

A Generalization of Transformer Networks to Graphs^{*}

Machine Learning for Natural Language Processing
Seminar (Spring 2025)
Presented by: Jaykumar Bhagiya



UNIVERSITÄT
DES
SAARLANDES

^{*}Dwivedi, Vijay Prakash, and Xavier Bresson. "A generalization of transformer networks to graphs." arXiv preprint arXiv:2012.09699 (2020).

Graph Transformer Architecture

A Generalization of **Transformer** Networks to **Graphs**

Transformer Neural Networks

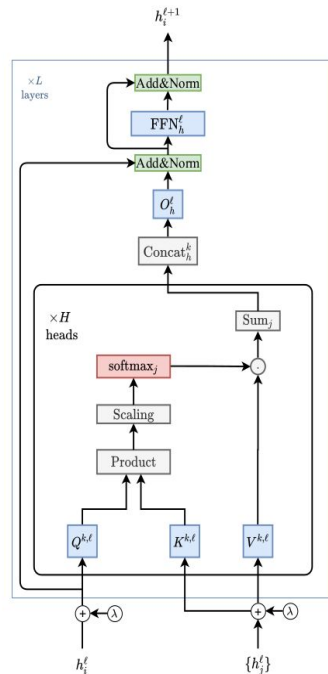
from Natural Language Processing
(sequential structured data)

Graph Neural Networks

from Graph Representation Learning
(arbitrary structured data)

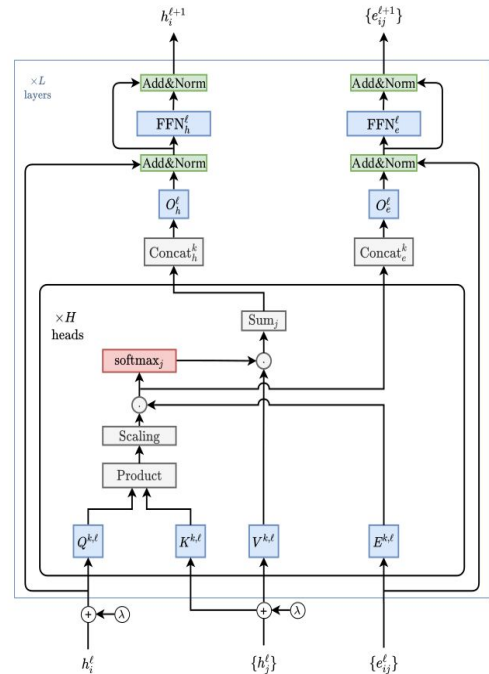
Outline

- Background & Motivation
 - Natural Language Processing (NLP)
 - Graph Representation Learning (GRL)
- Objective
- Key Design Aspects
 - Graph Sparsity
 - Positional Encodings
- Proposed Architecture
- Numerical Experiments
- Conclusion
- Q&A



Graph Transformer Layer

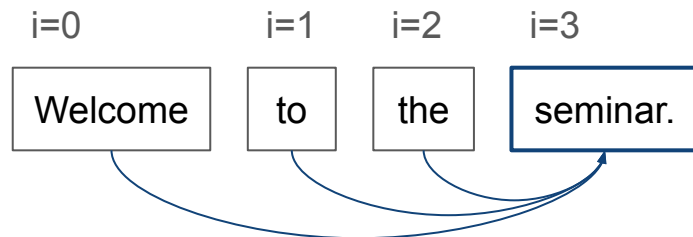
Δ Laplacian EigVecs as Positional Encoding



Graph Transformer Layer with edge features

Connecting the two fields - [I] NLP

- Transformers [1] based networks are widely successful in Natural Language Processing (NLP)
- Use of **multi-head attention mechanism**.
- Transformers are a **State of the art** in language modeling and several NLP applications. [2,3,4]
- A **universal** neural network **architecture** for handling sequential datasets.



Attention to every other word.

- [1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I. (2017). Attention is all you need.
- [2] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.
- [3] Radford, A., Narasimhan, K., Salimans, T. and Sutskever, I., (2018). Improving language understanding by generative pre-training.
- [4] Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. and Agarwal, S., (2020). Language models are few-shot learners.

Connecting the two fields - [II] NLP

$$\hat{h}_i^{\ell+1} = O_h^\ell \parallel_{k=1}^H \left(\sum_{j \in \mathcal{S}} w_{ij}^{k,\ell} V^{k,\ell} h_j^\ell \right),$$

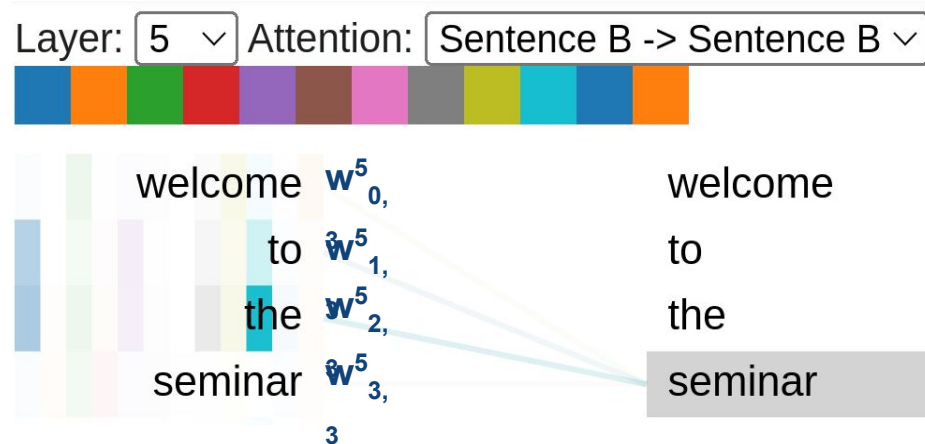
word $\rightarrow \hat{h}_i^{\ell+1}$

other word in a sentence $\rightarrow j \in \mathcal{S}$

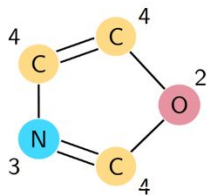
multi-heads $\rightarrow H$

$$\text{where, } w_{ij}^{k,\ell} = \text{softmax}_j \left(\frac{Q^{k,\ell} h_i^\ell \cdot K^{k,\ell} h_j^\ell}{\sqrt{d_k}} \right),$$

attention score $\rightarrow w_{ij}^{k,\ell}$

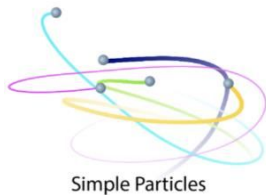


Connecting the two fields - [I] GRL



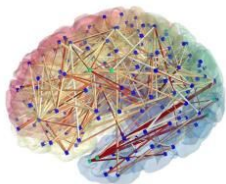
Chemistry [1]

- Learn on molecules and predict chemical properties
- Use in drug repurposing



Physics [2]

- Learn from interactions of particles in systems
- Accelerate physics research



Neuroscience [3]

- Learn functions of brain regions through connectivity
- Accelerate brain-understanding and neuro-disease research

[1] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry.

