A Scalable and Effective Alternative to Graph Transformers

Kaan Sancak[1], Zhigang Hua[2], Jin Fang[2], Yan Xie[2], Andrey Malevich[2], Bo Long[2], Muhammed Fatih Balın[1], Ümit V. Çatalyürek[*1]

[1]School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, GA, USA
[2]Meta AI
kaan@gatech.edu, {zhua, fangjin, yanxie, amelevich, bolong}@meta.com, {balin, umit}@gatech.edu

Abstract

Graph Neural Networks (GNNs) have shown impressive performance in graph representation learning, but they face challenges in capturing long-range dependencies due to their limited expressive power. To address this, Graph Transformers (GTs) were introduced, utilizing self-attention mechanism to effectively model pairwise node relationships. Despite their advantages, GTs suffer from quadratic complexity w.r.t. the number of nodes in the graph, hindering their applicability to large graphs. In this work, we present Graph-Enhanced Contextual Operator (GECO), a scalable and effective alternative to GTs that leverages neighborhood propagation and global convolutions to effectively capture local and global dependencies in quasiliniear time. Our study on synthetic datasets reveals that GECO reaches $169\times$ speedup on a graph with 2M nodes w.r.t. optimized attention. Further evaluations on diverse range of benchmarks showcase that GECO scales to large graphs where traditional GTs often face memory and time limitations. Notably, GECO consistently achieves comparable or superior quality compared to baselines, improving the SOTA up to 4.5%, and offering a scalable and effective solution for large-scale graph learning.

Extended version — https://arxiv.org/pdf/2406.12059
Code — https://github.com/kaansancak/GECO

1 Introduction

Graph Neural Networks (GNNs) have been state-of-the-art (SOTA) models for graph representation learning showing superior quality across different tasks spanning node, link, and graph level prediction (Gori, Monfardini, and Scarselli 2005; Scarselli et al. 2009; Kipf and Welling 2017; Zhang and Chen 2018; Zhang et al. 2018). Despite their success, GNNs have fundamental limitations that affect their ability to capture long-range dependencies in graphs. These dependencies refer to nodes needing to exchange information over long distances effectively, especially when the distribution of edges is not directly related to the task or when there are missing edges in the graph (Dwivedi et al. 2022b). This limitation can further lead to information over-squashing caused by repeated propagations within GNNs (Li, Han, and Wu 2018; Alon and Yahav 2020; Topping et al. 2022).

Graph Transformers (GTs) (Dwivedi and Bresson 2020; Ying et al. 2021; Wu et al. 2021) were introduced to overcome the limitations of GNNs by incorporating the self-attention mechanism (Vaswani et al. 2017), and achieved SOTA across various benchmarks. GTs can model long-range dependencies by attending to potential neighbors among the entire set of nodes. However, GTs suffer from quadratic complexity which stems from that each node needs to attend to every other node, preventing GTs' widespread adoption in large-scale real-world scenarios. As mini-batch sampling methods for GTs remain under-explored, the primary application of GTs has been on smaller datasets, such as molecular ones (Freitas et al. 2021; Hu et al. 2021; Dwivedi et al. 2022b, 2023a). Consequently, exploring novel efficient and high-quality attention replacements remains a crucial research direction to unlock the full potential of GTs for large-scale graphs. Recently, global convolutional language models have emerged as promising alternatives for attention (Romero et al. 2022; Li et al. 2023). Specifically, Hyena (Poli et al. 2023) has demonstrated impressive performance, offering efficient processing of longer contexts with high quality.

In this work, we aim to find an efficient alternative to dense attention mechanisms to scale graph transformers without sacrificing the modeling quality. The key challenge lies in designing an efficient model that can effectively capture both local and long-range dependencies within large graphs. To address this, we propose a novel compact layer called Graph-Enhanced Contextual Operator (GECO), which combines local propagations and global convolutions. The convolution filters in GECO encompass all nodes, serving as a substitute for dense attention in the graph domain. GECO consists of four main components: (1) local propagation to capture local context, (2) global convolution to capture global context with quasilinear complexity, (3) data-controlled gating for context-specific operations on each node, and (4) positional/structural encoder for feature encoding and graph ordering.

Our evaluation has two main objectives: **(O1):** Matching SOTA GT quality on small graph datasets emphasized by the community. **(O2):** Scaling to larger graphs where traditional attention mechanisms are impractical due to computational constraints. Extensive evaluations across diverse benchmarks demonstrate that GECO scales to larger datasets and con-

[*]Amazon Web Services. This publication describes work performed at the Georgia Institute of Technology and is not associated with AWS.

sistently delivers strong quality, often achieving SOTA or competitive results. The main contributions of work include:

- We developed GECO, a compact layer consisting of local and global blocks with quasilinear time complexity. Unlike prior work, GECO is a refined layer without intermediate parameters and nonlinearities between local and global blocks, applying residuals to the layer as a whole.
- To our knowledge, GECO is the first to employ a global convolution model to develop a scalable and effective alternative to self-attention based GTs. Notably, it improves computational efficiency while preserving prediction quality, and in most cases, it leads to improvements.
- We demonstrated that GECO scales to large graphs that are infeasible for self-attention based GTs due to their intrinsic quadratic complexity. GECO further enhances prediction accuracy by up to 4.5% for large graph datasets.
- We demonstrated GECO's ability to capture long-range dependencies, achieving SOTA on the majority of long-range graph benchmark with improvements up-to 4.3%.

2 Background and Related Work

2.1 Graph Neural Networks (GNNs)

A graph $G = (V, E)$ comprises a set of vertices V and edges $E \subseteq V \times V$. $A \in \mathbb{R}^{N\times N}$ is the adjacency matrix and a weighted edge $(u \to v) \subseteq E$ exists between source u and target v if $A_{u,v} \neq 0$. The feature matrix $X^{(0)} \in \mathbb{R}^{N\times d^{(0)}}$ maps v to a feature vector $x_v^{(0)} \in \mathbb{R}^{d^{(0)}}$. $\mathcal{N}(v) = \{u \mid (u \to v) \in E\}$ is the incoming neighbors of v. GNNs adopt an Aggregate-Combine framework (Hamilton, Ying, and Leskovec 2017) to compute layer-l representation $h_v^{(l)}$:

$$h_v^{(l)} = \text{Combine}^{(l)}\big(\alpha_v^{(l)}, h_v^{(l-1)}\big) \tag{1}$$

$$\alpha_v^{(l)} = \text{Aggregate}^{(l)}\big(\big\{h_u^{(l-1)} : u \in \mathcal{N}(v)\big\}\big) \tag{2}$$

Additionally, a pooling function generates graph representation, $h_G = \text{Pool}\big(\big\{h_v^{(L)} | v \in V\big\}\big)$.

