# EXPHORMER: Sparse Transformers for Graphs

**Hamed Shirzad** [*1] **Ameya Velingker** [*2] **Balaji Venkatachalam** [*3] **Danica J. Sutherland** [1 4] **Ali Kemal Sinop** [2]

## Abstract

Graph transformers have emerged as a promising architecture for a variety of graph learning and representation tasks. Despite their successes, though, it remains challenging to scale graph transformers to large graphs while maintaining accuracy competitive with message-passing networks. In this paper, we introduce EXPHORMER, a framework for building powerful and scalable graph transformers. EXPHORMER consists of a sparse attention mechanism based on two mechanisms: virtual *global nodes* and *expander graphs*, whose mathematical characteristics, such as spectral expansion, pseduorandomness, and sparsity, yield graph transformers with complexity only linear in the size of the graph, while allowing us to prove desirable theoretical properties of the resulting transformer models. We show that incorporating EXPHORMER into the recently-proposed GraphGPS framework produces models with competitive empirical results on a wide variety of graph datasets, including state-of-the-art results on three datasets. We also show that EXPHORMER can scale to datasets on larger graphs than shown in previous graph transformer architectures. Codes can be found at `https://github.com/hamed1375/Exphormer`.

## 1. Introduction

Graph learning has become an important and popular area of study that has yielded impressive results on a wide variety of graphs and tasks, including molecular graphs, social network graphs, knowledge graphs, and more. While much research around graph learning has focused on graph neural networks (GNNs), which are based on local *message-passing*, a more recent approach to graph learning that has garnered much interest involves the use of *graph transformers* (GTs). Graph transformers largely operate by encoding graph structure in the form of a *soft inductive bias*. These can be viewed as a graph adaptation of the Transformer architecture (Vaswani et al., 2017) that are successful in modeling sequential data in applications such as natural language processing.

Graph transformers allow nodes to attend to all other nodes in a graph, allowing for direct modeling of long-range interactions, in contrast to GNNs. This allows them to avoid several limitations associated with local message passing GNNs, such as oversmoothing (Oono & Suzuki, 2020), oversquashing (Alon & Yahav, 2021; Topping et al., 2022), and limited expressivity (Morris et al., 2019; Xu et al., 2018). The promise of graph transformers has led to a large number of different graph transformer models that have been proposed in recent years (Dwivedi & Bresson, 2020; Kreuzer et al., 2021; Ying et al., 2021; Mialon et al., 2021).

One major challenge for graph transformers is their poor scalability, as the standard global attention mechanism incurs time and memory complexity of $O(|V|^2)$, *quadratic* in the number of nodes in the graph. While this cost is often acceptable for datasets with small graphs (e.g., molecular graphs), it can be prohibitively expensive for datasets containing larger graphs, where graph transformer models often do not fit in memory even for high-memory GPUs, and hence would require much more complex and slower schemes to apply. Moreover, despite the expressivity advantages of graph transformer networks (Kreuzer et al., 2021), these architectures have often lagged message-passing counterparts in accuracy in many practical settings.

A recent breakthrough came with the advent of GraphGPS (Rampásek et al., 2022), a modular framework for constructing networks by combining local message passing and a global attention mechanism together with a choice of various positional and structural encodings. To attempt to overcome the quadratic complexity of the "dense" full transformer and improve scalability, the architecture allows for "sparse" attention mechanisms, like Performer (Choro-

[*]Equal contribution [1]Department of Computer Science, University of British Columbia, Vancouver, BC, Canada [2]Google Research, Mountain View, California, USA [3]Google, Mountain View, California, USA [4]Alberta Machine Intelligence Institute, Edmonton, Alberta, Canada. Correspondence to: Hamed Shirzad <shirzad@cs.ubc.ca>, Ameya Velingker <ameyav@google.com>, Balaji Venkatachalam <bave@google.com>.

*Proceedings of the $40^{th}$ International Conference on Machine Learning*, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

manski et al., 2021) or Big Bird (Zaheer et al., 2020). This combination of Transformers and GNNs achieves state-of-the-art performance on a wide variety of datasets.

In almost all cases, however, the best results of Rampásek et al. were obtained by combining a message-passing network with a full transformer; their sparse transformers performed relatively poorly by comparison, and indeed their ablation studies showed that on a number of datasets it is better to avoid using attention at all than to use their implementation of BigBird. This may be related to the fact that sparse attention mechanisms like BigBird have largely been designed for *sequences*; this is natural for language tasks, but graphs behave quite differently. Thus, it is natural to ask whether one can design sparse attention mechanisms more tailored to learning interactions on general graphs.

Another major question concerns graph transformers' scalability. While BigBird and Performer are linear attention mechanisms, they still incur computational overhead that dominates the per-epoch computation time for moderately-sized graphs. The GraphGPS work tackles datasets with graphs of up to 5,000 nodes, a regime in which the full-attention transformer is in fact computationally faster than many sparse linear-attention mechanisms. Perhaps more suitable sparse attention mechanisms could enable their framework to operate on even larger graphs. Additionally, for smaller graphs, they may require a much smaller batch size to fit into the GPU memory, resulting in slower training, or the need for high-end GPUs. For instance, on the MalNet-Tiny dataset, GraphGPS with a full transformer runs out of memory (on a 40GB NVIDIA A100 GPU) with a batch size of just 16; as we shall see, our techniques allow us to extend the batch size to 256 without any memory issues.

**Our contributions.** We propose sparse attention mechanisms with computational cost linear in the number of nodes and edges. We introduce EXPHORMER that combines the two techniques for creating sparse overlay graphs. The first sparse attention mechanism is to use *global nodes* — nodes that are connected to all other nodes of the graph. The number of additional edges added to the graph is linear in the number of nodes. We also introduce *expander graphs* as a powerful primitive in designing scalable graph transformer architectures. The expander graph as an overlay graph has edges linear in the number of nodes. Expander graphs have several desirable properties – small diameter, spectral approximation of a complete graph, good mixing properties – which make them a suitable ingredient in a sparse attention mechanism. We are able to show that EXPHORMER, which combines expander graphs with global nodes and local neighborhoods, spectrally approximates the full attention mechanism with only a small number of layers, and has universal approximation properties. Its attention scheme provides good inductive bias for places the model "should look," in addition to being more efficient and less memory-intensive.

We show that EXPHORMER are a powerful Transformer that produces results comparable to GraphGPS with a full transformer. Moreover, when combined with MPNNs in the GraphGPS framework, we can achieve SOTA or close to SOTA. EXPHORMER (1) produces better results than other sparse transformers on all datasets we evaluate; (2) has results comparable to and in some cases better than the full transformer, despite having fewer parameters; (3) achieves state-of-the-art results on many datasets, better than MPNNs, including on Long Range Graph Benchmark (LRGB; Dwivedi et al., 2022) datasets that require long-range dependencies between nodes; and (4) can scale to larger graphs than previously shown.

## 2. Related Work

**Graph Neural Networks (GNNs).** Early works in the area of graph learning and GNNs include the development of a number of architectures such as GCN (Defferrard et al., 2016; Kipf & Welling, 2017), GraphSage (Hamilton et al., 2017), GIN (Xu et al., 2018), GAT (Veličković et al., 2018), GatedGCN (Bresson & Laurent, 2017), and more. GNNs are based on a message-passing architecture that generally confines their expressivity to the limits of the 1-Weisfeiler-Lehman (1-WL) isomorphism test (Xu et al., 2018).

A number of recent papers have sought to augment GNNs to improve their expressivity. For instance, one approach has been to use *additional features* that allow nodes to be distinguished – such as using a one-hot encoding of the node (Murphy et al., 2019) or a random scalar feature (Sato et al., 2021) – or to encode positional or structural information of the graph – e.g., skip-gram based network embeddings (Qiu et al., 2018), substructure counts (Bouritsas et al., 2020), or Laplacian eigenvectors (Dwivedi et al., 2021). Another direction has been to *modify the message passing rule* to allow the network to take further advantage of the graph structure – including directional graph networks (DGN; Beaini et al., 2021) that use Laplacian eigenvectors to define directional flows for anisotropic message aggregation – or to *modify the underlying graph* over which message passing occurs with *higher-order GNNs* (Morris et al., 2019) or the use of substructures such as junction trees (Fey et al., 2020) and simplicial complexes (Bodnar et al., 2021).[1]

**Graph transformer architectures.** Attention mechanisms have been extremely successful in sequence mod-

[1] Deac et al. (2022), in work concurrent to and independent of ours, propose an expander-based graph learning mechanism for message-passing networks, alternating between layers based on the input graph with ones based on an auxiliary expander graph. This scheme is rather different from ours; we do not compare further.

eling since the seminal work of Vaswani et al. (2017). The GAT architecture (Veličković et al., 2018) proposed using an attention mechanism to determine how a node aggregates information from its neighbors; it does not use a positional encoding for nodes, limiting its ability to exploit global structural information. GraphBert (Zhang et al., 2020) uses the graph structure to determine an encoding of the nodes, but not for the underlying attention mechanism.

Graph transformer models typically operate on a fully-connected graph in which every pair of nodes is connected, regardless of the connectivity structure of the original graph. Spectral Attention Networks (SAN) (Kreuzer et al., 2021) make use of *two* attention mechanisms, one on the fully-connected graph and one on the original edges of the input graph, while using Laplacian positional encodings for the nodes. Graphormer (Ying et al., 2021) uses a single dense attention mechanism but adds structural features in the form of centrality and spatial encodings. Meanwhile, GraphiT (Mialon et al., 2021) incorporates relative positional encodings based on diffusion kernels.

GraphGPS (Rampásek et al., 2022) proposed a general framework for combining message-passing networks with attention mechanisms, while allowing for the mixing and matching of positional and structural embeddings. Specifically, the framework also allows for sparse transformer models like BigBird (Zaheer et al., 2020) and Performer (Choromanski et al., 2021).

Moreover, recent works have proposed sampling-based scalable graph transformers, e.g., Gophormer (Zhao et al., 2021), NAGphormer (Chen et al., 2022b). Another line of work aims to learn data dependencies beyond the given graph structure using linear time transformers. For instance, Nodeformer (Wu et al., 2022) is inspired by Performer (Choromanski et al., 2021) and leverages the kernelized Gumbel-Softmax operator to enable the propagation of information among all pairs of nodes in a computationally efficient manner. Another work is Difformer (Wu et al., 2023), which, on the other hand, is a continuous time diffusion-based Transformer model. Even though these models can scale up to millions of nodes, both of them use a random subset of a maximum of 100K nodes as mini-batches, and the attention mechanism takes place only over the nodes in the batch. Transformers are also applied on the spectral GNNs (Bo et al., 2023).

