Exploring the Efficacy of Attention Mechanism in Graph Neural Networks

Venkata Saikiranpatnaik Balivada [1] **Apoorva Kumar** [1] **Ashutosh Parida** [1]

Abstract

The transformer architecture is quite ubiquitous in both language and vision tasks. The architecture is responsible for the emergence of foundation models, that are touted as general, robust, and scalable solvers, useful for a wide variety of tasks. However, despite years of research, attention-based models have had limited success in exhibiting these important and desirable properties on graph-structured data. In this report, we attempt to explain this discrepancy by analyzing the core properties inherent to popular attention-based graph neural models. Finally, we notice that Graph Transformers lack generality in their graph encoding mechanism and humbly propose a greedy algorithm with competitive performance on EDGES and TRIANGLES tasks.

1. Introduction

Graph neural networks are a set of neural architectures that are designed to solve classification and regression tasks on graph-structured input. There are plenty of applications in this domain. Popular examples include graph clustering (Kipf & Welling, 2017), social media recommendation (Wang et al., 2010), bot recognition in social networks (Hamilton, 2020), molecular analysis (Wang et al., 2023), topic classification (McCallum et al., 2000) and many more. Solving graph problems is particularly challenging since the data is not sampled from an independent distribution. The interdependence is instead encoded in a graph structure and provided as input. To handle this, researchers proposed new methods to learn from graphical data. In other words, the graph neural models do not just approximate the task as a function of data $f(X)$ but also the graph structure $f(X, A)$ [1].

In other news, the paper (Vaswani et al., 2023) introduces a new Transformer architecture for the machine translation task. The architecture was hugely influential in a wide variety of tasks as evidenced by (Devlin et al., 2019) and (Brown et al., 2020). Large models were being trained on internet-scale data and then fine-tuned for specific tasks. Researchers have then figured out that the models specialized without even fine-tuning and simply by prompting the model (Raffel et al., 2023). Transformers were also adapted to vision tasks (Dosovitskiy et al., 2021). Following this, researchers have built foundation models (Bommasani et al., 2022) to take inputs in multiple modalities while also performing surprisingly well on a wide range of tasks with minimal fine-tuning. Such lofty success is attributed to the nature of transformers to generalize, be robust, and importantly scale to large amounts of data. Since then, researchers have attempted to explain these desirable properties and the work done by (Park & Kim, 2022) is a step forward in that direction. We are heavily inspired by their work.

While foundation models have had a huge impact on input modalities such as text, images, and speech, the best-performing models for graph-structured problems still incorporate domain knowledge into their models. As a result, there is an explosion of architectures (You et al., 2021), and researchers are in search of a generally applicable model. For the purpose of this project, we limit ourselves to analyzing Graph Attention Networks (GATs) (Brody et al., 2022) and Graph Transformers (GTs) (Rampášek et al., 2023).

We make the following contributions to the best of our ability,

1. Similar to the AlterNet architecture proposed in (Park & Kim, 2022), Graph Transformers (GTs) are a combination of Message Passing and Global Self Attention mechanisms. We explain the purpose of each important component in Graph Transformers.

2. We discuss the limitations of GATs compared to Graph Transformers and we explain why GATs have competitive performance on natural graph problems despite the limitations.

3. We discuss issues with the generalization of positional encoding mechanisms in GTs and propose a new algorithm to learn the best encoding for the task.

Please find the code for our experiments at `https://github.com/bvskp-projects/graphgps`.

[1] A represents the adjacency matrix of the graph and has all the information necessary about the graph structure.

2. Background

In this section, we discuss concepts that are helpful to understand our work better.

2.1. Spectral Graph Theory

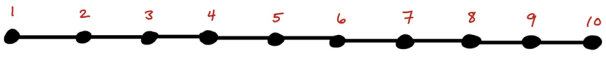
Spectral Graph Theory deals with eigenvalues and eigenvectors of the Graph Laplacian. The Theory not only provides a strong theoretical foundation for graph neural networks but also provides some mechanisms to make them better. Here we discuss a notion of frequency and signals and how the eigenvalues and eigenvectors relate to it. First, the Laplacian is defined as the difference between the degree matrix and the adjacency matrix, $\mathcal{L} = D - A$. This matrix is symmetric for undirected graphs and hence posseses real eigenvalues and eigenvectors. Below is an important result (Hamilton, 2020)

