EXPHORMER

SPARSE TRANSFORMERS FOR GRPAHS

Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam,
Danica J. Sutherland , Ali Kemal Sinop

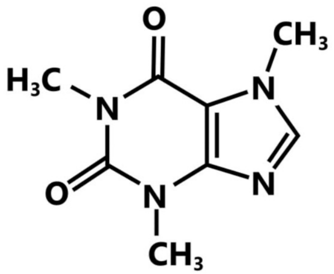
Seminar in Deep Neural Networks
Presenter: Johannes Herter
May 21, 2023

ETH zürich

GRAPH LEARNING: GRAPHS ARE WIDELY APPLICABLE ACROSS DOMAINS

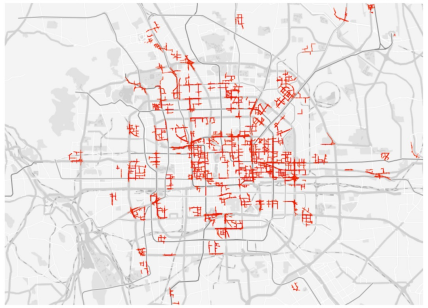
Social networks

Chemoinformatics

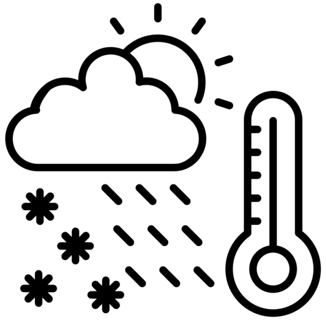


Product recommendations

