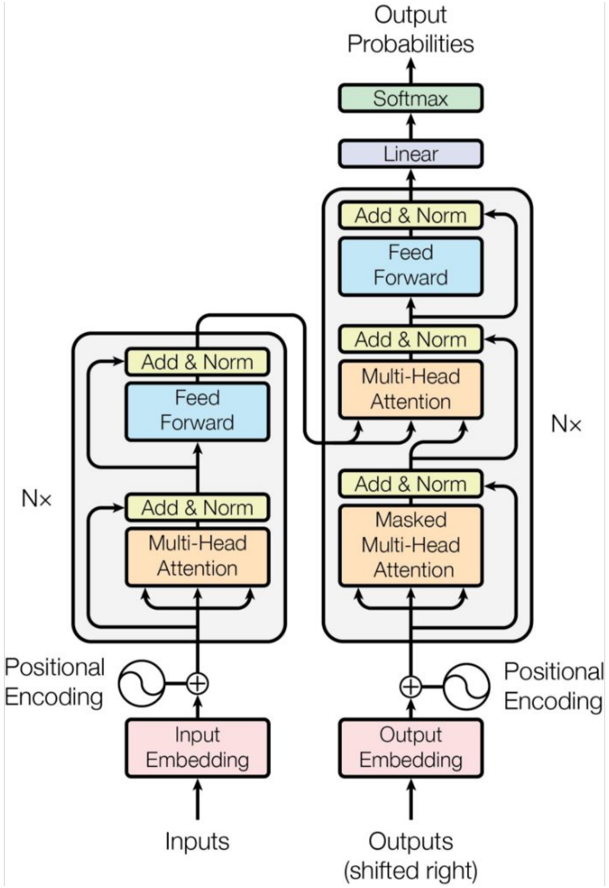
A Generalization of Transformer Networks to Graphs

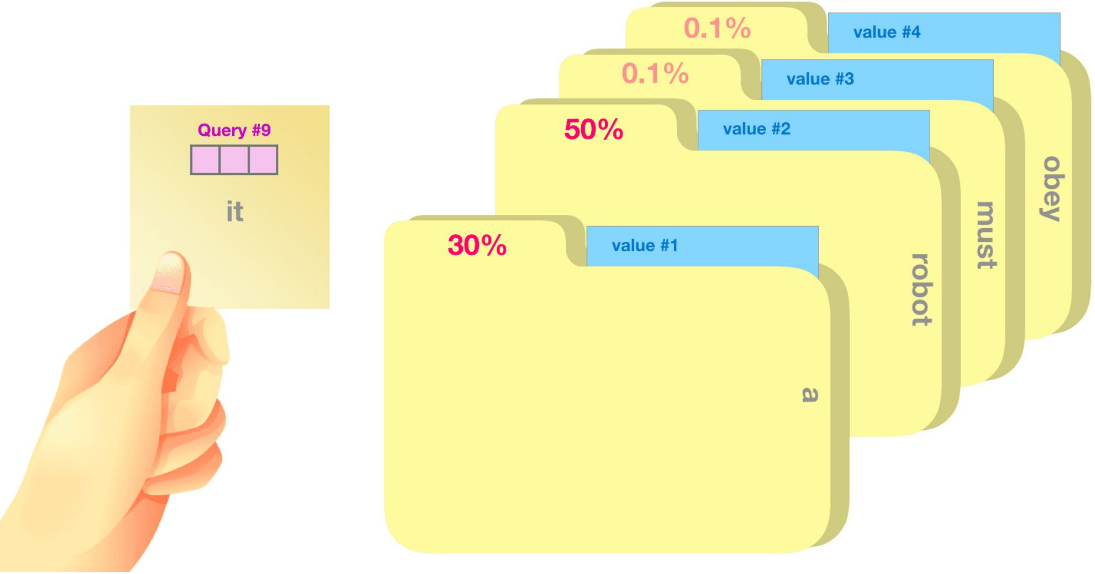
Vijay Prakash Dwivedi, Xavier Bresson

Paper report by

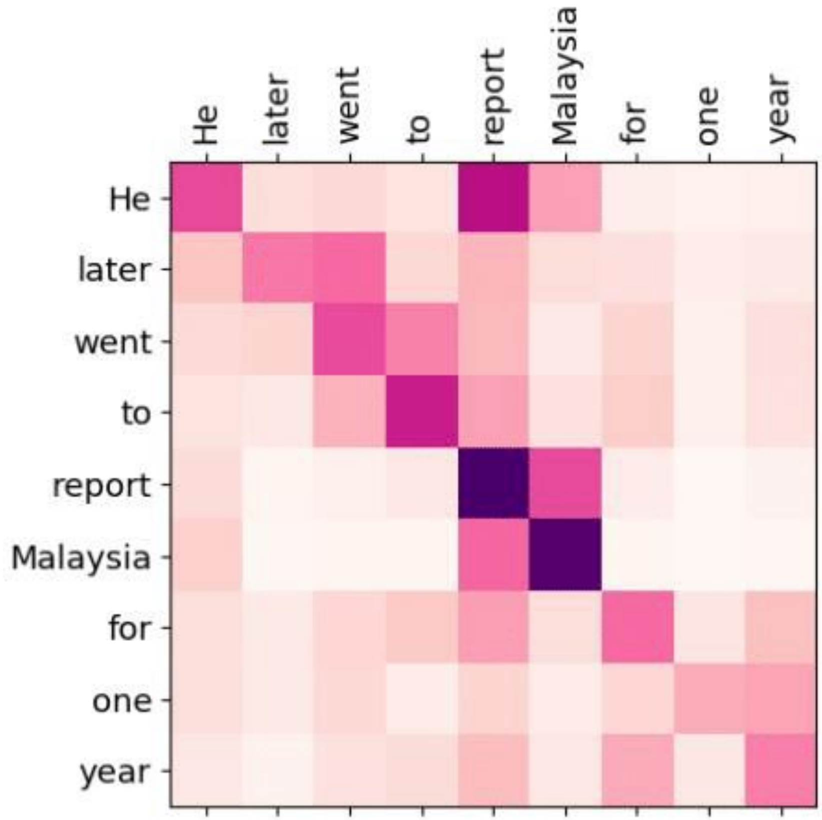
Aleksandr Kariakin, Lev Leontev, Nikita Ivlev