[2] Cranmer, M. D., Xu, R., Battaglia, P., & Ho, S. (2019). Learning Symbolic Physics with Graph Networks.

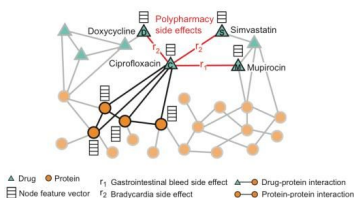
[3] Parisot, S., Ktena, S. I., Ferrante, E., Lee, M., Guerrero, R., Glocker, B., & Rueckert, D. (2018). Disease prediction using graph convolutional networks: Application to Autism Spectrum Disorder and Alzheimer's disease.

Connecting the two fields - [II] GRL



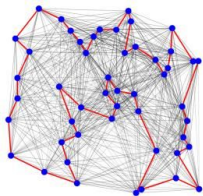
Social networks [1]

- Learn from multifaceted interactions among users
- Use for commercial and social applications



Medicine [2]

- Learn the effects of multiple drugs on body proteins
- Use for efficient multi-drug medical therapies



Combinatorial Optimization [3]

- Exploit the fact that most CO problems are rep. as graphs
- Develop better approximated solutions for NP-hard problems

[11] Monti, F., Bronstein, M., & Bresson, X. (2017). Geometric matrix completion with recurrent multi-graph neural networks.

[2] Huang, K., Fu, T., Xiao, C., Glass, L., Zitnik, M., Xia, C. & Sun, J. (2020). DeepPurpose: a Deep Learning Library for Drug-Target Interaction Prediction and Applications to Repurposing and Screening

[3] Joshi, C. K., Cappart, Q., Rousseau, L. M., Laurent, T., & Bresson, X. (2020). Learning TSP Requires Rethinking Generalization.

Connecting the two fields - [III] GRL

- Graph representation learning: **Graph Neural Networks**
- A NN model which takes in a **graph** with **node** (and **edge**) features and builds **abstract representations** of nodes (and edges)

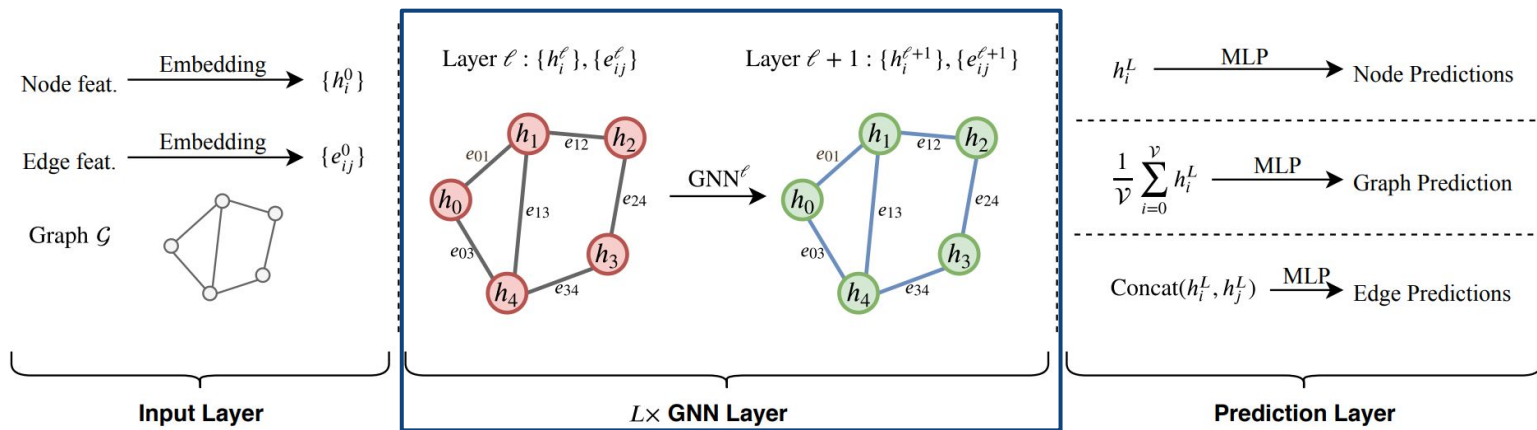


Figure from [1]: End-to-end schematic diagram of a L layer GNN model, in which every layer builds abstract representations of nodes and edges using the message-passing framework, i.e. nodes exchange messages with their first-hop neighbors at every layer

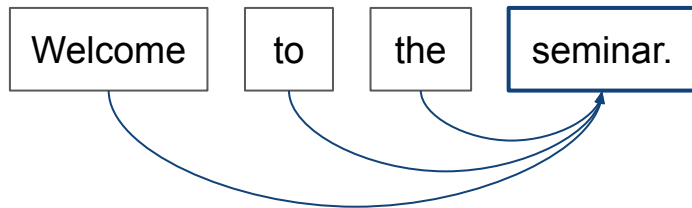
Objective

- The goal of the paper is to generalize Transformers to arbitrary graphs, i.e., design a **Graph Transformer**.
- Extend key principles behind Transformer's success to graph structured datasets.

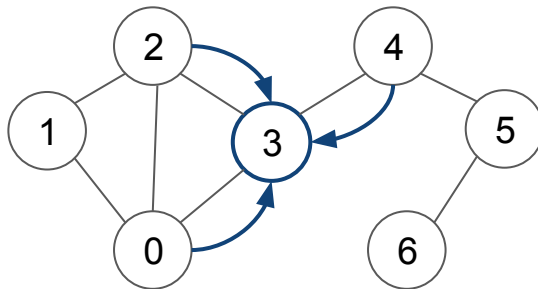
- **Obvious Idea:** Use attention to local neighbors in a graph instead of attending to all nodes, like GATs [1].

But, is that enough?

Attention to every other word.



