ARE MORE LAYERS BENEFICIAL TO GRAPH TRANSFORMERS?

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite that going deep has proven successful in many neural architectures, the existing graph transformers are relatively shallow. In this work, we explore whether more layers are beneficial to graph transformers, and find that current graph transformers suffer from the bottleneck of improving performance by increasing depth. Our further analysis reveals the reason is that deep graph transformers are limited by the vanishing capacity of global attention, restricting the graph transformer from focusing on the critical substructure and obtaining expressive features. To this end, we propose a novel graph transformer model named DeepGraph that explicitly employs substructure tokens in the encoded representation, and applies local attention on related nodes to obtain substructure based attention encoding. Our model enhances the ability of the global attention to focus on substructures and promotes the expressiveness of the representations, addressing the limitation of self-attention as the graph transformer deepens. Experiments show that our method unblocks the depth limitation of graph transformers and results in state-of-the-art performance across various graph benchmarks with deeper models.

1 INTRODUCTION

Transformers have recently gained rapid attention in modeling graph-structured data (Zhang et al., 2020; Dwivedi & Bresson, 2020; Maziarka et al., 2020; Ying et al., 2021; Chen et al., 2022). Compared to other graph neural networks, graph transformer implies global attention mechanism to enable information passing between every two nodes, which is advantageous to learn long-range dependency of the graph stuctures (Alon & Yahav, 2020). In transformer, the graph structure information can be encoded into node feature (Kreuzer et al., 2021) or attentions (Ying et al., 2021) by a variant of methods flexibly with strong expressiveness, avoiding the inherent limitations of encoding paradigms that pass the information along graph edges. Global attention (Bahdanau et al., 2015) also enables explicit focus on essential parts among the nodes to model crucial substructures in the graph.

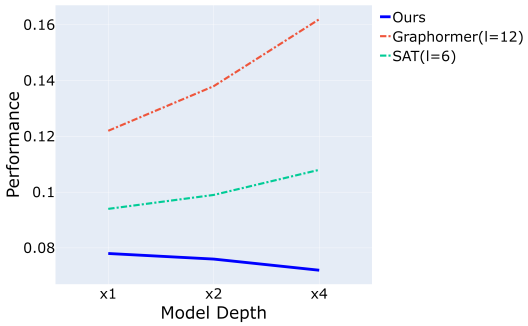

Figure 1: Performance on ZINC dataset of different graph transformers by varying their depths. (Lower is better.)

Graph transformer in current studies is usually shallow, i.e., less than 12 layers. Scaling depth is proven to increase the capacity of neural networks exponentially (Poole et al., 2016), and empirically improve transformer performance in nature language processing (Liu et al., 2020a; Bachlechner et al., 2021) and computer vision (Zhou et al., 2021). Graph neural network also benefits from more depth when properly designed (Chen et al., 2020a; Liu et al., 2020b; Li et al., 2021). However, it is still not clear whether the capability of graph transformers in graph tasks can be strengthened by increasing model depth. So we conduct experiments and find that current graph transformers encounter the bottleneck of improving performance by increasing depth. The further deepening will

hurt performance when model exceeds 12 layers, which seems to be the upper limit of the current graph transformer depth, as Figure 1 shows.

In this work, we aim to answer why more self-attention layers become a disadvantage for graph transformers, as well as how to address these issues with the proper model design. Self-attention mechanism (Bahdanau et al., 2015) makes a leap in model capacity by dynamically concentrating on critical parts of the input (Chen et al., 2022), i.e., substructures of a graph, and obtaining particular features. Substructures are the basic intrinsic features of graph data widely used in data analysis (Yu et al., 2020) and machine learning (Shervashidze et al., 2009), as well as graph model interpretability (Miao et al., 2022). Although the self-attention module appears to be very beneficial for automatically learning important substructure features in the graph, our analysis indicates that this ability vanishes as depth grows, restricting the deeper graph transformer from learning useful structure features. Specifically, we focus on the influence of attention on different substructure patterns, which we found to decrease after each self-attention layer. As a direct consequence, it is difficult for deep models to autonomously learn effective attention patterns of substructures and obtain expressive graph substructure features.

We further propose a graph transformer model named DeepGraph with a simple but effective method to enhance substructure encoding ability, enabling deeper graph transformer to achieve better performance. The proposed model explicitly introduces local attention mechanism on substructures by employing additional substructure tokens in the model representation and applying local attention to nodes related to those substructures. Our method not only introduces the substructure based attention to encourage the model to focus on substructure feature, but also enlarges the attention capacity theoretically, which improves the expressiveness of representations learned on substructures.

In summary, our contributions are as follows:

- We present the bottleneck of graph transformers' performance when depth increases, illustrating the depth limitation of current graph transformers. We theoretically study the bottleneck from the perspective of attention capacity's decay with layers, and demonstrate the difficulty for deep models to learn effective attention patterns of substructures and obtain informative graph substructure features.
- According to the theoretical finding, we propose a simple but effective substructure token based local attention mechanism in graph transformer, promoting focus on local substructure features of deeper graph transformer and improving the expressiveness of learned representations.
- Experiments show that our method unblocks the depth limitation of graph transformer and achieves state-of-the-art results on standard graph benchmarks with deeper models.

2 Related Work

Graph transformers Transformer with the self-attention based encoding mechanism has been the mainstream method in nature language processing (Vaswani et al., 2017; Devlin et al., 2019; Liu et al., 2019; Lewis et al., 2020; Brown et al., 2020; Raffel et al., 2020), and is also proven competitive for image and video processing in computer vision (Dosovitskiy et al., 2020; Touvron et al., 2021; Carion et al., 2020). The pure transformer is a set mapping function with no relation information between tokens and requires additional position encoding to inform the model of structure information. Recent works apply transformers in graph tasks by designing a variety of structure encoding techniques. Some works embed structure information into graph nodes by methods including Laplacian vector, random walk, or other feature (Zhang et al., 2020; Dwivedi & Bresson, 2020; Kreuzer et al., 2021; Kim et al., 2022; Wu et al., 2021). Some other works introduce structure information into attention by graph distance, path embedding or feature encoded by GNN (Park et al., 2022; Maziarka et al., 2020; Ying et al., 2021; Chen et al., 2022; Mialon et al., 2021; Choromanski et al., 2022). The two types of methods correspond to absolute position encoding or relative position encoding in transformers, respectively. Other works use transformer as a module of the whole model (Bastos et al., 2022; Guo et al., 2022).

Deep neural networks There are many works focus on solving obstacles and release potential of deep neural networks in feed-forward network (FNN) (Telgarsky, 2016; Yarotsky, 2017) and convolutional neural network (CNN) (He et al., 2016a; Simonyan & Zisserman, 2014; Xiao et al., 2018;

He et al., 2016b). For graph neural networks, early studies reveal severe performance degradation when stacking many layers (Li et al., 2019; Xu et al., 2018b; Li et al., 2018), caused by problems including over-smoothing and gradient vanishing (Oono & Suzuki, 2019; Zhao & Akoglu, 2019; Rong et al., 2019; Nt & Maehara, 2019). Recent works alleviate these problems by residual connection, dropout, and other methods (Chen et al., 2020a; Li et al., 2020; 2021), and deepen GNN into hundreds and even thousands of layers with better performance. Deep transformer is also proved powerful in nature language processing (Wang et al., 2022) and computational vision (Zhou et al., 2021), while better initialization (Zhang et al., 2019) or architecture (Wang et al., 2019; Liu et al., 2020a; Bachlechner et al., 2021) are proposed to solve optimization instability and over-smoothing (Shi et al., 2021; Wang et al., 2021; Dong et al., 2021).