**Sparse Transformers.** Standard (dense) transformers have quadratic complexity in the number of tokens, which limits their scalability to extremely long sequences. By contrast, *sparse transformer* models improve computational and memory efficiency by restricting the attention pattern, i.e., the pairs of nodes that can interact with each other. In addition to BigBird and Performer, there have been a number of other proposals for sparse transformers; Tay et al. (2020) provide a survey.

## 3. Sparse Attention on Graphs

This section describes three *sparse* patterns that can be used in transformers in individual layers of a graph transformer architecture. We begin by describing graph attention mechanisms in general.

### 3.1. Attention mechanism on graphs

An attention mechanism on $n$ tokens can be modeled by a directed graph $H$ on $[n] = \{1, 2, \ldots, n\}$, where a directed edge from $i$ to $j$ indicates a direct interaction between tokens $i$ and $j$, i.e., an inner product that will be computed by the attention mechanism. More precisely, a transformer block can be viewed as a function on the $d$-dimensional embeddings for each of $n$ tokens, mapping from $\mathbb{R}^{d\times n}$ to $\mathbb{R}^{d\times n}$. Let $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) \in \mathbb{R}^{d\times n}$. A generalized (dot-product) attention mechanism $\text{ATTN}_H : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$ with attention pattern given by $H$ is defined by

$$\text{ATTN}_H(\mathbf{X})_{:,i} = \mathbf{x}_i + \sum_{j=1}^{h} \mathbf{W}_O^j \mathbf{W}_V^j \mathbf{X}_{\mathcal{N}_H(i)} \cdot \sigma\left(\left(\mathbf{W}_K^j \mathbf{X}_{\mathcal{N}_H(i)}\right)^T \left(\mathbf{W}_Q^j \mathbf{x}_i\right)\right),$$

where $h$ is the number of heads and $m$ is the head size, while $\mathbf{W}_K^j, \mathbf{W}_Q^j, \mathbf{W}_V^j \in \mathbb{R}^{m\times d}$ and $\mathbf{W}_O^j \in \mathbb{R}^{d\times m}$. (The subscript $K$ is for "keys," $Q$ for "queries," $V$ for "values," and $O$ for "output.") Here $\mathbf{X}_{\mathcal{N}_H(i)}$ denotes the submatrix of $\mathbf{X}$ obtained by picking out only those columns corresponding to elements of $\mathcal{N}_H(i)$, the neighbors of $i$ in $H$. We can see that the total number of inner product computations for all $i \in [n]$ is given by the number of edges of $H$. A (generalized) *transformer block* consists of $\text{ATTN}_H$ followed by a feedforward layer:

$$\text{FF}(\mathbf{X}) = \text{ATTN}_H(\mathbf{X}) + \mathbf{W}_2 \cdot \text{ReLU}(\mathbf{W}_1 \cdot \text{ATTN}_H\left(\mathbf{X}\right) + \mathbf{b}_1 \mathbf{1}_n^T) + \mathbf{b}_2 \mathbf{1}_n^T,$$

where $\mathbf{W}_1 \in \mathbb{R}^{r\times d}$, $\mathbf{W}_2 \in \mathbb{R}^{d\times r}$, $\mathbf{b}_1 \in \mathbb{R}^r$, and $\mathbf{b}_2 \in \mathbb{R}^d$.

In the standard setting, the $n$ tokens are part of a sequence (e.g., language applications). However, we are concerned with the *graph transformer* setting in which the tokens are nodes of some underlying graph $G = (V, E)$ with $V = [n]$. The attention computation is nearly identical, except that one can also optionally augment it with edge features, as is

done in SAN (Kreuzer et al., 2021):

$$\text{ATTN}_H(\mathbf{X})_{:,i} = \mathbf{x}_i + \sum_{j=1}^{h} \mathbf{W}_O^j \mathbf{W}_V^j \mathbf{X}_{\mathcal{N}_H(i)} \cdot \sigma\left(\left(\mathbf{W}_E^j \mathbf{E}_{\mathcal{N}_H(i)} \odot \mathbf{W}_K^j \mathbf{X}_{\mathcal{N}_H(i)}\right)^T \left(\mathbf{W}_Q^j \mathbf{x}_i\right)\right),$$

where $\mathbf{W}_E^j \in \mathbb{R}^{m \times d_E}$, $\mathbf{E}_{\mathcal{N}_H(i)}$ is the $d_E \times |\mathcal{N}_H(i)|$ matrix whose columns are $d_E$-dimensional edge features for the edges connected to node $i$, and $\odot$ denotes element-wise multiplications.

The most typical cases of graph transformers use full (dense) attention, where every token attends to every other node: $H$ is the fully-connected directed graph. As this results in computational complexity $O(n^2)$ for the transformer block, which is prohibitively expensive for large graphs, we wish to replace full attention with a *sparse* attention mechanism, where $H$ has $o(n^2)$ edges – ideally, $O(n)$.

A number of sparse attention mechanisms have been proposed to address the aforementioned issue (see Tay et al., 2020), but the vast majority are designed specifically for functions on *sequences*. EXPHORMER, on the other hand, is a *graph-centric* sparse attention mechanism that makes use of the underlying structure of the input graph $G$. We introduce three sparse patterns: expander graphs, global connectors, and local neighborhoods as three patterns that can be used in transformers. They can combine together to make an EXPHORMER layer, but it is not necessary to have all components in each layer.

GraphGPS (Rampásek et al., 2022) shows an effective way to include the transformer layers beside an MPNN model. We use the same architecture as GraphGPS, replacing only the transformer part with an EXPHORMER layer. We will show that sparse transformers can get comparable and even better results on many benchmarks compared to full transformers, or sparse transformer variants originally designed for sequences.

### 3.2. The EXPHORMER Architecture

We now describe the details of the construction of EXPHORMER, an expander-based sparse attention mechanism for graph transformers with $O(|V| + |E|)$ computation, where $G = (V, E)$ is the underlying input graph. The EXPHORMER architecture constructs an interaction graph $H$ that consists of three main components, as shown in Figure 1. The construction always has bidirectional edges, and so $H$ can be viewed as an undirected graph. The mechanism uses three types of edges:

1. **Expander graph attention**: Edges from a random *expander graph* can be used as attention patterns. These graphs have several useful theoretical properties related to spectral approximation and random walk mixing (see Section 4), which allow propagating information between pairs of nodes that are distant in the input graph $G$ without connecting all pairs of nodes. They introduce many alternative short paths between the nodes and avoid the information bottleneck that can be caused by the virtual nodes. In particular, we use a regular expander graph of constant degree, which allows the number of edges to be just $O(|V|)$. The process we use to construct a random expander graph is described below.

2. **Global attention**: The next component is *global attention*, whereby a small number of virtual nodes are added to the interaction graph, and each such node is connected to all the non-virtual nodes. These nodes enable a global "storage sink" and they have universal approximator functions for full transformers. We will generally add a constant number of virtual nodes, in which case the total number of edges due to global attention will be $O(|V|)$.

3. **Local neighborhood attention**: Another desirable property to capture is *locality*. Graphs carry much more topological structure than sequences, and the neighborhoods of individual nodes carry a lot of information about connectivity. Thus, we model local interactions by allowing each node $v$ to attend to every other node that is an immediate neighbor of $v$ in $G$: that is, $H$ includes the input graph edges $E$ as well as their reverses, introducing $O(|E|)$ interaction edges. One generalization would be to allow direct attention

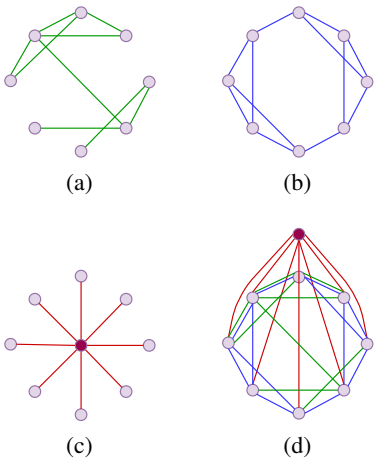


*Figure 1.* The components of EXPHORMER: (a) shows local neighborhood attention, i.e., edges of the input graph. (b) shows an expander graph with degree 3. (c) shows global attention with a single virtual node. (d) All of the aforementioned components are combined into a single interaction graph that determines the attention pattern of EXPHORMER.

*Table 1.* Comparison of EXPHORMER with baselines on various datasets. Best results are colored in **first**, **second**, **third**.

| Model | CIFAR10 Accuracy ↑ | MalNet-Tiny Accuracy ↑ | MNIST Accuracy ↑ | CLUSTER Accuracy ↑ | PATTERN Accuracy ↑ |
|---|---|---|---|---|---|
| GCN (Kipf & Welling, 2017) | 55.71±0.381 | 81.0 | 90.71±0.218 | 68.50 ± 0.976 | 71.89 ± 0.334 |
| GIN (Xu et al., 2018) | 55.26±1.527 | 88.98±0.557 | 96.49±0.252 | 64.72 ± 1.553 | 85.39 ± 0.136 |
| GAT (Veličković et al., 2018) | 64.22±0.455 | 92.1 ±0.242 | 95.54±0.205 | 70.59 ± 0.447 | 78.27 ± 0.186 |
| GatedGCN (Bresson & Laurent, 2017; Dwivedi et al., 2020) | 67.31±0.311 | **92.23±0.65** | 97.34±0.143 | 73.84 ± 0.326 | 85.57 ± 0.088 |
| PNA (Corso et al., 2020) | 70.35±0.63 | – | 97.94±0.12 | – | – |
| DGN (Beaini et al., 2021) | **72.84±0.417** | – | – | – | 86.68±0.034 |
| CRaWl (Toenshoff et al., 2021) | 69.01±0.259 | – | 97.94±0.050 | – | – |
| GIN-AK+ (Zhao et al., 2022b) | 72.19±0.13 | – | – | – | **86.85±0.057** |
| SAN (Kreuzer et al., 2021) | – | – | – | 76.69±0.65 | 86.58±0.037 |
| K-Subgraph SAT (Chen et al., 2022a) | – | – | – | 77.86±0.104 | **86.85±0.037** |
| EGT (Hussain et al., 2021) | 68.70±0.409 | | **98.17±0.087** | **79.23±0.348** | **86.82±0.020** |
| GraphGPS (Rampásek et al., 2022) | **72.30±0.356** | **93.50±0.41** | **98.05±0.126** | **78.02±0.180** | 86.69±0.059 |
| EXPHORMER (ours) | **74.69±0.125** | **94.02 ± 0.209** | **98.55 ± 0.039** | **78.07 ± 0.037** | 86.74±0.015 |

within $k$-hop neighborhoods, but this might introduce a superlinear number of interactions on general graphs.

