

Graph Neural Networks

Third Tutorial

A review on Graph Transformers

Sahar Almahfouz Nasser

Department of Electrical Engineering

Indian Institute of Technology Bombay

The Tutorial's Agenda:

- Transformer's Input Vs. Graph
- History of Transformers
- Graph Transformers

Why I got interested in topic?



Why I got interested in topic?

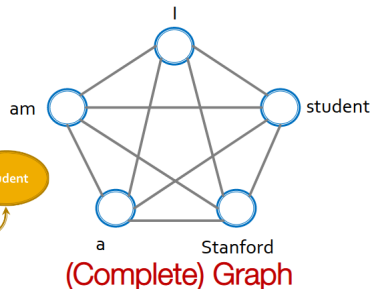
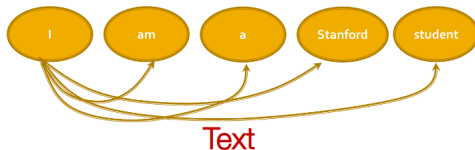


First paper on graph transformer network is Gradient-Based Learning Applied to Document Recognition by Yann Lecun 1998 [LeCun et al., 1998]

GNN Vs. Transformer

Transformer layer can be seen as a special GNN that runs on a fully-connected “word” graph!

Since each word attends to **all the other words**, **the computation graph** of a transformer layer is identical to that of a GNN on the **fully-connected “word” graph**.



- In NLP: sequential structured data
- In Graphs: arbitrary structured data

- Most Graph Transformer Architectures address the problem of over-squashing and limited long-range dependencies in GNNs, but they also significantly increase the complexity from $O(E)$ to $O(N^2)$

- Swin Transformer [Liu et al., 2021]
 - Linear complexity w.r.t number of patches(shifted window)
 - Partitioned window trick for cross window interactions
 - Patch merging to have hierarchical vision transformer
- MetaFormer [Yu et al., 2022] In a transformer architecture what matters is not the token mixing technique
- First paper on graph transformer network is Gradient-Based Learning Applied to Document Recognition by Yann Lecun 1998 [LeCun et al., 1998]

MetaFormer [Yu et al., 2022]

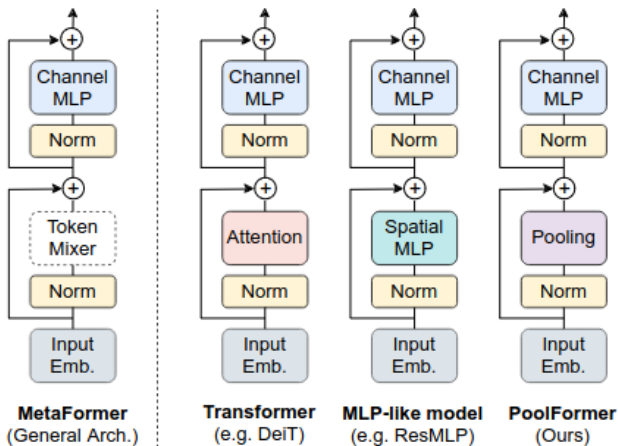


Figure 2: What is truly responsible for the success of the transformers and their variants?

$$X = \text{InputEmb}(I)$$

$$Y = \text{TokenMixer}(\text{Norm}(X)) + X$$

$$Z = \sigma(\text{Norm}(Y)W_1)W_2 + Y$$

where $X \in \mathbb{R}^{N \times C}$, InputEmb : is the patch embedding function, I : is the input image, $W_1 \in \mathbb{R}^{C \times rC}$, $W_2 \in \mathbb{R}^{rC \times C}$, r is the MLP expansion ratio, σ : is ReLU or GELU

Graph Transformer Networks (Neurips-2019)

- GTNs [Yun et al., 2019] are useful when having heterogeneous graphs
- GTNs form new homogeneous graphs from heterogeneous graphs by creating new connections between nodes