$$x^T \mathcal{L} x = \sum_{(a,b)\in\mathcal{E}} (x(a) - x(b))^2 \tag{1}$$

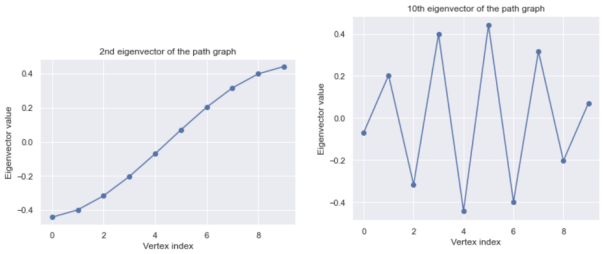
Observe that the quadratic form measures how the vector x changes across edges, i.e. when x changes slowly across edges then the quadratic value is low, and when x changes wildly the value is high. Say x is an eigenvector ψ_i, then $x^T \mathcal{L} x = \psi_i \mathcal{L} \psi_i = \lambda_i \psi_i^T \psi_i = \lambda_i$. Hence, the eigenvector with the lowest eigenvalue changes the slowest across edges (does not change at all), and the eigenvector with the largest eigenvalue changes the most across edges. This notion is very similar to signals having low and high frequencies. Thus, we can treat the eigenvectors of a Graph Laplacian as basis signals and their corresponding eigenvalues as the corresponding frequencies. We look at a path graph for demonstration 1a. A low-frequency signal is illustrated in 1b and high-frequency one in 1c, both represented by low and high eigenvectors respectively. Note that the smallest eigenvalue is always zero, so we typically ignore it.

2.2. Homophilic and Heterophilic Tasks

Early applications of graph neural networks involved node classification tasks where two nodes are connected by an edge if they are similar and thus tend to have similar labels (Hamilton, 2020). These graphs are said to exhibit homophily. Low-frequency signals are great representations of such graphs since the node values change very slowly across edges. Graph clustering and topic classification are great examples of this type of task. On the other hand, when nodes are connected by an edge if they belong to opposite classes, they exhibit heterophily. High-frequency signals form great representations since the node values change wildly across edges. The graph coloring problem is a great example.



(a) A path graph with 10 vertices



(b) Smallest eigenvector (c) Largest eigenvector

Figure 1. Demonstration of signals and their frequencies in an example graph

2.3. Message Passing Graph Neural Networks

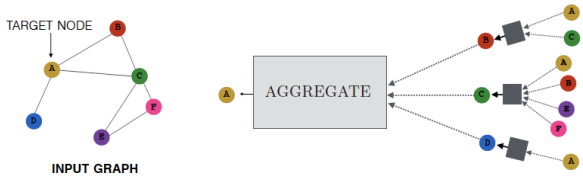


Figure 2. An illustration of neural message passing mechanism.

Message passing (Gilmer et al., 2017) is an influential framework for solving graph neural problems and is proposed as an extension of the convolution mechanism to the graph domain (Bronstein et al., 2017). In this framework, we pass messages from each node to their neighboring nodes in a messaging step. These messages are then aggregated from the neighbors to produce the hidden embeddings for the next layer (Hamilton, 2020). See figure 2.

Computing the $k+1$th layer embedding $h_i^{(k+1)}$ requires the following operations

$$m_{ij}^{(k+1)} = \phi^{(k+1)}(h_i^k, h_j^k, e_{ij}^k) \tag{2}$$

$$agg_i^{(k+1)} = \bigoplus_{j\in\mathcal{N}(i)} m_j^{(k+1)} \tag{3}$$

$$h_i^{(k+1)} = \gamma^{(k+1)}(h_i^k, agg_i^{(k+1)}) \tag{4}$$

Eq 2 refers to the messaging step where a message is constructed using the source node, target node, and edge embeddings of the previous layer. Next, we collect the messages from our neighbors and then aggregate them using some

reduction operation such as mean or sum, eq 3. Finally, we compute the hidden embeddings for that layer and node using an update operation, eq 4. MP-GNNs are good at capturing graph structure and next, we discuss Graph Attention Network, an example message-passing network.