We use learnable embeddings for expander and global connection edge features, and virtual nodes' features. Dataset edge features are used for the local neighborhood edge features. We always use local neighborhoods in EXPHORMER layers, but expander graphs and global attention are not always helpful; depending on the dataset, we may use both, or only one or the other. We discuss this further in Appendix D.

Some previous graph-oriented transformers, such as the SAN architecture (Kreuzer et al., 2021), use separate attention mechanisms for different sources of edges. By using a single attention mechanism, EXPHORMER achieves a more compact model that can still distinguish the "types" of attention edges based on edge features.

**Generating a Random Regular Expander** We now describe the means by which we generate a random regular expander. Let $G = (V, E)$ be the original graph, where $V = \{1, 2, \ldots, n\}$. For the purposes of experimentation (in Tables 1 to 5), we use the random graph process analyzed in Friedman (2003) (see Theorem C.2 in Appendix C) to generate a random $d$-regular graph $G' = (V, E')$ on the same node set $V$ as follows:

- Pick $d/2$ permutations $\pi_1, \pi_2, \ldots, \pi_{d/2}$ on $V$, each $\pi_i$ chosen independently and uniformly among all $n$ permutations.
- Then, choose

$$E' = \left\{(i, \pi_j(i)), (i, \pi_j^{-1}(i)) : j \in [d/2], i \in [n]\right\},$$

where $[k]$ denotes $\{1, 2, \ldots, k\}$.

The above process works for even $d$, and Theorem C.2 shows that the resulting graph will be a $d$-regular near-Ramanujan graph with high probability. In practice, we will generate expander graphs using this process and throw out any graphs that fail to be near-Ramanujan, according to a desired threshold (this is a low probability event).

We provide further details in Appendix C, where we also discuss other algorithms for generating expander graphs.

## 4. Theoretical Properties of EXPHORMER

EXPHORMER is based on expander graphs, which have a number of properties that make them suitable as a key building block of our approach. In this section, we describe relevant properties of expander graphs along with their implications for EXPHORMERs.

### 4.1. Expander Graphs Approximate Complete Graphs

For simplicity, let us consider $d$-*regular* graphs (where every node has $d$ neighbors). Suppose $G$ is a $d$-regular undirected graph on $n$ vertices. Let $A_G$ be the $n \times n$ adjacency matrix of $G$. It is known that $A_G$ has $n$ real eigenvalues $d = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq -d$. The graph $G$ is said to be an $\epsilon$-*expander* if $\max\{|\lambda_2|, |\lambda_n|\} \leq \epsilon d$ (Hoory et al., 2006).

Expander graphs are sparse approximations of complete graphs. As we discuss, expander graphs with only $O(n)$ edges exist that can preserve certain desirable properties of the complete graph with $\Theta(n^2)$ edges.

#### 4.1.1. SPECTRAL PROPERTIES

A useful tool to study expanders is the *Laplacian* matrix of a graph, which captures several important spectral properties. Letting $D_G$ denote the $n \times n$ diagonal matrix whose $i$-th diagonal entry is the degree of the $i$-th node, we define $L_G = D_G - A_G$ to be the Laplacian of $G$. The following theorem is well-known in spectral graph theory.

**Theorem 4.1.** *(Spielman, 2019, Section 27.2) A $d$-regular $\epsilon$-expander $G$ on $n$ vertices spectrally approximates the*

*complete graph $K_n$ on $n$ vertices:*[2]

$$(1-\epsilon)\frac{1}{n}L_K \preceq \frac{1}{d}L_G \preceq (1+\epsilon)\frac{1}{n}L_K.$$

Spectral approximation is known to preserve the cut structure in graphs. As a result, a sparse attention mechanism based on expander edges retains spectral properties of the full attention mechanism: cuts, vertex expansion, and so on.

#### 4.1.2. MIXING PROPERTIES

Another property of expanders is that random walks mix well. Let $G = (V, E)$ be a $d$-regular $\epsilon$-expander. Consider a random walk $v_0, v_1, v_2, \ldots$ on $G$, where $v_0 \sim \pi^{(0)}$, and then each subsequent $v_{t+1}$ is one of the $d$ neighbors of $v_t$ chosen uniformly at random. We then have $v_t \sim \pi^{(t)}$, given recursively by $\pi^{(t+1)} = D_G^{-1} A_G \pi^{(t)}$. It turns out that after a logarithmic number of steps, a random walk from a starting probability distribution on the vertices is close to uniformly distributed along all nodes of the graph.

**Lemma 4.2.** *(Hoory et al., 2006, Theorem 3.2) Let $G = (V, E)$ be a $d$-regular $\epsilon$-expander graph on $n = |V|$ nodes. For any initial distribution $\pi^{(0)} : V \to \mathbb{R}^+$ and any $\delta > 0$, $\pi^{(t)}$ satisfies*

$$\|\pi^{(t)} - \tfrac{1}{n}\|_1 \leq \delta$$

*as long as $t \geq \frac{1}{2(1-\varepsilon)} \log(n/\delta^2)$.*

In an attention mechanism of a transformer, one can consider the graph of pairwise interactions (i.e., $i$ is connected to $j$ if $i$ and $j$ attend to each other). If the attention mechanism is dense, then each node is connected to every other node and it is trivial to see that every pair of nodes interacts with each other in a single transformer layer. In a *sparse* attention mechanism, on the other hand, some pairs of nodes are not directly connected, meaning that a single transformer layer will not model interactions between all pairs of nodes. However, if we stack transformer layers on top of each other, the stack will be able to model longer range interactions. In particular, a consequence of the above lemma is that if our sparse attention mechanism is modeled after an $\epsilon$-expander graph, then stacking at least $t = \frac{1}{2(1-\epsilon)} \log(n/\delta^2)$ layers will model "most" pairwise interactions between nodes.

Relatedly, the diameter of expander graphs is asymptotically logarithmic in the number of nodes.

**Theorem 4.3.** *(Hoory et al., 2006, Section 2.4) Suppose $G = (V, E)$ is a $d$-regular $\epsilon$-expander graph on $n$ vertices. Then, for every vertex $v$ and $k \geq 0$, the $k$-hop neighborhood $B(v, r) = \{w \in V : \mathrm{dist}(v, w) \leq k\}$ has*

$$|B(v, r)| \geq \min\{(1 + c)^k, n\}$$

*for some constant $c > 0$ depending on $d, \epsilon$. In particular, we have that $\mathrm{diam}(G) = O_{d,\epsilon}(\log n)$.*

As a consequence, we obtain the following, which shows that using logarithmically many successive transformer layers allows each node to propagate information to every node.

**Corollary 4.4.** *If a sparse attention mechanism on $n$ nodes is a $d$-regular $\epsilon$-expander graph, then stacking $O_{d,\epsilon}(\log n)$ transformer layers models all pairwise node interactions.*

### 4.2. Universal Approximability of EXPHORMER Models

The expander graph attention and global attention components of EXPHORMER ensure that a sublinear number of graph transformer layers is sufficient to allow each node to interact (directly or indirectly) with every other node, alleviating potential underreaching issues (even in the absence of global nodes, $O(\log n)$ layers are sufficient, by Corollary 4.4). Nevertheless, a natural question is whether a sparse transformer model based on EXPHORMER allows universal approximation of functions.

Dense transformer models are known to be universal approximators, i.e., with the use of positional encodings, they can approximate any continuous sequence-to-sequence function on a compact domain arbitrarily closely (Yun et al., 2020a). In the realm of graph learning, this provides a compelling motivation for considering graph transformers over standard MPNNs, which are known to be limited in expressive power by the Weisfeiler-Lehman (WL) hierarchy.

Universal approximation properties for dense transformers do not automatically hold for sparse transformers, but we show every continuous function $f : [0, 1]^{d\times|V|} \to \mathbb{R}^{d\times|V|}$ can be approximated to any desired accuracy by an EXPHORMER network using either global attention or a suitable version of expander attention. Details are in Appendix E.

## 5. Experiments

In this section, we evaluate the empirical performance of graph transformer models based on EXPHORMER on a wide variety of graph datasets with graph prediction and node prediction tasks (Dwivedi et al., 2020; Hu et al., 2020; Freitas et al., 2021; Shchur et al., 2018; Namata et al., 2012). In particular, we present experiments on fifteen benchmark datasets, including image-based graph datasets (CIFAR10, MNIST, PascalVOC-SP, COCO-SP), synthetic SBM datasets (PATTERN, CLUSTER), code graph datasets (MalNet-Tiny), and molecular datasets (Peptides-Func, Peptides-Struct, PCQM-Contact). Moreover, we demonstrate the ability of EXPHORMER to allow graph transformers to scale to larger graphs (with more than 5,000 nodes) by including results on five transductive graph datasets consisting of citation networks (CS, Physics, ogbn-arxiv) and

[2]For matrices $A$ and $B$, we say that $A \preceq B$ if $B - A$ is a positive semi-definite matrix.

*Table 2.* Comparison of attention mechanisms in GPS. EXPHORMER outperforms other sparse transformer architectures (BigBird and Performer) while also beating the full transformer GPS models on three of four datasets. Best results are colored in **first**, **second**, **third**.

| Model/Dataset | **Cifar10** Accuracy ↑ | **MalNet-Tiny** Accuracy ↑ | **PascalVOC-SP** F1 score ↑ | **Peptides-Func** AP ↑ |
|---|---|---|---|---|
| GPS (MPNN-only) | $69.948 \pm 0.499$ | $92.23 \pm 0.65$ | $0.3016 \pm 0.0031$ | $0.6159 \pm 0.0048$ |
| GPS-BigBird | $70.480 \pm 0.106$ | $92.34 \pm 0.34$ | $0.2762 \pm 0.0069$ | $0.5854 \pm 0.0079$ |
| GPS-Performer | **$70.670 \pm 0.338$** | **$92.64 \pm 0.78$** | **$0.3724 \pm 0.0131$** | **$0.6475 \pm 0.0056$** |
| GPS-Transformer | **$72.305 \pm 0.344$** | **$93.50 \pm 0.41$** | **$0.3736 \pm 0.0158$** | **$0.6535 \pm 0.0041$** |
| EXPHORMER | **$74.69 \pm 0.125$** | **$94.02 \pm 0.21$** | **$0.3975 \pm 0.0037$** | **$0.6527 \pm 0.0043$** |

co-purchasing networks (Computer, Photo).