GTN's Architecture

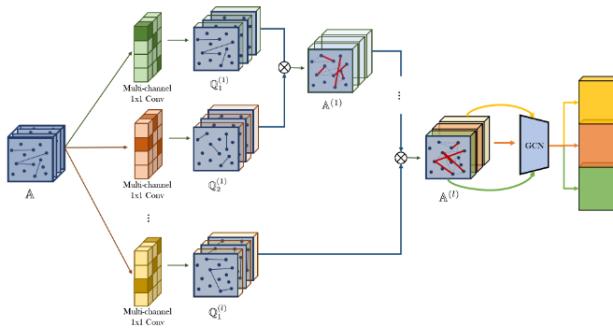


Figure 3: Graph Transformer Networks (GTNs) learn to generate a set of new meta-path adjacency matrices $A^{(l)}$ using GT layers and perform graph convolution as in GCNs on the new graph structures. Multiple node representations from the same GCNs on multiple meta-path graphs are integrated by concatenation and improve the performance of node classification. [Yun et al., 2019]

GTN's Architecture

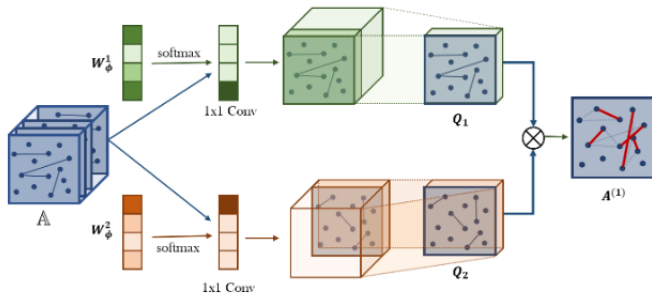


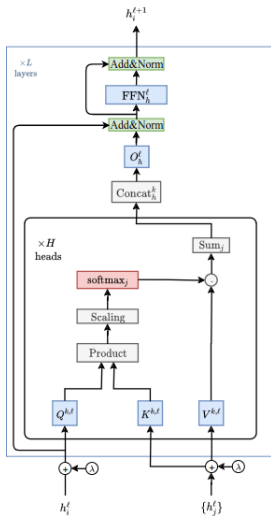
Figure 4: Graph Transformer Layer softly selects adjacency matrices (edge types) from the set of adjacency matrices A of a heterogeneous graph G and learns a new meta-path graph represented by $A^{(1)}$ via the matrix multiplication of two selected adjacency matrices Q_1 and Q_2 . The soft adjacency matrix selection is a weighted sum of candidate adjacency matrices obtained by 1×1 convolution with non-negative weights from $\text{softmax} W_\phi^1$ [Yun et al., 2019]

A Generalization of Transformer Networks to Graphs (2020) [Dwivedi and Bresson, 2020]

The paper addresses the

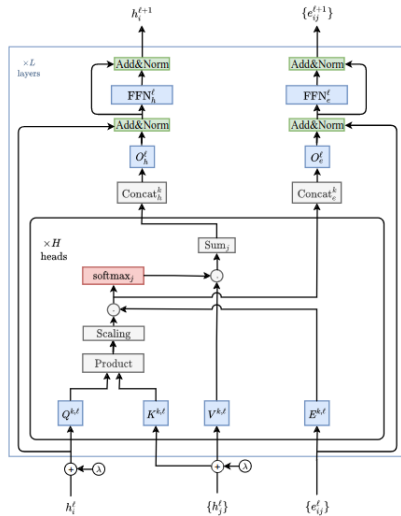
- The differences between **NLP transformer** and **Graph transformer** in terms of attention (dense global attention versus sparse and local attention)
- The positional encoding and edge encoding
- Using an architecture identical to NLP transformer architecture