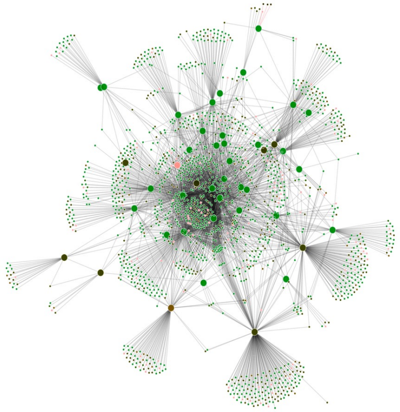
Traffic prediction



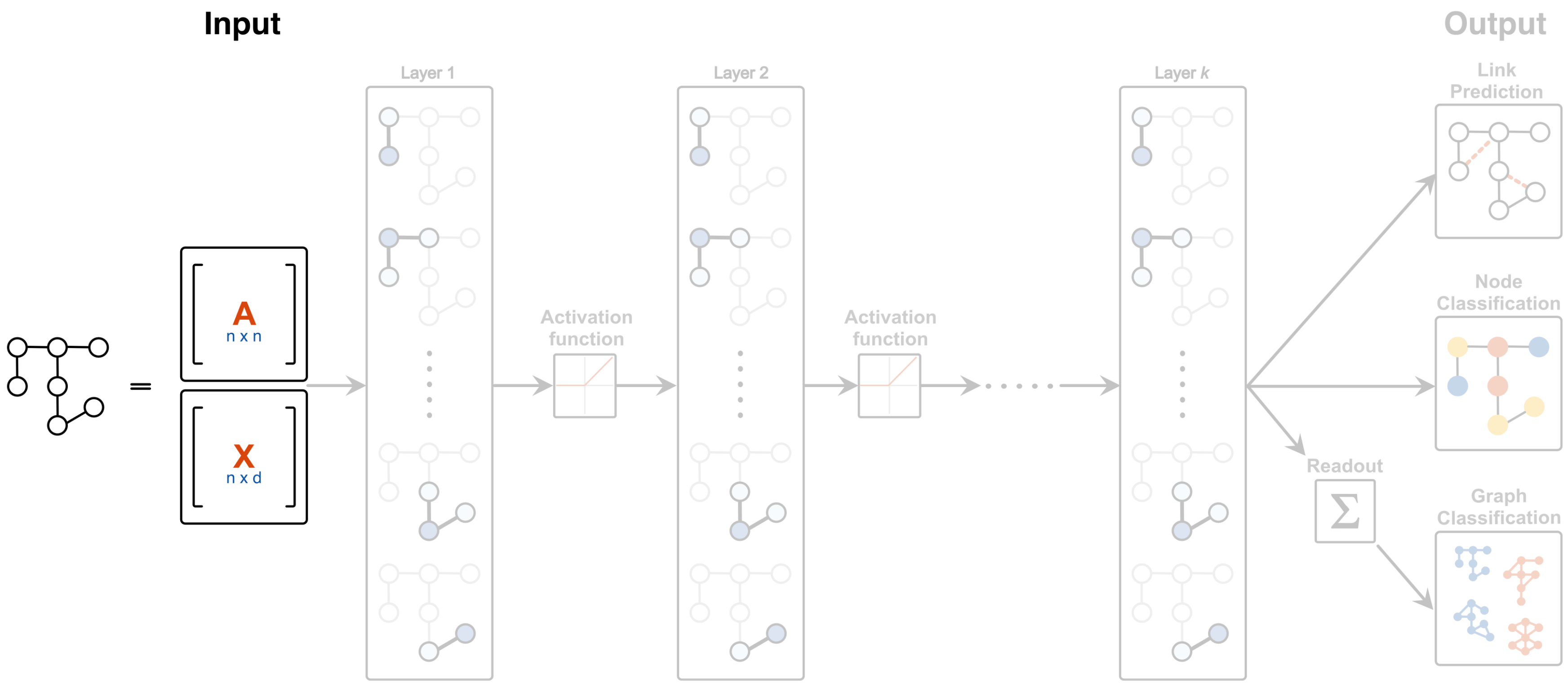
Weather forecasting



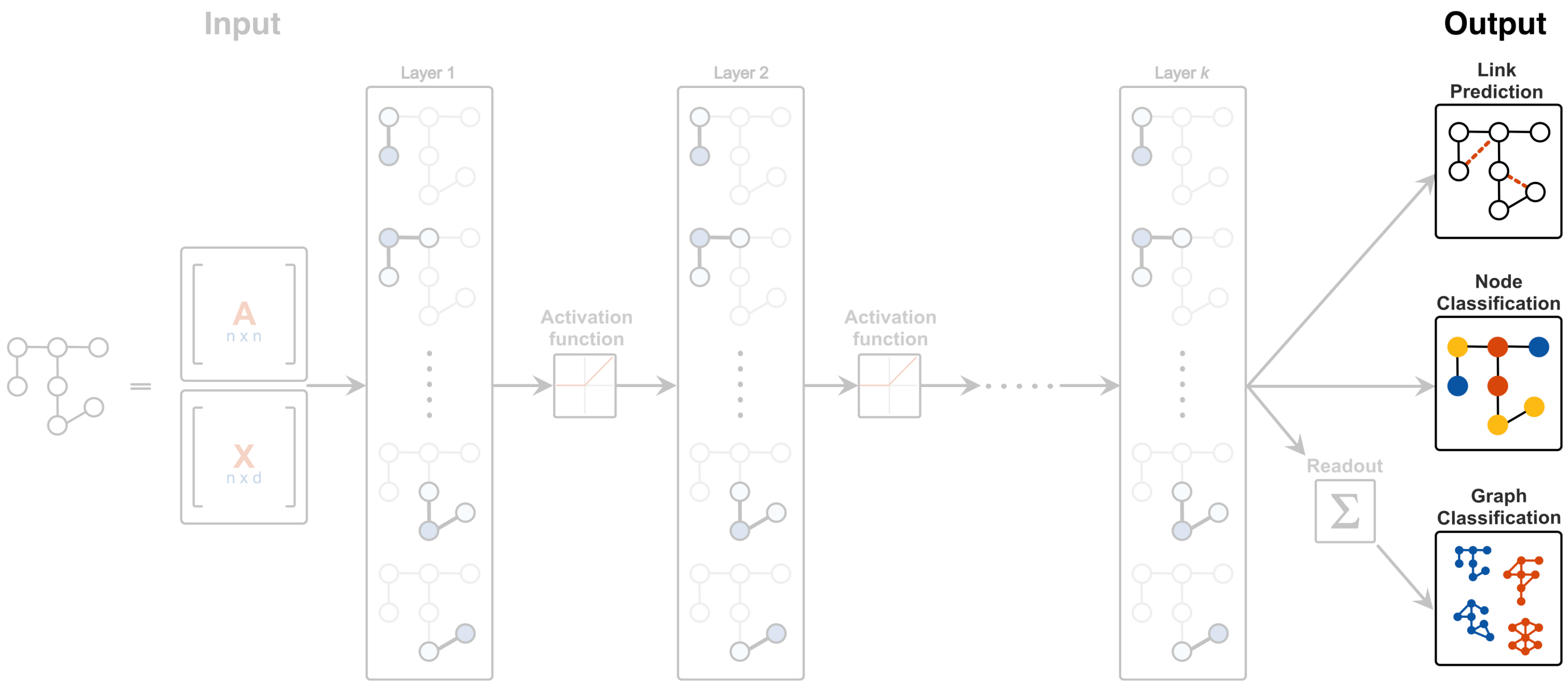
Protein-protein associations



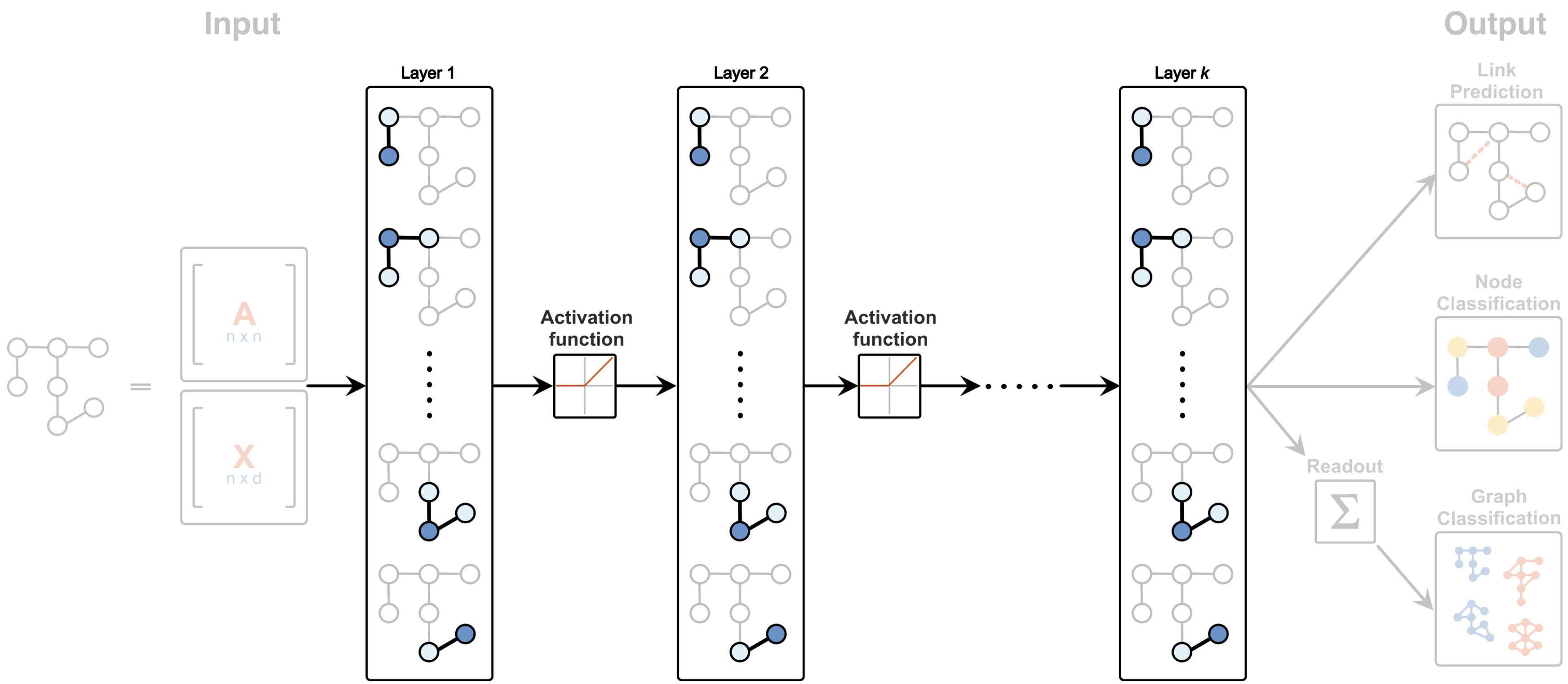
CURRENT PARADIGM USES MESSAGE PASSING



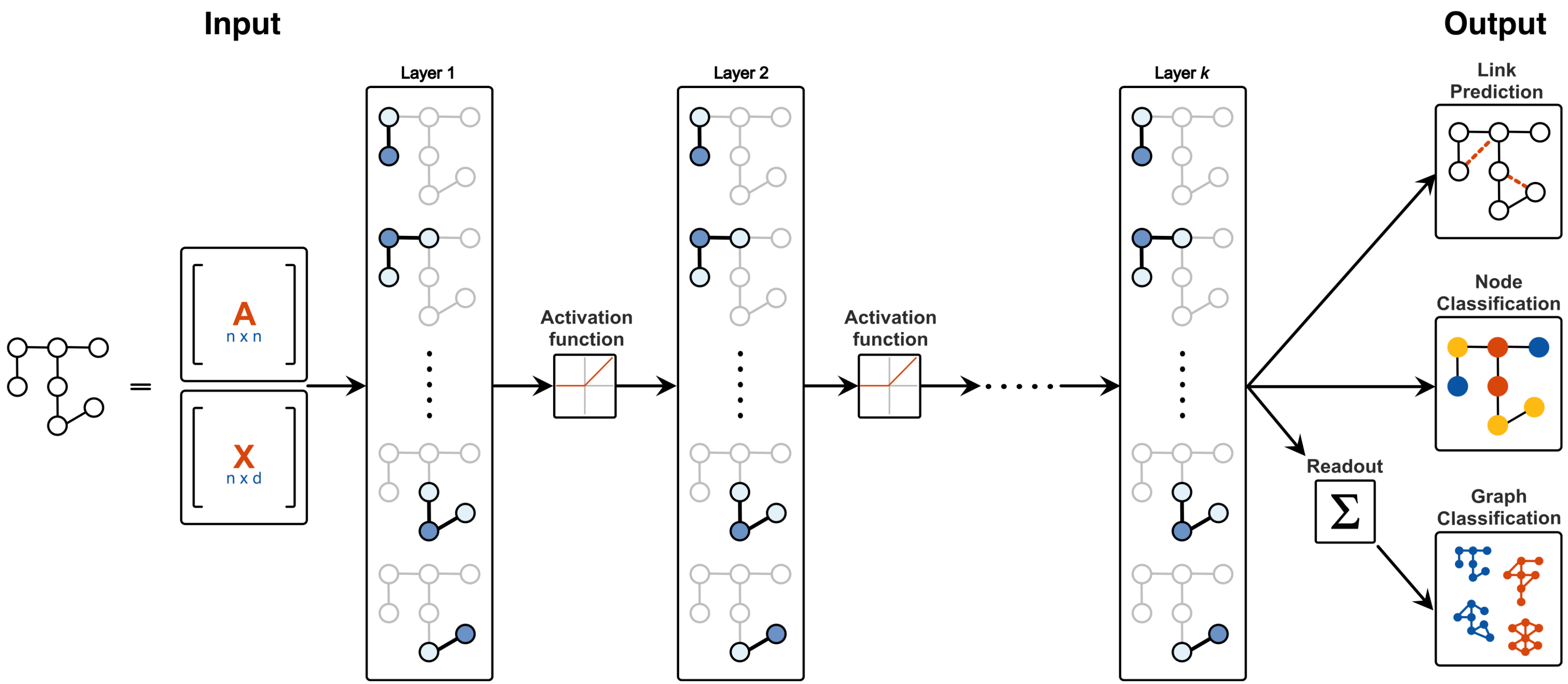
CURRENT PARADIGM USES MESSAGE PASSING



CURRENT PARADIGM USES MESSAGE PASSING

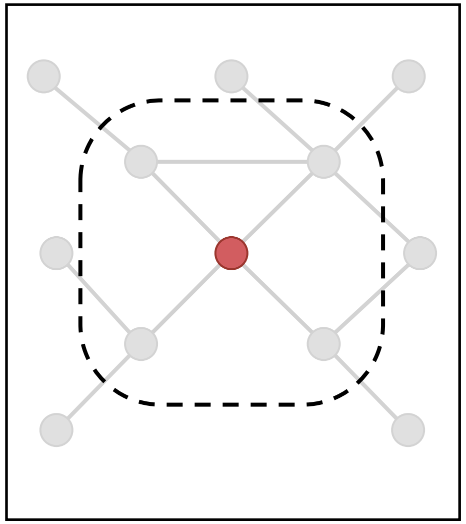


CURRENT PARADIGM USES MESSAGE PASSING



