Low-Width Approximations and Sparsification for Scaling Graph Transformers

Hamed Shirzad
University of British Columbia
shirzad@cs.ubc.ca

Balaji Venkatachalam
Google
bave@google.com

Ameya Velingker
Google Research
ameyav@google.com

Danica J. Sutherland
University of British Columbia & Amii
dsuth@cs.ubc.ca

David P. Woodruff
Carnegie Mellon University & Google Research
dwoodruf@cs.cmu.edu

Abstract

Graph Transformers have shown excellent results on a diverse set of datasets. However, memory limitations prohibit these models from scaling to larger graphs. With standard single-GPU setups, even training on medium-sized graphs is impossible for most Graph Transformers. While the $\mathcal{O}(nd^2 + n^2d)$ complexity of each layer can be reduced to $\mathcal{O}((n + m)d + nd^2)$ using sparse attention models such as Exphormer for graphs with n nodes and m edges, these models are still infeasible to train on training on small-memory devices even for medium-sized datasets. Here, we propose to sparsify the Exphormer model even further, by using a small "pilot" network to estimate attention scores along the graph edges, then training a larger model only using $\mathcal{O}(n)$ edges deemed important by the small network. We show empirically that attention scores from smaller networks provide a good estimate of the attention scores in larger networks, and that this process can yield a large-width sparse model nearly as good as the large-width non-sparse model.

1 Introduction

The predominant story of the last half-decade of machine learning has been the runaway success of Transformer models (Vaswani et al., 2017), across domains from natural language processing (Vaswani et al., 2017; Devlin et al., 2018; Zaheer et al., 2020b) to computer vision (Dosovitskiy et al., 2020) and more recently in geometric deep learning (Dwivedi and Bresson, 2020; Kreuzer et al., 2021; Ying et al., 2021; Rampásek et al., 2022; Shirzad et al., 2023; Müller et al., 2023). Conventional ("full") Transformers, however, have a time and memory complexity of $\mathcal{O}(nd^2 + n^2d)$, where n is the number of entities (nodes, in the case of graphs), and d is the width of the network. Many attempts have been made to make Transformers more efficient (Tay et al., 2020). One major line of work involves *sparsifying* the attention mechanism, constraining attention from $\mathcal{O}(n^2)$ all-pairs to some smaller set of connections: BigBird, for sequential data, uses sliding windows, Erdős-Renyi auxiliary graphs, and universal connectors (Zaheer et al., 2020a); Exphormer, for graphs, uses input graph edges, expander graph edges, and universal connections (Shirzad et al., 2023). We refer to these as *sparse attention networks* and call (i, j) an *attention edge* if node i attends to node j.

Sparse graph attention models such as Exphormer (Shirzad et al., 2023) reduce each layer's complexity from $\mathcal{O}(nd^2 + n^2d)$ to $\mathcal{O}((n + m)d + nd^2)$, for graphs with n nodes, m edges, and width-d hidden features. Even so, training is still very memory-intensive for medium to large graphs. Also, for densely-connected graphs where $m = \Theta(n^2)$, there is no asymptotic improvement in complexity. Our goal is to scale effective graph Transformers, particularly Exphormer, to larger graphs.

NeurIPS 2023 New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers 2023).

One general approach for scaling models to larger graphs is based on batching techniques. Prominent approaches include egocentric subgraphs and random node subsets (Wu et al., 2022, 2023a). Egocentric subgraphs choose a node and include all of its k-hop neighbors, but the expander graphs used in Exphormer, which provide short connections from each node to all other nodes, guarantee that the size of these subgraphs will grow exponentially in the number of layers – prohibitively expensive for larger graphs. A similar issue arises with universally-connected nodes, whose representation depends on all other nodes. For uniformly random subset batching, as the number of batches b into which the graph is divided grows, each edge has chance $\frac{1}{b}$ to appear in a given step. Thus, b can not be very large without dropping important edges. A similar problem can happen in random neighbor sampling methods such as GraphSAGE (Hamilton et al., 2017). Although this model works well on message-passing neural networks (MPNNs) which only use the graph edges, using it for expander-augmented graphs will select only a small ratio of the expander edges, thereby breaking the universality properties provided by the expander graph. Expander graphs enable global information propagation and, when created using Hamiltonian cycles and added self-loops (Shirzad et al., 2023, Theorem E.3), produce a model that can provably approximate the full Transformer. Despite universality properties, not all of these edges turn out to be important in practice: we expect some neighboring nodes in the updated graph to have more of an effect on a given node than others. Thus, removing low-impact neighbors can improve the scalability of the model. The challenge is to identify low-impact edges without needing to train the (too-expensive) full model.