Architecture



Graph Transformer Layer

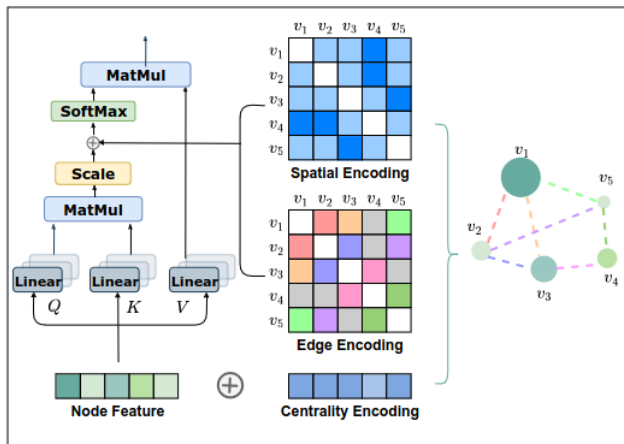
λ Laplacian EigVecs as Positional Encoding



Graph Transformer Layer with edge features

- NLP: dense structure, and scalable to consider full attention
- Graphs: arbitrary connectivity structure, and full attention is not feasible
- [Dwivedi et al., 2020] Proved that Laplacian PE generalize the sinusoidal PE used in NLP transformers
- The architecture extended to have edge representation which can be critical to tasks with rich informations on the edges

Do Transformers Really Perform Badly for Graph Representation? (Neurips, 2021) [Ying et al., 2021]



Encodings

- Structural Encoding: centrality encoding

$$h_i^{(0)} = x_i + Z_{deg^-(v_i)}^- + Z_{deg^+(v_i)}^+$$

where x_i is the node features vector, and Z 's are learnable vectors.

- Spatial Encoding: a function Φ measures the relation (the shortest path) between two nodes i and j

$$A_{ij} = \frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d}} + b_{\Phi(v_i, v_j)}$$

where b is a scalar

- Edge Encoding:

$$A_{ij} = \frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d}} + b_{\Phi(v_i, v_j)} + c_{i,j}$$

Where $c_{i,j} = \frac{1}{N} \sum_1^N x_{en}(W_n^E)^T$

Do Transformers Really Perform Badly for Graph Representation? (Neurips, 2021) [Ying et al., 2021]

In this work:

- They showed that the current GNNs such as GINE, GCN, GraphSAGE,..etc, are special cases of graph transformer
- The usage of the virtual node (similar to the class token in transformer) helps in representing the whole graph and these information transferred to all the nodes
- Result: Graphormer does not suffer from oversmoothing

Recipe for a General, Powerful, Scalable Graph Transformer (Neurips, 2022) [Rampášek et al., 2022]

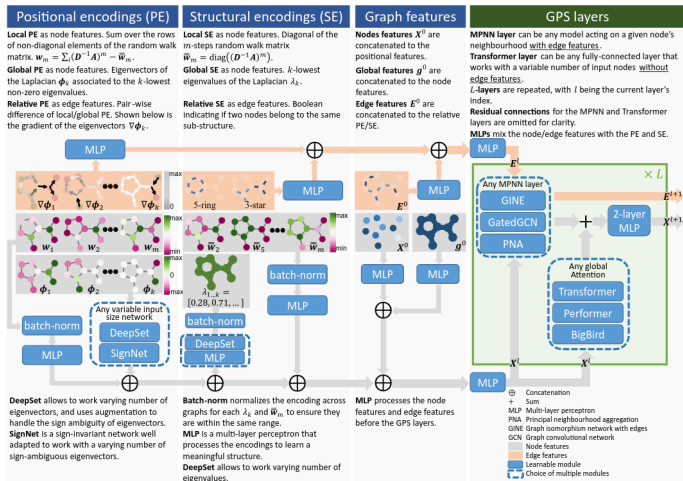


Figure 5: Modular GPS graph Transformer, with examples of PE and SE.

