Graph-Agnostic Linear Transformers

Zhiyu Guo[a,b], Yang Liu[a,*], Xiang Ao[a,b,c], Yateng Tang[d], Xinhuan Chen[d], Xuehao Zheng[d], Qing He[a,b]

[a] *State Key Lab of AI Safety, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China*
[b] *University of Chinese Academy of Sciences, Beijing, 100190, China*
[c] *Institute of Intelligent Computing Technology, Chinese Academy of Sciences, Suzhou, 215100, China*
[d] *Independent Researcher, China*

Abstract

Graph Transformers (GTs), as emerging foundational encoders for graph-structured data, have shown promising performance due to the integration of local graph structures with global attention mechanisms. However, the complex attention functions and their coupling with graph structures incur significant computational overhead, particularly in large-scale graphs. In this paper, we decouple graph structures from Transformers and propose the Graph-Agnostic Linear Transformer (GALiT). In GALiT, graph structures are solely utilized to denoise raw node features before training, as our findings reveal that these denoised features have integrated the main information of the graph structure and can replace it to guide Transformers. By excluding graph structures from the training and inference stages, GALiT serves as a graph-agnostic model which significantly reduces computational complexity. Additionally, we simplify the linear attention functions inherited from traditional Transformers, which further reduces computational overhead while still capturing the relationships between nodes. Through weighted combination, we integrate the denoised features into the attention mechanism, as our theoretical analysis reveals the key role of the synergy between linear attention and denoised features in enhancing representation diversity. Despite decoupling graph structures and simplifying attention mechanisms, our model surprisingly outperforms most GNNs and GTs on benchmark graphs. Experi-

*Corresponding author.
Email address: liuyang2023@ict.ac.cn (Yang Liu)

mental results indicate that GALiT achieves high efficiency while maintaining or even enhancing performance.

Keywords: Graph Neural Network, Graph Transformer, Linear Attention, Graph-Agnostic Model

1. Introduction

Learning on graphs containing nodes and edges is a fundamental problem in data mining, with extensive applications in social and natural sciences [1, 2, 3, 4, 5, 6, 7]. A major challenge is to extract effective node representations from graph-structure data for various downstream tasks, especially under limited computational resources.

As emerging foundational encoders, Graph Transformers (GTs) [8, 9, 10, 11] have shown promising performance in learning node representations for graph-structured data. Unlike Graph Neural Networks (GNNs) [12, 13, 14, 15] recursively aggregating information from neighboring nodes, GTs adaptively aggregate information from all nodes with the global attention mechanism inherited from traditional Transformers [16]. Meanwhile, they leverage graph structures to generate edge embeddings [9], guide the learning of attention weights [17], or directly combine Transformers with GNNs [11]. Therefore, due to the integration of graph structures with Transformers, GTs can utilize local relational information while capturing global latent relationships, thereby achieving superior expressiveness compared to GNNs. Nevertheless, these advantages are achieved at the expense of efficiency, which limits the applicability of GTs on large-scale graphs.

The coupling of graph structures with Transformers is a primary factor affecting the scalability of GTs. Several efforts [18, 17, 19, 11] have been devoted to improving the scalability of GTs. SGFormer [11] combines Transformers with only a shallow GNN to enhance the scalability, but sacrifices the expressiveness of multi-layer GNNs and does not eliminate the computational overhead introduced by graph structures. Contrastively, we argue that decoupling graph structures from Transformers is a more effective way to eliminate the complexity induced by graph structures. To illustrate this in a simple case, we first denoise the raw node features using graph structures. Then the denoised features are directly put into a two-layer MLP without the graph structure information. The results in Figure 1 show that the MLP with denoised features achieves better performance in terms of both efficacy

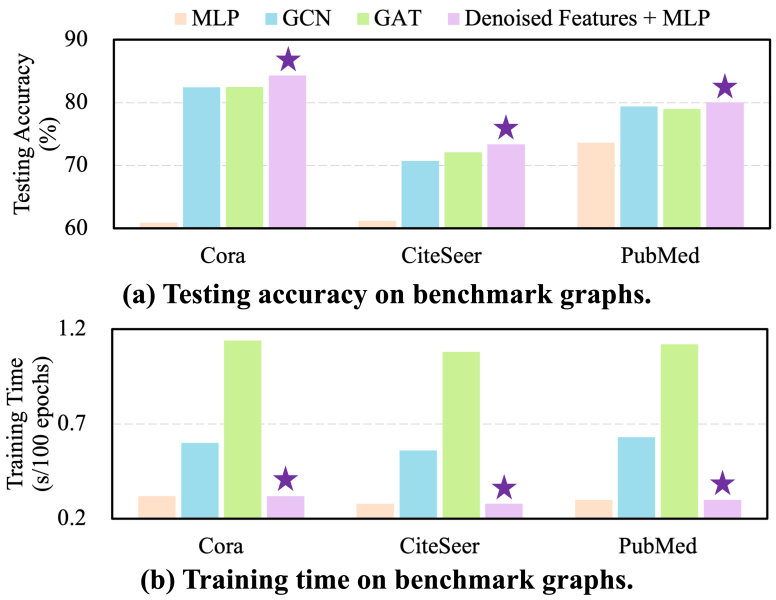

Figure 1: Inspirational results for node classification on benchmark graphs, including testing accuracy and training time per 100 epochs. We find that merely utilizing the raw node features denoised through graph structures with a two-layer MLP not only achieves effective classification results but also significantly reduces computational overhead. These results inspire our simplification of existing Graph Transformers.

and efficiency. Therefore, we believe that the denoised raw node features have already integrated the main information of the graph structure and can be utilized to guide the adaptive aggregation process within Transformers.

Additionally, the complex attention functions in existing GTs introduce unnecessary computational overhead and increase the risk of overfitting. Specifically, feature vectors are transformed into query and key vectors through separate linear transformations, followed by computing attention weights using scaled dot-product. To reduce the $\mathcal{O}(N^2)$ complexity of the traditional Softmax attention [16], recent studies [9, 17, 20, 11] have introduced linear attention [21, 22] into GTs and achieved $\mathcal{O}(N + E)$ complexity, making them applicable to large-scale graphs. To further simplify the linear attention mechanism and reduce computational overhead, we replace the complex attention functions with low-parameter weighted or mapped cosine similarity. Moreover, through weighted combination, we integrate denoised features into the attention mechanism, as our theoretical analysis reveals the key

role of the synergy between linear attention and denoised features in enhancing representation diversity. Experimental results demonstrate that our simplification can reduce computational overhead while still capturing the relationships between nodes.

In this paper, we integrate the above two techniques of decoupling graph structures and simplifying attention mechanisms into general GT frameworks, and propose our model named Graph-Agnostic Linear Transformer (GALiT). In GALiT, graph structures are solely utilized to obtain denoised raw node features before training. These pre-processed features are cached and then guide the adaptive aggregation process within Transformers. Therefore, GALiT serves as a model that is agnostic to the graph structure during the training and inference phases. This distinction is crucial, as it significantly reduces computational complexity by confining the graph dependency to a one-time, low-cost pre-processing step. Additionally, we simplify the linear attention functions inherited from traditional Transformers, further reducing computational overhead while still capturing the relationships between nodes. Despite decoupling graph structures and simplifying attention mechanisms, our model surprisingly outperforms most GNNs and GTs on benchmark graphs. Experimental results demonstrate that GALiT can achieve high efficiency while maintaining or even enhancing performance. Beyond current results, we believe that the proposed methodology could establish a new technical path for the design of simplified graph encoders.

In summary, our contributions can be listed as follows:

- We reveal that decoupling graph structures from Transformers is an effective way to reduce the computational complexity of Graph Transformers.
- We simplify the traditional linear attention mechanism in GTs using a weighted or mapped form of cosine similarity, which further reduces computational overhead while still capturing the relationships between nodes.
- Extensive experiments are conducted on homophilic, heterophilic, and large-scale graphs to validate the rationality and effectiveness of the proposed model.

The remainder of this paper is organized as follows. Section 2 introduces definitions and the problem statement of this paper. Section 3 details the

proposed GALiT framework, and Section 4 illustrates the experimental results. Section 5 surveys the related research in the literature, and Section 6 concludes the paper.

2. Preliminary

We denote a graph as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where the node set $\mathcal{V} = \{v_1, \ldots, v_N\}$ comprises N nodes and the edge set $\mathcal{E} = \{e_1, \ldots, e_E\}$ comprises E edges including self-loops. Each node is associated with a D-dimensional feature vector, and all node features are denoted as $\mathbf{X} \in \mathbb{R}^{N \times D}$. The graph structure of $\mathcal{G}$ can be represented as an adjacency matrix $\mathbf{A} \in \{0, 1\}^{N \times N}$, where $\mathbf{A}_{ij} = 1$ if $(i, j) \in \mathcal{E}$ and $\mathbf{A}_{ij} = 0$ otherwise. Additionally, we denote all node labels as one-hot vectors $\mathbf{Y} \in \{0, 1\}^{N \times C}$, where C is the number of classes. In general, learning representations on graphs aims to generate node embeddings $\mathbf{Z} = f(\mathbf{X}, \mathbf{A}; \theta) \in \mathbb{R}^{N \times d}$ that are useful for downstream tasks, where d denotes the dimension of embeddings and θ denotes the parameters of model f.

2.1. Graph Neural Networks

Graph Neural Networks (GNNs) [12, 13, 23, 24] are classic encoders for graph-structured data. The core of GNNs lies in the message-passing paradigm, which recursively aggregates information from neighboring nodes to compute the node representations:

$$\bar{\mathbf{Z}}_i^{(l)} = f^{(l)}\left(\mathbf{Z}_i^{(l)}\right), \quad \mathbf{Z}_i^{(l+1)} = \sum_{(i,j)\in\mathcal{E}} \tilde{\mathbf{S}}_{ij}^{(l)} \bar{\mathbf{Z}}_j^{(l)}, \tag{1}$$

where $\mathbf{Z}_i^{(l)}$ denotes the node embedding at the l-th layer, and $f^{(l)}$ denotes feature transformation. There are different strategies for computing the aggregation weights $\tilde{\mathbf{S}}_{ij}$. For instance, Graph Convolutional Networks (GCN) [12] employ the normalized adjacency matrix as fixed aggregation weights, i.e., $\tilde{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, where $\mathbf{D}$ is the diagonal matrix with $\mathbf{D}_{ii}$ being the degree of node v_i. In contrast, Graph Attention Networks (GAT) [13] introduce an attention mechanism to adaptively compute the aggregation weights.