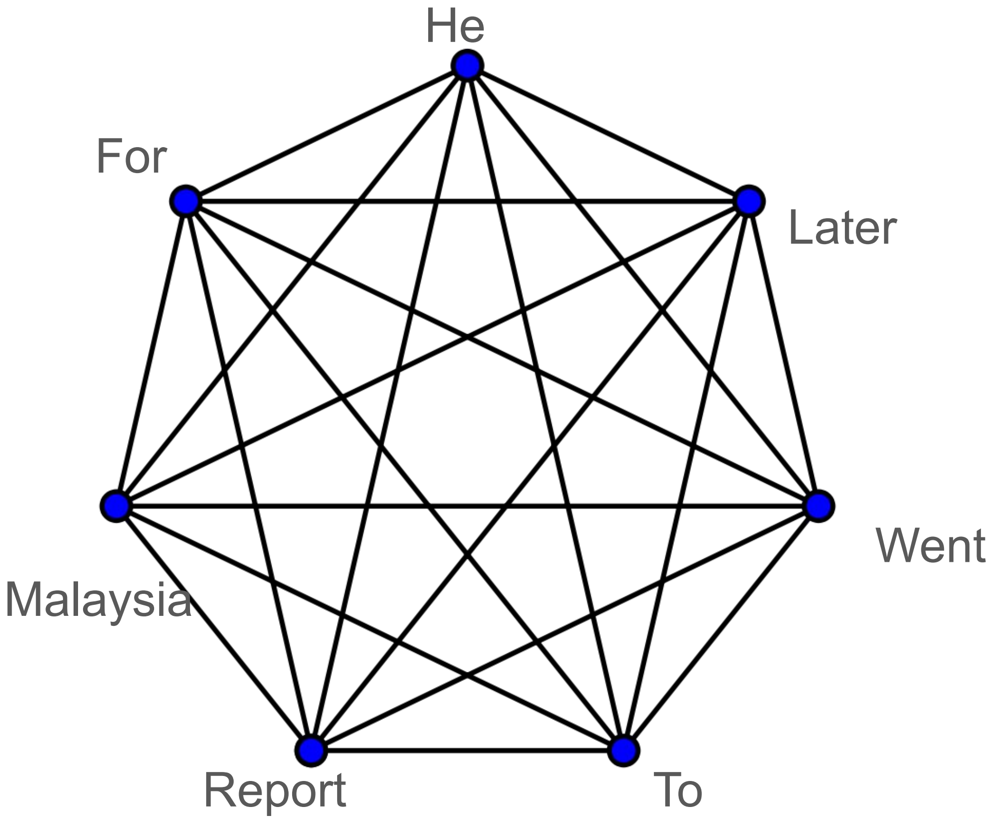
Recall: Transformer, Attention



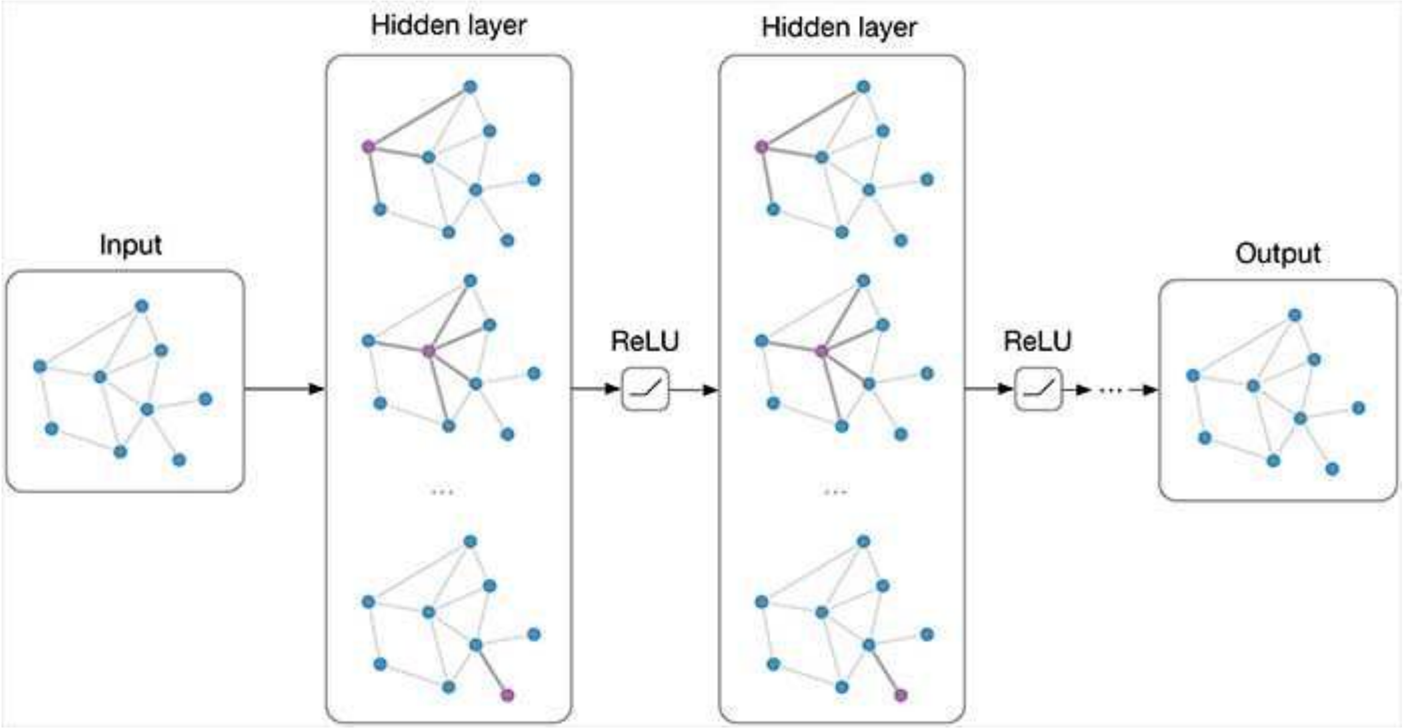


Attention is a complete graph between words

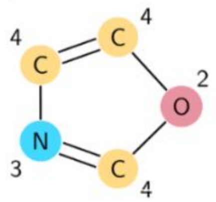




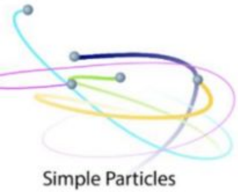
Graph Neural Networks (GNNs)



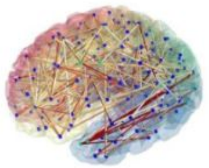
Merging Graphs and Transformers



Chemistry [1]
- Learn on molecules and predict chemical properties
- Use in drug repurposing



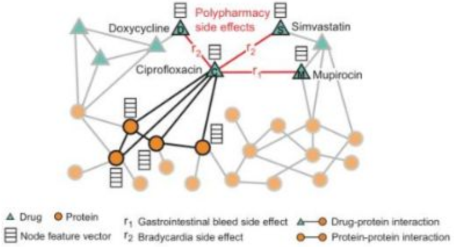
Physics [2]
- Learn from interactions of particles in systems
- Accelerate physics research



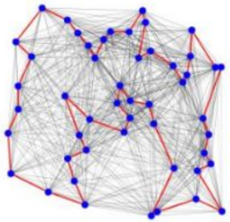