Graph substructure Substructure is one of the basic intrinsic features of graph data, which is widely used in both graph data analysis (Shervashidze et al., 2009; Yanardag & Vishwanathan, 2015; Rossi et al., 2020) and deep graph neural models (Chen et al., 2020b; Bouritsas et al., 2022; Bodnar et al., 2021; Zhang & Li, 2021; Zhao et al., 2021). In application, substructure related methods are widely used in various domain including computational chemistry (Murray & Rees, 2009; Jin et al., 2018; 2019; Duvenaud et al., 2015; Yu et al., 2020), computational biology (Koyutürk et al., 2004) and social network (Jiang et al., 2010). Certain substructures can also be the pivotal feature for graph property prediction, which is a fundamental hypothesis in graph model interpretability studies (Ying et al., 2019; Miao et al., 2022), helping to understand how a graph model makes decisions.

3 PRELIMINARY

3.1 GRAPH AND SUBSTRUCTURE

We denote graph data as $\{G_i, y_i\}$, where a graph $G = (N, E, x, e)$ includes nodes $N = \{1 \dots |G|\}$ and corresponding edges $E \subset N \times N$, while x and e are node features and edge features. Label y can be graph-wise or node-wise, depending on the task definition. Given graph G, a substructure is defined as $G^S = \{N^S, E^S, x^S, e^S\}$, where $N^S \subset N, E^S = (N^S \times N^S) \cap E$, i.e. nodes of G^S form a subset of the graph G and edges are all the existing edges in G between nodes subset.

3.2 TRANSFORMER

The core module of the transformer is self-attention. Let $H_l = [h_1, \cdots, h_n]^\top \in \mathbb{R}^{n \times d_h}$ be the hidden representation in layer l, where n is the number of token, and d_h is the dimension of hidden embedding of each token. The self-attention mapping $\hat{\mathrm{Attn}}$ with parameter $W^V, W^Q, W^K \in \mathbb{R}^{d_h \times d_k}$ and $W^O \in \mathbb{R}^{d_k \times d_h}$ is

$$\begin{aligned} \hat{A} &= \frac{(H_l W^Q)(H_l W^K)^\top}{\sqrt{d_K}}, \quad A = \mathrm{softmax}(\hat{A}), \\ \hat{\mathrm{Attn}}(H_l) &= A H_l W^V W^O = A H_l W^{VO}, \end{aligned} \tag{1}$$

where $W^{VO} \triangleq W^V W^O$. In practice, a complete transformer layer also contains a two-layer fully-connected network FFN, and layer normalization with residual connection is also applied to both self-attention module and fully-connected network:

$$H_{l+1} = \mathrm{FFN}(\mathrm{Attn}(H_l)), \tag{2}$$

where $\mathrm{Attn}(H) = \mathrm{LN}(AHW^{VO}))$, $\mathrm{FFN}(H') = \mathrm{LN}(ReLU(H'W^{F_1} + \mathbf{1}{b^{F_1}}^T)W^{F_2} + \mathbf{1}{b^{F_2}}^T)$. $\mathrm{LN}(F(X)) = (X + F(X) - \mathbf{1}{b^N}^T)D$ is the layer normalization, where D is diagonal matrix with normalizing coefficients.

The vanilla transformer does not contain any structure information of graph, treating input tokens as a set. Structure information can be encoded into token representations or attentions. Our model adopts the distance and path-based relative position encoding methods as Ying et al. (2021), which utilizes graph distance DIS and shortest path SP information to guide attention calculation:

$$\hat{A}_{ij} = \frac{\left(h_i W^Q\right)\left(h_j W^K\right)^T}{\sqrt{d_K}} + b^D_{\mathrm{DIS}(i,j)} + \underset{k \in \mathrm{SP}(i,j)}{\mathrm{Mean}} b^E_{e_k} \tag{3}$$

We also employ deepnorm (Wang et al., 2022) residual connection method in out model, which adjust the residual connection in layer normalization by constant η: $\mathrm{LN}(F(X)) = (\eta X + F(X) + \mathbf{1}b^{N^T})D$ to stabilize the optimization of deep transformers.

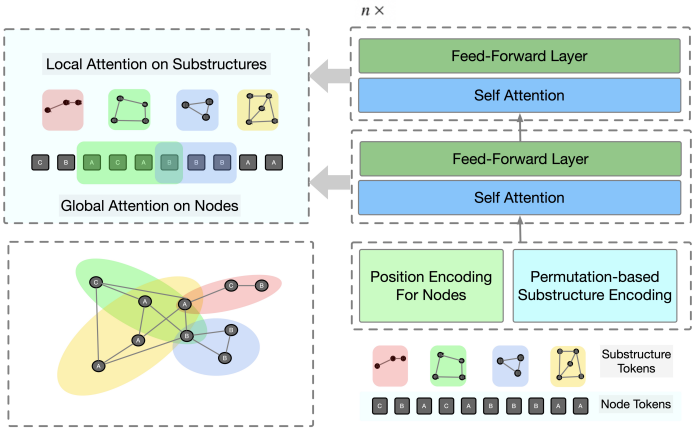

Figure 2: Overview of the proposed graph encoding framework.

4 Theoretical Results

In this section, we study the capacity of global attention to explain the depth bottleneck in graph transformers. As stated above, substructures are fundamental graph features to represent graph structure and predict graph properties. In the graph transformer, the global attention mechanism is expected to focus on important substructures and learn expressive substructure representations automatically. However, we find that in graph transformer, attention's influence decreases as depth grows. We define the attention capacity first, then analyze how the attention capacity varies with depth. Finally, we show that local attention on substructures is a good solution to the problem.

4.1 Definition of Attention Capacity

We define capacity of attention as the maximum difference between representations learned by different substructure attention patterns. Define a fixed attention pattern e of substructure G^S as an attention vector that only focus on nodes of G^S, i.e., $e \in \Delta^n, \mathrm{supp}(e) \in N^S$, where Δ^n is the attention space with dimension n, i.e. $\Delta^n = \{a | a \in [0,1]^n, a^T\mathbf{1} = 1\}$, and $\mathrm{supp}(e)$ returns all the non-zero dimensions. Given a graph G with several important substructures $G_1^S \dots G_m^S$ and corresponding attention patterns $e_1 \dots e_m$, we consider the attention space

$$\Delta_S^n \triangleq \{Ec | c \in \Delta^m\}, \tag{4}$$

where columns of E is the base vectors of attention patterns on substructures we are interested, i.e., $E = (e_1, e_2 \dots, e_m)$, and c is linear combination weight belongs to attention space with dimension m. The defined attention space constructs attention patterns focusing on different substructures.