Recipe for a General, Powerful, Scalable Graph Transformer (Neurips, 2022) [Rampášek et al., 2022]

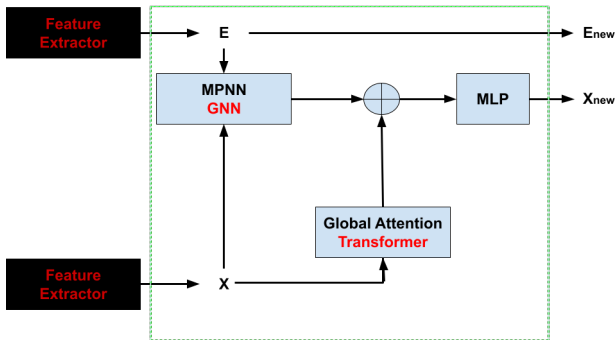


Figure 6: Simplified Architecture

Main Ingredients

- positional/structural encoding
- local message-passing mechanism
- global attention mechanism

Recipe for a General, Powerful, Scalable Graph Transformer (Neurips, 2022) [Rampášek et al., 2022]

Table 1: The proposed categorization of positional encodings (PE) and structural encodings (SE). Some encodings are assigned to multiple categories in order to show their multiple expected roles.

Encoding type	Description	Examples
Local PE <i>node features</i>	Allow a node to know its position and role within a local cluster of nodes. <i>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).</i>	<ul style="list-style-type: none"> 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 <i>node features</i>	Allow a node to know its global position within the graph. <i>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.</i>	<ul style="list-style-type: none"> 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 <i>edge features</i>	Allow two nodes to understand their distances or directional relationships. <i>Edge embedding that is correlated to the distance given by any global or local PE, such as the distance between two words.</i>	<ul style="list-style-type: none"> 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 <i>node features</i>	Allow a node to understand what sub-structures it is a part of. <i>Given an SE of radius m, the more similar the m-hop subgraphs around two nodes are, the closer their local SE will be.</i>	<ul style="list-style-type: none"> 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 <i>graph features</i>	Provide the network with information about the global structure of the graph. <i>The more similar two graphs are, the closer their global SE will be.</i>	<ul style="list-style-type: none"> 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 <i>edge features</i>	Allow two nodes to understand how much their structures differ. <i>Edge embedding that is correlated to the difference between any local SE.</i>	<ul style="list-style-type: none"> Pair-wise distance, encoding, or gradient of any local SE. Boolean indicating if two nodes are in the same sub-structure [5] (similar to the gradient of sub-structure enumeration).

	Ablation	ZINC	PCQM4Mv2 subset	CIFAR10	MalNet -Tiny
		MAE ↓	MAE ↓	Acc. ↑	Acc. ↑
Global Attention	<i>none</i>	0.070	0.1213	69.95	92.23
	Transformer	0.070	0.1159	72.31	93.50
	Performer	0.071	0.1142	70.67	92.64
	BigBird	0.071	0.1237	70.48	92.34
MPNN	<i>none</i>	0.217	0.3294	68.86	73.90
	GINE	0.070	0.1284	71.11	92.27
	GatedGCN	0.086	0.1159	72.31	92.64
	PNA	0.070	0.1409	73.42	91.67

	Ablation	ZINC	PCQM4Mv2 subset	CIFAR10	MalNet -Tiny
		MAE ↓	MAE ↓	Acc. ↑	Acc. ↑
PE/SE	<i>none</i>	0.113	0.1355	71.49	92.64
	RWSE	0.070	0.1159	71.96	92.77
	LapPE	0.116	0.1201	72.31	92.74
	SignNet ^{MLP}	0.090	0.1158	71.74	92.57
	SignNet ^{DeepSets}	0.079	0.1144	72.37	93.13
	PEG ^{LapFig}	0.161	0.1209	72.10	92.27

*Encodings are color-coded by their **positional** or **structural** type.

Figure 7: Ablation Study

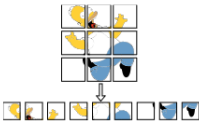
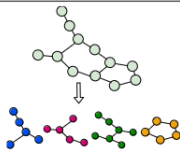
- Updating the edge/node features of the graph by combining the outputs (feature embeddings) of local message passing layer (**Gated GCN/GINE/PNA**) and global attention layer (**transformer**) while using the edges encodings from the message passing layer only.

