KDLGT: A Linear Graph Transformer Framework via Kernel Decomposition Approach

Anonymous Author(s)

Abstract

In recent years, graph Transformers (GTs) have been demonstrated as a robust architecture for a wide range of graph learning tasks. However, the quadratic complexity of GTs limits their scalability on large-scale data, in comparison to Graph Neural Networks (GNNs). In this work, we propose the Kernel Decomposition Linear Graph Transformer (KDLGT), an accelerating framework for building scalable and powerful GTs. KDLGT employs the kernel decomposition approach to rearrange the order of matrix multiplication, thereby reducing complexity to linear. Additionally, it categorizes GTs into three distinct types and provides tailored accelerating methods for each category to encompass all types of GTs. Furthermore, we provide a theoretical analysis of the performance gap between KDLGT and self-attention to ensure its effectiveness. Under this framework, we select two representative GTs to design our models. Experiments on both real-world and synthetic datasets indicate that KDLGT not only achieves state-of-the-art performance on various datasets but also reaches an acceleration ratio of approximately 10 on graphs of certain sizes.

1 Introduction

Recent years have seen significant advancements in the field of graph learning, with notable successes across a variety of domains including social networks [Li *et al.*, 2021; Zhong *et al.*, 2020], molecular graphs [Huang *et al.*, 2020; Wang *et al.*, 2021], and knowledge graphs [Liu *et al.*, 2021; Yasunaga *et al.*, 2021]. One of the key approaches in this field is Graph Neural Networks (GNNs), which have been widely adopted as a powerful embedding approach for various graph learning tasks [Kipf and Welling, 2017; Veličković *et al.*, 2018; Hamilton *et al.*, 2017]. The foundation of GNNs is the local sparse message-passing mechanism, which enables the nodes on the graphs to iteratively exchange messages through the edges connecting them. However, the limitations of the message-passing mechanism have become increasingly apparent in recent research [Xu *et al.*, 2019; Morris *et al.*, 2019; Maron *et al.*, 2019], leading to a series of works [Ying *et al.*, 2021; Zhang *et al.*, 2020; Chen *et al.*, 2022] that have turned to Transformer architectures[Vaswani *et al.*, 2017] in pursuit of new breakthroughs.

Graph Transformers (GTs) represent a successful endeavor to deploy Transformer architectures to graph data. By enabling nodes to attend to all other nodes within the graphs, GTs encode graph structures as a *soft inductive bias*, rather than the hard-coded message-passing approach. In contrast to GNNs, GTs utilize absolute and relative positional encodings (APEs and RPEs) to characterize graph topological structures, viewing the graphs as complete entities and enabling long-range interactions for nodes. This overcomes limitations inherent in the message-passing paradigm, such as limited expressiveness [Xu *et al.*, 2019], over-smoothing [Alon and Yahav, 2021], and over-squashing [Alon and Yahav, 2021] issues. However, despite these achievements, there remain a plethora of challenges to be addressed in this area of research.

One of the most significant challenges faced by GTs is the poor scalability. This is due to the global attention mechanism, which results in quadratic time and memory complexity with respect to the number of nodes in the graph. This problem is particularly pronounced when utilizing GTs on datasets consisting of larger graphs, such as citation and social network graphs, as limited GPU memory and excessive running time impede their performance. Consequently, in the application scenarios, GTs do not demonstrate a significant advantage over GNNs.

In this work, we present the Kernel Decomposition Linear Graph Transformer (KDLGT), an accelerating framework for building scalable and powerful GTs. Unlike previous model-specific approaches, KDLGT is model-agnostic and aims to provide a general solution for accelerating all GT models. To achieve this, we employ the kernel decomposition approach, which rearranges the matrix multiplication order of self-attention by designing a kernel function decomposition for the *softmax* function. Under this approach, the way to deal with RPE matrices becomes the key to reducing time complexity. Inspired by the fact that the RPE matrices of undirected graphs are symmetric and can be decomposed into products of low-dimensional matrices, we categorize RPE matrices into three types, which cover the most commonly used RPEs. Using KDLGT, we select two representative RPEs (such as Shortest Path Distance) and design our models. Additionally, we provide a theoretical analysis of

the difference between the KDLGT framework and the traditional self-attention method, and prove that this gap can be effectively bounded.

We conduct experiments on 10 real-world datasets to demonstrate the superior performance of our proposed KDLGT framework. Additionally, we evaluate KDLGT on a series of synthetic graphs with varying scales to verify its efficiency. The experimental results show that KDLGT not only improves learning efficiency significantly but also preserves precision performance effectively and achieves state-of-the-art results on a variety of datasets. Furthermore, as anticipated, the acceleration ratio increases as the graph size increases, indicating the strong scalability of KDLGT. In particular, KDLGT can achieve an acceleration ratio of approximately 10 on graphs of certain sizes. The contributions of this paper are listed as follows:

- We propose the KDLGT framework, which successfully reduces the quadratic complexity of GTs to linear and improves the scalability of GTs greatly.
- We provide a tight upper bound of the difference gap between KDLGT and GTs theoretically to illustrate KDLGT is a well-defined approximation of GTs.
- We conduct experiments on both real-world and synthetic datasets. The experimental results indicate that KDLGT can significantly improve the learning efficiency while preserving precision performance of GTs well.

2 Related Works

2.1 Fast Transformers

There have been a lot of works attempting to improve the efficiency of Transformer models. During the earliest period, researchers tend to restrict the context of self-attention to predefined, fixed patterns, thus limiting the size of the attention matrix and computational complexity. One example of this approach is the chunking paradigm, which involves dividing the input sequence into fixed blocks and considering the local receptive field of each block [Parmar *et al.*, 2018; Qiu *et al.*, 2020]. Another approach is to limit attention to certain fixed intervals. Models such as the Sparse Transformer [Child *et al.*, 2019] and Longformer [Beltagy *et al.*, 2020] utilize stridden and dilated context windows for attention.