2.2. Graph Transformers

Graph Transformers (GTs) [8, 25, 26, 27] have emerged as powerful graph encoders. Unlike GNNs recursively aggregating information from neighboring nodes, GTs adaptively aggregate information from all nodes:

$$\bar{\mathbf{Z}}_i^{(l)} = f^{(l)}\left(\mathbf{Z}_i^{(l)}\right), \quad \mathbf{Z}_i^{(l+1)} = \sum_j \tilde{\mathbf{S}}_{ij}^{(l)} \bar{\mathbf{Z}}_j^{(l)}, \tag{2}$$

To compute the aggregation weights $\tilde{\mathbf{S}}_{ij}$, existing GTs typically inherit the Softmax attention mechanism from traditional Transformers. Specifically, feature embeddings $\mathbf{Z}$ are transformed into query vectors $\mathbf{Q} \in \mathbb{R}^{N\times d}$ and key vectors $\mathbf{K} \in \mathbb{R}^{N\times d}$ through linear transformations, followed by the computation of attention weights with scaled dot-product and non-linear normalization:

$$\mathbf{Q} = \mathbf{Z}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{Z}\mathbf{W}_K, \quad \tilde{\mathbf{S}} = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right), \tag{3}$$

where $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{d\times d}$ are learnable projection matrices and bias vectors, respectively. Meanwhile, various strategies have been employed to integrate graph structures with attention mechanisms, such as generating edge embeddings, guiding the learning of attention weights, or directly combining Transformers with GNNs. Generally, these strategies can be viewed as expansions in the computation of the aggregation weights $\tilde{\mathbf{S}}_{ij}$.

2.3. Graph Signal Denoising

Recent studies [28, 29, 30, 11] have interpreted graph representation learning as a graph signal denoising process, unifying various GNNs and GTs under this framework. Specifically, the noisy signal is denoted by $\bar{\mathbf{Z}}$, and the objective of graph signal denoising is to recover the clean signal $\mathbf{Z}$ from the following optimization problem:

$$\mathcal{L} = \frac{1}{2}\sum_i \left\|\mathbf{Z}_i - \bar{\mathbf{Z}}_i\right\|^2 + \frac{\lambda}{4}\sum_{i,j} \tilde{\mathbf{S}}_{ij} \left\|\mathbf{Z}_i - \mathbf{Z}_j\right\|^2, \tag{4}$$

where $\|\cdot\|$ denotes the L2 norm. In this framework, the first term embodies the fidelity principle, which guides $\mathbf{Z}$ to align with $\bar{\mathbf{Z}}$; the second term embodies the smoothness principle, which employs Laplacian regularization to ensure the smoothness of $\mathbf{Z}$ across the attention graph $\tilde{\mathbf{S}}$; $\lambda > 0$ serves as a trade-off weight parameter. Both the message-passing mechanism of

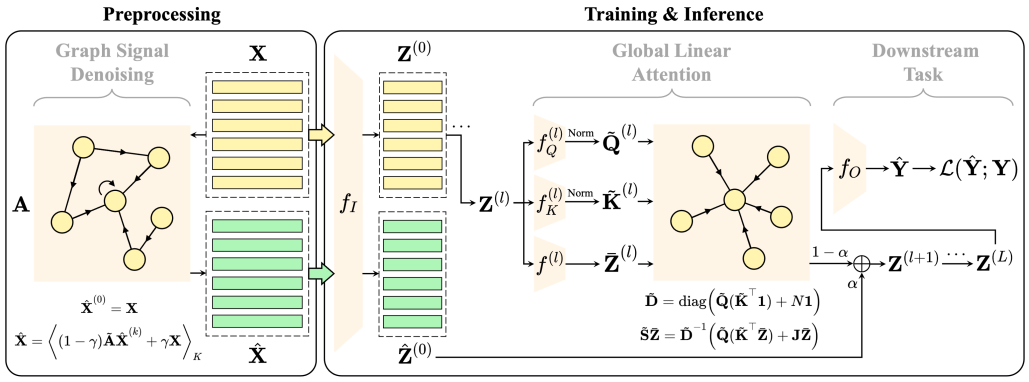

Figure 2: Illustration of our proposed GALiT model on an example graph, with node features $\mathbf{X}$ and adjacency matrix $\mathbf{A}$. Before training, the graph structure $\mathbf{A}$ is utilized to obtain the denoised raw node features $\hat{\mathbf{X}}$. These pre-processed features can replace the direct use of the graph structure, guiding the adaptive aggregation process within Transformers. During training and inference, the initial raw feature embeddings $\mathbf{Z}^{(0)}$ and the pre-processed feature embeddings $\hat{\mathbf{Z}}^{(0)}$ are generated by a shallow neural layer f_I. In the global linear attention module, our simplified linear attention functions are used to capture latent relationships and propagate information across all node pairs with $\mathcal{O}(N)$ complexity. The outputs from the attention module are combined with the denoised feature embeddings $\hat{\mathbf{Z}}^{(0)}$, and then serve as the input for the next layer. After L layers, the output node representations $\mathbf{Z}^{(L)}$ are transformed by a shallow neural layer f_O for downstream tasks (e.g. node classification).

GNNs and the attention mechanism of GTs can be interpreted as methods to approximate the solution to the above optimization problem, with their differences lying in how they construct the attention graph $\tilde{\mathbf{S}}$.

3. Methodology

In this section, we elaborate on the two proposed techniques of decoupling graph structures and simplifying attention mechanisms, and integrate them into general GT frameworks.

3.1. Model Overview

In this work, we propose the GALiT model, which incorporates two key innovations: decoupling the graph structure from the Transformer framework and simplifying the attention mechanism. As illustrated in Figure 2, the graph structure $\mathbf{A}$ is first used to obtain denoised node features $\hat{\mathbf{X}}$ before training. These pre-processed features replace the direct use of the graph

structure, thus guiding the adaptive aggregation within the Transformer layers. During both training and inference, the raw feature embeddings $\mathbf{Z}^{(0)}$ and the denoised feature embeddings $\hat{\mathbf{Z}}^{(0)}$ are generated by a shallow neural layer f_I. The simplified linear attention mechanism in the global attention module captures latent node relationships and propagates information efficiently with $\mathcal{O}(N)$ complexity. The output of this module is combined with the denoised feature embeddings and serves as input for the subsequent layers. After passing through L layers, the final node embeddings $\mathbf{Z}^{(L)}$ are processed by a shallow neural layer f_O for downstream tasks such as node classification.

3.2. Decoupling Graph Structures

In existing Graph Transformers, the coupling of graph structures with Transformers incurs significant computational overhead, particularly in large-scale graphs. To enhance the scalability of GTs, we propose to decouple graph structures from Transformers, ensuring they are not involved in the training and inference stages.

Inspired by the interpretation of graph representation learning as an adaptive graph signal denoising process, we explore a simple case in which raw node features are denoised along the graph structure in an unsupervised manner. Specifically, we treat the raw node features $\mathbf{X}$ as a noisy signal and aim to recover the clean signal $\hat{\mathbf{X}} \in \mathbb{R}^{N\times D}$ by solving the following optimization problem:

$$\mathcal{L} = \frac{1}{2}\sum_i \left\|\hat{\mathbf{X}}_i - \mathbf{X}_i\right\|^2 + \frac{\xi}{4}\sum_{(i,j)\in\mathcal{E}} \left\|\frac{\hat{\mathbf{X}}_i}{\sqrt{\mathbf{D}_{ii}}} - \frac{\hat{\mathbf{X}}_j}{\sqrt{\mathbf{D}_{jj}}}\right\|^2. \tag{5}$$

In this framework, the first term guides $\hat{\mathbf{X}}$ to be close to $\mathbf{X}$, while the second term applies Laplacian regularization to ensure the smoothness of $\hat{\mathbf{X}}$ across the graph structure. The hyper-parameter $\xi > 0$ serves as a trade-off weight between these two terms. The above convex optimization problem has an analytical solution, and an approximate solution can be obtained through recursive computation:

$$\hat{\mathbf{X}} = \gamma\left(\mathbf{I} - (1-\gamma)\tilde{\mathbf{A}}\right)^{-1}\mathbf{X}, \tag{6}$$

$$\hat{\mathbf{X}} \doteq \left\langle (1-\gamma)\tilde{\mathbf{A}}\hat{\mathbf{X}}^{(k)} + \gamma\mathbf{X}\right\rangle_K, \quad \hat{\mathbf{X}}^{(0)} = \mathbf{X}, \tag{7}$$

where $\gamma = (1+\xi)^{-1} \in (0,1)$, and K denotes the number of recursive iterations. Due to the $\mathcal{O}(N^3)$ complexity of matrix inversion, recursive computation is commonly used to approximate the denoised features $\hat{\mathbf{X}}$, particularly in the case of large-scale graphs.

We explore the direct use of denoised features $\hat{\mathbf{X}}$ for node classification with a two-layer MLP, and results are shown in Figure 1. We find that this approach outperforms typical GNNs, even without integrating graph structures during training and inference stages. This finding suggests that the denoised features $\hat{\mathbf{X}}$ have already integrated the main information of the graph structure and can replace it to guide the adaptive aggregation process within Transformers. Therefore, we solely utilize the graph structure to obtain denoised raw node features $\hat{\mathbf{X}}$ before training. These pre-processed features are cached and then input into each Transformer layer:

$$\mathbf{Z}^{(0)} = f_I(\mathbf{X}), \quad \hat{\mathbf{Z}}^{(0)} = f_I(\hat{\mathbf{X}}), \tag{8}$$

$$\bar{\mathbf{Z}}^{(l)} = f^{(l)}\left(\mathbf{Z}^{(l)}\right), \quad \mathbf{Z}^{(l+1)} = (1-\alpha)\tilde{\mathbf{S}}^{(l)}\bar{\mathbf{Z}}^{(l)} + \alpha\hat{\mathbf{Z}}^{(0)}, \tag{9}$$

where $\alpha \in (0,1)$ is a weight hyper-parameter, and $\mathbf{Z}^{(0)} \in \mathbb{R}^{N\times d}$ and $\hat{\mathbf{Z}}^{(0)} \in \mathbb{R}^{N\times d}$ are the initial raw feature embeddings and pre-processed feature embeddings, respectively, obtained through a feature transformation f_I (e.g., a single-layer MLP). In this framework, $\tilde{\mathbf{S}}$ is computed solely by the global attention mechanism inherited from Transformers, making this a graph-agnostic model, which excludes graph structures from training and inference.

We theoretically analyze the proposed modification based on the interpretation of each Transformer layer as an optimization step for an adaptive graph signal denoising problem.