Our approach We first train a small-width network in order to estimate pairwise attention score patterns, which we then use to sparsify the graph and train a larger network. Training with a much smaller width $d' \ll d$ reduces the time and memory complexity by at least a factor of d/d'. It is not obvious *a priori* that attention scores learned from the smaller network will be a good estimator for those in the larger network, but we present an experimental study verifying that attention scores are surprisingly consistent as the network size reduces. We also introduce two additions to the model to improve this consistency. Training this initial network can still be memory-intensive, but as the small width means the matrix multiplications are small, it is practical to train this initial model on CPU for systems with significant amounts of RAM without needing to distribute the calculation. Once this initial model is trained and attention scores are found, scores can be reused for any training of the second network. Attention scores can be shared as edge features for future use.

As mentioned previously, we use the attention scores obtained from the trained low-width network to sparsify the graph. By selecting a fixed number of edges per attention layer for each node, we reduce the complexity of each layer to $O(nd^2 + ndc)$, where c is a constant number of neighbors per node. This sparsification alleviates the effect of a large number of edges, and allows for initial training with a larger degree expander graph, since most of the expander edges will be filtered for the final network. This sparsification differs from conventional graph sparsification algorithms (for MPNNs) in two ways. First, we use expander edges, self-loops, and graph edges and sparsify the combination of these patterns together. Second, this sparsification is layer-wise, which means that in a multi-layer network the sparsity pattern will vary from layer to layer. Another advantage of this approach is that the fixed number of neighbors for each node enables matrix calculations instead of the edge-wise calculations used by Kreuzer et al. (2021); Shirzad et al. (2023), improving the speed of the model. After this reduction, batching can be done based on the edges over different layers, enabling Transformers to be effectively batchable while still effectively approximating the main Transformer model.

To summarize, the contributions of this paper are as follows: 1.) We experimentally analyze the similarity of attention scores for networks of different widths, and propose two small architectural changes to improve this similarity. 2.) We propose layer-wise sparsification, by sampling according to the learned attention scores. 3.) Our two-phase training process decreases memory consumption significantly, while maintaining competitive accuracy.

2 Related Work

Graph Transformer Architectures Attention mechanisms were proposed in early (message-passing) Graph Neural Network (GNN) architectures such as Graph Attention Networks (GAT) (Veličković et al., 2018), where they guide node aggregation among neighbors, without using positional encodings. GraphBert (Zhang et al., 2020) finds node encodings based on the underlying graph structure.

Subsequent works have proposed full-fledged graph transformer models that generalize sequence transformers (Dwivedi and Bresson, 2020) and are not limited to message passing between nodes of the input graph; these include Spectral Attention Networks (SAN) (Kreuzer et al., 2021), Graphormer Ying et al. (2021), GraphiT (Mialon et al., 2021), etc. GraphGPS (Rampásek et al., 2022) combines attention mechanisms with message passing, allowing the best of both worlds.

Sparse Transformers Several recent works have proposed various scalable graph transformer architectures. NAGphormer (Chen et al., 2022a) and Gophormer (Zhao et al., 2021) use a sampling-based approach. On the other hand, Difformer (Wu et al., 2023a) proposes a continuous time diffusion-based transformer model. Exphormer (Shirzad et al., 2023) proposes a sparse graph that combines the input graph with edges of an expander graph as well as virtual nodes. They show that their model works better than applying other sparse transformer methods developed for sequences. Another work, NodeFormer (Wu et al., 2022), which is inspired by Performer (Choromanski et al., 2021), uses the Gumbel-Softmax operator as a kernel to efficiently propagate information among all pairs of nodes. SGFormer (Wu et al., 2023b) shows that just using a one layer transformer network can sometimes improve the results of GCN-based network and the low-memory footprint can help scale to large networks. Perhaps most conceptually similar to our work is Skeinformer (Chen et al., 2022b), which uses sketching techniques to accelerate self-attention.