In parallel, another line, the low-rank method, focuses on optimizing the self-attention architecture by approximating the self-attention matrix [Wang *et al.*, 2020]. The primary objective is to reduce the computational complexity of matrix multiplication from N^2 to kN, where N is the number of tokens and k is a constant dependent on the specific model. One notable example is Linformer [Wang *et al.*, 2020], which shrinks the length dimension of the keys and values to a lower-dimensional representation. Besides, by treating self-attention as kernel functions, various low-rank methods can be developed through kernelization approaches [Katharopoulos *et al.*, 2020; Choromanski *et al.*, 2020; Peng *et al.*, 2020], which adopt an efficient kernelized reconstruction of the self-attention matrix, thereby avoiding computing N^2 matrices.

2.2 Graph Transformers

The initial attempt to incorporate attention mechanism into graph-based models can be traced back to the Graph Attention Networks (GAT) architecture [Veličković *et al.*, 2018], which only considers the weights between nodes and one-hop neighbors on the graphs. With the development of Transformer architectures, it is found that it is effective to adopt the global receptive field of the Transformers on the graphs [Ying *et al.*, 2021; Kreuzer *et al.*, 2021; Chen *et al.*, 2022; Zhang *et al.*, 2020; Mialon *et al.*, 2021]. Different from the Transformer architectures applied to sequences, the design of graph Transformers emphasizes the use of positional encodings to capture the topological signal of the graphs. Without such encodings, the Transformer can only operate on the fully-connected graphs.

SAN [Kreuzer *et al.*, 2021] adopts Laplacian positional encodings for the nodes and combines two types of attention mechanism, one for virtual fully-connected graphs while another for real graph edges. Graphormer [Ying *et al.*, 2021] incorporates three different positional encodings, namely centrality, spatial and edge encoding respectively, combined with dense attention architecture. Besides, it is also the first time to introduce pair-wise graph distances to project RPEs in graph Transformers. Further, GraphiT [Mialon *et al.*, 2021] introduces RPEs based on diffusion kernels. GraphTrans [Wu *et al.*, 2021] proposes the first hybrid architecture by stacking Message Passing Neural Network and Transformer layers. Later, SAT [Chen *et al.*, 2022] proposes subtree and subgraph extractors to extract structural features on the graphs, and then uses the similarity scores between features to define positional encodings.

3 Preliminaries

In this section, we recap the preliminaries in self-attention and graph Transformers.

Vaswani Self-Attention

The self-attention module is the key component of the Transformer architectures. It can be represented as the following formulation:

$$\mathrm{Att}\left(\mathbf{Q},\mathbf{K},\mathbf{V}\right)=\mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{T}}{\sqrt{d}}\right)\mathbf{V}, \tag{1}$$

where $\mathbf{Q},\mathbf{K},\mathbf{V}\in\mathbb{R}^{N\times d}$, N and d denote the length of sequence and embedding dimension, respectively.

Further, after organizing the above formulation, we have:

$$\mathrm{Att}(\mathbf{Q},\mathbf{K},\mathbf{V})=\mathbf{D}^{-1}\mathbf{A}\mathbf{V}, \tag{2}$$

$$\mathbf{A}=\exp\left(\frac{\mathbf{Q}\mathbf{K}^{T}}{\sqrt{d}}\right), \tag{3}$$

$$\mathbf{D}=\mathrm{diag}(\mathbf{A}\mathbf{1}_{N}), \tag{4}$$

where $\mathbf{1}_N\in\mathbb{R}^N$ is an all-one vector. In the following discussions, we will ignore the constant $\sqrt{d}$, since we can simply set $\mathbf{Q}'=\mathbf{Q}/\sqrt{d}$ to replace $\mathbf{Q}$. It can be noticed that the multiplication complexity between $\mathbf{D}^{-1}$ and $\mathbf{AV}$ is $O(Nd)$, while the computation complexity of $\mathbf{D},\mathbf{A}$ and $\mathbf{AV}$ is $O(N^2)$, $O(N^2d)$ and $O(N^2d)$, respectively.

Graph Transformers

Compared with sequence data, graph data has rich structure features. In GNNs, the edge-based message passing methods are usually used to describe the graph structures. While for graph Transformers, positional encodings are usually used to construct an RPE matrix to describe structures. In general, it is formulated as:

$$\text{Att}(\mathbf{Q},\mathbf{K},\mathbf{V},\mathbf{B}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}} + \mathbf{B}\right)\mathbf{V}, \quad (5)$$

where $\mathbf{B} \in \mathbb{R}^{N\times N}$ is RPE matrix. (For preventing ambiguity, in the sequel, N represents the number of nodes in the graph.)

4 Methodology

In this section, we introduce KDLGT acceleration framework and our models in detail. First, we begin by summarizing the effective accelerating approaches for Transformers on sequential data.

4.1 Kernel Decomposition Approach

Most existing methods for accelerating self-attention adopt the kernel decomposition approach. The core idea is to construct a kernel function ϕ to rearrange the order of matrix multiplication and nonlinear function exp in (3), which can be formulated as:

$$\mathbf{A} = \exp(\mathbf{Q}\mathbf{K}^T) \approx \phi(\mathbf{Q})\phi(\mathbf{K})^T =: \mathbf{Q}'(\mathbf{K}')^T, \quad (6)$$

where $\mathbf{Q}', \mathbf{K}' \in \mathbb{R}^{N\times r}$, r is the new embedding dimension. In this way, the computation of attention score matrix $\mathbf{A}$ can be avoided. Instead, we can first compute the multiplication of $(\mathbf{K}')^T$ and $\mathbf{V}$, and then compute the result between $\mathbf{Q}'$ and $(\mathbf{K}')^T\mathbf{V}$. The time complexity of the two steps is both $O(Nrd)$. For (4), similarly, we can reverse the matrix multiplication order by letting $(\mathbf{K}')^T\mathbf{1}_N$ first. In specific, we can approximate Vaswani self-attention as followings:

$$\hat{\text{Att}}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \hat{\mathbf{D}}^{-1}(\mathbf{Q}'((\mathbf{K}')^T\mathbf{V})), \quad (7)$$

$$\hat{\mathbf{D}} = \text{diag}(\mathbf{Q}'((\mathbf{K}')^T\mathbf{1}_N)). \quad (8)$$

The time complexity is reduced from $O(N^2d)$ to $O(Nrd)$.