Theorem 1. *For any given attention matrix $\tilde{\mathbf{S}} \in \mathbb{R}^{N\times N}$, Eq. (9) is equivalent to a gradient descent operation with step size $\alpha \in (0,1)$ for the following optimization problem:*

$$\mathcal{L} = \frac{1}{2}\sum_i \left\|\mathbf{Z}_i - \hat{\mathbf{Z}}_i^{(0)}\right\|^2 + \frac{\lambda}{4}\sum_{i,j}\tilde{\mathbf{S}}_{ij}\left\|\mathbf{Z}_i - \mathbf{Z}_j\right\|^2, \tag{10}$$

where the first term guides $\mathbf{Z}$ to be close to $\hat{\mathbf{Z}}^{(0)}$, while the second term ensures the smoothness of $\mathbf{Z}$ across the attention graph $\mathbf{S}_{ij}$. The hyper-parameter $\lambda = 1/\alpha - 1$ serves as a trade-off weight between the above two terms.

Proof. The gradient of $\mathcal{L}$ w.r.t. $\mathbf{Z}_i$ can be computed by:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{Z}_i} = \mathbf{Z}_i - \hat{\mathbf{Z}}_i^{(0)} + \frac{1-\alpha}{\alpha} \sum_j \tilde{\mathbf{S}}_{ij} \left(\mathbf{Z}_i - \mathbf{Z}_j \right). \tag{11}$$

The gradient descent operation with step size α that aims to minimize the cost function $\mathcal{L}$ at the current layer l is given by the following equations:

$$\mathbf{Z}_i^{(l+1)} = \bar{\mathbf{Z}}_i^{(l)} - \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{Z}_i}\bigg|_{\mathbf{Z}=\bar{\mathbf{Z}}^{(l)}} \tag{12}$$

$$= (1-\alpha) \sum_j \tilde{\mathbf{S}}_{ij}^{(l)} \bar{\mathbf{Z}}_j^{(l)} + \alpha \hat{\mathbf{Z}}_i^{(0)} + (1-\alpha) \left(1 - \sum_j \tilde{\mathbf{S}}_{ij}^{(l)} \right) \bar{\mathbf{Z}}_i^{(l)}. \tag{13}$$

Due to the normalization of the attention weights, i.e., $\sum_j \tilde{\mathbf{S}}_{ij} = 1$, Eq. (13) can be simplified as:

$$\mathbf{Z}^{(l+1)} = (1-\alpha) \tilde{\mathbf{S}}^{(l)} \bar{\mathbf{Z}}^{(l)} + \alpha \hat{\mathbf{Z}}^{(0)}. \tag{14}$$

Thus, Eq. (9) is proven to be equivalent to a gradient descent operation for the given graph signal denoising problem. □

Based on the above theoretical analysis, each modified Transformer layer can also be interpreted as an approximation to solving the adaptive graph signal denoising problem described by Eq. (10). In this framework, the revised fidelity term guides $\mathbf{Z}$ to be close to $\hat{\mathbf{Z}}^{(0)}$, which has been empirically proven to integrate the main information of the graph structure.

3.3. Simplifying Attention Mechanisms

Traditional Transformers mainly adopt softmax attention mechanism, given by $\mathbf{S} = \exp(\mathbf{Q}\mathbf{K}^\top/\sqrt{d})$. Due to this nonlinear similarity measure, the attention map $\mathbf{S}$ is constructed by computing the similarity between all query-key pairs, resulting in a computational complexity of $\mathcal{O}(N^2)$. This has created a computational bottleneck, limiting the application of Transformers on large graphs. Comparably, linear attention [21] is considered as an effective alternative, which restricts the computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. Specifically, purpose-designed kernels are employed to approximate the original similarity function:

$$\mathbf{S}_{ij} = \phi(\mathbf{Q}_i)\phi(\mathbf{K}_j)^\top, \quad \tilde{\mathbf{S}}_{ij} = \frac{\mathbf{S}_{ij}}{\sum_k \mathbf{S}_{ik}}, \tag{15}$$

where ϕ represents a feature mapping function that transforms the input vectors into a higher-dimensional space. Hence, each layer of the self-attention mechanism described by Eqs. (2) and (3) can be reformulated as:

$$\mathbf{Z}_i = \frac{\phi(\mathbf{Q}_i) \cdot \left(\sum_j \phi(\mathbf{K}_j)^\top \bar{\mathbf{Z}}_j \right)}{\phi(\mathbf{Q}_i) \cdot (\sum_k \phi(\mathbf{K}_k)^\top)}. \tag{16}$$

Both $\sum_j \phi(\mathbf{K}_j)^\top \bar{\mathbf{Z}}_j$ and $\sum_k \phi(\mathbf{K}_k)$ have a computational complexity of $\mathcal{O}(N)$. Notably, these calculations are performed only once for all nodes, thus the overall complexity of the modified self-attention is $\mathcal{O}(N)$.

However, linear attention mechanisms typically reduce the model's expressiveness, and kernel functions depend heavily on careful design, which may introduce computational overhead and training instability [31]. Related work [32] has mathematically proven that kernel-based linear attention usually suffers from unbounded gradients, causing unstable convergence during training:

$$\left| \frac{\partial p_{ik}}{\partial s_{ij}} \right| \leq |s_{ij}|^{-1}, \tag{17}$$

where $s_{ij} = \phi(\mathbf{Q}_i)\phi(\mathbf{K}_j)^\top$ and $p_{ik} = s_{ik} / \sum_j s_{ij}$. Since $|s_{ij}|^{-1}$ can be arbitrarily large, the gradient of kernel-based linear attention has no upper bound.

Although recent studies [9, 17, 20, 11] have integrated linear attention [31, 21] into GTs, these methods still rely on the complex attention function inherited from traditional Transformers, in which feature vectors are transformed into query and key vectors through separate linear transformations. Although more complex attention functions can capture more intricate relationships, they also introduce unnecessary computational overhead and increase the risk of overfitting. Drawing on methods developed for graph structure learning to model latent relationships between nodes [33, 34, 35], we extend linear attention to a generalized form by using node embeddings directly as inputs:

$$\tilde{\mathbf{S}} = c + g(\mathbf{Z})h(\mathbf{Z})^\top, \tag{18}$$

where $g(\cdot)$ and $h(\cdot)$ are feature transformations with the same output dimension; and $c > 0$ is a constant. To obtain non-negative attention weights and ensure bounded gradients, we further constrain the forms of $g(\cdot)$ and $h(\cdot)$ as follows:

$$g(\mathbf{Z}) = \frac{g^*(\mathbf{Z})\sqrt{c-\epsilon}}{\mathrm{Norm}(g^*(\mathbf{Z}))}, \quad h(\mathbf{Z}) = \frac{h^*(\mathbf{Z})\sqrt{c-\epsilon}}{\mathrm{Norm}(h^*(\mathbf{Z}))}, \tag{19}$$

where $\epsilon > 0$ is a small constant; $g^*(\cdot)$ and $h^*(\cdot)$ are feature transformations with the same output dimension; and $\text{Norm}(\cdot)$ denotes a normalization operation, using either the L2 norm or the Frobenius norm. Specifically, the L2 norm is computed for single nodes, while the Frobenius norm is computed over all nodes:

$$\begin{aligned} \text{L2}: &\quad \text{Norm}\left(g^*(\mathbf{Z}_i)\right) = \left\|g^*(\mathbf{Z}_i)\right\|_2, \\ \text{F}: &\quad \text{Norm}\left(g^*(\mathbf{Z})\right) = \left\|g^*(\mathbf{Z})\right\|_{\mathcal{F}}. \end{aligned} \tag{20}$$

To ensure the stability of the attention mechanism, we establish bounds on the attention weights and their gradients. The following theorem formalizes these bounds.

Theorem 2. *Under the given constraints, the attention weights have a lower bound, and the gradients have an upper bound. Specifically,*

$$s_{ij} \geq \epsilon, \qquad \left|\frac{\partial p_{ik}}{\partial s_{ij}}\right| \leq |s_{ij}|^{-1} \leq \epsilon^{-1}. \tag{21}$$

Proof. We start by demonstrating the lower bound for the attention weights:

$$\begin{aligned} s_{ij} &= c + g(\mathbf{Z}_i)h(\mathbf{Z}_j)^\top \\ &\geq c - \left|g(\mathbf{Z}_i)h(\mathbf{Z}_j)^\top\right| \\ &\geq c - \|g(\mathbf{Z}_i)\|_2 \, \|h(\mathbf{Z}_j)\|_2 \\ &\geq c - (\sqrt{c-\epsilon})^2 = \epsilon, \end{aligned} \tag{22}$$

where we have used the following steps:

- The first inequality follows from the definition $s_{ij} = c + g(\mathbf{Z}_i)h(\mathbf{Z}_j)^\top$.
- The second inequality applies the Cauchy-Schwarz Inequality.
- The final inequality uses $\|g(\mathbf{Z}_i)\|_2^2 \leq c - \epsilon$ and $\|h(\mathbf{Z}_j)\|_2^2 \leq c - \epsilon$, as stated in Eqs. (19) and (20).

Next, we show the upper bound for the gradients of the attention weights:

$$\left|\frac{\partial p_{ik}}{\partial s_{ij}}\right| \leq |s_{ij}|^{-1} \leq \epsilon^{-1}, \tag{23}$$

□

Table 1: Details of the four attention functions.

Attention Function	Number of Parameters	Complexity
cos	0	-
w-cos	d	$\mathcal{O}(Nd)$
m-cos	$(d+1)d$	$\mathcal{O}(Nd^2)$
qk-cos	$2(d+1)d$	$\mathcal{O}(2Nd^2)$

In summary, the theorem ensures that the attention mechanism remains stable by providing the upper bound on the attention weights and their gradients, thereby promoting stable convergence during training. In general cases, we set $c = 1$.