2.4. Graph Attention Networks

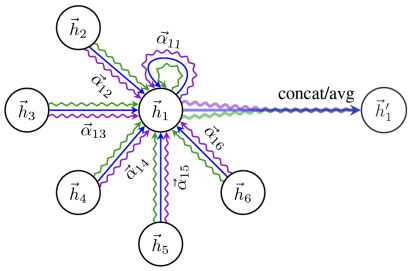


Figure 3. An illustration of graph attention networks.

(Veličković et al., 2018) first introduced the attention mechanism in graph neural networks and was soon superseded by (Brody et al., 2022). At a high level, each node aggregates a weighted sum of its neighbor's embeddings. The weights are computed using additive attention mechanism (Bahdanau et al., 2016) and is different from scaled dot product attention introduced in (Vaswani et al., 2023). The message passing mechanism is illustrated in 3. The message phase involves computing (a) the attention score α_{uv} and (b) the hidden embedding of the source node z_j. The aggregation is then simply a score-weighted sum of the embeddings $\sum_{v \in \mathcal{N}(u)} \alpha_{uv} z_v$. See the original paper (Brody et al., 2022) for more details.

One important distinction from the popular transformers architecture is that we only attend to our neighbors. Next, we discuss how to alleviate this restriction using graph transformers.

2.5. Graph Transformers

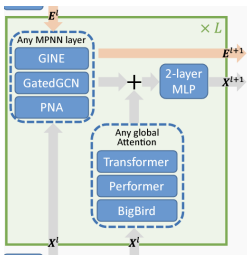


Figure 4. An illustration of the Graph Transformer architecture

Recently, researchers have been combining the Transformers architecture with the message passing mechanism. We use the popular term Graph Transformers to describe all such architectures. See figure 4 for a high-level overview of each layer of the network as described in the work (Rampášek et al., 2023). First, the network computes node and edge embeddings to run the message passing phase. Then, the message passing output is added to the output of the global self-attention mechanism. Finally, the sum is passed through a 2-layer MLP at the end. Note that we replace positional encodings (of the original Transformer) with graph encodings to encode graph structure into the global self-attention and the message passing modules.

2.6. Graph Encodings for Transformers

Encoding type	Description	Examples
Local PE *node features*	Allow a node to know its position and role within a local cluster of nodes. *Within a **cluster**, the closer two nodes are to each other, the closer their local PE will be, such as the position of a word in a sentence (not in the text).*	• Sum each column of non-diagonal elements of the m-steps random walk matrix. • Distance between a node and the centroid of a cluster containing the node.
Global PE *node features*	Allow a node to know its global position within the graph. *Within a **graph**, the closer two nodes are, the closer their global PE will be, such as the position of a word in a text.*	• Eigenvectors of the Adjacency, Laplacian [15, 36] or distance matrices. • SignNet [39] (includes aspects of relative PE and local SE). • Distance from the graph's centroid. • Unique identifier for each connected component of the graph.
Relative PE *edge features*	Allow two nodes to understand their distances or directional relationships. *Edge embedding that is correlated to the distance given by any global or local PE, such as the distance between two words.*	• Pair-wise node distances [38, 3, 36, 63, 44] based on shortest-paths, heat kernels, random-walks, Green's function, graph geodesic, or any local/global PE. • Gradient of eigenvectors [3, 36] or any local/global PE. • PEG layer [57] with specific node-wise distances. • Boolean indicating if two nodes are in the same cluster.
Local SE *node features*	Allow a node to understand what substructures it is a part of. *Given an SE of radius m, the more similar the m-hop subgraphs around two nodes are, the closer their local SE will be.*	• Degree of a node [63]. • Diagonal of the m-steps random-walk matrix [16]. • Time-derivative of the heat-kernel diagonal (gives the degree at $t = 0$). • Enumerate or count predefined structures such as triangles, rings, etc. [6, 68]. • Ricci curvature [54].
Global SE *graph features*	Provide the network with information about the global structure of the graph. *The more similar two graphs are, the closer their global SE will be.*	• Eigenvalues of the Adjacency or Laplacian matrices [36]. • Graph properties: diameter, girth, number of connected components, # of nodes, # of edges, nodes-to-edges ratio.
Relative SE *edge features*	Allow two nodes to understand how much their structures differ. *Edge embedding that is correlated to the difference between any local SE.*	• Pair-wise distance, encoding, or gradient of any local SE. • Boolean indicating if two nodes are in the same substructure [5] (similar to the gradient of sub-structure enumeration).