For the experimental setup, we combine EXPHORMER together with MPNNs in the GraphGPS framework (Rampásek et al., 2022), which constructs graph transformer models by composing attention mechanisms with message-passing schemes, together with an appropriate choice of positional and structural encodings.

In addition to comparisons with baselines on the aforementioned datasets, we also run ablation studies on the various attention components of EXPHORMER (from Section 3.2).

In summary, our experiments show that: (a) EXPHORMER achieves SOTA performance on a variety of datasets, (b) EXPHORMER consistently outperforms other sparse attention mechanisms while often surpassing dense transformers using fewer parameters, and (c) EXPHORMER successfully allows GraphGPS to overcome memory bottlenecks and scale to larger graphs (on $> 10,000$ nodes) while still providing competitive performance.

### 5.1. Comparison to Sparse Attention Mechanisms

We first discuss a comparison of our EXPHORMER-based models with other *sparse* transformer architectures, which are a natural first point of comparison. In particular, we perform a series of experiments that compare EXPHORMER-based architectures with sparse transformer baselines in the GraphGPS framework. More specifically, for each dataset, we compare our best EXPHORMER model with GraphGPS models that use BigBird and Performer for the underlying attention mechanism.

The results are shown in Table 2. Note that on each of the four highlighted datasets, an EXPHORMER model outperforms the sparse attention models GPS-BigBird and GPS-Performer. Furthermore, it even outperforms the full (dense) transformer model (GPS-Transformer) on three of the datasets while performing competitively on the fourth. Since GraphGPS models make use of both attention (transformer) and message passing components, we additionally point out that the EXPHORMER-based sparse attention component is, indeed, providing lift over a standard MPNN baseline that does not make use of attention at all.

### 5.2. Benchmarking GNNs Datasets

We have showed in Section 5.1 that EXPHORMER consistently outperforms relevant sparse attention baselines. The natural next question is how EXPHORMER performs relative to a wider range of baselines, including not just graph transformer models but also other architectures such as MPNNs.

Table 1 shows results on five datasets, including four from the Benchmarking GNNs collection (Dwivedi et al., 2020) as well as the code graph dataset MalNet-Tiny (Freitas et al., 2021). We note that an EXPHORMER-based graph transformer with message-passing in the GraphGPS framework yields *state-of-the-art (SOTA) performance on three of the datasets* and is competitive with the best single model accuracies on the remaining datasets.

Observe that on all five datasets, our EXPHORMER models provide better accuracy than GraphGPS models based on *full (dense) transformers*. Additionally, the EXPHORMER models outperform the full transformer-based SAN model as well as a variety of MPNN baselines.

*Remark* 5.1. Our EXPHORMER models often outperform full transformer models with a much smaller number of parameters. For instance, on PATTERN, our results reported in Table 1 are obtained from an EXPHORMER model with just 90,000 parameters, compared to 340,000 parameters in the comparable full transformer GraphGPS model! Similarly, on CLUSTER, our EXPHORMER model uses just 280,000 parameters, as compared to 500,000 parameters for the full transformer GraphGPS model. This results in simpler models with time and memory advantages. For further details on hyperparameters, see Table 7 in the appendix.

*Remark* 5.2. EXPHORMER models often enable larger batch sizes during training. On MalNet-Tiny, which contains some graphs with around 5,000 nodes, the full transformer GraphGPS model runs out of memory on a 40GB NVIDIA A100 GPU when using a batch size of just 16, while our sparse EXPHORMER models handle a batch size of 256.

### 5.3. Long-Range Graph Benchmark

We have additionally conducted a set of experiments on the Long-Range Graph Benchmark (LRGB, Dwivedi et al.,

*Table 3.* Comparison of EXPHORMER with baselines from the Long-Range Graph Benchmarks (LRGB, Dwivedi et al., 2022). Best results are colored in **first**, **second**, **third**.

| Model | PascalVOC-SP F1 score ↑ | COCO-SP F1 score ↑ | Peptides-Func AP ↑ | Peptides-Struct MAE ↓ | PCQM-Contact 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** |
| GraphGPS | **0.3748 ± 0.0109** | **0.3412 ± 0.0044** | **0.6535 ± 0.0041** | **0.2500 ± 0.0005** | 0.3337 ± 0.0006 |
| Exphormer (ours) | **0.3975 ± 0.0037** | **0.3455 ± 0.0009** | **0.6527 ± 0.0043** | **0.2481 ± 0.0007** | **0.3637 ± 0.0020** |

*Table 4.* Accuracy of models with different attention mechanisms on transductive graph datasets (numbers in top rows, other than arXiv, are from Chen et al., 2022b). Chen et al. did not report NAGphormer results on this dataset.

| Model | ogbn-arxiv | Computer | Photo | CS | Physics |
|---|---|---|---|---|---|
| SAN | OOM | 89.83 ± 0.16 | 94.86 ± 0.10 | 94.51 ± 0.15 | OOM |
| GraphGPS | OOM | OOM | 95.06 ± 0.13 | 93.93 ± 0.12 | OOM |
| NAGphormer | NA | 91.22 ± 0.14 | 95.49 ± 0.11 | 95.75 ± 0.09 | 97.34 ± 0.03 |
| EXPHORMER | 72.44 ± 0.28 | 91.59 ± 0.31 | 95.27 ± 0.42 | 95.77 ± 0.15 | 97.16 ± 0.13 |

*Table 5.* Results for the full model versus removing each of the components. For the Peptides-Struct dataset, lower is better; for all the others, higher is better.

| Dataset | No Local Edges | No Expander Edges | No Global Nodes | All Components |
|---|---|---|---|---|
| Cifar10 | 74.62 ± 0.12 | 74.53 ± 0.19 | 74.68 ± 0.19 | **74.69 ± 0.13** |
| Malnet-Tiny | 92.64 ± 0.55 | **94.02 ± 0.21** | 92.48 ± 0.33 | 92.06 ± 0.18 |
| Pattern | 86.59 ± 0.03 | 86.70 ± 0.02 | 86.14 ± 0.08 | **86.74 ± 0.02** |
| PascalVOC-SP | 0.3708 ± 0.0039 | 0.3588 ± 0.0013 | **0.3975 ± 0.0037** | 0.3682 ± 0.0042 |
| Peptides-Struct | 0.2631 ± 0.0007 | **0.2481 ± 0.0007** | 0.2655 ± 0.0003 | 0.2643 ± 0.0008 |
| Computer | 90.34 ± 0.45 | 91.48 ± 0.41 | **91.59 ± 0.31** | 91.43 ± 0.53 |

2022), which consists of five challenging datasets that test a model's ability to learn *long-range dependencies* in input graphs. The results are presented in Table 3, which shows that our EXPHORMER-*based sparse attention models are able to outperform GraphGPS on three of the five datasets, achieving SOTA performance*. Furthermore, on the remaining two datasets in the LRGB suite, EXPHORMER is competitive with the best single model results, obtained by full attention GraphGPS models.

### 5.4. Scaling to Larger Graphs

One of the shortcomings of graph transformer architectures has been their poor scalability to larger graphs on thousands of nodes. Dense attention mechanisms (e.g., SAN and Graphormer) have quadratic memory complexity and time complexity, which restrict their applicability to datasets on graphs with relatively few nodes, such as molecular graphs.

GraphGPS has made use of sparse attention mechanisms, but even so, the architecture handles graphs of up to about 5,000 nodes (e.g., MalNet-Tiny, Freitas et al., 2021), often with very small batch size (see Remark 5.2).

A natural question is whether the sparse attention mechanism of EXPHORMER models allows us to extend graph transformer architectures to significantly larger graphs while providing competitive performance. Indeed, we show this is the case by training EXPHORMER models on larger datasets, including Amazon Computer, Amazon Photo, Coauthor CS, Coauthor Physics (Shchur et al., 2018), which has up to 35K nodes and 250K edges. We also show the use of EXPHORMER on ogbn-arxiv, which has 169K nodes and 1.1M edges. The results are shown in Table 4. Standard GraphGPS models suffer from Out Of Memory (OOM) issues on a number of these datasets, demonstrating the utility of EXPHORMER. To provide meaningful comparisons to other transformer architectures, we thus include NAGphormer (Chen et al., 2022b), a scalable sampling-based graph transformer architecture, as a baseline. EXPHORMER performs competitively and, in fact, achieves SOTA accuracy on Computer: the best non-transformer method has an

accuracy of 90.74 (Luo et al., 2022).

We would also like to highlight that EXPHORMER can be used for even larger graphs, with hundreds of thousands of nodes. In particular, we provide results on ogbn-arxiv (Hu et al., 2020), a challenging transductive dataset consisting of a single large graph of the citation network of over 160K arXiv papers, containing over a million edges. Specifically, we achieve a test accuracy of **0.7244** using the EXPHORMER architectures. At the time of writing, a relevant leaderboard (ogb) shows 0.7966 as the highest reported test accuracy (Zhao et al., 2022a). Table 15 compares EXPHORMER to other MPNNs and sparse transformers. A dense full transformer does not even fit in memory on a 40GB NVIDIA A100 GPU (even for a tiny network of only 2 layers and 32 hidden dimensions). We then fixed the network size to 3 hidden layers and 96 hidden dimensions to be able to fit the sparse models in memory. Notice that BigBird and Performer have significantly longer epoch times and worse performance than with just a MPNN. EXPHORMER with virtual nodes only (without expander edges) improves on the MPNN and provides an accuracy of 0.7222. EXPHORMER with expander edges further improves the accuracy to 0.7244.

### 5.5. Ablation Studies

Here we analyze the effect of each of the components of the model. Our EXPHORMER model has three main components: local neighborhood, expander edges, and virtual nodes. In Table 5, we analyze which parts of the model have been useful, and which parts can be removed without reducing the performance. In all of these experiments, the MPNN part is fixed, so it gives a baseline number for the results, and the transformer part boosts over the MPNN part. Through these experiments we can see that local neighborhood have been always a good option to add to the *Exphormer*, but between virtual nodes and expander graphs, sometimes one of them causes reduction in the performance. This issue is discussed further in Appendix D.

## 6. Conclusion

EXPHORMER is a new class of sparse graph transformer architectures. We introduced two types of sparse networks with virtual nodes and using expander graphs. We have shown that the mathematical properties of our architecture make EXPHORMER a suitable choice for graph learning. Our sparse architecture uses fewer training parameters, is faster to train and has memory complexity linear in the size of the graph. These properties help us scale to larger graphs which were typically elusive to other Transformer-based methods. EXPHORMER outperforms other sparse transformers and performs comparably or better than full transformers. We have shown the applicability of EXPHORMER on a wide variety of graph learning datasets. Combining EXPHORMER with MPNN in the GraphGPS framework allows us to obtain state-of-the-art empirical results on a number of datasets.