Let $\kappa(\mathbf{x},\mathbf{y}) = \exp(\mathbf{x}^T\mathbf{y}) \approx \phi(\mathbf{x})^T\phi(\mathbf{y})$, where $\mathbf{x},\mathbf{y} \in \mathbb{R}^d$. It can be noticed that the most important component in the kernel decomposition approach is the design of kernel function ϕ. In this work, we adopt the following well-defined function ϕ:

$$\phi(\mathbf{x}) := \frac{1}{\sqrt{r}}\exp\left(-\frac{||\mathbf{x}||^2}{2}\right)\left(\exp(\mathbf{w}_1^T\mathbf{x}),\ldots,\exp(\mathbf{w}_r^T\mathbf{x})\right), \quad (9)$$

where $\mathbf{w}_i \sim \mathcal{N}(0,\mathbf{I}_d)$, $r \le d$ is a sampling number as well as embedding dimension. We do not discuss here the validity and stability of function ϕ and refer readers to [Choromanski *et al.*, 2020] for further details. Specifically, we adopt Gram-Schmidt orthogonalization on $\mathbf{w}_i$ in experiments to ensure that they are linearly independent towards each other for the validity of sampling.

4.2 Kernel Decomposition Linear Graph Transformer

Further, when we turn our perspective to GT, the RPE matrix $\mathbf{B}$ becomes the most significant difference between GTs and Transformers. Therefore, we focus on the RPE matrix $\mathbf{B}$ and provide a detailed analysis.

Here, we only consider undirected graphs. In this scenario, as the graph structure is symmetric, $\mathbf{B}$ should also be a symmetric matrix. Suppose that the rank of $\mathbf{B}$ is $d' \le N$, and its d' nonzero eigenvalues are $\lambda_i, i = 1,\ldots,d'$, then we can decompose $\mathbf{B}$ as follows:

$$\begin{aligned}\mathbf{B} =& \mathbf{U}\text{diag}([\lambda_1,\ldots,\lambda_{d'},0,\ldots,0])\mathbf{U}^T\\ =& \mathbf{U}\text{diag}([\lambda_1,\ldots,\lambda_{d'},0,\ldots,0])[:,0:d'](\mathbf{U}\text{diag}([1,\ldots,1,0,\ldots,0])[:,0:d'])^T.\end{aligned}$$

Let $\mathbf{B}_q = \mathbf{U}\text{diag}([\lambda_1,\ldots,\lambda_{d'},0,\ldots,0])[:,0:d']$, $\mathbf{B}_k = \mathbf{U}\text{diag}([1,\ldots,1,0,\ldots,0])[:,0:d']$, where $\mathbf{B}_q,\mathbf{B}_k \in \mathbb{R}^{L\times d'}$,$[:,i:j]$ represents the i to j rows of the second dimension, then we have $\mathbf{B} = \mathbf{B}_q\mathbf{B}_k^T$.

Inspired by this, we categorize RPE matrices into the following three types for discussion and propose the Kernel Decomposition Linear Graph Transformer (KDLGT) accelerating framework.

Multiplication Decomposition

$\mathbf{B} = \mathbf{B}_q\mathbf{B}_k^T$, where $\mathbf{B}_q,\mathbf{B}_k \in \mathbb{R}^{N\times d'}$ and $d' \ll N$, we have:

$$\langle\mathbf{Q}_i,\mathbf{K}_j\rangle + \langle\mathbf{B}_q^{(i)},\mathbf{B}_k^{(j)}\rangle = \langle[\mathbf{Q}_i,\mathbf{B}_q^{(i)}],[\mathbf{K}_j,\mathbf{B}_k^{(j)}]\rangle.$$

Therefore, $\mathbf{Q}\mathbf{K}^T + \mathbf{B} = [\mathbf{Q},\mathbf{B}_q][\mathbf{K},\mathbf{B}_k]^T$, where $[,]$ represents concatenation operator. If we view $[\mathbf{Q},\mathbf{B}_q]$ and $[\mathbf{K},\mathbf{B}_k]$ as $\mathbf{Q}$ and $\mathbf{K}$ in (6), then it can be accelerated by kernel decomposition approach, and the complexity is $O(N(d+d')r)$.

Addition Decomposition

$\mathbf{B}_{ij} = \mathbf{b}_i + \mathbf{b}_j$, $\mathbf{b} \in \mathbb{R}^N$, we have:

$$\langle\mathbf{Q}_i,\mathbf{K}_j\rangle + \mathbf{b}_i + \mathbf{b}_j = \langle[\mathbf{Q}_i,\mathbf{b}_i,1],[\mathbf{K}_j,1,\mathbf{b}_j]\rangle.$$

Thus $\mathbf{Q}\mathbf{K}^T + \mathbf{B} = [\mathbf{Q},\mathbf{b},\mathbf{1}_N][\mathbf{K},\mathbf{1}_N,\mathbf{b}]^T$. Same as above, the complexity is $O(N(d+2)r)$ using the kernel decomposition approach.

Ineffective Decomposition

Suppose the rank of $\mathbf{B}$ is $d' = O(N)$, in this case, there is no direct effective acceleration method, such as the shortest path distance (SPD). However, we can still design approximate schemes to replace this type of RPE matrix (See subsection 4.3).

4.3 Our Models

As shown in Figure 1, we propose two types of models suitable for different decomposition scenarios under the KDLGT framework.