3 Preliminaries and Notation

Notation We use H to denote the attention pattern, and $\mathcal{N}_H(i)$ the neighbors of node i under that pattern. Our primary "driver" is then h-head attention: using $\odot$ for element-wise multiplication,

$$\textsc{Attn}_H(\mathbf{X})_{:,i} = \mathbf{x}_i + \sum_{j=1}^{h} \mathbf{V}^j \cdot \sigma\left(\left(\mathbf{E}^j \odot \mathbf{K}^j\right)^T \mathbf{Q}_i^j + \mathbf{B}^j\right),$$

where $\mathbf{V}^j = \mathbf{W}_V^j \mathbf{X}_{\mathcal{N}_H(i)}$, $\mathbf{K} = \mathbf{W}_K^j \mathbf{X}_{\mathcal{N}_H(i)}$, and $\mathbf{Q}_i^j = \mathbf{W}_Q^j \mathbf{x}_i$, are linear mappings of the node features for the neighbors $\mathbf{X}_{\mathcal{N}_H(i)}$, and $\mathbf{E}^j = \mathbf{W}_E^j \mathcal{E}_{\mathcal{N}_H(i)}$ and $\mathbf{B}^j = \mathbf{W}_B^j \mathcal{E}_{\mathcal{N}_H(i)}$ are linear maps of the edge features $\mathcal{E}$, which is a $d_E \times |\mathcal{N}_H(i)|$ matrix of features for the edges coming in to node i. Compared to prior work, we introduced $\mathbf{B}^j$ as a simpler route for the model to adjust the importance of different edge types. We always use $d_E = d$, and have each layer output features of the same width as its input, so that each of the $\mathbf{W}^j$ parameter matrices are $d \times d$.

Exphormer EXPHORMER is an expander-based sparse attention mechanism for graph transformers that uses $O(|V| + |E|)$ computation, where $G = (V, E)$ is the underlying input graph. Exphormer creates an interaction graph H that consists of three main components: (1) Edges from the original graph, which captures the structure of the local neighborhood; (2) Virtual nodes, which are connected to all the nodes; and (3) Expander graph edges.

Exphormer uses a constant-degree random *expander graph*, with $\mathcal{O}(n)$ edges. Expander graphs have several useful theoretical properties related to spectral approximation and random walk mixing, which allow propagating information between pairs of nodes that are distant in the input graph G without explicitly connecting all pairs of nodes. The expander edges introduce many alternative short paths between the nodes and avoid the information bottleneck that can be caused by the virtual nodes.

Our network compared to Exphormer We use Exphormer as the base model because it is an efficient sparse model. We cannot fit a full transformer in memory for a dataset like Physics (with 34K nodes). We add self-loops for every node and use $d/2$ random Hamiltonian cycles to construct our expander graph as described in (Shirzad et al., 2023, Appendix C, Theorem E.3). We do not add virtual nodes in the first attention score estimator network; the resulting network is still a universal approximator. The best results from the Exphormer paper combined MPNNs with the Transformer architecture, but in this work for scalability reasons we avoid the MPNN component as well. We also make two additional changes explained in Section 4.1.

4 Method

Our method is a two-phase training process. The goal of the first model, called the *Attention Score Estimator Network*, is to estimate the attention scores for a larger network. This model is not particularly accurate; its only goal is for each node to learn the more important neighbors. The second model uses the learned attention scores per layer of the first network to sparsify the attention edges in that layer. This model will be the final predictor, and its hyperparameters will be tuned for the best results.

4.1 Attention Score Estimator Network

For this network, we use a width of 4 or 8, with just one attention head, in our training. Unless training fails drastically, we do not intend to focus on improving the accuracy in this step. This network will be trained with as many layers as the final network we want to train. Because it is so narrow, it has many fewer parameters and hence much less memory and time complexity; it is thus much cheaper to train. Moreover, we only need to do this training once per number of layers we consider, even if we try many hyperparameters for the final model. Compared to Exphormer, we use a much higher-degree expander graph: 30 instead of the 6 used for most transductive graphs by Shirzad et al. (2023). As most of the considered datasets do not have edge features, we use a learnable embedding for each type of edge (graph edge, expander edge, or self-loop). We also make two small changes to the architecture and the training process of this model, discussed below.

In Section 5.1, we show experimentally that the low-width network is a good estimator of the attention scores for a large-width network.