## Acknowledgements

This work was supported in part by the Natural Sciences and Engineering Resource Council of Canada, the Canada CIFAR AI Chairs program, the BC DRI Group, Compute Ontario, Calcul Québec, and the Digital Resource Alliance of Canada.

## References

OGB leaderboard for arxiv dataset. https://ogb.stanford.edu/docs/leader_nodeprop/#ogbn-arxiv. Accessed: 2023-01-26.

Alon, N. Explicit expanders of every degree and size. *Combinatorics*, 41(4):447–463, 2021.

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021.

Beaini, D., Passaro, S., Létourneau, V., Hamilton, W., Corso, G., and Liò, P. Directional graph networks. In *International Conference on Machine Learning (ICML)*, pp. 748–758. PMLR, 2021.

Bo, D., Shi, C., Wang, L., and Liao, R. Specformer: Spectral graph neural networks meet transformers. *arXiv preprint arXiv:2303.01028*, 2023.

Bodnar, C., Frasca, F., Otter, N., Wang, Y. G., Liò, P., Montufar, G. F., and Bronstein, M. Weisfeiler and lehman go cellular: Cw networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Bouritsas, G., Frasca, F., Zafeiriou, S., and Bronstein, M. M. Improving graph neural network expressivity via subgraph isomorphism counting. *arXiv preprint arXiv:2006.09252*, 2020.

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

Chen, D., O'Bray, L., and Borgwardt, K. Structure-aware transformer for graph representation learning. *arXiv:2202.03036*, 2022a.

Chen, J., Gao, K., Li, G., and He, K. Nagphormer: Neighborhood aggregation graph transformer for node classification in large graphs. *CoRR*, abs/2206.04910, 2022b.

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlós, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D. B., Colwell, L. J., and Weller, A. Rethinking attention with performers. In *ICLR*, 2021.

Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Veličković, P. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020.

Deac, A., Lackenby, M., and Veličković, P. Expander graph propagation. In *Learning on Graphs 2022*, 2022. URL https://arxiv.org/abs/2210.02997.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852, 2016.

Dwivedi, V. P. and Bresson, X. A generalization of transformer networks to graphs. *CoRR*, abs/2012.09699, 2020.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *CoRR*, abs/2003.00982, 2020. URL https://arxiv.org/abs/2003.00982.

Dwivedi, V. P., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. Graph neural networks with learnable structural and positional representations. *arXiv preprint arXiv:2110.07875*, 2021.

Dwivedi, V. P., Rampásek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A. T., and Beaini, D. Long range graph benchmark. *CoRR*, abs/2206.08164, 2022.

Fey, M., Yuen, J.-G., and Weichert, F. Hierarchical inter-message passing for learning on molecular graphs. *arXiv preprint arXiv:2006.12179*, 2020.

Freitas, S., Dong, Y., Neil, J., and Chau, D. H. A large-scale database for graph representation learning. In Vanschoren, J. and Yeung, S. (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021.

Friedman, J. A proof of Alon's second eigenvalue conjecture. In Larmore, L. L. and Goemans, M. X. (eds.), *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pp. 720–724. ACM, 2003.

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035, 2017.

Hoory, S., Linial, N., and Wigderson, A. Expander graphs and their applications. *Bull. Amer. Math. Soc.*, 43(04): 439–562, August 2006. ISSN 0273-0979.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Hu, W., Fey, M., Ren, H., Nakata, M., Dong, Y., and Leskovec, J. OGB-LSC: A large-scale challenge for machine learning on graphs. *CoRR*, abs/2103.09430, 2021.

Hussain, M. S., Zaki, M. J., and Subramanian, D. Edge-augmented graph transformers: Global self-attention is enough for graphs. *arXiv preprint arXiv:2108.03348*, 2021.

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

Kreuzer, D., Beaini, D., Hamilton, W. L., Létourneau, V., and Tossou, P. Rethinking graph transformers with spectral attention. *arXiv preprint arXiv:2106.03893*, 2021.

Lubotzky, A., Phillips, R., and Sarnak, P. Ramanujan graphs. *Comb.*, 8(3):261–277, 1988.

Luo, Y., Luo, G., Yan, K., and Chen, A. Inferring from references with differences for semi-supervised node classification on graphs. *Mathematics*, 10(8), 2022. ISSN 2227-7390. doi: 10.3390/math10081262. URL https://www.mdpi.com/2227-7390/10/8/1262.

Margulis, G. A. Explicit group-theoretic constructions of combinatorial schemes and their applications in the construction of expanders and concentrators. *Problemy Peredachi Informatsii*, 24(1):51–60, 1988. ISSN 0555-2923.

Mialon, G., Chen, D., Selosse, M., and Mairal, J. Graphit: Encoding graph structure in transformers. *CoRR*, abs/2106.05667, 2021.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019.

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

Namata, G. M., London, B., Getoor, L., and Huang, B. Query-driven active surveying for collective classification. In *Workshop on Mining and Learning with Graphs*, 2012. URL http://linqs.cs.umd.edu/basilic/web/Publications/2012/namata:mlg12-wkshp/namata-mlg12.pdf.

Oono, K. and Suzuki, T. Graph neural networks exponentially lose expressive power for node classification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., and Tang, J. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pp. 459–467, 2018.

Rampásek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. Recipe for a general, powerful, scalable graph transformer. *CoRR*, abs/2205.12454, 2022.

Sato, R., Yamada, M., and Kashima, H. Random features strengthen graph neural networks. In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, pp. 333–341. SIAM, 2021.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. Pitfalls of graph neural network evaluation. *CoRR*, abs/1811.05868, 2018.

Spielman, D. A. Spectral and algebraic graph theory, 2019. URL http://cs-www.cs.yale.edu/homes/spielman/sagt. Version dated December 19, 2019.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. Efficient transformers: A survey. *CoRR*, abs/2009.06732, 2020. URL https://arxiv.org/abs/2009.06732.

Toenshoff, J., Ritzert, M., Wolf, H., and Grohe, M. Graph learning with 1d convolutions on random walks. *arXiv:2102.08786*, 2021.

Topping, J., Giovanni, F. D., Chamberlain, B. P., Dong, X., and Bronstein, M. M. Understanding over-squashing and bottlenecks on graphs via curvature. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

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

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

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., and Liu, T.-Y. Do transformers really perform bad for graph representation? *ArXiv*, abs/2106.05234, 2021.

Yun, C., Bhojanapalli, S., Rawat, A. S., Reddi, S. J., and Kumar, S. Are transformers universal approximators of sequence-to-sequence functions? In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020a.

Yun, C., Chang, Y., Bhojanapalli, S., Rawat, A. S., Reddi, S. J., and Kumar, S. O(n) connections are expressive enough: Universal approximability of sparse transformers. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020b.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontañón, S., Pham, P., Ravula, A., Wang, Q., Yang, L., and Ahmed, A. Big bird: Transformers for longer sequences. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *NeurIPS*, 2020.

Zhang, J., Zhang, H., Xia, C., and Sun, L. Graph-bert: Only attention is needed for learning graph representations. *CoRR*, abs/2001.05140, 2020.

Zhao, J., Li, C., Wen, Q., Wang, Y., Liu, Y., Sun, H., Xie, X., and Ye, Y. Gophormer: Ego-graph transformer for node classification. *CoRR*, abs/2110.13094, 2021.

Zhao, J., Qu, M., Li, C., Yan, H., Liu, Q., Li, R., Xie, X., and Tang, J. Learning on large-scale text-attributed graphs via variational inference, 2022a. URL https://arxiv.org/abs/2210.14709.

Zhao, L., Jin, W., Akoglu, L., and Shah, N. From stars to subgraphs: Uplifting any GNN with local structure awareness. In *International Conference on Learning Representations*, 2022b.

# A. Dataset Descriptions

Below, we provide descriptions of the datasets on which we conduct experiments.

**CIFAR10 and MNIST** (Dwivedi et al., 2020) CIFAR10 and MNIST are the graph equivalents of the image classification datasets of the same name. A graph is created by constructing the 8-nearest neighbor graph of the SLIC superpixels of the image. These are both 10-class graph classification problems.

**PascalVOC-SP and COCO-SP** (Dwivedi et al., 2021) These are similar graph versions of image datasets, but they are larger images and the task is to perform node classification, i.e. semantic segmentation of superpixels. These graphs are larger, and the task more complex, than CIFAR10 and MNIST.

**CLUSTER and PATTERN** (Dwivedi et al., 2020) PATTERN and CLUSTER are node classification problems. Both are synthetic datasets that are sampled from a Stochastic Block Model (SBM), is a popular way to model communities. In PATTERN, the prediction task is to identify if a node belongs to one of the 100 possible predetermined subgraph patterns. In CLUSTER, the goal is to classify nodes into six different clusters with the same distribution.

**MalNet-Tiny** (Freitas et al., 2021) Malnet-Tiny is a smaller dataset generated from a larger dataset for identifying malware based on function call graphs from Android APKs. The tiny dataset contains 5000 graphs, each with up to 5000 nodes. The task is to predict the graph as being benign or from one of four types of malware.

**ogbn-arxiv** (Hu et al., 2021) The ogbn-arxiv dataset consists of one large directed graph of 169343 nodes and 1,166,243 edges representing a citation network between all computer science papers on arXiv that were indexed by the Microsoft academic graph. Nodes in the graph represent papers, while a directed edge indicates that a paper cites another. Each node has an 128-dimensional feature vector derived from embeddings of words in the title and abstract of the underlying paper. The prediction task is a 40-class node classification problem — to identify the primary category of each arXiv paper, as listed by the authors. Moreover, the nodes of the citation graph are split into 90K training nodes, 30K validation notes, and 48K test nodes.

**Coauthor datasets** Coauthor CS and Physics are co-authorship graphs from Microsoft Academic Graph. The nodes represent the authors and two authors who share a paper are connected by an edge. The node features are from the keywords in the papers. The class represent the active area of study for the author.

**Amazon datasets** Amazon Computers and Amazon photo are Amazon co-purchase graphs. Nodes represents products purchased an edges indicate pairs of products purchased together. Node features are bag-of-words encoded reiews of the products. Class labels are the product category.