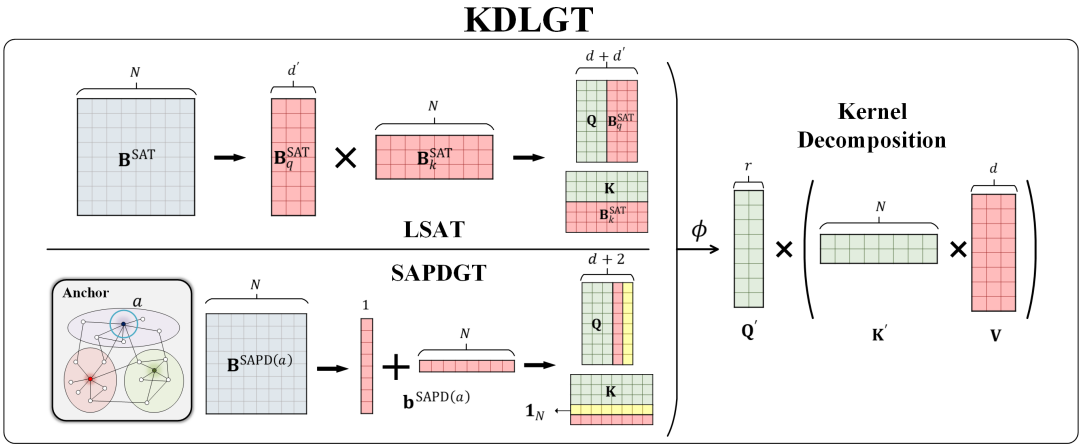

Figure 1: Illustration of the accelerating procedure of the KDLGT framework. The top and bottom of the left side represent the LSAT and SAPDGT modules, which are of the multiplication decomposition and addition decomposition types, respectively. The right side represents the rearrange of matrix multiplication order in kernel decomposition approach.

Linear Structure-Aware Transformer

There are lots of designs of RPE that match multiplication decomposition. We take Structure-Aware Transformer (SAT) [Chen *et al.*, 2022] as an example, which summarizes a series of multiplication-decomposition RPEs. Specifically, the RPE matrix can be represented as:

$$\mathbf{B}_{ij}^{\mathrm{SAT}} = \kappa_{\mathrm{SAT}}(\varphi(v_i, G), \varphi(v_j, G)), \tag{10}$$

$$\kappa_{\mathrm{SAT}}(\mathbf{x}, \mathbf{x}') = \langle \mathbf{W}_q \mathbf{x}, \mathbf{W}_k \mathbf{x}' \rangle / \sqrt{d}, \tag{11}$$

where $\mathbf{W}_q, \mathbf{W}_k \in \mathbb{R}^{d \times d}$ are parameter matrices and $\varphi(v, G)$ is a structure extractor that extracts vector representations of some subgraph centered at v in the graph G. The structure extractor here includes k-subtree GNN extractor and k-subgraph GNN extractor, which describe the structural information of different granularities, respectively.

It can be easily noticed that:

$$\mathbf{B}^{\mathrm{SAT}} = \mathbf{S}\mathbf{W}_q(\mathbf{S}\mathbf{W}_k)^T / \sqrt{d}, \tag{12}$$

where $\mathbf{S} \in \mathbb{R}^{N \times d}, \mathbf{S}_i = \varphi(v_i, G)$. Then, let $\mathbf{B}_q^{\mathrm{SAT}} = \mathbf{S}\mathbf{W}_q$, $\mathbf{B}_k^{\mathrm{SAT}} = \mathbf{S}\mathbf{W}_k / \sqrt{d}$, we have:

$$\mathbf{B}^{\mathrm{SAT}} = \mathbf{B}_q^{\mathrm{SAT}}(\mathbf{B}_k^{\mathrm{SAT}})^T. \tag{13}$$

It is clear that the SAT model can be accelerated under the multiplication decomposition framework. Additionally, other similar RPE designs can also be implemented within this framework.

Shortest Anchor Path Distance Graph Transformer

Shortest Path Distance (SPD) is a commonly used type of RPE in GTs. However, unfortunately, in most cases, the rank of the SPD RPE matrix is quite high, which makes it difficult and meaningless to decompose (ineffective decomposition). To address this issue, we propose a new type of distance, Shortest Anchor Path Distance (SAPD) to approximate SPD distance, which can be easily decomposed.

The general idea is that we only focus on the distances between anchor nodes and nodes rather than recording the distances between all pairs of nodes on the graph, thereby accelerating the computation. Anchor nodes, as a kind of coordinates, can re-characterize the distance relationship between nodes on the graph. Take arbitrary node $a \in V$ as anchor node, we define SAPD induced by a as followings:

$$d_a(v_i, v_j) = \frac{1}{2} d(v_i, a) + \frac{1}{2} d(a, v_j), \tag{14}$$

where $v_i, v_j \in V, 1 \le i, j \le N$, $d(\cdot, \cdot)$ denotes SPD.

In order to ensure rationality, we adopt linear transformation f instead of the embedding method similar to SPD for encoding RPE:

$$\mathbf{B}_{ij}^{\mathrm{SAPD}(a)} = f(d_a(v_i, v_j)) = \frac{1}{2} f(d(v_i, a)) + \frac{1}{2} f(d(a, v_j)). \tag{15}$$

In experiments, we set $f(x) = cx$, where $c \in \mathbb{R}$ is a learnable parameter. Considering that the farther the distance is, the lower the weight should be, we set $c < 0$.

Let $\mathbf{b}^{\mathrm{SAPD}(a)} = [\frac{1}{2} f(d(v_i, a))]_{i=1}^N$, then we have:

$$\mathbf{B}_{ij}^{\mathrm{SAPD}(a)} = \mathbf{b}_i^{\mathrm{SAPD}(a)} + \mathbf{b}_j^{\mathrm{SAPD}(a)}. \tag{16}$$

Therefore, $\mathbf{B}^{\mathrm{SAPD}(a)}$ can do addition decomposition and thus accelerate. In practice, we select $K \ll N$ nodes on the graph as anchor nodes and pool the embedding about different anchor nodes to get node features. The overall time complexity is $O(NK(d+2)r)$, which is still linear.

In order to ensure that the anchor nodes are evenly distributed on the graph, we adopt a greedy algorithm[Pavan and Pelillo, 2006] with time complexity of $O(N)$ to solve the k-dominant set on the graph as the anchor nodes set, where k is a receptive field hyperparameter. In this way, we can ensure that there is at least one anchor node in the k-hop neighborhood of each node and the absolute error between SPD and SAPD is within k, that is:

$$|d(v_i, v_j) - d_a(v_i, v_j)| \le k. \tag{17}$$

5 Theoretical Analysis

In order to ensure the effectiveness of KDLGT, we analyze the difference gap in the attention matrix distribution of Vaswani self-attention and the kernel decomposition method, draw the following conclusion and make a detailed proof.