Attention only to local neighbors



Related Work

- Recent works for Graph Transformer in literature: [1, 2, 3, 4]
- Some attempts to generalize transformer's attention mechanism to graphs, with several focused on specialized cases such as:
 - Full-graph attention
 - Temporal networks
 - Heterogeneous graphs , etc
- **Open questions** on:
 - Local v/s global attention
 - Sparse structure v/s full graph
 - Positional encoding candidates
 - Scalability and pre-training

[1] Li, Y.; Liang, X.; Hu, Z.; Chen, Y.; and Xing, E. P. (2019). Graph Transformer

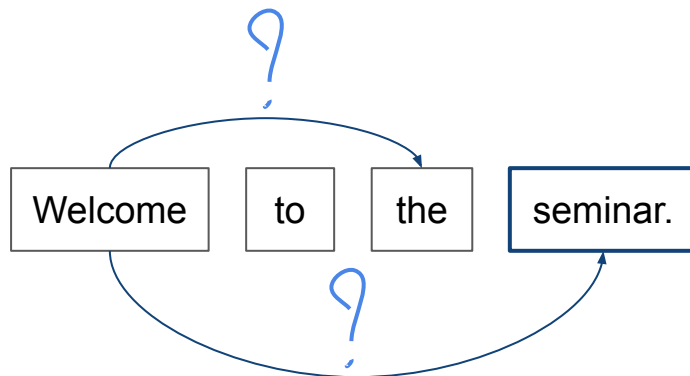
[2] Nguyen, D. Q.; Nguyen, T. D.; and Phung, D. (2019). Universal Self-Attention Network for Graph Classification.

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

[4] Zhang, J.; Zhang, H.; Sun, L.; and Xia, C. 2020. Graph-Bert: Only Attention is Needed for Learning Graph Representations.

Key Design Aspects - [I] Graph Sparsity

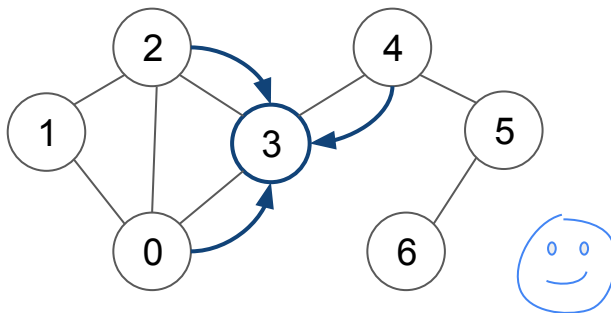
- NLP Transformers consider fully connected graph [1] and this choice can be justified for two reasons:
 - Difficult to find meaningful sparse connections, i.e. **absence of sparse structure**.
 - NLP Transformer sentences often contain less than tens or hundreds of words, hence **scalable to consider full attention**.



Sequence of words in NLP

Key Design Aspects - [II] Graph Sparsity

- But, graphs have arbitrary connectivity structure,
- And **full attention is not feasible** (millions/billions of nodes)
- GNNs are state of the art on several application domains since they **consider sparse structure** into account while learning feature representations.
- Sparsity is a **good inductive bias** for generalization.



Graph with arbitrary structure

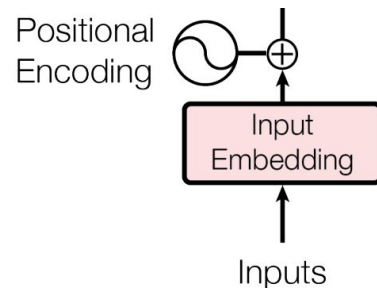
Positional Encodings (PE) - [I] From Transformer

- NLP Transformers are, mostly, supplied with a positional encoding for each word.
- Adds unique vectors to word embeddings based on their position in a sentence.
- Uses sine and cosine functions of varying frequencies to create continuous binary encodings.

<Sequence>
Welcome to the seminar

v/s

<Multi-set>
{Welcome, to, the, seminar}



PE in Original
Transformer

Positional Encodings (PE) - [II] From Transformer

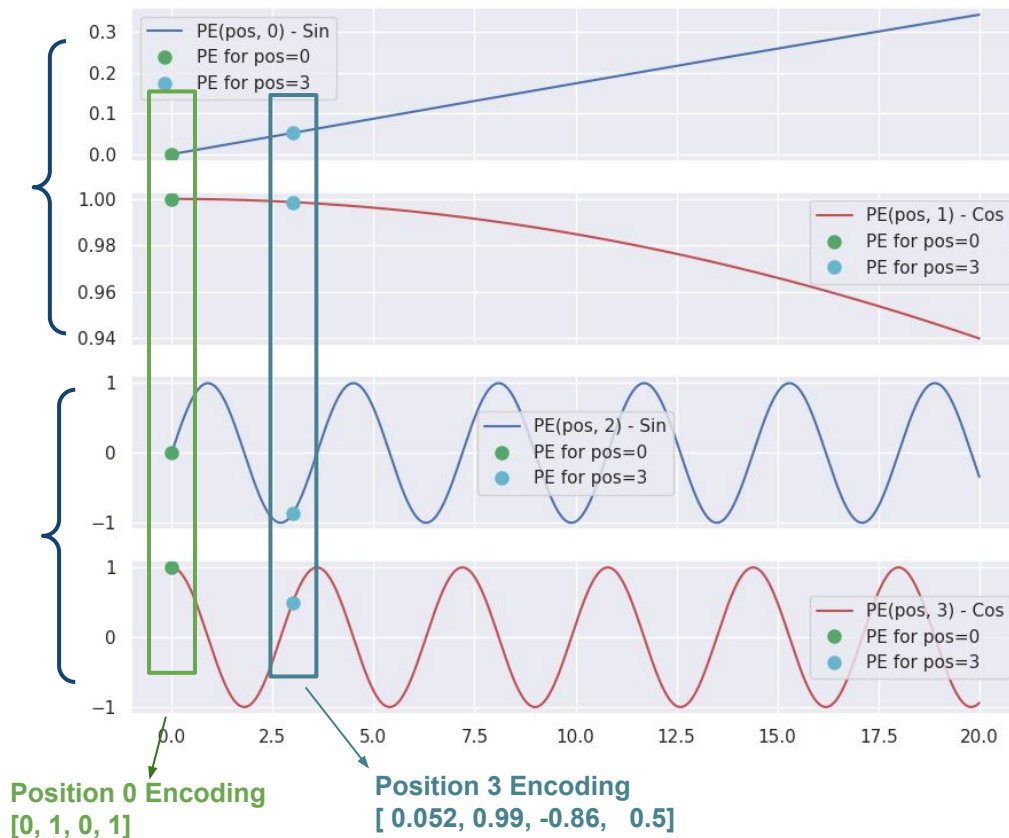
$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$

		pos	PE(pos, 2i) or PE(pos, 2i+1) with $d_{\text{model}} = 4$				
Welcome	→	0	→	PE(0,0) = 0	PE(0,1) = 1	PE(0,2) = 0	PE(0,3) = 1
to	→	1	→	PE(1,0) = 0.017	PE(1,1) = 0.99	PE(1,2)= 0.98	PE(1, 3)= -0.17
the	→	2	→	PE(2,0) = 0.034	PE(2,1) = 0.99	PE(2,2) = -0.34	PE(2,3) = -0.94
seminar	→	3	→	PE(3,0) = 0.052	PE(3,1) = 0.99	PE(3,2) = -0.86	PE(3,3) = 0.5