Figure 5. Kinds of graph encodings.

We now discuss the nature of these graph encodings. The sinusoidal mechanism described in (Vaswani et al., 2023) cannot be used here, since unlike text, graphs are not sequential input. Here, we discuss two popular graph encodings. In **Laplacian Position Encoding (LapPE)** (Dwivedi & Bresson, 2021), the graph encoding consists of k smallest eigenvectors. Intuitively, this encoding can be used to extract low-frequency information. For example, this can be helpful for edge detection or computing node similarity. In **Random Walk Structural Encoding (RWSE)** (Dwivedi et al., 2022), we compute the probability of a node coming back to itself in a random walk with j hops, where j ranges from $1, .., k$. Intuitively, this encoding determines the local structure that the node is part of. For example, the encoding does well in counting the number of triangles in a graph because the random walk comes back in 3 hops in a triangular structure (Müller et al., 2023). See figure 5 for more.

3. On properties of a Graph Transformer

Graph Transformers differ from vanilla Transformers in that they have an additional Message Passing component and that they use graph-based encodings. So, first, we discuss the importance of these major components and then discuss some limitations of popular graph encodings.

3.1. Ablation Study

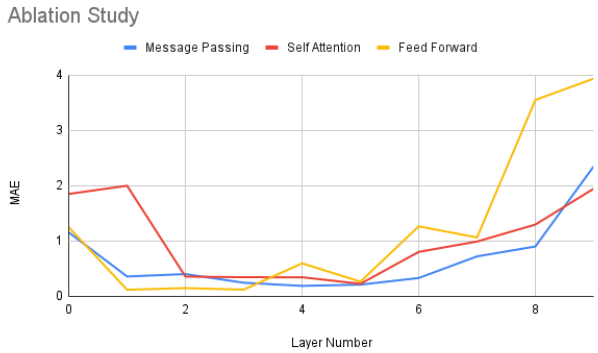


Figure 6. Ablation Study to analyze the importance of each component.

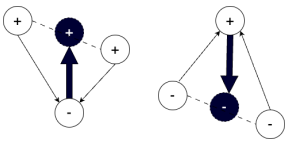
In this setup, we first train a 10-layer model for the ZINC dataset (Gómez-Bombarelli et al., 2018) using RWSE encoding. Then, during inference, we disable one of Message Passing, Global Self Attention, or Feed Forward network at each layer. The results are reported in figure 6 [2] and reveal the following insights,

3.1.1. Attention plays an important role in early layers

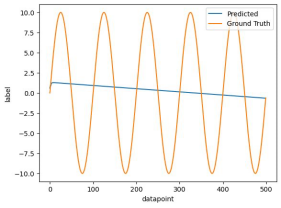
The global self attention module shoulders the responsibilities of Message Passing (in a GNN) in the early layers. This is important since Message Passing layers act as aggressive low pass filters and remove too much information from the input. For this reason, it is crucial that their influence is stronger towards the end than the start.

(NT & Maehara, 2019) give a formal reasoning for why an unweighted combination filters out high frequencies. Here, we give an **informal** argument for why any convex combination of neighborhood smoothens the signal. Consider a high frequency signal such as the one shown in figure 7a where a negative value is sandwiched between two positive values. A convex combination of the neighbors of the middle element brings it closer to its neighbors. Generalizing this to the path graphs, the values are brought closer and closer to the horizontal axis, thus reducing the frequency of

[2]MAE is mean average error where lower is better.



(a) Any convex combination dims out high frequencies



(b) GAT heavily smoothens the initial signal

Figure 7. Smoothening effect demonstration

Table 1. Depth scaling in GAT

Depth	2	3	4	5
Cora (Acc %)	84.5	78.9	73.4	64.2

the signal. We demonstrate this effect with GAT on a path graph of length 500 in figure 7b.

This effect also means that GATs scale poorly with increasing the depth of the network. We demonstrate this in table 1.

Finally, we haven't found sufficient evidence in our experiments to conclude that the Attention module learns long-range dependencies. We speculate that this behavior is dependent on the quality of the graph encoding used and leave this for future work.