Definition 1 *The attention capacity is defined as:*

$$\begin{aligned} F_H &= \max_{A_1, A_2 \in (\Delta_S^n)^n} |\hat{Attn}_{A_1}(H) - \hat{Attn}_{A_2}(H)|_F \\ &= \max_{C_1, C_2 \in (\Delta^m)^n} |C_1^T E^T H W^{VO} - C_2^T E^T H W^{VO}|_F, \end{aligned} \tag{5}$$

where $(\Delta_S^n)^n$ is matrix space with each column in space Δ_S^n, and similiarly for $(\Delta^m)^n$.

Attention capacity is an important property for graph transformer. The larger capacity means paying attention to different substructures can yield very different features. On the contrary, a small capacity means different substructures' attention patterns do not have a large effect on learned representations. It will be more difficult for the self-attention module with smaller capacity to learn attention patterns because the model is not sensitive to different substructures, and the learned representation is less informative with focused substructure.

4.2 ATTENTION CAPACITY DECREASE WITH DEPTH

Next, we illustrate that the stacked self-attention layers can cause decreased attention capacity for deeper layers. We first demonstrate that the self-attention module decreases the capacity of attention, then discuss the case with fully-connected layers.

Theorem 1 (The stacked attention modules decrease the capacity of attention) *We analyze capacity of attention F_{H_l} for stacked only attention modules $H_{i+1} = Attn(H_i) = (H_i + A_i H_i W_i^{VO} - \boldsymbol{1} b_i^{N^T}) D_i$. Assume $H_i W_i^{VO} D_i$ is full row rank for each layer i. Due to the property of layer normalization, assume the output of attention module equals to its input, i.e., for input X, $|\operatorname{Attn}(X)|_F \approx |X|_F$. Then the attention capacity of layer l is upper bounded by*

$$F_{H_l} \leq 2 \prod_{i=1}^{l-1} (\alpha_i) \sqrt{m} |P_m E^T|_2 |W_l^{VO}|_2 |P H_1|_F \tag{6}$$

,where $\alpha_i = \frac{(|PH_i + PA_i H_i W_i^{VO}) D_i|_F}{|(PH_i + A_i P H_i W_i^{VO} - \boldsymbol{1} b_i^{N^T}) D_i|_F} < 1$ with probability 1, and $P \triangleq (I - \frac{1}{n} \boldsymbol{1}\boldsymbol{1}^T)$, $P_m \triangleq (I - \frac{1}{m} \boldsymbol{1}\boldsymbol{1}^T)$.

Proof is in Appendix 1. The analysis above reveals that the self-attention module decreases the upper bound of attention capacity exponentially. We next consider the case when fully-connected layers are also included in each layer, just like the architecture in practice.

Theorem 2 (The upper bound of attention capacity after stacked transformer layers) *For transformer layer with self-attention $H_i' = \operatorname{Attn}(H_i) = (H_i + A_i H_i W_i^{VO} - \boldsymbol{1} b_i^{N^T}) D_i$, and fully-connected layer $H_{i+1} = \operatorname{FFN}(H_i') = (H_i' + ReLU(H_i' W_i^{F_1} + \boldsymbol{1} b_i^{F_1{}^T}) W_i^{F_2} + \boldsymbol{1} b_i^{F_2{}^T} - \boldsymbol{1} b_i^{N_2{}^T}) D_i'$, with the same assumption as the previous, attention capacity of layer l is upper bounded as follows:*

$$F_{H_l} \leq 2 \prod_{i=1}^{l-1} (\alpha_i \gamma_i) \sqrt{m} |P_m E^T|_2 |W_l^{VO}|_2 |P H_1|_F \tag{7}$$

,where $\alpha_i = \frac{(|PH_i + PA_i H_i W_i^{VO}) D_i|_F}{|(PH_i + A_i P H_i W_i^{VO} - \boldsymbol{1} b_i^{N^T}) D_i|_F} < 1$ with probability 1, and $\gamma_i = |D_i'|_2 (1 + |W_i^{F_1}|_2 |W_i^{F_2}|_2)$.

Proof is in Appendix 2. The theory shows that the upper bound of the attention capacity is affected by the combined effect of the two coefficients when the fully-connected layer is considered. Shi et al. (2021) shows that there always exists a fraction of data with $|D_i'|_2 < 1$, which dominants γ_i. In these cases, attention capacity vanishing happens at least for part of the data.

The decreasing attention capacity with depth reveals two problems in graph transformer. First, it is hard for graph transformer to pay attention to useful substructures when depth grows. Second, feature learned by attention mechanism loses expressiveness w.r.t substructures in deep layers. Both of these problems are pivotal to graph representation learning because the structural features of graphs are essential components of graph data.

An important reason for the reduction of attention capacity is the dense connection of global attention (Rong et al., 2019), which can attract representations into **1**-stretched space and increase denominator of α in Theorem 1 and 2. Intuitively, the node information interacts too densely in global attention, squashing local information in features of the deeper layers. Consequently, attention mechanism in the deep layers can no longer effectively distinguish the features at different positions, thereby weakening the expressive ability of the deep self-attention modules.

Following this line of thought, explicit substructure based local attention mechanism is a reasonable strategy to reduce the decay of attention capacity vanishing, where each node can only attend to other nodes that belong to the same substructure. We next provide proof that the problem of capacity decay can be alleviated if local attention on substructures is applied in each layer.

Theorem 3 *Define substructure based local attention as where each node only attends nodes that belong to the same substructures. Assume that for each node, the number of nodes belonging to*

the same substructures is at most r, where $r < n$. Let α^s be the decay coefficient of this model in Theorem 1, and α be the decay coefficient of the global attention model. Denoted minimum of α^s and α for all possible representation H_i and model parameters by α^{s*} and α^* respectively. Then we have $\alpha^{s*} > \alpha^*$.

The proof is in Appendix 3. This theory illustrates that substructure based local attention not only introduces the substructure based attention to encourage the model to focus on substructure features but also enlarges the attention capacity of deep layers, which promotes the representation capacity of the model when it grows deep.

5 APPROACH

As we analyzed above, explicit substructure based attention mechanism is a reasonable strategy to reduce the decay of attention capacity vanishing and promotes the representation capacity when model grows deep. However, replacing global attention directly with local attention may lead to insufficient learning ability of the model for long-range dependencies. To better introduce local attention to substructure in self-attention models, we propose a novel substructure token based local attention method, DeepGraph, as Figure 2 shows. DeepGraph tokenizes the graph into node level and substructure level. Local attention is applied to the substructure token and corresponding nodes, while the global attention between nodes is still preserved. This provides a better solution for explicitly introducing attention to substructures in the model while combining global attention, to achieve a better global-local encoding balance.

Our method first samples substructures of the graph, then encodes these substructures into tokens, and finally enforces local attention on these substructures. In this section, we introduce the construction and sampling of substructures, the encoding of substructures, and the form of local attention, respectively.

5.1 SUBSTRUCTURE SAMPLING

The diverse types of substructures in current studies can be divided into two classes: neighbors which contain local features, including k-hop neighbors and random walk neighbors, and geometric substructures representing graph topological features, like circles, pathes, stars and other special graphs. In this study, we explore all of these substructures towards a general and flexible substructure based local attention framework. In practice, these substructures can be pre-computed once and cached for reuse. Details of substructures are in Appendix.