Based on extended linear attention, each layer of the global attention mechanism described by Eqs. (2) and (3) can be reformulated as:

$$\mathbf{Z}_i = \frac{c\sum_j \bar{\mathbf{Z}}_j + g(\mathbf{Z}_i)\sum_j h(\mathbf{Z}_j)^\top \bar{\mathbf{Z}}_j}{cN + g(\mathbf{Z}_i)\sum_j h(\mathbf{Z}_j)^\top}. \tag{24}$$

Based on the framework of extended linear attention, we introduce four alternative similarity functions, as we experimentally show that different graphs are suited to different attention mechanisms. We substitute the complex attention function with simpler, low-parameter alternatives: cosine similarity (cos), weighted cosine similarity (w-cos), and mapped cosine similarity (m-cos). These simplifications are mathematically represented as follows:

$$\begin{aligned} \text{cos}: &\quad \mathbf{Q}_i = \mathbf{K}_i = \mathbf{Z}_i, \\ \text{w-cos}: &\quad \mathbf{Q}_i = \mathbf{K}_i = \mathbf{Z}_i \circ \mathbf{w}^\top, \\ \text{m-cos}: &\quad \mathbf{Q}_i = \mathbf{K}_i = \mathbf{Z}_i\mathbf{W}_M + \mathbf{b}_M. \end{aligned} \tag{25}$$

In w-cos, $\mathbf{w} \in \mathbb{R}^d$ is a learnable vector that weights each dimension of the feature embeddings $\mathbf{Z}$, and $\circ$ denotes the Hadamard product. In m-cos, $\mathbf{W}_M \in \mathbb{R}^{d\times d}$ and $\mathbf{b}_M \in \mathbb{R}^d$ are a learnable projection matrix and a bias vector for feature transformation. For consistency, we refer to the traditional attention function, which computes queries and keys through separate linear transformations, as query-key cosine similarity (qk-cos). Note that m-cos is essentially a simplified version of qk-cos, i.e., $\mathbf{W}_Q = \mathbf{W}_K$ and $\mathbf{b}_Q = \mathbf{b}_K$.

We summarize and compare the parameter counts and computational complexities of the four introduced attention functions (cos, w-cos, m-cos,

qk-cos) in Table 1. We find that the simplified attention functions significantly reduce both the number of parameters and computational overhead. Our experiments will further demonstrate that these simplifications are still capable of capturing the relationships between nodes.

To compute the aggregation weights $\tilde{\mathbf{S}}_{ij}$, existing GTs typically inherit the attention mechanism from traditional Transformers, as described by Eq. (3). Due to the non-linear Softmax normalization, the attention map $\tilde{\mathbf{S}}$ is constructed by computing the similarity between all query-key pairs, resulting in $\mathcal{O}(N^2)$ computational complexity. This creates a computational bottleneck, limiting the applicability of GTs on large-scale graphs. In contrast, linear attention is considered as an effective alternative, which reduces the computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. Specifically, the attention weights $\tilde{\mathbf{S}}_{ij}$ are directly computed through the dot-product of normalized queries and keys:

$$\mathbf{Q} = \mathbf{Z}\mathbf{W}_Q, \;\; \tilde{\mathbf{Q}} = \frac{\mathbf{Q}}{\|\mathbf{Q}\|_p}, \;\; \mathbf{K} = \mathbf{Z}\mathbf{W}_K, \;\; \tilde{\mathbf{K}} = \frac{\mathbf{K}}{\|\mathbf{K}\|_p}, \tag{26}$$

$$\mathbf{S} = \tilde{\mathbf{Q}}\tilde{\mathbf{K}}^\top + \mathbf{J}, \quad \tilde{\mathbf{D}} = \mathrm{diag}\left(\tilde{\mathbf{Q}}(\tilde{\mathbf{K}}^\top \mathbf{1}) + N\mathbf{1}\right), \quad \tilde{\mathbf{S}} = \tilde{\mathbf{D}}^{-1}\mathbf{S}, \tag{27}$$

where $\|\cdot\|_p$ typically denotes the L2 norm or the Frobenius norm, $\mathbf{J}$ is an $N \times N$ all-one matrix to ensure the non-negativity of attention weights, and $\mathbf{1}$ is an N-dimensional all-one column vector to compute the diagonal matrix $\tilde{\mathbf{D}}$ for the normalization of attention weights. By introducing the linear attention mechanism, each Transformer layer described by Eq. (2) can be reformulated as:

$$\bar{\mathbf{Z}}^{(l)} = f^{(l)}\left(\mathbf{Z}^{(l)}\right), \quad \mathbf{Z}^{(l+1)} = \tilde{\mathbf{D}}^{-1}\left(\tilde{\mathbf{Q}}(\tilde{\mathbf{K}}^\top \bar{\mathbf{Z}}^{(l)}) + \mathbf{J}\bar{\mathbf{Z}}^{(l)}\right). \tag{28}$$

By changing the order of matrix operations, specifically by computing $\tilde{\mathbf{K}}^\top \bar{\mathbf{Z}}$ prior to $\tilde{\mathbf{Q}}\tilde{\mathbf{K}}^\top$, the linear attention can achieve $\mathcal{O}(N)$ complexity, thereby significantly reducing computational overhead compared to the Softmax attention.

3.4. Graph-Agnostic Linear Transformers

We integrate the above two techniques of decoupling graph structures and simplifying attention mechanisms into general GT frameworks, and name our model as Graph-Agnostic Linear Transformer (GALiT). In GALiT, the graph

structure $\mathbf{A}$ is solely utilized to approximately obtain the denoised raw node features $\hat{\mathbf{X}}$ before the training and inference stages:

$$\hat{\mathbf{X}} = \left\langle (1-\gamma)\tilde{\mathbf{A}}\hat{\mathbf{X}}^{(k)} + \gamma\mathbf{X} \right\rangle_K, \quad \hat{\mathbf{X}}^{(0)} = \mathbf{X}, \tag{29}$$

where $\tilde{\mathbf{A}}$ is the symmetric normalized adjacency matrix, $\gamma \in (0,1)$ is the trade-off weight, and K is the number of recursive iterations. The pre-processed features $\hat{\mathbf{X}}$ are cached and then guide the adaptive aggregation process within Transformers:

$$\mathbf{Z}^{(0)} = f_I(\mathbf{X}), \quad \hat{\mathbf{Z}}^{(0)} = f_I(\hat{\mathbf{X}}), \quad \bar{\mathbf{Z}}^{(l)} = f^{(l)}\left(\mathbf{Z}^{(l)}\right), \tag{30}$$

$$\mathbf{Z}^{(l+1)} = (1-\alpha)\tilde{\mathbf{D}}^{-1}\left(\tilde{\mathbf{Q}}(\tilde{\mathbf{K}}^\top\bar{\mathbf{Z}}^{(l)}) + \mathbf{J}\bar{\mathbf{Z}}^{(l)}\right) + \alpha\hat{\mathbf{Z}}^{(0)}, \tag{31}$$

where f_I denotes the feature transformation (e.g., a single-layer MLP) used to generate the initial raw feature embeddings $\mathbf{Z}^{(0)}$ and the denoised feature embeddings $\hat{\mathbf{Z}}^{(0)}$. $f^{(l)}$ can be either a shallow neural layer or an identity mapping. The matrices $\tilde{\mathbf{Q}}$, $\tilde{\mathbf{K}}$, and $\tilde{\mathbf{D}}$ are computed following the linear attention mechanism described by Eq. (26)$\sim$(27), with $\alpha \in (0,1)$ serving as the trade-off weight parameter. Additionally, we simplify the attention mechanism inherited from traditional Transformers, as described by Eq. (18), which further reduces computational overhead while still capturing the relationships between nodes. Notably, this simplification is still within the framework of linear attention described by Eq. (26)$\sim$(27).

After L layers of propagation, we obtain the node representations $\mathbf{Z}^{(L)} \in \mathbb{R}^{N\times d}$ and employ a shallow neural layer f_O to generate the predicted labels $\hat{\mathbf{Y}} \in \mathbb{R}^{N\times C}$. Subsequently, we compute the cross-entropy loss using the ground truth labels $\mathbf{Y} \in \{0,1\}^{N\times C}$ to guide the training process:

$$\hat{\mathbf{Y}} = f_O\left(\mathbf{Z}^{(L)}\right), \quad \mathcal{L}(\hat{\mathbf{Y}};\mathbf{Y}) = -\sum_{i=1}^{N}\sum_{c=1}^{C}\mathbf{Y}_{ic}\log(\hat{\mathbf{Y}}_{ic}). \tag{32}$$

By integrating the techniques of decoupling graph structures and simplifying attention mechanisms, our proposed GALiT reduces the computational complexity from $\mathcal{O}(N+E)$ to $\mathcal{O}(N)$, which surpasses existing GNNs and GTs. Simultaneously, our experiments will demonstrate that these simplifications do not compromise the model's expressiveness capacity. Therefore, we believe that the proposed methodology could establish a new technical path for the design of simplified graph encoders.

3.5. Complexity Analysis

We compare the time complexity of GALiT with other state-of-the-art efficient Graph Transformers, including NodeFormer [17] and SGFormer [11], in Table 2. While all these methods utilize linear attention to reduce the quadratic complexity of global attention, significant differences exist in how they handle graph structural information.

Both NodeFormer and SGFormer rely on graph-based regularization or message passing during the training process, which introduces a dependency on the number of edges E. Consequently, their training complexity remains $\mathcal{O}(N + E)$. In contrast, GALiT completely decouples the graph structure from the training loop. By confining the graph dependency to a one-time pre-processing step (feature denoising), the training complexity of GALiT is strictly $\mathcal{O}(N)$. As we will demonstrate in the experimental section, the cost of pre-processing is negligible compared to training time. This linear scalability with respect to the number of nodes makes GALiT significantly more efficient and memory-friendly for massive graphs where $E \gg N$.

Table 2: Complexity comparison of different Graph Transformers.

Model	Attention Complexity	Structural Dependency	Overall Training Complexity
NodeFormer	$\mathcal{O}(N)$	Edge Regularization	$\mathcal{O}(N + E)$
SGFormer	$\mathcal{O}(N)$	GNN Component	$\mathcal{O}(N + E)$
GALiT	$\mathcal{O}(N)$	**Decoupled (Pre-processing)**	$\mathcal{O}(\mathbf{N})$

3.6. Theoretical Analysis

In this section, we theoretically elucidate why the integration of simple linear attention with structural information can effectively enhance node representations on graphs.

One of the factors restricting the expressiveness of linear attention is feature diversity, which can be partly attributed to the rank of the attention matrix [22]. The non-linear attention matrix usually has full rank, thereby exhibiting diversity when aggregating features. However, achieving full rank is challenging for linear attention. Mathematically, the rank of the linear attention matrix $\mathbf{P}$ is constrained by the number of nodes N and the dimension

of embeddings d for each head:

$$\begin{aligned}\operatorname{rank}(\mathbf{P}) &= \operatorname{rank}\left(f(\mathbf{Z})g(\mathbf{Z})^{\top}\right) \\ &\leq \min\left\{\operatorname{rank}\left(f(\mathbf{Z})\right), \operatorname{rank}\left(g(\mathbf{Z})\right)\right\} \\ &\leq \min\left\{N, d\right\} = d.\end{aligned} \tag{33}$$

In graph-based tasks, N is typically much greater than d ($N \gg d$), thus the upper bound of the rank is restricted to a lower ratio, leading to the homogenization of many rows in the attention map.