3.1.2. MLPs play an important role in later layers

As can be seen from the figure, the error increases heavily in later layers. This sharply contrasts the insight in (Meng et al., 2023) where MLPs are more influential in the early-mid layers while attention influences accuracy more towards the later layers. Based on this behavior, we conclude that the graph structure is processed before node-level data. Deeper analysis is left for future work.

3.1.3. Mid-Life Crisis

We observe that the middle layers have a lower impact on performance. Moreover, all the components are essential to the performance of the model including the Message Passing Module. We illustrate this by first observing that the model converges faster with message passing than without in figure 8. Moreover, the ZINC dataset is a heterogeneous dataset where the edges can be single, double, or triple bonds and this information cannot be passed in graph encoding but can only be passed via message passing. Thus the model underperforms with only transformers, see 2. That brings us to the limitations of graph encodings.

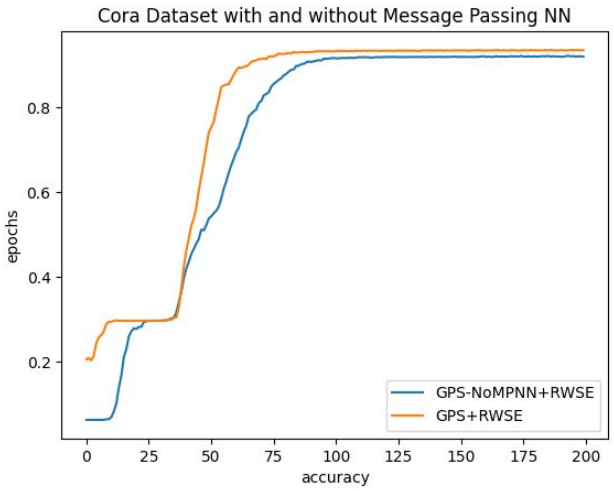


Figure 8. Model converges quicker with message passing on the Cora dataset.

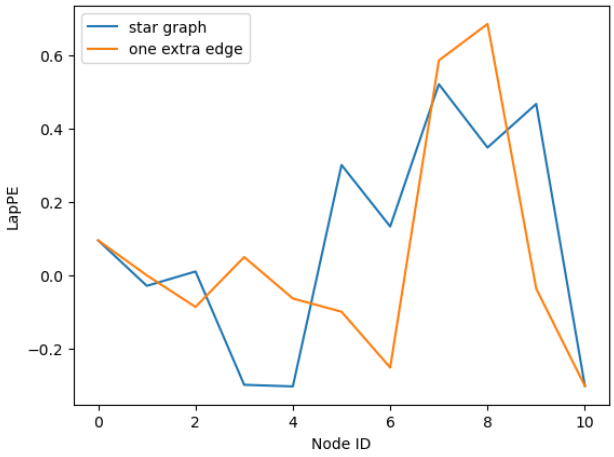


Figure 9. One additional edge significantly affects the Laplacian encoding

Table 2. Demonstrates that Message Passing is crucial in a heterogeneous dataset

ZINC/MSE	TRAIN	TEST	VAL
W/O MESSAGES	0.05702	0.64985	0.19842
MESSAGE PASSING	0.00587	0.2447	0.04305

3.2. Limitations of Graph Encodings

Apart from the above limitation, graph encodings do not possess some of the desirable properties of foundation models.

3.2.1. ROBUSTNESS

Small changes in graph structure can make huge changes in their encodings. This is in contrast to text or image input where changing data does not influence positional encoding. See figure 9 where we show the Laplacian encoding before and after adding an edge. To make matters worse, the eigenvectors are computed using iterative, approximate methods. Computing the first few eigenvalues of the Cora dataset yields values $3.2250\times10^{-8}, 5.4510\times10^{-9}, 4.8691\times10^{-8}$. These are not even in ascending order as they should be.

3.2.2. GENERALIZATION

Encodings are task-dependent and require domain knowledge. See figure 4 for one such illustration where LapPE does better at the EDGES task while RWSE performs better at the TRIANGLES task.

3.2.3. SCALABILITY

Computing eigenvectors is cubic in time complexity and even the approximate methods are expensive. State-of-the-art encodings do not scale to billion-scale graphs.

4. GAT: Performance on natural graphs

In the previous section, we discussed the limitations of GATs compared to GTs. However, in this section, we demonstrate that GAT's low-pass nature plays strongly to natural homogeneous[3] graphs with power scaling. Here, natural graphs refer to naturally occurring graphs such as social networks.