Lemma 1. *Let*

$$P(\mathbf{Q}_i, \mathbf{K}_j) = \frac{\mathbb{E}[\kappa(\mathbf{Q}_i, \mathbf{K}_j)]}{\mathbb{E}[\sum_{k=1}^{N} \kappa(\mathbf{Q}_i, \mathbf{K}_k)]},$$
$$P'(\mathbf{Q}_i, \mathbf{K}_j) = \frac{\mathbb{E}[\phi(\mathbf{Q}_i)^T \phi(\mathbf{K}_j)]}{\mathbb{E}[\sum_{k=1}^{N} \phi(\mathbf{Q}_i)^T \phi(\mathbf{K}_k)]}.$$

Then we have:

$$\frac{1 - c^{\max}}{1 + c^{\max}} \leq \frac{P'(\mathbf{Q}_i, \mathbf{K}_j)}{P(\mathbf{Q}_i, \mathbf{K}_j)} \leq \frac{1 + c^{\max}}{1 - c^{\max}},$$

in which $c_{ij} = \sqrt{\frac{1}{r}(\exp(\|\mathbf{Q}_i + \mathbf{K}_j\|^2) - 1)}$*, while* $c^{\max} = \max\{c_{ij}\}_{i,j=1}^{N}$.

Proof. For simplicity, here we note $\kappa'(\mathbf{Q}_i, \mathbf{K}_j) = \phi(\mathbf{Q}_i)^T \phi(\mathbf{K}_j)$. According to Lemma 2 in [Choromanski *et al.*, 2020], we have:

$$\mathrm{MSE}^{\kappa,\kappa'}(\mathbf{Q}_i, \mathbf{K}_j) = (c_{ij}\kappa(\mathbf{Q}_i, \mathbf{K}_j))^2, \tag{18}$$

where $\mathrm{MSE}^{\kappa,\kappa'}(\mathbf{Q}_i, \mathbf{K}_j) = \mathbb{E}[(\kappa(\mathbf{Q}_i, \mathbf{K}_j) - \kappa'(\mathbf{Q}_i, \mathbf{K}_j))^2]$. Since for any random variable $\mathbf{X}$, $\mathbb{E}(\mathbf{X}^2) = \mathbb{E}^2(\mathbf{X}) + \mathrm{Var}(\mathbf{X})$, $\mathbb{E}[\mathbf{X}] \leq \sqrt{\mathbb{E}[\mathbf{X}^2]}$ holds, we have:

$$\mathbb{E}[\|\kappa(\mathbf{Q}_i, \mathbf{K}_j) - \kappa'(\mathbf{Q}_i, \mathbf{K}_j)\|] \leq c_{ij}\kappa(\mathbf{Q}_i, \mathbf{K}_j). \tag{19}$$

Expanding and rearranging (19), we can derive:

$$\begin{aligned}(1 - c_{ij})\mathbb{E}[\kappa(\mathbf{Q}_i, \mathbf{K}_j)] \leq & \mathbb{E}[\kappa'(\mathbf{Q}_i, \mathbf{K}_j)] \\ \leq & (1 + c_{ij})\mathbb{E}[\kappa(\mathbf{Q}_i, \mathbf{K}_j)]\end{aligned} \tag{20}$$

Let $c_i^{\max} = \max\{c_{ij}\}_{j=1}^{N}$. Here, we assume that $c_{ij} \in [0, 1)$. Moreover, we obtain:

$$\frac{1 - c_{ij}}{1 + c_i^{\max}} \leq \frac{P'(\mathbf{Q}_i, \mathbf{K}_j)}{P(\mathbf{Q}_i, \mathbf{K}_j)} \leq \frac{1 + c_{ij}}{1 - c_i^{\max}}. \tag{21}$$

Finally, after further scaling, we have:

$$\frac{1 - c^{\max}}{1 + c^{\max}} \leq \frac{P'(\mathbf{Q}_i, \mathbf{K}_j)}{P(\mathbf{Q}_i, \mathbf{K}_j)} \leq \frac{1 + c^{\max}}{1 - c^{\max}}. \tag{22}$$

□

It can be easily verified that $P'(\mathbf{Q}_i, \mathbf{K}_j) = P(\mathbf{Q}_i, \mathbf{K}_j)$ for $\forall i, j \in \{1, 2, \ldots, N\}$ if $c^{\max} = 0$, which means that the distance of Vaswani and approximated self-attention matrix distributions is upper-bounded by the constant $c^{\max}$. Besides, this upper bound increases monotonically with $c^{\max}$. It is worth noting that we have assumed that the range of c_{ij} is $[0, 1)$ in the proof. This assumption can be validated from an experimental point of view. In practice, after normalization layers which are stacked following self-attention blocks, the mean and variance of Q_i and K_j are 0 and 1, respectively. Therefore, $\mathbb{E}[c_{ij}] = 0$, and this also implies that $c^{\max}$ has a large probability distribution around 0, which we also verify in Section 6.2. In conclusion, the difference between the two matrix distributions can be effectively controlled theoretically. Therefore, our KDLGT framework is a well-defined linear approximation of quadratic GTs.

6 Experiments

In this section, we present an evaluation of the precision and efficiency of our proposed Linear Structure-Aware Transformer (LSAT) and Shortest Anchor Path Distance Graph Transformer (SAPDGT) in comparison to state-of-the-art models on several graph benchmark datasets.

6.1 Experimental Setup

Datasets

We investigate the performance of LSAT and SAPDGT on both real-world datasets and synthetic datasets. The dataset statistical details are presented in Table 1.