Neuroscience [5]
- Learn functions of brain regions through connectivity
- Accelerate brain-understanding and neuro-disease research

Numerous such examples of graph data.

Social networks [3]
- Learn from multi-faceted interactions among users
- Use for commercial and social applications

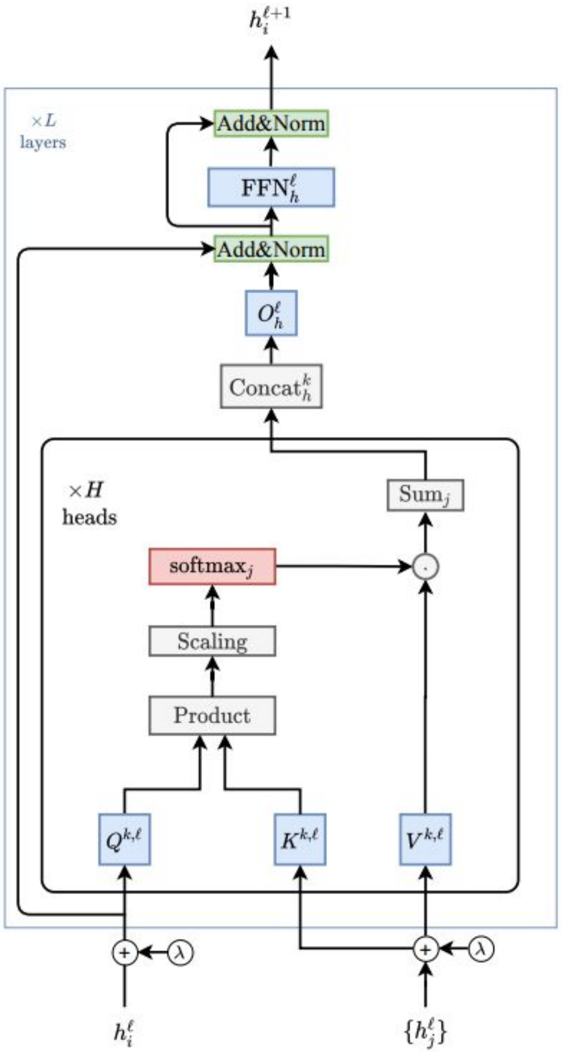


Medicine [4]
- Learn the effects of multiple drugs on body proteins
- Use for efficient multi-drug medical therapies



Combinatorial Optimization [6]
- Exploit the fact that most CO problems are rep. as graphs
- Develop better approximated solutions for NP-hard problems

