Introduction to Graph Transformers

Vijay Prakash Dwivedi

Postdoctoral Scholar

Stanford University

28 December 2024

Vijay Prakash Dwivedi

@vijaypradwi

Birgunj, Nepal

Postdoc at Stanford University (‘24 – now)

PhD, 2019-2023
(Outstanding PhD Thesis Award)
College of Computing and Data Science
Nanyang Technological University, Singapore

Industry:
ASUS AICS, Singapore (‘23 – ‘24),
Snap Inc., Seattle, USA (‘23),
Sea AI Lab, Singapore, (‘22-’23),
Sony Group Corporation, Tokyo (‘22),

Why Graph Transformers?

- Transformers have been successful in extracting meaningful representations from data, and at scale!

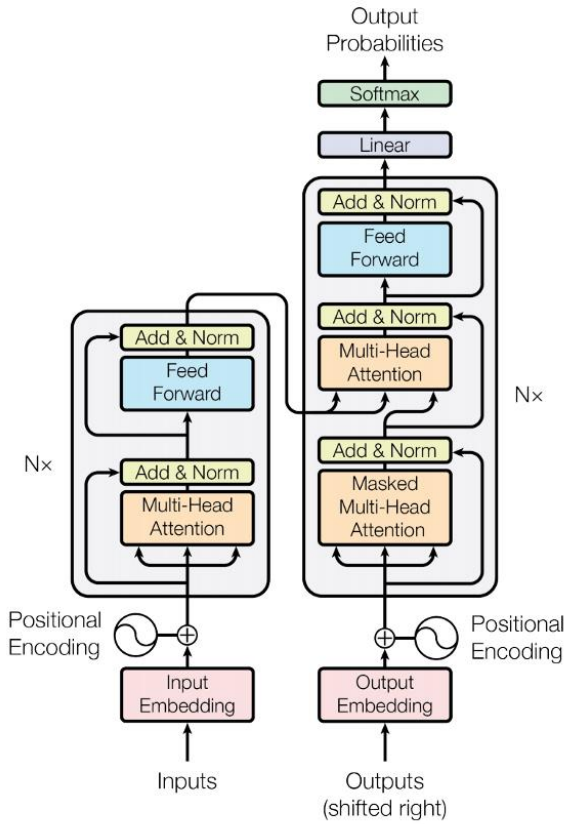
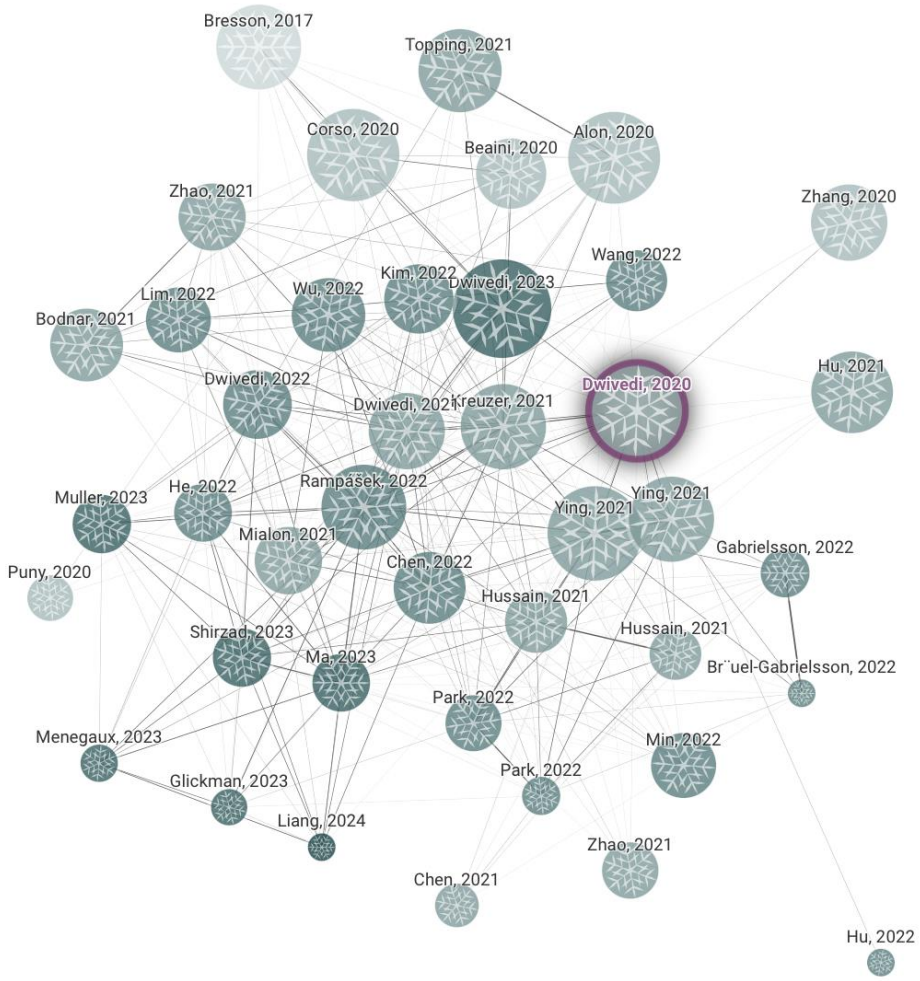


Figure 1: The Transformer - model architecture.

Why Graph Transformers?

- Transformers have been successful in extracting meaningful representations from data, and at scale!

- Graphs (or, Networks) are languages to represent problems where complex interactions among objects are represented as connectivity structure

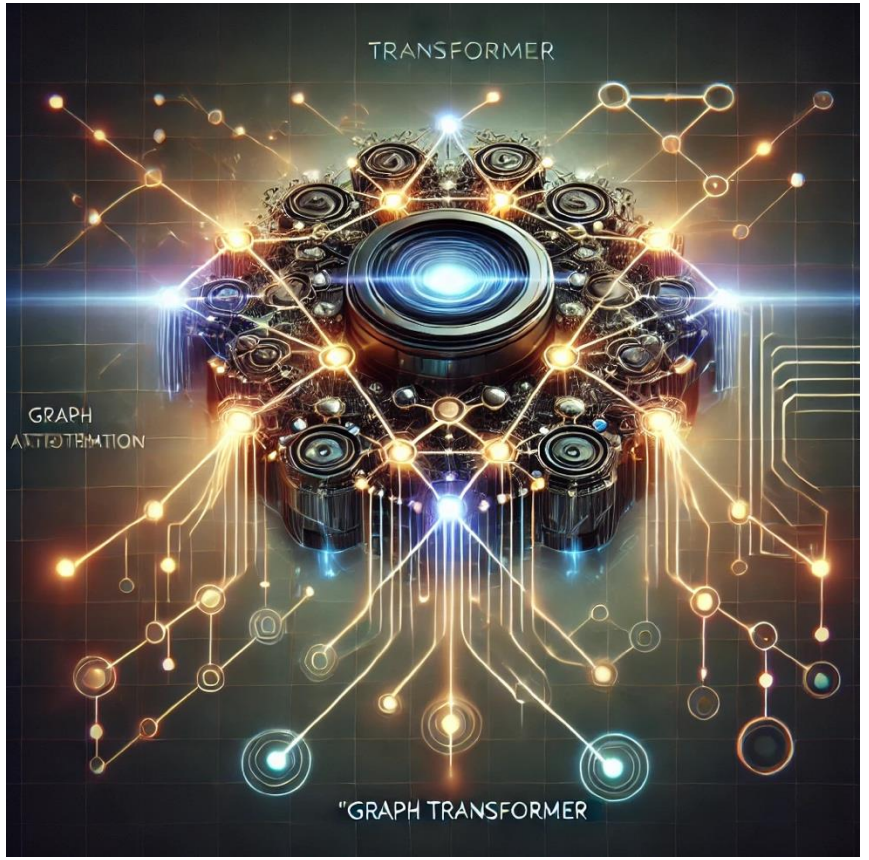


Why Graph Transformers?

- Transformers have been successful in extracting meaningful representations from data, and at scale!
- Graphs (or, Networks) are languages to represent problems where complex interactions among objects are represented as connectivity structure
- Graph structured datasets are available in abundance

Why Graph Transformers?

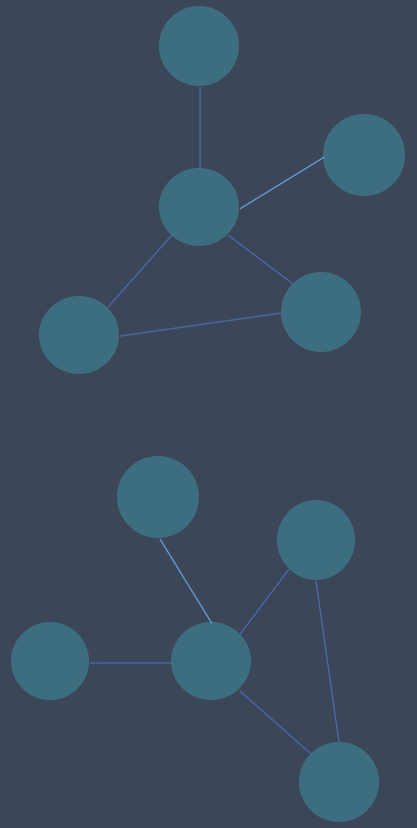
- Transformers have been successful in extracting meaningful representations from data, and at scale!
- Graphs (or, Networks) are languages to represent problems where complex interactions among objects are represented as connectivity structure
- Graph structured datasets are available in abundance
- Let's generalize Transformers for such graph datasets!



The image was generated by gpt-4o using "graph transformer" prompt

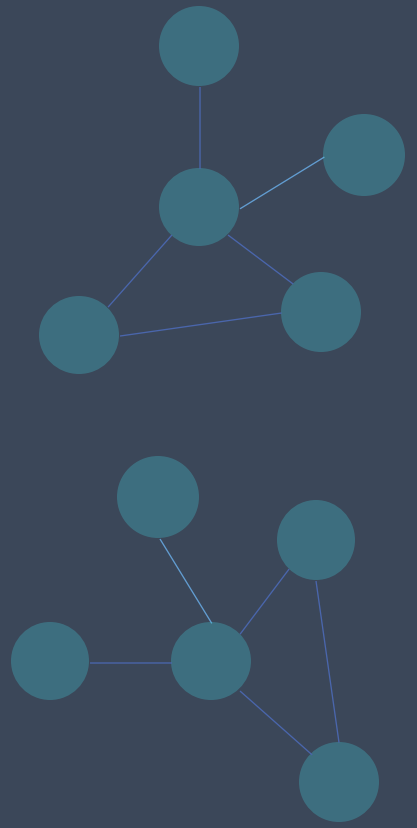
Outline of Presentation

- Transformers
- Attention as Message Passing
- Graph Transformers
- Bottlenecks and Remedies
- Recent Research

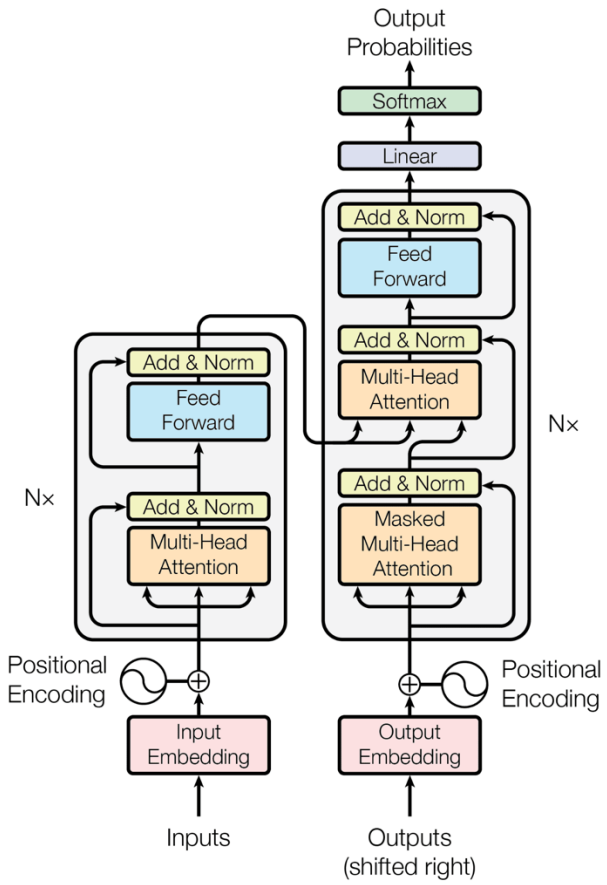


Outline of Presentation

- **Transformers**
- Attention as Message Passing
- Graph Transformers
- Bottlenecks and Remedies
- Recent Research



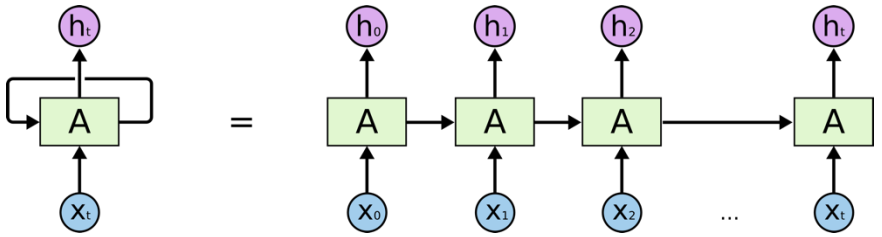
Transformers



- Transformers [1] are deep neural network architectures that operates on sets of input tokens
- Use of multi-head attention mechanism
- Capture dependencies between different tokens
- Popularized from seq2seq models in NLP
- Why Transformers ? →

[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.

Why Transformers

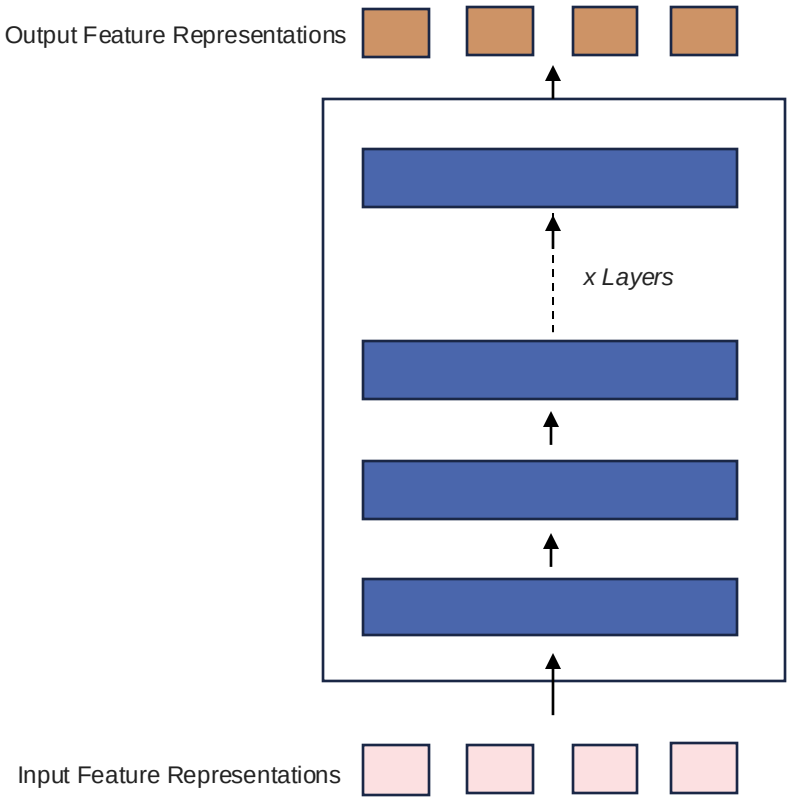


Recurrent Neural Networks (RNNs) for sequential inputs. Source colah.github.io

Transformers **offer several advantages** over traditional recurrent neural networks:

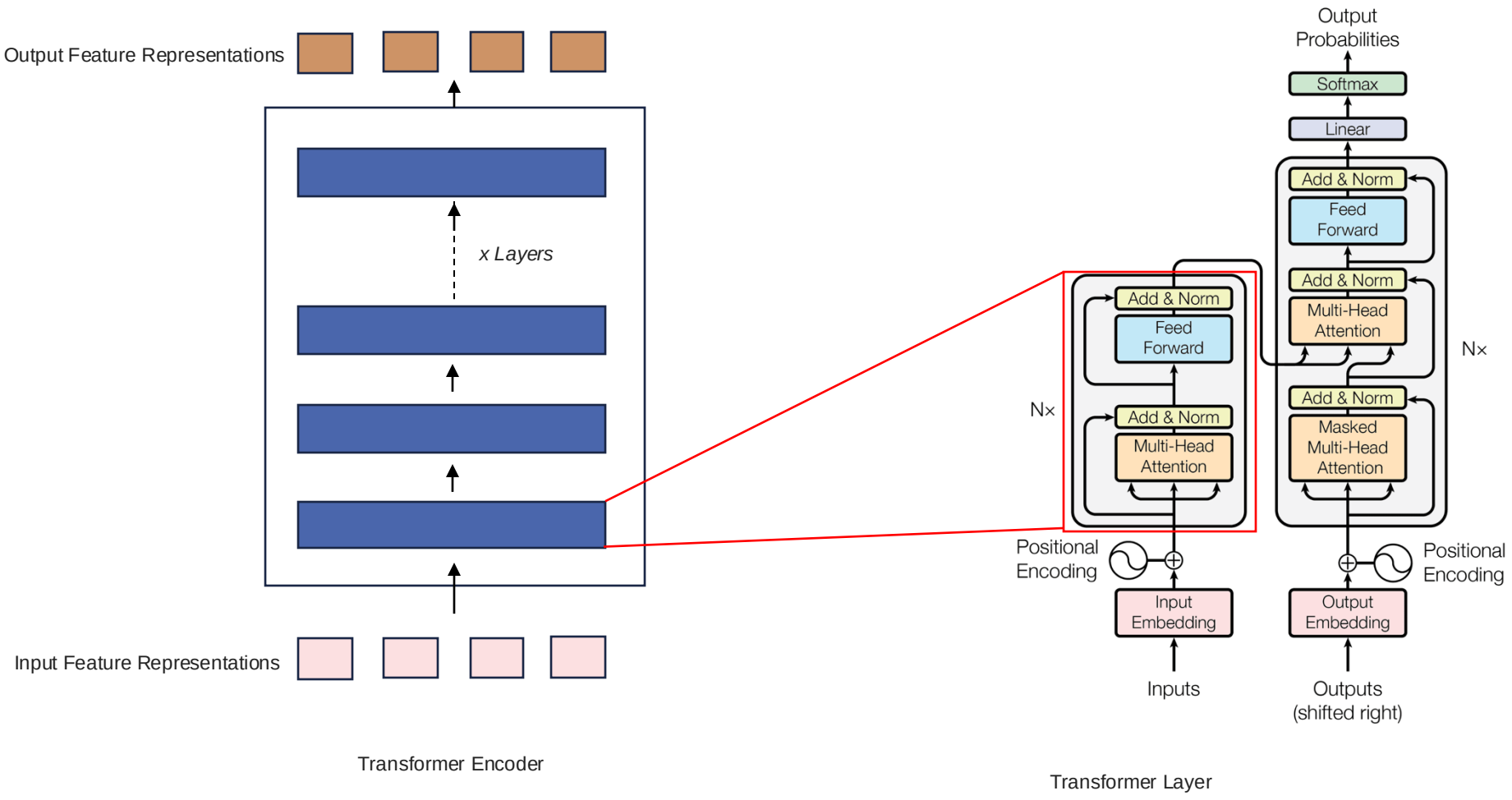
- Long term dependencies: self-attention allows capturing dependencies across input tokens.
- Global context: can model interactions among distant tokens
- Parallelization & Scalability: can operate over all tokens in the input at one time, unlike RNNs.
- Interpretability: attention mechanism allows to investigate implicit dependencies between tokens.

