \text{ReLU}\left(\boldsymbol{H}_{\tau(v)} \boldsymbol{W}^{(1)}_{\tau(v)}\right) \boldsymbol{W}^{(2)}_{\tau(v)}. \quad (5)$$

F. Optimal Configuration Search

The goal of UNIFIEDGT is to identify the optimal architecture for handling diverse graph properties in a specific domain. We employ Bayesian optimization (BO), a proven approach in neural architecture search [37, 38, 39], to search for such configurations. This approach utilizes a probabilistic model to predict the performance of new architectures based on prior results, efficiently guiding the search. This contrasts with supernet-based methods, which require training a single, large network encompassing many sub-architectures, often leading to substantial computational overhead. BO offers a more efficient alternative by sequentially exploring the architecture space on a smaller network.

IV. EXPERIMENTS

In this section, we demonstrate the performance of UNIFIEDGT in terms of modeling real-world homogeneous, heterogeneous, heterophilic, and long-range graph datasets.

A. Datasets

We comprehensively evaluate UNIFIEDGT using four categories of datasets (shown in Table V): large-scale homogeneous (`ogbn-products` and `ogbn-papers100M`) and heterogeneous graphs (`ogbn-mag` and `MAG240M`) from the Open Graph Benchmark [6], large-scale heterophilic graphs [12] (`arxiv-year` and `snap-patents`) and graph datasets with long-range dependencies from the Long Range Graph Benchmark [13] (`VOCSuperpixels` and `COCOSuperpixels`). Additionally, we create two heterogeneous graph datasets, `oag-cs` and `oag-eng` from the Open Academic Graph (OAG) [40, 41, 42], with more edge types than existing graphs to evaluate our model on a large number of edge types. These datasets focus on the Paper-Venue classification task in two fields, computer science (`cs`) and engineering (`eng`), and consist of more than 3000 prediction classes. Following [9], we use papers from 1900–2016 for training, 2017 for validation, and 2018–2019 for testing.

B. Baselines

We compare UNIFIEDGT with MLP [43] and four classes of state-of-the-art GNN and GT models. The first class of baselines is designed for homogeneous graphs, and includes GCN [44], GraphSAGE [27], GAT [45], GIN [46], and a scalable GT model, NAGphormer [18]. Other existing GTs designed for small graphs on these benchmarks lead to out-of-memory errors. Note that one of the scalable GT models, Gophormer [16], has no code or model available yet, but its performance has been shown to be comparable with NAGphormer [18]. The second class of baselines is designed for heterogeneous graphs, and includes relational GCN (R-GCN) [7], GraphSAGE (R-GraphSAGE), GAT (R-GAT), HAN [8], and HGT [9]. The third class of baselines

TABLE V: Statistics of graph datasets. The datasets are ordered by the number of nodes.

	arxiv-year	oag-eng	oag-cs	ogbn-mag	ogbn-products	snap-patents	ogbn-papers100M	MAG240M
# Nodes (types)	169,343 (1)	929,315 (4)	1,112,691 (4)	1,939,743 (4)	2,449,029 (1)	2,923,922 (1)	111,059,956 (1)	244,160,499 (3)
# Edges (types)	1,166,243 (1)	12,346,854 (22)	27,537,448 (22)	42,182,144 (7)	123,718,280 (1)	13,975,788 (1)	1,615,685,872 (1)	3,454,471,824 (5)
# Features	128	768	768	128	100	269	128	768
# Classes	5	3,958	3,515	349	47	5	172	153

TABLE VI: Comparison of average accuracy (%) and standard deviation of various GNN methods designed for homogeneous/heterogeneous graphs on six graph benchmarks. R-GCN, R-GraphSAGE, and R-GAT are equivalent to their homogeneous counterparts in homogeneous datasets, and so those experiments were skipped. We highlight the top **first**, **second**, and **third** results. OOM indicates that the method ran out of memory on a compute node with 1TB of main memory, and OOT indicates that the experiment took longer than 1 day.