**Peptides-Func, Peptides-Struct, and PCQM-Contact** (Dwivedi et al., 2021) These datasets are molecular graphs introduced as a part of the Long Range Graph Benchmark (LRGB). Graphs in these datasets have relatively large diameters: the average diameter for PCQM-Contact is 9.86±1.79, and for the Peptides datasets 56.99±28.72. Average shortest path lengths are 4.63±0.63 and 20.89±9.79 accordingly. On PCQM-Contact the task is edge-level, and we need to rank the edges. Peptides-Func is a multi-label graph classification task, with 10 labels. Peptides-Struct is graph-level regression of 11 structural properties of the molecules.

Table 6 shows a summary of the statistics of the aforementioned datasets.

# B. More Experimental Results

## B.1. Hyperparameters

Our hyperparameter choices, including the optimizer, positional encodings, and structural encodings, were guided by the instructions in GraphGPS (Rampásek et al., 2022). There were some cases, however, when more layers with smaller dimensions gave better results in EXPHORMER. This may be due to the fact that each node gets fewer inputs for each layer, but EXPHORME requires more layers in order to propagate well. Additionally, we observed that Equivstable Laplacian Positional Encoding (ESLapPE) performed better than normal Laplacian Positional Encoding (LapPE) in some cases, in

*Table 6.* Dataset statistics

| Dataset | Graphs | Avg. nodes | Avg. edges | Prediction Level | No. Classes | Metric |
|---|---|---|---|---|---|---|
| MNIST | 70,000 | 70.6 | 564.5 | graph | 10 | Accuracy |
| CIFAR10 | 60,000 | 117.6 | 941.1 | graph | 10 | Accuracy |
| PATTERN | 14,000 | 118.9 | 3,039.3 | inductive node | 2 | Accuracy |
| CLUSTER | 12,000 | 117.2 | 2,150.9 | inductive node | 6 | Accuracy |
| MalNet-Tiny | 5,000 | 1,410.3 | 2,859.9 | graph | 5 | Accuracy |
| PascalVOC-SP | 11,355 | 479.4 | 2,710.5 | inductive node | 21 | F1 |
| COCO-SP | 123,286 | 476.9 | 2,693.7 | inductive node | 81 | F1 |
| PCQM-Contact | 529,434 | 30.1 | 61.0 | inductive link | (link ranking) | MRR |
| Peptides-func | 15,535 | 150.9 | 307.3 | graph | 10 | Average Precision |
| Peptides-struct | 15,535 | 150.9 | 307.3 | graph | 11 (regression) | Mean Absolute Error |
| ogbn-arxiv | 1 | 169,343 | 1,166,243 | node | 40 | Accuracy |
| Amazon Computer | 1 | 13381 | 245778 | node | 10 | Accuracy |
| Amazon Photo | 2 | 7487 | 119043 | node | 8 | Accuracy |
| Coauthor CS | 1 | 18333 | 81894 | node | 15 | Accuracy |
| Coauthor Physics | 1 | 34493 | 247962 | node | 5 | Accuracy |

cases we replaced you can see ESLapPE version of GraphGPS results in Tables 10, 11, 13 and 14. Except for the large scale graphs, which we use GCN, we have used CustomGatedGCN beside the Exphormer in all of the experiments.

Through our model, some extra hyperparameters are introduced — the degree of the graph expander and the number of virtual nodes. For these hyperparameters, we used linear search and found that expander degree 6-22 was the most effective. Depending on the graph size, we used 1-6 virtual nodes. As the number of hyperparameters is large, grid search was not feasible on all the parameters. As a result, for other hyperparameters, we did a linear search for each parameter to find out which parameters work better.

To make fair comparisons, we used a similar parameter-budget to GraphGPS. For PATTERN and CLUSTER, we used a parameter-budget of 500K, and for CIFAR and MNIST, we used a parameter-budget of around 100K. See details in Tables 7 to 9.

*Table 7.* Hyperparameters used for EXPHORMER for datasets: CIFAR10, MNIST, MalNet-Tiny, PATTERN, CLUSTER.

| Hyperparameter | CIFAR10 | MNIST | MalNet-Tiny | PATTERN | CLUSTER |
|---|---|---|---|---|---|
| Num Layers | 5 | 5 | 5 | 4 | 20 |
| Hidden Dim | 40 | 40 | 64 | 40 | 32 |
| Num Heads | 4 | 4 | 4 | 4 | 8 |
| Dropout | 0.1 | 0.1 | 0 | 0.0 | 0.0 |
| PE | ESLapPE | ESLapPE | None | ESLapPE | ESLapPE |
| Batch Size | 16 | 16 | 16 | 32 | 16 |
| Learning Rate | 0.001 | 0.001 | 0.0005 | 0.0002 | 0.0002 |
| Num Epochs | 150 | 150 | 150 | 120 | 150 |
| Expander Degree | 10 | 10 | - | 14 | 6 |
| Num Virtual Nodes | 1 | 1 | 4 | 4 | 3 |
| Num parameters | 111,095 | 111,015 | 286,277 | 91,045 | 282,970 |

### B.2. Full Comparison of Attention Mechanisms

*Remark* B.1. EXPHORMER has some conceptual similarities with BigBird, as mentioned previously. For instance, we also make use of virtual global attention nodes, corresponding to BIGBIRD-ETC.

However, our approach departs from that of BigBird in some important ways. While BigBird uses $w$-width "window attention" to capture locality of reference, we use local neighborhood attention to capture locality and graph topology. In particular, the interaction graph due to window attention in BigBird can be viewed as a Cayley graph on $\mathbb{Z}_n$, which is sequence-centric, while EXPHORMER is graph-centric and, therefore, uses the structure of the input graph itself to capture locality. BigBird, as implemented for graphs by Rampásek et al. (2022), instead simply orders the graph nodes in an arbitrary

*Table 8.* Hyperparameters used for EXPHORMER for LRGB datasets.

| Hyperparameter | PascalVOC-SP | COCO-SP | Peptides-Func | Peptides-Struct | PCQM-Contact |
|---|---|---|---|---|---|
| Num Layers | 4 | 7 | 8 | 4 | 7 |
| Hidden Dim | 96 | 72 | 64 | 88 | 64 |
| Num Heads | 8 | 4 | 4 | 4 | 4 |
| Dropout | 0.15 | 0 | 0.12 | 0.12 | 0 |
| PE | LapPE | LapPE | LapPE | LapPE | LapPE |
| Batch Size | 32 | 32 | 128 | 128 | 128 |
| Learning Rate | 0.0005 | 0.0005 | 0.0003 | 0.0003 | 0.0003 |
| Num Epochs | 300 | 300 | 200 | 200 | 200 |
| Expander Degree | 14 | 22 | - | - | - |
| Num Virtual Nodes | 0 | 0 | 1 | 6 | 6 |
| Num parameters | 509,301 | 498,993 | 445,866 | 426,427 | 395,936 |

*Table 9.* Hyperparameters used for EXPHORMER for large transductive graph datasets.

| Hyperparameter | OGBN-Arxiv | Computer | Photo | CS | Physics |
|---|---|---|---|---|---|
| Num Layers | 3 | 4 | 4 | 4 | 4 |
| Hidden Dim | 96 | 80 | 64 | 72 | 72 |
| Num Heads | 2 | 2 | 2 | 2 | 2 |
| Dropout | 0.3 | 0.4 | 0.4 | 0.4 | 0.4 |
| PE | - | - | - | - | - |
| Learning Rate | 0.01 | 0.001 | 0.001 | 0.001 | 0.001 |
| Num Epochs | 600 | 150 | 100 | 70 | 70 |
| Expander Degree | 6 | 6 | 6 | 6 | 6 |
| Num Virtual Nodes | 0 | 0 | 0 | 0 | 0 |
| Num parameters | 268,264 | 302,570 | 202,632 | 686,103 | 801,293 |

sequence and uses windows within that sequence.

Both BigBird and EXPHORMER also make use of a random attention model. While BigBird uses an Erdős-Rényi graph on $|V|$ nodes, our approach is to use a $d$-regular expander for fixed constant $d$. The astute reader may recall that a Erdős-Rényi graph $G(n, p)$ has spectral expansion properties for large enough $p$. However, it is known that $p = \frac{\log n}{n}$ is the connectivity threshold, i.e., for $p < (1 - \epsilon)\frac{\log n}{n}$, $G(n, p)$ is almost surely a disconnected graph. Therefore, in order to obtain even a connected graph in the Erdős-Rényi model – let alone one with expansion properties – one would need $p = \Omega\left(\frac{\log n}{n}\right)$, giving superlinear complexity for the number of edges. BigBird uses $p = \Theta(1/n)$, keeping a linear number of edges but losing expansion properties. Our expander graphs, by contrast, allow both a linear number of edges and guaranteed spectral expansion properties.

We will see in the practical experiments of Section 5 that EXPHORMER-based models often substantially outperform BigBird-based equivalents, with fewer parameters.

In Section 5.1, we presented two approaches for the comparison of models trained using different attention mechanisms — fixing the hyperparameters and fixing a budget on the total number of trainable parameters. The results showed the advantage of EXPHORMER over other attention mechanisms for CIFAR10 (Table 10) and PATTERN (Table 13). Here, we present similar results for the remaining datasets — MNIST in Table 11; MalNet-Tiny in Table 12; PATTERN in Table 13; and CLUSTER in Table 14.

## C. Details of Expander Graph Construction

A major component of EXPHORMER is the use of the edges of an expander graph. In this section, we provide details of the specific expander graphs we use as well and quantify their spectral expansion properties.