Transformer Encoder

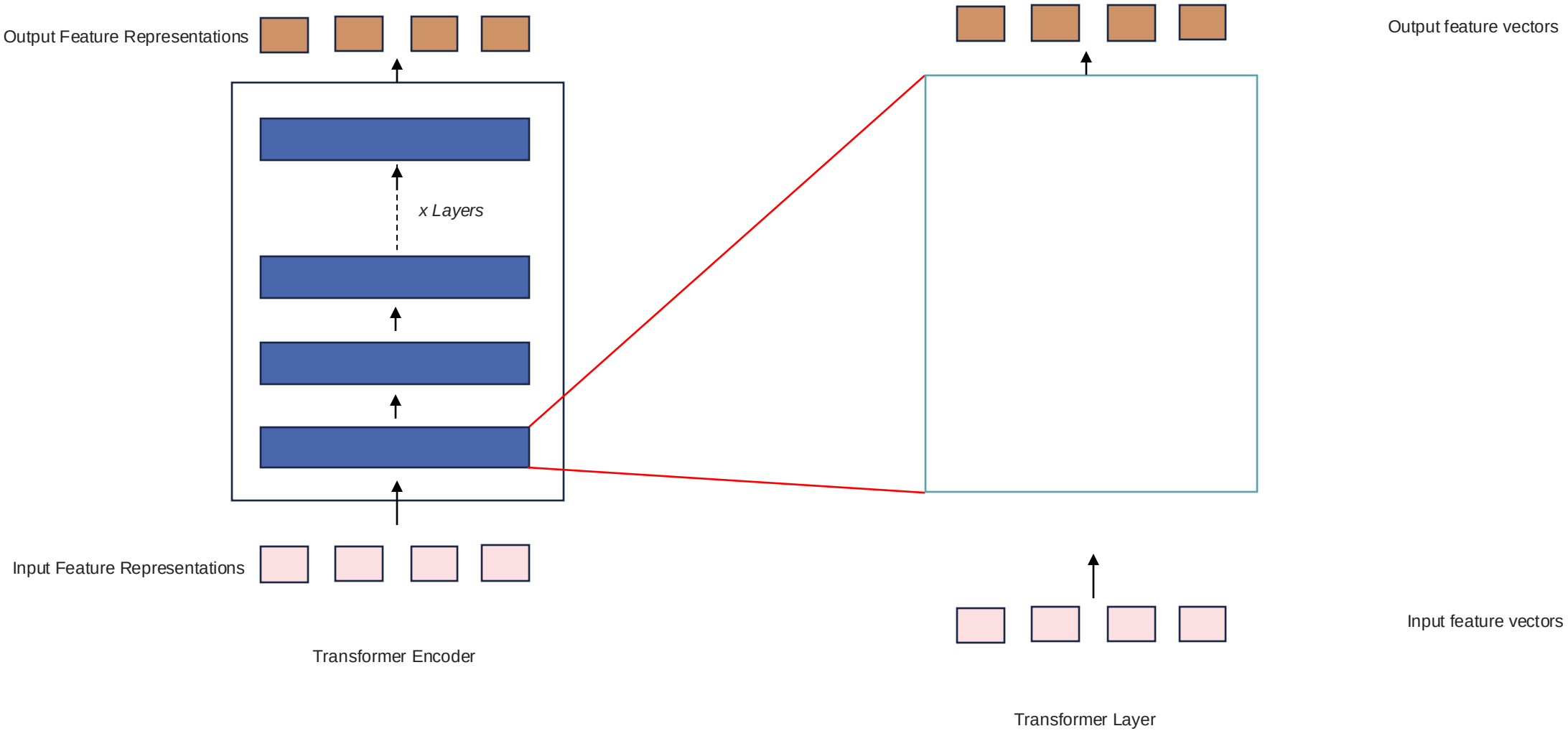


Transformer Encoder

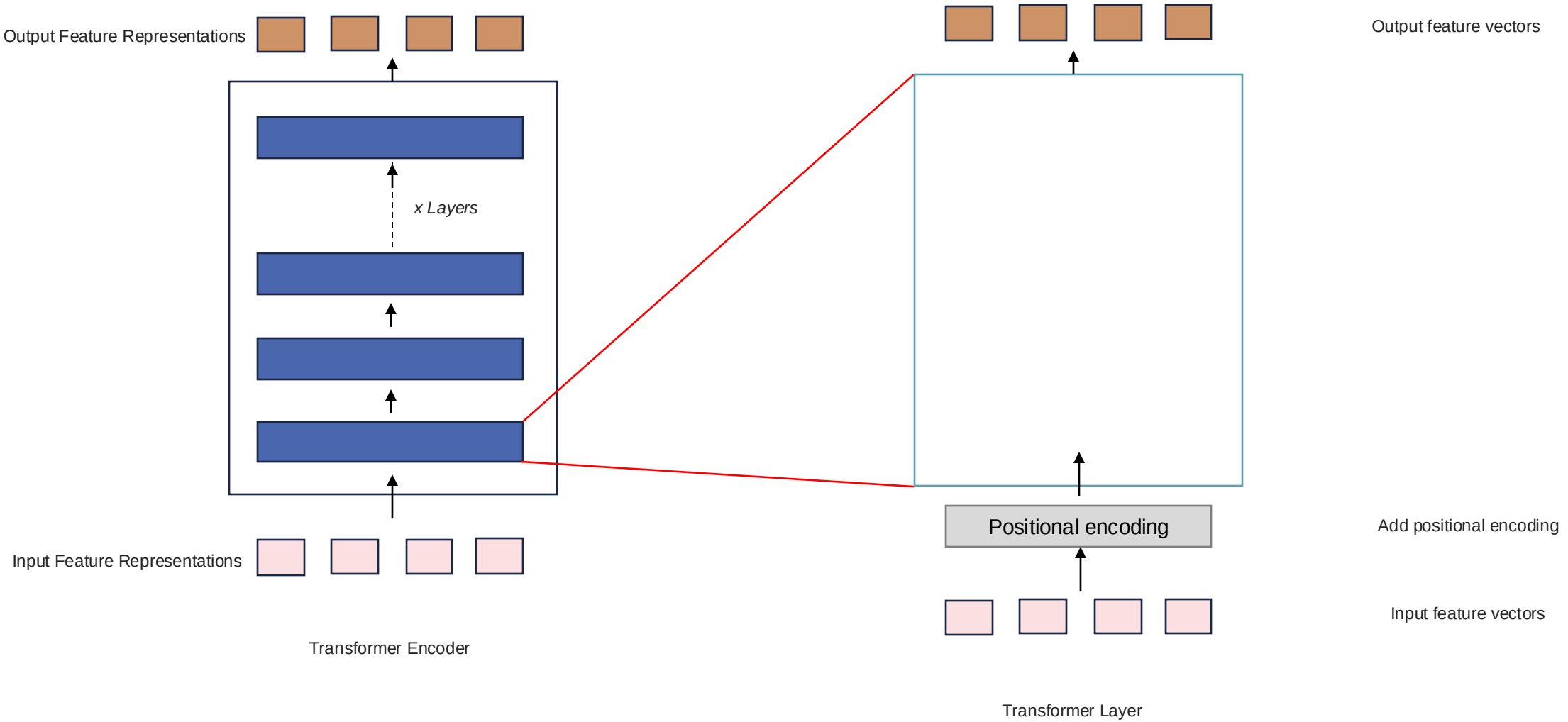
Transformer Encoder



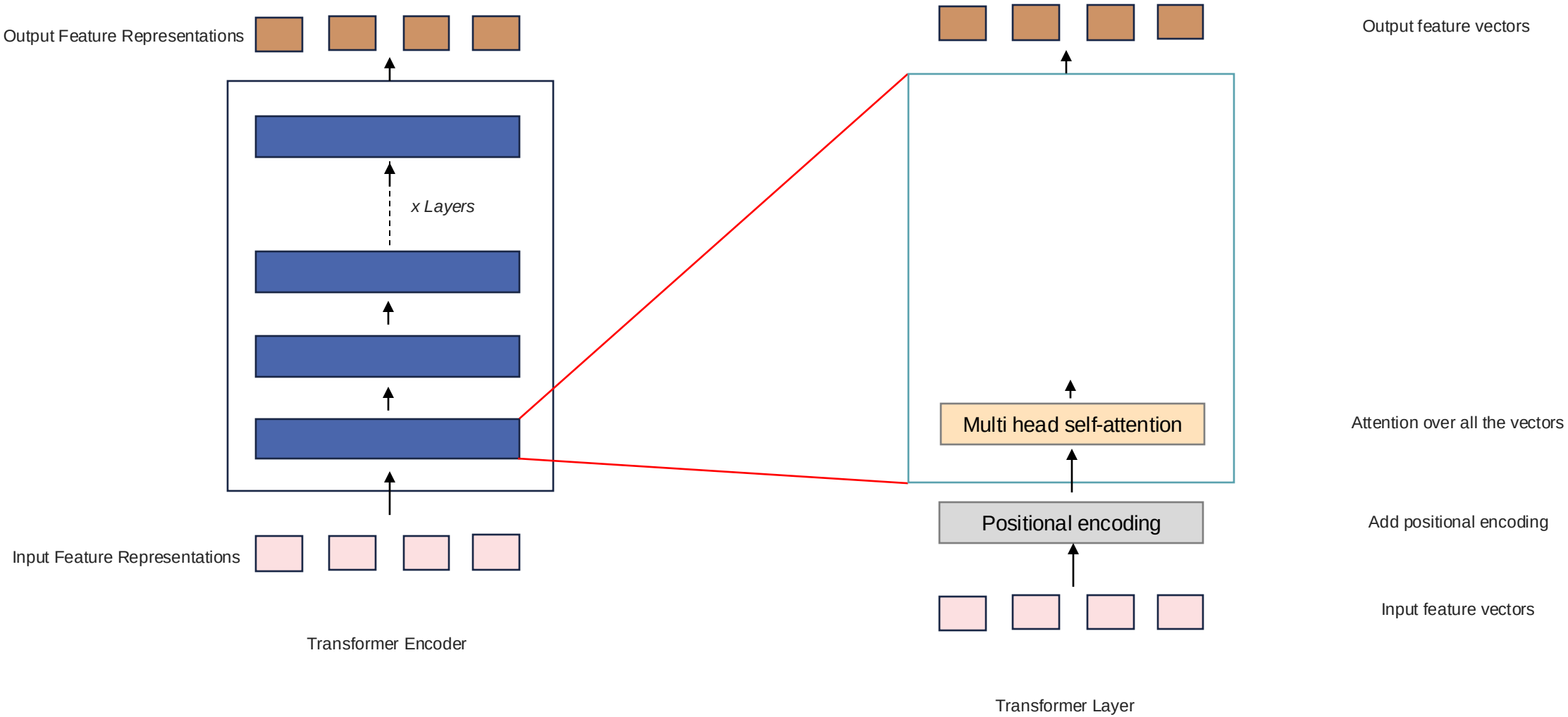
Transformer Encoder



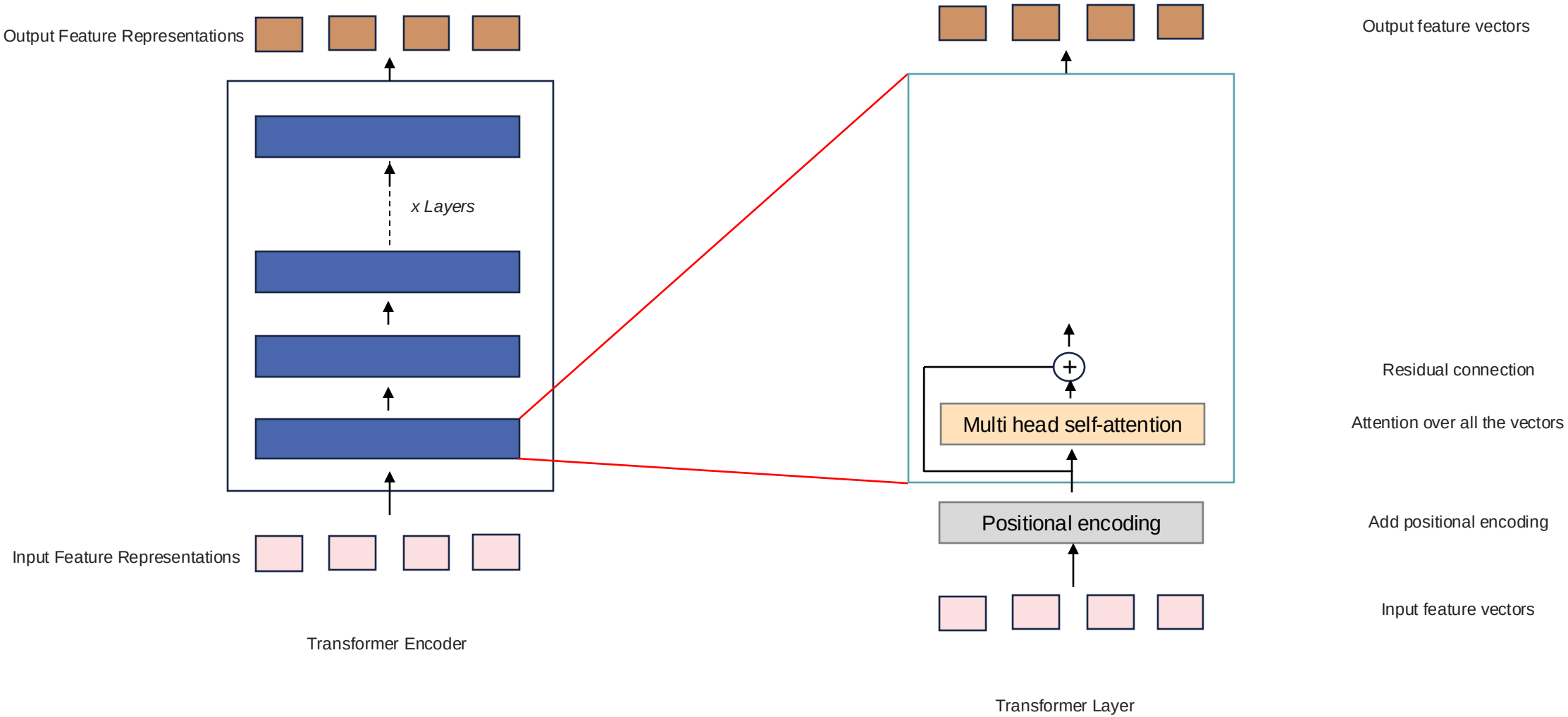
Transformer Encoder



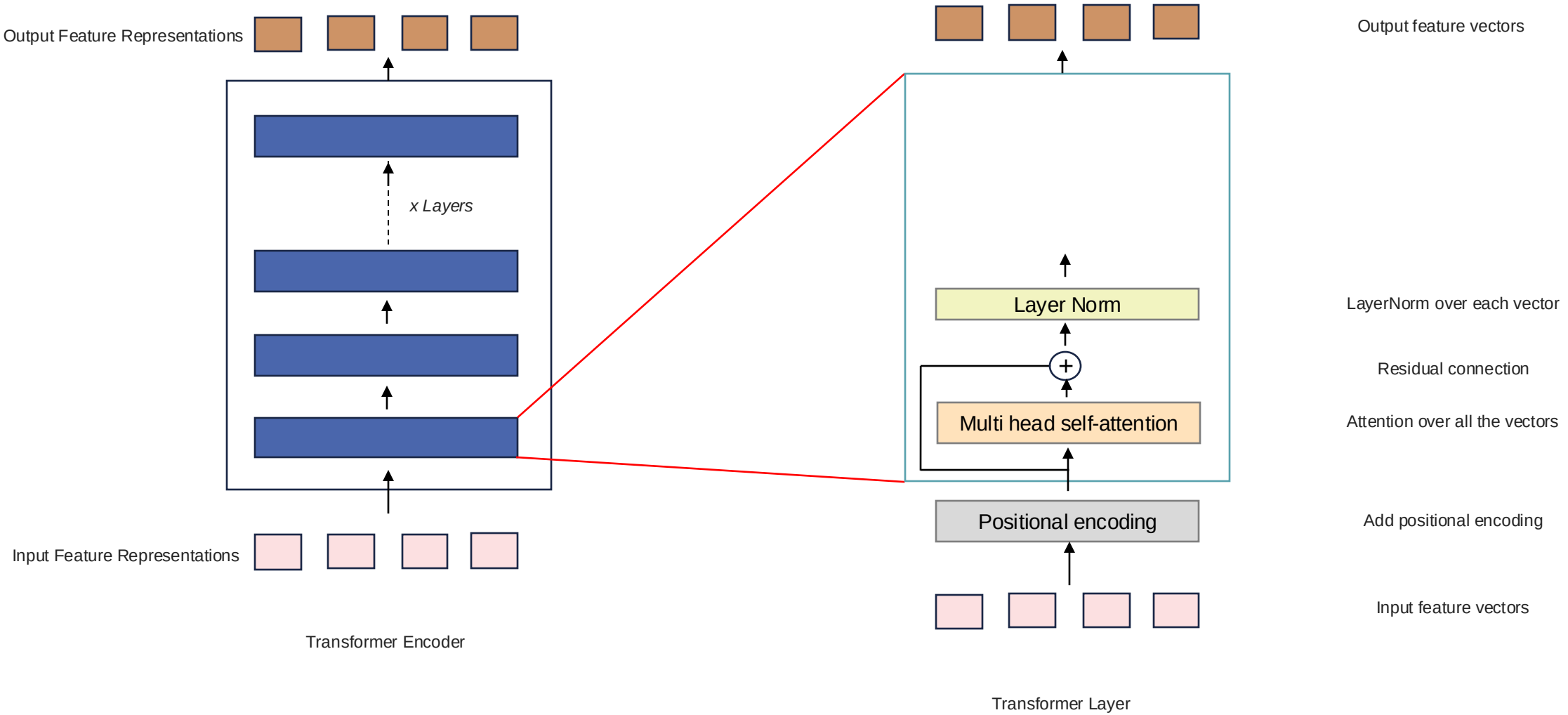
Transformer Encoder



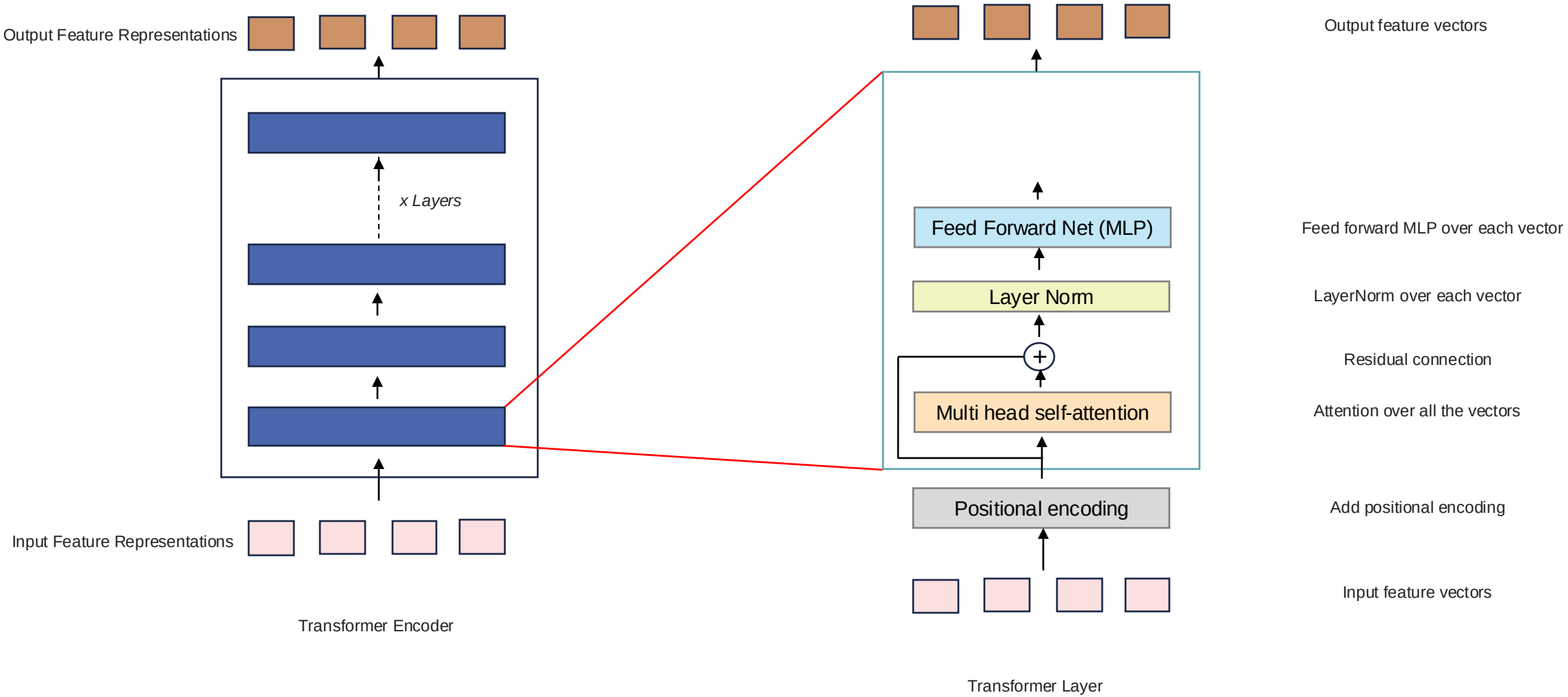
Transformer Encoder



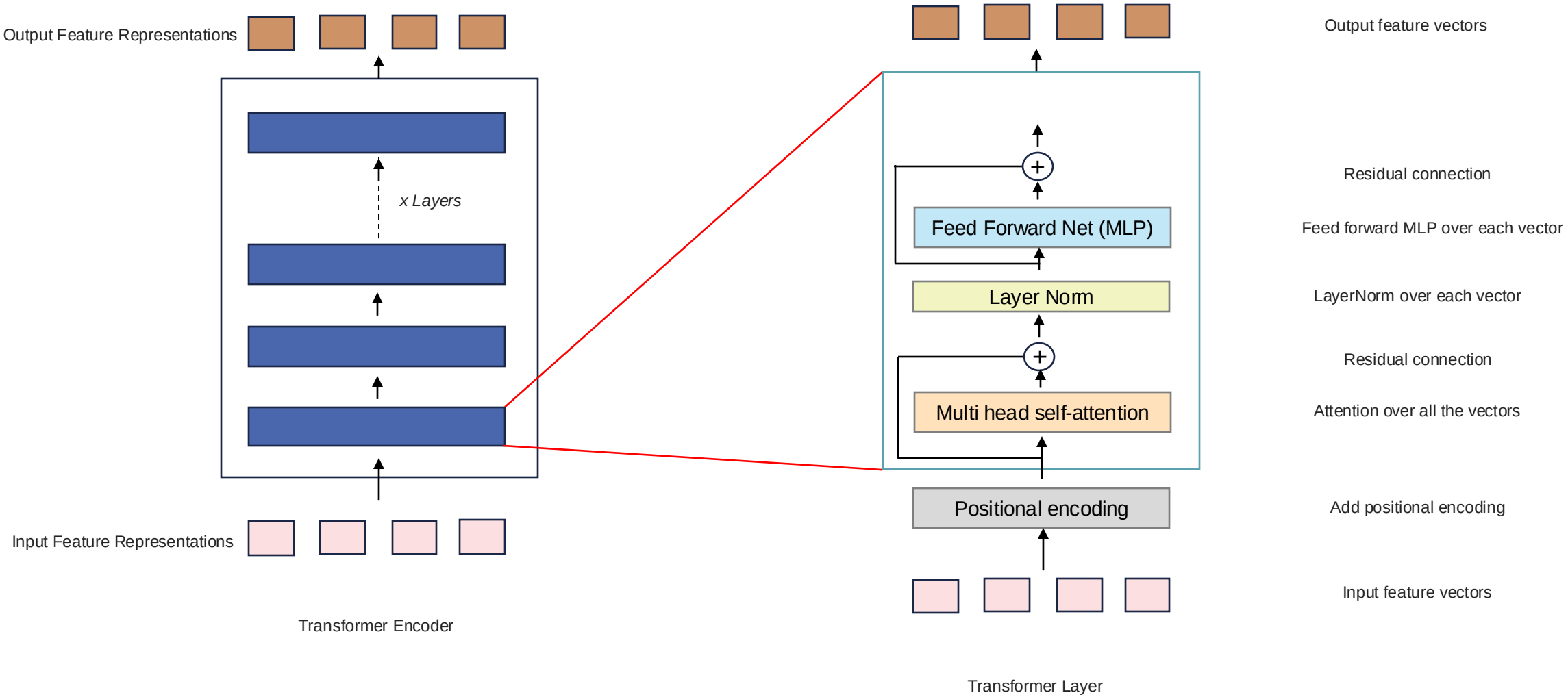
Transformer Encoder



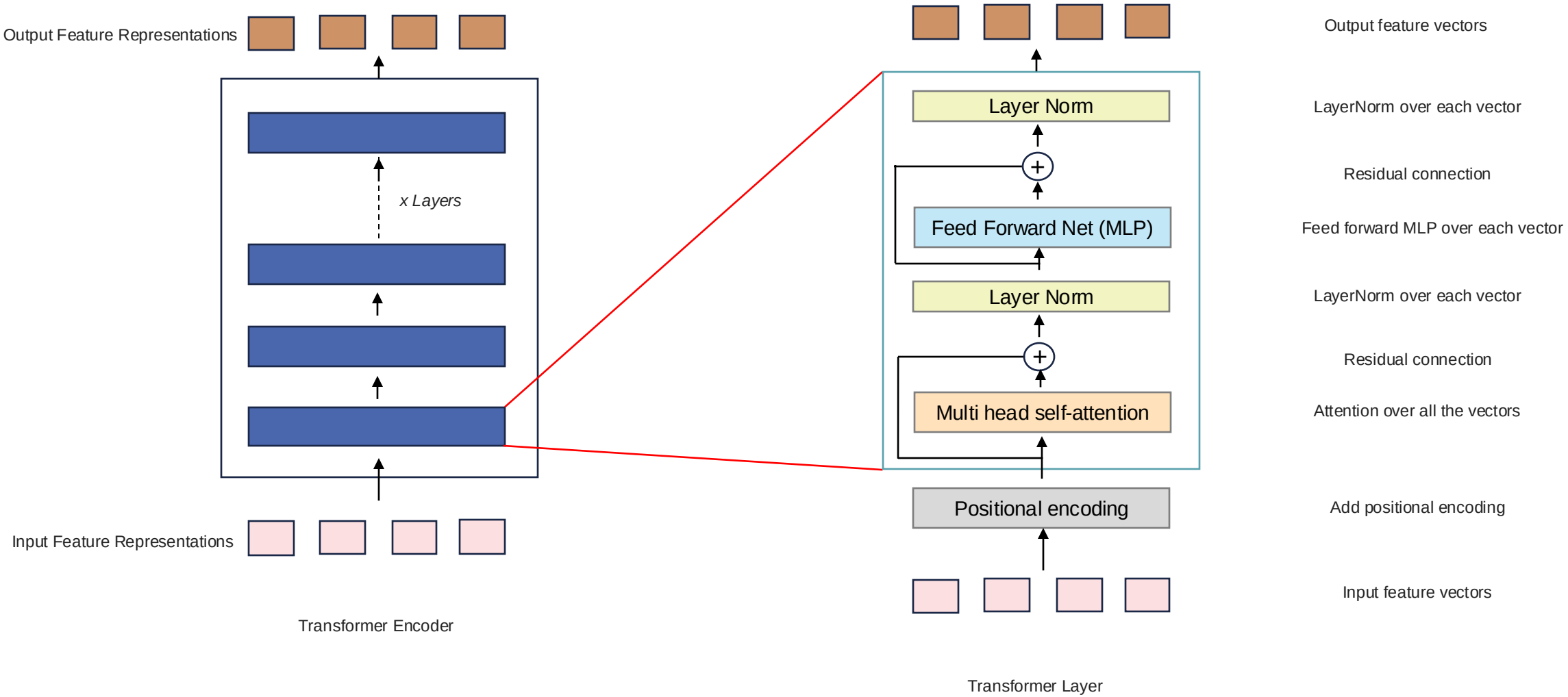
Transformer Encoder



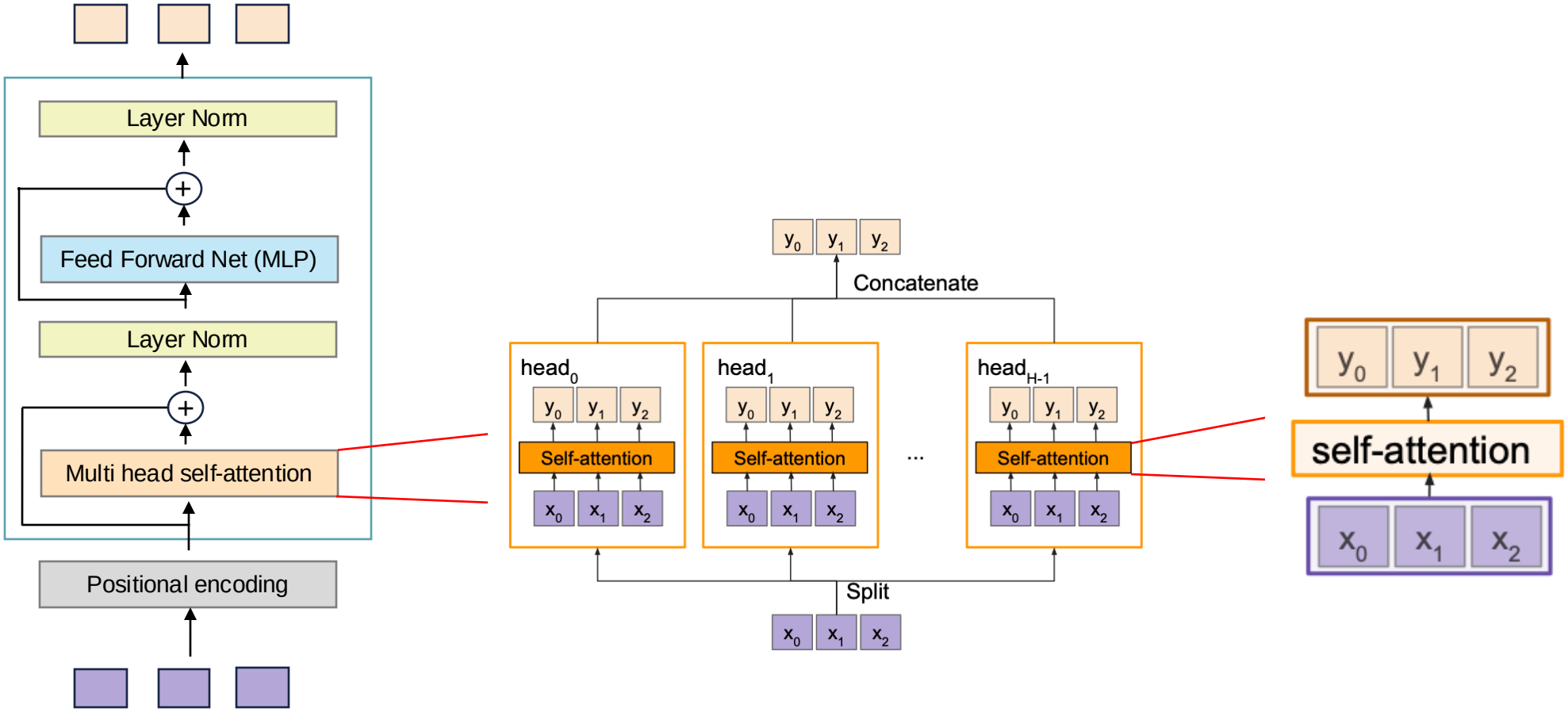
Transformer Encoder



Transformer Encoder



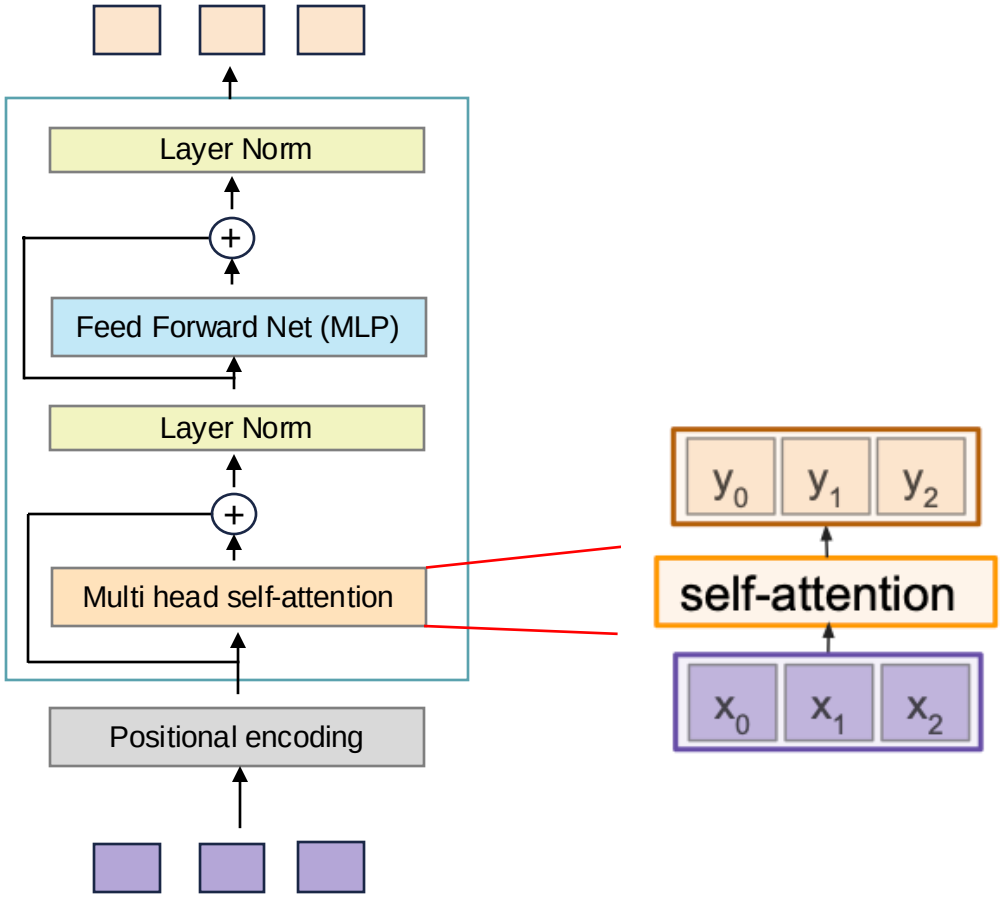
Multi Head Self Attention



Transformer Layer

Adapted from: cs231n Stanford

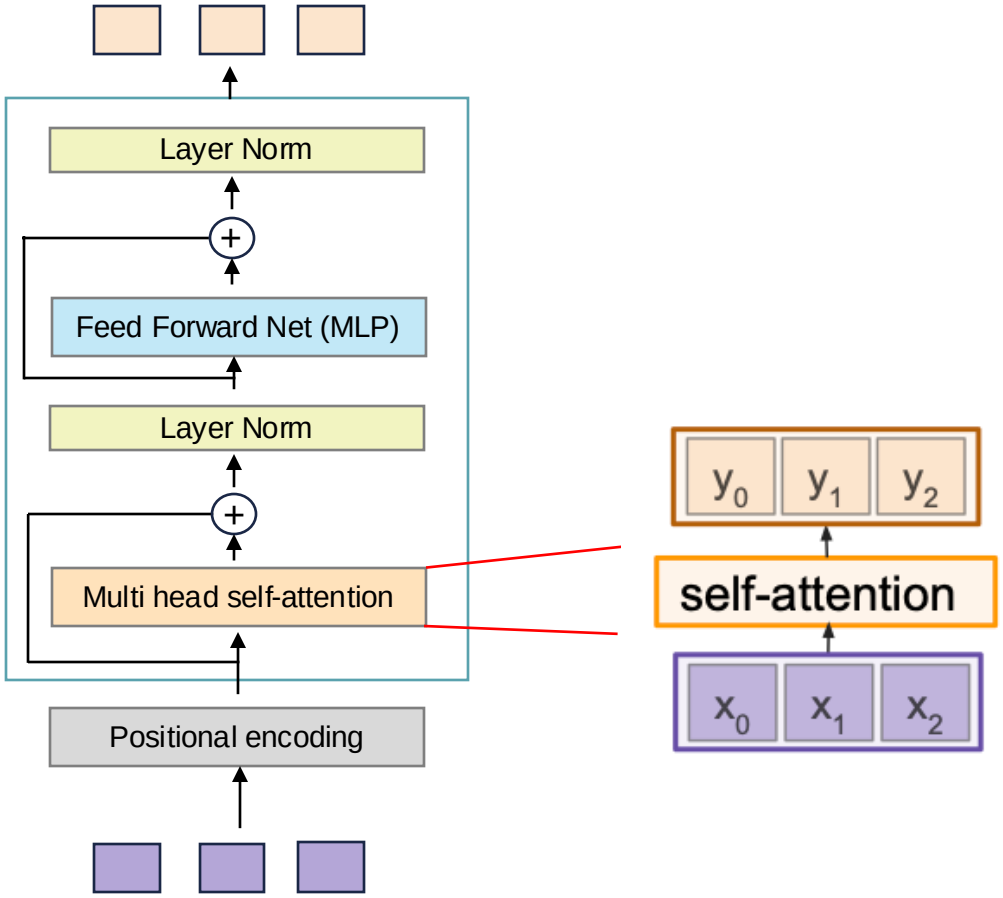
Self Attention



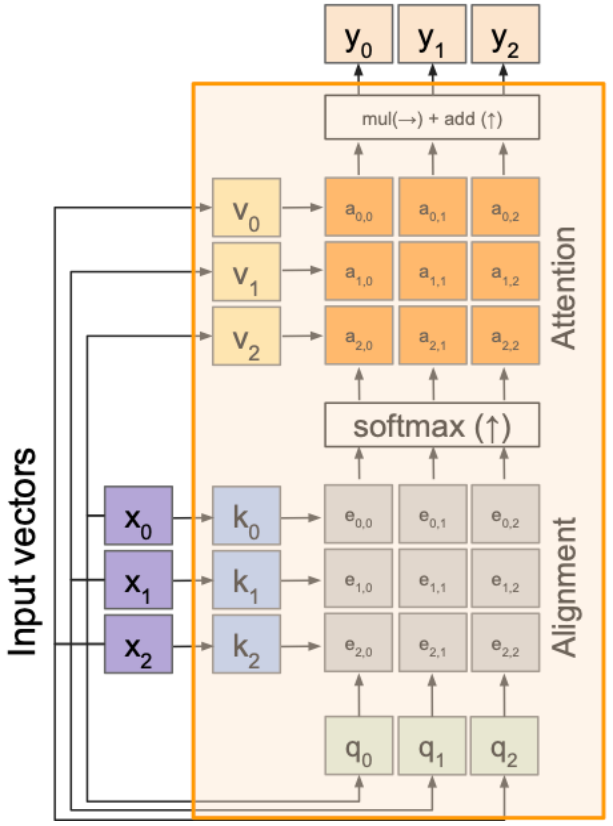
Transformer Layer

Adapted from: cs231n Stanford

Self Attention



Transformer Layer



Outputs:
context vectors: **y** (shape: D_v)

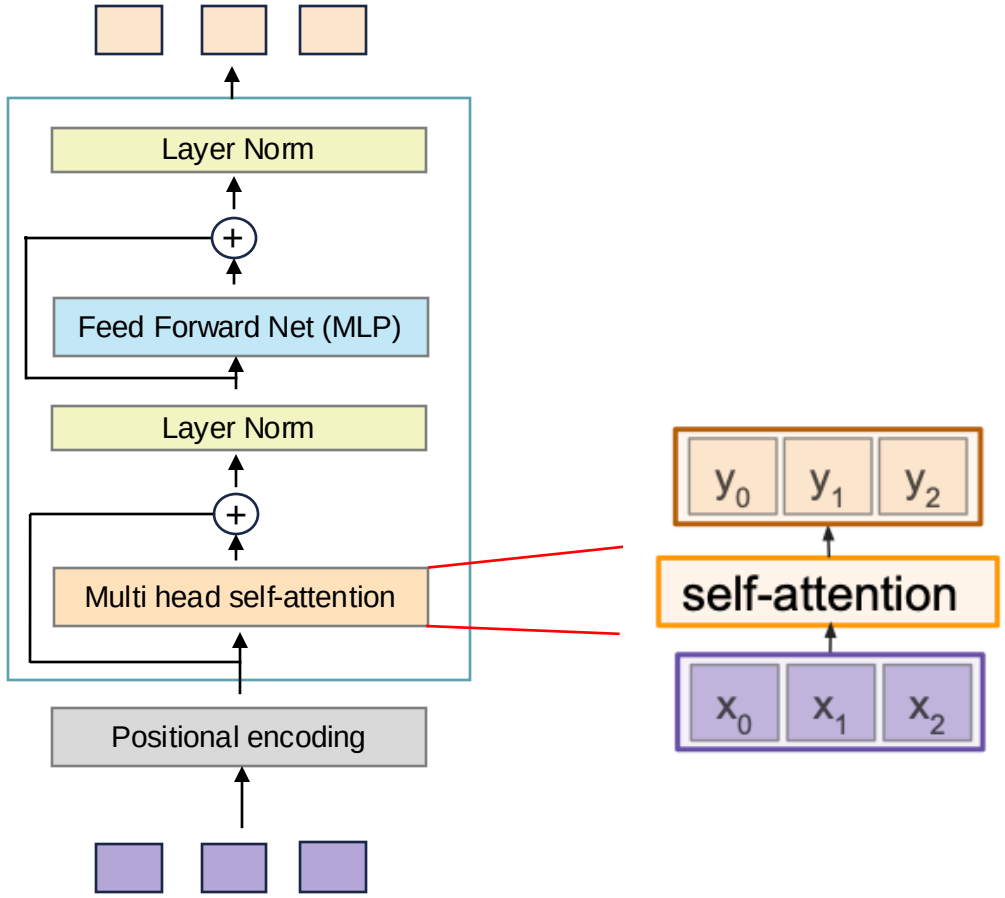
Operations:
Key vectors: $k = xW_k$
Value vectors: $v = xW_v$
Query vectors: $q = xW_q$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $a = \text{softmax}(e)$
Output: $y_j = \sum_i a_{i,j} v_i$