MPNNS INCORPORATE A LOCAL VIEW OF THE GRAPH, WHICH CAN CAUSE ISSUES

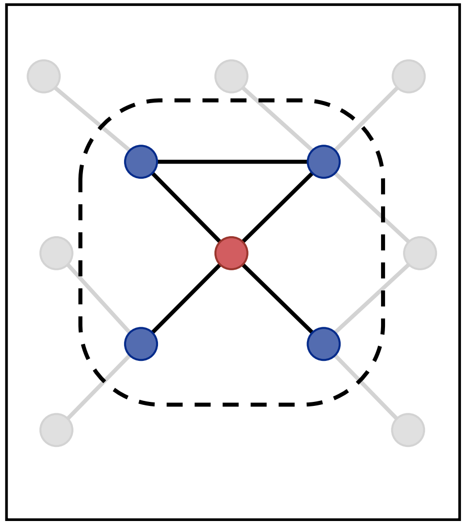
Under-reaching



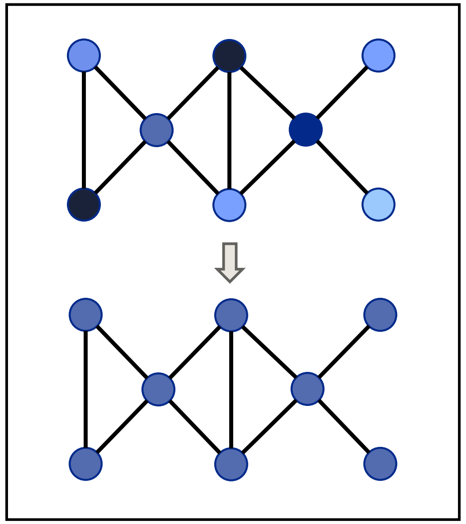
a node cannot see some nodes
(Barceló et al., 2020)

GNNS INCORPORATE A LOCAL VIEW OF THE GRAPH, WHICH CAN CAUSE ISSUES

Under-reaching



Over-smoothing

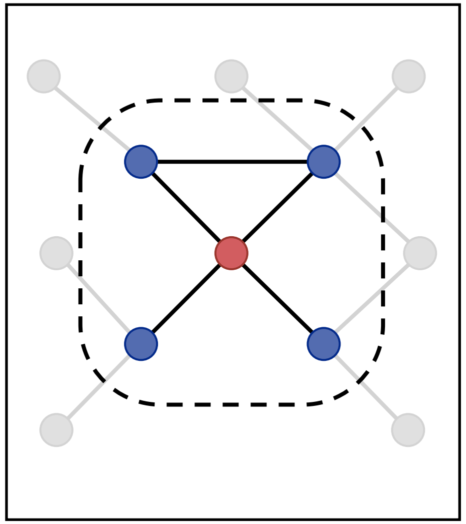


all nodes start to look *the same*

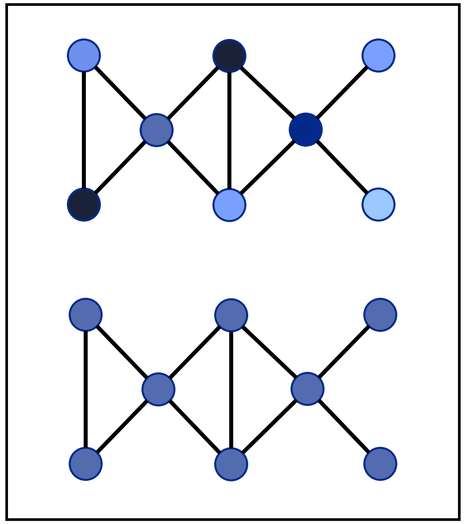
(Oono and Suzuki, 2020)