First, observe that labels in a homophilic dataset do not vary a lot across edges and are thus low-frequency signals. Next, since GATs are low-pass filters (as demonstrated in the previous section), they have a strong inductive bias for the task and hence generate strong representations (Park & Kim, 2022).

Moreover, we postulate that natural graph datasets are seldom heterophilic. We illustrate this using the graph coloring problem. Scale-free networks such as Barabási–Albert (Albert & Barabási, 2002) graphs are known to approximate natural graphs closely. In such a graph, a node with degree k occurs in the graph with probability k^{-3}. Hence, a node with degree $\sqrt[3]{n}$ occurs with probability $\frac{1}{n}$, where n is the total number of nodes in the graph. Hence, we expect one node in a graph to have a degree of $\sqrt[3]{n}$. (Chung et al., 2003) proved that the largest eigenvalue grows as $\sqrt{m}$ where m is the maximum degree. Hence, we conclude that the largest

[3]Heterogeneous graphs have an easy time being heterophilic.

Table 3. Demonstrates GAT's competitive performance on the Cora dataset

ACC (%)	RWSE	NO MP	LAPPE	GATV2
TEST	72.8	61.2	81.9	**84.5**
TRAIN	93.2	87.4	**97.7**	80.45

Algorithm 1 ϵ-Greedy Algorithm

Require: A = Available set of encodings
Require: a = current encoding
Require: MAL = moving average loss indexed by encoding
if ShouldRestart **then**
  if with prob. ϵ_t **then**
    a = pick at random from $A - \{a\}$
  else
    $a = \arg\min_a MAL[a]$
  end if
end if
Run current encoding a

eigenvalue grows as $\sqrt[6]{n}$ where n is the size of the graph. Assuming that the largest eigenvalue is a good approximation (Spielman, 2018) of the coloring problem, the number of colors required scales with the number of nodes. Here, the colors are a proxy for labels and it is not natural to have discrete labels scaling with the size of the graph in a node classification problem. For example, all planar graphs [4] can be colored in 4 colors, a constant. Therefore, we conclude that most natural graph tasks are homophilic and thus a good fit for GAT.

Finally, we illustrate the performance of GAT on Cora, a naturally occurring citation graph in 3. GAT not only beats GTs on test accuracies, but it also demonstrates excellent generalization due to its strong inductive bias. On the other hand, GTs overfit the dataset exhibiting significant differences in training and test accuracies.

5. Towards encoding-agnostic Graph Transformers

In the previous sections, we discussed how graph encodings are task-specific. Here, we pursue a more generic encoder by ensembling popular encodings. First, we try out a convex combination of these encodings. However, this approach yields poor performance and hence we seek a different algorithm. To that end, inspired by the $\epsilon-$greedy exploration mechanism from reinforcement learning, we propose a new algorithm in 1. At the start of each episode, we greedily pick the best-performing action with prob. $1 - \epsilon_t$ and randomly pick a different encoding otherwise. We use a moving average loss associated with each encoding as a signal to pick the best-performing action. To demonstrate the efficacy of the algorithm, we use the two popular encoders LapPE and RWSE. The results of our algorithm on the two synthetic datasets EDGES and TRIANGLES are shown in table 4. We pick these datasets because LapPE does well on the EDGES task while RWSE performs well on the TRAINGLES task. Moreover, the LapPE+RWSE encoding in the table refers to a concatenation of those encodings and has accuracies mid-way between the two. On the other hand, our algorithm beats the best-performing encoding on both tasks. Despite the competitive performance, there are a few limitations. First, we would have preferred to learn the correct graph encoding using backpropagation instead of injecting loss into the algorithm. Next, our model is limited to choosing between a predefined set of encodings and thus does not completely remove the need for engineering encodings by hand. Finally, our algorithm introduces even more hyperparameters to handle ϵ and Restart schedule.

[4] Many molecular datasets are planar.

Table 4. Test accuracies for different graph encodings on synthetic datasets

ENCODING	LAPPE	RWSE	LAP+RW	OURS
EDGES ACC (%)	97.75	95.29	97.73	**99.90**
TRI ACC (%)	45.94	76.82	66.56	**83.98**

6. Experimental Setup

See the README file at https://github.com/bvskp-projects/GraphGPS/blob/main/README.md for reproducing our experiments. We provide some details here for completeness.