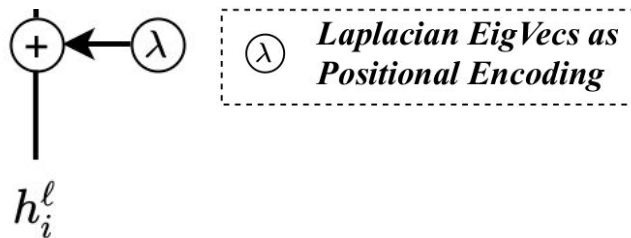
Positional Encodings (PE) - [III] From Transformer

- **Low Frequencies:**
 - Capture long-range relationships and encode global position information.
- **High Frequencies:**
 - Capture short-range relationships and encode local position information.



Key Design Aspects - [I] Positional Encodings (PE)

- It's natural to use a mechanism to encode node positions in Graph Transformers as explored in recent research in GNNs. [1,2,3] (for Position integration)
- PEs in graph is challenging and hard to get canonical node position information.



PE in Proposed Graph
Transformer

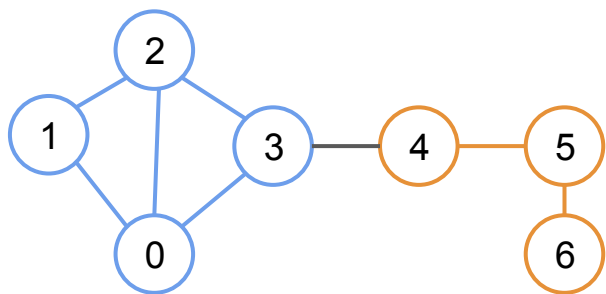
- **Laplacian Eigenvectors** [4] corresponds to a node, as the node positional encoding – namely **Laplacian PE (λ)** [3].

[1] Srinivasan, B. and Ribeiro, B., (2019). On the equivalence between positional node embeddings and structural graph representations.
[2] You, J., Ying, R. and Leskovec, J., (2019). Position-aware graph neural networks.
[3] Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. (2020). Benchmarking graph neural networks.
[4] Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation.

Key Design Aspects - [II] Positional Encodings (PE)

What are Laplacian Eigenvectors?

- The Laplacian Matrix (L) encodes the graph's structure: $L = D - A$, where D is the degree matrix and A is the adjacency matrix.
- It is positive semi-definite, meaning eigenvalues (λ) are non-negative ($\lambda \geq 0$).
- Eigenvectors of L solves $Lu = \lambda u$, where u is an eigenvector and λ is the eigenvalue.



$$D = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A = \begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

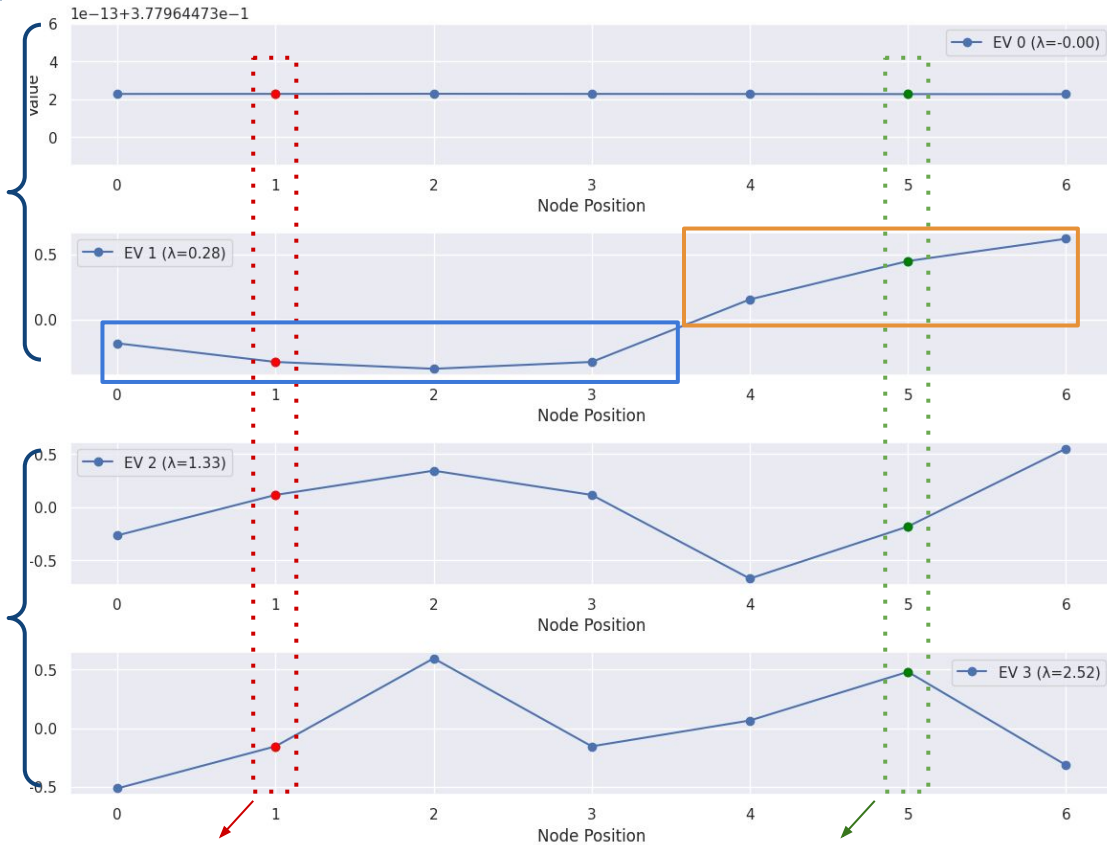
Key Design Aspects - [III] Positional Encodings (PE)

- **Small Eigenvalues:**

- Represent smooth, global structures of the graph.
- Capture broad relationships between distant nodes.

- **Large Eigenvalues:**

- Represent rapid variations, capturing local, fine-grained details of the graph.