Challenges. GNNs efficiently scale to large graphs with linear complexity, $\mathcal{O}(|V| + |E|)$, but they struggle with capturing long-range dependencies, often requiring many hops and nonlinearities for information to traverse distant nodes (Alon and Yahav 2020; Dwivedi et al. 2022b). GTs effectively resolve this via dense pairwise attention.

2.2 Graph Transformers (GTs)

Graph Transformers (GTs) generalize Transformers (Vaswani et al. 2017) to graphs. At the core of Transformer lies the multi-head self-attention (MHA) (Vaswani et al. 2017), which maps the input $H \in \mathbb{R}^{N\times d}$ to $\mathbb{R}^{N\times d}$ as:

$$Attn(H) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right) \tag{3}$$

$$y = \text{SelfAttention}(H) = Attn(H)V \tag{4}$$

here *query* ($Q = HW_q$), *key* ($K = HW_k$), and *value* ($V = HW_v$) are linear projections of the input, $W_q, W_k, W_v \in \mathbb{R}^{d\times d}$. The attention matrix $Attn(H)$ captures the pair-wise similarities of the input.

Based on their *focus of attention*, we group GTs into three: **Sparse GTs** use the adjacency matrix as an attention mask, allowing nodes to pay attention to *their neighbors*, which facilitates weighted neighborhood aggregation (Veličković et al. 2018; Xu et al. 2019). **Layer GTs** use a GNN to generate hop-tokens, followed by MHA on these tokens, where nodes pay attention to *their layer embeddings* (Chen et al. 2023; Fu et al. 2024). While both Sparse and Layer GTs use attention, they still struggle with long-range dependencies as their attention is restricted to fixed number of hops. **Dense GTs** use attention on fully connected graph, enabling nodes to pay attention to *all the other nodes* regardless of their distances. GTs incorporate positional encodings (PE) to provide topological information to otherwise graph-unaware models (Dwivedi and Bresson 2020; Kreuzer et al. 2021; Ying et al. 2021; Chen, O'Bray, and Borgwardt 2022; Zhao et al. 2023; Ma et al. 2023). Refer to the extended version detailed related work.

Challenges. Dense GTs introduce computational and memory bottlenecks due to their increased complexity from $\mathcal{O}(|V| + |E|)$ to $\mathcal{O}(|V|^2)$, restricting their application to large graphs. GraphGPS (Rampasek et al. 2022) offers a modular framework that combines GNNs with a global attention module, including subquadratic Transformer approximations (Zaheer et al. 2020; Kreuzer et al. 2021). Unfortunately, the subquadratic models compromise quality while MHA based ones struggle with scalability. Therefore, finding a subquadratic attention replacement with a good quality remains a challenge, and our work is dedicated to tackle this problem.

Other Related Work. Exphormer (Shirzad et al. 2023) enhances GraphGPS by using attention on expander graphs. (He et al. 2023) generalizes ViT (Dosovitskiy et al. 2020) and MLP-Mixer (Tolstikhin et al. 2021) to graphs. (Zhang et al. 2022) formulates an adversary bandit problem to sample nodes. HSGT (Zhu et al. 2023) learns multi-level hierarchies via coarsening. GOAT (Kong et al. 2023) uses dimensionality reduction to reduce computational cost of MHA. (Diao and Loynd 2023) utilizes additional edge updates.

2.3 Attention Alternatives

Consider an input u of length N and a filter z. A circular convolution can be computed at each position t of the input u, ranging from 0 to $N-1$, as follows:

$$y_t = (u * z)_t = \sum_{i=0}^{N-1} u_i z_{(t-i) \bmod N} \tag{5}$$

where we assume a single-channel input and filter, which can be easily extended to multi-channel. CNNs (LeCun et al. 1998) optimize z_t at every K steps, where K is a fixed filter size. This *explicit* parametrization captures local patterns within every K steps. Alternatively, *implicit* parameterizations represent the filter as a learnable function (Tay et al. 2021; Gu, Goel, and Re 2021; Romero et al. 2022; Fu et al. 2023). Convolutions can be efficiently computed through Fast Fourier Transform (FFT) in quasilinear time, offering a significant advantage.

A convolution is referred to as a global convolution when the filter has the same length as the input. Global convolutional models have demonstrated the ability to capture longer

contexts through pairwise interactions (dot products) at any input position by proper filter parametrization, offering a promising alternative to attention (Gu, Goel, and Re 2021; Romero et al. 2022; Li et al. 2023). Recently, Hyena (Poli et al. 2023) proposed a sequence model that combines short explicit convolutions and global implicit convolutions using a similar global filter design as CKConv and SGConv (Romero et al. 2022; Li et al. 2023), and it stands out by matching Transformer's quality in quasilinear time. Please refer to the extended version for details. However, to the best of our knowledge, there has been no prior work dedicated to designing and utilizing global convolutional models for graphs.

3 Proposed Architecture: GECO

We present Graph-Enhanced Contextual Operator (GECO), a novel compact layer developed to replace dense attention with quasilinear time and memory complexity. It draws inspiration from recent advancements in global convolutional models and offers a promising approach to capture local and global dependencies with subquadratic operators. Unlike Hyena, which focuses on sequences, GECO is designed for graphs, combining local propagations with global convolutions with random permutation strategies. By utilizing the topological information of the adjacency matrix, it effectively captures local dependencies. Furthermore, it introduces a new global convolution filter design for graphs to capture global dependencies.

As illustrated in Figure 1, GECO starts with positional encodings and proceeds through multiple layers of GECO, each followed by a feed-forward neural network (FFN). We introduce the main components in the following subsections.

3.1 Graph Structural/Positional Encodings

Structural and positional encodings play a pivotal role in the realm of GTs. In our approach, we follow the foundational work established in prior literature concerning these encodings (Dwivedi and Bresson 2020; Kreuzer et al. 2021; Ying et al. 2021; Dwivedi et al. 2022a). To seamlessly integrate these encodings with the original input features, we employ a concatenation method. Given a positional/structural encoding matrix $U \in \mathrm{R}^{N \times d_u}$, where d_u represents the encoding dimension, we combine it with the original node features denoted by X. This process results in a new feature matrix X^*, defined as follows: $X^* = [X, U]$. For further details on relative encodings, please refer to the extended version.