The number of substructures in a graph often far exceeds the number of nodes. For the feasibility of the actual computation, we sample a certain substructure in each computation to encode the graph. Formally, at each time of encoding, for a graph G with the corresponding substructure set $\{G^S\}$, we sample a subset $\{G_1^S, G_2^S \dots G_m^S\}$ as input to our model:

$$\{G_1^S, G_2^S \dots G_m^S\} = \mathrm{SubstructureSampling}(\{G^S\}), \hat{y} = \mathrm{DeepGraph}(G, \{G_1^S, G_2^S \dots G_m^S\}) \tag{8}$$

To reduce bias caused by uneven substructure density, we hope sampled substructures cover every node of the graph as evenly as possible. Due to the divergent radios of different types of substructures in a graph, we also balance the substructure types in the sampling. The details of $\mathrm{SubstructureSampling}$ are in Appendix.

5.2 SUBSTRUCTURE TOKEN ENCODING

The input embedding contains nodes tokens embedding $\{h_1, h_2, \dots, h_n\}$ and substructure tokens embedding $\{h_{n+1}, \dots, h_{n+m}\}$, encoded by node feature encoder g_n and substructure encoder g_s individually. The token embedding of graph nodes is mapping from graph node feature x to feature vector h: $h_i = g_n(x_i), i \in \{1, 2, \dots n\}$.

For the substructure tokens, we apply permutation-based structure encoding (Murphy et al., 2019; Chen et al., 2020b; Nikolentzos & Vazirgiannis, 2020) by encoding the substructure adjacency matrix A_i^S directly. The core idea of permutation-based encoding is pooling the encoding output of permuted graph information over all the possible permutations, which is permutation-invariant and

has no information loss. We further apply depth-first search (DFS) with the order of node degree to reduce the possible order of substructure nodes. The details of DFS is in Appendix. The formal definition of substructure token encoder is

$$h_{n+i} = \underset{\pi \in \mathrm{DFS}(A_i^S)}{Pool} g_s(\pi(A_i^S)), i \in \{1, 2, \ldots m\} \tag{9}$$

In practice, sampling is applied instead of pooling. We found that only one sample is enough during training for the model to learn the substructure stably.

5.3 Local Attention on Substructures

After adding substructure tokens, localized attention is applied to the substructure and its corresponding nodes. Given input embedding $H_1 = (h_1, h_2, \ldots, h_{n+m})$, mask M is added in self-attention module $\hat{\mathrm{Attn}}_{\mathrm{m}}(H, M)$ as

$$\hat{\mathrm{Attn}}_{\mathrm{m}}(H_l, M) = \mathrm{softmax}(\hat{A} + M)H_l W^{VO} \tag{10}$$

where $M_{ij} \in \{0, -\infty\}$, i.e., the elements of mask M can be 0 or $-\infty$, leading to a sparse attention matrix A with zero in position of $-\infty$ in M. In our model, the mask is defined as follows to induce local attention to substructure tokens and corresponding nodes: $M_{ij} = -\infty$ if $i + n \in \{n+1, n+2, \ldots n+m\}, j \notin N_i^S$ or $j + n \in \{n+1, n+2, \ldots n+m\}, i \notin N_j^S$, and $M_{ij} = 0$ otherwise. The substructure tokens only apply local attention to corresponding nodes.

The attention in our model is a combination of local attention on substructures and global attention on all nodes. The substructure nodes serve as the core of local attention. In each step, the substructure token only pays attention to its corresponding nodes, integrating the representation from the whole substructure. Nodes belonging to the same substructure all share the message from the substructure token, which marks the nodes inside the substructure and increases the distinguishability between nodes from different substructures, promoting the representation capacity w.r.t substructure.

6 Experiments

In this section, we aim to validate the performance of DeepGraph empirically. Specifically, we attempt to answer the following questions: **(i)** How does DeepGraph perform compared to existing transformers on popular benchmarks? **(ii)** Does DeepGraph's performance improve when depth increases, and is it more effective compared to other methods of deepening transformer? **(iii)** How much does each part of our model affect the model's performance? We first conduct experiments to evaluate our model on four datasets, including graph and node prediction tasks, compared with both state-of-art graph transformer models and their deeper versions augmented by recent methods toward deep transformers. Then we validate the effect of different components of our architecture by ablation studies. The codes of experiments will be released.

6.1 Datasets

Our method is validated on the tasks of the graph property prediction and node classification, specifically including PCQM4M-LSC (Hu et al., 2020), ZINC (Dwivedi et al., 2020), CLUSTER (Dwivedi et al., 2020) and PATTERN (Dwivedi et al., 2020), widely used in graph transformer studies. PCQM4M-LSC is a large-scale graph-level prediction dataset with more than 3.8M molecules. ZINC consists of 12,000 graphs with a molecule property to regression. CLUSTER and PATTERN are challenging node classification tasks with graph sizes varying from dozens to hundreds, containing 14,000 and 12,000 graphs, respectively.

6.2 Baselines

We choose recent state-of-art graph transformer models: GT (Dwivedi & Bresson, 2020), Graph-BERT(Zhang et al., 2020), SAN (Kreuzer et al., 2021), Graphormer (Ying et al., 2021) and SAT (Chen et al., 2022), covering various graph transformer methods including absolute and relative position embedding. Graph neural networks baselines includes GCN (Kipf & Welling, 2016), GAT Veličković et al. (2018) and GIN (Xu et al., 2018a). We first compare our model with the standard

	Graph regression		Node classification	
	PCQM4M-LSC ↓	ZINC ↓	CLUSTER ↑	PATTERN ↑
GCN	0.1691	0.367 ± 0.011	68.498 ± 0.976	71.892 ± 0.334
GAT	-	0.384 ± 0.007	70.587 ± 0.447	78.271 ± 0.186
GIN	0.1537	0.526 ± 0.051	64.716 ± 1.553	85.387 ± 0.136
Graph-BERT	-	0.267 ± 0.012	70.790 ± 0.537	75.489 ± 0.216
GT-sparse	-	0.226 ± 0.014	73.169 ± 0.622	84.808 ± 0.068
GT-Full	-	0.598 ± 0.049	27.121 ± 8.471	56.482 ± 3.549
SAN-Sparse	-	0.198 ± 0.004	75.738 ± 0.106	81.329 ± 2.150
SAN-Full	-	0.139 ± 0.006	76.691 ± 0.247	86.581 ± 0.037
Graphormer	0.1234	0.122 ± 0.006	-	-
SAT	-	0.094 ± 0.008	77.856 ± 0.104	86.865 ± 0.043
DeepGraph (12)	0.1220	0.078 ± 0.006	77.526 ± 0.122	90.015 ± 0.038
DeepGraph (24)	0.1206	0.076 ± 0.004	77.810 ± 0.167	90.522 ± 0.059
DeepGraph (48)	**0.1193**	**0.072 ± 0.004**	**77.912 ± 0.138**	**90.657 ± 0.062**

Table 1: Comparison of our DeepGraph to SOTA methods on graph regression and node classification tasks.

state-of-art models. Then we compare our model with the deepened versions of the state-of-arts augmented by recent algorithms designed for deep transformers, including the fusion method (Shi et al., 2021), and reattention method (Zhou et al., 2021). As we use deepnorm to stabilize our training process, we compare DeepGraph to naive deepnorm in the ablation study.