Inputs:
Input vectors: **x** (shape: N x D)

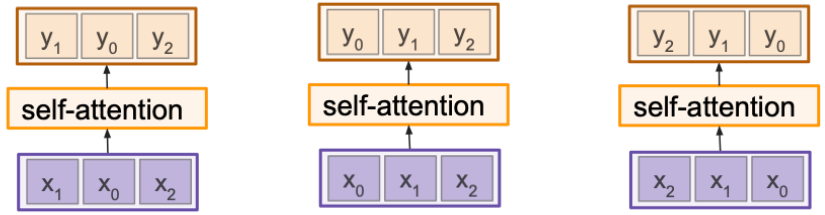
Adapted from: cs231n Stanford

Self Attention

- Self Attention is Permutation Equivariant

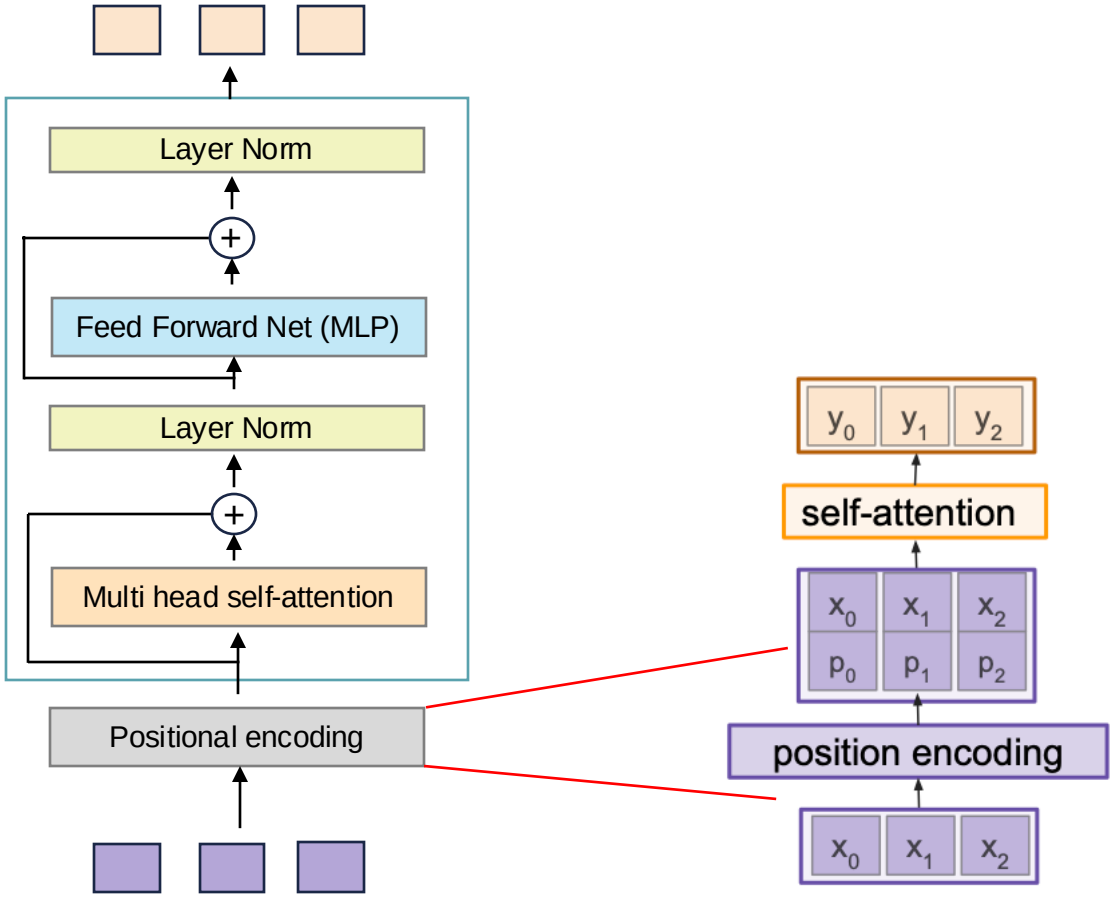


Transformer Layer



Adapted from: cs231n Stanford

Positional Encoding



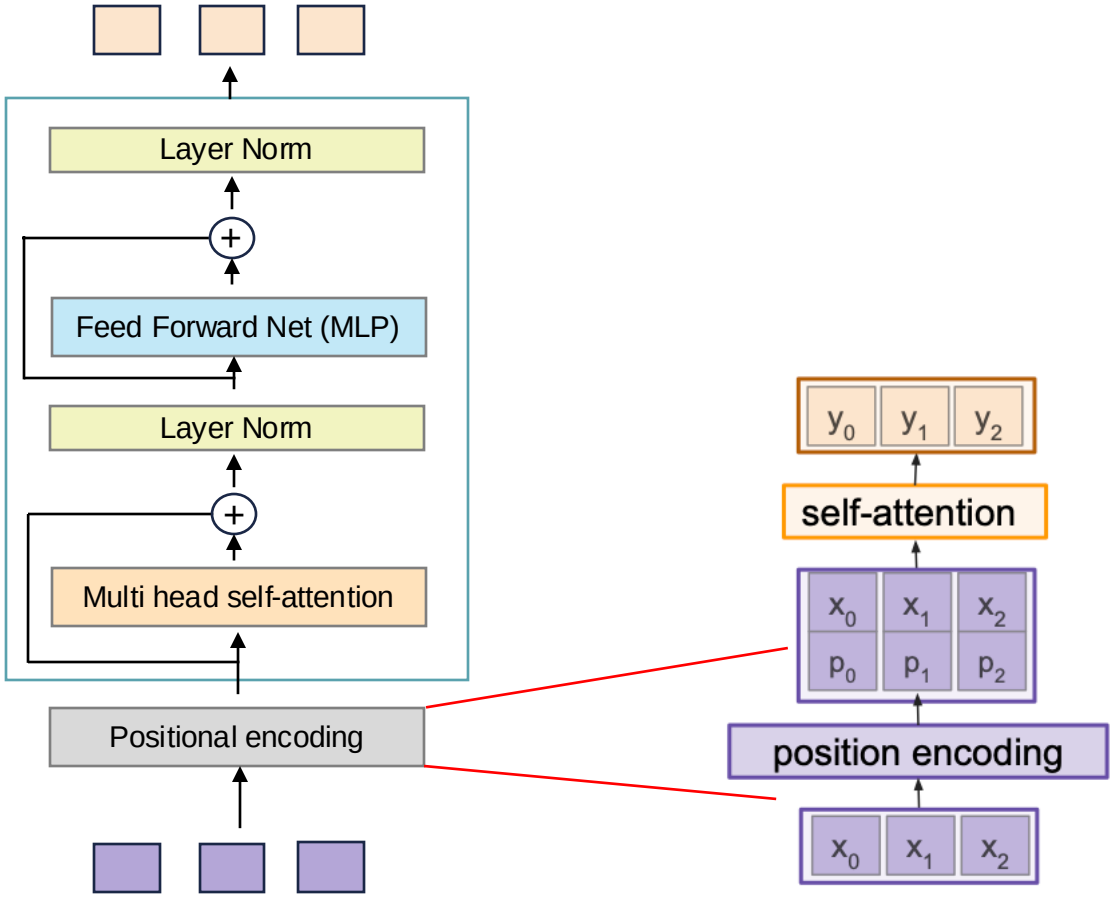
Transformer Layer

- Self Attention is Permutation Equivariant

- How can we encode the order of tokens?
- How can we encode the underlying structure of the tokens?

Adapted from: cs231n Stanford

Positional Encoding



Transformer Layer

- Self Attention is Permutation Equivariant

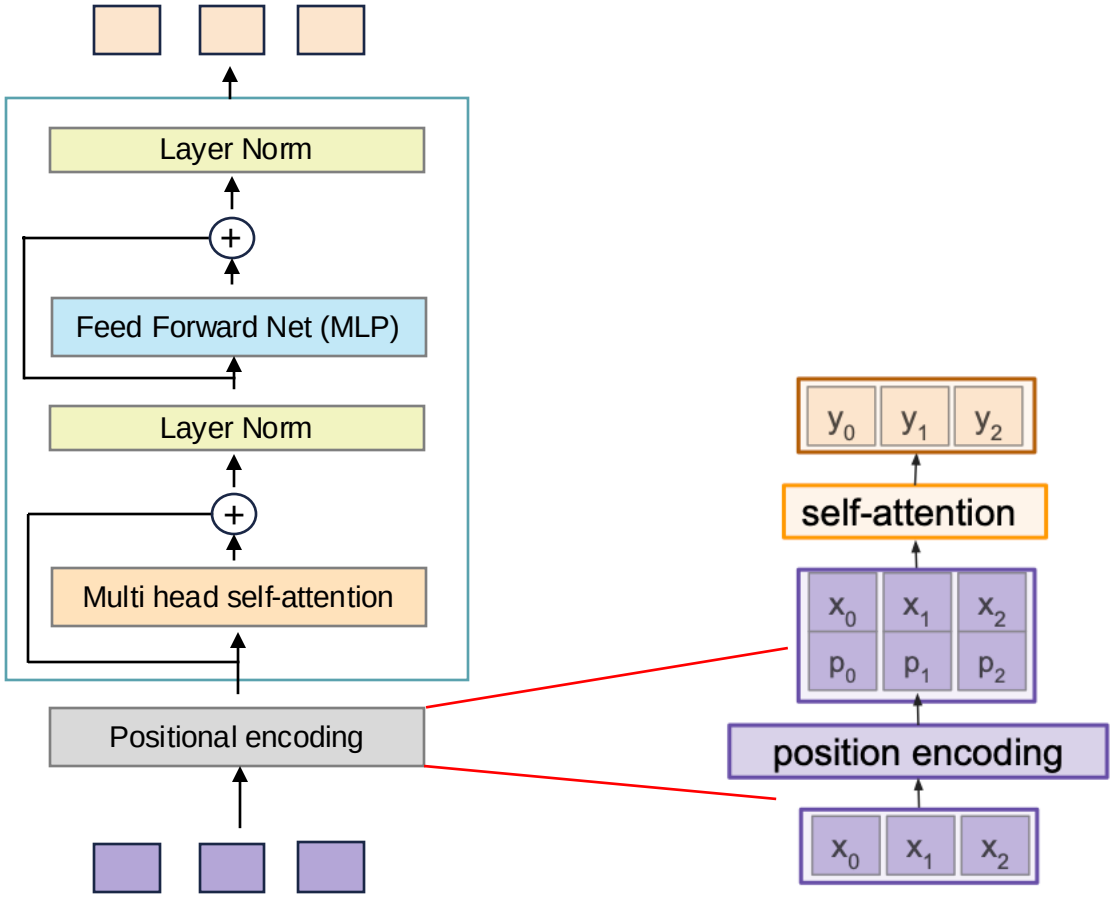
- How can we encode the order of tokens?
- How can we encode the underlying structure of the tokens?

- Concatenate positional vectors to input vectors

- **Positional encoding should be:**
  - Unique for each time-step
  - Distance preserving
  - Generalizable to longer sentences
  - Deterministic

Adapted from: cs231n Stanford

Positional Encoding



Transformer Layer

- Self Attention is Permutation Equivariant

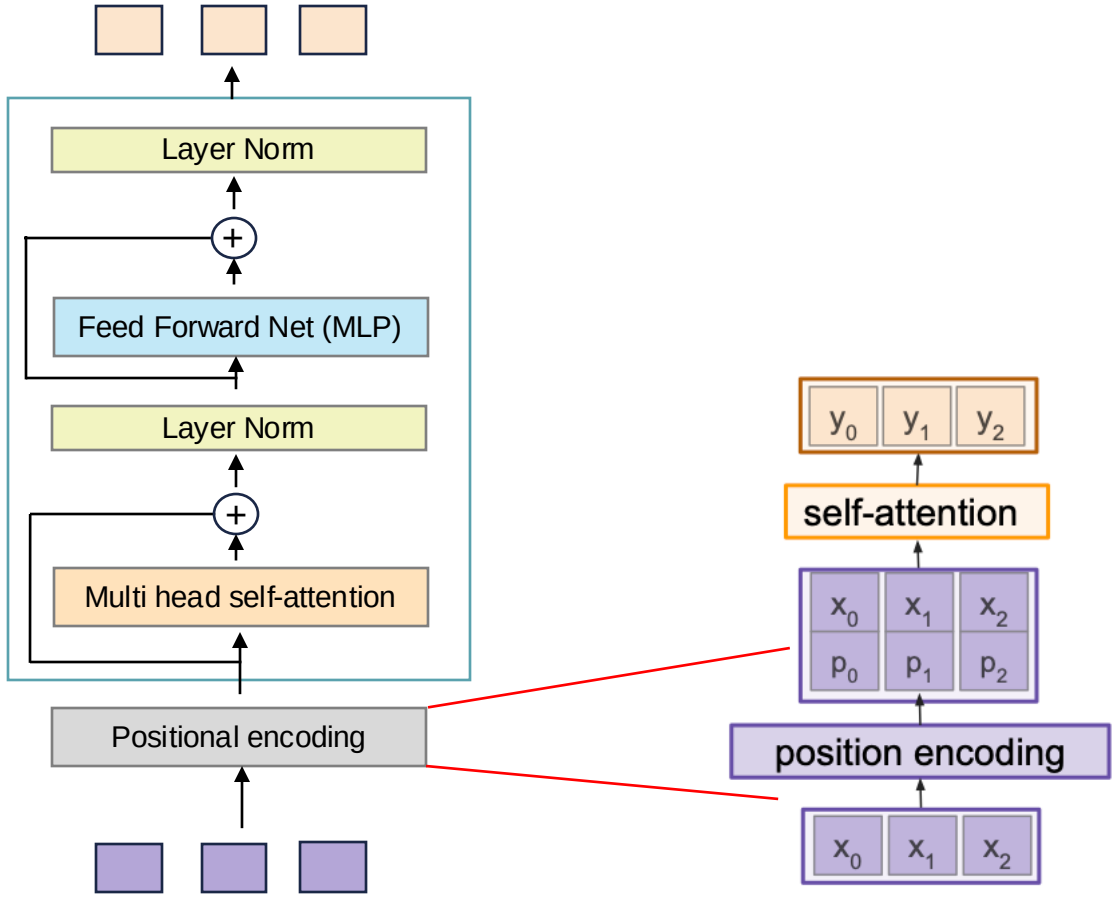
- How can we encode the order of tokens?
- How can we encode the underlying structure of the tokens?