The key to overcoming the low-rank problem lies in integrating structural information from graphs. Empirically, the rank of the normalized adjacency matrix from real-world graphs is often close to N. For example, in benchmark graph datasets such as Cora, CiteSeer, and PubMed, the ranks exceed 90% of the total number of nodes. Therefore, through a weighted combination of the linear attention matrix and the normalized adjacency matrix, the upper bound of rank for the resulting attention matrix approaches N, and its lower bound approaches $N - d$:

$$\begin{aligned}\operatorname{rank}\left(\alpha\mathbf{P} + (1-\alpha)\tilde{\mathbf{A}}\right) &\leq \min\left\{N, \operatorname{rank}(\mathbf{P}) + \operatorname{rank}(\tilde{\mathbf{A}})\right\} \\ &\doteq \min\left\{N, \operatorname{rank}(\mathbf{P}) + N\right\} = N, \\ \operatorname{rank}\left(\alpha\mathbf{P} + (1-\alpha)\tilde{\mathbf{A}}\right) &\geq \left|\operatorname{rank}(\mathbf{P}) - \operatorname{rank}(\tilde{\mathbf{A}})\right| \\ &\doteq \left|\operatorname{rank}(\mathbf{P}) - N\right| \geq N - d.\end{aligned} \tag{34}$$

Moreover, employing the multi-head attention mechanism can mitigate the low-rank problem, given that averaging across multiple attention matrices can raise the upper bound of rank for the resulting attention matrix:

$$\operatorname{rank}(\tilde{\mathbf{P}}) = \operatorname{rank}\left(\textstyle\sum_{h=1}^{H} \mathbf{P}^{(h)}/H\right) \leq \min\left\{N, Hd\right\}. \tag{35}$$

Notably, the above strategies incur little computational overhead. The computational complexity is $\mathcal{O}(N)$ for the global attention $\tilde{\mathbf{P}}\mathbf{Z}$ and $\mathcal{O}(E)$ for the inherent attention $\tilde{\mathbf{A}}\mathbf{Z}$, thus the overall computational complexity is $\mathcal{O}(N + E)$. The GALiT method proposed in this paper further decouples the graph structure from the Transformer, i.e., the graph structure is not used during both the training and inference phases. Therefore, the overall computational complexity is further reduced to $\mathcal{O}(N)$. It is important to note that this decoupling does not affect the theoretical analysis process

mentioned above, as the graph structure is used for information aggregation between nodes during the preprocessing stage. Hence, the integration of denoised features and the attention mechanism can be viewed as a collaborative interaction between the graph structure and the attention mechanism.

4. Experiments

In this section, we conduct a comprehensive evaluation of the proposed GALiT model on benchmark datasets, with the aim of answering the following research questions:

- **RQ1:** Does GALiT outperform the state-of-the-art methods on benchmark graphs?
- **RQ2:** What is the scalability and efficiency of GALiT on large-scale graphs?
- **RQ3:** How do the key components contribute to the performance of GALiT?
- **RQ4:** What is the sensitivity of GALiT with respect to different hyperparameters?
- **RQ5:** How well does the empirical evidence align with the theoretical justifications of GALiT?

4.1. Results on Medium-scale Graphs (RQ1)

4.1.1. Datasets

We first perform experiments on six commonly used benchmark graph datasets, which include both homophilic and heterophilic graphs. The following describes each dataset:

- `Cora`, `CiteSeer`, and `PubMed`: These are citation networks where each node represents a scientific paper, and edges represent citation links between the papers. Node features are derived from the bag-of-words representation of the documents, with the task being the classification of academic topics. These datasets are homophilic, meaning that connected nodes tend to share the same class label.

Table 3: Information for node classification datasets.

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	5,429	1,433	7
CiteSeer	3,327	4,732	3,703	6
PubMed	19,717	44,324	500	3
Squirrel	5,201	216,933	2,089	5
Chameleon	2,277	36,101	2,325	5
Actor	7,600	29,926	931	5

- `Squirrel` and `Chameleon`: These datasets are web link networks derived from Wikipedia, where nodes represent web pages, and edges represent mutual links between them. Node features capture the presence of specific nouns on the pages, and the classification task is to categorize the web pages based on their average monthly traffic. Unlike the citation networks, these are heterophilic graphs, where connected nodes are more likely to belong to different classes.

- `Actor`: This dataset represents an actor co-occurrence network, where nodes correspond to actors, and edges denote co-occurrences of actors on the same Wikipedia pages. The task is to classify actors based on their Wikipedia profiles. Like the web link networks, this dataset is heterophilic in nature.

Table 6 summarizes the details of these six datasets.

4.1.2. Baselines

We evaluate the performance of GALiT by comparing it with representative baseline methods. 1) Traditional GNNs: MLP [36], GCN [12], GraphSAGE [37], GAT [13], SGC [15], and APPNP [14]. These traditional GNNs follow the message-passing paradigm propagating information along observed graph structures. Notably, MLP can be viewed as a special case of GCN without considering graph structures. 2) Heterophilic GNNs: JKnet [38], H_2GCN [39], and GloGNN [40]. These GNN-based methods are specifically designed for heterophilic graphs, where neighboring nodes have lower similarity compared to homophilic graphs. Nevertheless, these methods still rely on observed graph structures. 3) Graph Transformers: ANS-GT [41],

Table 4: Performance comparison for node classification on homophilic graphs. Mean and standard deviation of testing accuracy (%) are listed. The best results are bolded and the second-best results are underlined.

Datasets		Cora	CiteSeer	PubMed
# Statistics	# Nodes	2,708	3,327	19,717
	# Edges	5,429	4,732	44,324
Traditional GNNs	MLP	57.8 ± 0.1	61.2 ± 0.1	73.2 ± 0.1
	GCN	81.5 ± 0.5	71.1 ± 0.4	79.2 ± 0.2
	GraphSAGE	82.1 ± 0.3	71.8 ± 0.4	79.2 ± 0.3
	GAT	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3
	SGC	80.9 ± 0.2	72.1 ± 0.3	78.5 ± 0.1
	APPNP	83.3 ± 0.4	71.7 ± 0.5	80.1 ± 0.3
Heterophilic GNNs	JKNet	81.5 ± 0.5	70.7 ± 0.9	78.7 ± 0.4
	H_2GCN	82.3 ± 0.7	71.6 ± 0.5	79.6 ± 0.5
	GloGNN	81.7 ± 0.4	72.3 ± 0.7	78.4 ± 0.5
Graph Transformers	ANS-GT	82.8 ± 1.1	71.1 ± 0.7	79.4 ± 0.9
	NAGphormer	82.3 ± 0.7	70.9 ± 0.9	78.5 ± 0.5
	NodeFormer	82.2 ± 0.9	72.5 ± 1.1	79.9 ± 1.0
	DIFFormer	83.4 ± 1.3	<u>74.6 ± 0.8</u>	78.3 ± 0.5
	SGFormer	<u>84.5 ± 0.8</u>	72.6 ± 0.2	<u>80.3 ± 0.6</u>
	GALiT	**85.1 ± 0.6**	**75.6 ± 0.8**	**80.8 ± 0.4**

NAGphormer [18], NodeFormer [17], DIFFormer [20], and SGFormer [11]. These methods incorporate the global attention mechanism of Transformers to capture latent relations for enhancing representation on graphs. They also employ strategies such as node sampling or linear attention to reduce computational overhead.

4.1.3. Settings

All datasets adhere to the widely accepted fixed splits. Specifically, `Cora`, `CiteSeer`, and `PubMed` use the fixed splits proposed in [12], with 20 nodes per class for training, 500 nodes for validation, and 1000 nodes for testing. `Squirrel` and `Chameleon` follow the splits from [42], which filters overlapped nodes in the original datasets and introduce 10 fixed splits of 48%/32%/20% nodes for training/validation/testing. `Actor` adheres to

Table 5: Performance comparison for node classification on heterophilic graphs. Mean and standard deviation of testing accuracy (%) are listed. The best results are bolded and the second-best results are underlined.

Datasets		Squirrel	Chameleon	Actor
# Statistics	# Nodes	5,201	2,277	7,600
	# Edges	216,933	36,101	29,926
Traditional GNNs	MLP	36.5 ± 1.8	36.7 ± 4.7	34.0 ± 0.5
	GCN	39.5 ± 1.5	40.9 ± 4.1	29.7 ± 0.3
	GraphSAGE	36.1 ± 2.0	37.8 ± 4.1	30.1 ± 0.7
	GAT	35.6 ± 2.1	39.2 ± 3.1	29.9 ± 0.6
	SGC	39.4 ± 1.9	38.8 ± 2.1	27.7 ± 1.0
	APPNP	35.6 ± 1.8	39.0 ± 3.2	32.1 ± 1.7
Heterophilic GNNs	JKNet	39.8 ± 1.4	39.5 ± 4.0	32.2 ± 0.9
	H_2GCN	35.1 ± 1.1	38.8 ± 3.1	34.8 ± 2.0
	GloGNN	35.1 ± 1.2	41.0 ± 3.5	36.6 ± 1.4
Graph Transformers	ANS-GT	40.9 ± 1.8	43.5 ± 3.0	36.1 ± 1.6
	NAGphormer	37.9 ± 2.1	38.8 ± 2.0	34.3 ± 0.9
	NodeFormer	38.5 ± 1.5	34.7 ± 4.1	36.9 ± 1.0
	DIFFormer	40.3 ± 0.8	44.8 ± 2.7	<u>37.9 ± 1.2</u>
	SGFormer	<u>41.8 ± 2.2</u>	<u>44.9 ± 3.9</u>	<u>37.9 ± 1.1</u>
	GALiT	**42.3 ± 1.8**	**45.7 ± 4.2**	**38.2 ± 1.1**

the splits proposed in [43], with 10 fixed splits of 48%/32%/20% nodes for training/validation/testing. For all the compared methods, we cite the reported results under these fixed splits, if available. Otherwise, we report the average accuracy and standard deviation across 10 runs under the fixed training/validation/testing splits. Details on all hyper-parameter settings are provided in the supplemental material.

4.1.4. Results

The experimental results are presented in Table 4 and 5. We observe that **GALiT significantly outperforms the traditional GNNs.** This suggests that our proposed model, by decoupling graph structures and simplifying attention mechanisms, can effectively leverage both local relational information and global latent relationships to enhance node representations. In

Table 6: Information for large-scale node classification datasets.

Dataset	Nodes	Edges	Features	Classes
ogbn-arxiv	169,343	1,166,243	128	40
pokec	1,632,803	30,622,564	65	2
Amazon2M	2,449,029	61,859,140	100	47

particular, GALiT's ability to handle complex graphs with reduced computational complexity indicates its practical potential for large-scale applications, where traditional methods might struggle due to the heavy computational overhead.