6.3 Settings

We implement the models with 12, 24, and 48 layers. The hidden dimension is 80 for ZINC and PATTERN, 48 for CLUSTER, and 768 for PCQM4M-LSC. The models are trained by Adam optimizer, with warm-up and learning rate decay for learning rate adjusting. Reported results are the average of over 4 seeds. Our model uses both geometric substructure and neighbors as substructure patterns. The geometric substructure includes k-path, k-star, and k-circle. The neighbors include k-hop neighbor and k-step random walk neighbor. We imply both geometric substructures and k-hop neighbors on ZINC and PCQM4M-LSC. As for CLUSTER and PATTERN, only random walk neighbors are used due to the large scale and dense connection of these two datasets. The effect of different substructure patterns is shown in the ablation study.

6.4 Main Results

Table 1 summarizes our experimental results on the graph regression and node classification tasks. Data of the baseline models are copied from their papers. The depth of our model varies from 12 to 48 layers. As the results illustrated, our 12-layer-model outperforms baseline models on PCQM4M-LSC, ZINC, and PATTERN, especially on ZINC and PATTERN, surpassing the previous best result reported significantly. As model depth increase, our model achieves consistent improvement, especially on dataset ZINC. Note that our model with 48 layers outperforms baseline models on all the datasets, proving the effectiveness of more stacked DeepGraph layers.

6.5 Effect of Deepening

We next compare DeepGraph with other baseline models after deepening. We choose Graphormer and SAT as baseline models to deepen by 2 and 4 times compared to the original version. In addition to deepening naively, we also enhance the baseline model by reproducing fusion and reattention in the baseline models.

	DeepGraph 12		DeepGraph 48	
	ZINC	CLUSTER	ZINC	CLUSTER
- local attention	0.135	76.167	0.141	76.133
- substructure encoding	0.085	76.531	0.124	77.884
- deepnorm	0.080	77.202	0.074	77.682
DeepGraph	**0.078**	**77.526**	**0.072**	**77.912**

Table 2: Ablation study of local attention, substructure encoding, and deepnorm on ZINC and CLUSTER.

	DeepGraph 12	DeepGraph 48
Geometric substructures only	0.078	0.075
Neighbors only	0.086	0.079
Both	**0.078**	**0.072**

Table 3: Sensitivity study of different substructure types on ZINC.

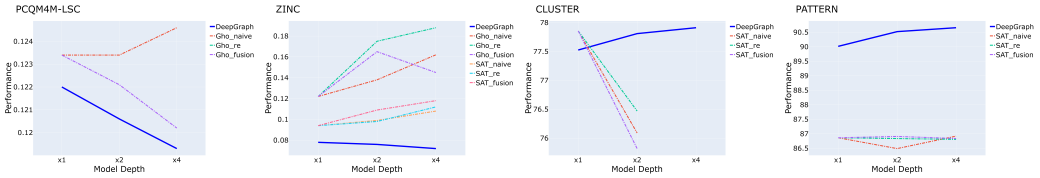

Figure 3: Comparison of DeepGraph to SOTA methods deepened by different deep transformer methods.

As shown in table 3, DeepGraph outperforms baselines with a significant margin. Naive deepening decreases performance on all the datasets for both two base models. Note that all the 4-time deeper SAT fail to converge on CLUSTER due to the training difficulty, as well as Graphormer with reattention. Fusion improves deeper Graphormer on PCQM4M-LSC, and reattention boosts 2x deep SAT on CLUSTER slightly, but none of them are consistent. Our methods surpass fusion and reattention methods stably on all 4 datasets, supporting our theoretical claim that sensitivity to substructures is crucial in deep graph transformers.

6.6 Ablation and Sensitivity Study

Finally, we validate the effectiveness of each component of our model by ablation study. We drop the components, including the local attention, the substructure encoding, and deepnorm method. Note that without local attention, the model is only a relative position based graph transformer with deepnorm residual connection. Without substructure encoding, randomly initialized embedding is applied to each substructure token. We conduct experiments on ZINC and CLUSTER, individually representing two types of tasks. The result is in Table 2. It can be found that without local attention, performance dropped a lot on both datasets, and the deeper model performs worse than the shallow ones, proving the effectiveness of the proposed methods. The substructure encoding also plays an important role, emphasizing the importance of structural information. Finally, deepnorm also contributes to the model performance by stabilizing the optimization, especially for the 48-layer models.

We validate the sensitivity of our model to different substructure types by testing the performance of DeepGraph with only geometric substructures or neighbors, as shown in Table 3. While both types of substructures contribute to the performance, geometric substructures play a more critical role in the ZINC dataset. This result is expected because specific structures like rings are important fundamental features for the molecule data. At the same time, it can be further strengthened when combined with neighbor substructures. Our model can flexibly use different forms of substructure to fully utilize prior knowledge of data. Note that for CLUSTER and PATTERN, only neighbor type substructures are utilized, showing that the neighbor information also plays an important role in the complex graph.

7 Conclusions

This work presents the bottleneck of graph transformers' performance when depth increases. We find that deep graph transformers are limited by the capacity bottleneck of graph attention by theoretical analysis. We further propose a novel graph transformer model that applies substructure based local attention with additional substructure tokens substructure based attention encoding. Our model enhances the ability of the global attention mechanism to focus on substructures and addresses the limitation of encoding expressive features as the model deepens. Experiments show that our model breaks the depth bottleneck and achieves state-of-the-art performance on popular graph benchmarks.

REFERENCES

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations*, 2020.

Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. Rezero is all you need: Fast convergence at large depth. In *Uncertainty in Artificial Intelligence*, pp. 1352–1361. PMLR, 2021.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

Anson Bastos, Abhishek Nadgeri, Kuldeep Singh, Hiroki Kanezashi, Toyotaro Suzumura, and Isaiah Onando Mulang. Investigating expressiveness of transformer in spectral domain for graphs. *arXiv preprint arXiv:2201.09332*, 2022.

Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Lio, Guido F Montufar, and Michael Bronstein. Weisfeiler and lehman go cellular: Cw networks. *Advances in Neural Information Processing Systems*, 34:2625–2640, 2021.

Giorgos Bouritsas, Fabrizio Frasca, Stefanos P Zafeiriou, and Michael Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pp. 213–229. Springer, 2020.

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

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pp. 1725–1735. PMLR, 2020a.

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? *Advances in neural information processing systems*, 33:10383–10395, 2020b.

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*, pp. 3962–3983. PMLR, 2022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *International Conference on Machine Learning*, pp. 2793–2803. PMLR, 2021.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems*, 28, 2015.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

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

Lingbing Guo, Qiang Zhang, and Huajun Chen. Unleashing the power of transformer for graphs. *arXiv preprint arXiv:2202.10581*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016b.

Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3464–3473, 2019.

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.