- Concatenate positional vectors to input vectors

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

Adapted from: cs231n Stanford

Positional Encoding

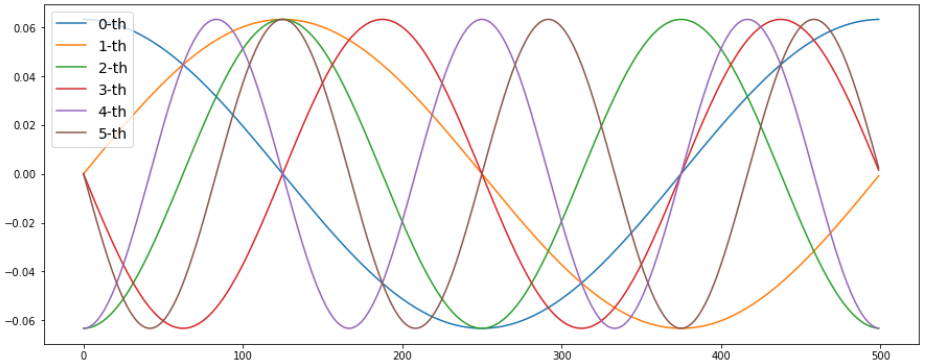


Transformer Layer

- Self Attention is Permutation Equivariant

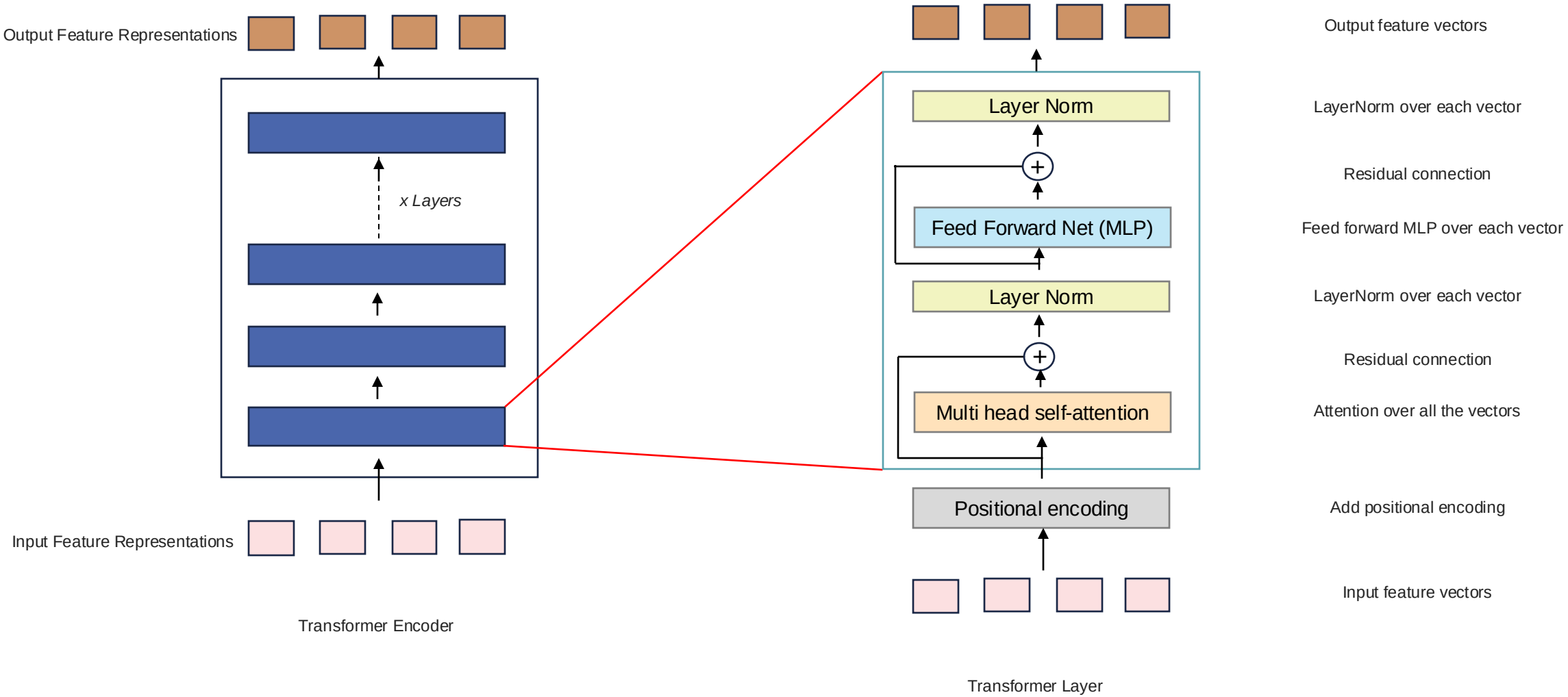
- How can we encode the order of tokens?
- How can we encode the underlying structure of the tokens?

- Concatenate positional vectors to input vectors



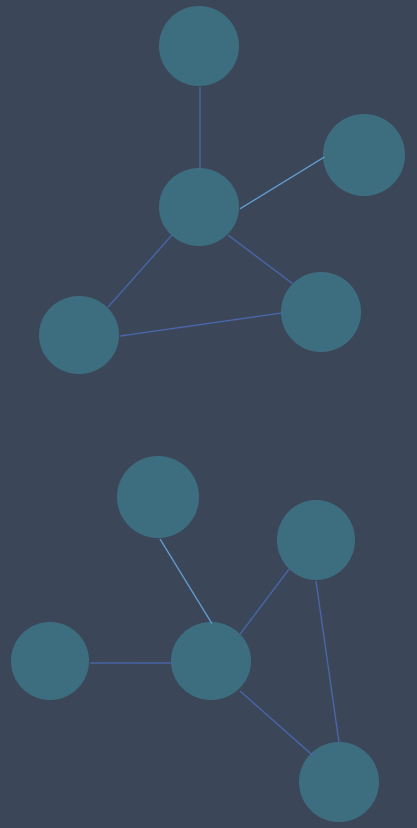
- an example sentence with 500 words

Transformer Encoder: Recap

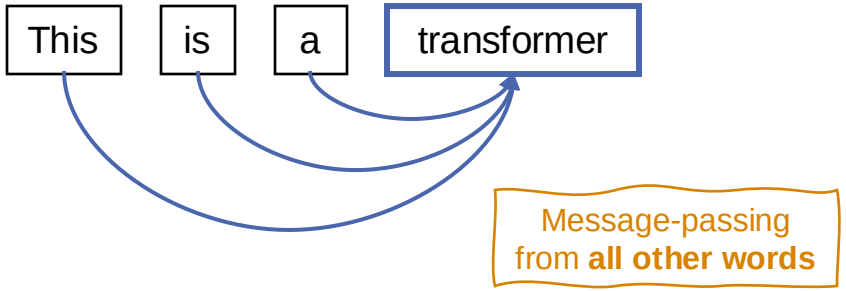


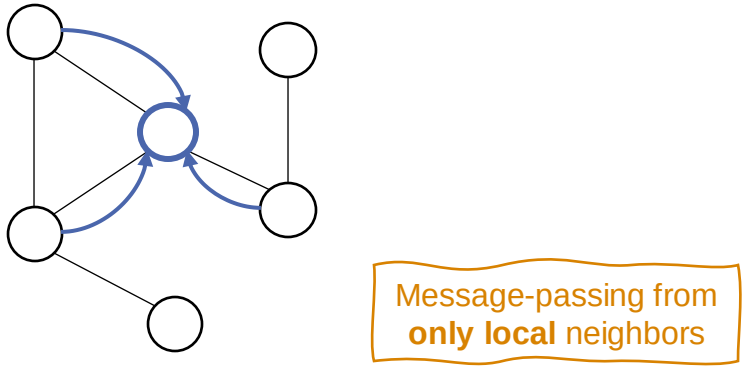
Outline of Presentation

- Transformers
- **Attention as Message Passing**
- Graph Transformers
- Bottlenecks and Remedies
- Recent Research



Attention as a case of message-passing





Graph Neural Network (or, MPNN) Pipeline

- A Graph Neural Network (GNN) is a NN model which takes in a graph with node and edge attributes and builds abstract representations of nodes and edges by taking the explicit connectivity structure into account.

Broad definition which covers several categories of GNN models

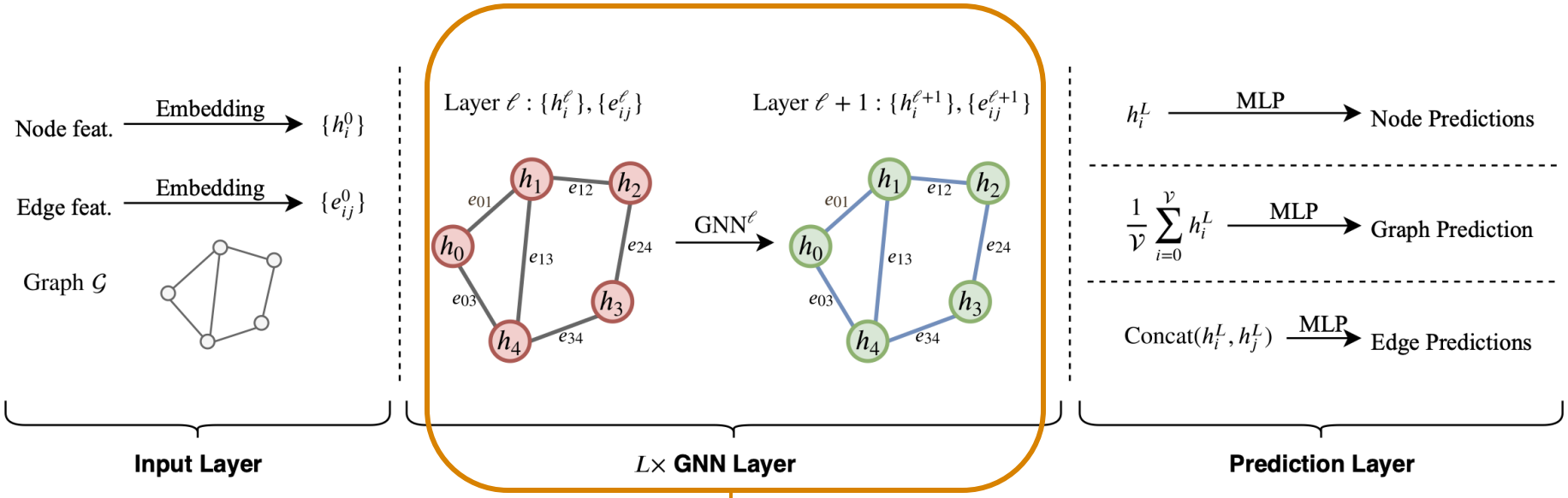
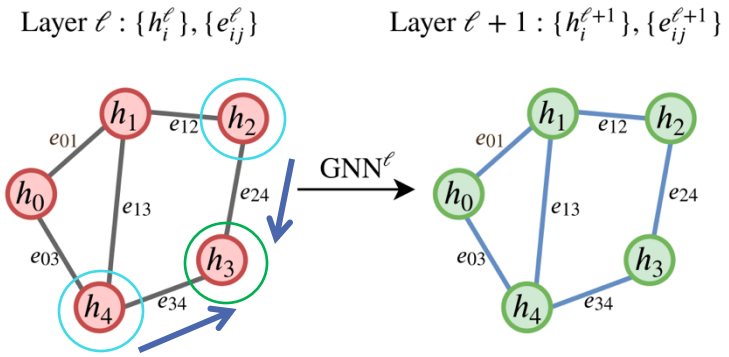


Fig: An end-to-end schematic diagram of a L layer GNN model, a category of GNNs which builds abstract representations of nodes and edges using the message-passing framework, *i.e.* nodes exchange messages with their first-hop neighbors at every layer

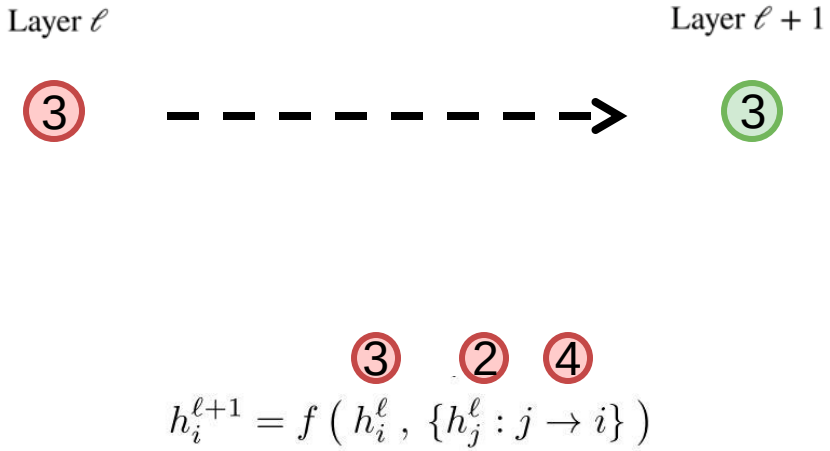
Message-Passing

Message-Passing

- Use of message-passing [1] to aggregate neighborhood information and update node features



$$h_i^{\ell+1} = f\left(h_i^\ell, \{h_j^\ell : j \to i\}\right)$$



E.g. GCN [2] $$h_i^{\ell+1} = \sigma\left(W^\ell \frac{1}{\sqrt{\deg_i}\sqrt{\deg_j}} \sum_{j \in \mathcal{N}_i} h_j^\ell\right)$$

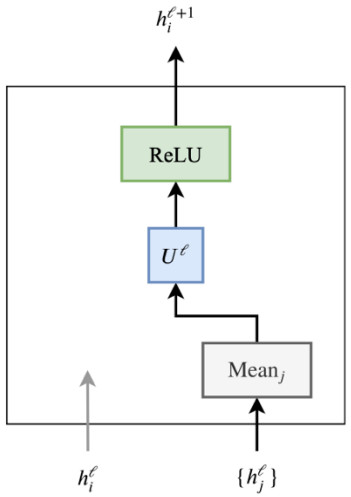
How to design the most powerful ***f*** function?
=> Active area of research!

[1] Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O. and Dahl, G.E. 2017. Neural message passing for quantum chemistry.
[2] Kipf, T. N., & Welling, M. 2017. Semi-supervised classification with graph convolutional networks.

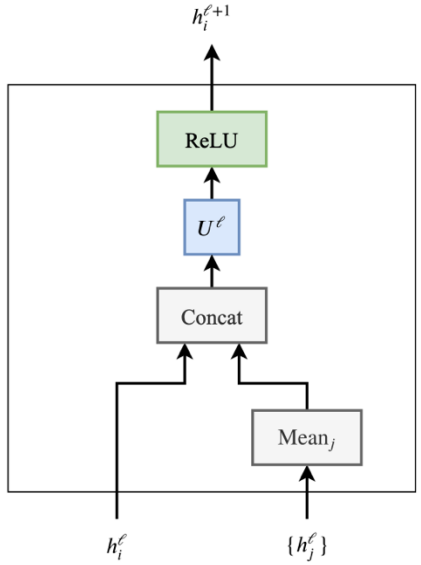
Different designs of GNNs

(based on Message-Passing)

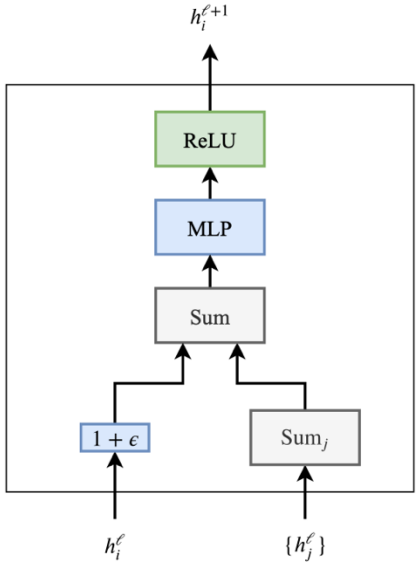
$$h_i^{\ell+1} = f\left(h_i^{\ell}, \{h_j^{\ell} : j \to i\}\right)$$



GCN Layer [1]



GraphSage Layer [2]



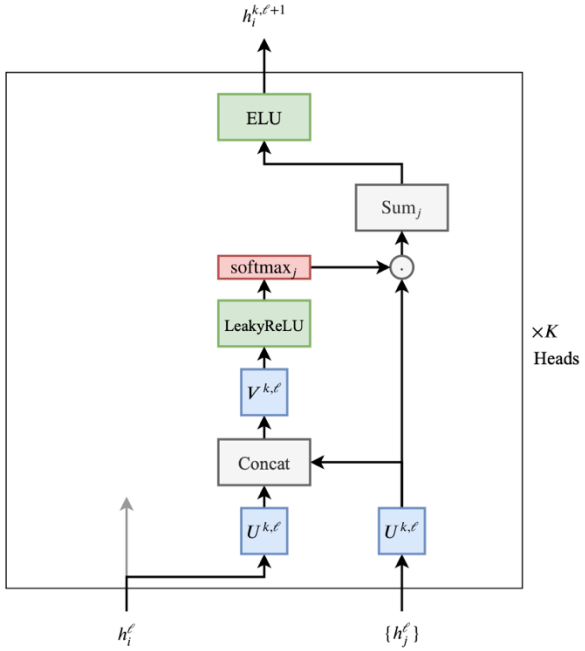
GIN Layer [3]

[1] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks.
[2] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs.
[3] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks?

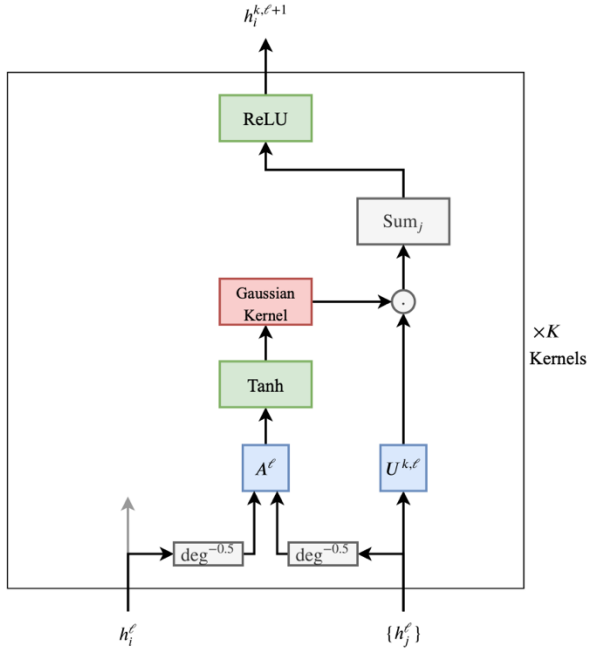
Different designs of GNNs

(based on Message-Passing)

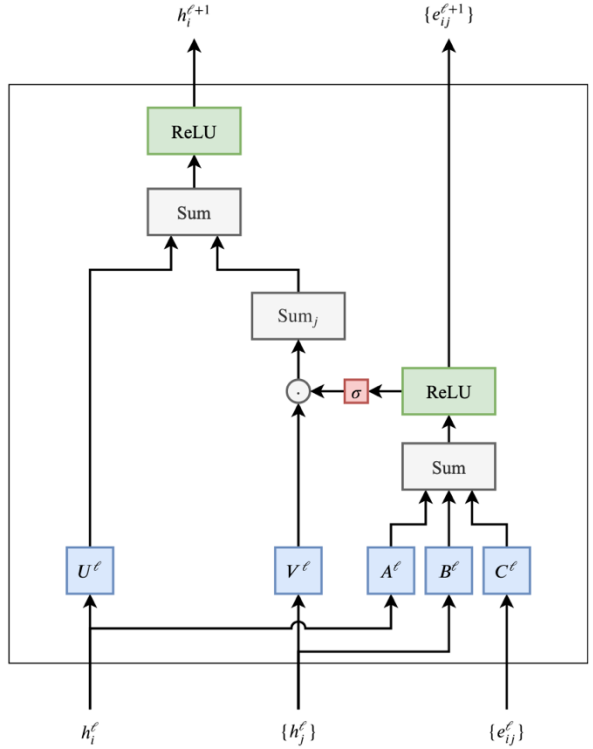
$$h_i^{\ell+1} = f\left(h_i^{\ell}, \{h_j^{\ell} : j \to i\}\right)$$



GAT Layer [1]



MoNet Layer [2]

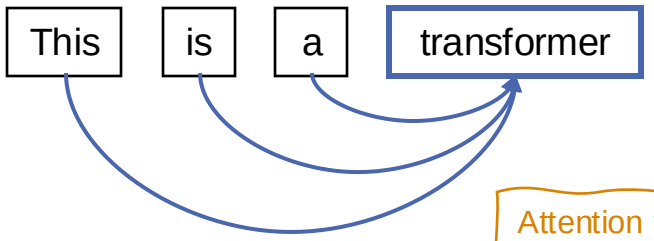


GatedGCN Layer [3]

[1] Petar Veli˘kovi´c, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks.
[2] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M. Bronstein. 2017. Geometric deep learning on graphs and manifolds using mixture model cnns.
[3] Xavier Bresson and Thomas Laurent. 2017. Residual gated graph convnets.

Attention in Transformer Neural Networks

- Use of **multi-head attention mechanism**.



multi-heads

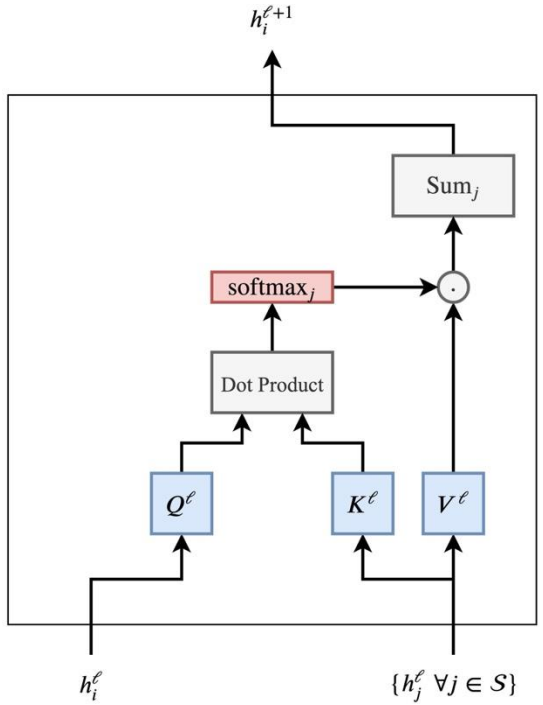
$$\hat{h}_i^{\ell+1} = O_h^{\ell} \Big\Vert_{k=1}^{H} \Big(\sum_{j \in \mathcal{S}} w_{ij}^{k,\ell} V^{k,\ell} h_j^{\ell} \Big),$$

word

other words in a sentence

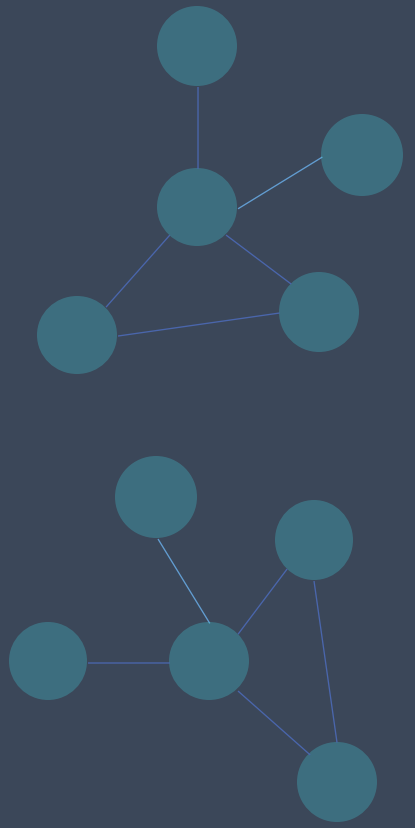
$$\text{where, } w_{ij}^{k,\ell} = \text{softmax}_j \Big(\frac{Q^{k,\ell} h_i^{\ell} \cdot K^{k,\ell} h_j^{\ell}}{\sqrt{d_k}} \Big),$$

attention score



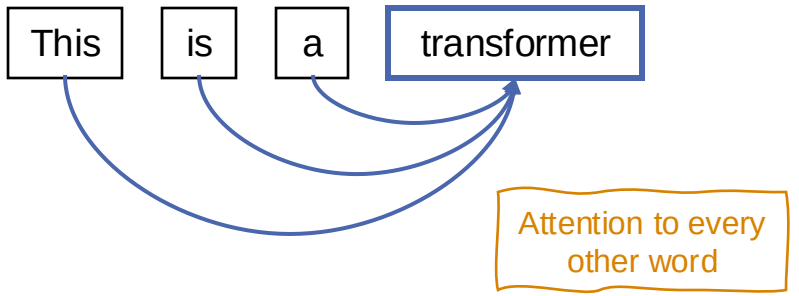
Outline of Presentation

- Transformers
- Attention as Message Passing
- **Graph Transformers**
- Bottlenecks and Remedies
- Recent Research

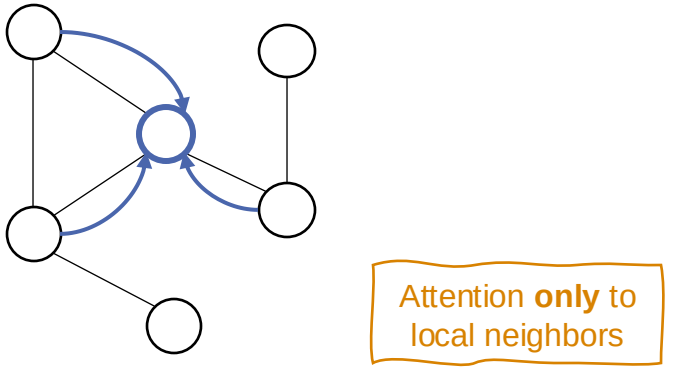


Generalization of Transformers to Graphs

- Generalize Transformers to arbitrary graphs, *i.e.*, design a **Graph Transformer [1].**
- How? 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 [2].