We ran our experiments on a mix of instgpu-03.cs.wisc.edu, instgpu-04.cs.wisc.edu, Euler[5], and Google Collab. Most experiments were run for 200 epochs. We use the excellent framework introduced by GraphGPS (Rampášek et al., 2023) to develop and run our experiments.

To conduct our ablation study 6, we introduce new config options **pretrained.disable_layer** to specify a 0-indexed layer to disable and **pretrained.disable_node** which must be one of **MP** (Message Passing), **SA** (Self Attention), or **FF** (Feed Forward).

We provide ipython notebooks to illustrate the robustness aspects of LapPE. We use Networkx (NetworkX Developers, 2023) library for generating graphs and (PyTorch Geometric Development Team, 2023) to compute Laplacian graph

[5] https://wacc.wisc.edu/resources/docs/faqs.html

encodings.

Users can now specify **GNE** in **dataset.node_encoder_name** to try out our new encoder. Most of our configs are available in folders **configs/cs762** or **custom_configs**.

7. Related Work

We are inspired by a plethora of amazing work in this exciting and emerging field.

How do Vision Transformers Work? (Park & Kim, 2022) explains why attention works well for vision tasks and provides an explanation for harmony between Convolutions and self-attention. We discuss the efficacy of attention for graph-structured tasks and explore the properties of a recent promising direction that employs transformers. An in-depth analysis is left for future work.

Recipe for a General, Powerful, Scalable Graph Transformer (Rampášek et al., 2023) provides the core framework we base our work on. Their ablation studies are directed toward an architectural search to find the right components for best performance. Instead, our ablation studies are conducted at inference time and provide insights into the roles of different components in a fixed architecture.

A Comprehensive Survey on Graph Neural Networks (Wu et al., 2021) is a literature survey and has a much broader focus. Moreover, the survey does not discuss Graph Transformers because they are relatively recent and this work was published even before.

Attending to Graph Transformers (Müller et al., 2023) is a more recent literature survey that primarily focuses on Graph Transformers. However, we have a narrower but deeper focus and also propose a new generic algorithm to ensemble graph encodings.

Revisiting Graph Neural Networks: All We Have is Low-Pass Filters (NT & Maehara, 2019) proves why GCNs are low-pass filters while we illustrate a similar property for GATs on path graphs.

Locating and Editing Factual Associations in GPT (Meng et al., 2023) explores the importance of MLPs and self attention layerwise. While they do it for sequential tasks, we do it in the graph setting.

Graph Attention Retrospective (Fountoulakis et al., 2023) discusses the expressive power of GATs in the context of robustness. Here, we discuss the strong inductive bias of GATs helpful for solving node classification tasks on natural homogeneous graphs.

Generalization in Graph Neural Networks: Improved PAC-Bayesian Bounds on Graph Diffusion (Ju et al., 2023) provides a generalization bound on message passing networks that scales with the square root of maximum degree. We demonstrate that GATs generalize well on natural graphs despite the large maximum degree owing to their strong inductive bias.

8. Conclusion

In this report, we first explore the properties of the main components of a Graph Transformer. We find that attention layers have an impact early on and feed forward layers later on. We provide some high-level insights into this behavior but defer a deeper analysis to future work. Then, we demonstrate some limitations of graph encodings and identify that they should be a primary focus for Transformers to make a similar impact on graph tasks. Next, we discuss the nature of natural graphs and demonstrate the strong performance of GAT on an example task Cora. Finally, we develop a greedy algorithm that automatically picks the appropriate graph encoding for the task.

9. Course Survey

Table 5. Course Evaluation Completion

NAME	COMPLETED
VENKATA SAIKIRANPATNAIK BALIVADA	YES
APOORVA KUMAR	YES
ASHUTOSH PARIDA	YES

References

Albert, R. and Barabási, A.-L. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1): 47–97, January 2002. ISSN 1539-0756. doi: 10.1103/revmodphys.74.47. URL http://dx.doi.org/10.1103/RevModPhys.74.47.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate, 2016.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L., Goel, K., Goodman, N., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D. E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Koh, P. W., Krass, M., Krishna, R., Kuditipudi, R., Kumar, A.,