Chuntao Jiang, Frans Coenen, and Michele Zito. Finding frequent subgraphs in longitudinal social network data using a weighted graph mining approach. In *International Conference on Advanced Data Mining and Applications*, pp. 405–416. Springer, 2010.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pp. 2323–2332. PMLR, 2018.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical graph-to-graph translation for molecules. *arXiv preprint arXiv:1907.11223*, 2019.

Jinwoo Kim, Tien Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. *arXiv preprint arXiv:2207.02505*, 2022.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Mehmet Koyutürk, Ananth Grama, and Wojciech Szpankowski. An efficient algorithm for detecting frequent subgraphs in biological networks. *Bioinformatics*, 20(suppl_1):i200–i207, 2004.

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.

Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 66–71, 2018.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880, 2020.

Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9267–9276, 2019.

Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*, 2020.

Guohao Li, Matthias Müller, Bernard Ghanem, and Vladlen Koltun. Training graph neural networks with 1000 layers. In *International conference on machine learning*, pp. 6437–6449. PMLR, 2021.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*, 2018.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. Understanding the difficulty of training transformers. In *2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020*, pp. 5747–5763. Association for Computational Linguistics (ACL), 2020a.

Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 338–348, 2020b.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Łukasz Maziarka, Tomasz Danel, Sławomir Mucha, Krzysztof Rataj, Jacek Tabor, and Stanisław Jastrzebski. Molecule attention transformer. *arXiv preprint arXiv:2002.08264*, 2020.

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

Siqi Miao, Mia Liu, and Pan Li. Interpretable and generalizable graph learning via stochastic attention mechanism. In *International Conference on Machine Learning*, pp. 15524–15543. PMLR, 2022.

Ryan Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Relational pooling for graph representations. In *International Conference on Machine Learning*, pp. 4663–4673. PMLR, 2019.

Christopher W Murray and David C Rees. The rise of fragment-based drug discovery. *Nature chemistry*, 1(3):187–192, 2009.

Giannis Nikolentzos and Michalis Vazirgiannis. Random walk graph neural networks. *Advances in Neural Information Processing Systems*, 33:16211–16222, 2020.

Hoang Nt and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2019.

Wonpyo Park, Woong-Gi Chang, Donggeon Lee, Juntae Kim, et al. Grpe: Relative positional encoding for graph transformer. In *ICLR2022 Machine Learning for Drug Discovery*, 2022.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International conference on machine learning*, pp. 4055–4064. PMLR, 2018.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. *Advances in neural information processing systems*, 29, 2016.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2019.

Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, Sungchul Kim, Anup Rao, and Yasin Abbasi-Yadkori. A structural graph representation learning framework. In *Proceedings of the 13th international conference on web search and data mining*, pp. 483–491, 2020.

Nino Shervashidze, SVN Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. Efficient graphlet kernels for large graph comparison. In *Artificial intelligence and statistics*, pp. 488–495. PMLR, 2009.

Han Shi, JIAHUI GAO, Hang Xu, Xiaodan Liang, Zhenguo Li, Lingpeng Kong, Stephen MS Lee, and James Kwok. Revisiting over-smoothing in bert from the perspective of graph. In *International Conference on Learning Representations*, 2021.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Matus Telgarsky. Benefits of depth in neural networks. In *Conference on learning theory*, pp. 1517–1539. PMLR, 2016.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pp. 10347–10357. PMLR, 2021.

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, 2017.

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

Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. Deepnet: Scaling transformers to 1,000 layers. *arXiv preprint arXiv:2203.00555*, 2022.

Peihao Wang, Wenqing Zheng, Tianlong Chen, and Zhangyang Wang. Anti-oversmoothing in deep vision transformers via the fourier domain analysis: From theory to practice. In *International Conference on Learning Representations*, 2021.

Qiang Wang, Bei Li, Tong Xiao, Jingbo Zhu, Changliang Li, Derek F Wong, and Lidia S Chao. Learning deep transformer models for machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1810–1822, 2019.

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.

Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. In *International Conference on Machine Learning*, pp. 5393–5402. PMLR, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018a.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pp. 5453–5462. PMLR, 2018b.

Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1365–1374, 2015.

Dmitry Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94: 103–114, 2017.

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.

Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnn explainer: A tool for post-hoc explanation of graph neural networks. *arXiv preprint arXiv:1903.03894*, 2019.

Shuo Yu, Yufan Feng, Da Zhang, Hayat Dino Bedru, Bo Xu, and Feng Xia. Motif discovery in networks: a survey. *Computer Science Review*, 37:100267, 2020.

Biao Zhang, Ivan Titov, and Rico Sennrich. Improving deep transformer with depth-scaled initialization and merged attention. In *2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing*, pp. 898–909. Association for Computational Linguistics (ACL), 2019.

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.

Muhan Zhang and Pan Li. Nested graph neural networks. *Advances in Neural Information Processing Systems*, 34:15734–15747, 2021.

Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. In *International Conference on Learning Representations*, 2019.

Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From stars to subgraphs: Uplifting any gnn with local structure awareness. In *International Conference on Learning Representations*, 2021.

Daquan Zhou, Bingyi Kang, Xiaojie Jin, Linjie Yang, Xiaochen Lian, Zihang Jiang, Qibin Hou, and Jiashi Feng. Deepvit: Towards deeper vision transformer. *arXiv preprint arXiv:2103.11886*, 2021.

8 APPENDIX

Proof 1 *for brevity, define* $P \triangleq (I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)$*, and* $P_m \triangleq (I - \frac{1}{m}\mathbf{1}\mathbf{1}^T)$*. We first introduce an important property of* P*: For any matrix* B *with left eigenvector* $B\mathbf{1} = \beta\mathbf{1}$*, we have* PBC=$PBPC, \forall C$*. To prove it, we can deompose* $C = PC + \frac{1}{n}\mathbf{1}\mathbf{1}^T C$*. Then*

$$PBC = PB(PC + \frac{1}{n}\mathbf{1}\mathbf{1}^T C) = PBPC + \beta\frac{1}{n}P\mathbf{1}\mathbf{1}^T C = PBPC$$

We next prove the main theorem by recursion. By definition,

$$\begin{aligned} F_H &= \max_{\theta_1,\theta_2} |\hat{Attn}_{\theta_1}(H) - \hat{Attn}_{\theta_2}(H)|_F \\ &= \max_{A_1,A_2 \in (\Delta_S^n)^n} |A_1^T H W^{VO} - A_2^T H W^{VO}|_F \\ &= \max_{C_1,C_2 \in (\Delta^m)^n} |C_1^T E^T H W^{VO} - C_2^T E^T H W^{VO}|_F \\ &= \max_{C_1,C_2 \in (\Delta^m)^n} |(C_1 - C_2)^T E^T H W^{VO}|_F \end{aligned} \tag{11}$$

Note that $(C_1 - C_2)^T\mathbf{1} = \mathbf{1} - \mathbf{1} = 0$*, so we can multiply* P *before it:*

$$
\begin{aligned}
F_H &= \max_{C_1, C_2 \in (\Delta^m)^n} |(C_1 - C_2)^T E^T H W^{VO}|_F \\
&= \max_{C_1, C_2 \in (\Delta^m)^n} |(C_1 - C_2)^T P_m E^T H W^{VO}|_F \\
&\leq \max_{C_1, C_2 \in (\Delta^m)^n} |(C_1 - C_2)^T|_2 |P_m E^T H W^{VO}|_F \\
&\leq 2\sqrt{m} |P_m E^T H W^{VO}|_F
\end{aligned} \tag{12}
$$