Proposed architecture

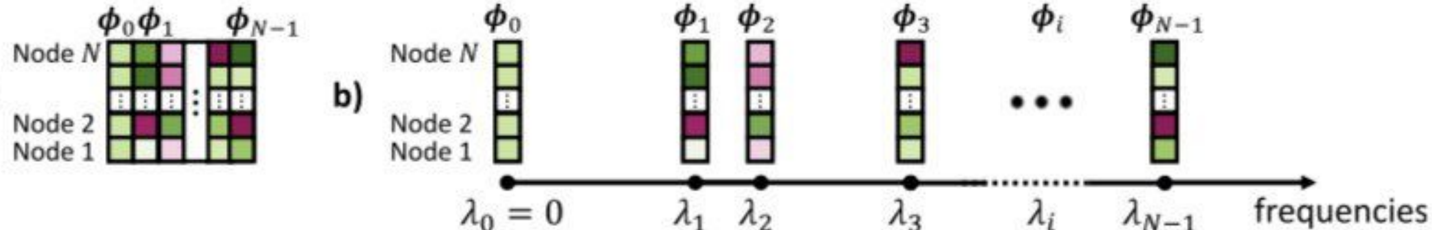


Laplacian Positional Encoding

Eigenvectors are defined via the factorization of the graph Laplacian matrix;

$$\Delta = \mathrm{I} - D^{-1/2} A D^{-1/2} = U^T \Lambda U, \qquad (1)$$

where A is the $n \times n$ adjacency matrix, D is the degree matrix, and Λ, U correspond to the eigenvalues and eigenvectors respectively. We use the k smallest non-trivial eigenvectors of a node as its positional encoding and denote by λ_i for node i.



Laplacian Positional Encoding

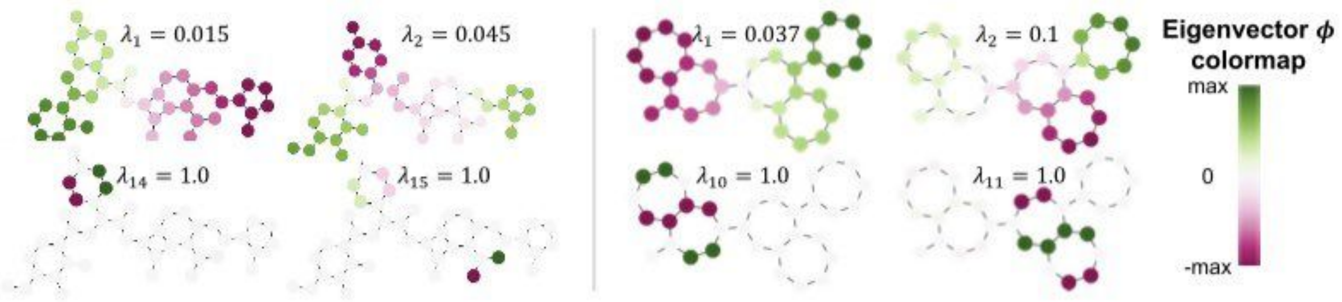
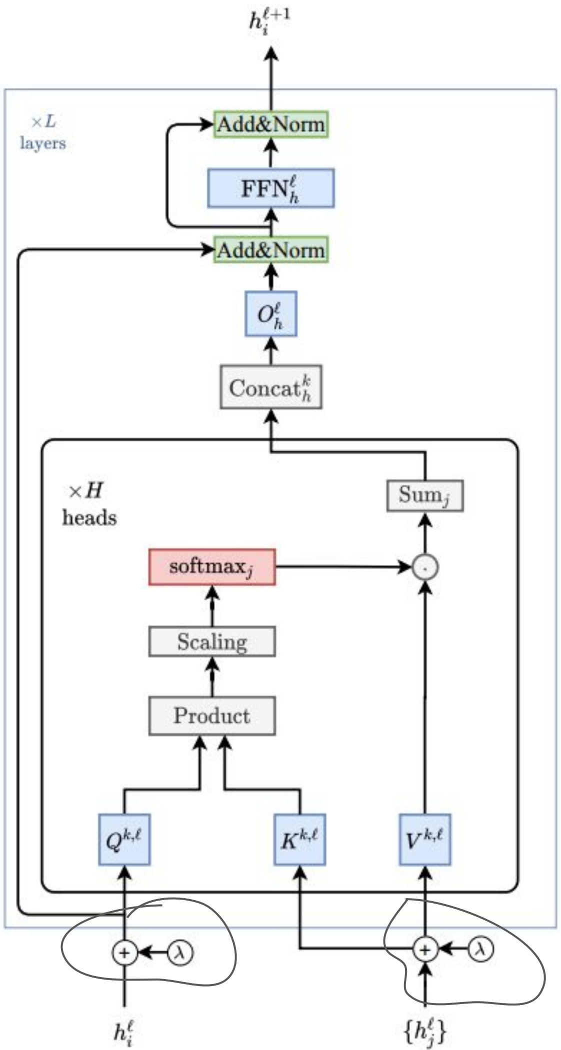