For real-world datasets, 6 graph-level datasets and 4 node-level datasets are adopted. The benchmarking-GNN [Dwivedi *et al.*, 2020] (ZINC), OGB [Hu *et al.*, 2020] (OGBG-MOLHIV) and TUD [Morris *et al.*, 2020] (MUTAG, COX2_MD, PROTEINS, NCI1) are all popular graph-level benchmark datasets, where each graph represents a molecule, and nodes represent atoms in the molecules. The Cora, Citeseer and PubMed [Yang *et al.*, 2016] are popular citation datasets, whose nodes represent academic papers and node features are the word bag of papers. LastFM-Asia [Rozemberczki and Sarkar, 2020] is a social network that was collected from the public API in March 2020. Their nodes are LastFM users from Asian countries and edges are mutual follower relationships between them. The node features are extracted based on the artists liked by the users. In the experiments, for the datasets without public splits, we use random split with the ratio of training/validation/test sets being $7/1.5/1.5$.

For synthetic datasets, we generate a series of graphs for efficiency experiments. The size of the synthetic graphs increases from 2^{11} to 2^{14} in proportion to $\sqrt{2}$. (When the size is a non-integer, it will be rounded down.) Besides, in order to limit the density of the graph, we adopt 6-regular graphs here.

Dataset	# graphs	# classes	Avg # nodes	Avg # edges
ZINC	~250,000	–	23.2	49.8
OGBG-MOLHIV	41127	2	25.5	27.5
MUTAG	188	2	17.9	57.5
COX2_MD	303	2	41.2	43.5
PROTEINS	1113	2	39.1	184.7
NCI1	4110	2	29.8	94.5
Cora	1	7	2708	5429
Citeseer	1	6	3312	4732
PubMed	1	3	19717	44338
LastFM-Asia	1	18	7624	27806

Table 1: Statistics of real-world datasets. The symbol in # classes column represents the regression task.

Baselines

In the experiments, in addition to comparing with SAT and Graphormer which are quadratic-complexity graph Transformers and used as precision performance upper bound, we also select the following effective graph Transformers as strong baselines.

- **GPS** [Rampášek *et al.*, 2022] GPS is an effective GT architecture designed through GNN + Performer

paradigm. In experiments, we use its Laplacian eigenvectors encodings (LapPE) and random-walk structural encoding (RWSE) as position encodings. In order to avoid the influence of the GNN encoder in the GPS model, we set None (no GNN encoder) as the comparison experiment.

- **GKAT** [Choromanski *et al.*, 2022] GKAT is a GT model which applies low-rank masked attention via Random Walks Graph-Nodes Kernel (RWGNK).
- **DGT** [Park *et al.*, 2022] DGT reduces self-attention quadratic time complexity by performing sparse attention with dynamically sampled key and value pairs.

Settings

Our models are implemented in PyTorch [Paszke *et al.*, 2017]. We use Adam [Kingma and Ba, 2015] as the optimizer and set hyper-parameter ϵ to 1e-7 and (β_1, β_2) to $(0.99, 0.999)$, respectively. Besides, the initial learning rate is set to 1e-4 with a linear decay learning rate scheduler. The training and inference batch sizes are both set to 128. All models are trained and evaluated on 3 NVIDIA RTX 3090 GPUs for the fairness of efficiency comparison.

6.2 Experimental Results

Graph and Node Representation Experiments

Table 2 summarizes the performance of LSAT and SAPDGT on graph-level and node-level datasets. First of all, in general it can be observed that the precision performance degradation of LSAT and SAPDGT is not significant compared to SAT and Graphormer, respectively. Additionally, it is noteworthy that SAPDGT achieves better results than its upper bound Graphormer on the MUTAG, PROTEINS, and NCI1 datasets, and LSAT also outperforms its upper bound SAT on the Cora, Citeseer, and LastFM-Asia datasets. This highlights the exceptional generalization capability of our proposed KDLGT framework.

Specifically, on graph-level tasks, LSAT and SAPDGT achieve state-of-the-art results on the OGBG-MOLHIV, COX2_MD, PROTEINS, and NCI1 datasets. Besides, there is a significant performance margin between LSAT, SAPDGT and baselines on the NCI1 dataset. Furthermore, by comparing the results of GPS with or without GNN encoders, it can be observed that the contribution of GNN encoders to the performance of GPS is substantial. If the blessing of GNN encoders is lost, the performance of GPS will drop sharply. We think that this method of using GNN encoders to enhance performance is not specific to GPS and is applicable to various GT models, including our proposed models. Therefore, it is fairer and more reasonable to compare the performance of our model with the GPS model without GNN encoders. Under these conditions, it can be observed that LSAT and SAPDGT achieve state-of-the-art results on all graph-level datasets.

In regards to node-level tasks, unfortunately, it can be noticed that SAT and Graphormer do not perform well on node classification datasets with high edge homogeneity such as the Cora, Citeseer, and PubMed datasets. As a result, LSAT and SAPDGT do not exhibit significant advantages over the baselines on these datasets. Conversely, on the node classification datasets with low edge homogeneity such as LastFM-Asia, LSAT outperforms the baselines by a significant margin. This illustrates that LSAT is more suitable for learning on large-scale, low-homogeneity graph data.

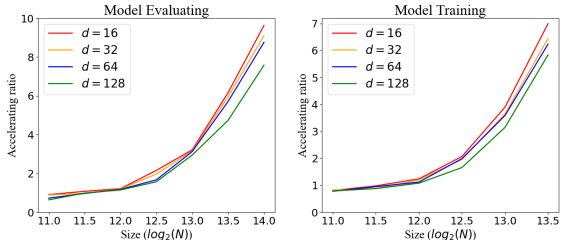

Figure 2: The accelerating ratio experimental results of the LSAT towards the SAT. The left and right sub-graphics represent the results of evaluating (forward only) and training (including both forward and backward), respectively. The parameter d denotes the embedding dimension of the models. The exceeding of the x-tick range indicates the out-of-memory problem of the SAT.

Effeiency Experiments

Here, we take the multiplication decomposition type, specifically LSAT, as an example, and use the accelerating ratio as a metric to analyze the performance of our proposed KDLGT framework under different graph sizes and model settings of embedding dimension d. The number of attention heads is fixed at 4 and the number of label classes of synthetic data is set to 5. The results of evaluating (forward only) and training (including both forward and backward) are recorded and presented in Figure 2.