Normalizing V Having a smaller attention score, $\alpha_{ij} < \alpha_{ij'}$, does not necessarily mean that j's contribution to i's new features is smaller than that of j': we may have $\|\mathbf{V}_j\| \gg \|\mathbf{V}_{j'}\|$. Although Transformers typically use layer normalization, the do not typically do so after mapping X to $\mathbf{V}$. We normalize the rows of $\mathbf{V}$ to have the same vector sizes for all nodes. In our experiments, normalizing to size one reduced the performance significantly; however, adding a learnable global scale s, so that $\mathbf{V}_i$ becomes $s\,\mathbf{V}_i/||\mathbf{V_i}||_2$, maintained performance while making attention scores more meaningful.

Variable Temperature One of the side goals is to have sharper attention scores, guiding the nodes to get their information from as few nodes as possible. Using temperature in the attention mechanism is a natural way to do this, where logits before the softmax will be divided by a temperature factor τ. Normal attention corresponds to $\tau = 1$; smaller τ means sharper attention scores. However, setting the temperature to a small value from the beginning will make the random initialization more significant, and increase the randomness in the training process. Instead, we start with $\tau = 1.0$ and gradually anneal it to 0.05 by the end of the training. We set an initial phase for c epochs where we use $\tau = 1$; this lets the model learn which neighbors are more important for each node slowly. Then we multiply the τ with a factor f after each epoch, getting a temperature in epoch $t > c$ of $\max(f^{t-c}, 0.05)$. For fast-converging models, we use $c = 5, f = 0.98$; for slower-converging ones, we use $c = 10, f = 0.95$.

4.2 Sparser Attention Pattern

The memory and time complexity of Exphormers is linearly dependent on the number of edges. Reducing the number of edges can alleviate memory consumption. Additionally, a sparser pattern means that batching techniques such as ego subgraphs can be effectively applied to the model. In this work, we analyze how effective the sparser model can work and up to what factor we can sparsify. Further improvements, such as batching techniques, are planned as future work. Having the same degree for each node's attention pattern also means that attention can be calculated as (much-more-optimized) standard matrix multiplications, rather than the propagation techniques used in Exphormer and SAN (Kreuzer et al., 2021).

To sparsify the graph, in each epoch, we sample a new set of edges according to the learned attention scores from the smaller network. The reason why we do this rather than a simpler strategy such as selecting top-scored edges is that in many cases several nodes can have very similar node features. If we assume nodes $u_1, u_2, \ldots, u_p$ from the neighbors of node v have almost the same features, and if the attention scores for these nodes are $\alpha_1, \alpha_2, \ldots, \alpha_p$, any linear combination of $\sum_{i=1}^{p} \alpha_i = \alpha$ will