*Table 10.* Results with varying attention and MPNNs on CIFAR10. EXPHORMER with MPNN provides the highest accuracy. Also, pure transformer models based on EXPHORMER (without the use of MPNNs) are comparable.

| Model | #Layers | Hidden layers | #Positional encoding | Expander degree | #Parameters | Time Epch/Total | Accuracy |
|---|---|---|---|---|---|---|---|
| GPS-MPNN: GatedGCN | 3 | 52 | LapPE | - | 79,654 | 43s/1.18h | $69.95 \pm 0.499$ |
| GPS: Transformer | 3 | 52 | LapPE | - | 70,762 | 40s/1.11h | $68.86 \pm 1.138$ |
| GPS: Transformer + MPNN | 5 | 40 | ESLapPE | - | 111,735 | 104s/2.89h | $73.53 \pm 0.238$ |
| GPS: Transformer + MPNN | 3 | 52 | LapPE | - | 112,726 | 62s/1.72h | $72.31 \pm 0.344$ |
| GPS: Performer + MPNN | 5 | 40 | ESLapPE | - | 283,935 | 132s/3.65h | $70.18 \pm 0.095$ |
| GPS: Performer + MPNN | 3 | 52 | LapPE | - | 239,554 | 77s/2.14h | $70.67 \pm 0.338$ |
| GPS: BigBird + MPNN | 5 | 40 | ESLapPE | - | 128,335 | 243s/6.75h | $70.51 \pm 0.256$ |
| GPS: BigBird + MPNN | 3 | 52 | LapPE | - | 129,418 | 145s/4h | $70.48 \pm 0.106$ |
| EXPHORMER w/o MPNN | 5 | 44 | ESLapPE | 10 | 84,134 | 64s/1.78h | $72.33 \pm 0.155$ |
| EXPHORMER w/o MPNN | 7 | 44 | ESLapPE | 10 | 119,022 | 80s/2.23h | $72.88 \pm 0.166$ |
| EXPHORMER | 5 | 40 | ESLapPE | 10 | 111,095 | 115s/3.21h | $74.75 \pm 0.194$ |
| EXPHORMER | 5 | 44 | ESLapPE | 10 | 133,819 | 114s/3.19h | $\mathbf{75.03 \pm 0.186}$ |

*Table 11.* Ablation studies results for MNIST

| Model | #Layers | Hidden layers | #Positional encoding | Expander degree | #Parameters | Time Epoch/Total | Accuracy |
|---|---|---|---|---|---|---|---|
| GPS: Transformer + MPNN | 5 | 40 | ESLapPE | - | 111,655 | 131s/5.45h | $98.336 \pm 0.0189$ |
| GPS: Transformer + MPNN | 3 | 52 | LapPE | - | 115,394 | 76s/2.13h | $98.051 \pm 0.126$ |
| GPS: Performer + MPNN | 5 | 40 | ESLapPE | - | 283,855 | 156s/6.52h | $98.34 \pm 0.0349$ |
| GPS: BigBird + MPNN | 5 | 40 | ESLapPE | - | 128,255 | 267s/11.11h | $98.176 \pm 0.0146$ |
| EXPHORMER w/o MPNN | 5 | 44 | ESLapPE | 10 | 92,146 | 75s/3.14h | $98.08 \pm 0.051$ |
| EXPHORMER w/o MPNN | 7 | 44 | ESLapPE | 10 | 127,698 | 93s/3.87h | $98.238 \pm 0.0387$ |
| EXPHORMER | 5 | 40 | ESLapPE | 10 | 111,015 | 132s/5.49h | $98.414 \pm 0.035$ |
| EXPHORMER | 5 | 44 | ESLapPE | 10 | 133,731 | 137s/5.72h | $\mathbf{98.424 \pm 0.018}$ |

### C.1. Ramanujan Graphs

A natural question is how strong the spectral expansion properties of a $d$-regular graph can be, i.e., for how large an $\epsilon > 0$ does a $d$-regular $\epsilon$-expander exist. The following theorem gives a bound on how large the spectral gap can be.

**Theorem C.1** (Alon-Boppana)**.** *Let $d > 0$. The eigenvalues of the adjacency matrix of a $d$-regular graph on $n$ nodes satisfy*

$$\max\{|\lambda_2|, |\lambda_n|\} \geq 2\sqrt{d-1} - o_n(1).$$

*In other words, a $d$-regular $\epsilon$-expander graph can exist only for $\epsilon \geq \frac{2\sqrt{d-1}}{d} - o_n(1)$.*

As it turns out, there exist $\epsilon$-expander graphs with $\epsilon$ achieving this bound. In fact, a $d$-regular $\epsilon$-expander graph satisfying $\epsilon \leq \frac{2\sqrt{d-1}}{d}$ is known as a *Ramanujan graph*. Ramanujan graphs are essentially the best possible spectral expanders, and several constructions have been considered over the years (Lubotzky et al., 1988; Margulis, 1988).

### C.2. Random Regular Graphs

While there exist deterministic constructions of Ramanujan graphs, they are often algebraic/number theoretic in nature and therefore exist only for specific choices of $d$ (e.g., the constructions of Lubotzky et al. (1988) as well as independently of Margulis (1988), for which one requires $d \equiv 2 \pmod 4$ and $d-1$ to be a prime). Recently, the work of Alon (2021) showed a construction of strongly explicit near-Ramanujan graphs of every degree, but it should be noted that the construction needs the number of nodes to be sufficiently large. It is, therefore, often convenient to use a probabilistic construction of an expander.

A natural choice for an expander graph is a *random* $d$-regular graph on $n$ vertices, formed by taking $d/2$ independent uniform permutations on $\{1, 2, \ldots, n\}$. Friedman (2003) proved a conjecture of Alon, establishing that random regular graphs are *weakly-Ramanujan*.

**Theorem C.2.** *(Friedman, 2003, Theorem 1.1) Fix $\epsilon > 0$ and an even integer $d > 2$. Then, suppose $G = (V, E)$ is a random graph generated by taking $d/2$ independent uniformly random permutations $\pi_1, \pi_2, \ldots, \pi_{d/2}$ on $V = \{1, 2, \ldots, n\}$*

*Table 12.* Ablation studies results for MalNet-Tiny. We want to remark that these results are based on one virtual node. For the best result reported in the main paper, we have used 4 virtual nodes, which led to much better numbers. The model marked with * did not fit in memory with batch size 16, and was trained with batch size 8.

| Model | #Layers | Hidden layers | #Positional encoding | Expander degree | #Parameters | Time Epoch/Total | Accuracy |
|---|---|---|---|---|---|---|---|
| GPS-MPNN: GatedGCN | 5 | 64 | - | - | 199,237 | 6s/0.25h | $92.23 \pm 0.65$ |
| GPS: Performer | 5 | 64 | - | - | 421,957 | 41s/1.73h | $73.90 \pm 0.58$ |
| GPS: Transformer + MPNN* | 5 | 64 | - | - | 282,437 | 94s/3.94h | $\mathbf{93.50 \pm 0.41}$ |
| GPS: Performer + MPNN | 5 | 64 | - | - | 527,237 | 46s/1.90h | $92.64 \pm 0.78$ |
| GPS: BigBird + MPNN | 5 | 64 | - | - | 324,357 | 130s/5.43h | $92.34 \pm 0.34$ |
| EXPHORMER w/o MPNN | 5 | 80 | - | 10 | 283,173 | 25.2s/1.05h | $92.18 \pm 0.292$ |
| EXPHORMER w/o MPNN | 8 | 64 | - | 10 | 296,325 | 35.2s/1.47h | $92.24 \pm 0.291$ |
| EXPHORMER | 5 | 64 | - | 10 | 286,277 | 25.1s/1.05h | $94.02 \pm 0.209$ |

*Table 13.* Ablation studies results for PATTERN

| Model | #Layers | Hidden dimension | #Positional encoding | Expander degree | #Parameters | Time Epoch/Total | Accuracy |
|---|---|---|---|---|---|---|---|
| GPS: Transformer + MPNN | 4 | 40 | ESLapPE | - | 91,165 | 18s/0.59h | $86.114 \pm 0.103$ |
| GPS: Transformer + MPNN | 6 | 64 | LapPE-16 | - | 337,201 | 32s/0.89h | $86.685 \pm 0.059$ |
| GPS: Performer + MPNN | 4 | 40 | ESLapPE | - | 228,925 | 20s/0.68h | $83.342 \pm 0.294$ |
| GPS: BigBird + MPNN | 4 | 40 | ESLapPE | - | 104,445 | 29s/0.97h | $86.005 \pm 0.148$ |
| EXPHORMER w/o MPNN | 4 | 40 | ESLapPE | 14 | 56,801 | 13s/0.44h | $85.622 \pm 0.007$ |
| EXPHORMER w/o MPNN | 4 | 56 | ESLapPE | 14 | 109,985 | | |
| EXPHORMER | 4 | 40 | ESLapPE | 14 | 91,045 | 21s/0.71h | $\mathbf{86.734 \pm 0.008}$ |

*and then choosing the edge set as*

$$E = \left\{(i, \pi_j(i)), (i, \pi_j^{-1}(i)) : 1 \leq j \leq d, 1 \leq i \leq n\right\}.$$

*Then, with probability* $1 - O(n^{-\Omega(\sqrt{d})})$*, G satisfies* $\lambda_j(G) \leq 2\sqrt{d-1} + \epsilon$ *for* $j = 2, \ldots, n$*, where* $d = \lambda_1(G) \geq \lambda_2(G) \geq \cdots \geq \lambda_n(G) \geq -d$ *are the eigenvalues of the adjacency matrix of* $G$*.*

In the experiments reported in Tables 1 to 5, we use graphs generated according to the random process described above, with self loops removed, to instantiate the expander graph component of EXPHORMER.

**A simple variant.** A simple variant of the aforementioned random process is to instead choose a single permutation on $nd/2$ elements formed by taking $d/2$ copies of $1, 2, \ldots, n$. More precisely, given an input graph $(V, E)$, one can generate an expander graph $(V, E')$ as follows:

- Let $s = (\underbrace{1, \ldots, 1}_{d/2}, \underbrace{2, \ldots, 2}_{d/2}, \ldots, \underbrace{n, \ldots, n}_{d/2})$.
- Pick a permutation $\pi$ on $\{1, 2, \ldots, nd/2\}$ uniformly at random from the $(nd/2)!$ possible permutations.
- Let $E'$ be the multiset $\{(s_i, s_{\pi(i)}), (s_{\pi(i)}, s_i) : 1 \leq i \leq nd/2\}$.

*Table 14.* Ablation studies results for CLUSTER

| Model | #Layers | Hidden dimension | #Positional encoding | Expander degree | #Parameters | Time Epoch/Total | Accuracy |
|---|---|---|---|---|---|---|---|
| GPS: Transformer + MPNN | 20 | 32 | ESLapPE | - | 285,210 | 91s/3.79h | $78.053 \pm 0.044$ |
| GPS: Transformer + MPNN | 16 | 48 | LapPE-10 | - | 502,054 | 86s/2.40h | $78.016 \pm 0.180$ |
| GPS: Performer + MPNN | 20 | 32 | ESLapPE | - | 1,512,090 | 120s/5.01h | $\mathbf{78.292 \pm 0.046}$ |
| GPS: BigBird + MPNN | 20 | 32 | ESLapPE | - | 328,090 | 224s/9.32h | $77.468 \pm 0.023$ |
| EXPHORMER w/o MPNN | 20 | 32 | ESLapPE | 6 | 171,590 | 49s/2.03h | $76.922 \pm 0.044$ |
| EXPHORMER | 20 | 32 | ESLapPE | 6 | 282,970 | 91s/3.78h | $78.22 \pm 0.045$ |

*Table 15.* Comparison of attention mechanisms on the ogbn-arxiv dataset by fixing the network size.

| Model | Accuracy | Epoch times | #Parameters |
|---|---|---|---|
| GCN+EXPHORMER with expander edges | $72.44 \pm 0.28$ | 1.72 | 268,552 |
| GCN+EXPHORMER with virtual nodes | $72.22 \pm 0.13$ | 1.76 | 269,704 |
| GCN only | $71.35 \pm 0.31$ | 0.21 | 74,152 |
| GCN+BigBird | $71.16 \pm 0.19$ | 17.85 | 325,480 |
| GCN+Performer | $70.92 \pm 0.04$ | 5.77 | 452,776 |

For the ablation studies in Tables 10 to 14, we use the above variant to generate the expander graph component of EXPHORMER (once again removing self loops). We find that, in practice, the above variant produces near-Ramanujan graphs most of the time.