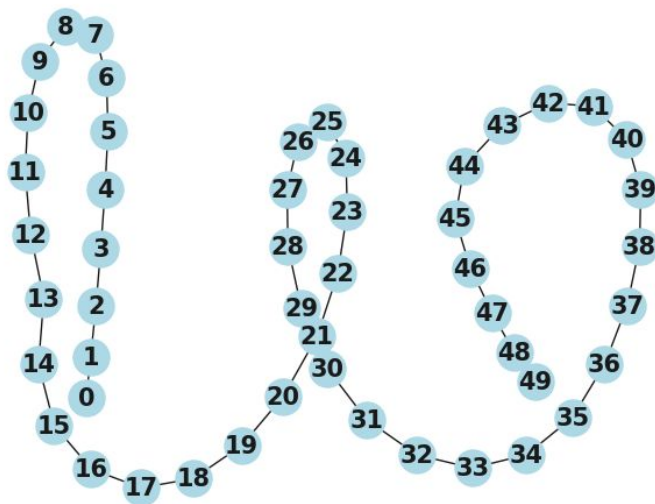


Position Encoding for Node 1

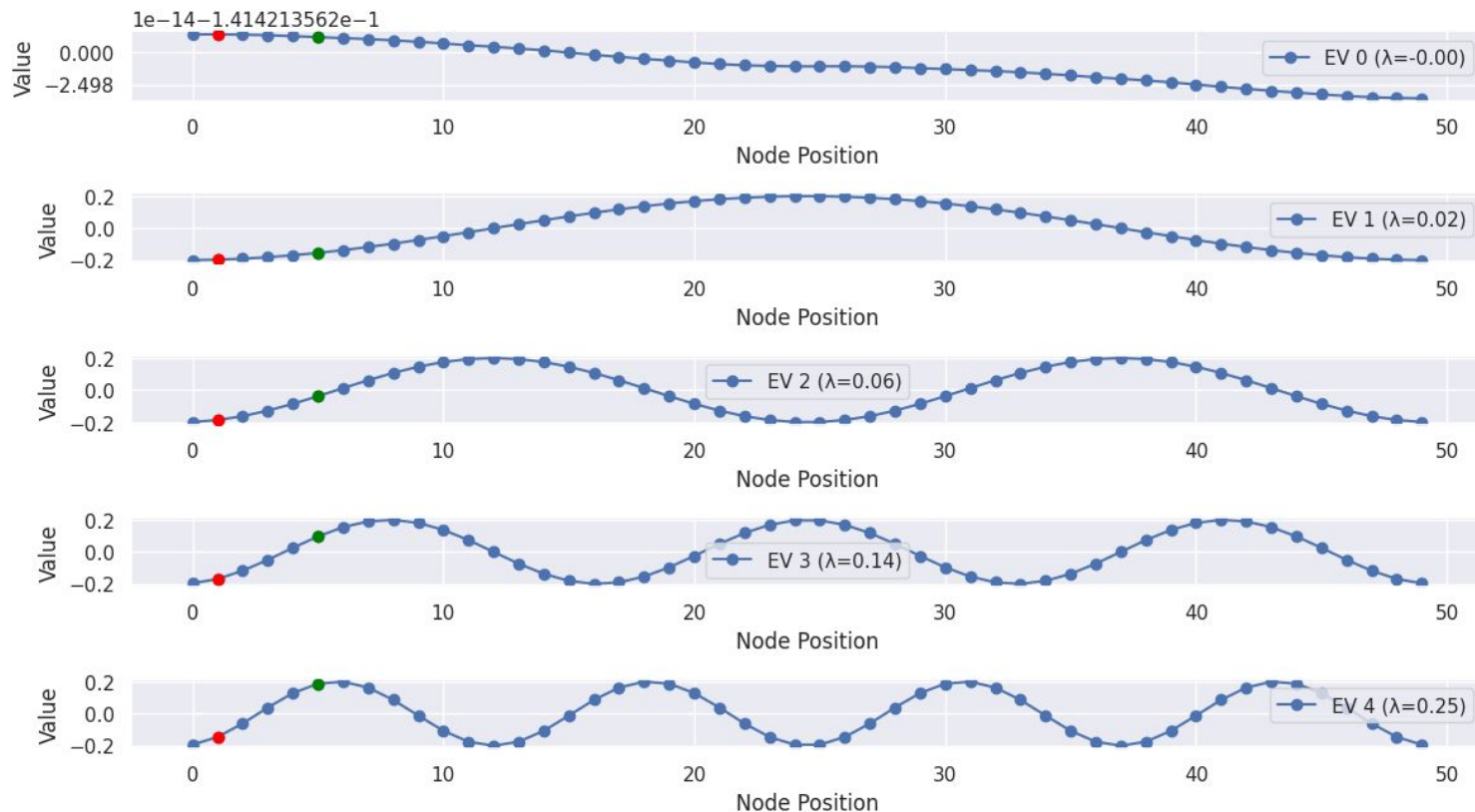
Position Encoding for Node 5

Key Design Aspects - [IV] Positional Encodings (PE)