3.2 Local Propagation Block (LCB)

Local Propagation Block (LCB) aggregates neighborhood embeddings for each node and concatenates them with the original ones by utilizing the explicit topological information present in the adjacency matrix. This is similar to the traditional feature propagation with a dense skip connection, and *no parameters* are involved. LCB is expressed as follows:

$$h_v^{*(l)} = [h_v^{(l-1)}, \alpha_v^{(l)}] \quad or \quad H^{*(l)} = [H^{(l-1)}, AH^{(l-1)}] \quad (6)$$

where $\alpha_v^{(l)}$ and $h_v^{(l-1)}$ are defined as before. Instead of adding self-edges for each node, we concatenate $\alpha_v^{(l)}$ and $h_v^{(l-1)}$, enabling our model to distinguish node and propagation embeddings. The doubling of dimension at every layer is prevented by FFNs after each GECO block. Moreover, rather than solely relying on $h_v^{(l)}$, local attention mechanisms similar to those found in GAT (Veličković et al. 2018) can be incorporated. Alternative LCB approaches are further discussed in Sec. 4.3.

Proposition 3.1 *LCB can be computed in $\mathcal{O}(N + M)$ using Sparse Matrix Matrix (SpMM) multiplication between $X^{(l)}$ and A in linear time complexity, where $M = |E|$.*

3.3 Global Context Block (GCB)

Efforts have aimed at creating efficient attention alternatives to capture longer contexts via low-rank approximation, factorization, and sparsification, often leading to trade-offs between efficiency and quality (Catania, Spitale, and Garzotto 2023). Meanwhile, recent sequence models opt for linear convolutions or RNNs which offer near-linear time complexity (Gu, Goel, and Re 2021; Fu et al. 2023; Peng et al. 2023; Li et al. 2023; Poli et al. 2023). Building upon the evolving research, for the first time, we explore whether global convolutions can capture global context within graph structures.

However, designed for sequences, many of the global convolutional models lack graph handling capabilities inherently. This leads us to a key question: Can we develop an operator that effectively processes graphs using global convolutions? Our investigation has yielded positive results, leading to Global Context Block (GCB), a novel operator with graph awareness. Below, we highlight the key distinctions and enhancements of GCB compared to Hyena. Furthermore, we provide ablation studies and empirical comparisons that demonstrate significant quality improvements.

Graph-to-sequence: Since we focus on graphs, we arrange both A and X using permutation π and convert them into time-correlated sequences, aligning node IDs with time (t).

All-to-all information flow: As our setup lacks causality, we remove the causal mask from the global convolution kernel. This allows information to flow mutually between all nodes, respecting the natural dynamics of graph data. The non-causal filters are vital because the relationship between nodes is not inherently sequential or unidirectional. Nodes can have mutual or bidirectional influences, and their relationships are not bound by a linear sequence like words in a sentence.

Graph-aware context: (1) The original proposal by (Li et al. 2023) for global convolutional models for sequences involves exponential decay modulation for convolution filters, assigning higher weights to nearby points in the sequence. In contrast, we aim to minimize the impact of the permutation π during model training. Therefore, we treat all nodes equally regardless of their distance under π by eliminating this decay. (2) Unlike Hyena (Poli et al. 2023), GECO does not employ short convolution along the sequence length, as π may not reflect a locality-sensitive order. Instead, GECO utilizes LCB for local dependencies by leveraging the adjacency matrix. In addition, we first apply LCB before generating input projections, which further reduces the number of parameters in comparison to the prior work.

Window of the global convolution: We set the window size for global convolutions to match the number of nodes, en-

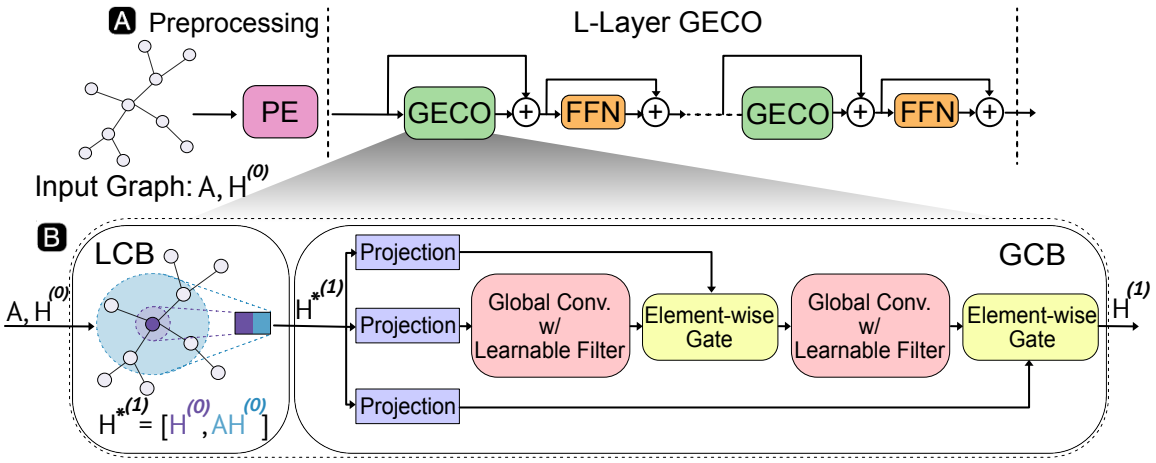

Figure 1: A Our architecture comprises Positional Encoding (PE) block and Graph-Enhanced Contextual Operators (GECOs) layers. PE adds positional encodings as a preprocessing step and each GECO is followed by an FFN. B A GECO layer contains a Local Propagation Block (LCB) aggregating neighborhood embeddings and concatenating with originals to capture local dependencies, and a Global Context Block (GCB) efficiently capturing global dependencies via global convolutions.

suring the inclusion of all nodes within the convolution operation. Without the adjacency matrix, no explicit context is present for graphs. Thus, shorter window lengths hold no meaningful interpretation. This is similarly reasoned by the natural dynamics of graph data where node permutations do not introduce proximity-based context.