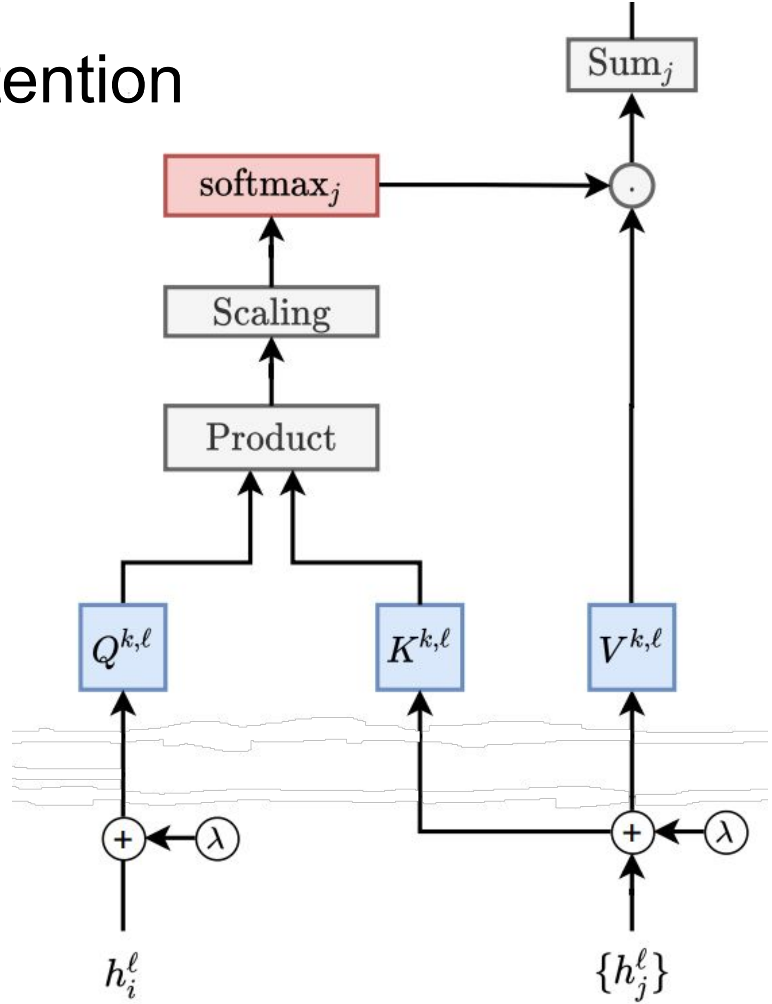
Figure 3: Examples of eigenvalues λ_i and eigenvectors ϕ_i for molecular graphs. The low-frequency eigenvectors ϕ_1, ϕ_2 are spread accross the graph, while higher frequencies, such as ϕ_{14}, ϕ_{15} for the left molecule or ϕ_{10}, ϕ_{11} for the right molecule, often resonate in local structures.

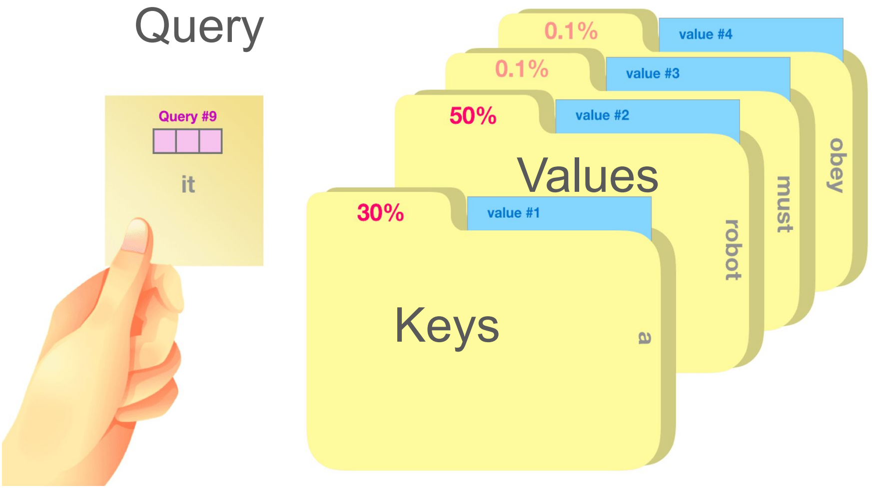
Just as the Fourier transform captures the frequency content of a signal, Laplacian eigenvectors capture the structural content of a graph. They help encode distance-aware information, which means that nearby nodes have similar positional features, and farther nodes have dissimilar positional features. Essentially, Laplacian eigenvectors help in understanding the geometrical structure of the graph.

Laplacian Positional Encoding



Attention

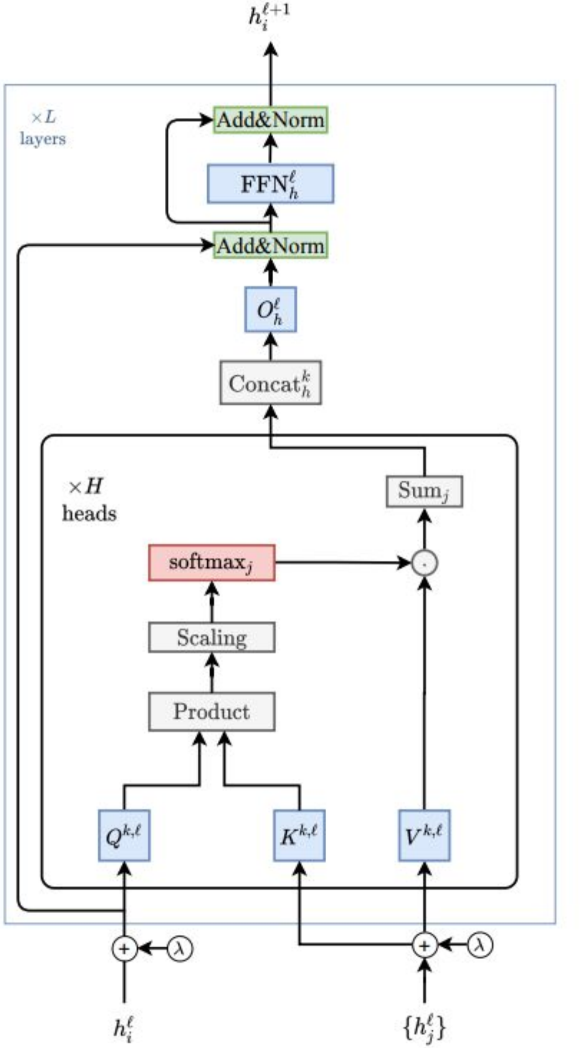




Attention Vector = $\sum_{j\in\mathcal{N}_i} w_{ij}^{k,\ell} V^{k,\ell} h_j^\ell$