A Generalization of ViT/MLP-MIXER to Graphs (Meta, Dec/2022) [He et al., 2022]

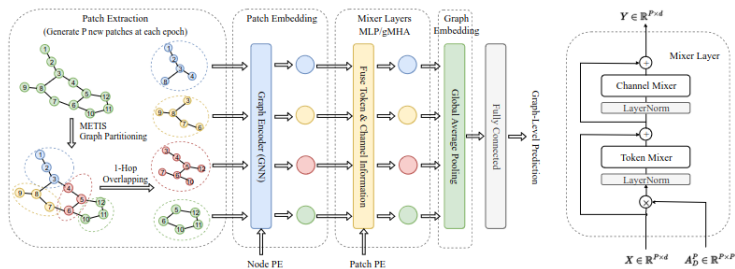
- The architecture achieves: long range dependencies, expressivity, efficiency (speed, low memory, linear complexity)
- Patching, Overlapped Patches, Node/Patch position encoding
- It shows high expressivity in terms of graph isomorphism

Comparison

Table 1: Differences between ViT/MLP-Mixer components for images and graphs.

	Images	Graphs
		
Input	Regular grid Same data resolution (Height, Width)	Irregular domain Variable data structure (# Nodes and # Edges)
Patch Extraction	Via pixel reordering Non-overlapping patches Same patches at each epoch	Via graph clustering algorithm Overlapping patches Different patches at each epoch
Patch Encoder	Same patch resolution (Patch Height, Patch Width) MLP (equivalently CNN)	Variable patch structure (# Nodes and # Edges) GNN (e.g. GCN, GAT, GT)
Positional Information	Implicitly ordered (No need for explicit PE)	No universal ordering Node PE for patch encoder Patch PE for token mixer
ViT / MLP-Mixer	MLP / Channel mixer MHA / Token mixer	MLP / Channel mixer gmHA / Token mixer

Architecture



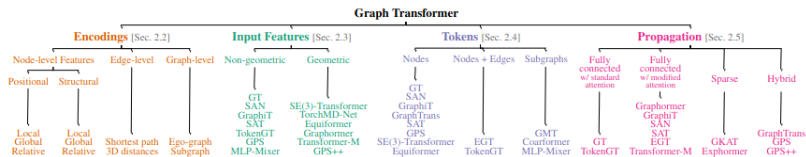
$$U = X + (W_2 \sigma(W_1 \text{LayerNorm}(X))) \in \mathbb{R}^{P \times d},$$

$$Y = U + (W_4 \sigma(W_3 \text{LayerNorm}(U)^T))^T \in \mathbb{R}^{P \times d},$$





Attending to Graph Transformers (Intel, 2023) [Müller et al., 2023]

- Review on transformers' categories

Attending to Graph Transformers



References I

-  Dwivedi, V. P. and Bresson, X. (2020).
A generalization of transformer networks to graphs.
arXiv preprint arXiv:2012.09699.
-  Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. (2020).
Benchmarking graph neural networks.
-  He, X., Hooi, B., Laurent, T., Perold, A., LeCun, Y., and Bresson, X. (2022).
A generalization of vit/mlp-mixer to graphs.
arXiv preprint arXiv:2212.13350.
-  LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998).
Gradient-based learning applied to document recognition.
Proceedings of the IEEE, 86(11):2278–2324.

References II



Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021).

Swin transformer: Hierarchical vision transformer using shifted windows.

In Proceedings of the IEEE/CVF international conference on computer vision, pages 10012–10022.



Müller, L., Galkin, M., Morris, C., and Rampášek, L. (2023).

Attending to graph transformers.

arXiv preprint arXiv:2302.04181.



Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. (2022).

Recipe for a general, powerful, scalable graph transformer.

Advances in Neural Information Processing Systems, 35:14501–14515.

References III



Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., and Liu, T.-Y. (2021).

Do transformers really perform badly for graph representation?

Advances in Neural Information Processing Systems,
34:28877–28888.



Yu, W., Luo, M., Zhou, P., Si, C., Zhou, Y., Wang, X., Feng, J., and Yan, S. (2022).

Metaformer is actually what you need for vision.

In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 10819–10829.



Yun, S., Jeong, M., Kim, R., Kang, J., and Kim, H. J. (2019).
Graph transformer networks.

Advances in neural information processing systems, 32.

Thank you