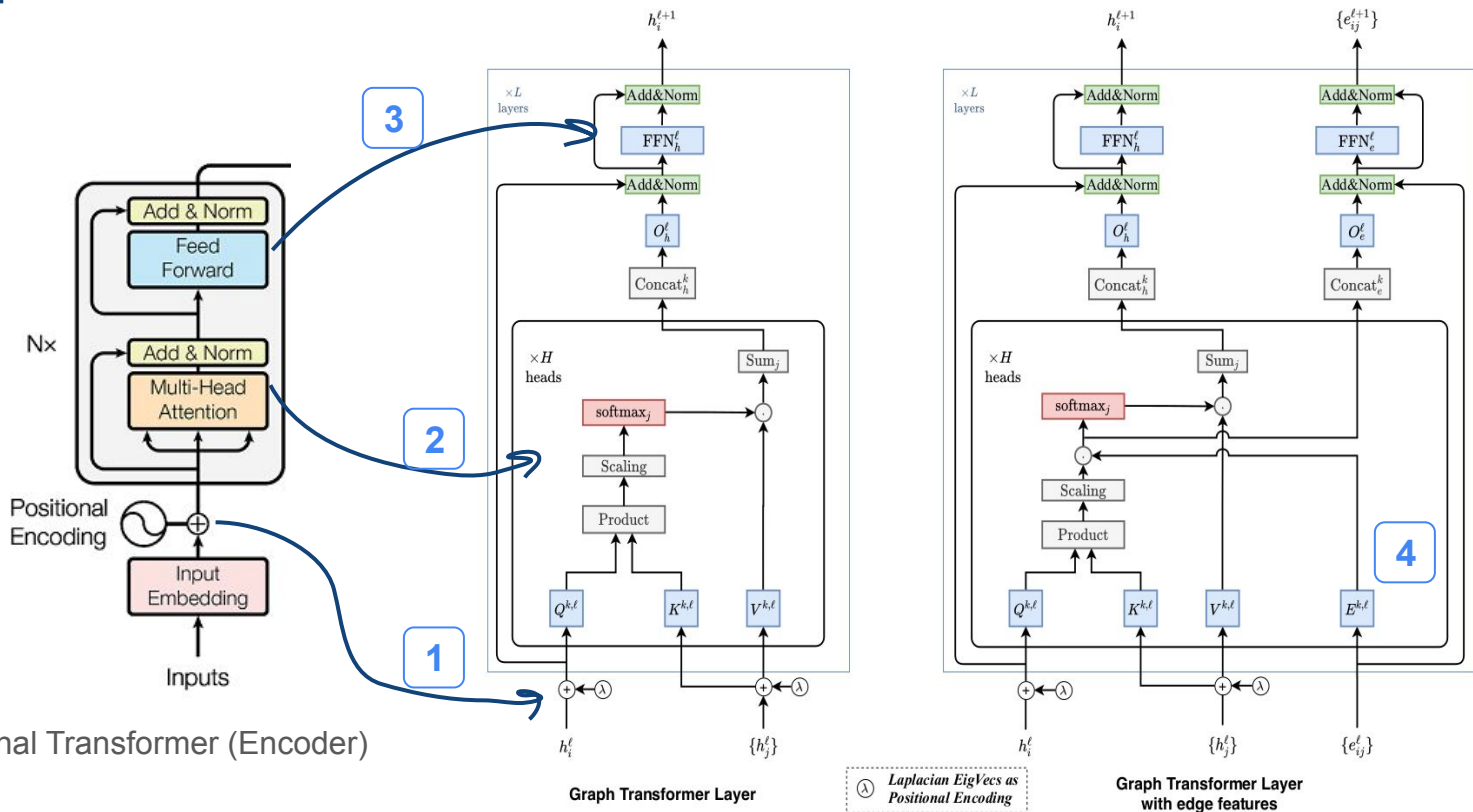
- Representing a sentence as a graph allows its Laplacian eigenvectors to generate positional encodings similar to sine and cosine functions in NLP Transformers.
- This demonstrates how graph-based positional encoding unifies with NLP methods



Key Design Aspects - [V] Positional Encodings (PE)



Proposed Architecture for Generalization



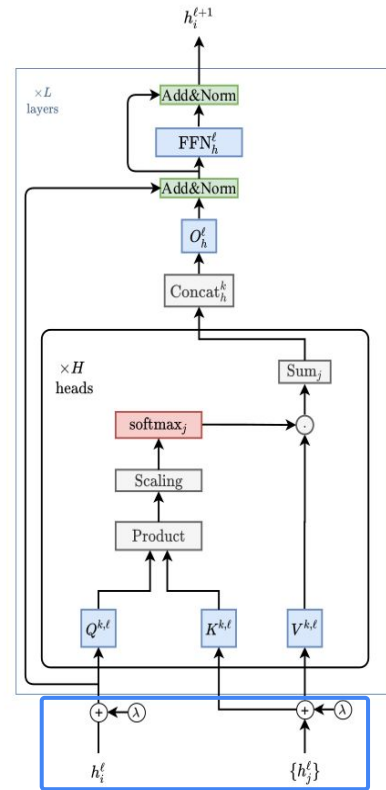
Proposed Architecture for Generalization

Compared to the original transformer, the **highlights** of the proposed architecture are:

(1) Positional encoding is represented by **Laplacian PE – k smallest non-trivial eigenvectors of a node.**

$$\hat{h}_i^0 = A^0 \alpha_i + a^0 \quad ; \quad e_{ij}^0 = B^0 \beta_{ij} + b^0,$$

$$\lambda_i^0 = C^0 \lambda_i + c^0 \quad ; \quad h_i^0 = \underbrace{\hat{h}_i^0 + \lambda_i^0}_{\text{Embed the pre-computed PE}}$$



Graph Transformer Layer

Proposed Architecture for Generalization

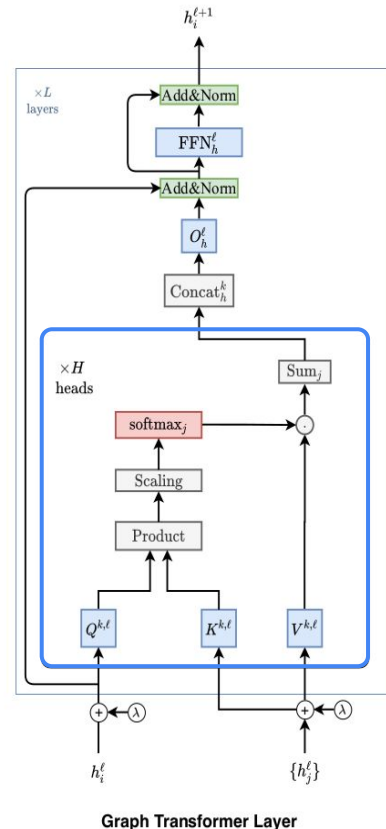
Compared to the original transformer, the **highlights** of the proposed architecture are:

(2) The *attention* mechanism is a function of **neighborhood connectivity** for each node.

$$\text{node } \hat{h}_i^{\ell+1} = O_h^\ell \parallel \left(\sum_{j \in \mathcal{N}_i} w_{ij}^{k,\ell} V^{k,\ell} h_j^\ell \right),$$

← node
← node neighborhood

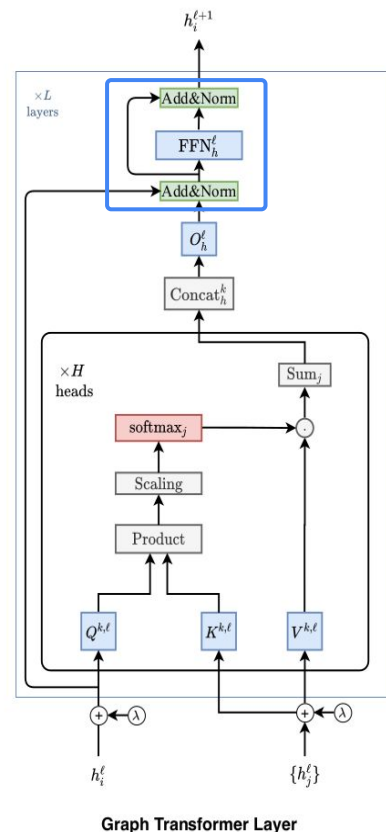
$$\text{where, } w_{ij}^{k,\ell} = \text{softmax}_j \left(\frac{Q^{k,\ell} h_i^\ell \cdot K^{k,\ell} h_j^\ell}{\sqrt{d_k}} \right),$$



Proposed Architecture for Generalization

Compared to the original transformer, the **highlights** of the proposed architecture are:

- (3) The layer normalization is replaced by a **batch normalization layer**.