GNNS INCORPORATE A LOCAL VIEW OF THE GRAPH, WHICH CAN CAUSE ISSUES

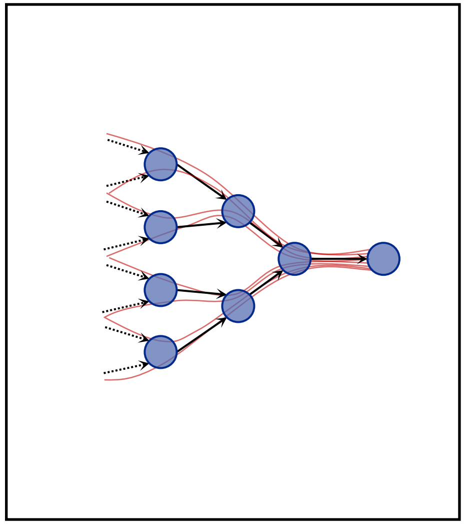
Under-reaching



Over-smoothing



Over-squashing

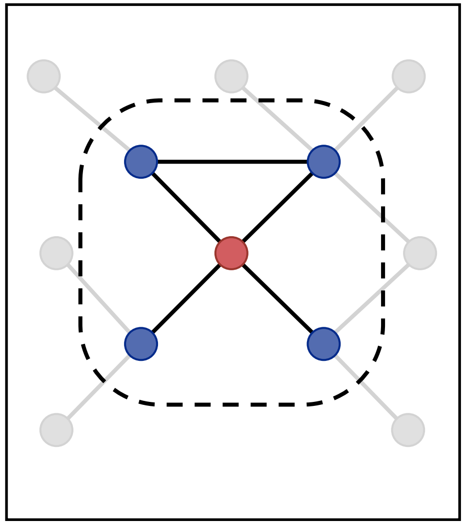


information gets *"lost"* in transit

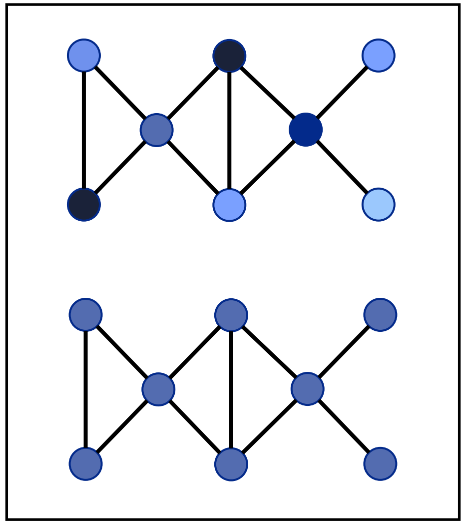
(Topping et al., 2022)

GNNS INCORPORATE A LOCAL VIEW OF THE GRAPH, WHICH CAN CAUSE ISSUES

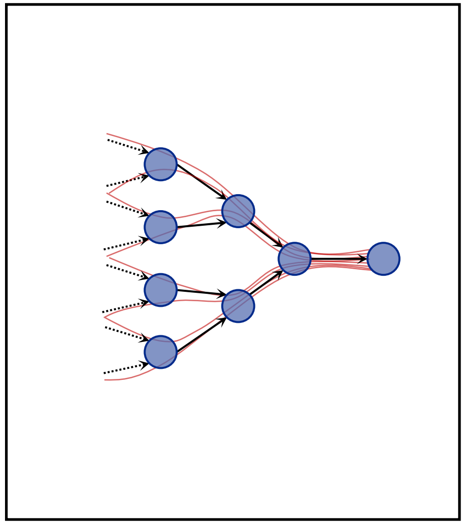
Under-reaching



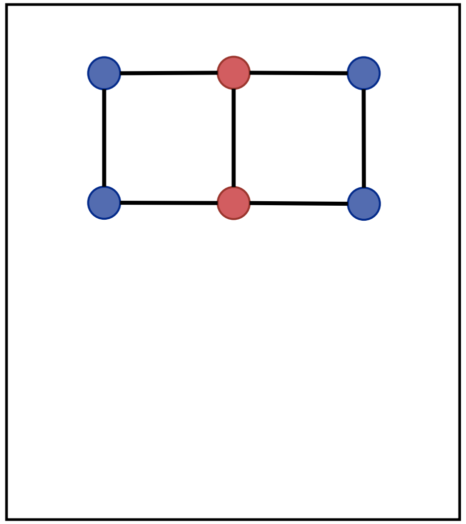
Over-smoothing



Over-squashing



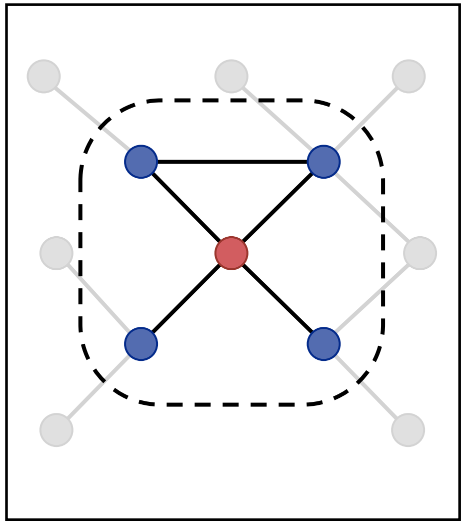
Limited Expressivity



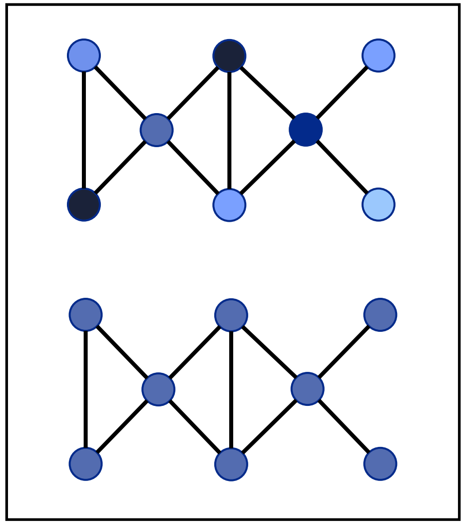
at most as expressive as 1-WL
(Morris et al., 2019)