Ladhak, F., Lee, M., Lee, T., Leskovec, J., Levent, I., Li, X. L., Li, X., Ma, T., Malik, A., Manning, C. D., Mirchandani, S., Mitchell, E., Munyikwa, Z., Nair, S., Narayan, A., Narayanan, D., Newman, B., Nie, A., Niebles, J. C., Nilforoshan, H., Nyarko, J., Ogut, G., Orr, L., Papadimitriou, I., Park, J. S., Piech, C., Portelance, E., Potts, C., Raghunathan, A., Reich, R., Ren, H., Rong, F., Roohani, Y., Ruiz, C., Ryan, J., Ré, C., Sadigh, D., Sagawa, S., Santhanam, K., Shih, A., Srinivasan, K., Tamkin, A., Taori, R., Thomas, A. W., Tramèr, F., Wang, R. E., Wang, W., Wu, B., Wu, J., Wu, Y., Xie, S. M., Yasunaga, M., You, J., Zaharia, M., Zhang, M., Zhang, T., Zhang, X., Zhang, Y., Zheng, L., Zhou, K., and Liang, P. On the opportunities and risks of foundation models, 2022.

Brody, S., Alon, U., and Yahav, E. How attentive are graph attention networks?, 2022.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, July 2017. ISSN 1558-0792. doi: 10.1109/msp.2017.2693418. URL http://dx.doi.org/10.1109/MSP.2017.2693418.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020.

Chung, F., Lu, L., and Vu, V. Eigenvalues of Random Power law Graphs. *Annals of Combinatorics*, 7 (1):21–33, June 2003. ISSN 0219-3094. doi: 10.1007/s000260300002. URL https://doi.org/10.1007/s000260300002.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

Dwivedi, V. P. and Bresson, X. A generalization of transformer networks to graphs, 2021.

Dwivedi, V. P., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. Graph neural networks with learnable structural and positional representations, 2022.

Fountoulakis, K., Levi, A., Yang, S., Baranwal, A., and Jagannath, A. Graph attention retrospective, 2023.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry, 2017.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, January 2018. ISSN 2374-7951. doi: 10.1021/acscentsci.7b00572. URL http://dx.doi.org/10.1021/acscentsci.7b00572.

Hamilton, W. L. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159, 2020.

Ju, H., Li, D., Sharma, A., and Zhang, H. R. Generalization in graph neural networks: Improved pac-bayesian bounds on graph diffusion, 2023.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks, 2017.

McCallum, A. K., Nigam, K., Rennie, J., and Seymore, K. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval*, 3(2): 127–163, July 2000. ISSN 1573-7659. doi: 10.1023/A:1009953814988. URL https://doi.org/10.1023/A:1009953814988.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in gpt, 2023.

Müller, L., Galkin, M., Morris, C., and Rampášek, L. Attending to graph transformers, 2023.

NetworkX Developers. Networkx documentation, 2023. URL https://networkx.org/documentation/stable/index.html. Accessed: December 19, 2023.

NT, H. and Maehara, T. Revisiting graph neural networks: All we have is low-pass filters, 2019.

Park, N. and Kim, S. How do vision transformers work?, 2022.

PyTorch Geometric Development Team. AddLaplacianEigenvectorPE, 2023. URL https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.transforms.AddLaplacianEigenvectorPE.html. Accessed: December 19, 2023.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. Recipe for a general, powerful, scalable graph transformer, 2023.

Spielman, D. A. Spectral and Algebraic Graph Theory, 2018. URL https://www.cs.yale.edu/homes/spielman/561/lect06-18.pdf.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2023.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks, 2018.

Wang, Y., Li, Z., and Barati Farimani, A. *Graph Neural Networks for Molecules*, pp. 21–66. Springer International Publishing, 2023. ISBN 9783031371967. doi: 10.1007/978-3-031-37196-7_2. URL http://dx.doi.org/10.1007/978-3-031-37196-7_2.

Wang, Z., Tan, Y., and Zhang, M. Graph-based recommendation on social networks. In *Graph-Based Recommendation on Social Networks*, pp. 116–122, 04 2010. doi: 10.1109/APWeb.2010.60.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, January 2021. ISSN 2162-2388. doi: 10.1109/tnnls.2020.2978386. URL http://dx.doi.org/10.1109/TNNLS.2020.2978386.

You, J., Ying, R., and Leskovec, J. Design space for graph neural networks, 2021.