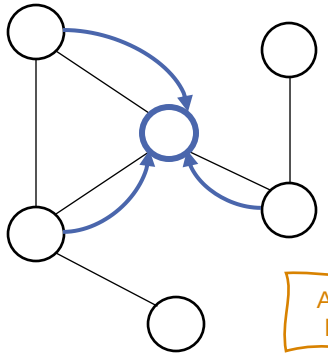
- **Open questions** on:
  - Local v/s global attention
  - Sparse structure v/s full graph
  - Positional encoding candidates
  - Scalability and pre-training

[1] Dwivedi, V.P. and Bresson, X. (2020). A generalization of transformer networks to graphs.
[2] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P. and Bengio, Y. (2018). Graph attention networks.

Graph Transformer

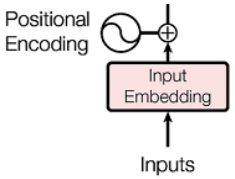
- **Proposed Idea**

1. Use Graph Sparsity **and** **Laplacian Positional Encodings (LapPE)*.**

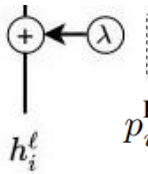


Attention **only** to local neighbors

full attention is expensive
(millions/billions of nodes)



PE in Original Transformer



Laplacian EigVecs as Positional Encoding

$$p_i^{\text{LapPE}} = [\ U_{i1}, U_{i2}, \cdots, U_{ik}\ ] \in \mathbb{R}^k$$

Take k (< n) smallest non-trivial eigenvectors

PE in Proposed Graph Transformer

Eigen-decomposition[1]

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

where A is the $n \times n$ adjacency matrix, D is the degree matrix, and Λ, U correspond to the eigenvalues and eigenvectors respectively.

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

[1] Belkin, M. and Niyogi, P., (2003). Laplacian eigenmaps for dimensionality reduction and data representation.

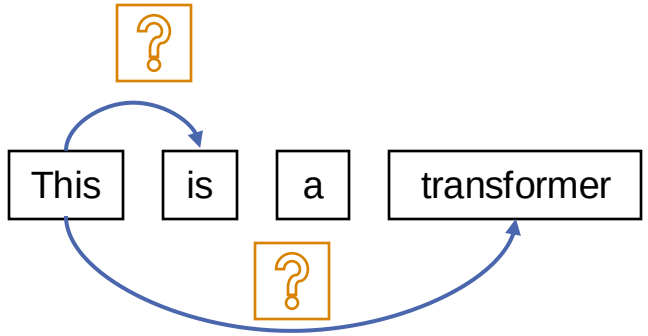
Graph Transformer

1. Graph Sparsity

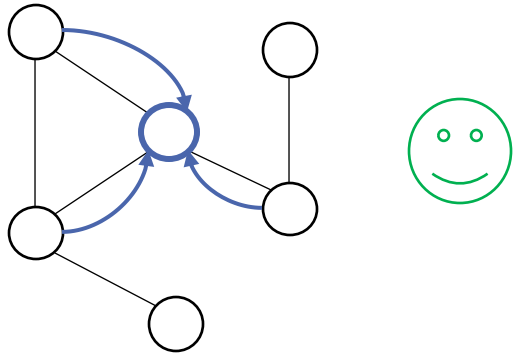
- NLP Transformers consider fully connected graph [1] and this choice can be justified for two reasons:
  - Difficult to find meaningful sparse connections between words, i.e. **absence of sparse structure**.
  - NLP Transformer sentences often contain less than tens or hundreds of words, hence **scalable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

- **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.



Sequence of words in NLP



Graph with arbitrary structure

[1] Joshi, C., (2020). Transformers are graph neural networks.

Graph Transformer

2. Positional Encodings (PE)

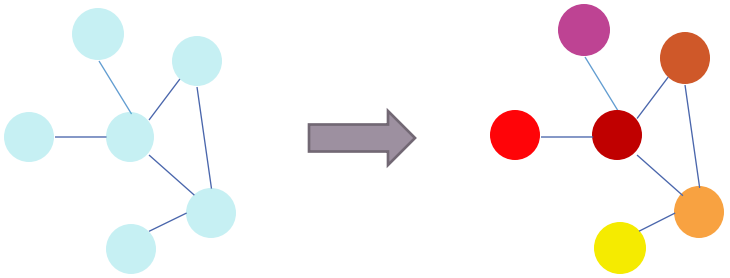
- NLP (Original) Transformers are supplied with a positional encoding for each word.

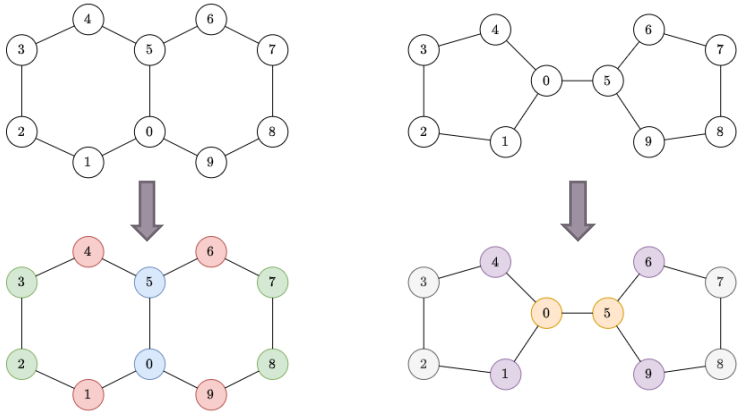
<Sequence>
This is a transformer

v/s

<Multi-set>
{This, is, a, transformer}

- CNNs[1] implicitly encode spatial position[2], RNNs build on sequences[3]
- What about **graphs** and how GNNs incorporate node positional information?





Decalin

Bicyclopentyl

[1] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition.
[2] Islam, M.A., Jia, S. and Bruce, N.D., 2020. How much position information do convolutional neural networks encode?
[3] Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory.
[4] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need.

Graph Transformer

2. Positional Encodings (PE)

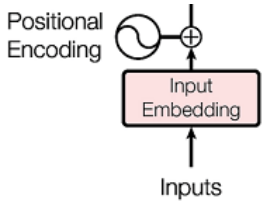
- NLP (Original) Transformers are supplied with a positional encoding for each word.

<Sequence>
This is a transformer

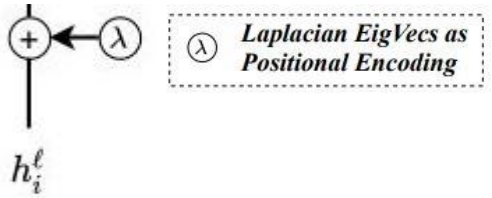
v/s

<Multi-set>
{This, is, a, transformer}

- Its natural to use a mechanism to encode node positions in Graph Transformers, as explored in recent research for Position integration in GNNs. [1,2,3]
- PEs in graph is challenging and hard to get canonical node position information.
- We use **Laplacian Eigenvectors** [4] corresponding to a node, as the node positional encoding – namely **Laplacian PE (λ)** [3], which generalize the sinusoidal PE (~) used in NLP Transformers.



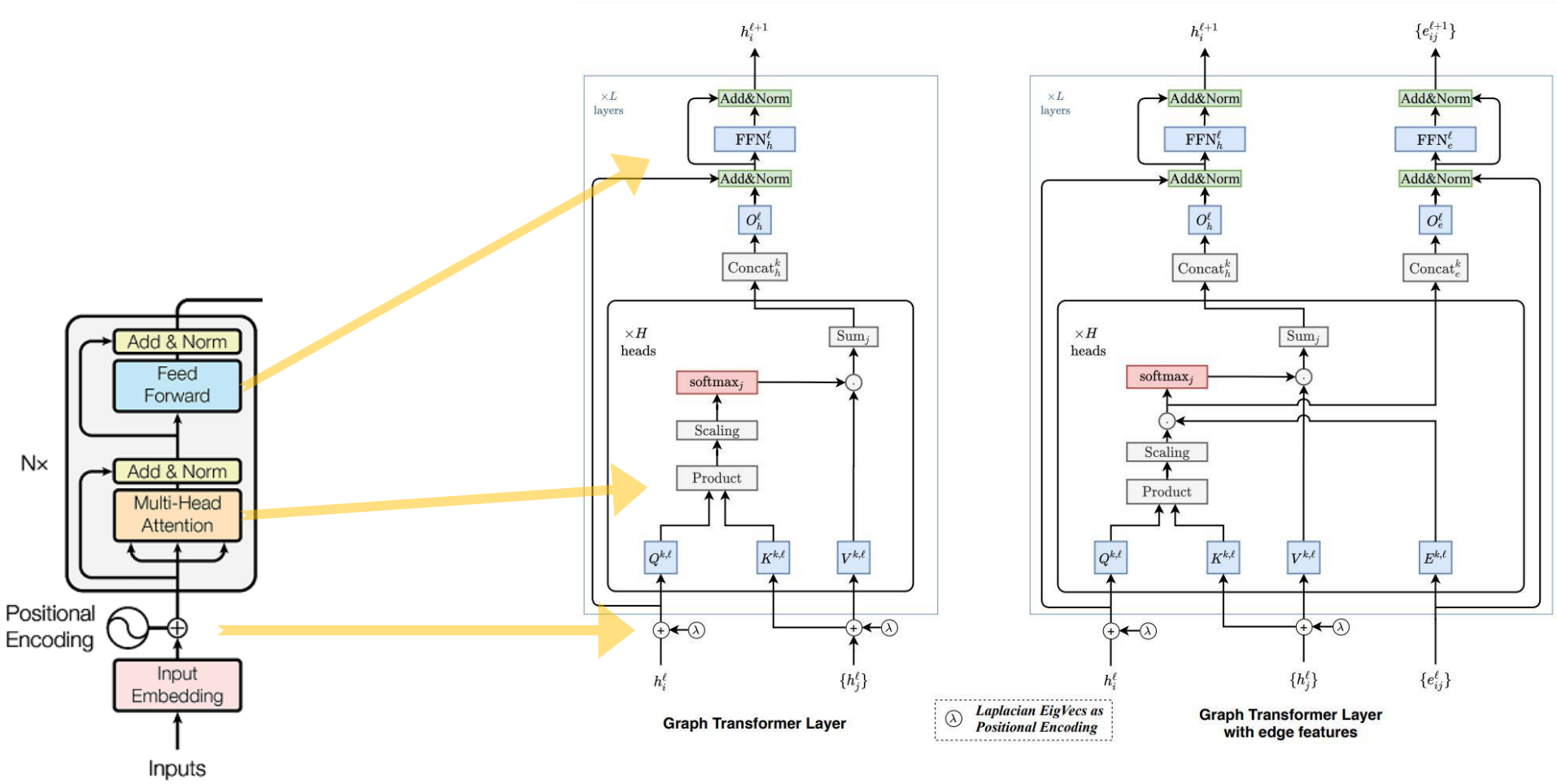
PE in Original Transformer



PE in Proposed Graph Transformer

[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.

Graph Transformer



Original Transformer (Encoder)

Proposed Graph Transformer

Graph Transformer

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

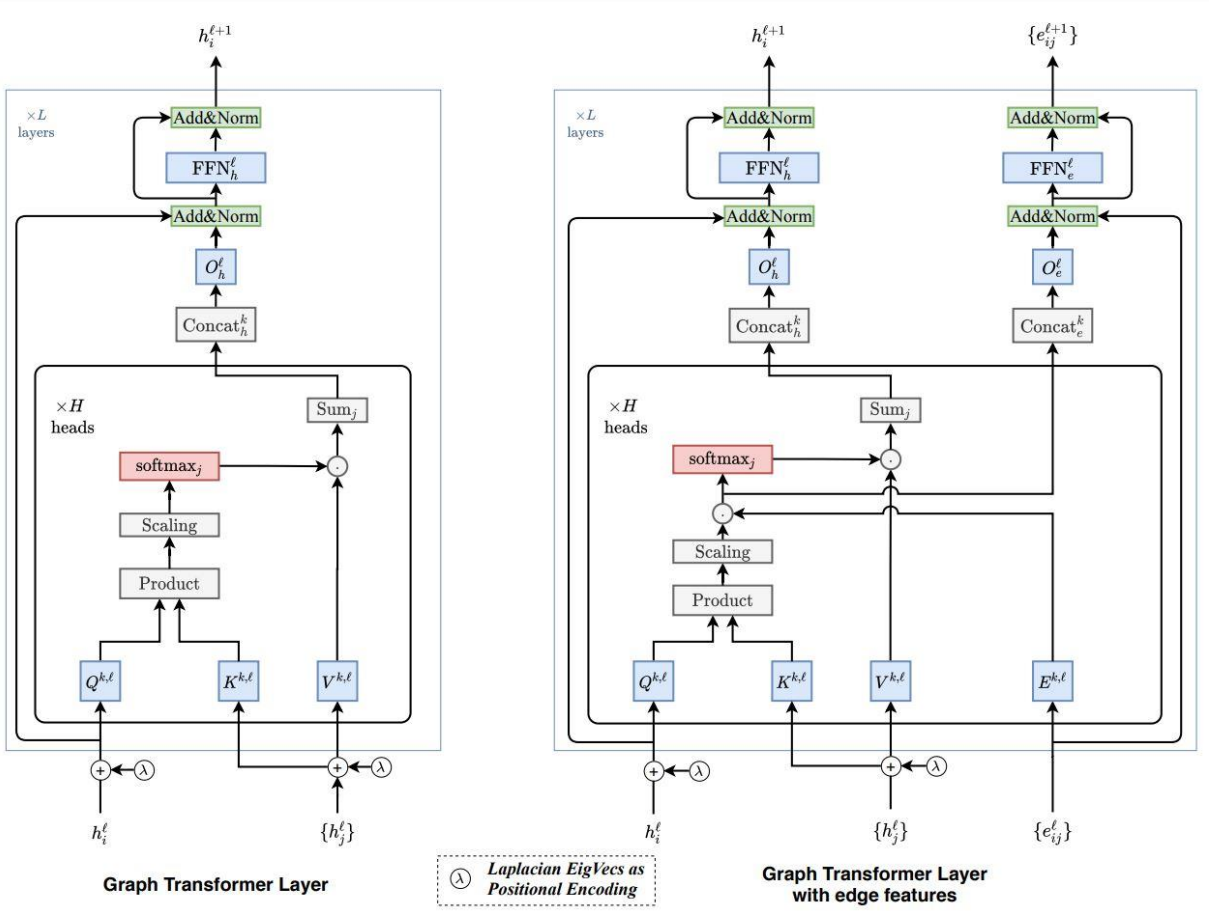
1. The ***attention*** mechanism is a function of ***neighborhood connectivity*** for each node.

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

Node

Node neighborhood

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



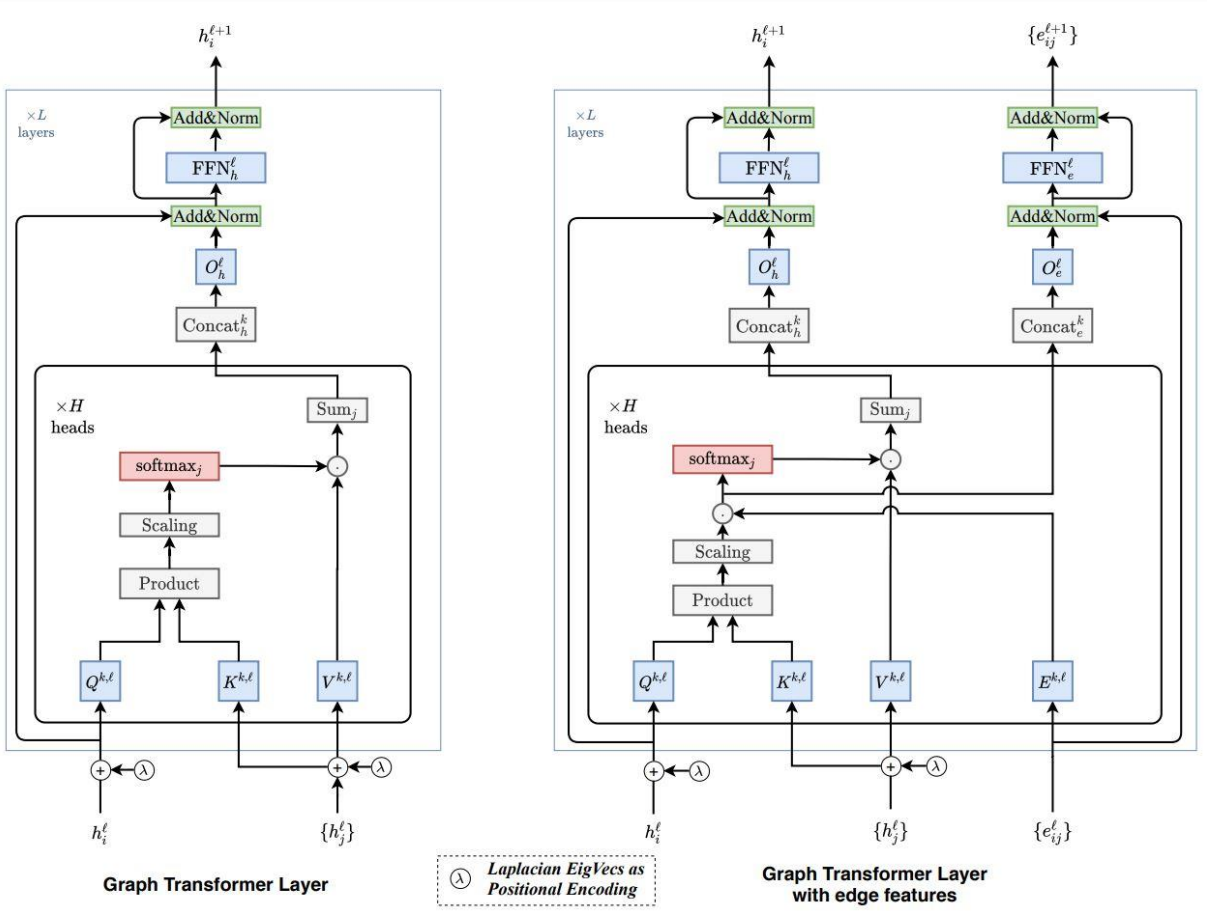
Proposed Graph Transformer

Graph Transformer

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

1. The ***attention*** mechanism is a function of ***neighborhood connectivity*** for each node.

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



Proposed Graph Transformer

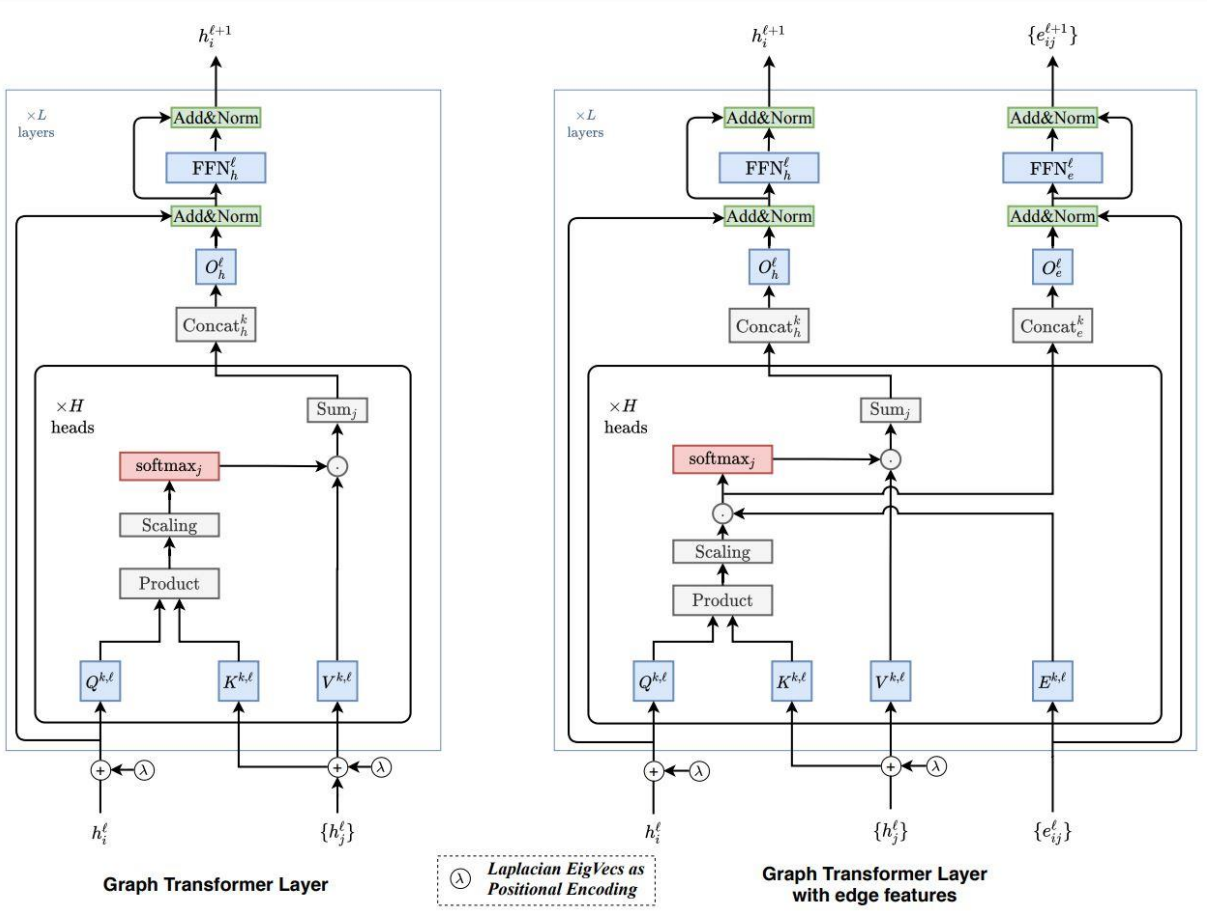
Graph Transformer

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

1. The ***attention*** mechanism is a function of ***neighborhood connectivity*** for each node.

2. Positional encoding is represented by ***Laplacian PE***.

3. The layer normalization is replaced by a ***batch normalization layer***.



Proposed Graph Transformer

Graph Transformer

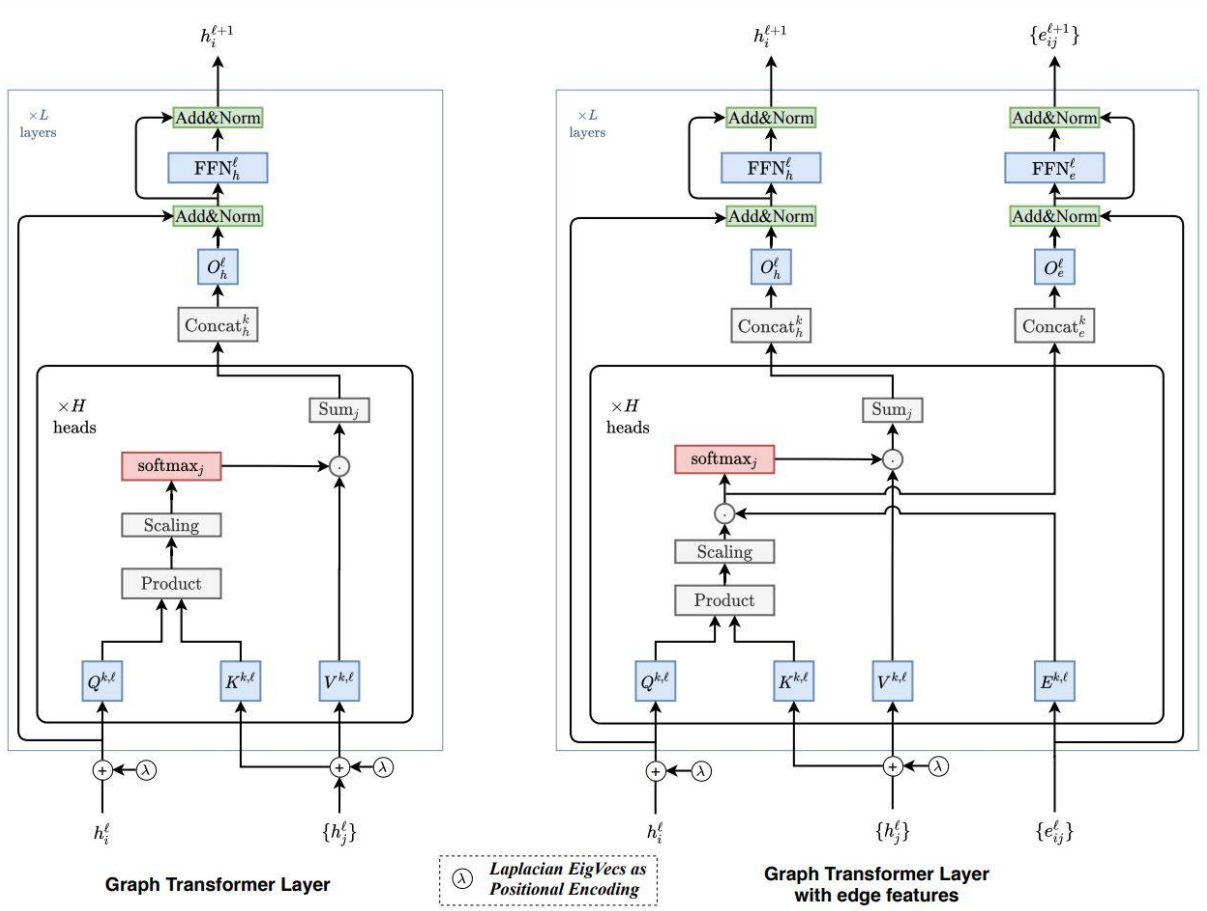
Compared to the original transformer, the **highlights** of the proposed architecture are:

1. The ***attention*** mechanism is a function of ***neighborhood connectivity*** for each node.

2. Positional encoding is represented by ***Laplacian PE***.

3. The layer normalization is replaced by a ***batch normalization layer***.

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



Proposed Graph Transformer

Graph Transformer

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

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

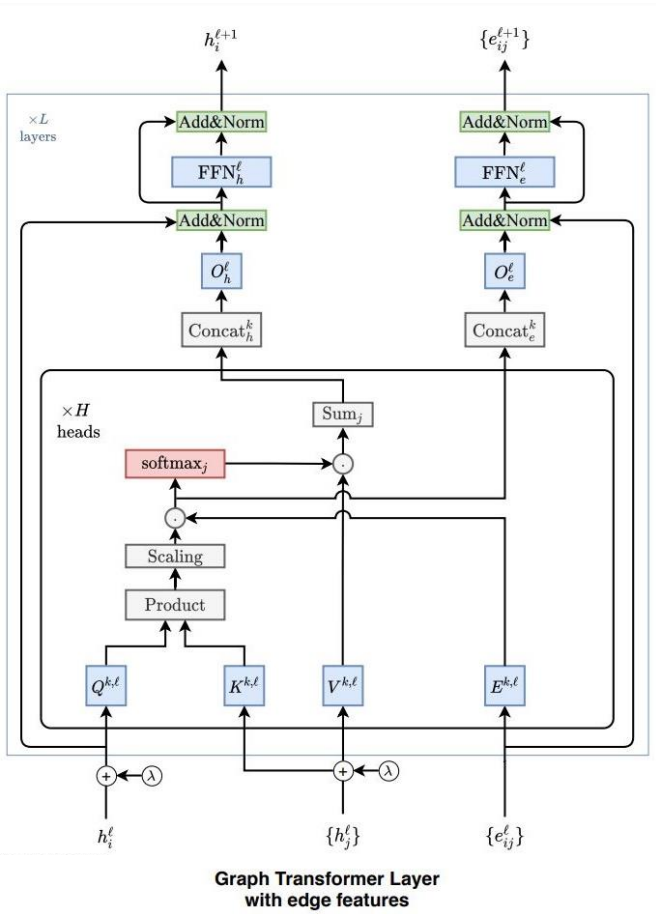
$$\hat{h}_i^{\ell+1} = O_h^\ell \Big\Vert_{k=1}^{H} \Big(\sum_{j \in \mathcal{N}_i} w_{ij}^{k,\ell} V^{k,\ell} h_j^\ell \Big),$$

$$\hat{e}_{ij}^{\ell+1} = O_e^\ell \Big\Vert_{k=1}^{H} \Big(\hat{w}_{ij}^{k,\ell} \Big), \quad \text{where,}$$

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

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

Available edge information



Graph Transformer Layer with edge features

Graph Transformer

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 2: Comparison of our best performing scores (from Table 1) on each dataset against the GNN baselines (GCN (Kipf and Welling 2017), GAT (Veličković et al. 2018), GatedGCN(Bresson and Laurent 2017)) of 500k model parameters. **Note:** Only GatedGCN and GT models use the available edge attributes in ZINC.