		Homogeneous Datasets		Heterogeneous Datasets			
		ogbn-products	ogbn-papers100M	ogbn-mag	oag-cs	oag-eng	MAG240M
	MLP	61.40 ± 0.36	46.90 ± 0.44	24.57 ± 0.69	9.43 ± 0.16	20.93 ± 0.70	49.65 ± 0.79
Homogeneous	GCN	77.42 ± 0.51	62.60 ± 0.24	44.90 ± 0.58	19.44 ± 0.76	32.60 ± 0.53	64.11 ± 0.20
	GraphSAGE	77.99 ± 0.32	64.35 ± 0.33	44.49 ± 0.20	**23.02 ± 0.37**	**37.75 ± 0.68**	63.50 ± 0.51
	GAT	**79.12 ± 0.67**	**64.69 ± 0.56**	**51.30 ± 0.27**	21.24 ± 0.53	34.18 ± 0.83	56.22 ± 0.09
	GIN	74.00 ± 0.69	63.08 ± 0.42	41.66 ± 0.44	19.08 ± 0.36	30.68 ± 0.45	**65.35 ± 0.07**
	NAGphormer	75.71 ± 0.39	63.82 ± 0.16	42.47 ± 0.74	16.49 ± 0.55	31.85 ± 0.80	OOM
	GOAT	**82.00 ± 0.43**	OOT	OOT	OOT	OOT	OOT
Heterogeneous	R-GCN	–	–	46.93 ± 0.46	**23.10 ± 1.09**	**37.10 ± 0.49**	64.03 ± 0.11
	R-GraphSAGE	–	–	**50.94 ± 0.44**	22.81 ± 0.63	36.11 ± 0.45	64.09 ± 0.47
	R-GAT	–	–	41.51 ± 0.47	21.03 ± 0.59	35.90 ± 0.60	55.74 ± 0.30
	HAN	69.11 ± 0.75	62.56 ± 0.20	39.00 ± 0.22	13.14 ± 1.96	27.81 ± 0.69	60.39 ± 0.28
	HGT	75.33 ± 0.35	**64.56 ± 0.41**	50.23 ± 0.48	22.51 ± 0.40	35.51 ± 0.52	**64.82 ± 0.14**
	UNIFIEDGT (ours)	**80.98 ± 0.12**	**66.34 ± 0.43**	**53.26 ± 0.29**	**26.59 ± 1.08**	**41.39 ± 0.51**	**66.65 ± 0.40**

is optimized for heterophilic graphs, and includes jumping knowledge networks (GCNJK and GATJK) [47], MixHop [48], GPRGNN [49], and GOAT [19]. The fourth class of baselines is optimized for graphs with long-range dependencies, and includes GatedGCN [50], Transformer+LapPE [14], GraphGPS [14]. Running GOAT took longer than 1 day in most of our datasets, and resulted in a time-out (OOT) in our computing environment.

C. Training and Evaluation

The hyper-parameters of each baseline were initially set based on the authors' official experimental settings and subsequently fine-tuned for optimal performance. For the training of UNIFIEDGT, we perform a Bayesian optimization-based search on each dataset to obtain the optimal architecture configuration [51]. For ogbn-products, ogbn-mag, oga-cs, and oag-eng, we do not use GNN composition, and use HGSampling, the Metapath2Vec graph encoding, and a 1-hop neighbor attention mask. For ogbn-papers100M, we do not use a graph encoding, and use HGSampling, a parallel GCN with a GT, and a 1-hop neighbor attention mask. For MAG240M, we do not use a graph encoding, and use HGSampling, a prefixed GNN with a GT, and a 1-hop neighbor attention mask. For the heterophilic datasets, we use HGSampling, a parallel GCN with a GT, the Metapath2Vec graph encoding, and a 2-hop neighbor attention mask. For the datasets with long-range dependencies, we do not use graph sampling or an attention mask, and use a parallel GatedGCN with a GT and a Laplace graph encoding. Test performance is reported for the learned parameters corresponding to the highest validation performance. Following the official evaluation metric of each dataset, we use classification accuracy as the metric for most of the datasets, while using the F1 score as the metric for the datasets with long-range dependencies.

D. Experimental Results

We report our experimental results in Tables VI–VIII, where average accuracy and standard deviations are calculated over 5 runs with different random seeds. Table VI lists the classification results on homogeneous and heterogeneous graph datasets, and Table VII and Table VIII list the results on heterophilic graphs and graphs with long-range dependencies, respectively. We make the following observations. First, UNIFIEDGT consistently outperforms all of the existing methods on most of the graph datasets, except GOAT on the ogbn-products dataset. However, GOAT requires significantly longer training time and memory. This validates the efficacy of UNIFIEDGT as a comprehensive solution for handling large graphs with data heterogeneity, heterophily, and long-range dependencies. On homogeneous and heterogeneous graph datasets (Table VI), UNIFIEDGT outperforms the other methods with an average accuracy improvement of about 2.4% over the second-best method, and outperforms the best GT baseline by about 3.7% on average. We also observe that models that account for data heterogeneity differently exhibit highly varying performance on heterogeneous graphs. For example, while the NAGphormer model, designed for homogeneous graphs, is competitive with HGT (a baseline model

TABLE VII: Comparison of average classification accuracy (%) and standard deviation of various GNN methods optimized for graph heterophily on two heterophilic graph datasets. We highlight the top **first**, **second**, and **third** results.