Moreover, we notice that GALiT outperforms traditional GNNs on heterophilic graphs, and **it even surpasses GNNs specifically designed for heterophily.** One possible explanation is that when graphs contain substantially more spurious edges than valid edges, their structural information becomes nearly inconsequential or challenging to leverage for downstream tasks. Instead of designing intricate mechanisms to filter the original graph structures, it might be more effective to directly reconstruct the graph topology. This approach seems to be especially beneficial in the case of noisy or sparse graphs, where traditional methods tend to lose their effectiveness due to the inherent complexity of handling irrelevant edges.

Furthermore, **compared to the state-of-the-art graph Transformers, GALiT also demonstrates impressive performance.** The comparison with ANS-GT and NAGphormer underscores that our simplified linear attention mechanism exhibits expressiveness comparable to non-linear Softmax attention. While NodeFormer, DIFFormer, and SGFormer also adopt linear attention, they retain the query-key paradigm inherent to traditional Transformers and integrate GCN to leverage graph structural information. In contrast, our model utilizes more simplified linear attention functions and decouples graph structures from Transformers. This simplification not only improves computational efficiency but also enhances GALiT's ability to generalize across a wide variety of graph types without being overly constrained by graph-specific structural biases.

4.2. Results on Large-scale Graphs (RQ2)

4.2.1. Datasets

To assess the scalability and efficiency of our model, we extend our experiments to three large-scale datasets, which pose additional challenges due to their size and complexity. These datasets include:

- `ogbn-arxiv`: This dataset is a large citation network where each node represents an Arxiv paper, and edges indicate citation relationships. The classification task is to predict the subject areas of Arxiv CS papers. This dataset is significantly larger than the benchmark datasets, with over 169,000 nodes and 1.16 million edges, making it a more complex challenge for node classification.

- `pokec`: Derived from a social network, the `pokec` dataset includes nodes that represent user profiles, and edges represent interactions or relationships between users. The goal is to predict the gender of users. The `pokec` dataset contains over 1.6 million nodes and 30.6 million edges, offering a significant test for both model scalability and computational efficiency.

- `Amazon2M`: This dataset is based on the Amazon Co-Purchasing network, where nodes represent products and edges represent frequently co-purchased items. The task is to classify products into their top-level Amazon categories. The `Amazon2M` dataset is the largest in our study, consisting of over 2.4 million nodes and more than 61 million edges, testing the limits of the model's scalability.

Table 6 summarizes the statistical details of these three large-scale datasets.

4.2.2. Baselines

Considering the applicability to large-scale graphs, we use efficient GNNs as comparison methods, including MLP [36], GCN [12], GAT [13], SGC [15], and SIGN [44]. Furthermore, we compare our method with three graph Transformers that utilize linear attention to reduce the computational complexity, namely NodeFormer [17], DIFFormer [20], and SGFormer [11].

Table 7: Performance comparison for node classification on large-scale graphs. Mean and standard deviation of testing accuracy (%) are listed. The best results are bolded and the second-best results are underlined.

Datasets	Large-scale Graphs		
	ogbn-arxiv	pokec	Amazon2M
# Nodes	169,343	1,632,803	2,449,029
# Edges	1,166,243	30,622,564	61,859,140
MLP	55.50 ± 0.23	60.04 ± 0.04	63.47 ± 0.11
GCN	71.74 ± 0.29	62.12 ± 1.39	83.88 ± 0.10
GAT	69.90 ± 0.12	62.37 ± 0.73	85.20 ± 0.31
SGC	67.79 ± 0.27	52.11 ± 0.82	81.23 ± 0.12
SIGN	70.28 ± 0.25	68.01 ± 0.25	80.97 ± 0.29
NodeFormer	59.90 ± 0.42	70.32 ± 0.45	87.85 ± 0.24
DIFFormer	72.21 ± 0.60	69.24 ± 0.76	<u>89.24 ± 0.28</u>
SGFormer	<u>72.63 ± 0.13</u>	<u>73.76 ± 0.24</u>	89.09 ± 0.10
GALiT	**73.05 ± 0.22**	**73.77 ± 0.03**	**91.52 ± 0.02**

4.2.3. Settings

For `ogbn-arxiv`, we follow the public OGB splits [45]. For `pokec`, we follow the splits used in the recent work [17], which randomly splits the nodes into 10% for training, 10% for validation, and 80% for testing in each round. For `Amazon2M`, we follow the splits used in the recent work [11], which randomly splits the nodes into 50% for training, 25% for validation, and 25% for testing in each round. Details of the hyper-parameter settings are provided in the supplemental material.

4.2.4. Results

The results are presented in Table 7. We observe that GALiT consistently outperforms the traditional GNNs. This suggests that even on large-scale graphs, the denoised raw node features can also integrate the main information of the graph structure, and the global linear attention can capture latent relationships beyond the original graph structure to enhance the node representations. Additionally, our method surpasses existing linear graph Transformers across all the datasets. This highlights the advantages of our framework, which decouples graph structures from Transformers and simplifies the attention mechanisms while maintaining or even enhancing model

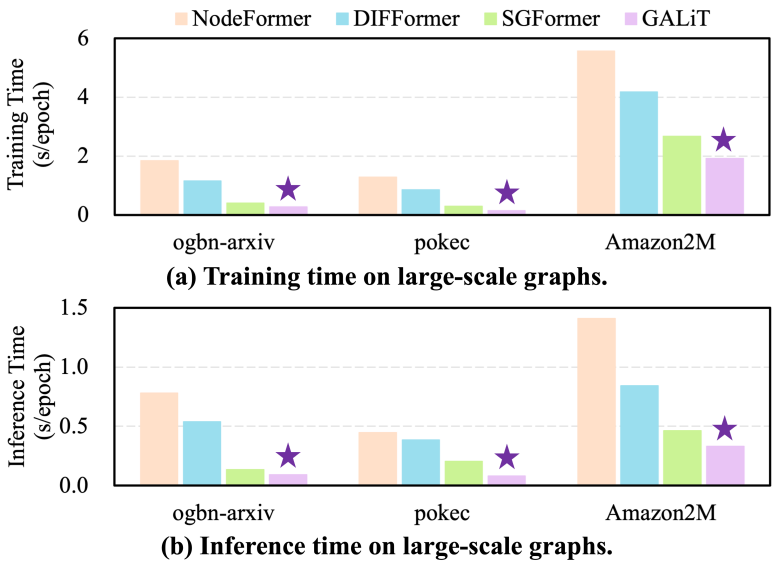

Figure 3: Performance comparison for node classification on large graphs. Training and inference times are presented.

performance.

We further clarify that the term "Graph-Agnostic" in our model primarily refers to the training and inference stages. While the graph structure is used in the pre-processing stage for feature denoising, this operation is highly efficient and incurs negligible computational overhead compared to model training. To validate this, we report the pre-processing time across different datasets in Table 8. The results show that even for massive graphs like Amazon2M, the pre-processing takes only about 1.8 seconds, confirming that the decoupling strategy effectively removes the burden of graph structures from the core learning pipeline.

Table 8: Preprocessing time across different datasets.

Dataset	ogbn-arxiv	Pokec	Amazon2M
Time (s)	0.0439 ± 0.0087	0.4472 ± 0.1342	1.8150 ± 0.2926

In Figure 3, we present the training and inference times of various competing GTs on three large-scale graphs. Notably, GALiT demonstrates significant improvements in training and inference efficiency compared to these

Table 9: Performance of GALiT with different attention functions and normalization methods. Mean and standard deviation of testing accuracy (%) are listed. The best results are bolded and the second-best results are underlined.

Dataset		CiteSeer	PubMed	Squirrel	Chameleon
w/o att		73.3 ± 0.9	80.0 ± 0.4	40.5 ± 2.4	43.0 ± 3.6
cos	L2	**75.7 ± 0.6**	80.3 ± 0.7	41.9 ± 2.3	43.3 ± 3.1
	F	75.0 ± 0.7	**80.8 ± 0.4**	41.8 ± 2.3	43.4 ± 3.2
w-cos	L2	75.5 ± 0.8	80.5 ± 0.7	<u>42.1 ± 2.1</u>	43.8 ± 3.7
	F	75.0 ± 0.7	80.6 ± 0.5	41.7 ± 2.3	43.5 ± 3.4
m-cos	L2	<u>75.6 ± 0.9</u>	79.7 ± 0.6	**42.3 ± 1.8**	**45.7 ± 4.8**
	F	74.7 ± 1.0	80.6 ± 0.4	41.6 ± 2.3	44.2 ± 3.2
qk-cos	L2	<u>75.6 ± 0.8</u>	79.6 ± 0.6	41.9 ± 2.2	<u>45.3 ± 3.9</u>
	F	75.0 ± 0.9	<u>80.7 ± 0.4</u>	41.5 ± 2.7	44.1 ± 3.1

linear attention-based GTs. Importantly, SGFormer enhances scalability by reducing the number of model layers, at the cost of diminishing the model's expressive power. In contrast, GALiT achieves a reduction in computational overhead while enhancing model performance. This allows GALiT to maintain high accuracy even on large-scale graphs.

4.3. Ablation Study (RQ3)

In our proposed GALiT framework, we refer to the traditional attention function as query-key cosine similarity (qk-cos), and introduce three simplified attention functions: cosine similarity (cos), weighted cosine similarity (w-cos), and mapped cosine similarity (m-cos). To validate the effectiveness of these simplified attention functions, we conduct experiments on four benchmark datasets: `CiteSeer`, `PubMed`, `Squirrel`, and `Chameleon`. Furthermore, to demonstrate the global linear attention mechanism can enhance node representations by leveraging denoised raw node features, we conduct experiments without the attention mechanism (i.e., $\alpha = 1$), rendering the model equivalent to directly utilizing denoised raw node features with MLPs.