GNNS INCORPORATE A LOCAL VIEW OF THE GRAPH, WHICH CAN CAUSE ISSUES

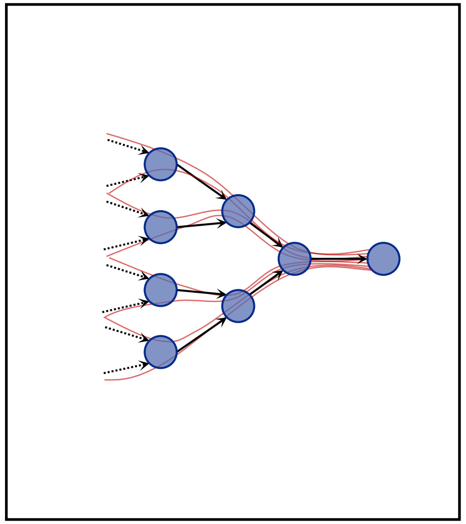
Under-reaching



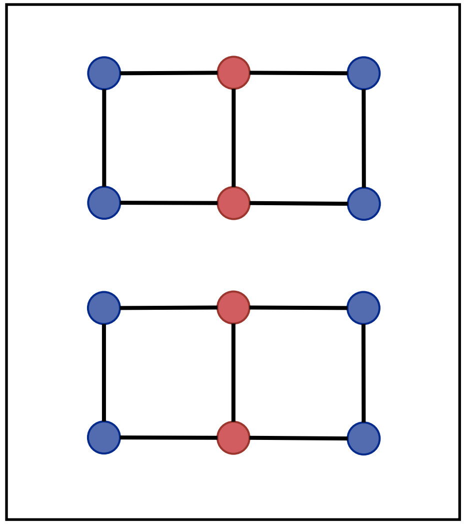
Over-smoothing



Over-squashing



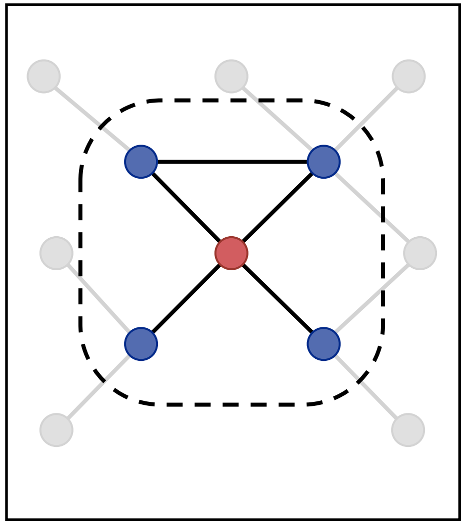
Limited Expressivity



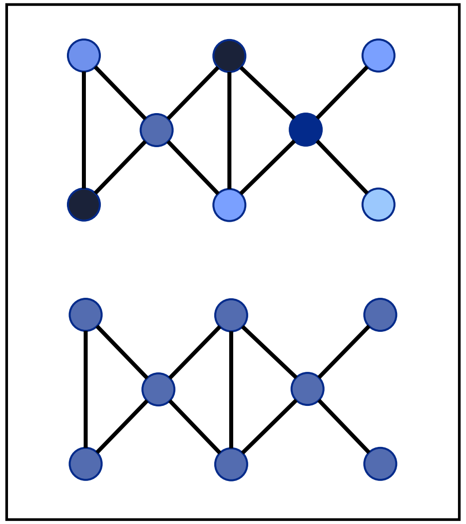
at most as expressive as 1-WL
(Morris et al., 2019)

GNNS INCORPORATE A LOCAL VIEW OF THE GRAPH, WHICH CAN CAUSE ISSUES

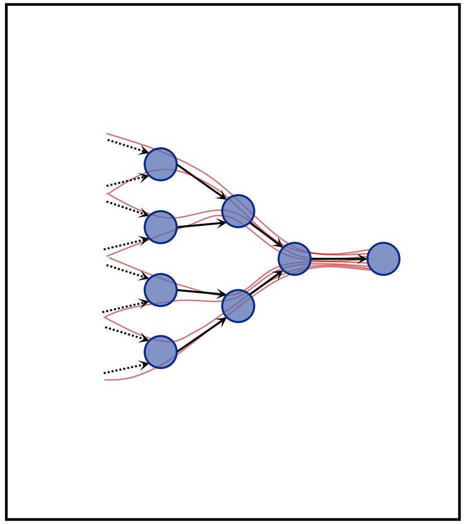
Under-reaching



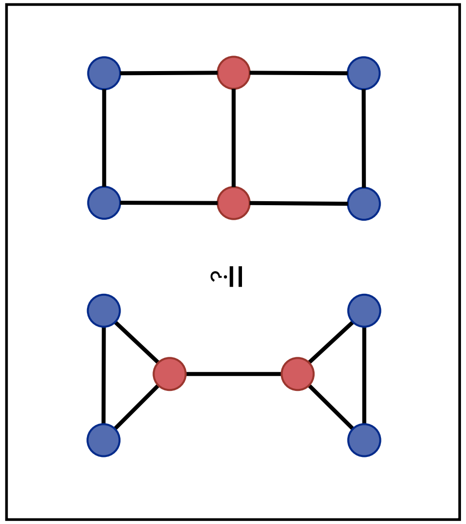
Over-smoothing



Over-squashing



Limited Expressivity



at most as expressive as 1-WL
(Morris et al., 2019)

COULD WE OVERCOME THESE ISSUES BY INCORPORATING

GLOBAL INFORMATION?

INTEGRATING GLOBAL INFORMATION

Ways of incorporating global information into graph learning?

- Virtual nodes
- Unique node identifiers
- Adding global information e.g. as node features
- Transformers

INTEGRATING GLOBAL INFORMATION VIA TRANSFORMERS

Why use transformers?

- ❖ Transformers have had success across a variety of domains
  - Natural language processing (OpenAI,2023)
  - Computer vision (Dosovitskiy et al.,2020)
  - Speech (Gulati et al.,2020)
  - Biological sequence modelling (Rives et al.,2021)

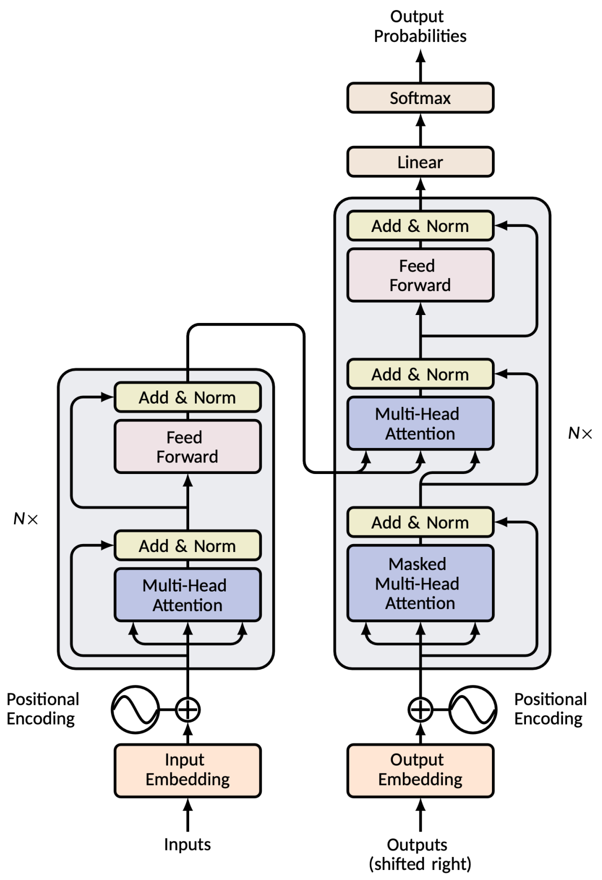
INTEGRATING GLOBAL INFORMATION VIA TRANSFORMERS

Why use transformers?

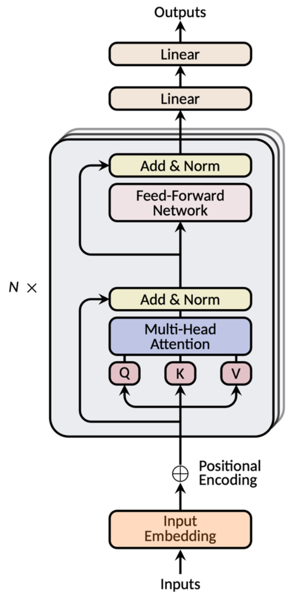
- ❖ Transformers have had success across a variety of domains
- ❖ A single self-attention layer addresses the issues of message passing GNNs
- ❖ End-to-end trainable
- ❖ Doesn’t require hand-crafted features