	MLP	GCN	GCNJK	GAT	GATJK	GraphSAGE	HGT	MixHop	GPRGNN	NAGphormer	GOAT	UnifiedGT
`arxiv-year`	36.92 ± 0.52	46.82 ± 0.30	48.29 ± 0.57	46.00 ± 0.41	44.88 ± 0.74	49.37 ± 0.29	**51.85 ± 0.39**	49.25 ± 0.33	41.91 ± 0.43	48.22 ± 0.20	**53.57 ± 0.18**	**57.23 ± 0.59**
`snap-patents`	30.69 ± 0.22	46.82 ± 0.12	48.9 ± 0.20	39.85 ± 0.40	45.24 ± 0.36	50.95 ± 0.45	50.71 ± 0.42	52.78 ± 0.44	39.55 ± 0.16	**54.80 ± 0.13**	**54.97 ± 0.23**	**57.90 ± 0.58**

TABLE VIII: Comparison of average F1 score and standard deviation of GNN methods tailored for long-range dependencies on two datasets provided by the Long Range Graph Benchmark. We highlight the top **first**, **second**, and **third** results.

	MLP	GCN	GIN	GatedGCN	Transformer+LapPE	GPS	UnifiedGT
`VOCSuperpixels`	6.41 ± 0.11	14.51 ± 0.40	15.37 ± 0.34	**29.35 ± 0.54**	26.50 ± 0.77	**35.18 ± 0.97**	**36.79 ± 0.65**
`COCOSuperpixels`	3.02 ± 0.05	7.41 ± 0.11	8.07 ± 0.24	**27.05 ± 0.51**	24.81 ± 0.36	**33.16 ± 0.29**	**33.26 ± 0.33**

TABLE IX: Ablation studies on the `ogbn-mag` (heterogeneous) and `arxiv-year` (homogeneous, heterophilic) dataset. Accuracy (%) and standard deviation of applying proposed cross-type attention masking. Results of GT-based models that use graph encoding are marked in **bold**, while the other results do not use graph encoding.

	`ogbn-mag`	`arxiv-year`
FullGT	30.92 ± 0.38	36.85 ± 0.39
	38.61 ± 0.26	**46.93 ± 0.39**
k-HopGT	51.61 ± 0.14	52.45 ± 0.70
	53.26 ± 0.29	**56.87 ± 0.78**

TABLE X: Accuracy (%) and standard deviation of integrating a GNN with FullGT and k-HopGT using parallel/prefixed/prefixed and parallel connections.

		`ogbn-mag`	`arxiv-year`
FullGT +	prefixed GCN	50.14 ± 0.30	51.73 ± 0.32
	parallel GCN	51.01 ± 0.25	51.56 ± 0.68
	prefixed & parallel GCN	49.96 ± 0.12	51.53 ± 0.19
k-HopGT +	prefixed GCN	51.27 ± 0.10	52.66 ± 0.64
	parallel GCN	51.79 ± 0.38	51.50 ± 0.74
	prefixed & parallel GCN	50.79 ± 0.31	51.66 ± 0.83

designed for heterogeneous graphs) on homogeneous graph datasets, it significantly underperforms HGT and UnifiedGT on heterogeneous graph datasets.

For heterophilic graphs (Table VII), UnifiedGT outperforms the second-best method, GOAT, by about 3.3%. This achievement is attributed to our simple, innovative approach of mixing local and global information through the integration of local message passing with long-range attention.

Table VIII summarizes results on graphs with long-range dependencies. UnifiedGT achieves competitive and better performance against the second-best solution, GraphGPS, on both long-range graph datasets, demonstrating its effectiveness in handling datasets with long-range dependencies.

E. Ablation Studies

In this section, we perform ablation studies on a fully connected GT (FullGT) and a GT with the proposed k-hop neighbor attention masking (k-HopGT) to verify the effectiveness of our proposed components.