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

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.

Positional Encodings for Graphs

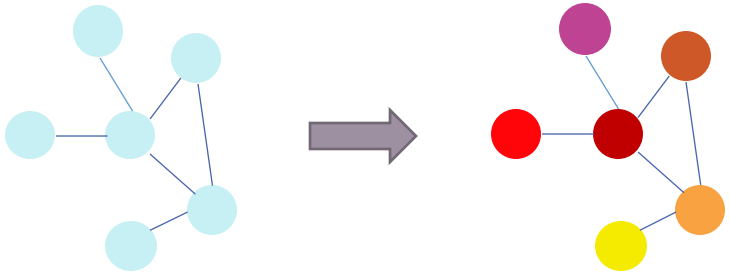
1. Positional Encodings (PEs) in graph is challenging and hard to get canonical node position information.

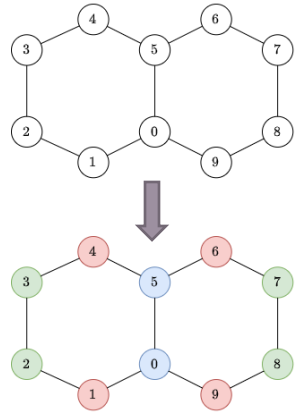
2. Message passing based GNNs can be made more expressive [1,2] and universal approximators with unique node identifiers [3]

3. PEs can inform GNNs (1-WL equivalent) about global structural information.

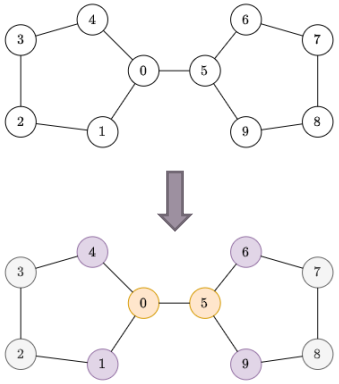
Questions –

1. How does Laplacian PEs help GNNs?
2. Are there other methods to extract positional encoding?





Decalin



Bicyclopentyl

[1] You, J., Ying, R. and Leskovec, J., 2019, May. Position-aware graph neural networks.
[2] Murphy, R., Srinivasan, B., Rao, V. and Ribeiro, B., 2019, May. Relational pooling for graph representations.
[3] Loukas, A., 2020. What graph neural networks cannot learn: depth vs width.

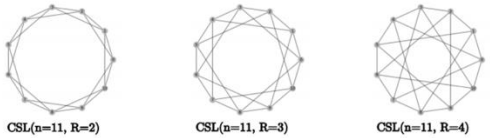
Positional Encodings for Graphs

- **Experimental Results**

1. **Any MPNN** augmented with LapPE **improves** performance against their non-PE counterparts.
2. Best performing in CSL, CYCLES, GraphTheoryProp datasets.

Model	L	#Param	Test Accuracy Mean±s.d.	Test Accuracy Max	Test Accuracy Min	Train Accuracy Mean±s.d.	Train Accuracy Max	Train Accuracy Min	#Epoch	Epoch/ Total
Laplacian PE augmented MP-GCNs: Node Positional Encoding with Laplacian Eigenvectors										
MLP	4	101235	22.567±6.089	46.667	10.000	30.389±5.712	43.333	10.000	109.39	0.16s/0.03hr
GCN	4	103847	**100.000±0.000**	100.000	100.000	100.000±0.000	100.000	100.000	125.64	0.40s/0.07hr
GraphSage	4	105867	99.933±0.467	100.000	96.667	100.000±0.000	100.000	100.000	155.00	0.50s/0.11hr
MoNet	4	105579	99.967±0.332	100.000	96.667	100.000±0.000	100.000	100.000	130.39	0.49s/0.09hr
GAT	4	101710	99.933±0.467	100.000	96.667	100.000±0.000	100.000	100.000	133.18	0.61s/0.12hr
GatedGCN	4	105407	99.600±1.083	100.000	96.667	100.000±0.000	100.000	100.000	147.06	0.66s/0.14hr
GIN	4	107304	99.333±1.333	100.000	96.667	100.000±0.000	100.000	100.000	62.98	0.44s/0.04hr
No Node Positional Encoding										
All MP-GCNs	4	100K	10.000±0.000	10.000	10.000	10.000±0.000	10.000	10.000	-	-

CSL Dataset



Train samples →	200	500	1000	5000
Model / L / #Param	Test Acc±s.d.			
From SMP paper, Vignac et al. [46]				
SMP 8 -	87.700±N.A.	**97.400±N.A.**	97.600±N.A.	99.500±N.A.
Our experiments				
GIN-NoPE 4 100774	70.718±0.423	75.507±0.229	74.950±2.187	85.787±1.922
GIN-LapPE 4 102864	76.097±9.561	94.295±0.648	96.790±0.269	99.380±0.112
GatedGCN-NoPE 4 103933	50.000±0.000	50.000±0.000	50.000±0.000	50.000±0.000
GatedGCN-LapPE 4 105263	**94.850±0.221**	96.712±0.156	**98.778±0.086**	**99.740±0.027**

CYCLES Dataset

Model	L	Test Average	Test Dist.	Test Ecc.	Test Lap.	Test Conn.	Test Diam.	Test Rad.
From SMP paper, Vignac et al. [46]								
PNA	-	-3.13	-2.89	-2.89	-3.77	-2.61	-3.04	-3.57
Fast SMP	-	-3.53	-3.31	-3.36	-4.30	-2.72	-3.65	-3.82
SMP	-	-3.59	-3.59	-3.67	-4.27	-2.97	-3.58	-3.46
Our experiments								
GIN-NoPE	8	-3.19±0.11	-2.81±0.11	-2.42±0.09	-4.39±0.18	-2.07±0.13	-3.06±0.11	-4.39±0.13
GIN-LapPE	8	-3.21±0.13	-2.87±0.03	-2.83±0.07	-3.99±0.04	-2.00±0.15	-3.27±0.07	-4.31±0.15
GatedGCN-NoPE	8	-3.22±0.13	-2.76±0.17	-2.36±0.12	-3.92±0.15	-2.65±0.11	-3.35±0.16	-4.31±0.08
GatedGCN-LapPE	8	-3.51±0.11	-3.23±0.08	-3.35±0.08	-4.03±0.21	-2.60±0.12	-3.57±0.05	-4.32±0.13
PNA-NoPE	8	-3.38±0.02	-2.91±0.03	-2.52±0.04	-4.07±0.02	-2.99±0.07	-3.28±0.05	-4.51±0.03
PNA-LapPE	8	**-3.63±0.04**	-3.18±0.02	-3.30±0.06	-4.18±0.04	-2.88±0.08	-3.60±0.03	-4.65±0.05

GraphTheoryProp Dataset

Other Positional Encodings

- LapPE has limitations due to being based on eigenvectors
- When selecting k (<<n) eigenvectors, the number of possibilities is 2^k due to sign ambiguity
- One way to address is to use random sign flipping during training which we adopt
- Other recent works use SignNet[1] and follow-ups.

	PE type	L	#Param	Test Acc.±s.d.	Train Acc.±s.d.	#Epochs	Epoch/Total
CSL	No PE	4	104007	10.000±0.000	10.000±0.000	54.00	0.58s/0.05hr
	EigVecs-20	4	105407	68.633±7.143	99.811±0.232	107.16	0.59s/0.09hr
	Rand sign(EigVecs)	4	105407	**99.767±0.394**	99.689±0.550	188.76	0.59s/0.16hr
	Abs(EigVecs)	4	105407	99.433±1.133	100.000±0.000	143.64	0.60s/0.12hr
	Fixed node ordering	4	106807	10.533±4.469	76.056±14.136	60.56	0.59s/0.05hr
	Rand node ordering	4	106807	11.133±2.571	10.944±2.106	91.60	0.60s/0.08hr
PATTERN	No PE	16	502223	85.605±0.105	85.999±0.145	62.00	646.03s/11.36hr
	EigVecs-2	16	505421	86.029±0.085	86.955±0.227	65.00	645.36s/11.94hr
	Rand sign(EigVecs)	16	502457	**86.508±0.085**	86.801±0.133	65.75	647.94s/12.08hr
	Abs(EigVecs)	16	505421	86.393±0.037	87.011±0.172	62.00	645.90s/11.41hr
	Fixed node ordering	16	516887	80.133±0.202	98.416±0.141	45.00	643.23s/8.27hr
	Rand node ordering	16	516887	85.767±0.044	85.998±0.063	64.50	645.09s/11.79hr
CLUSTER	No PE	16	502615	73.684±0.348	88.356±1.577	61.50	399.44s/6.97hr
	EigVecs-20	16	504253	75.520±0.395	89.332±1.297	49.75	400.50s/5.70hr
	Rand sign(EigVecs)	16	504253	**76.082±0.196**	88.919±0.720	57.75	399.66s/6.58hr
	Abs(EigVecs)	16	504253	73.796±0.234	91.125±1.248	58.75	398.97s/6.68hr
	Fixed node ordering	16	517435	69.232±0.265	92.298±0.712	51.00	400.40s/5.82hr
	Rand node ordering	16	517435	74.656±0.314	82.940±1.718	61.00	397.75s/6.88hr
COLLAB	No PE	3	40965	52.635±1.168	96.103±1.876	95.00	453.47s/12.09hr
	EigVecs-20	3	41889	52.326±0.678	96.700±1.296	95.00	452.40s/12.10hr
	Rand sign(EigVecs)	3	41889	**52.849±1.345**	96.165±0.453	94.75	452.75s/12.08hr
	Abs(EigVecs)	3	41889	51.419±1.109	95.984±1.157	95.00	451.36s/12.07hr
	PE type	L	**#Param**	**Test MAE±s.d.**	**Train MAE±s.d.**	**#Epochs**	**Epoch/Total**
ZINC	No PE	16	504153	0.354±0.012	0.095±0.012	165.25	10.52s/0.49hr
	EigVecs-8	16	505011	0.319±0.010	0.038±0.007	143.25	10.62s/0.43hr
	Rand sign(EigVecs)	16	505011	**0.214±0.013**	0.067±0.019	185.00	10.70s/0.56hr
	Abs(EigVecs)	16	505011	**0.214±0.009**	0.035±0.011	167.50	10.61s/0.50hr
	Fixed node ordering	16	507195	0.431±0.007	0.044±0.009	118.25	10.62s/0.35hr
	Rand node ordering	16	507195	0.321±0.015	0.177±0.015	184.75	10.55s/0.55hr

TABLE 3.13: Study of positional encodings (PEs) with the GatedGCN model [1]. Performance reported on the test sets of CSL, ZINC, PATTERN, CLUSTER and COLLAB (higher is better, except for ZINC). **Red**: the best model.

[1] Lim, D., Robinson, J., Zhao, L., Smidt, T., Sra, S., Maron, H. and Jegelka, S., 2022. Sign and basis invariant networks for spectral graph representation learning.

Other Positional Encodings

- We propose **RWPE** based on the Random Walk diffusion process*
- Take RW_{ii} from every k^{th} step to construct RWPE of node I

$$p_i^{\text{RWPE}} = \left[\text{RW}_{ii}, \text{RW}_{ii}^2, \cdots, \text{RW}_{ii}^k \right] \in \mathbb{R}^k$$

- Thus, RWPE **encodes** the landing probability of a node to itself in 1 to k steps of random walk

Meaningful higher-order structure information!

- RWPE closely based on Distance Encoding (DE) [1] but we do not consider all relative RW_{ij} as in DE

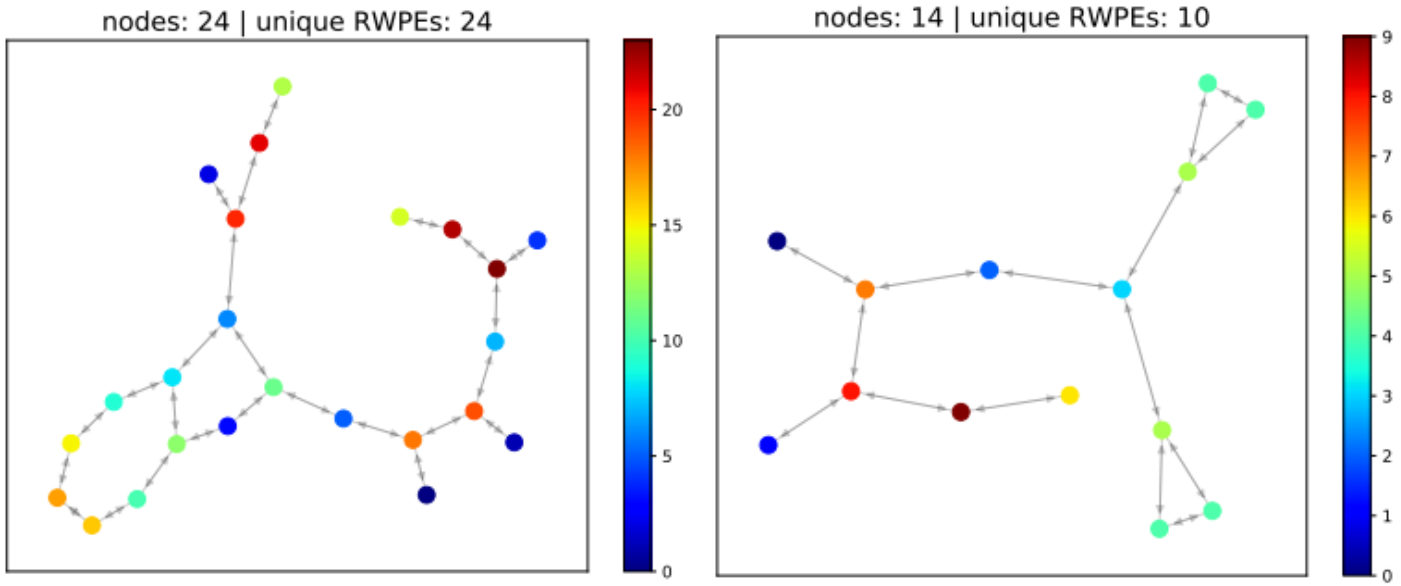
[*] Dwivedi, V.P., Luu, A.T., Laurent, T., Bengio, Y. and Bresson, X., 2022. Graph neural networks with learnable structural and positional representations.

[1] Li, P., Wang, Y., Wang, H. and Leskovec, J., 2020. Distance Encoding--Design Provably More Powerful GNNs for Structural Representation Learning.

Other Positional Encodings

Characteristics of RWPE

- Unique representation for a node “given a distinct k-hop” neighborhood when considering a sufficient k



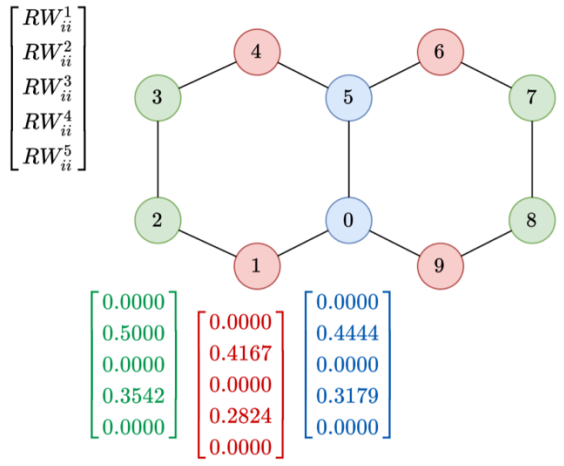
(a) ZINC molecule (val index 91)

(b) ZINC molecule (val index 212)

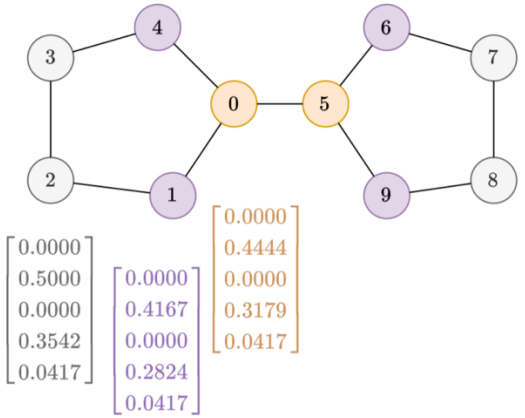
Other Positional Encodings

Characteristics of RWPE

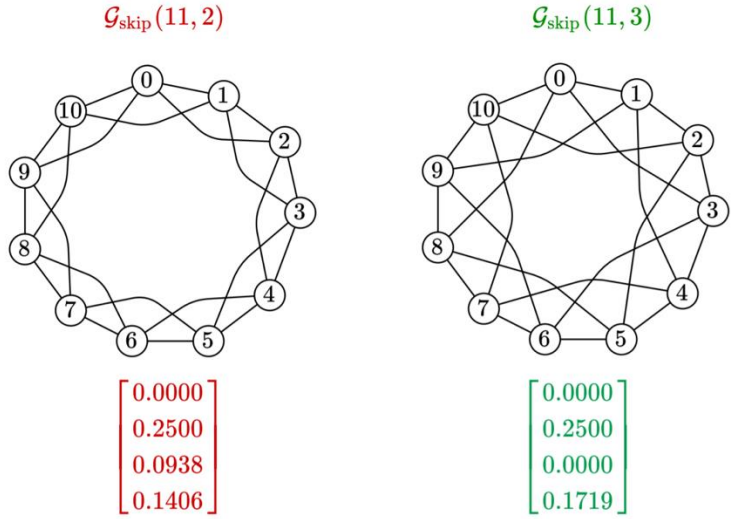
- No sign ambiguity as in LapPE
- Can distinguish graph-pairs that failed to be classified correctly by MP-GNNs (1-WL)



Decalin



Bicyclopentyl



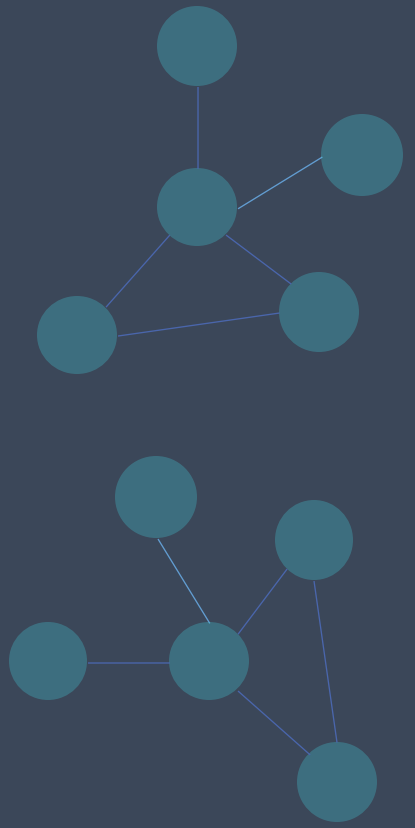
A pair of Circular Skip Link Graphs

Takeaways so far ...

- Transformers can be generalized for learning on graph structured datasets.
- Positional Encodings / Structural Encodings are key design components.
- There exists multiple ways to inject positional/structural information from the original graph structure, and several works aim to address respective limitations.

Outline of Presentation

- Transformers
- Attention as Message Passing
- Graph Transformers
- **Bottlenecks and Remedies**
- Recent Research



Bottleneck of GNNs

- Computational graph of a GCN or any MPNN

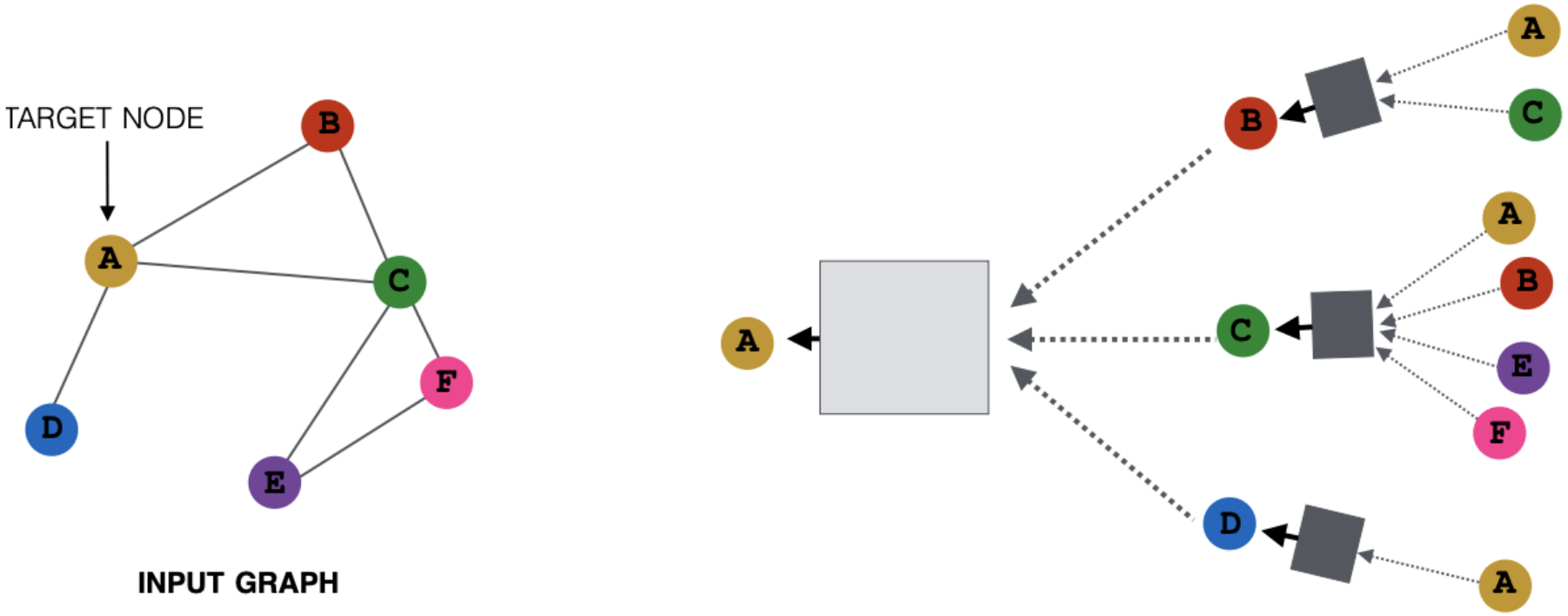


Fig from Stanford cs224w

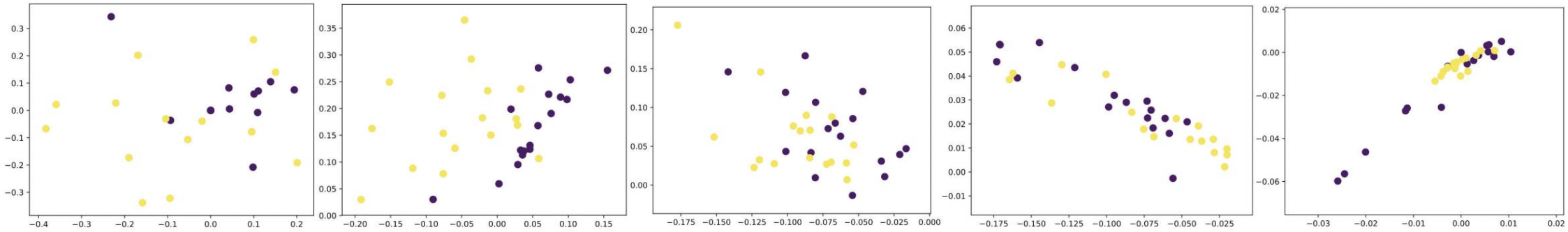
Bottleneck of GNNs

- **Smoothing:**
  - Graph Convolution in the GCN is a form of Laplacian smoothing
  - Smoothing makes the features of nodes in the same cluster similar

$$h^{\ell+1}_{n\times d} = D^{-1/2}AD^{-1/2}h^{\ell}W^{\ell}$$

$$h^{\ell+1} = \sigma\left(D^{-1/2}AD^{-1/2}h^{\ell}W^{\ell}\right)$$

- **Over-smoothing:**
  - If a GCN is deep, the features may be over-smoothed and embeddings converge to a subspace



(a) 1-layer (b) 2-layer (c) 3-layer (d) 4-layer (e) 5-layer

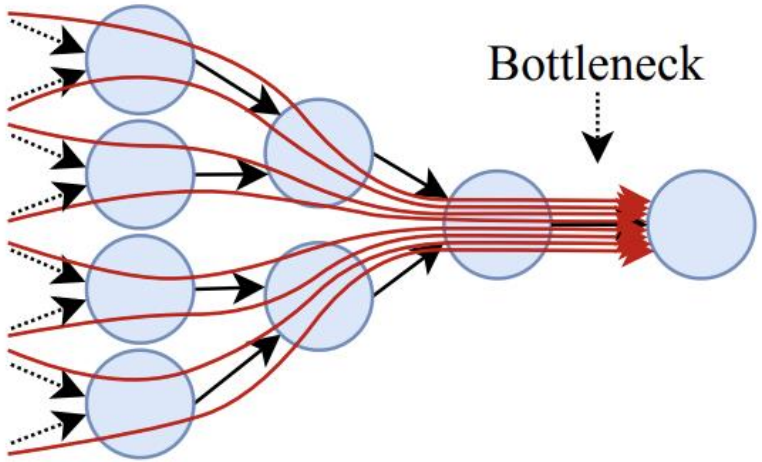
Figure 2: Vertex embeddings of Zachary's karate club network with GCNs with 1,2,3,4,5 layers.