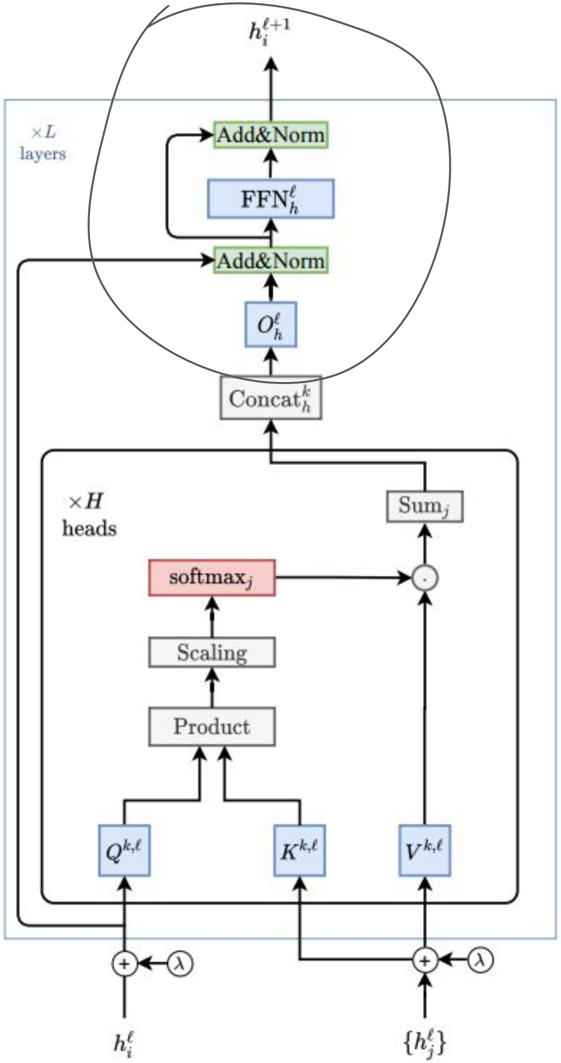
$$w_{ij}^{k,\ell} = \mathrm{softmax}_j\left(\frac{Q^{k,\ell}h_i^\ell \cdot K^{k,\ell}h_j^\ell}{\sqrt{d_k}}\right)$$

Multi-head attention



Feed Forward Network

$$
\begin{aligned}
\hat{\hat{h}}_i^{\ell+1} &= \text{Norm}\left(h_i^{\ell} + \hat{h}_i^{\ell+1}\right), \\
\hat{\hat{\hat{h}}}_i^{\ell+1} &= W_2^{\ell}\text{ReLU}(W_1^{\ell}\hat{\hat{h}}_i^{\ell+1}), \\
h_i^{\ell+1} &= \text{Norm}\left(\hat{\hat{h}}_i^{\ell+1} + \hat{\hat{\hat{h}}}_i^{\ell+1}\right)
\end{aligned}
$$



BatchNorm

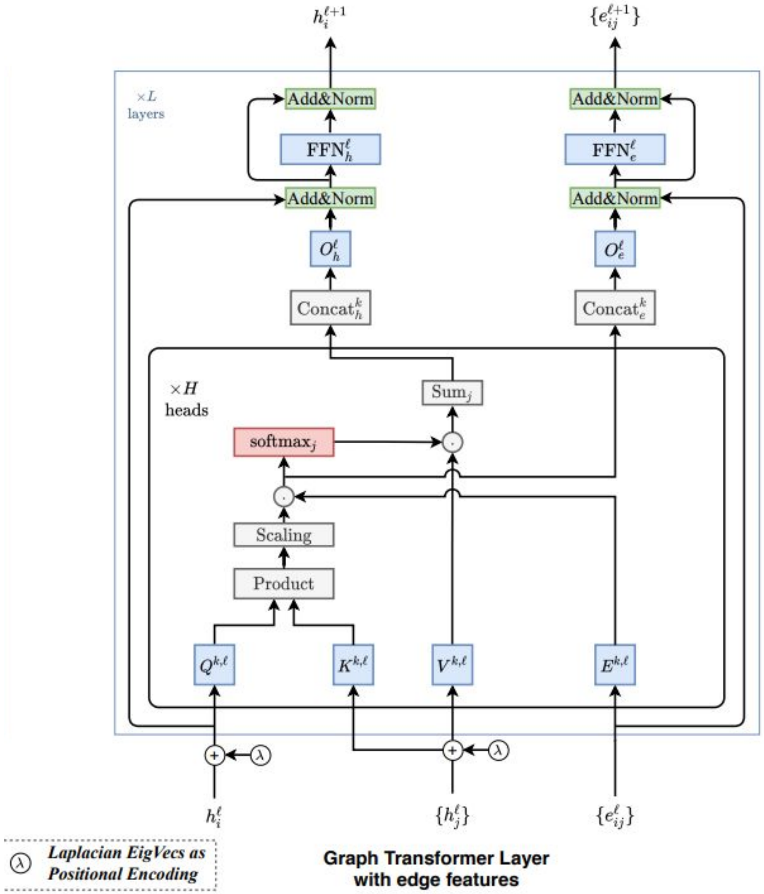
$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \text{ and } \sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2.$$

$$\hat{x}_i^{(k)} = \frac{x_i^{(k)} - \mu_B^{(k)}}{\sqrt{\left(\sigma_B^{(k)}\right)^2 + \epsilon}}$$