Algorithm 1: Forward pass of GCB Operator

Input: Node embeddings $\mathbf{X} \in \mathbb{R}^{N \times d}$; Order K; PE dim d_e;

1. $P_1, \ldots, P_K, V = \mathsf{Projection}(\mathbf{X})$ *# Linear projections* P_i
2. $F_1, \ldots, F_K = \mathsf{Filter}(N, d_e)$ *# Position based filters* F_i

 # Update V *until all projections are exhausted*

for $i = 1, \ldots, K$ **do**

3. In parallel across d: $V_t \leftarrow (P_i)_t \cdot \mathsf{FFTConv}(F_i, V)_t$

end for

4. Return V

Algorithm 1 presents the GCB (notations unified with (Poli et al. 2023)). Given a node embedding matrix X, GCB generates $(K+1)$ projections, where K is a hyperparameter controlling its recurrence. In this work, we set $K = 2$, and in this case, the three projections serve roles similar to query, key, and value. For each projection, a filter is learned by a simple FFN, with node IDs used for filters' positional encoding. Subsequently, the value V is updated using global convolutions with one projection and filter at a time, followed by element-wise multiplication gating, until all projections are processed. GCB is formally expressed as:

$$y = v \odot (f_q * (q \odot (f_k * k))) \tag{7}$$

We assume single-channel features and omit layer notations for simplicity. $q, k, v \in \mathbb{R}^{N \times 1}$ are linear projections of the input, and $f_k, f_q \in \mathbb{R}^{N \times 1}$ are learnable filters with circular symmetry. $\odot$ denotes Hadamard product (element-wise multiplication), and $*$ denotes circular convolution.

3.4 Surrogate Attention Analysis

One natural question that arises is why the GCB is a meaningful replacement for GT's self-attention. To answer this, we can rewrite the attention matrix as $Attn(H) = \mathrm{Softmax}\left(\frac{HW_Q(HW_K)^T}{\sqrt{d}}\right)$ and interpret it as a normalized adjacency matrix, where the pairwise similarity scores are edge weights learned through the attention mechanism. GCB with its modified filter design also learns a surrogate attention matrix that can be interpreted as an adjacency matrix that takes the global context into account. However, it is computed efficiently without storing the entire dense matrix, enabling scaling to larger datasets using similar compute resources. For details, refer to the extended version.

Proposition 3.2 *GCB computes a surrogate attention matrix in* $\mathcal{O}(N \log N)$ *by using Fast Fourier Transform (FFT) and element-wise multiplication.*

3.5 Pitfalls of Permutation Sensitivity and Mitigation

The GECO has certain pitfalls in terms of permutation sensitivity. While typical GNNs use permutation invariant functions (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017), GCB's short and global convolutions are shift-invariant but not permutation invariant. Importantly, a line of research focuses on order-sensitive GNNs (Murphy et al. 2019b,a; Chen et al. 2020; Sato, Yamada, and Kashima 2021; Huang et al. 2022; Chatzianastasis et al. 2023) for enhanced expressibility. Notably GraphSAGE (Hamilton, Ying, and Leskovec 2017) and (Moore and Neville 2017) with LSTM have shown outperforming results. By replacing short convolutions with LCB, we make the local mixing permutation invariant. However, global convolutions remain order-sensitive. To mitigate GCB's permutation sensitivity, we have explored different random permutation strategies.

Static Random: We randomly permute the graph once before training as a naive baseline and compare the performance

variations between different runs. Surprisingly, we observed that the final results are not significantly impacted by different orderings, which we elaborate in Section 4.3.

Dynamic Random: Consider parametrized function $\vec{f}$ with parameters W, and permutation π. With $N!$ permutations sampled, a permutation-sensitive function can recover the original target permutation function $\overline{\vec{f}}$ (Murphy et al. 2019b):

$$\overline{\vec{f}}(X;W) = \frac{1}{N!}\sum_{\pi\in\Pi_N}\vec{f}(A_\pi, X_\pi; W) \tag{8}$$

However, $N!$ is intractable for large graphs, so one option is to sample permutations during training. Consequently, we sample a random permutation per epoch per layer during model training. Similar strategies have been also used for positional encodings, such as random sign flipping for Laplacian PE (Dwivedi et al. 2023a). This helps model to see many different permutations during training, potentially memorize permutation invariance and gain robustness to different permutations. (Murphy et al. 2019b) further proves such strategies approximates $\overline{\vec{f}}$ with decreasing variance as more permutations are sampled.

Proposition 3.3 *GECO with dynamic random sampling strategy is an approximate solution to original target permutation invariant function.*

While we observe robustness to different orderings, understanding and addressing this limitation is crucial for broadening GECO's applicability.

Algorithm 2: End-to-end GECO Model Training

Input: Adj. matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$; Node features $\mathbf{X} \in \mathbb{R}^{N\times d}$; Edge features $\mathcal{E} \in \mathbb{R}^{M\times d_e}$;

1. $\mathbf{X}, \mathbf{A} = GraphPositionalEncoder(\mathbf{X}, \mathbf{A}, \mathcal{E})$
for $\ell = 0, \ldots, L-1$ **do**
2. $\pi = SamplePermutation()$
3. $\mathbf{X}^{(0)} = Permute(\mathbf{A}, \mathbf{X}, \pi)$
4. $\mathbf{X}^{(l+1)} = LayerNorm(GECO(\mathbf{X}^{(l)}, \mathbf{A}) + \mathbf{X}^{(l)})$
5. $\mathbf{X}^{(l+1)} = LayerNorm(FFN(\mathbf{X}^{(l+1)}) + \mathbf{X}^{(l+1)})$
end for
5. Return $\mathbf{X}^L \in \mathbb{R}^{N\times D}$

Algorithm 2 presents the end-to-end training with dynamic permutation. We start by positional encodings. The training is further broken into two main blocks. LCB propagates neighborhood embeddings and applies normalization, which is followed by GCB. Each GECO is followed by an FFN, such that $\mathrm{FFN}(X) = \sigma(XW_1)W_2$, where $W_1, W_2 \in \mathbb{R}^{d\times d}$ are the linear layer weights. Both the GECO and FFN use skip connections, normalization, and dropout. Alternatively, line 2 can be moved out of the for loop to achieve static permutation strategy to order the nodes as a preprocessing step. GECO uses three quasilinear operators and can be computed in $\mathcal{O}(N\log N + M)$. For the complete algorithm and complexity analysis, refer to the extended version.

3.6 Comparison with Prior Work

Hybrid Approaches. Prior works (Wu et al. 2021; Dwivedi and Bresson 2020; Lin, Wang, and Liu 2021; Min et al. 2022) straightforwardly combine GNNs and Transformers as separate local and global modules. In contrast, GECO's LCB and GCB are not auxiliary modules but integrated components of a new compact layer design, refining the model by removing intermediate parameters and non-linearities. This design uses skip connections for the entire layer rather than separate components. Please refer to the extended version for details.