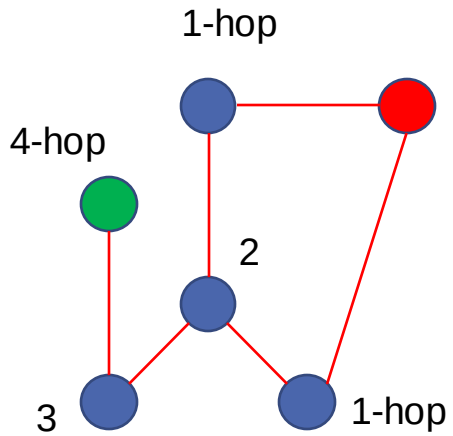
- Solution: Towards the exploitation of global graph structure

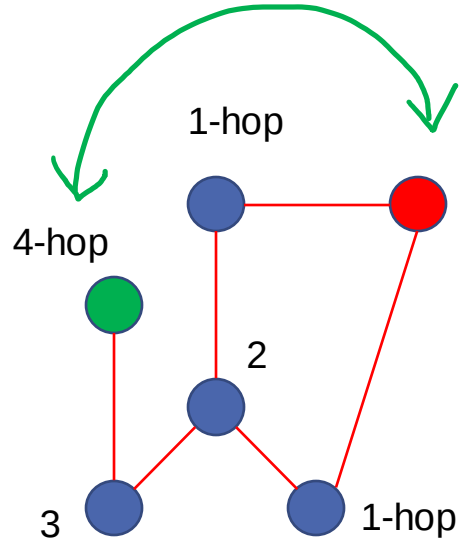
[1] Li, Q., Han, Z. and Wu, X.M., 2018, April. Deeper insights into graph convolutional networks for semi-supervised learning.

Bottleneck of GNNs

- **Oversquashing and Information Bottleneck [1]**
  - For long-range tasks, we require several GNN layers
  - As GNN layers increase, the receptive field of target node grows exponentially
  - Exponential amount of messages are required to be squashed into a single feature vector
  - Consequently, it even prevents fitting the training data







- Solution: Use a Fully Connected Graph (FCG) at the final layer

[1] Alon, U. and Yahav, E., 2020. On the bottleneck of graph neural networks and its practical implications.

Other Graph Transformer Architectures

- Fully connected [$O(N^2)$] Graph Transformers developed which perform better thanks to **PE-focused** innovations!

Spectral Attention Networks (SAN) [1]

- SAN use a Learnable PE module that applies a Transformer encoder on a sequence of eigenvalues/vectors to generate a fixed sized PE
- During full-attention in the main Transformer, separate learnable parameters are maintained for real and non-real edges

Injecting PE at node inputs

GraphiT [2]

- Use of diffusion geometry to capture short and long-range information
- The diffusion distance is multiplied with attention score

Injecting PE with attention score

Graphormer [3]

- Use of centrality, spatial and edge encoding to improve node inputs and attention.

Injecting PE with node inputs as well as attention score

Note: Other previous works such as Graph-BERT [4] use a combination of structural and relative encodings after a link-less subgraph batching

[1] Kreuzer, D., Beaini, D., Hamilton, W.L., Létourneau, V. and Tossou, P., 2021. Rethinking Graph Transformers with Spectral Attention.
[2] Mialon, G., Chen, D., Selosse, M. and Mairal, J., 2021. GraphiT: Encoding Graph Structure in Transformers.
[3] 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?
[4] Zhang, J., Zhang, H., Xia, C. and Sun, L., 2020. Graph-bert: Only attention is needed for learning graph representations.

Takeaways so far ...

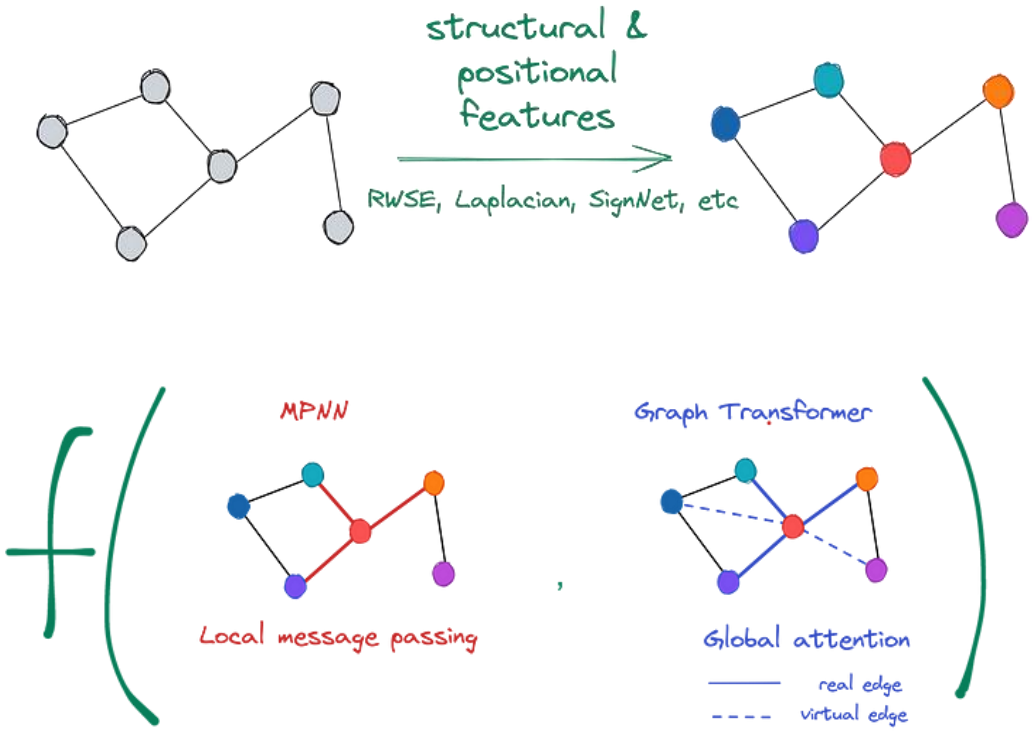
- Research on Graph Transformers with both local graph aggregation and global attention.
- Local graph representation learning is bottlenecked by local message passing.
- Global graph representation learning is computationally expensive.

GraphGPS

We propose **GraphGPS*** as a unified Graph Transformer framework, learning from prior research

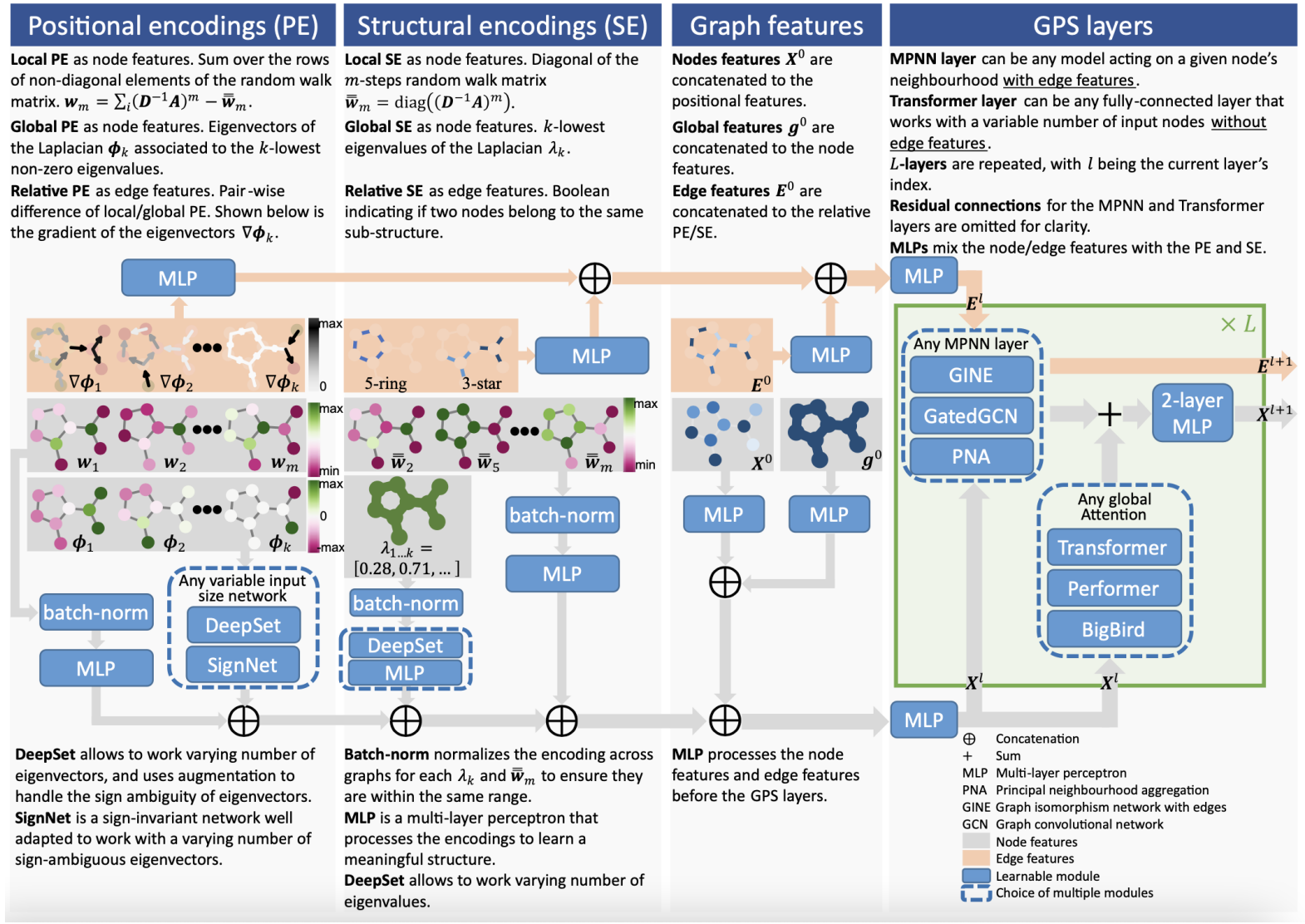
In brief, GraphGPS is a hybrid MPNN+Transformer framework which:

- Incorporates graph structure through positional and structural encodings
- Uses an MPNN layer to operate on the input graph structure
- Uses a global attention layer to operate on the global graph



[*] Rampášek, L., Galkin, M., Dwivedi, V.P., Luu, A.T., Wolf, G. and Beaini, D., 2022. Recipe for a general, powerful, scalable graph transformer.

GraphGPS



GraphGPS framework

$$
\begin{aligned}
X^{\ell+1}, E^{\ell+1} &= \mathrm{GPS}^{\ell}\left(X^{\ell}, E^{\ell}, A\right) \\
\text{with}\quad X_M^{\ell+1}, E^{\ell+1} &= \mathrm{MPNN}_e^{\ell}\left(X^{\ell}, E^{\ell}, A\right), \\
X_T^{\ell+1} &= \mathrm{Attention}^{\ell}\left(X^{\ell}\right), \\
X^{\ell+1} &= \mathrm{MLP}^{\ell}\left(X_M^{\ell+1} + X_T^{\ell+1}\right)
\end{aligned}
$$

Characteristics:

- General
- Powerful
- Scalable

GraphGPS

Table 3: Test performance in five benchmarks from [15]. Shown is the mean ± s.d. of 10 runs with different random seeds. Highlighted are the top **first**, **second**, and **third** results.

Model	ZINC	MNIST	CIFAR10	PATTERN	CLUSTER
	MAE ↓	Accuracy ↑	Accuracy ↑	Accuracy ↑	Accuracy ↑
GCN [33]	0.367 ± 0.011	90.705 ± 0.218	55.710 ± 0.381	71.892 ± 0.334	68.498 ± 0.976
GIN [61]	0.526 ± 0.051	96.485 ± 0.252	55.255 ± 1.527	85.387 ± 0.136	64.716 ± 1.553
GAT [56]	0.384 ± 0.007	95.535 ± 0.205	64.223 ± 0.455	78.271 ± 0.186	70.587 ± 0.447
GatedGCN [7, 15]	0.282 ± 0.015	97.340 ± 0.143	67.312 ± 0.311	85.568 ± 0.088	73.840 ± 0.326
GatedGCN-LSPE [16]	0.090 ± 0.001	–	–	–	–
PNA [13]	0.188 ± 0.004	97.94 ± 0.12	70.35 ± 0.63	–	–
DGN [3]	0.168 ± 0.003	–	**72.838 ± 0.417**	86.680 ± 0.034	–
GSN [6]	0.101 ± 0.010	–	–	–	–
CIN [5]	**0.079 ± 0.006**	–	–	–	–
CRaWl [53]	0.085 ± 0.004	**97.944 ± 0.050**	69.013 ± 0.259	–	–
GIN-AK+ [68]	**0.080 ± 0.001**	–	**72.19 ± 0.13**	**86.850 ± 0.057**	–
SAN [36]	0.139 ± 0.006	–	–	86.581 ± 0.037	76.691 ± 0.65
Graphormer [63]	0.122 ± 0.006	–	–	–	–
K-Subgraph SAT [9]	0.094 ± 0.008	–	–	**86.848 ± 0.037**	**77.856 ± 0.104**
EGT [29]	0.108 ± 0.009	**98.173 ± 0.087**	68.702 ± 0.409	**86.821 ± 0.020**	**79.232 ± 0.348**
GPS (ours)	**0.070 ± 0.004**	**98.051 ± 0.126**	**72.298 ± 0.356**	86.685 ± 0.059	**78.016 ± 0.180**

Model	PCQM4Mv2			
	Test-dev MAE ↓	Validation MAE ↓	Training MAE	# Param.
GCN	0.1398	0.1379	n/a	2.0M
GCN-virtual	0.1152	0.1153	n/a	4.9M
GIN	0.1218	0.1195	n/a	3.8M
GIN-virtual	0.1084	0.1083	n/a	6.7M
GRPE [48]	0.0898	0.0890	n/a	46.2M
EGT [29]	0.0872	**0.0869**	n/a	89.3M
Graphormer [51]	n/a	**0.0864**	0.0348	48.3M
GPS-small	n/a	0.0938	0.0653	6.2M
GPS-medium	n/a	**0.0858**	0.0726	19.4M

Graph**GPS** framework

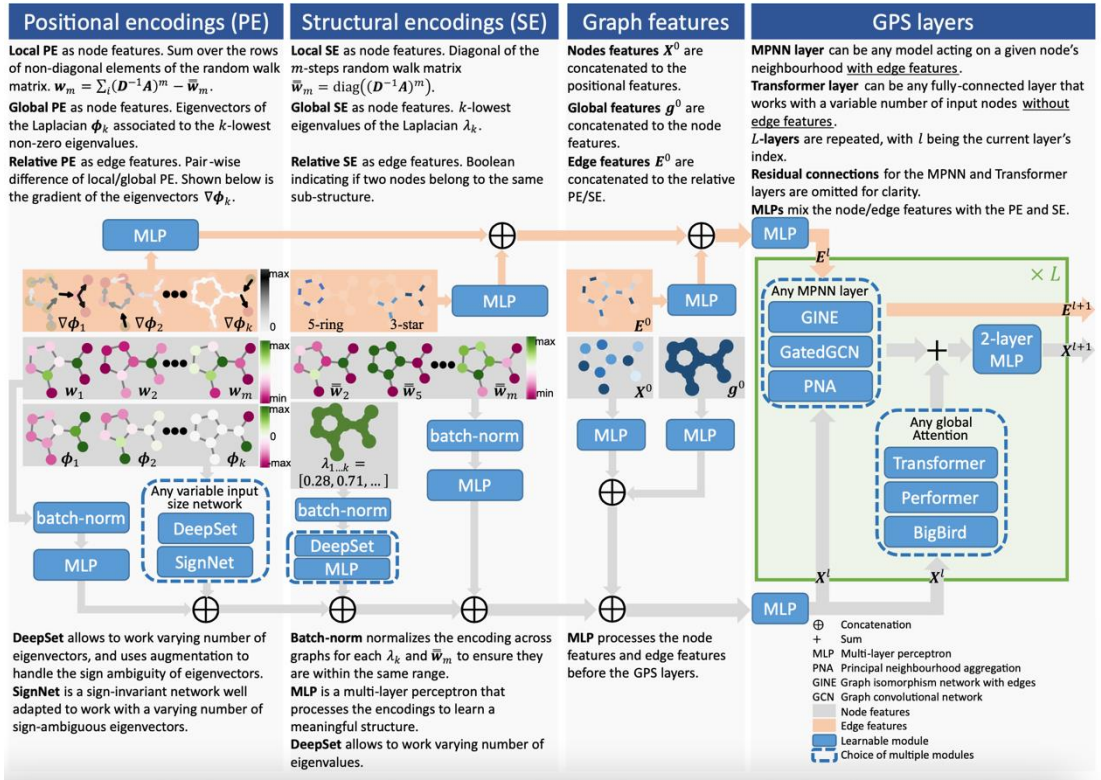
$$
\begin{aligned}
X^{\ell+1}, E^{\ell+1} &= \mathrm{GPS}^{\ell}\left(X^{\ell}, E^{\ell}, A\right) \\
\text{with } X_M^{\ell+1}, E^{\ell+1} &= \mathrm{MPNN}_e^{\ell}\left(X^{\ell}, E^{\ell}, A\right), \\
X_T^{\ell+1} &= \mathrm{Attention}^{\ell}\left(X^{\ell}\right), \\
X^{\ell+1} &= \mathrm{MLP}^{\ell}\left(X_M^{\ell+1} + X_T^{\ell+1}\right)
\end{aligned}
$$

Characteristics:

- General
- Powerful
- Scalable

GraphGPS++

GPS++: Optimized GPS model for Molecular Property Prediction [1]



$$\mathbf{X}^0 = \mathtt{Dense}\left(\left[\mathbf{X}^{\mathrm{atom}} \mid \mathbf{X}^{\mathrm{LapVec}} \mid \mathbf{X}^{\mathrm{LapVal}} \mid \mathbf{X}^{\mathrm{RW}} \mid \mathbf{X}^{\mathrm{Cent}} \mid \mathbf{X}^{\mathrm{3D}}\right]\right)$$
$$\mathbf{E}^0 = \mathtt{Dense}\left(\left[\mathbf{E}^{\mathrm{bond}} \mid \mathbf{E}^{\mathrm{3D}}\right]\right)$$

OGB Get Started Updates Large-Scale Challenge

Leaderboard for PCQM4Mv2

MAE on the test-dev and validation sets. The lower, the better.

Package: >=1.3.2

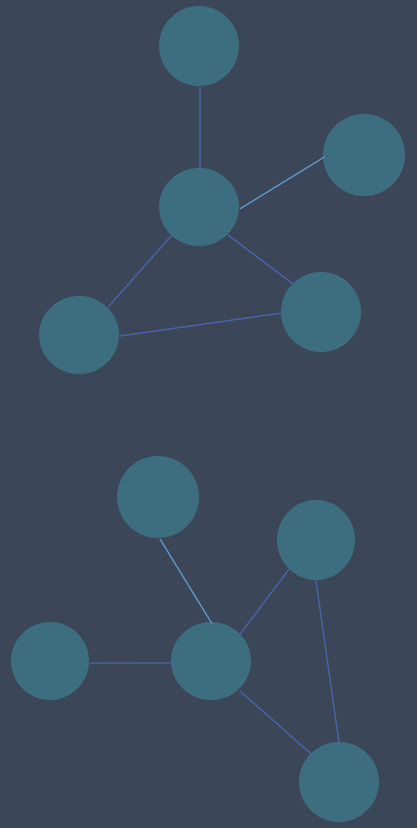
Rank	Method	Ensemble	Test-dev MAE	Validation MAE	Team
1	**GPS++**	Yes	0.0720	0.0778	GraphcoreValenceMILA
2	**MolNet_Ensemble**	Yes	0.0753	0.0797	polixir.ai
3	**Global-ViSNet**	No	0.0766	0.0784	ViSNet
4	**Transformer-M**	No	0.0782	0.0772	FML Lab@PKU

<-- Positional and Structural Encodings

[1] Masters, D., Dean, J., Klaser, K., Li, Z., Maddrell-Mander, S., Sanders, A., Helal, H., Beker, D., Rampášek, L. and Beaini, D., (2022). GPS++: An Optimised Hybrid MPNN/Transformer for Molecular Property Prediction.

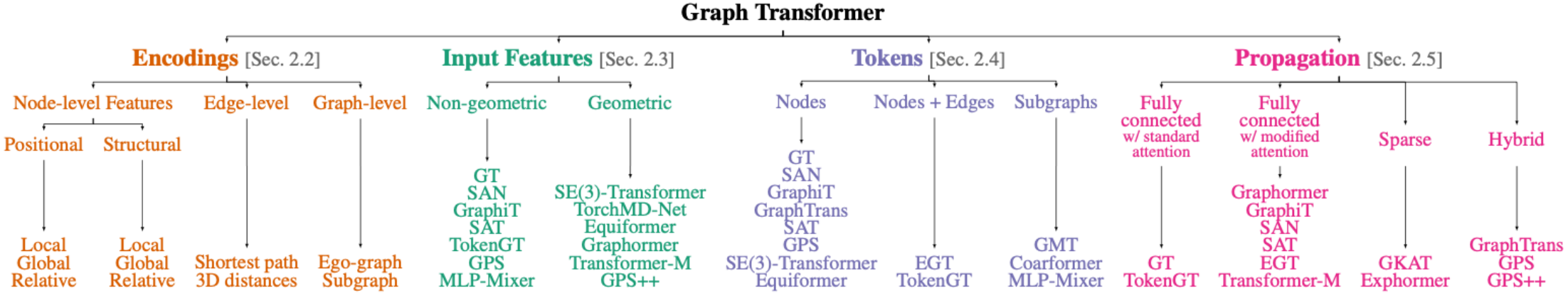
Outline of Presentation

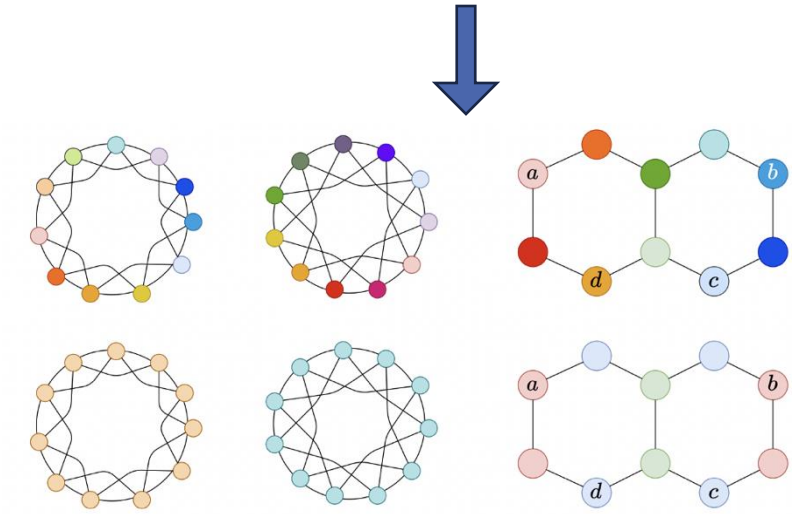
- Transformers
- Attention as Message Passing
- Graph Transformers
- Bottlenecks and Remedies
- **Recent Research**



Recent Research

Taxonomy of Graph Transformers [1]

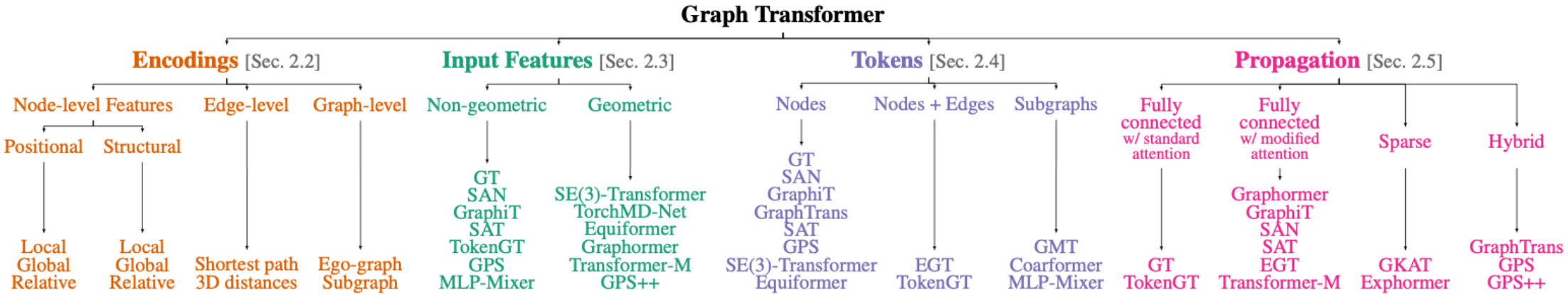


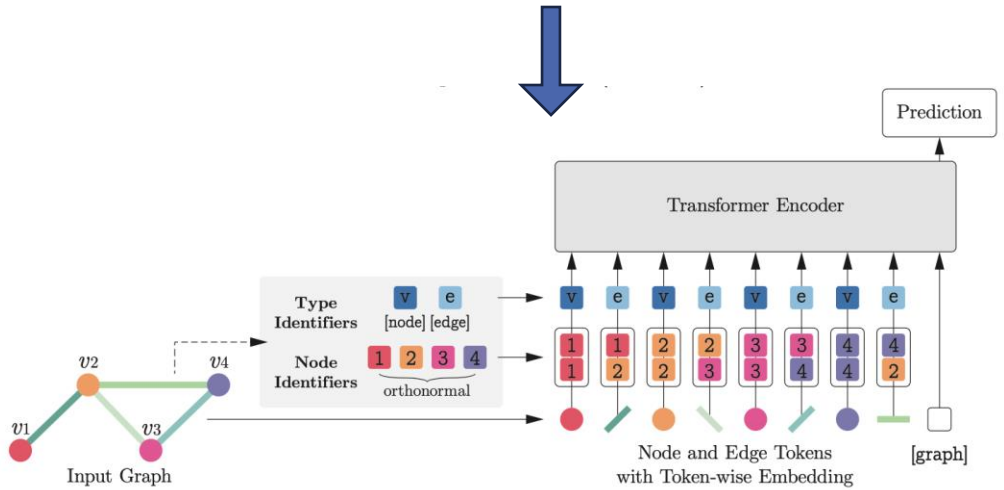


[1] Müller, L., Galkin, M., Morris, C. and Rampášek, L., (2023). Attending to Graph Transformers.