1) Structural prior: We evaluate the effect of two kinds of structural prior injection methods: graph encodings and attention masking. From Table IX, we observe that graph encoding is beneficial for FullGT, with accuracy increasing from ∼31% to ∼38% on `ogbn-mag`, and from ∼37% to ∼47% on `arxiv-year`. However, they still underperform many other baselines in Table VI and Table VII. When k-hop neighbor attention masking is applied (k-HopGT), the performance improves on both datasets. Note that adding graph encoding on GT with attention masking is still beneficial (1.6% improvement on `ogbn-mag`, 4.4% improvement on `arxiv-year`), but the performance gain is less significant compared to only adding attention masking. This verifies that attention masking is indeed an effective technique for graph structure injection.

2) GNN Composition: We evaluate the effect of adding a GNN (we use GCN as an example) to the GT architecture and validate that the combined architecture can lead to improved results (Table X). First, we observe that incorporating the GNN with the GT consistently improves performance compared to Table IX. Remarkably, all configurations—whether prefixed, parallel, or a hybrid of both—demonstrate superior accuracy over models employing only a single component (GCN or FullGT alone). For example, all combinations of GCN and FullGT achieve around 49.96–51.01% accuracy on `ogbn-mag`, while GCN alone only achieves 44.90% accuracy as shown in Table VI, and FullGT alone only achieves 30.92% accuracy, as shown in Table IX. This demonstrates the benefit of the combination of a GNN with a GT.

V. Related Works

A. Graph Transformers

Graph transformers [17, 52, 14, 53, 9, 54, 19] extend the transformative capabilities of conventional transformer architectures, which have made significant strides in both natural language processing and computer vision. By using powerful attention mechanisms, they can effectively capture long-range dependencies in graph structures. Transformers overcome limitations in traditional GNNs [44, 27], such as over-smoothing and over-squashing [5]. To deal with the idiosyncrasies of graph data, like local structure and relative positional information, specialized techniques such as graph-specific positional and structure encodings have been proposed [55, 56]. Current research primarily focuses on homogeneous graphs [17, 19] or specialized applications like heterogeneous molecule graphs [9, 8, 14]. Only a few existing works [19] explore the power of GTs in heterophilic graph learning. Our work aims to bridge this gap by providing a graph transformer framework that can handle large heterogeneous and heterophilic graphs.

B. Learning on Large-Scale Graphs

The computational intricacies of learning on large-scale graphs frequently cause traditional GNNs to run out of memory [57]. To mitigate this, various solutions have been proposed. Graph partitioning [58] splits the graph into smaller sub-graphs to facilitate parallel computing [59]. Sampling methods [9] reduce the computational load by focusing on representative subgraphs [60, 27, 28]. Mini-batching techniques strive for a balance between computational efficiency and learning efficacy [61]. All of these strategies aim to optimize learning outcomes while preserving the graph's structural and feature information. This paper applies these strategies to graph transformers to perform learning on large heterogeneous and heterophilic graphs.

VI. Conclusion

In this paper, we proposed UNIFIEDGT the first framework for real-world graph learning that can simultaneously handle the four challenges of (1) data heterogeneity, (2) graph heterophily, (3) long-range dependencies, and (4) scalability to large graphs. UNIFIEDGTincludes various options for graph sampling, structural prior injection, graph attention and a combination of local/global information, and can be configured via neural architecture search. Using UNIFIEDGT, we performed a systematic empirical study and demonstrated its superior performance on several graph benchmark datasets. We showed that the optimal configuration in UNIFIEDGT can consistently outperform all of the existing baselines.

Acknowledgements

This work is funded by the MIT-IBM AI Watson Lab, NSF awards #CCF-1845763, #CCF-2316235, #CCF-2403237, #IIS-2339989, and #IIS-2406439, DOE Early Career Award #DE-SC0018947, Google Faculty Research Award, Google Research Scholar Award, DARPA under contract No. HR00112490370 and No. HR001124S0013, DHS CINA, Amazon-Virginia Tech Initiative for Efficient and Robust Machine Learning, Cisco, 4-VA, Commonwealth Cyber Initiative, and Virginia Tech. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the government.