TYPES OF TRANSFORMER ARCHITECTURES

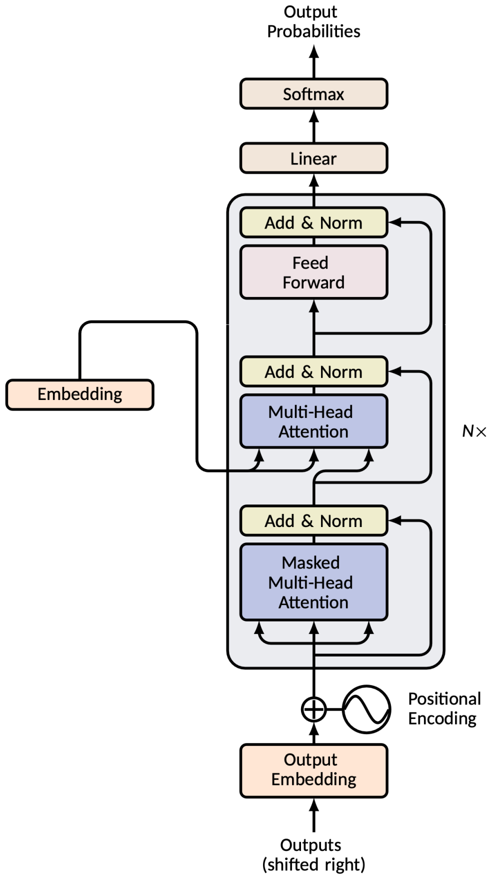
Encoder-Decoder



Encoder-only

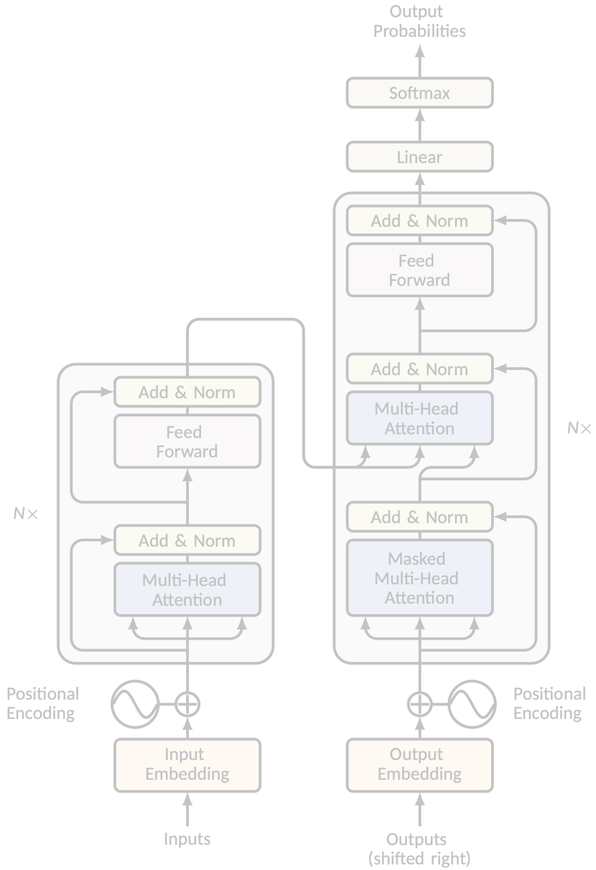


Decoder-only

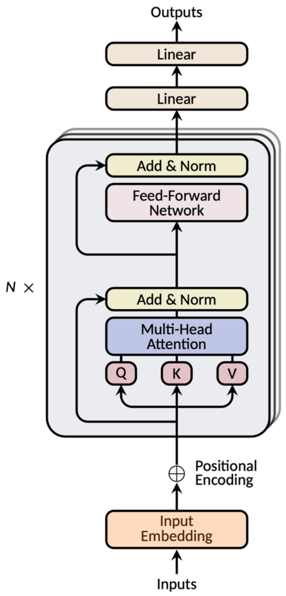


TYPES OF TRANSFORMER ARCHITECTURES

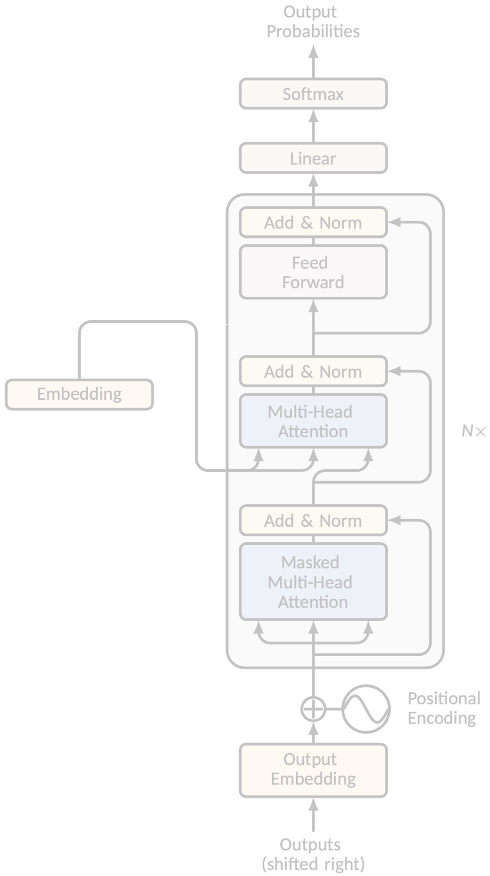
Encoder-Decoder



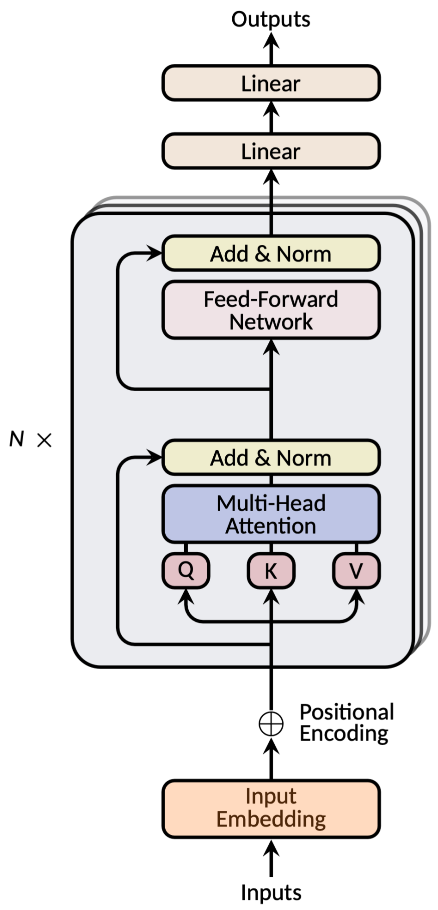
Encoder-only



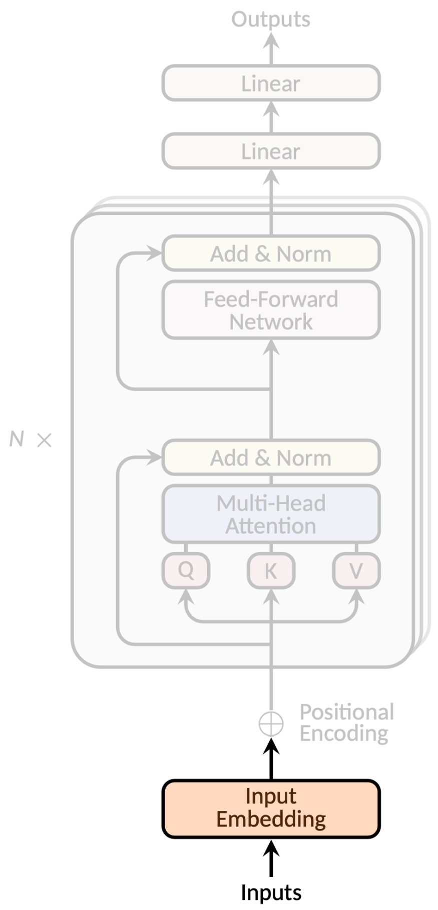
Decoder-only



HOW CAN WE ADAPT THE TRANSFORMER TO WORK WELL ON GRAPH DATA?