NAgphormer's Hop2Token is a preprocessing step decoupled from model training, where feature propagation iterations are performed to generate node tokens (Chen et al. 2023). Such decoupling methods separate training from feature propagation, hindering model to learn complex relationships between consecutive layers (Wu et al. 2019). In contrast, LCB's feature propagation is coupled with learnable parameters of GCB. LCB is not a preprocessing step but rather an integral pre-step to GCB during model training. In the extended version, we further discuss NAgphormer's recovery as a specific JKNets (Xu et al. 2018) instance.

Graph-Mamba (Wang et al. 2024) has recently adapted Mamba (Gu and Dao 2023) for graphs. It focuses on node ordering strategies based on prioritization while using off-the-shelf permutation-sensitive components. In contrast, we refine the layer design, introduce LCB, aim to mitigate permutation-sensitivity, and further incorporate random permutation strategies for improved robustness. Notably, our evaluation also targets large node prediction datasets, unlike Graph-Mamba's focus on small graph-level tasks.

Orthogonal research: (1) *Model-agnostic methods:* Universally applicable feature encoding and initialization/tuning methods (Dwivedi and Bresson 2020; Kreuzer et al. 2021; Ying et al. 2021; Mialon et al. 2021; Chen, O'Bray, and Borgwardt 2022; Zhao et al. 2023; Tönshoff et al. 2023; Ma et al. 2023). (2) *Scaling methods* such as GOAT (Kong et al. 2023), HSGT (Zhu et al. 2023), and LargeGT (Dwivedi et al. 2023b) that leverage self-attention. GECO offers an alternative kernel combinable with these methods. Section 4.3 provides evidence on when this combination is beneficial based on input size. While Graph-ViT/MLP-Mixer propose alternatives, they are limited to graph-level tasks and require graph re-partitioning at every epoch, which can be costly for large node-level tasks. In contrast, GECO does not require partitioning and is applicable to node-level tasks as well.

4 Experiments

4.1 Objective 1: Prediction Quality

We assess the GECO on benchmarks outlined in Table **??**, where each dataset contains many small graphs, with an average number of nodes ranging from tens to five hundred. Consequently, **scalability is not a significant concern** for these datasets as the computational load is determined by the average number of nodes. As evidence, even the most computation-intensive GTs, such as Graphormer, can be trained on these datasets using Nvidia-V100 (32GB) or Nvidia-A100 (40GB) GPUs (Ying et al. 2021; Rampasek et al. 2022). The experiments in this section aim to demonstrate GECO's competitive predictive quality compared to GT baselines, as many of them encounter memory or time issues with larger graphs. Nevertheless, for these evaluations,

Model	PascalVOC-SP	COCO-SP	Peptides-func	Peptides-struct	PCQM-Contact
	F1 score ↑	F1 score ↑	AP ↑	MAE ↓	MRR ↑
GCN	0.1268 ± 0.0060	0.0841 ± 0.0010	0.5930 ± 0.0023	0.3496 ± 0.0013	0.3234 ± 0.0006
GINE	0.1265 ± 0.0076	0.1339 ± 0.0044	0.5498 ± 0.0079	0.3547 ± 0.0045	0.3180 ± 0.0027
GatedGCN	0.2873 ± 0.0219	0.2641 ± 0.0045	0.5864 ± 0.0077	0.3420 ± 0.0013	0.3218 ± 0.0011
GatedGCN+RWSE	0.2860 ± 0.0085	0.2574 ± 0.0034	0.6069 ± 0.0035	0.3357 ± 0.0006	0.3242 ± 0.0008
Transformer+LapPE	0.2694 ± 0.0098	0.2618 ± 0.0031	0.6326 ± 0.0126	0.2529 ± 0.0016	0.3174 ± 0.0020
SAN+LapPE	0.3230 ± 0.0039	0.2592 ± 0.0158	0.6384 ± 0.0121	0.2683 ± 0.0043	**0.3350 ± 0.0003**
SAN+RWSE	0.3216 ± 0.0027	0.2434 ± 0.0156	0.6439 ± 0.0075	0.2545 ± 0.0012	0.3341 ± 0.0006
GPS w/ Transformer	**0.3748 ± 0.0109**	**0.3412 ± 0.0044**	**0.6535 ± 0.0041**	**0.2500 ± 0.0005**	0.3337 ± 0.0006
Exphormer	**0.3975 ± 0.0037**	**0.3455 ± 0.0009**	**0.6527 ± 0.0043**	**0.2481 ± 0.0007**	**0.3637 ± 0.0020**
GECO (Ours)	**0.4210 ± 0.0080**	**0.3320 ± 0.0032**	**0.6982 ± 0.0045**	**0.2464 ± 0.0009**	**0.3526 ± 0.0016**

Table 1: LRGB: the **first**, **second**, and **third** are highlighted, with results reused from (Rampasek et al. 2022; Shirzad et al. 2023).

PCQM4Mv2	GCN	GCN-virtual	GIN-virtual	GRPE	EGT	Graphormer	GPS-sm	GPS-med	GECO
Train MAE ↓	n/a	n/a	n/a	n/a	n/a	0.0348	0.0653	0.0726	0.0578
Val. MAE ↓	0.1379	0.1153	0.1083	0.0890	0.0869	**0.0864**	0.0938	**0.0858**	**0.0841**
# Param.	2.0M	4.9M	6.7M	46.2M	89.3M	48.3M	6.2M	19.4M	6.2M

Table 2: PCQM4Mv2 Eval.: the **first**, **second**, and **third** are highlighted. *Validation* set is used for evaluation as *test* set is private. We reuse results from (Rampasek et al. 2022).

we begin by creating a hybrid GNN+GECO by replacing the attention module used in GraphGPS. For dataset and hyperparameter details please refer to the extended version.

Long Range Graph Benchmark (LRGB). Table 1 presents our evaluation on the LRGB, a collection of graph tasks designed to test a model's ability to capture long-range dependencies. The results show that GECO outperforms baselines across most datasets, with improvements up-to 4.3%. For the remaining datasets, it ranks among the top three, with quality within 1.3% of the best baseline. By capturing long-range dependencies effectively, GECO surpasses the performance of MHA in most cases without compromising quality. Notably, GECO's F1 score on PascalVOC increased from 0.4053 to 0.4210 without positional encodings, resulting in enhanced quality with a simplified model.