,where $|(C_1 - C_2)^T| \leq |C_1|_2 + |C_2|_2$*, and* $|C|_2 \leq \sqrt{|C|_1 |C|_\infty} = \sqrt{m}, \forall C$*.*

We further decompose $|PE^T HW^{VO}|_F$ *into*

$$
|P_m E^T H W^{VO}|_F = |P_m E^T P H W^{VO}|_F \leq |P_m E^T|_2 |W^{VO}|_2 |PH|_F \tag{13}
$$

For layer l, we recursively compute $|PH_{i+1}|_F$ *by* $|PH_i|_F$ *for layer* $i = l-1, l-2, \ldots, 1$*:*

$$
\begin{aligned}
|PH_{i+1}|_F &= |P(H_i + A_i H_i W_i^{VO} - \mathbf{1} b_i^{N^T}) D_i|_F \\
&= |P(PH_i + A_i P H_i W_i^{VO} - \mathbf{1} b_i^{N^T}) D_i|_F \\
&= \frac{|P(PH_i + A_i P H_i W_i^{VO} - \mathbf{1} b_i^{N^T}) D_i|_F}{|(PH_i + A_i P H_i W_i^{VO} - \mathbf{1} b_i^{N^T}) D_i|_F} \frac{|(PH_i + A_i P H_i W_i^{VO} - \mathbf{1} b_i^{N^T}) D_i|_F}{|PH_i|_F} |PH_i|_F \\
&= \alpha_i \lambda_i |PH_i|_F
\end{aligned} \tag{14}
$$

,where $\alpha_i = \frac{|P(PH_i + A_i P H_i W_i^{VO} - \mathbf{1} b_i^{N^T}) D_i|_F}{|(PH_i + A_i P H_i W_i^{VO} - \mathbf{1} b_i^{N^T}) D_i|_F}$*, and* $\lambda_i = \frac{|(PH_i + A_i P H_i W_i^{VO} - \mathbf{1} b_i^{N^T}) D_i|_F}{|PH_i|_F}$*.*

Note that $P = (I - \frac{1}{n}\mathbf{1}\mathbf{1}^T)$ *is a contraction mapping, so* $\alpha_i \leq 1$*, and equal to 1 if and only if* $A_i P H_i W_i^{VO} D_i - \mathbf{1} b_i^{N^T} D_i$ *is orthogonal to the* $\mathbf{1}$*-stretched space, which is usually impossible, because* $-\mathbf{1} b_i^{N^T} D_i$ *is in the* $\mathbf{1}$*-stretched space, and the probability that projection of* $A_i P H_i W_i^{VO} D_i$ *on* $\mathbf{1}$*-stretched space equals to* $\mathbf{1} b_i^{N^T} D_i$ *is 0 due to the property of random matrix. We further show that* $A_i P H_i W^{VO}$ *is full-rank with probability 1: As assumption,* $H_i W_i^{VO} D_i$ *is full-rank, and* $\mathbf{1}$*-stretched space is the only subspace that* $P H_i W_i^{VO} D_i$ *is orthogonal to, i.e.*

$$
x^T P H_i W_i^{VO} D_i = 0 \Leftrightarrow x = \beta \mathbf{1}
$$

. So if $\mathbf{1}^T A_i P H_i W_i^{VO} D_i = 0$*,* $\mathbf{1}^T A_i = \beta \mathbf{1}^T$*, i.e.* $A_i^T \mathbf{1} = \beta \mathbf{1}$*, the probability of which is 0 for random parameters in transformer.*

For coefficient λ_i*, due to the property of layer normalization that keeps norm of output equals to input, i.e.* $|(X + A_i X W_i^{VO}) D_i|_F / |X|_F \approx 1, \forall X$*, we have* $\lambda \approx 1$*.*

Based on the analysis above, we can bound F_{H_l} *recursively:*

$$
F_{H_l} \leq 2 \prod_{i=1}^{l-1} (\alpha_i) \sqrt{m} |P_m E^T|_2 |W_l^{VO}|_2 |PH_1|_F \tag{15}
$$

, where $\alpha_i = \frac{|P(PH_i + A_i P H_i W_i^{VO} - \mathbf{1} b_i^{N^T}) D_i|_F}{|(PH_i + A_i P H_i W_i^{VO} - \mathbf{1} b_i^{N^T}) D_i|_F} = \frac{(|PH_i + P A_i H_i W_i^{VO}) D_i|_F}{|(PH_i + A_i P H_i W_i^{VO} - \mathbf{1} b_i^{N^T}) D_i|_F} < 1$ *with probability 1.*

Proof 2 *As the same method in the previous proof, we need to recursively bound* $|PH_{i+1}|_F$ *by* $|PH_i|_F$ *for layer* $i = l-1, l-2, \ldots, 1$*. Deote* $H_i' = \mathrm{Attn}(H_i)$*, and* $H_{i+1} = \mathrm{FFN}(H_i')$*. Similar to the previous proof, we have*

$$
\begin{aligned}
|PH_i'|_F &= |P(H_i + A_i H_i W_i^{VO} - \boldsymbol{1} b_i^{N^T}) D_i|_F \\
&= |P(PH_i + A_i PH_i W_i^{VO} - \boldsymbol{1} b_i^{N^T}) D_i|_F \\
&= \frac{|P(PH_i + A_i PH_i W_i^{VO} - \boldsymbol{1} b_i^{N^T}) D_i|_F}{|(PH_i + A_i PH_i W_i^{VO} - \boldsymbol{1} b_i^{N^T}) D_i|_F} \frac{|(PH_i + A_i PH_i W_i^{VO} - \boldsymbol{1} b_i^{N^T}) D_i|_F}{|PH_i|_F} |PH_i|_F \\
&= \alpha_i \lambda_i |PH_i|_F
\end{aligned} \tag{16}
$$

As for PH_{i+1}, we can bound it as

$$
\begin{aligned}
|PH_{i+1}|_F &= |P(H_i' + ReLU(H_i' W_i^{F_1} + \boldsymbol{1} b_i^{F_1{}^T}) W_i^{F_2} + \boldsymbol{1} b_i^{F_2{}^T} - \boldsymbol{1} b_i^{N_2{}^T}) D_i'|_F \\
&= |PH_i' D_i'|_F + |PReLU(H_i' W_i^{F_1} + \boldsymbol{1} b_i^{F_1{}^T}) W_i^{F_2} D_i'|_F \\
&\leq |PH_i'|_F |D_i'|_2 + |PReLU(H_i' W_i^{F_1} + \boldsymbol{1} b_i^{F_1{}^T})|_F |W_i^{F_2}|_2 |D_i'|_2 \\
&\leq |PH_i'|_F |D_i'|_2 + |P(H_i' W_i^{F_1} + \boldsymbol{1} b_i^{F_1{}^T})|_F |W_i^{F_2}|_2 |D_i'|_2 \\
&\leq |PH_i'|_F |D_i'|_2 + |PH_i'|_F |W_i^{F_1}|_2 |W_i^{F_2}|_2 |D_i'|_2 \\
&= |D_i'|_2 (1 + |W_i^{F_1}|_2 |W_i^{F_2}|_2) |PH_i'|_F
\end{aligned} \tag{17}
$$