Proposed Architecture for Generalization

Compared to the original transformer, the **highlights** of the proposed architecture are:

(4) The architecture is extended to have **edge representation**, which can be critical to tasks with rich information on the edges.

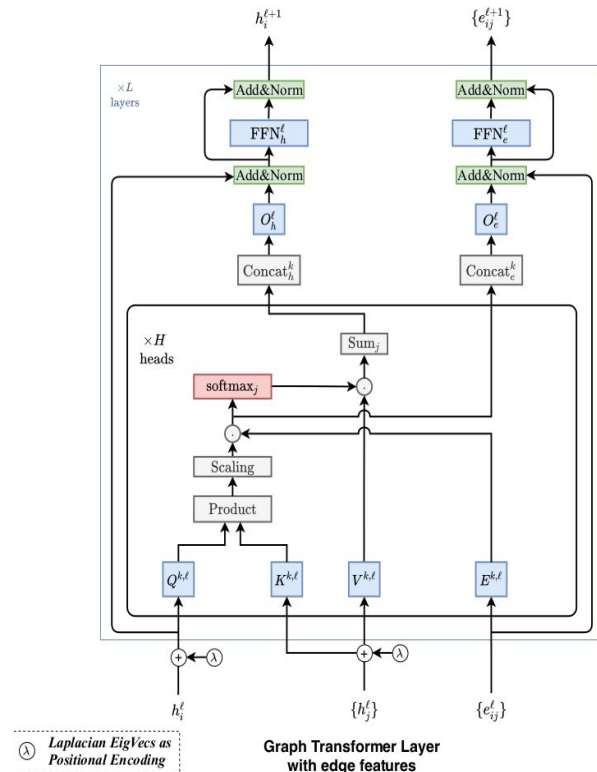
$$\hat{h}_i^{\ell+1} = O_h^\ell \parallel \left(\sum_{k=1}^H \sum_{j \in \mathcal{N}_i} w_{ij}^{k,\ell} V^{k,\ell} h_j^\ell \right),$$

$$\hat{e}_{ij}^{\ell+1} = O_e^\ell \parallel \left(\hat{w}_{ij}^{k,\ell} \right), \text{ where,}$$

$$w_{ij}^{k,\ell} = \text{softmax}_j(\hat{w}_{ij}^{k,\ell}),$$

$$\hat{w}_{ij}^{k,\ell} = \left(\frac{Q^{k,\ell} h_i^\ell \cdot K^{k,\ell} h_j^\ell}{\sqrt{d_k}} \right) \cdot E^{k,\ell} e_{ij}^\ell,$$

Available edge information



Numerical Experiments

- The proposed **Graph Transformer (GT)** is **evaluated** on **ZINC**, **CLUSTER**, **PATTERN** datasets from the Benchmarking GNNs [1].

Dataset	#Graphs	#Classes	Avg. Nodes	Avg. Edges	Node feat. (dim)	Edge feat. (dim)
ZINC	12000	–	23.16	49.83	Atom Type (28)	Bond Type (4)
PATTERN	14000	2	117.47	4749.15	Node Attr (3)	N.A.
CLUSTER	12000	6	117.20	4301.72	Node Attr (7)	N.A.

Table: Summary statistics of all datasets. Numbers in parentheses of Node features and Edge features are the dimensions.

Numerical Experiments

Results

- GT outperforms vanilla GCNs [1], and GATs.
- GT reaches competitive performance w/ GatedGCN [3] which report the best performance on these datasets in [1].
- This work was only targeted for a straightforward generalization of transformers, and not SOTA performance.

Model	ZINC	CLUSTER	PATTERN
GNN BASELINE SCORES from (Dwivedi et al. 2020)			
GCN	0.367±0.011	68.498±0.976	71.892±0.334
GAT	0.384±0.007	70.587±0.447	78.271±0.186
GatedGCN	0.214±0.013	76.082±0.196	86.508±0.085
OUR RESULTS			
GT (Ours)	0.226±0.014	73.169±0.622	84.808±0.068

Table: Performance Measure for ZINC is MAE (lower is better), and for CLUSTER, PATTERN is Accuracy (higher is better).

[1] Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. (2020). Benchmarking graph neural networks.

[2] Kipf, T. N., and Welling, M. (2016). Semi-supervised classification with graph convolutional networks.

[3] Bresson, X. and Laurent, T. (2017). Residual gated graph convnets.

Addressing questions

- Recall **Open questions** on:
 - Local v/s global attention
 - Sparse structure v/s full graph
 - Positional encoding candidates

Dataset	LapPE	L	#Param	Sparse Graph				Full Graph			
				Test Perf.±s.d.	Train Perf.±s.d.	#Epoch	Epoch/Total	Test Perf.±s.d.	Train Perf.±s.d.	#Epoch	Epoch/Total
Batch Norm: True; Layer Norm: False											
ZINC	x	10	588353	0.264±0.008	0.048±0.006	321.50	28.01s/2.52hr	0.724±0.013	0.518±0.013	192.25	50.27s/2.72hr
	$\checkmark$	10	588929	0.226±0.014	0.059±0.011	287.50	27.78s/2.25hr	0.598±0.049	0.339±0.123	273.50	45.26s/3.50hr
CLUSTER	x	10	523146	72.139±0.405	85.857±0.555	121.75	200.85s/6.88hr	21.092±0.134	21.071±0.037	100.25	595.24s/17.10hr
	$\checkmark$	10	524026	73.169±0.622	86.585±0.905	126.50	201.06s/7.20hr	27.121±8.471	27.192±8.485	133.75	552.06s/20.72hr
PATTERN	x	10	522742	83.949±0.303	83.864±0.489	236.50	299.54s/19.71hr	50.889±0.069	50.873±0.039	104.50	621.33s/17.53hr
	$\checkmark$	10	522982	84.808±0.068	86.559±0.116	145.25	309.95s/12.67hr	54.941±3.739	54.915±3.769	117.75	683.53s/22.77hr

- Attention to local neighbors is critical in generalization.