PCQM4Mv2. Table 2 shows that GECO outperforms both GNN and GT baselines on PCQM4Mv2 in prediction quality. Notably, GECO uses only **1/8** and **1/3** of the parameters required by Graphormer and GraphGPS, respectively.

4.2 Objective 2: Scalability for Larger Graphs

We assess GECO on 4 benchmark datasets where each graph contains a much larger number of nodes. Notably, traditional Dense GTs struggle to handle such large graphs due to their quadratic complexity while GECO succeeds with its superior computational and memory efficiency. In the following experiments, we design our models using only GECO blocks, following Algorithm 2. For simplicity, we avoid using structural/positional encodings as computing them may be infeasible for large graphs. For details on datasets and hyperparameters, please refer to the extended version.

Unlike previous works that exhibit a trade-off between quality and scalability, GECO scales efficiently to large datasets and achieves superior quality across all compared to Dense GTs (Graphormer/GraphGPS), which suffer from OOM/timeout issues. Remarkably, GECO demonstrates significant predictive superiority, surpassing Dense GT baseline methods by up to 4.5%. On Arxiv, GECO outperforms recently proposed GT works Exphormer and GOAT (Kong et al. 2023) up to 0.7%. Notably, Graphormer with sampling falls short in achieving competitive quality across all datasets. When comparing GECO to various baselines, including orthogonal methods, GECO remains competitive. It outperforms various baselines on Flickr, Arxiv, and Reddit, except for Yelp where the coarsening approach HSGT (Zhu et al. 2023) surpasses GECO. We leave the exploration of combining GECO with orthogonal methods such as expander graphs (Shirzad et al. 2023), hierarchical learning (Zhu et al. 2023), and dimensionality reduction (Kong et al. 2023) as future work to potentially get even better results. Overall, the results highlight that the global context can enhance the modeling quality for large node prediction datasets, justifying our motivation to find efficient high-quality attention alternatives. To the best of our knowledge, GECO is the first attempt to capture pairwise node relations without heuristics at scale. Our evaluation illustrates its effectiveness as a Dense GT alternative for large graphs.

4.3 Ablation Studies

Permutation Robustness. In Table 4, we investigate GECO's robustness to different permutations using the static and dynamic random strategies outlined in Section 3.5. The results indicate negligible differences between different strategies with dynamic random showing slightly higher mean on multiple datasets. In addition, we note that these strategies has negligible overhead in training time. However, all strategies seems to fall into similar confidence intervals, hence we favor the simpler strategy in our experiments.

Hyena Comparison. Table 4 compares GECO with the off-the-shelf Hyena by setting its filter size as the entire graph. GECO consistently outperforms the off-the-shelf Hyena with

Model Accuracy	Flickr Accuracy	Arxiv Accuracy	Reddit Accuracy	Yelp Micro-F1 Score
GCN	50.90 ± 0.12	70.25 ± 0.22	92.78 ± 0.11	40.08 ± 0.15
SAGE	**53.72 ± 0.16**	72.00 ± 0.16	**96.50 ± 0.03**	**63.03 ± 0.20**
GraphSaint	51.37 ± 0.21	67.95 ± 0.24	95.58 ± 0.07	29.42 ± 1.32
Cluster-GCN	49.95 ± 0.15	68.00 ± 0.59	95.70 ± 0.06	56.39 ± 0.64
GAT	50.70 ± 0.32	71.59 ± 0.38	**96.50 ± 0.11**	61.58 ± 1.37
Graphormer	OOM	OOM	OOM	OOM
Graphormer-SAMPLE	51.93 ± 0.21	70.43 ± 0.20	93.05 ± 0.22 5	60.01 ± 0.45
SAN	OOM	OOM	OOM	OOM
SAT	OOM	OOM	OOM	OOM
SAT-SAMPLE	50.48 ± 0.34	68.20 ± 0.46	93.37 ± 0.32	60.32 ± 0.65
ANS-GT	–	68.20 ± 0.46	95.30 ± 0.81	–
GraphGPS w/ Transformer	OOM	OOM	OOM	OOM
Exphormer	52.60 ± 0.18	**72.44 ± 0.28**	95.90 ± 0.15	60.80 ± 1.56
HSGT	**54.12 ± 0.51**	**72.58 ± 0.31**	–	**63.47 ± 0.45**
GECO (Ours)	**55.55 ± 0.25**	**73.10 ± 0.24**	**96.65 ± 0.05**	**63.18 ± 0.59**

Table 3: Accuracy on large node prediction datasets: the **first**, **second**, and **third** are highlighted. We reuse the results from (Han et al. 2023; Shirzad et al. 2023; Zeng et al. 2021), and run Exphormer locally except Arxiv. – indicates that the data was either not included in the original work or could not be successfully reproduced.

Dataset Perm.	Hyena Static	GECO Static	GECO Dynamic
Flickr	46.97 ± 0.08	55.73 ± 0.27	55.80 ± 0.38
Arxiv	56.04 ± 0.61	73.08 ± 0.28	73.12 ± 0.22
Reddit	69.24 ± 0.54	96.62 ± 0.05	96.68 ± 0.06
Yelp	50.08 ± 0.31	63.23 ± 0.50	63.20 ± 0.42

Table 4: Accuracy with different permutation strategies (Natural/Static Random/Dynamic Random) with GECO, alongside a comparison with default Hyena (Poli et al. 2023).

a significant margin. This underscores the efficacy of GECO, particularly in its application of global convolutions for graphs, and distinctly sets its apart from the Hyena.

Model	Local Block	Pas.VOC-SP F1 ↑	Pep.-func AP ↑	Pep.-struct MAE ↓
Transformer	N/A	**0.2762**	0.6333	**0.2525**
Performer	N/A	0.2690	0.5881	0.2739
GECO	Conv-1	**0.2752**	0.6589	0.2587
GECO	Conv-10	0.1757	**0.6819**	**0.2516**
GECO	Conv-20	0.1645	**0.6706**	0.2534
GECO	Conv-40	0.1445	0.6517	0.2547
GECO	LCB	**0.3220**	**0.6876**	**0.2454**

Table 5: Ablation study on the LCB alternatives: the **first**, **second**, and **third** are highlighted. Conv-x indicates 1D Convolution with a filter size of x.