So the final conclusion is

$$
F_{H_l} \leq 2 \prod_{i=1}^{l-1} (\alpha_i \gamma_i) \sqrt{m} |P_m E^T|_2 |W_l^{VO}|_2 |PH_1|_F \tag{18}
$$

, where $\alpha_i = \frac{|P(PH_i + A_i PH_i W_i^{VO} - \boldsymbol{1} b_i^{N^T}) D_i|_F}{|(PH_i + A_i PH_i W_i^{VO} - \boldsymbol{1} b_i^{N^T}) D_i|_F} = \frac{(|PH_i + PA_i H_i W_i^{VO}) D_i|_F}{|(PH_i + A_i PH_i W_i^{VO} - \boldsymbol{1} b_i^{N^T}) D_i|_F} < 1$ with probability 1, and $\gamma_i = |D_i'|_2 (1 + |W_i^{F_1}|_2 |W_i^{F_2}|_2)$.

Proof 3 *We denote global attention and local attention by A and A^s, respectively.*

By definition, $\alpha_i = \frac{(|PH_i + PA_i H_i W_i^{VO}) D_i|_F}{|(PH_i + A_i PH_i W_i^{VO} - \boldsymbol{1} b_i^{N^T}) D_i|_F}$. We analyse the worst case for α_i:

$$
\begin{aligned}
\alpha_i &= \frac{(|PH_i + PA_i H_i W_i^{VO}) D_i|_F}{|(PH_i + A_i PH_i W_i^{VO} - \boldsymbol{1} b_i^{N^T}) D_i|_F} \\
&= \frac{(|PH_i + PA_i H_i W_i^{VO}) D_i|_F}{|(PH_i + (P + \frac{1}{n} \boldsymbol{1}\boldsymbol{1}^T) A_i PH_i W_i^{VO} - \boldsymbol{1} b_i^{N^T}) D_i|_F} \\
&\geq \frac{(|PH_i + PA_i H_i W_i^{VO}) D_i|_F}{|(PH_i + PA_i PH_i W_i^{VO}) D_i|_F + |\frac{1}{n} \boldsymbol{1}\boldsymbol{1}^T A_i PH_i W_i^{VO} D_i|_F + |\boldsymbol{1} b_i^{N^T} D_i|_F}
\end{aligned} \tag{19}
$$

Further analysis on $|\frac{1}{n} \boldsymbol{1}\boldsymbol{1}^T A_i PH_i W_i^{VO} D_i|_F$ shows that

$$
\begin{aligned}
|\frac{1}{n} \boldsymbol{1}\boldsymbol{1}^T A_i PH_i W_i^{VO} D_i|_F &= |\boldsymbol{1}^T A_i PH_i W_i^{VO} D_i|_2 \\
&\leq |H_i W_i^{VO} D_i|_2 |PA_i^T \boldsymbol{1}|_2,
\end{aligned} \tag{20}
$$

where $PA_i^T \boldsymbol{1} = A_i^T \boldsymbol{1} - \boldsymbol{1}$, the element of which is between $[-1, n-1]$, and the sum of all elements of which of which is 0. The maximum of $|PA_i^T \boldsymbol{1}|_2$ is achieved when one element is $n-1$ and other elements are -1.

However, for local attention A_i^s that each node is only attended by at most r nodes, where $r < n$, the elements of $|PA_i^{s^T} \boldsymbol{1}|_2$ is only between $[-1, r-1]$, and thus the maximum $|PA_i^{s^T} \boldsymbol{1}|_2$ is less than $|PA_i^T \boldsymbol{1}|_2$.

We further address that the two inequalities in 19 and 20 both can be achieved for proper H_i and parameters. So we can conclude that for the worst case of α_i and α_i^s, denoted by α_i^* and α_i^{s*},

$$\alpha_i^* < \alpha_i^{s*}$$

9 Discussion of Substructure Based Attention Space

For a substructure G^S, the fixed attention pattern e is an ideal attention pattern to help to concentrate on the information of this substructure, for example, a uniform attention vector on a benzene ring, or attention on a two-hop neighbor graph attenuating according to the distance to the central node. Traditional GNN is also an example of such a fixed attention pattern, where each node takes uniform attention to its' neighbor nodes. Although we treat this pattern as fixed on the substructure, we denote that the definition is general because the value can be any learnable attention pattern.

The defined attention space construct attention patterns for all meaningful attention on substructures by containing the combination of all the elementary substructure pattern. As this substructure attention patterns are considered important indicative bias, the graph transformer is expected to learn attention patterns in this space to utilize the substructure feature. This definition is also general and expressive, as sufficient patterns can be added to include more general cases.

10 Substructures

The sizes of the different substructures used in this study are 2-hop neighbor, 10-step random walk neighbor, 3-8 circle, 2-6 star, and 4-8 path. For geometric substructures, we use the python package graph-tool for substructure matching. It takes about 3 minutes to match all the geometric substructures of ZINC, and about 2 hours for PCQM4M-LSC. We cache all the substructures for reuse.

11 SubstructureSampling

A greedy sampling algorithm is used for the substructure sampling process. In each step of the iteration, nodes with less coverage have a greater probability of being covered in the next iteration. At the beginning of sampling, we randomly select n_{init} substructures from all the substructures, where different types of substructures are balanced. In each iteration, we first mark nodes that are covered less than threshold $thre_1$ times by substructures that have been sampled already as set N_{left}. Then we calculate cnt_i for each substructure left, which is the number of nodes in N_{left} covered by substructure G_i^S. We select n_{sample} from substructures with top k cnt randomly. The iteration ends when each node is covered at least $thre$ times, or no substructure is left. In the algorithm, n_{init}, k, and n_{sample} are determined by $thre$ and nodes number to speed up the iteration and increase sampling variance, while retaining that nodes are covered as evenly as possible.

12 DFS

We use degree to sort nodes at the beginning of DFS and each step. In the beginning, DFS start from node uniformly sampled from nodes with the least degree. At each step, nodes are sorted according to degree, while nodes with the same degree are permuted randomly. This method keeps the graph encoding permutation invariant, and reduces possible permutations a lot. The drawback of permutation-based encoding is that the number of possible permutations becomes intractable when the graph is large. Fortunately, the substructure size is usually not larger than 10.

13 Related Work of Token Unit

The most suitable basic unit for transformer as a token has been studied in many works in natural language processing and computational vision. For nature language, sentences are separated into sub-words containing a variant number of primary characters (Kudo & Richardson, 2018). In computational vision, while previous works apply transformer directly on image pixels (Parmar et al., 2018; Hu et al., 2019), recent works find it more beneficial to treat patches as tokens (Dosovitskiy

et al., 2020), indicating the importance of proper segmentation of the input. For graph transformers, tokens in most current studies are nodes. We propose to encode substructures together with nodes in the graph as tokens in the transformer.