References

[1] K. Huang, C. Xiao, L. M. Glass, M. Zitnik, and J. Sun, "SkipGNN: Predicting molecular interactions with skip-graph networks," *Scientific Reports*, vol. 10, no. 1, pp. 1–16, 2020.
[2] T. Zhao, Y. Hu, L. R. Valsdottir, T. Zang, and J. Peng, "Identifying drug–target interactions based on graph convolutional network and deep neural network," *Briefings in Bioinformatics*, vol. 22, no. 2, pp. 2141–2150, 2021.
[3] D. Jiang, Z. Wu, C.-Y. Hsieh, G. Chen, B. Liao, Z. Wang, C. Shen, D. Cao, J. Wu, and T. Hou, "Could graph neural networks learn better molecular representation for drug discovery? a comparison study of descriptor-based and graph-based models," *Journal of Cheminformatics*, vol. 13, no. 1, pp. 1–23, 2021.
[4] D. Wang, J. Lin, P. Cui, Q. Jia, Z. Wang, Y. Fang, Q. Yu, J. Zhou, S. Yang, and Y. Qi, "A semi-supervised graph attentive network for financial fraud detection," in *IEEE International Conference on Data Mining (ICDM)*, 2019, pp. 598–607.
[5] Y. Song, C. Zhou, X. Wang, and Z. Lin, "Ordered GNN: Ordering message passing to deal with heterophily and over-smoothing," in *International Conference on Learning Representations (ICLR)*, 2023.
[6] W. Hu, M. Fey, H. Ren, M. Nakata, Y. Dong, and J. Leskovec, "OGB-LSC: A large-scale challenge for machine learning on graphs," in *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2021.
[7] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference (ESWC)*, 2018, pp. 593–607.
[8] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *Proceedings of the International Conference on World Wide Web (WWW)*, 2019, pp. 2022–2032.
[9] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous graph transformer," in *Proceedings of the International Conference on World Wide Web (WWW)*, 2020, pp. 2704–2710.
[10] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond homophily in graph neural networks: Current limitations and effective designs," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 7793–7804, 2020.
[11] J. Zhu, R. A. Rossi, A. Rao, T. Mai, N. Lipka, N. K. Ahmed, and D. Koutra, "Graph neural networks with heterophily," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 35, no. 12, 2021, pp. 11 168–11 176.
[12] D. Lim, F. Hohne, X. Li, S. L. Huang, V. Gupta, O. Bhalerao, and S. N. Lim, "Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 20 887–20 902, 2021.
[13] V. P. Dwivedi, L. Rampášek, M. Galkin, A. Parviz, G. Wolf, A. T. Luu, and D. Beaini, "Long range graph benchmark," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 22 326–22 340, 2022.
[14] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, "Recipe for a general, powerful, scalable graph transformer," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 14 501–14 515, 2022.
[15] Z. Wu, P. Jain, M. Wright, A. Mirhoseini, J. E. Gonzalez, and I. Stoica, "Representing long-range context for graph neural networks with global attention," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 13 266–13 279, 2021.
[16] J. Zhao, C. Li, Q. Wen, Y. Wang, Y. Liu, H. Sun, X. Xie, and Y. Ye, "Gophormer: Ego-graph transformer for node classification," *arXiv preprint arXiv:2110.13094*, 2021.
[17] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu, "Do transformers really perform bad for graph representation?" in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021, pp. 28 877–28 888.
[18] J. Chen, K. Gao, G. Li, and K. He, "NAGphormer: A tokenized graph transformer for node classification in large graphs," in *The International Conference on Learning Representations (ICLR)*, 2022.
[19] K. Kong, J. Chen, J. Kirchenbauer, R. Ni, C. B. Bruss, and T. Goldstein, "GOAT: A global transformer on large-scale graphs," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2023, pp. 17 375–17 390.
[20] V. P. Dwivedi and X. Bresson, "A generalization of transformer networks to graphs," *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021.
[21] D. Kreuzer, D. Beaini, W. Hamilton, V. Létourneau, and P. Tossou, "Rethinking graph transformers with spectral attention," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, pp. 21 618–21 629, 2021.
[22] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 22 118–22 133, 2020.
[23] A. Khatua, V. S. Mailthody, B. Taleka, T. Ma, X. Song, and W.-m. Hwu, "IGB: Addressing the gaps in labeling, features, heterogeneity, and size of public graph datasets for deep learning research," in *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2023.
[24] X. Zheng, Y. Liu, S. Pan, M. Zhang, D. Jin, and P. S. Yu, "Graph

neural networks for graphs with heterophily: A survey," *arXiv preprint arXiv:2202.07082*, 2022.

[25] X. Li, R. Zhu, Y. Cheng, C. Shan, S. Luo, D. Li, and W. Qian, "Finding global homophily in graph neural networks when meeting heterophily," in *International Conference on Machine Learning (ICML)*, 2022, pp. 13 242–13 256.