The experimental results are presented in Table 9. We observe that our model using any functions of the linear attention mechanism consistently outperforms the version without attention. This validates that **leveraging global information is indeed beneficial for enhancing node repre-**

sentations on graphs. We further observe that different datasets perform better with different forms of attention. For instance, `CiteSeer` and `PubMed` are better suited with the cosine similarity, while `Squirrel` and `Chameleon` show better performance with the mapped cosine. This suggests that **the query-key paradigm is not always the best strategy across various graphs,** highlighting the adaptability and flexibility of our extended linear attention framework. Additionally, we find that using only the non-parametric cosine similarity for global attention leads to significant improvements over traditional GNNs. Notably, cosine similarity introduces much lower computational overhead compared to other forms of attention. This indicates that **simple or even non-parametric global linear attention also has powerful expressive capabilities.**

To provide guidance on selecting the optimal attention function, we further analyze the relationship between graph homophily and the performance of different attention mechanisms. As shown in Table 10, there is a strong correlation between the homophily ratio and the required complexity of the attention function. For graphs with high homophily (e.g., CiteSeer, PubMed), neighbors tend to share similar labels. In such cases, the denoised features already aggregate sufficient discriminative information, allowing the simple *cos* function to perform optimally. Conversely, for graphs with low homophily (e.g., Squirrel, Chameleon), neighbors are often dissimilar. Here, the model relies on more expressive attention functions like *m-cos* to capture complex, long-range dependencies and global semantic similarities that local aggregation might miss. This analysis provides a practical heuristic for practitioners: utilize simpler attention mechanisms for homophilic graphs and more complex ones for heterophilic graphs.

Table 10: Homophily Ratio vs. Optimal Attention Function Performance.

Dataset	**CiteSeer**	**PubMed**	**ogbn-arxiv**	**Pokec**	**Squirrel**	**Chameleon**
Homophily Ratio	~ 0.74 (High)	~ 0.80 (High)	~ 0.65 (Med)	~ 0.44 (Low)	~ 0.22 (Low)	~ 0.23 (Low)
Best Func.	**cos**	**cos**	**w-cos**	**m-cos**	**m-cos**	**m-cos**

4.4. Sensitivity Analysis (RQ4)

To answer RQ4, we conduct a sensitivity analysis of the hyper-parameters within our proposed GALiT model. Among these, the most critical hyper-parameter is γ during the graph signal denoising stage, which determines the quality of the denoised original node features. A smaller γ indicates a

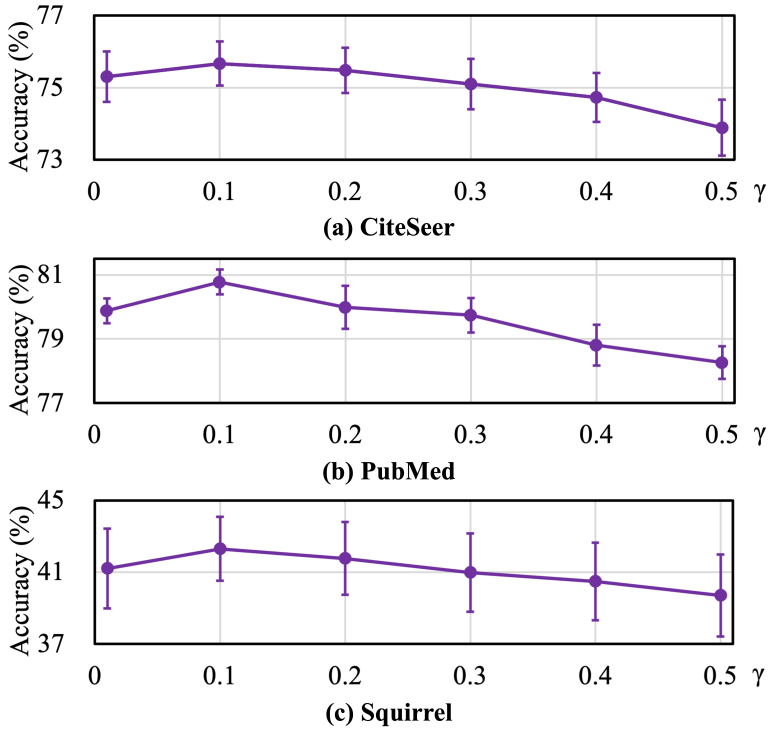

Figure 4: Performance of GALiT with different hyper-parameter γ in the graph signal denoising stage. The accuracy initially increases and then decreases with the rise of γ, highlighting a trade-off between fidelity and smoothing in the graph signal denoising process.

preference for global smoothing over local fidelity during denoising, implying a greater reliance on the graph structure. Notably, the case where $\gamma = 0$ in our analysis represents the scenario where raw node features are used directly without any graph-based denoising. As shown in the results, the performance is suboptimal at $\gamma = 0$, validating the necessity of the denoising step.We select three benchmark graphs for our experiments: `CiteSeer`, `PubMed`, and `Squirrel`.

The results depicted in Figure 4 reveal that the accuracy initially increases and then decreases with the rise of γ, highlighting a trade-off between fidelity and smoothing in denoising. This suggests that too much reliance on local fidelity can lead to underutilization of structural information. Similarly, an overemphasis on global smoothing might result in the loss of unique information. Moreover, on the heterophilic dataset `Squirrel`, optimal performance

is observed at lower γ values, unexpectedly. This dataset, where edges indicate web page links, shows that connected nodes tend to share similarities. This indicates that the proportion of edge-connected nodes of the same class does not fully capture the graph's nature.

In addition to γ, the number of recursive iterations K in the pre-processing stage is another critical hyper-parameter, analogous to the depth of receptive fields in GNNs. We analyze the sensitivity of K on three datasets, with K varying from 1 to 20. The results are reported in Table 11. We observe that performance generally improves as K increases from 1 to 10, indicating that aggregating information from a wider neighborhood is beneficial. However, for $K > 10$, performance tends to plateau or slightly degrade, likely due to the over-smoothing issue where node features become indistinguishable. Based on this analysis, we identify $K = 10$ as a robust choice that balances noise removal with feature distinctiveness across most datasets.

Table 11: Sensitivity Analysis of the number of recursive iterations K (Accuracy %).

Dataset	**K = 1**	**K = 3**	**K = 5**	**K = 10**	**K = 15**	**K = 20**
CiteSeer	69.0 ± 1.4	74.5 ± 0.7	75.2 ± 1.1	$\mathbf{75.6 \pm 0.8}$	75.4 ± 0.9	75.1 ± 0.7
PubMed	75.6 ± 1.8	80.0 ± 1.0	80.5 ± 0.7	$\mathbf{80.8 \pm 0.4}$	80.6 ± 0.5	80.3 ± 0.4
Squirrel	36.3 ± 2.7	41.2 ± 2.1	42.0 ± 2.8	$\mathbf{42.3 \pm 1.8}$	41.8 ± 2.2	41.5 ± 1.7

4.5. Theory Validation (RQ5)

As discussed in Section 4, our theoretical analysis indicates that the synergy between linear attention and graph structures can raise both the lower and upper bounds of rank for the attention matrix. Additionally, the multi-head attention mechanism can mitigate the low-rank problem, thereby augmenting representation diversity. To validate our theoretical justifications, we conduct experiments on two small-scale graphs, Cora and CiteSeer, given that the computational complexity of matrix rank is typically $O(N^3)$. We set the embedding dimension d to 256 and the number of attention heads H to 4. After training, we extract the multi-head attention matrix $\tilde{\mathbf{P}}$, the normalized adjacency matrix $\tilde{\mathbf{A}}$, and the weighted combined attention matrix $\alpha\tilde{\mathbf{P}} + (1-\alpha)\tilde{\mathbf{A}}$ from the first layer of the model. Subsequently, we employ Singular Value Decomposition to obtain the diagonal matrix containing singular values. The rank of the matrix is then determined by counting the number of singular values greater than a fixed threshold of 10^{-8}.

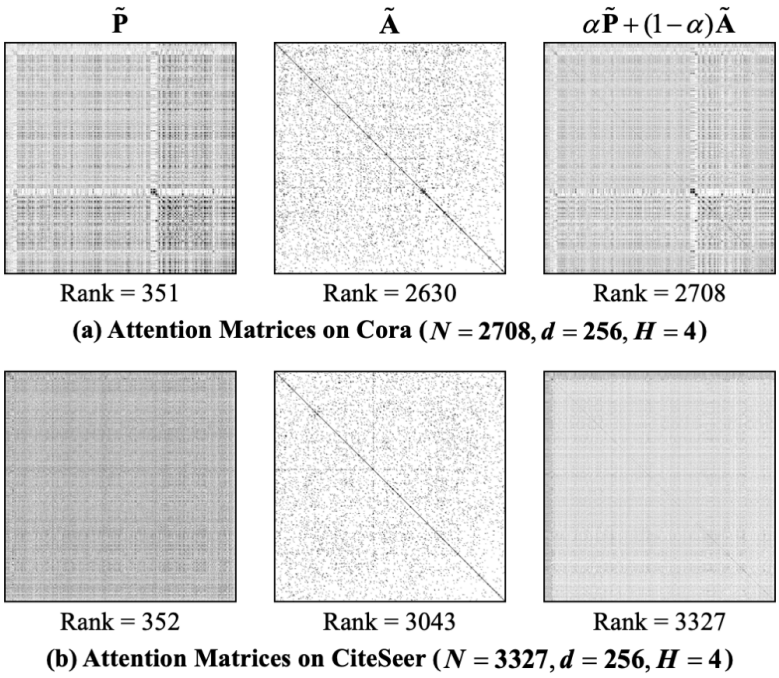

Figure 5: Visualization of the attention matrices on (a) Cora and (b) CiteSeer. Here, $\tilde{\mathbf{P}}$ represents the multi-head attention matrix, $\tilde{\mathbf{A}}$ denotes the normalized adjacency matrix, and $\alpha\tilde{\mathbf{P}} + (1-\alpha)\tilde{\mathbf{A}}$ is their weighted combination. The ranks of the matrices are also annotated.

The experimental results are presented in Figure 5, where we visualize these matrices and annotate them with their respective ranks. We observe that the normalized adjacency matrices $\tilde{\mathbf{A}}$ are nearly full rank, indicating that the original graph structures contain rich relational information. We also observe that the ranks of multi-head attention matrices $\tilde{\mathbf{P}}$ consistently exceed the rank upper bound d of single-head attention matrices. This suggests that the multi-head attention mechanism alleviates the low-rank problem, thereby enhancing the model's expressiveness to some extent. Furthermore, all the weighted combined attention matrices $\alpha\tilde{\mathbf{P}} + (1-\alpha)\tilde{\mathbf{A}}$ have full rank. This demonstrates that **the synergy between multi-head linear attention and graph structures breaks the constraints imposed by the low-rank problem**, thereby enhancing the model's representation diversity.