First of all, it can be noticed that LSAT exhibits significant acceleration efficiency compared to SAT on large-scale graph data, achieving a remarkable result of 9.63x under certain data scales. Since LSAT is linear complexity while SAT is quadratic, it is expected that the acceleration ratio should increase linearly with respect to graph size N in theory, which is also supported by the results presented in Figure 2. It should be noted that here the horizontal axis is $\log_2(N)$, thus the curve actually grows linearly.

Moreover, the embedding dimension d also plays an important role. Generally, as the ratio of N/d increases, the acceleration ratio becomes more pronounced. In extreme cases, such as when $N \leq 2^{11.5}$, the acceleration ratio may be less than 1 due to the relatively small size of the graph, where d becomes the dominant factor affecting efficiency. In this scenario, the time saved by the KDLGT framework is less than the time required for other additional operations (such as sampling), resulting in an overall increase in time. This indicates that the KDLGT framework is more suitable for accelerating large-scale data, which aligns with the goal of this research.

Lastly, it is not surprising that the KDLGT framework also optimizes the space complexity of the Vaswani self-attention. In the experiments, we notice that when $N > 2^{14}$, an out-of-memory (OOM) problem occurs for SAT while LSAT can still operate normally. In conclusion, the KDLGT framework achieves obvious advantages in both time complexity and space complexity when applied to large-scale graph data.

Model	ZINC	OGBG-MOLHIV	MUTAG	COX2_MD	PROTEINS	NCI1
	MAE↓	AUCROC↑	Acc↑	Acc↑	Acc↑	Acc↑
SAT	0.082 ± 0.004	79.54± 1.32	92.26± 1.66	70.63± 1.54	77.51± 2.43	81.69± 1.08
Graphormer	0.122 ± 0.006*	74.55± 1.06	92.30± 2.73	68.33± 0.71	75.10± 1.07	78.95± 1.52
GPS (None + LapPE)	0.425± 0.081	71.15± 1.59	87.21± 3.28	65.22± 1.01	72.02± 1.51	68.07± 1.33
GPS (GNN + LapPE)	**0.131± 0.003**	**76.60± 0.63**	**90.65± 1.23**	67.39± 0.62	75.60± 1.22	71.47± 0.89
GPS (None + RWSE)	0.213± 0.008	73.09± 0.88	86.21± 2.09	68.89± 0.96	71.43± 2.88	73.26± 1.97
GPS (GNN + RWSE)	**0.070± 0.004***	**78.80± 0.49***	**91.38± 0.77**	**73.91± 0.35**	74.91± 0.98	**75.53± 1.48**
GKAT	-	-	87.94± 1.54	63.03± 1.14	**75.80± 3.80***	75.20± 2.40*
LSAT	**0.130± 0.002**	**78.98± 1.78**	90.18± 1.85	**69.69± 1.32**	**76.22± 1.64**	**81.33± 0.71**
SAPDGT	0.159± 0.008	73.75± 1.41	**93.16± 3.32**	**67.76± 1.68**	**77.27± 2.05**	**81.99± 1.03**

Model	Cora	Citeseer	PubMed	LastFM-Asia
SAT	83.06± 0.81	73.83± 0.79	89.19± 0.29	85.33± 0.79
Graphormer	73.34± 0.43	64.51± 0.53	OOM	70.88± 0.94
GPS (None + LapPE)	80.98± 0.76	75.90± 0.65	OOM	72.57± 0.86
GPS (GNN + LapPE)	**84.05± 0.59**	**85.30± 0.28**	OOM	73.31± 0.41
GPS (None + RWSE)	82.08± 0.52	**81.30± 1.01**	OOM	78.14± 0.55
GPS (GNN + RWSE)	**89.37± 0.19**	**90.89± 0.38**	OOM	**80.83± 0.52**
GKAT	73.84± 0.94	69.22± 0.80	**72.31± 0.58**	**77.65± 1.02**
DGT	**87.45± 0.60***	77.04± 0.57*	**89.22± 0.14***	-
LSAT	83.37± 0.21	74.03± 0.32	**89.03± 0.06**	**85.42± 0.27**
SAPDGT	72.02± 0.85	62.63± 1.17	OOM	71.27± 1.34

Table 2: Test performance on graph-level (upper) and node-level (lower) datasets. Shown results are the mean ± s.d. of 10 runs with different random seeds. Results with * are taken from the corresponding works. OOM represents out of memory. Highlighted ones are the top **first**, **second**, **third** results, respectively.

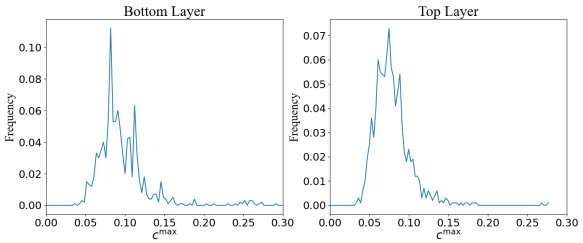

Figure 3: The distribution of the $c^{\max}$ in the bottom and top model layers.

The Distribution of the $c^{\max}$

We conduct the $c^{\max}$ distribution experiment on the NCI1 dataset as ablation studies. We adopt a three-layer LSAT model and fix the parameters after training for experiments. For visualization, we randomly sample $1,000$ graphs and use the frequency of occurrence to approximate the probability distribution. In particular, we directly obtain the maximum value of the $c^{\max}$ of different attention heads for simplicity. The experimental results are shown in Figure 3.

Overall, it can be observed that the distribution of the $c^{\max}$ is concentrated around 0, which supports our assumption in Section 5. Therefore, it is reasonable to assert that $c_{ij} \in [0, 1)$ for $\forall i, j \in \{1, 2, \ldots, N\}$.