[26] Z. Qin, X. Han, W. Sun, D. Li, L. Kong, N. Barnes, and Y. Zhong, "The devil in linear transformer," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022, pp. 7025–7041.

[27] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[28] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "GraphSAINT: Graph sampling based inductive learning method," in *International Conference on Learning Representations (ICLR)*, 2020.

[29] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 855–864.

[30] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2017, pp. 135–144.

[31] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 26, 2013.

[32] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *International Conference on Machine Learning (ICML)*, 2016, pp. 2071–2080.

[33] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *The International Conference on Learning Representations (ICLR)*, 2015.

[34] V. P. Dwivedi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, "Graph neural networks with learnable structural and positional representations," in *International Conference on Learning Representations*, 2022.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[36] G. Mialon, D. Chen, M. Selosse, and J. Mairal, "Graphit: Encoding graph structure in transformers," *arXiv preprint arXiv:2106.05667*, 2021.

[37] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 25, 2012.

[38] C. White, W. Neiswanger, and Y. Savani, "BANANAS: Bayesian optimization with neural architectures for neural architecture search," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 35, no. 12, 2021, pp. 10 293–10 301.

[39] B. Ru, X. Wan, X. Dong, and M. Osborne, "Interpretable neural architecture search via bayesian optimization with weisfeiler-lehman kernels," *arXiv preprint arXiv:2006.07556*, 2020.

[40] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-J. Hsu, and K. Wang, "An overview of microsoft academic service (MAS) and applications," in *Proceedings of the International Conference on World Wide Web (WWW)*, 2015, pp. 243–246.

[41] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "ArnetMiner: Extraction and mining of academic social networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2008, pp. 990–998.

[42] F. Zhang, X. Liu, J. Tang, Y. Dong, P. Yao, J. Zhang, X. Gu, Y. Wang, B. Shao, R. Li *et al.*, "OAG: Toward linking large-scale heterogeneous entity graphs," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2019, pp. 2585–2595.

[43] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[44] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[45] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[46] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations (ICLR)*, 2019.

[47] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 5453–5462.

[48] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. Ver Steeg, and A. Galstyan, "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *International Conference on Machine Learning (ICML)*. PMLR, 2019, pp. 21–29.

[49] E. Chien, J. Peng, P. Li, and O. Milenkovic, "Adaptive universal generalized pagerank graph neural network," in *International Conference on Learning Representations (ICLR)*, 2021.

[50] X. Bresson and T. Laurent, "Residual gated graph convnets," *arXiv preprint arXiv:1711.07553*, 2017.

[51] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy, "BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[52] E. Min, R. Chen, Y. Bian, T. Xu, K. Zhao, W. Huang, P. Zhao, J. Huang, S. Ananiadou, and Y. Rong, "Transformer for graphs: An overview from architecture perspective," *arXiv preprint arXiv:2202.08455*, 2022.

[53] S. Zhang, Q. Ning, and L. Huang, "Extracting temporal event relation with syntax-guided graph transformer," in *North American Chapter of the Association for Computational Linguistics (NAACL)*, July 2022, pp. 379–390.

[54] S. Yao, T. Wang, and X. Wan, "Heterogeneous graph transformer for graph-to-sequence learning," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020, pp. 7145–7154.

[55] D. Chen, L. O'Bray, and K. Borgwardt, "Structure-aware transformer for graph representation learning," in *International Conference on Machine Learning (ICML)*. PMLR, 2022, pp. 3469–3489.

[56] Y. Chen, J. You, J. He, Y. Lin, Y. Peng, C. Wu, and Y. Zhu, "SP-GNN: Learning structure and position information from graphs," *Neural Networks*, vol. 161, pp. 505–514, 2023.

[57] A. Gupta, P. Matta, and B. Pant, "Graph neural network: Current state of art, challenges and applications," *Materials Today: Proceedings*, vol. 46, pp. 10 927–10 932, 2021.

[58] P.-O. Fjällström, *Algorithms for Graph Partitioning: A Survey*. Linköping University Electronic Press, 1998.

[59] Y. Shao, H. Li, X. Gu, H. Yin, Y. Li, X. Miao, W. Zhang, B. Cui, and L. Chen, "Distributed graph neural network training: A survey," *ACM Computing Surveys*, vol. 56, no. 8, pp. 1–39, 2024.

[60] P. Hu and W. C. Lau, "A survey and taxonomy of graph sampling," *arXiv preprint arXiv:1308.5865*, 2013.

[61] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2019, pp. 257–266.