5. Related Work

5.1. Graph Neural Networks

Constructing GNNs with powerful expressiveness is fundamental in graph machine learning. GNNs typically employ the message-passing paradigm, recursively aggregating information from neighboring nodes to generate representations. Early models such as GCN [12] directly utilized observed structures for message passing. Subsequent models, such as GAT [13], integrated attention mechanisms into observed structures, enabling message passing influenced by attention weights. As research advanced, some models were developed to tackle challenges including over-smoothing [46], heterophily [47], imbalance [1], and out-of-distribution [48], yet they still largely rely on original graph structures. Moreover, graph structure learning [33] seeks to refine and acquire new structures for enhanced representation. However, due to computational complexity constraints, it is difficult to capture extensive latent relationships. Overall, most GNNs lean heavily on observed graph structures for their assumed reliability and relevance to downstream tasks.

5.2. Graph Transformers

Originating from natural language processing, Transformers [16] have revolutionized various domains with their unique attention mechanisms. Beyond the local message passing of GNNs, Transformers have emerged as potent graph encoders [41, 9, 49, 8, 10], capturing latent relations and refining representations through global attention. A significant challenge for traditional Transformers is the necessity to compute attention weights for all query-key pairs, leading to a computational bottleneck that limits their application on large-scale graphs. Nonetheless, recent studies have proposed linear Transformers [50], reducing the computational complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. Innovatively, several studies have integrated the linear attention mechanism into graph representation learning. NodeFormer [17] employs kernelized message passing and introduces random feature maps to approximate all-pair attention. DIFFormer [20] draws inspiration from energy constrained diffusion and introduces a simple normalized linear similarity function, achieving notable results across various tasks. SGformer [11] further underscores the potential of shallow linear Transformers in graph-based tasks through empirical and theoretical analysis. However, these methods adopt the query-key paradigm inherent to traditional Transformers, without exploring alternative forms of linear attention.

6. Conclusion

In this work, we propose the Graph-Agnostic Linear Transformer (GALiT), which is a framework of simplified Graph Transformers applicable to large-scale graphs. In GALiT, we first utilize graph structures to obtain the denoised raw node features before training. These denoised features are then used to guide the Transformers. Additionally, we simplify the attention functions inherited from traditional Transformers, which further reduces computational overhead while still capturing the relationships between nodes. Despite decoupling graph structures and simplifying attention mechanisms, experimental results indicate that GALiT achieves high efficiency while maintaining or even enhancing model performance. The efficiency and scalability of our proposed GALiT framework make it a strong candidate for practical applications in real-world graph learning tasks.

7. Acknowledgements

The research work is supported by the National Natural Science Foundation of China under Grant No. 62406307, 62576333, and 62476263, the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant No. XDB0680201. Xiang Ao is also supported by the Beijing Natural Science Foundation under Grant No. JQ25015, and the Innovation Funding of ICT, CAS under Grant No. E461060. Yang Liu is also supported by the Postdoctoral Fellowship Program of CPSF under Grant Number GZB20240761.

References

[1] Y. Liu, X. Ao, Z. Qin, J. Chi, J. Feng, H. Yang, Q. He, Pick and choose: a gnn-based imbalanced learning approach for fraud detection, in: Proceedings of the web conference 2021, 2021, pp. 3168–3177.

[2] Y. Ding, Y. Liu, Y. Ji, W. Wen, Q. He, X. Ao, Spear: A structure-preserving manipulation method for graph backdoor attacks, in: Proceedings of the ACM on Web Conference 2025, 2025, pp. 1237–1247.

[3] Q. Yuan, Y. Liu, Y. Tang, X. Chen, X. Zheng, Q. He, X. Ao, Dynamic graph learning with static relations for credit risk assessment, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 39, 2025, pp. 13133–13141.

[4] K. Li, Y. Chen, Y. Liu, J. Wang, Q. He, M. Cheng, X. Ao, Boosting the adversarial robustness of graph neural networks: An ood perspective, in: The Twelfth International Conference on Learning Representations, 2024.

[5] K. Li, Y. Liu, X. Ao, Q. He, Revisiting graph adversarial attack and defense from a data distribution perspective, in: The Eleventh International Conference on Learning Representations, 2023.

[6] Y. Liu, X. Ao, F. Feng, Y. Ma, K. Li, T.-S. Chua, Q. He, Flood: A flexible invariant learning framework for out-of-distribution generalization on graphs, in: Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining, 2023, pp. 1548–1558.

[7] Z. Guo, Y. Liu, X. Ao, Q. He, Grasp: Differentially private graph reconstruction defense with structured perturbation, in: Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2, 2025, pp. 767–777.

[8] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, T.-Y. Liu, Do transformers really perform bad for graph representation?(2021), URL https://arxiv. org/abs/2106.05234 (2021).

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

[10] D. Chen, L. O'Bray, K. Borgwardt, Structure-aware transformer for graph representation learning, in: International Conference on Machine Learning, PMLR, 2022, pp. 3469–3489.

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

[12] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).

[13] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903 (2017).

[14] J. Gasteiger, A. Bojchevski, S. Günnemann, Predict then propagate: Graph neural networks meet personalized pagerank, arXiv preprint arXiv:1810.05997 (2018).

[15] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, K. Weinberger, Simplifying graph convolutional networks, in: International conference on machine learning, PMLR, 2019, pp. 6861–6871.

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

[17] Q. Wu, W. Zhao, Z. Li, D. P. Wipf, J. Yan, Nodeformer: A scalable graph structure learning transformer for node classification, Advances in Neural Information Processing Systems 35 (2022) 27387–27401.

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

[19] C. Liu, Y. Zhan, X. Ma, L. Ding, D. Tao, J. Wu, W. Hu, Gapformer: Graph transformer with graph pooling for node classification, in: Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI-23), 2023, pp. 2196–2205.

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

[21] A. Katharopoulos, A. Vyas, N. Pappas, F. Fleuret, Transformers are rnns: Fast autoregressive transformers with linear attention, in: International conference on machine learning, PMLR, 2020, pp. 5156–5165.

[22] D. Han, X. Pan, Y. Han, S. Song, G. Huang, Flatten transformer: Vision transformer using focused linear attention, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 5961–5971.

[23] M. Chen, Z. Wei, Z. Huang, B. Ding, Y. Li, Simple and deep graph convolutional networks, in: International conference on machine learning, PMLR, 2020, pp. 1725–1735.

[24] S. Brody, U. Alon, E. Yahav, How attentive are graph attention networks?, arXiv preprint arXiv:2105.14491 (2021).

[25] H. Zhao, S. Ma, D. Zhang, Z.-H. Deng, F. Wei, Are more layers beneficial to graph transformers?, arXiv preprint arXiv:2303.00579 (2023).

[26] L. Ma, C. Lin, D. Lim, A. Romero-Soriano, P. K. Dokania, M. Coates, P. Torr, S.-N. Lim, Graph inductive biases in transformers without message passing, arXiv preprint arXiv:2305.17589 (2023).

[27] H. Shirzad, A. Velingker, B. Venkatachalam, D. J. Sutherland, A. K. Sinop, Exphormer: Sparse transformers for graphs, arXiv preprint arXiv:2303.06147 (2023).

[28] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains, IEEE signal processing magazine 30 (3) (2013) 83–98.

[29] Y. Ma, X. Liu, T. Zhao, Y. Liu, J. Tang, N. Shah, A unified view on graph neural networks as graph signal denoising, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 1202–1211.

[30] M. Zhu, X. Wang, C. Shi, H. Ji, P. Cui, Interpreting and unifying graph neural networks with an optimization framework, in: Proceedings of the Web Conference 2021, 2021, pp. 1215–1226.

[31] Z. Shen, M. Zhang, H. Zhao, S. Yi, H. Li, Efficient attention: Attention with linear complexities, in: Proceedings of the IEEE/CVF winter conference on applications of computer vision, 2021, pp. 3531–3539.

[32] Z. Qin, X. Han, W. Sun, D. Li, L. Kong, N. Barnes, Y. Zhong, The devil in linear transformer, arXiv preprint arXiv:2210.10340 (2022).

[33] Y. Chen, L. Wu, M. Zaki, Iterative deep graph learning for graph neural networks: Better and robust node embeddings, Advances in neural information processing systems 33 (2020) 19314–19326.

[34] Y. Liu, Y. Zheng, D. Zhang, H. Chen, H. Peng, S. Pan, Towards unsupervised deep graph structure learning, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 1392–1403.

[35] W. Zhao, Q. Wu, C. Yang, J. Yan, Graphglow: Universal and generalizable structure learning for graph neural networks, arXiv preprint arXiv:2306.11264 (2023).

[36] S. K. Pal, S. Mitra, Multilayer perceptron, fuzzy sets, classifiaction (1992).

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

[38] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, S. Jegelka, Representation learning on graphs with jumping knowledge networks, in: International conference on machine learning, PMLR, 2018, pp. 5453–5462.

[39] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, D. Koutra, Beyond homophily in graph neural networks: Current limitations and effective designs, Advances in neural information processing systems 33 (2020) 7793–7804.

[40] X. Li, R. Zhu, Y. Cheng, C. Shan, S. Luo, D. Li, W. Qian, Finding global homophily in graph neural networks when meeting heterophily, in: International Conference on Machine Learning, PMLR, 2022, pp. 13242–13256.

[41] Z. Zhang, Q. Liu, Q. Hu, C.-K. Lee, Hierarchical graph transformer with adaptive node sampling, Advances in Neural Information Processing Systems 35 (2022) 21171–21183.

[42] O. Platonov, D. Kuznedelev, M. Diskin, A. Babenko, L. Prokhorenkova, A critical look at the evaluation of gnns under heterophily: are we really making progress?, arXiv preprint arXiv:2302.11640 (2023).

[43] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, B. Yang, Geom-gcn: Geometric graph convolutional networks, arXiv preprint arXiv:2002.05287 (2020).

[44] F. Frasca, E. Rossi, D. Eynard, B. Chamberlain, M. Bronstein, F. Monti, Sign: Scalable inception graph neural networks, arXiv preprint arXiv:2004.11198 (2020).

[45] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, J. Leskovec, Open graph benchmark: Datasets for machine learning on graphs, Advances in neural information processing systems 33 (2020) 22118–22133.

[46] K. Zhou, X. Huang, Y. Li, D. Zha, R. Chen, X. Hu, Towards deeper graph neural networks with differentiable group normalization, Advances in neural information processing systems 33 (2020) 4917–4928.

[47] Y. Liu, X. Ao, F. Feng, Q. He, Ud-gnn: Uncertainty-aware debiased training on semi-homophilous graphs, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 1131–1140.

[48] S. Gui, X. Li, L. Wang, S. Ji, Good: A graph out-of-distribution benchmark, Advances in Neural Information Processing Systems 35 (2022) 2059–2073.

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

[50] H. Wu, J. Wu, J. Xu, J. Wang, M. Long, Flowformer: Linearizing transformers with conservation flows, arXiv preprint arXiv:2202.06258 (2022).