Taking the result of the bottom layer as an example, we can verify the distribution gap between SAT and LSAT attention matrix exactly. The expectation and maximum values of $c^{\max}$ are around 0.1 and 0.3, corresponding to the upper bound value $1 + c^{\max}/1 - c^{\max}$ of 1.22 and 1.86, respectively. The difference between the two distributions at the expectation value is not significant. Considering the extremely infrequent occurrence of maximum value, we think this numerical behavior is acceptable.

Furthermore, it can be noticed that the expected value and the maximum value of the top layer is smaller than those of the bottom layer. We speculate that this is the result of the normalization layers in the GTs. The data features which are fed to the top layer, compared with the bottom layer, have gone through more normalization layers. As a result, the mean and variance of them are more stable, so the distribution of the $c^{\max}$ is closer to 0. Therefore, it can be inferred that the closer to the top layer, the better the approximation performance of our KDLGT framework.

7 Conclusion

In this work, we present the Kernel Decomposition Linear Graph Transformer (KDLGT), an accelerating framework for building scalable and powerful GTs. Under KDLGT framework, we select two representative GTs and design our models LSAT and SAPDGT. On one hand, a rigorous theoretical analysis is conducted to ensure performance guarantees. On the other hand, a series of experiments are carried out to evaluate the KDLGT in terms of precision and efficiency. Both the theoretical analysis and experimental results demonstrate that the KDLGT not only significantly improves learning efficiency but also preserves the precision performance of the GTs and achieves state-of-the-art results on various datasets.

References

[Alon and Yahav, 2021] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2021.

[Beltagy *et al.*, 2020] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[Chen *et al.*, 2022] Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. Structure-aware transformer for graph representation learning. In *International Conference on Machine Learning*, pages 3469–3489. PMLR, 2022.

[Child *et al.*, 2019] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[Choromanski *et al.*, 2020] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In *International Conference on Learning Representations*, 2020.

[Choromanski *et al.*, 2022] Krzysztof Choromanski, Han Lin, Haoxian Chen, Tianyi Zhang, Arijit Sehanobish, Valerii Likhosherstov, Jack Parker-Holder, Tamas Sarlos, Adrian Weller, and Thomas Weingarten. From block-toeplitz matrices to differential equations on graphs: towards a general theory for scalable masked transformers. In *International Conference on Machine Learning*, pages 3962–3983. PMLR, 2022.

[Dwivedi *et al.*, 2020] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30:1024–1034, 2017.

[Hu *et al.*, 2020] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in Neural Information Processing Systems*, 33:22118–22133, 2020.

[Huang *et al.*, 2020] Kexin Huang, Cao Xiao, Lucas M Glass, Marinka Zitnik, and Jimeng Sun. Skipgnn: predicting molecular interactions with skip-graph networks. *Scientific reports*, 10(1):1–16, 2020.

[Katharopoulos *et al.*, 2020] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020.

[Kingma and Ba, 2015] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[Kreuzer *et al.*, 2021] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

[Li *et al.*, 2021] Yangyang Li, Yipeng Ji, Shaoning Li, Shulong He, Yinhao Cao, Yifeng Liu, Hong Liu, Xiong Li, Jun Shi, and Yangchao Yang. Relevance-aware anomalous users detection in social network via graph neural network. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[Liu *et al.*, 2021] Xiyang Liu, Huobin Tan, Qinghong Chen, and Guangyan Lin. Ragat: Relation aware graph attention network for knowledge graph completion. *IEEE Access*, 9:20840–20849, 2021.

[Maron *et al.*, 2019] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. *Advances in Neural Information Processing Systems*, 32:2153–2164, 2019.

[Mialon *et al.*, 2021] Grégoire Mialon, Dexiong Chen, Margot Selosse, and Julien Mairal. Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667*, 2021.

[Morris *et al.*, 2019] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4602–4609, 2019.

[Morris *et al.*, 2020] Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

[Park *et al.*, 2022] Jinyoung Park, Seongjun Yun, Hyeonjin Park, Jaewoo Kang, Jisu Jeong, Kyung-Min Kim, Jungwoo Ha, and Hyunwoo J Kim. Deformable graph transformer. *arXiv preprint arXiv:2206.14337*, 2022.

[Parmar *et al.*, 2018] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR, 2018.

[Paszke *et al.*, 2017] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[Pavan and Pelillo, 2006] Massimiliano Pavan and Marcello Pelillo. Dominant sets and pairwise clustering. *IEEE transactions on pattern analysis and machine intelligence*, 29(1):167–172, 2006.

[Peng *et al.*, 2020] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. Random feature attention. In *International Conference on Learning Representations*, 2020.

[Qiu *et al.*, 2020] Jiezhong Qiu, Hao Ma, Omer Levy, Wentau Yih, Sinong Wang, and Jie Tang. Blockwise self-attention for long document understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2555–2565, 2020.

[Rampášek *et al.*, 2022] Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *arXiv preprint arXiv:2205.12454*, 2022.

[Rozemberczki and Sarkar, 2020] Benedek Rozemberczki and Rik Sarkar. Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 1325–1334, 2020.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:5998–6008, 2017.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[Wang *et al.*, 2020] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[Wang *et al.*, 2021] Yuyang Wang, Jianren Wang, Zhonglin Cao, and Amir Barati Farimani. Molclr: Molecular contrastive learning of representations via graph neural networks. *arXiv preprint arXiv:2102.10056*, 2021.

[Wu *et al.*, 2021] Zhanghao Wu, Paras Jain, Matthew Wright, Azalia Mirhoseini, Joseph E Gonzalez, and Ion Stoica. Representing long-range context for graph neural networks with global attention. *Advances in Neural Information Processing Systems*, 34:13266–13279, 2021.

[Xu *et al.*, 2019] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[Yang *et al.*, 2016] Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning*, pages 40–48. PMLR, 2016.

[Yasunaga *et al.*, 2021] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2021.

[Ying *et al.*, 2021] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888, 2021.

[Zhang *et al.*, 2020] Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. Graph-bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*, 2020.

[Zhong *et al.*, 2020] Ting Zhong, Tianliang Wang, Jiahao Wang, Jin Wu, and Fan Zhou. Multiple-aspect attentional graph neural networks for online social network user localization. *IEEE Access*, 8:95223–95234, 2020.