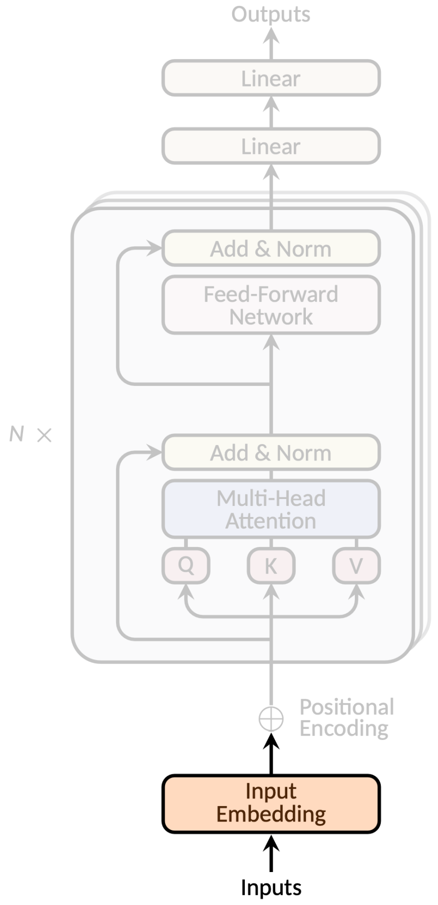


HOW CAN WE ADAPT THE TRANSFORMER TO WORK WELL ON GRAPH DATA?



NLP	Graph Learning
❖ Input tokenization	❖ Input treats the graph as bag of nodes

HOW CAN WE ADAPT THE TRANSFORMER TO WORK WELL ON GRAPH DATA?

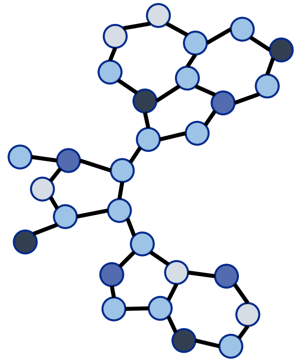


NLP

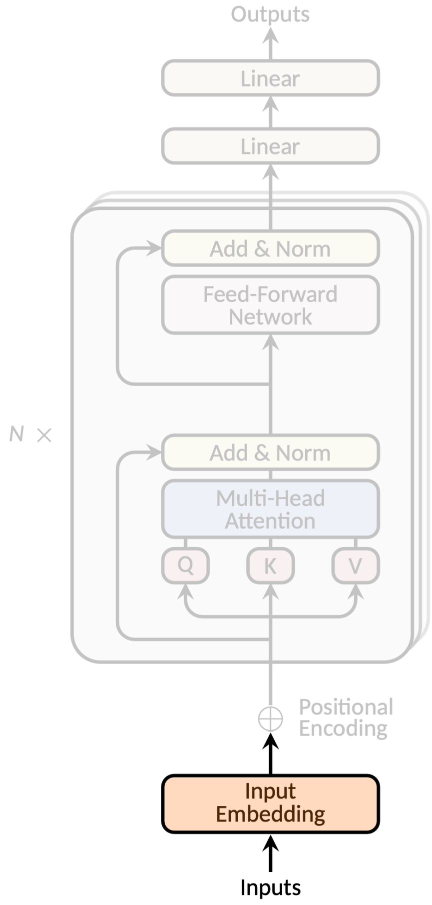
❖ Input tokenization

Graph Learning

❖ Input treats the graph as bag of nodes



HOW CAN WE ADAPT THE TRANSFORMER TO WORK WELL ON GRAPH DATA?