Addressing questions

- Recall **Open questions** on:
 - Local v/s global attention
 - Sparse structure v/s full graph
 - **Positional encoding candidates**
- Ablation study between Weisfeiler-Lehman positional encoding (WL-PE) [1] and Laplacian PE
- Laplacian PE is well placed PE candidate

Dataset	PE	#Param	Sparse Graph			Epoch/Total
			Test Perf.±s.d.	Train Perf.±s.d.	#Epoch	
Batch Norm: True; Layer Norm: False; $L = 10$						
ZINC	X	588353	0.264±0.008	0.048±0.006	321.50	28.01s/2.52hr
	L	588929	0.226±0.014	0.059±0.011	287.50	27.78s/2.25hr
	W	590721	0.267±0.012	0.059±0.010	263.25	27.04s/2.00hr
CLUSTER	X	523146	72.139±0.405	85.857±0.555	121.75	200.85s/6.88hr
	L	524026	73.169±0.622	86.585±0.905	126.50	201.06s/7.20hr
	W	531146	70.790±0.537	86.829±0.745	119.00	196.41s/6.69hr
PATTERN	X	522742	83.949±0.303	83.864±0.489	236.50	299.54s/19.71hr
	L	522982	84.808±0.068	86.559±0.116	145.25	309.95s/12.67hr
	W	530742	75.489±0.216	97.028±0.104	109.25	310.11s/9.73hr

Table 3: Analysis of GraphTransformer (GT) using different PE schemes. Notations **X**: No PE; **L**: LapPE (ours); **W**: WL-PE (Zhang et al. 2020). **Bold**: the best performing model for each dataset.

Limitations & Recent works

- **Challenge of Laplacian Eigenvectors:**

- Computing eigenvectors for positional encodings is $O(n^3)$, infeasible for graphs with millions of nodes.
- Large graphs demand excessive memory, limiting scalability on standard hardware.

- **Recent Advances:**

- **NAGphormer** [1]: Tokenizes multi-hop neighborhoods into sequences, enabling mini-batch training with $O(n \cdot k)$ complexity.
- **VCR-graphormer** [2]: Adds virtual connections to capture local-to-global dependencies, reducing complexity to $O(m + k \log k)$

- **Scalable Transformer Solutions:**

- Both process localized contexts via attention on node sequences, surpassing eigenvector-based methods while preserving key graph information.

[1] Chen, J., Gao, K., Li, G., & He, K. (2022). NAGphormer: A tokenized graph transformer for node classification in large graphs.

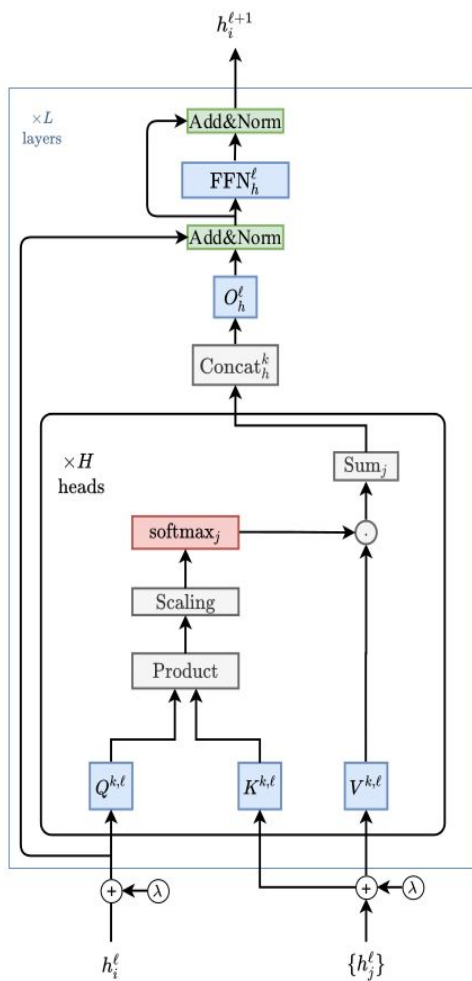
[2] Fu, D., Hua, Z., Xie, Y., Fang, J., Zhang, S., Sancak, K., ... & Long, B. (2024). VCR-graphormer: A mini-batch graph transformer via virtual connections.

Conclusion

- **Effective Generalization:**
 - Adapts transformer networks to arbitrary graphs, outperforming baseline GNNs (GCN, GAT) across benchmarks.
- **Key Enhancements:**
 - Laplacian eigenvectors as positional encodings and batch normalization boost performance universally.
- **Edge Feature Promise:**
 - Extension with edge features nears top GNN performance (e.g., GatedGCN on ZINC), ideal for domain-rich graphs.
- **Competitive Baseline:**
 - Simple, generic architecture establishes a strong foundation for future graph attention models.
- **Future Directions:**
 - Aims to enhance efficiency on large graphs and applicability to heterogeneous domains.

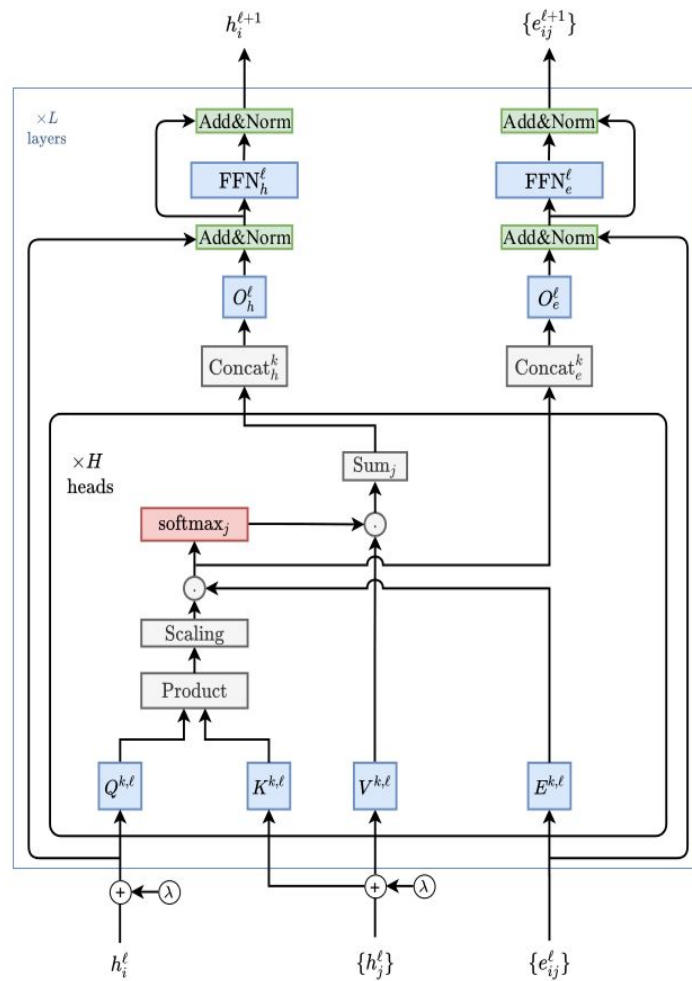
Thank You!

Questions?



Graph Transformer Layer

λ Laplacian EigVecs as
Positional Encoding



Graph Transformer Layer
with edge features