**Hamiltonian cycle variant.** Another interesting variant of the random graph process analyzed in Theorem C.2 is the one in which one once again takes $d/2$ independent random permutations, except that the permutations must be one of the $(n-1)!$ permutations consisting of a single cycle of length $n$. In other words, one chooses $d/2$ independent Hamiltonian cycles on $V$ and takes all edges from each of the Hamiltonian cycles. The work of Friedman (2003), in fact, analyzes this modified process:

**Theorem C.3.** *(Friedman, 2003, Theorem 1.2) Fix $\epsilon > 0$ and an even integer $d > 2$. Then, suppose $G = (V, E)$ is a random graph generated as follows: Take independent permutations $\pi_1, \pi_2, \ldots, \pi_{d/2}$ on $V = \{1, 2, \ldots, n\}$, where each $\pi_i$ is chosen uniformly at random from the $(n-1)!$ permutations whose cyclic decomposition consists of a single cycle. Then choose the edge set as*

$$E = \left\{(i, \pi_j(i)), (i, \pi_j^{-1}(i)) : 1 \leq j \leq d, 1 \leq i \leq n\right\}.$$

*Then, with probability $1 - O(n^{-\Omega(\sqrt{d})})$, $G$ satisfies $\lambda_j(G) \leq 2\sqrt{d-1} + \epsilon$ for $j = 2, \ldots, n$, where $d = \lambda_1(G) \geq \lambda_2(G) \geq \cdots \geq \lambda_n(G) \geq -d$ are the eigenvalues of the adjacency matrix of $G$.*

The advantage of using this random process for generating an expander graph is that it automatically satisfies the universal approximation property outlined in Appendix E (see Theorem E.3).

## D. On Expander Graphs versus Global Connectors

From ablation studies in Section 5.5, we can see that sometimes only having expander graphs leads to the best results, sometimes virtual nodes only, and sometimes using a combination of both. Although this is a hyperparameter to tune, we can have a general understanding from the datasets on which cases virtual nodes work better and vice versa. Here we remark on some of these patterns.

**Graph diameter**: Using virtual nodes, the diameter of the graph becomes exactly 2 (unless the original graph was already complete, with diameter 1). Expander graphs also help to reduce the diameter of the graph, but only achieve a (probabilistic) diameter guarantee of $O(\log n)$.

**Information Bottleneck**: Virtual nodes serve as a global sink, and so even a single node should be able to keep all the information flowing from the whole graph. In case the graphs are large or many nodes rely on the virtual nodes to pass information, though, virtual nodes are likely to cause information bottleneck, and may even reduce the performance of the model. Expander graphs, to the contrary, introduce many new paths; as they rely on the local information storage of the nodes, they do not have this same problem.

**Interference with the local structure**: Virtual nodes introduce a new node, through which all of their new connections pass. Expander edges, on the other hand, are much easier to confuse with the actual edges of the graph; this can make it more difficult for the model to use the graph structure appropriately. As our model uses shared weights among the types of activations, which substantially helps with model size, the only way to understand the edges are different is through the edge embeddings. For datasets like MalNet-Tiny which rely heavily on the structure, while node and edge features are less useful, expander edges can interfere with the information propagation.

We observed that on molecular datasets, virtual nodes generally help substantially, while expander edges don't help much or can even hurt. The situation for image-based graphs is the opposite, where especially on Pascal-VOC, expander graphs are highly useful to the model, but virtual nodes can cause information bottleneck.

Virtual nodes can also make incorporating ideas like the batching mechanism used in GraphSAGE (Hamilton et al., 2017) more difficult, where having virtual nodes means every batch includes the whole graph. Expander edges can still be used with batching techniques.

## E. Universality of EXPHORMER

In this section, we detail the universal approximation properties of EXPHORMER-based graph transformers.

The work of Yun et al. (2020a) showed that for sequences, transformers are universal approximators, i.e., they can approximate any *permutation equivariant* function mapping one sequence to another arbitrarily closely when provided with enough parameters. A function $f : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$ is said to be permutation equivariant if $f(\mathbf{XP}) = f(\mathbf{X})\mathbf{P}$, i.e., if permuting the columns of an input $\mathbf{X} \in \mathbb{R}^{d\times n}$ results in the columns of $f(\mathbf{X})$ being permuted the same way.

**Theorem E.1** (Yun et al., 2020a)**.** *Let $1 \leq p < \infty$ and $\epsilon > 0$. For any function $f : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$ that is permutation equivariant, there exists a transformer network $g$ such that $\ell^p(f, g) < \epsilon$.*

The same work shows an extension to all (not necessarily permutation equivariant) sequence-to-sequence functions that are defined on a compact domain, say, $[0, 1]^{d\times n}$ provided that one uses a positional encoding. More specifically, for any transformer $g : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$, one can define a *transformer with positional encoding* $g_p : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$ such that $g_p(\mathbf{X}) = g(\mathbf{X} + \mathbf{E})$. The following results shows that trainable positional encodings allow a transformer to approximate any sequence-to-sequence continuous function on the compact domain.

**Theorem E.2** (Yun et al., 2020a)**.** *Let $1 \leq p < \infty$ and $\epsilon > 0$. For any continuous function $f : [0, 1]^{d\times n} \to \mathbb{R}^{d\times n}$ that is permutation equivariant, there exists a transformer with positional encoding, $g_P$, such that $\ell^p(f, g) < \epsilon$.*

Note that the above theorems hold for *full* (dense) transformers. However, under certain conditions about the sparsity pattern, one can obtain similar universality for sparse attention mechanisms (Yun et al., 2020b).

One important consideration is that the aforementioned results hold for functions on *sequences*. Since we are concerned with functions on *graphs*, it is interesting to ask what the implications are for graph transformers.

We follow the approach of Kreuzer et al. (2021): Given a graph $G$, we can view a node transformer as a function from $\mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ which uses the padded adjacency matrix of $G$ as a positional encoding. Similarly, an edge transformer takes as input a sequence $((i, j), \sigma_{i,j})$ with $i, j \in [n]$ and $i \leq j$ such that $\sigma_{i,j} = 1$ if $i$ and $j$ are connected in $G$ or $\sigma_{i,j} = 0$ otherwise. Any ordering of these vectors corresponds to the same graph. Moreover, we can capture the relevant functions going from $\mathbb{R}^{N(N-1)/2\times 2} \to \mathbb{R}^{N(N-1)/2\times 2}$ with permutation equivariance. Ideally, they can be approximated as closely as desired by suitable transformers on the edge input.

Now, simply observe (see Kreuzer et al. 2021) that one can choose a function (in either the node transformer or edge transformer case) that is (a.) invariant under node permutations and (b.) maps non-isomorphic graphs to distinct values. We would like to apply one of the above thoerems to such a function.

However, we cannot quite apply Theorem E.1 or Theorem E.2, as it is specifically for full transformers in which all nodes are pairwise connected in the attention interaction graph.

Therefore, we provide the following theorem, which shows that a slight modification of EXPHORMER is able to provide universal approximation properties using just $O(n)$ edges.

**Theorem E.3.** *Suppose $H$ is the attention graph of a modified* EXPHORMER*, which contains the $n$ graph nodes and potentially more virtual nodes. If $H$ contains self loops and either*

1. *$H$ contains at least one node which is connected to all $n$ graph nodes (e.g. at least one virtual node is included), or*
2. *$H$ uses expander graph attention augmented by a Hamiltonian path.*

*Then, it follows that a sparse transformer model, with positional encodings and an attention mechanism following $H$, can universally approximate continuous functions $f : [0, 1]^{d\times n} \to \mathbb{R}^{d\times n}$. That is, for any $1 < p < \infty$ and $\epsilon > 0$, there exists a sparse transformer network $g$, which uses the attention graph $H$ and some positional encodings, such that $\ell^p(f, g) < \epsilon$.*

Note that in the above theorem, only one of the enumerated conditions needs to be satisfied; in other words, one does not need to use both global attention and expander graph attention components (see Section 5.1). This is relevant, as for

certain datasets, it may be desirable to use one of these attention components but not both (see Appendix D), but universal approximation properties do not depend on this choice.

*Proof of Theorem E.3.* Let $H$ be the attention graph of a modified sparse EXPHORMER attention mechanism satisfying the conditions in Theorem E.3.

First, assume that condition (1) is satisfied. This implies that there is a subgraph of $H$ which forms a star graph on the $[n]$ graph nodes (plus, potentially, the virtual node). In this case, the existence of our desired $g$ is directly implied by the following universal approximation result of Zaheer et al. (2020):

**Theorem E.4** (Zaheer et al., 2020)**.** *Let $1 < p < \infty$ and $\epsilon > 0$. For any graph $H$ on $[n]$ that contains the star graph, we have that if $f \in [0,1]^{n \times d} \to \mathbb{R}^{n \times d}$ is a continuous function, then there exists a sparse transformer network $g$ (with trainable positional encodings) such that $\ell^p(f, g) < \epsilon$.*

Otherwise, assume that instead condition (2) holds. Then, we can apply Theorem 1 of Yun et al. (2020b). In particular, we note that since (a.) $H$ contains self-loops, (b.) $H$ contains a Hamiltonian path, and (c.) $\mathrm{diam}(H) = O(\log n)$ due to Corollary 4.4, it follows that the assumptions of the theorem of Yun et al. (2020b) are satisfied. (Note that in our setup, $\rho$ is the softmax function, which satisfies their Assumption 2.) Hence, the desired $g$ exists for any $f, p, \varepsilon$.

This completes the proof. □

This result implies that there exist sparse transformers based on the variant of EXPHORMER in Theorem E.3 which can solve graph isomorphism problems. This does not, however, imply the existence of an efficient algorithm for solving graph isomorphism problems, as we have not shown these networks can be efficiently identified; see further discussion by Kreuzer et al. (2021).