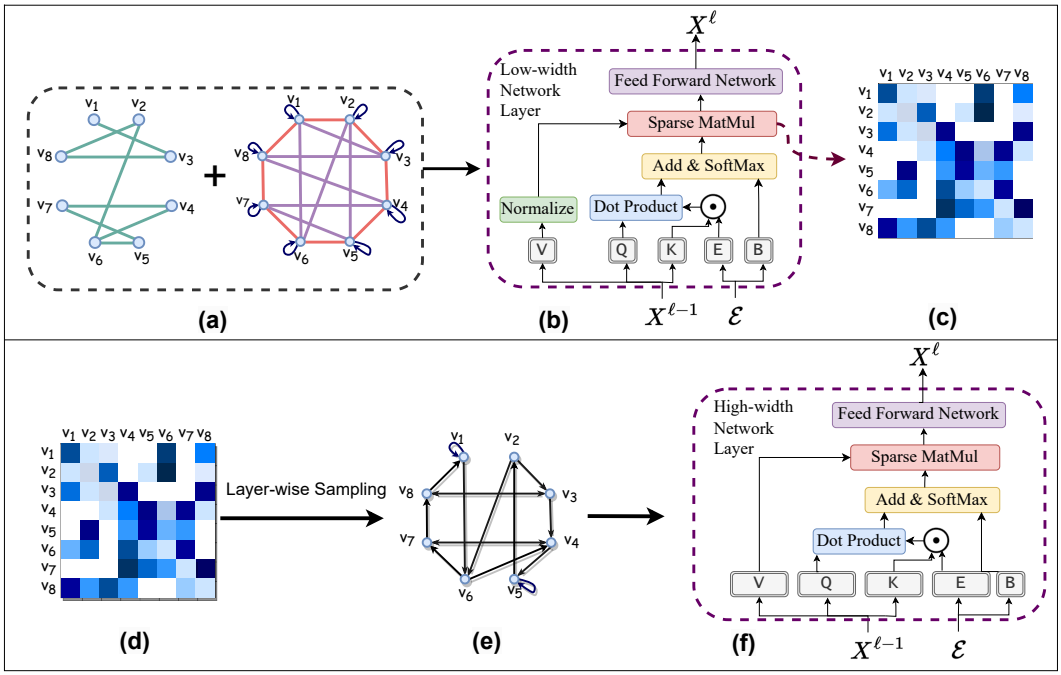

Figure 1: Steps of our method. **(a)** The attention mechanism for the attention score estimator network combines graph edges with an expander graph and self-loops. The expander graphs are constructed by combining a small number of Hamiltonian cycles – here two, in red and in purple – then confirming the spectral gap is large enough. **(b)** Self-attention layers in the estimator network use this sparse attention mechanism; its self-attention layers normalize $\mathbf{V}$. **(c, d)** Attention scores are extracted from this network for each layer, and used to sample, in **(e)**, a sparse directed graph, which becomes the attention graph for the final network **(f)**. This network, with a much larger feature dimension, does not normalize $\mathbf{V}$.

lead to the same result. If features are exactly the same α will be divided between these nodes, and even if α is large, each node's attention score from v can be small. By sampling, we have a total α chance of selecting any of the nodes $u_{1:p}$. On each epoch, we sample a new set of edges for each node from its original neighborhood.

4.3 Theoretical underpinnings for sparsification

There is a large body of work for sketching and approximating matrices. Our model follows the sampling approach and the approximation guarantee of Achlioptas et al. (2013, Theorem 4.3). Specifically, they show that the sketch values are not significantly different from the original matrix, which we also observe in practice (Section 5.1). Our method differs from their work in that we sample without replacement. Sampling without replacement works better in practice, and the normalization before edge selection in our second phase would not make a difference for duplicate samples.

5 Experimental Results

5.1 Attention Score Estimation

To show how well the smaller network estimates the attention scores for a larger network, we conduct experiments on two smaller datasets, where we can reasonably train the full network at higher width. To this end, we use Actor (Lim et al., 2021) and Photo (Shchur et al., 2018) datasets. We train the network for hidden dimensions, h varying from 4 to 64 for both datasets. For each h we train the network 100 times. We consider the distribution of attention scores for each node, and estimate the energy distance (Székely and Rizzo, 2013; an instance of the maximum mean discrepancy, Sejdinovic et al., 2013) for that node across each pair of h sizes.

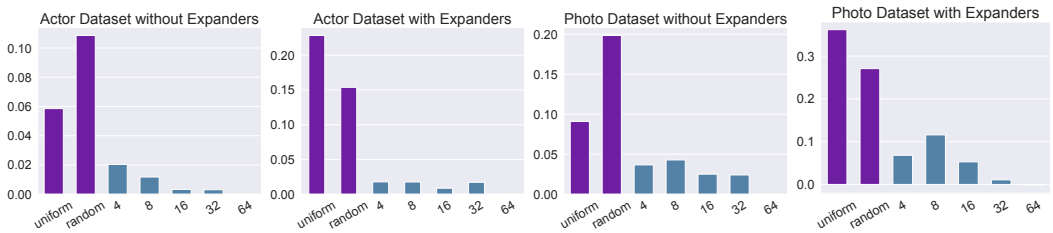

Figure 2: Energy distance between the attention scores of various networks to a network of width 64.

Model	Actor	Computer	Photo	CS	Physics
GCN	33.23±1.16	89.65 ± 0.52	92.70 ± 0.20	92.92 ± 0.12	96.18 ± 0.07
GAT	33.93±2.47	90.78 ± 0.13	93.87 ± 0.11	93.61 ± 0.14	96.17 ± 0.08
GRAPHSAINT	-	90.22 ± 0.15	91.72 ± 0.13	94.41 ± 0.09	96.43 ± 0.05
NODEFORMER	36.9 ± 1.0	-	-	-	-
EXPHORMER+GCN	38.68 ± 0.38	91.59 ± 0.31	95.27 ± 0.42	95.77 ± 0.15	97.16 ± 0.13
EXPHORMER W.O. MPNN	39.01 ± 0.69	91.16 ± 0.26	95.36 ± 0.17	95.19 ± 0.26	96.40 ± 0.20
Our model	38.44 ± 0.42	90.93 ± 0.08	94.43 ± 0.34	95.00 ± 0.15	96.70 ± 0.05

Table 1: Comparison of our model with other GNNs.