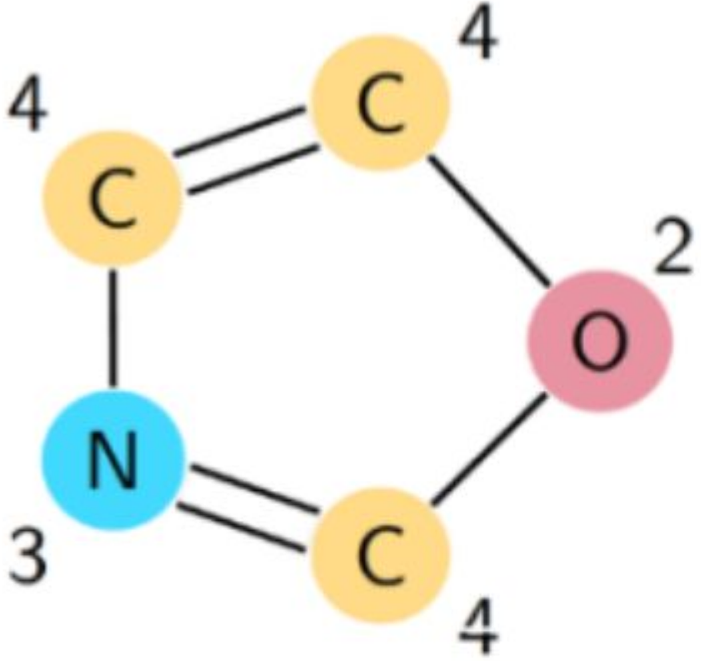
What if we have edge features?

- Just multiply the weights in the attention on the edge features! And then softmax

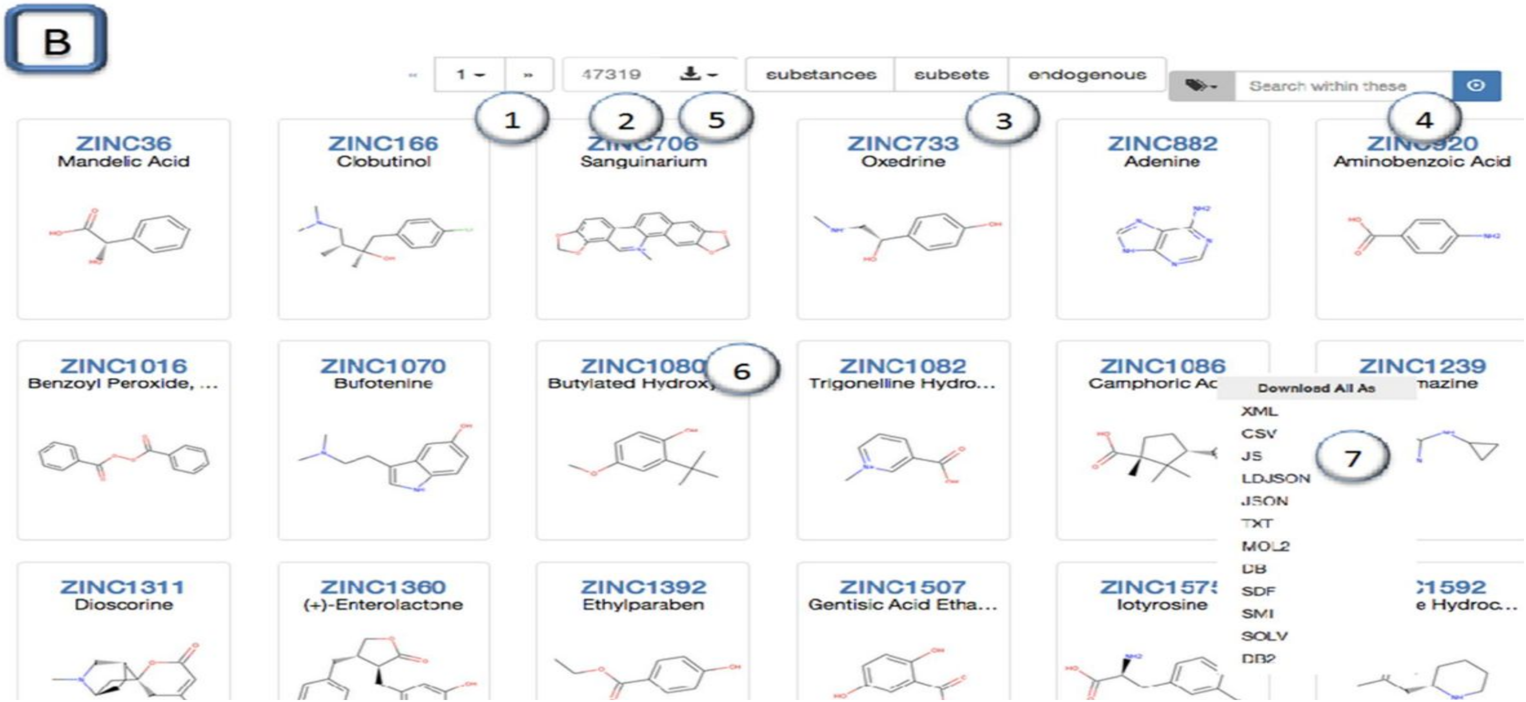
- Update the edge features with the new values