Local Propagation Block Alternatives. In GECO, we adopted LCB for graph-aware local context modeling instead of using 1D convolutions originally used in Hyena. This is motivated by the limitation of 1D convolutions in capturing local dependencies in graphs where node order does not imply proximity. At Table 5, we focus on exploring alternatives to LCB within our GECO module. We experimented with replacing LCB with 1D convolutions of various filter sizes to help understand its effectiveness. We consistently observed a diminishing trend in quality as filter sizes increased, which can be attributed to larger filter sizes leading to a mix of unrelated nodes within the graph. In contrast, GECO with LCB consistently outperformed its alternatives as well as the Transformer and Performer, highlighting its effectiveness in capturing local graph dependencies.

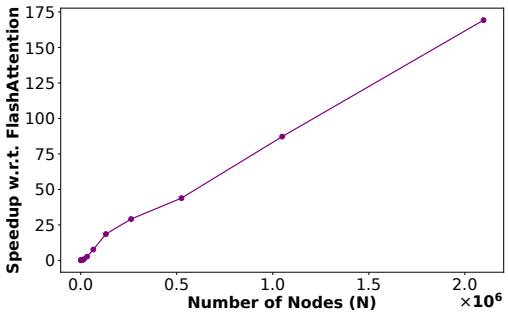

Figure 2: Relative speedup of GECO w.r.t. FlashAttention (Dao et al. 2022) characterized by $\mathcal{O}(N/\log N)$

Scaling Study. Figure 2 shows GECO's speedup w.r.t. the optimized attention, FlashAttention (Dao et al. 2022), for increasing numbers of nodes using synthetic datasets with similar sparsity patterns to those in Table 3. The results highlight that the speedup linearly increases with the number of nodes, and GECO reaches 169× speedup on a graph with 2M nodes, confirming its relative scalability. Details including runtime numbers can be found in the extended version.

5 Conclusion

We presented GECO, a novel model that replaces the compute-intensive MHA in GTs with an efficient and high-quality operator. With comprehensive evaluation, we demonstrated GECO effectively scales to large datasets with outperforming quality. We plan to explore alternatives for GCB, and combinations with orthogonal approaches for future work.

Acknowledgements

This work was partially done when the first author was a research scientist intern at Meta AI. This work was also partially supported by the NSF grant CCF-1919021.

References

Alon, U.; and Yahav, E. 2020. On the Bottleneck of Graph Neural Networks and its Practical Implications. In *International Conference on Learning Representations*.

Catania, F.; Spitale, M.; and Garzotto, F. 2023. Conversational Agents in Therapeutic Interventions for Neurodevelopmental Disorders: A Survey.

Chatzianastasis, M.; Lutzeyer, J.; Dasoulas, G.; and Vazirgiannis, M. 2023. Graph ordering attention networks. In *AAAI Conference on Artificial Intelligence*.

Chen, D.; O'Bray, L.; and Borgwardt, K. 2022. Structure-aware transformer for graph representation learning. In *International Conference on Machine Learning*.

Chen, J.; Gao, K.; Li, G.; and He, K. 2023. NAGphormer: A Tokenized Graph Transformer for Node Classification in Large Graphs. In *The International Conference on Learning Representations*.

Chen, Z.; Chen, L.; Villar, S.; and Bruna, J. 2020. Can graph neural networks count substructures? *Advances in neural information processing systems*.

Dao, T.; Fu, D. Y.; Ermon, S.; Rudra, A.; and Ré, C. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In *Advances in Neural Information Processing Systems*.

Diao, C.; and Loynd, R. 2023. Relational Attention: Generalizing Transformers for Graph-Structured Tasks. In *International Conference on Learning Representations*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv*.

Dwivedi, V. P.; and Bresson, X. 2020. A generalization of transformer networks to graphs. *arXiv*.

Dwivedi, V. P.; Joshi, C. K.; Luu, A. T.; Laurent, T.; Bengio, Y.; and Bresson, X. 2023a. Benchmarking Graph Neural Networks. *Journal of Machine Learning Research*.

Dwivedi, V. P.; Liu, Y.; Luu, A. T.; Bresson, X.; Shah, N.; and Zhao, T. 2023b. Graph Transformers for Large Graphs. *arXiv*.

Dwivedi, V. P.; Luu, A. T.; Laurent, T.; Bengio, Y.; and Bresson, X. 2022a. Graph Neural Networks with Learnable Structural and Positional Representations. In *International Conference on Learning Representations*.

Dwivedi, V. P.; Rampášek, L.; Galkin, M.; Parviz, A.; Wolf, G.; Luu, A. T.; and Beaini, D. 2022b. Long range graph benchmark. *Advances in Neural Information Processing Systems*, 35: 22326–22340.

Freitas, S.; Dong, Y.; Neil, J.; and Chau, D. H. 2021. A Large-Scale Database for Graph Representation Learning. In *Advances in Neural Information Processing Systems*.

Fu, D.; Hua, Z.; Xie, Y.; Fang, J.; Zhang, S.; Sancak, K.; Wu, H.; Malevich, A.; He, J.; and Long, B. 2024. VCR-Graphormer: A Mini-batch Graph Transformer via Virtual Connections. In *International Conference on Learning Representations*.

Fu, D. Y.; Dao, T.; Saab, K. K.; Thomas, A. W.; Rudra, A.; and Re, C. 2023. Hungry Hungry Hippos: Towards Language Modeling with State Space Models. In *International Conference on Learning Representations*.

Gori, M.; Monfardini, G.; and Scarselli, F. 2005. A new model for learning in graph domains. In *IEEE International Joint Conference on Neural Networks, 2005.*

Gu, A.; and Dao, T. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv*.

Gu, A.; Goel, K.; and Re, C. 2021. Efficiently Modeling Long Sequences with Structured State Spaces. In *International Conference on Learning Representations*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Han, X.; Zhao, T.; Liu, Y.; Hu, X.; and Shah, N. 2023. MLPInit: Embarrassingly Simple GNN Training Acceleration with MLP Initialization. In *International Conference on Learning Representations*.

He, X.; Hooi, B.; Laurent, T.; Perold, A.; LeCun, Y.; and Bresson, X. 2023. A generalization of vit/mlp-mixer to graphs. In *International Conference on Machine Learning*.

Hu, W.; Fey, M.; Ren, H.; Nakata, M.; Dong, Y.; and Leskovec, J. 2021. OGB-LSC: A Large-Scale Challenge for Machine Learning on Graphs. In *Advances in Neural Information Processing Systems*.