	Exphormer+GCN	Exphormer w.o. MPNN	BigBird+GCN	Performer+GCN	Nodeformer	Our Model
ogbn-arxiv	72.44 ± 0.28	71.27 ± 0.27	71.16 ± 0.19	70.92 ± 0.04	59.90 ± 0.42	70.82 ± 0.24

Table 2: Comparison of Transformer-based models' results on ogbn-arxiv.

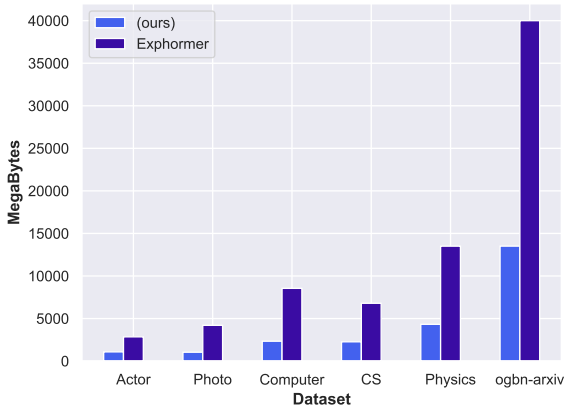

Figure 3: Comparing memory usage of our model with the Exphormer with expander degree 30.

We have conducted this experiment both when learning with just graph edges and when adding expander and self-loop edges. It might be that the model, just by examining the category of the edges, may give a lower score to one type, making distributions will seem more similar even though it has not identified a small number of important neighbors as we want. We show that in the presence of only one type of edge, the model can still consistently estimate which nodes should have a higher attention score.

As baselines, we compare attention scores from our model with the uniform distribution on the neighbors (each score for neighbors of node i will be $\frac{1}{d_i}$), and to a distribution with logits uniform over $[-8, 8]$. The choice of 8 here is because in the network we clip the logits with an absolute value higher than 8. Figure 2 shows that even width-4 networks provide far superior estimates of attention scores than these baselines.

5.2 Model quality

We conduct experiments on six transductive graph datasets: CS, Physics, ogbn-arxiv are academic graphs, Photo and Computer are co-purchasing graphs, and Actor is a heterophilic graph dataset from Lim et al. (2021). On CS, Physics, Photo, Computer, and Actor we use a random split of 60%/20%/20%. For the ogbn-arxiv, we use the standard split, based on the years of the publications. We keep the random split consistent as the smaller network should have access to the same information as the second network. Further dataset details are in Appendix A.

We train the smaller network once and then for the second network, we always use the same initial network's learned attention scores. Attention scores will be collected from the network training step with the highest validation accuracy. We measure the memory usage of our model as the maximum memory between the first and second network and compare it with both versions of Exphormer — with and without an MPNN.

Results are shown in Tables 1 and 2 and Figure 3. Our model obtains comparable accuracies with greatly-reduced memory consumption.

6 Conclusion & Future Work

In this work, we analyzed the alignment of the attention scores among models trained with different widths. We found that usually the smaller network's attention scores distributions align with the larger network's. Based on this observation we used a sampling algorithm to sparsify the graph on each layer and sampled a smaller number of edges per layer. As a result of these two steps, the model's memory consumption reduces significantly while achieving a competitive accuracy.

This strategy sets up opportunities for batching techniques that were not feasible with expander graphs of large degree. However, batching techniques and experiments on large datasets are planned future work. There are a number of hyperparameters in our model; in the future we want to tune them to improve accuracy. Theoretical analysis on the lower-width networks estimation power of the attention score and sparsifying method will also be fruitful for future work.

Acknowledgements

This work was supported in part by the Natural Sciences and Engineering Resource Council of Canada, the Fonds de Recherche du Québec - Nature et technologies (under grant ALLRP-57708-2022), the Canada CIFAR AI Chairs program, the BC DRI Group, Calcul Québec, and the Digital Resource Alliance of Canada.

References

Achlioptas, D., Karnin, Z. S., and Liberty, E. (2013). Near-optimal entrywise sampling for data matrices. *Advances in Neural Information Processing Systems*, 26.