Recent Research

Taxonomy of Graph Transformers [1]

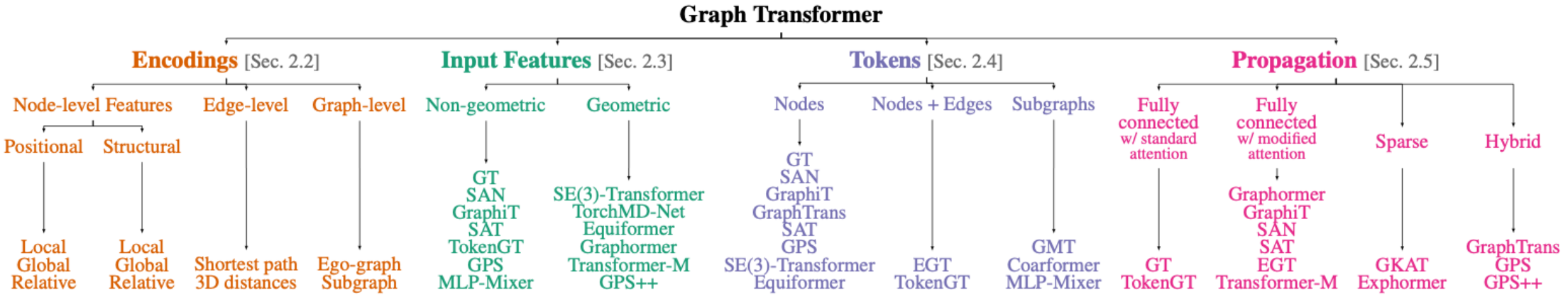


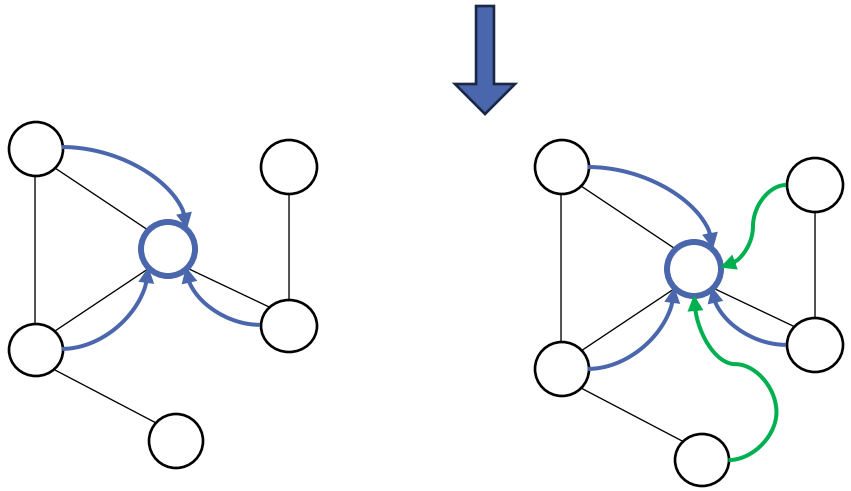


[1] Müller, L., Galkin, M., Morris, C. and Rampášek, L., (2023). Attending to Graph Transformers.

Recent Research

Taxonomy of Graph Transformers [1]





[1] Müller, L., Galkin, M., Morris, C. and Rampášek, L., (2023). Attending to Graph Transformers.

Graph Transformers for Large Graphs

Challenges in scalability of MPNNs or GTs

- Only using local aggregation produces information bottleneck
- Using full attention (to all nodes) produces computational infeasibility
- GraphGPS does not scale to single large graphs such as ogbn-* due to its dependency on the number of nodes
- How can we scale Graph Transformers for single large graphs?

		Transformer
Dataset	Train ratio	GPS
ogbn-arxiv	official 54%	OOM
ogbn-products	official 8%	OOM
Performance average		OOM
arxiv-year	random 10%	OOM
arxiv-year	random 20%	OOM
arxiv-year	random 50%	OOM
snap-patents	random 10%	OOM
snap-patents	random 20%	OOM
snap-patents	random 50%	OOM
Performance average		OOM
Overall performance average		OOM

Fig from [1]

[1] Kong, K., Chen, J., Kirchenbauer, J., Ni, R., Bruss, C.B. and Goldstein, T., 2023, July. GOAT: A global transformer on large-scale graphs.

Graph Transformers for Large Graphs

Desired Characteristics:

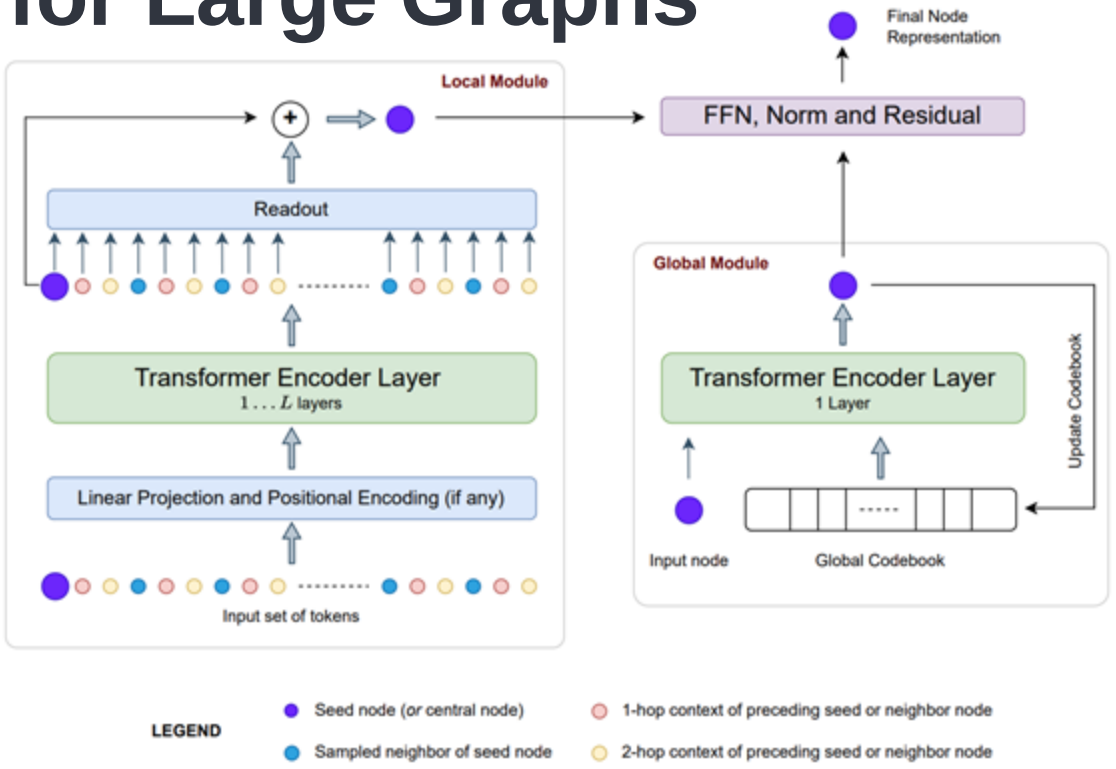
- **D1- Model Capacity.** A graph learning model should possess the ability to incorporate both local and global information from the original large graph.
  - *Local Graph Inductive Bias*
  - *Access to Non-Local Information*

- **D2- Scalability.** Computational feasibility is a necessity when we scale model to larger graphs.
  - *Efficient Neighbor Node Set Retrieval*
  - *Efficient Global Information Access*
  - *Distributed Training*

Graph Transformers for Large Graphs

We propose* **LargeGT**, a scalable Graph Transformer framework that

- combines local and global graph information while addressing their respective computational bottlenecks,
- can efficiently handle large graphs with millions of nodes.



1. Local Module

- New tokenization strategy for local features
- 4-hop receptive field with 2-hop operations
- Offline neighbor sampling for training efficiency

2. Global Module

- Codebook-based approach for global attention [1]
- Linear complexity to codebook size
- Independent of graph size for scalability

3. Update Step

$$
\begin{aligned}
\mathbf{S}_i &= \textsc{LocalNodes}_i(\mathbf{A}, K) \\
\mathbf{X}_i &= \textsc{InputTokens}_i(\mathbf{S}_i, \mathbf{H}_i^{\text{in}}, \mathbf{C}_i) \\
\mathbf{H}_i^{\text{local}} &= \textsc{LocalModule}(\mathbf{X}_i) \\
\mathbf{H}_i^{\text{global}} &= \textsc{GlobalModule}(\mathbf{H}_i^{\text{in}}) \\
\hat{\mathbf{H}}_i &= \text{FFN}(\mathbf{H}_i^{\text{local}} \,||\, \mathbf{H}_i^{\text{global}}) \\
\mathbf{H}_i^{\text{out}} &= \mathbf{H}_i^{\text{in}} + \textsc{Norm}(\hat{\mathbf{H}}_i)
\end{aligned}
$$

[*] Dwivedi, V.P., Liu, Y., Luu, A.T., Bresson, X., Shah, N. and Zhao, T., 2023. Graph transformers for large graphs.

[1] Kong, K., Chen, J., Kirchenbauer, J., Ni, R., Bruss, C.B. and Goldstein, T., 2023, July. GOAT: A global transformer on large-scale graphs.

Graph Transformers for Large Graphs

Algorithm 1 LocalNodes: Algorithm to fetch a multiset of local nodes from 1 and 2 hop neighbors for each node.

Require: A graph with adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, and the size of the multiset K.
Ensure: Return the matrix with K-sized multisets for each node $\mathbf{S} \in \mathbb{R}^{N \times K}$

Initialize: Multisets for all nodes $\mathbf{S} \in \mathbb{R}^{N \times K}$
for $i = 0$ to $N - 1$ **do**
  $\hat{\mathbf{T}} \leftarrow$ 1 and 2 hop neighbors of node i
  if $|\hat{\mathbf{T}}| \geq K - 1$ **then**
    $\mathbf{T} \leftarrow$ Randomly sample $K - 1$ nodes from $\hat{\mathbf{T}}$
  else if $|\hat{\mathbf{T}}| < K - 1$ and $|\hat{\mathbf{T}}| > 0$ **then**
    $\mathbf{T} \leftarrow$ Randomly sample from $\hat{\mathbf{T}}$ with replacement to make $K - 1$ nodes
  else
    $\mathbf{T} \leftarrow$ Randomly sample $K - 1$ nodes from $\{0, 1, 2, \ldots, N - 1\}$
  end if
  $\mathbf{S}_i \leftarrow \{i\} + \{\mathbf{T}\} \in \mathbb{R}^K$
end for

Algorithm 2 InputTokens: Algorithm for Mini-Batch Preparation for Local Module

Require: Mini-batch of M samples $\mathbf{S} \in \mathbb{R}^{M \times K}$ where K is the total size of the multiset of nodes for each node, Feature matrix $\mathbf{H} \in \mathbb{R}^{N \times D}$, Hop context features $\mathbf{C} \in \mathbb{R}^{N \times 2 \times D}$.
Ensure: Return the input data of all nodes in mini-batch $\mathbf{X} \in \mathbb{R}^{M \times 3K \times D}$.

Initialize: $\mathbf{X} \in \mathbb{R}^{M \times 3K \times D}$
for $i = 0$ to $M - 1$ **do**
  for $j = 0$ to $3K$ with step 3 **do**
    $\mathbf{X}_{i,j} \leftarrow \mathbf{H}[\mathbf{S}_{i,j}]$ # node feature for node i from the feature matrix $\mathbf{H}$
    $\mathbf{X}_{i,j+1} \leftarrow \mathbf{C}[\mathbf{S}_{i,j}, 0]$ # 1 hop context feature for the node i from $\mathbf{C}$
    $\mathbf{X}_{i,j+2} \leftarrow \mathbf{C}[\mathbf{S}_{i,j}, 1]$ # 2 hop context feature for the node i from $\mathbf{C}$
  end for
end for

Algorithm 3 GlobalModule: Algorithm to output global representations of nodes in mini-batch

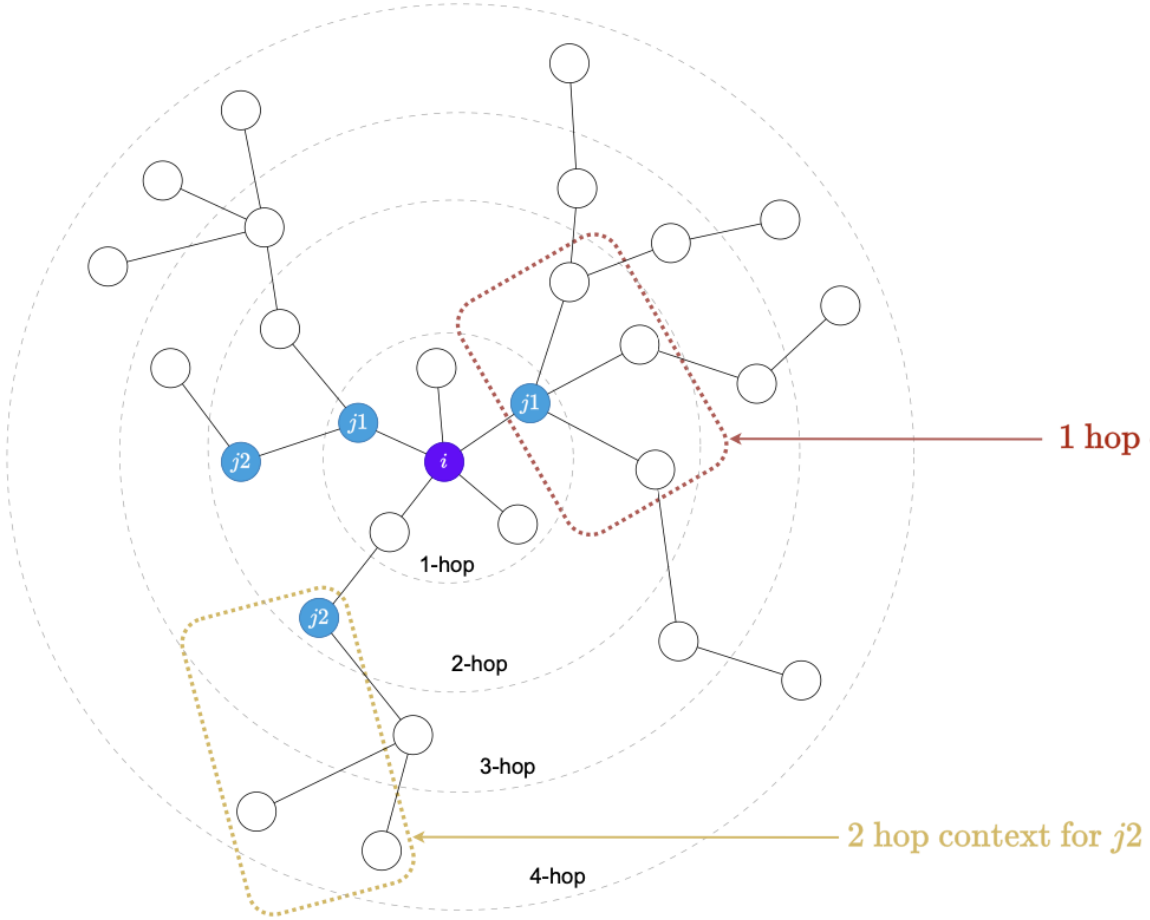
Require: Hidden features $\mathbf{H}_{\text{in}} \in \mathbb{R}^{M \times D}$ for M nodes in a mini-batch, $\boldsymbol{\mu}$ is the centroid computed by the K-Means algorithm, $\mathbf{P}$ is the centroid assignment index for each node.
Ensure: Return the output representations of all nodes in the mini-batch $\mathbf{H}^{\text{global}}$.

for $i = 0$ to $M - 1$ **do**
  $\mathbf{x} \leftarrow \text{MLP}_a(\mathbf{H}_i^{\text{in}})$
  $\mathbf{q} \leftarrow \mathbf{x}\mathbf{W}_Q$
  $\mathbf{K} \leftarrow \boldsymbol{\mu}\mathbf{W}_K$
  $\mathbf{V} \leftarrow \boldsymbol{\mu}\mathbf{W}_V$
  $\hat{\mathbf{H}}_i^{\text{global}} \leftarrow \text{Softmax}\left(\frac{\mathbf{q}\mathbf{K}^\top}{\sqrt{D}} + \log\left(\mathbf{1}_n \mathbf{P}\right)\right)\mathbf{V}$
  $\mathbf{H}_i^{\text{global}} \leftarrow \text{MLP}_b(\hat{\mathbf{H}}_i^{\text{global}})$
  Update $\boldsymbol{\mu}$ using $\mathbf{x}$ through Exponential Moving Average (EMA) K-Means algorithm.
end for

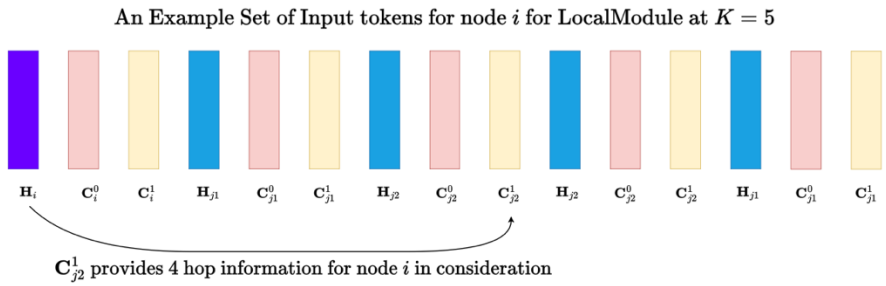
Algorithm 3 adapted from [1]

[1] Kong, K., Chen, J., Kirchenbauer, J., Ni, R., Bruss, C.B. and Goldstein, T., 2023, July. GOAT: A global transformer on large-scale graphs.

Graph Transformers for Large Graphs



An illustration of how the tokenization strategy prepares tokens for local module to attain receptive field of up to 4-hops for a node in consideration.

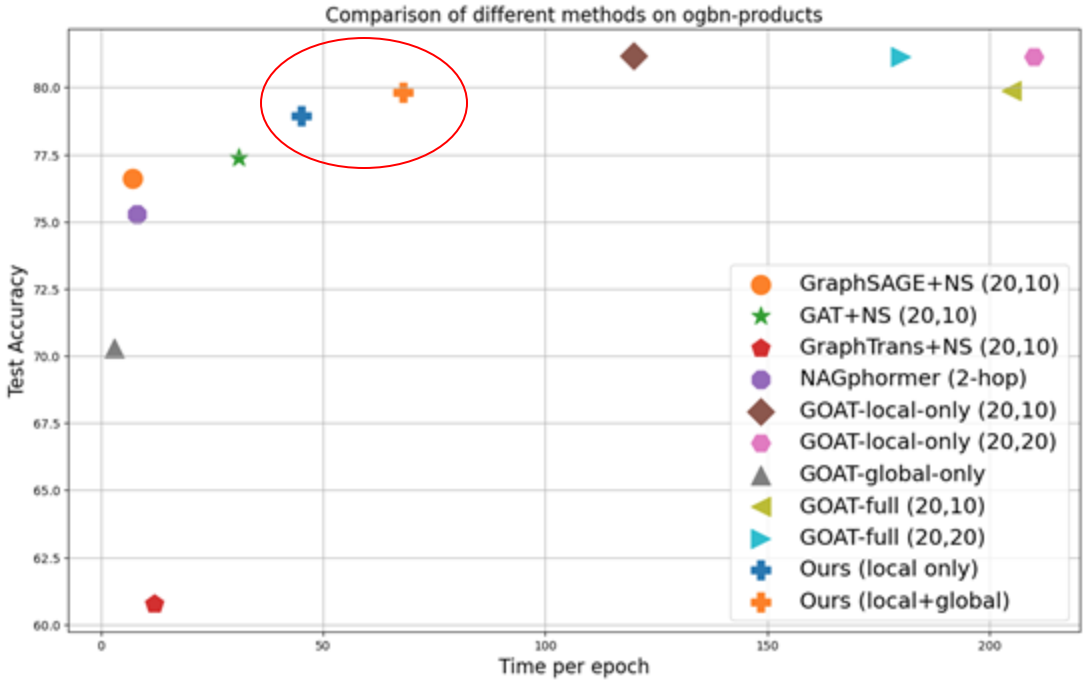


Graph Transformers for Large Graphs

- Node classification task on ogbn-products (2M+ nodes)
- Ideally, we want the model to be efficient and high performing, *i.e.*, left top of chart.

- Proposed method, LargeGT, maintains a good balance between speed and accuracy

- LargeGT improves baselines on other datasets



(A) ogbn-products

Model	Test Acc
GraphSAGE-δ	76.62±0.93
GAT-δ	77.38±0.59
GT-sparse-δ	60.76±0.00
NAGphormer-δ	75.28±0.04
GOAT-local-δ	**81.17±0.12**
GOAT-global-δ	70.28±1.95
GOAT-full-δ	**79.88±0.20**
LargeGT-local	**78.95±0.80**
LargeGT-full	**79.81±0.25**

(B) snap-patents

Model	Test Acc
GraphSAGE-δ	48.43±0.21
GAT-δ	45.92±0.22
GT-sparse-δ	47.81±0.00
NAGphormer-δ	**60.11±0.05**
GOAT-local-δ	40.95±0.16
GOAT-global-δ	42.65±0.07
GOAT-full-δ	50.28±0.14
LargeGT-local	**68.19±3.11**
LargeGT-full	**70.21±0.12**

(C) ogbn-papers100M

Model	Test Acc
GOAT-full-δ	61.12±0.10
LargeGT-full	**64.73±0.05**

Overall Takeaways

- Transformers can be generalized for learning on graph structured datasets.

- Positional Encodings / Structural Encodings are key design components.

- There exists multiple ways to inject positional/structural information from the original graph structure, and several works aim to address respective limitations.

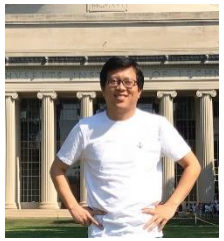
- A hybrid MPNN + Transformer architecture seems to outperform other models on graph learning (especially on graph prediction tasks).

- It is challenging to apply existing Graph Transformers on large single graphs without further design modifications to address the computational bottlenecks.

Thank you ☺

Questions?

Thanks to Collaborators



A.T. Luu
NTU
Singapore

X. Bresson
NUS
Singapore

T. Laurent
LMU
USA

Y. Bengio
Mila
Canada

C.K. Joshi
U of Cambridge
UK

L. Rampášek
Isomorphic Labs
UK

M. Galkin
Google
USA

D. Beaini
Valence Labs
Canada

A. Parviz
NJIT
USA

G. Wolf
U of Montréal
Canada

T. Zhao
Snap Inc
USA

N. Shah
Snap Inc
USA

Y. Liu
Snap Inc
USA