Huang, Z.; Wang, Y.; Li, C.; and He, H. 2022. Going deeper into permutation-sensitive graph neural networks. In *International Conference on Machine Learning*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*.

Kong, K.; Chen, J.; Kirchenbauer, J.; Ni, R.; Bruss, C. B.; and Goldstein, T. 2023. GOAT: A Global Transformer on Large-scale Graphs. In *International Conference on Machine Learning*. PMLR.

Kreuzer, D.; Beaini, D.; Hamilton, W. L.; L., V.; and Tossou, P. 2021. Rethinking Graph Transformers with Spectral Attention. In *Advances in Neural Information Processing Systems*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*.

Li, Q.; Han, Z.; and Wu, X.-M. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*.

Li, Y.; Cai, T.; Zhang, Y.; Chen, D.; and Dey, D. 2023. What Makes Convolutional Models Great on Long Sequence Modeling? In *International Conference on Learning Representations*.

Lin, K.; Wang, L.; and Liu, Z. 2021. Mesh graphormer. In *IEEE/CVF international conference on computer vision*.

Ma, L.; Lin, C.; Lim, D.; Romero-Soriano, A.; Dokania, P. K.; Coates, M.; Torr, P.; and Lim, S.-N. 2023. Graph inductive biases in transformers without message passing. In *International Conference on Machine Learning*.

Mialon, G.; Chen, D.; Selosse, M.; and Mairal, J. 2021. Graphit: Encoding graph structure in transformers. *arXiv*.

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

Moore, J.; and Neville, J. 2017. Deep collective inference. In *AAAI Conference on Artificial Intelligence*.

Murphy, R.; Srinivasan, B.; Rao, V.; and Ribeiro, B. 2019a. Relational pooling for graph representations. In *International Conference on Machine Learning*, 4663–4673. PMLR.

Murphy, R. L.; Srinivasan, B.; Rao, V.; and Ribeiro, B. 2019b. Janossy Pooling: Learning Deep Permutation-Invariant Functions for Variable-Size Inputs. In *International Conference on Learning Representations*.

Peng, B.; Alcaide, E.; Anthony, Q.; Albalak, A.; Arcadinho, S.; Cao, H.; Cheng, X.; Chung, M.; Grella, M.; GV, K. K.; et al. 2023. RWKV: Reinventing RNNs for the Transformer Era. *arXiv*.

Poli, M.; Massaroli, S.; Nguyen, E.; Fu, D. Y.; Dao, T.; Baccus, S.; Bengio, Y.; Ermon, S.; and Ré, C. 2023. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*.

Rampasek, L.; Galkin, M.; Dwivedi, V. P.; Luu, A. T.; Wolf, G.; and Beaini, D. 2022. Recipe for a General, Powerful, Scalable Graph Transformer. In *Advances in Neural Information Processing Systems*.

Romero, D. W.; Kuzina, A.; Bekkers, E. J.; Tomczak, J. M.; and Hoogendoorn, M. 2022. CKConv: Continuous Kernel Convolution For Sequential Data. In *International Conference on Learning Representations*.

Sato, R.; Yamada, M.; and Kashima, H. 2021. Random features strengthen graph neural networks. In *SIAM international conference on data mining (SDM)*, 333–341. SIAM.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2009. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1): 61–80.

Shirzad, H.; Velingker, A.; Venkatachalam, B.; Sutherland, D. J.; and Sinop, A. K. 2023. Exphormer: Sparse transformers for graphs. In *International Conference on Machine Learning*.

Tay, Y.; Dehghani, M.; Abnar, S.; Shen, Y.; Bahri, D.; Pham, P.; Rao, J.; Yang, L.; Ruder, S.; and Metzler, D. 2021. Long Range Arena : A Benchmark for Efficient Transformers. In *International Conference on Learning Representations*.

Tolstikhin, I. O.; Houlsby, N.; Kolesnikov, A.; Beyer, L.; Zhai, X.; Unterthiner, T.; Yung, J.; Steiner, A.; Keysers, D.; Uszkoreit, J.; et al. 2021. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34: 24261–24272.

Tönshoff, J.; Ritzert, M.; Rosenbluth, E.; and Grohe, M. 2023. Where Did the Gap Go? Reassessing the Long-Range Graph Benchmark. In *The Second Learning on Graphs Conference*.

Topping, J.; Giovanni, F. D.; Chamberlain, B. P.; Dong, X.; and Bronstein, M. M. 2022. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Wang, C.; Tsepa, O.; Ma, J.; and Wang, B. 2024. Graph-mamba: Towards long-range graph sequence modeling with selective state spaces. *arXiv*.

Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; and Weinberger, K. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*.

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

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*.

Xu, K.; Li, C.; Tian, Y.; Sonobe, T.; Kawarabayashi, K.-i.; and Jegelka, S. 2018. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, 5453–5462. PMLR.

Ying, C.; Cai, T.; Luo, S.; Zheng, S.; Ke, G.; He, D.; Shen, Y.; and Liu, T.-Y. 2021. Do Transformers Really Perform Badly for Graph Representation? In *Advances in Neural Information Processing Systems*.

Zaheer, M.; Guruganesh, G.; Dubey, K. A.; Ainslie, J.; Alberti, C.; Ontanon, S.; Pham, P.; Ravula, A.; Wang, Q.; Yang, L.; et al. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*.

Zeng, H.; Zhang, M.; Xia, Y.; Srivastava, A.; Malevich, A.; Kannan, R.; Prasanna, V.; Jin, L.; and Chen, R. 2021. Decoupling the Depth and Scope of Graph Neural Networks. In *Advances in Neural Information Processing Systems*.

Zhang, M.; and Chen, Y. 2018. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31.

Zhang, M.; Cui, Z.; Neumann, M.; and Chen, Y. 2018. An end-to-end deep learning architecture for graph classification. In *AAAI conference on artificial intelligence*, volume 32.

Zhang, Z.; Liu, Q.; Hu, Q.; and Lee, C.-K. 2022. Hierarchical Graph Transformer with Adaptive Node Sampling. In *Advances in Neural Information Processing Systems*.

Zhao, H.; Ma, S.; Zhang, D.; Deng, Z.-H.; and Wei, F. 2023. Are More Layers Beneficial to Graph Transformers? In *The International Conference on Learning Representations*.

Zhu, W.; Wen, T.; Song, G.; Ma, X.; and Wang, L. 2023. Hierarchical Transformer for Scalable Graph Learning. *arXiv*.