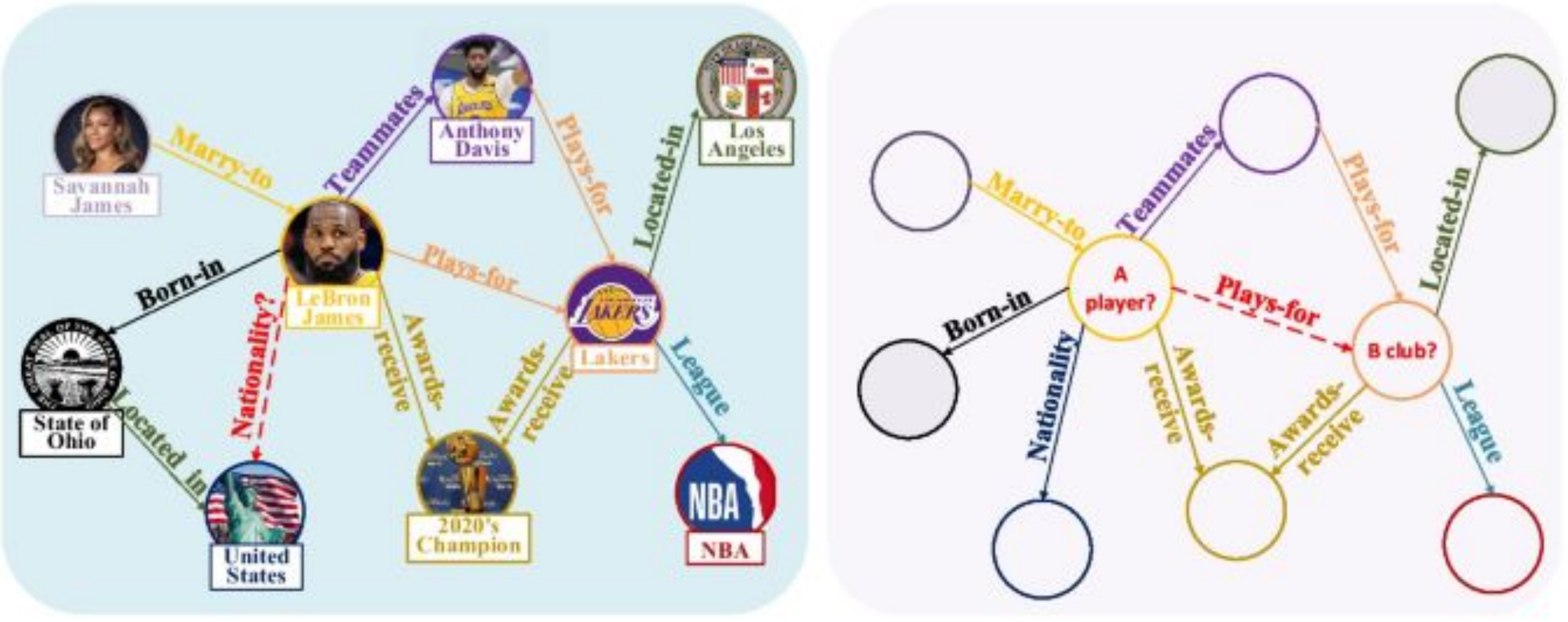
What if we have edge features? Example: molecules



Graph benchmark datasets: ZINC



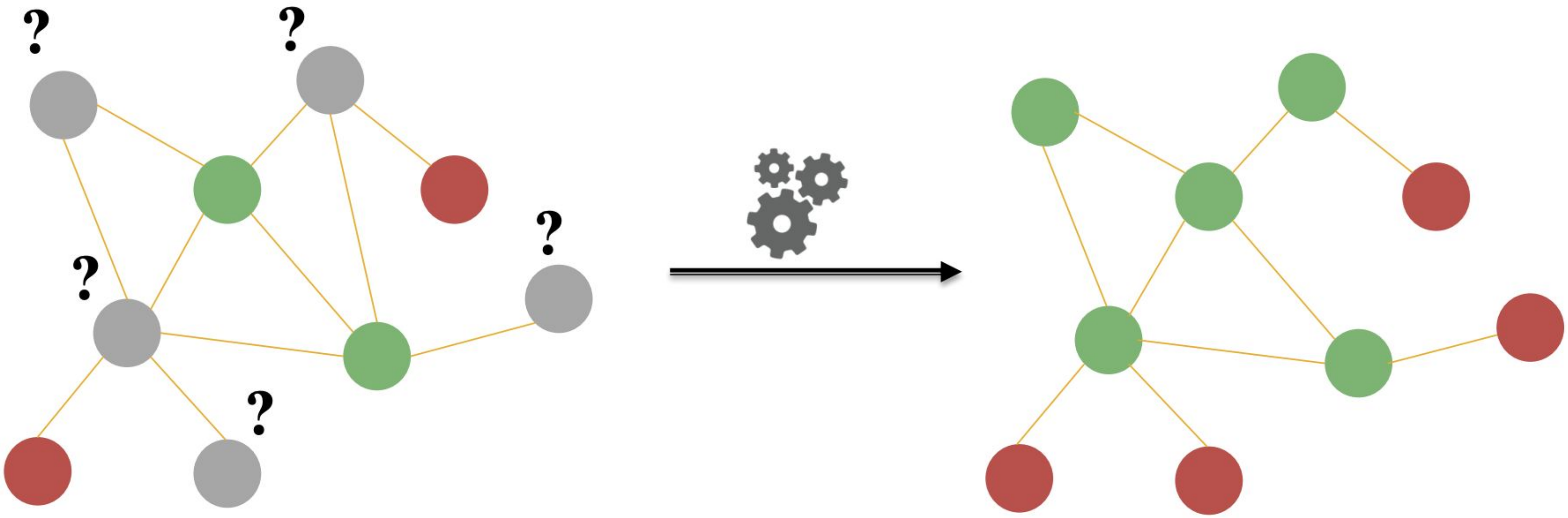
Other example of edge features: link prediction



(a) training graph and transductive link prediction

(b) inductive link prediction

Graph benchmark datasets: PATTERN and CLUSTER



Comparison to previous models

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

Comparison to other PEs

Dataset	PE	#Param	Sparse Graph Test Perf.±s.d.	Train Perf.±s.d.	#Epoch	Epoch/Total
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

Analysis of GraphTransformer (GT) using different PE schemes. Notations x: No PE; L: LapPE (ours); W: WLPE (Zhang et al. 2020). Bold: the best performing model for each dataset.

Thank you for your *attention*