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

Chen, Y., Zeng, Q., Hakkani-Tur, D., Jin, D., Ji, H., and Yang, Y. (2022b). Sketching as a tool for understanding and accelerating self-attention for long sequences. In Carpuat, M., de Marneffe, M., and Ruíz, I. V. M., editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 5187–5199. Association for Computational Linguistics.

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. (2021). Rethinking attention with performers. In *ICLR*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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 preprint arXiv:2010.11929*.

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

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

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. *CoRR*, abs/2103.09430.

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

Lim, D., Hohne, F., Li, X., Huang, S. L., Gupta, V., Bhalerao, O., and Lim, S. N. (2021). Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. *Advances in Neural Information Processing Systems*, 34:20887–20902.

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

Müller, L., Galkin, M., Morris, C., and Rampášek, L. (2023). Attending to graph transformers. *arXiv preprint arXiv:2302.04181*.

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

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

Sejdinovic, D., Sriperumbudur, B., Gretton, A., and Fukumizu, K. (2013). Equivalence of distance-based and RKHS-based statistics in hypothesis testing. *The Annals of Statistics*, 41(5):2263 – 2291.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. (2018). Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.

Shirzad, H., Velingker, A., Venkatachalam, B., Sutherland, D. J., and Sinop, A. K. (2023). Exphormer: Sparse transformers for graphs. In *ICML*.

Székely, G. J. and Rizzo, M. L. (2013). Energy statistics: A class of statistics based on distances. *Journal of Statistical Planning and Inference*, 143(8):1249–1272.

Tay, Y., Dehghani, M., Bahri, D., and Metzler, D. (2020). Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *NeurIPS*, pages 5998–6008.

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

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

Wu, Q., Zhao, W., Li, Z., Wipf, D. P., and Yan, J. (2022). Nodeformer: A scalable graph structure learning transformer for node classification. *NeurIPS*, 35:27387–27401.

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

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

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontañón, S., Pham, P., Ravula, A., Wang, Q., Yang, L., and Ahmed, A. (2020a). Big Bird: Transformers for longer sequences. In *NeurIPS*.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., et al. (2020b). Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

Zhang, J., Zhang, H., Xia, C., and Sun, L. (2020). Graph-Bert: Only attention is needed for learning graph representations. *arXiv preprint arXiv:2001.05140*.

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

A Dataset Descriptions

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

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.

Actor dataset Dataset created by co-occurrence of actor, director, writer and film on a wikipedia page. The node features are based on the wikipedia page. The goal is to correctly classify nodes into one of five categories (Pei et al., 2020).

Dataset	Graphs	Nodes	Edges	Prediction Level	No. Classes	Metric
ogbn-arxiv	1	169,343	1,166,243	Node	40	Accuracy
Amazon Computer	1	13381	245778	Node	10	Accuracy
Amazon Photo	1	7487	119043	Node	8	Accuracy
Coauthor CS	1	18333	81894	Node	15	Accuracy
Coauthor Physics	1	34493	247962	Node	5	Accuracy
Actor	1	7600	29926	Node	5	Accuracy

Table 3: Dataset statistics

B Hyperparameters

Both for the Exphormer without MPNN models and our initial model, we have used an expander graph of degree 30. All numbers in the Tables 1 and 2 and Figure 3 are reported with the same expander degree. For the initial network except for the ogbn-arxiv we have trained the initial network with a width of 4. We train the initial network until convergence. For the second network, we finetune hyperparameters for the best results. Hyperparameters have been reported in Table 4.

Hyperparameter	OGBN-Arxiv	Computer	Photo	CS	Physics	Actor
Num Layers	3	4	4	4	4	4
Constant Node Degree	6	5	5	5	5	5
Low-Width Dim (d')	8	4	4	4	4	4
Hidden Dim (d)	96	80	56	64	64	16
Num Heads	2	2	2	2	2	2
Dropout	0.3	0.4	0.5	0.4	0.4	0.3
Learning Rate	0.01	0.001	0.001	0.002	0.001	0.001
Num Epochs	600	150	250	120	80	100

Table 4: Hyperparameters used for training the networks. For the attention score estimator network, the learning rate is similar to the final network and training continues until the convergence. We do not use dropout in the attention score estimator network, and we only use one attention head in these networks. The number of layers is always equal between both networks.