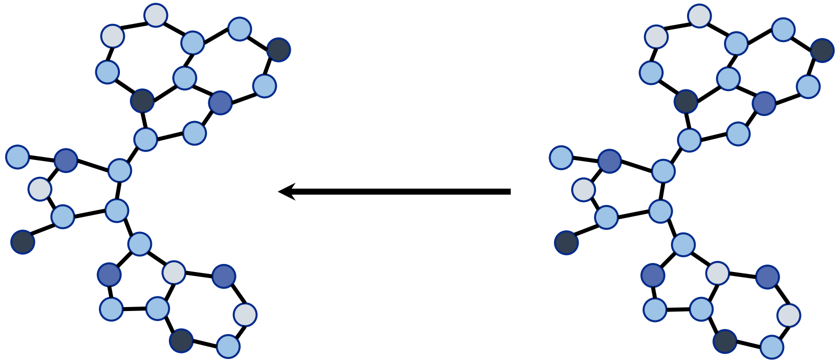


NLP

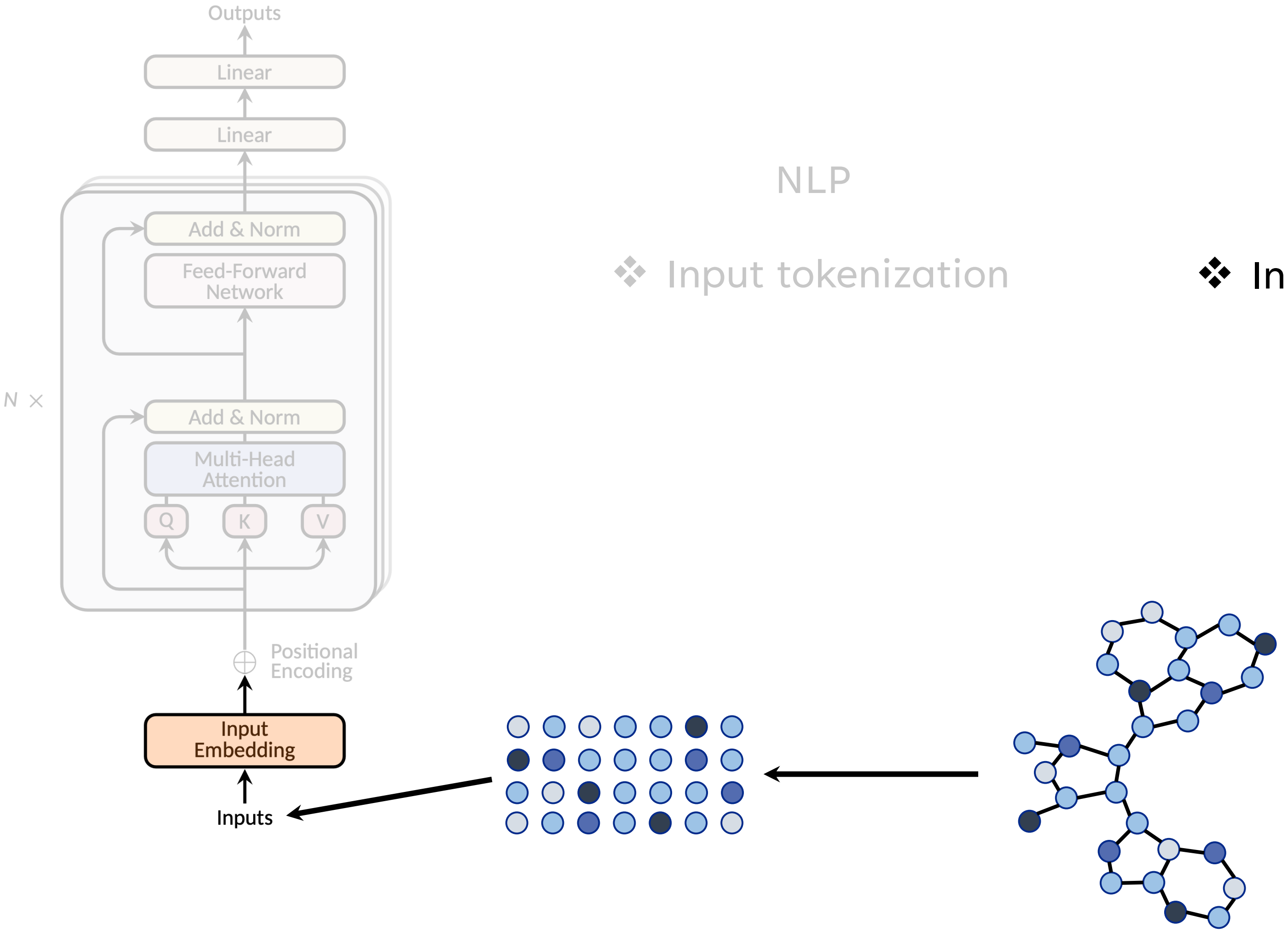
❖ Input tokenization

Graph Learning

❖ Input treats the graph as bag of nodes



HOW CAN WE ADAPT THE TRANSFORMER TO WORK WELL ON GRAPH DATA?



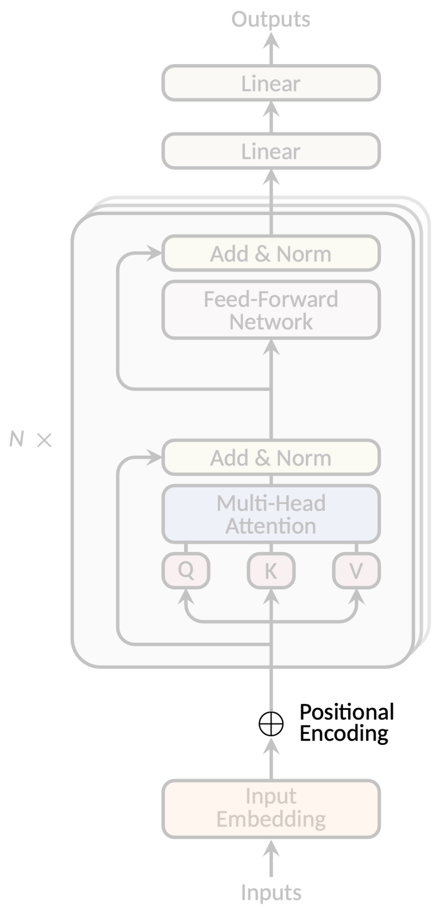
NLP

- Input tokenization

Graph Learning

- Input treats the graph as bag of nodes

HOW CAN WE ADAPT THE TRANSFORMER TO WORK WELL ON GRAPH DATA?



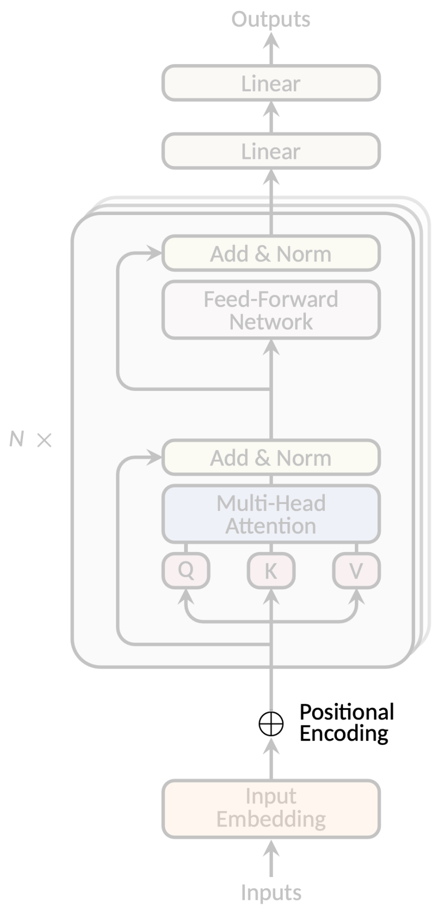
NLP

- ❖ Input tokenization
- ❖ Positional encoding
  - Captures a word's position in a text
  - Is added/ concatenated to the input

Graph Learning

- ❖ Input treats the graph as bag of nodes

HOW CAN WE ADAPT THE TRANSFORMER TO WORK WELL ON GRAPH DATA?



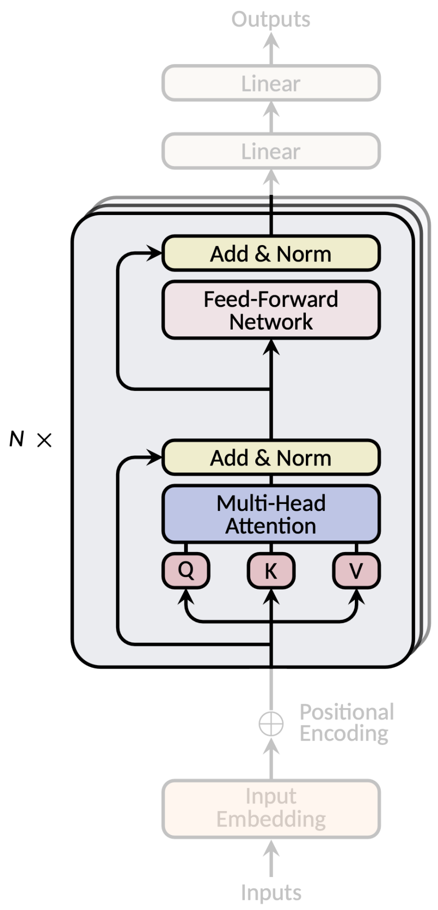
NLP

- ❖ Input tokenization
- ❖ Positional encoding

Graph Learning

- ❖ Input treats the graph as bag of nodes
- ❖ Encode structure via positional encoding

HOW CAN WE ADAPT THE TRANSFORMER TO WORK WELL ON GRAPH DATA?



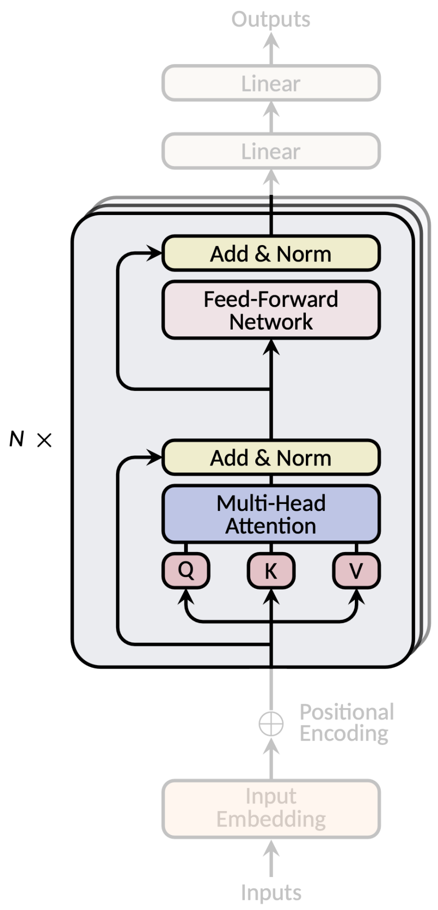
NLP

- Input tokenization
- Positional encoding
- Transformer layer with self-attention

Graph Learning

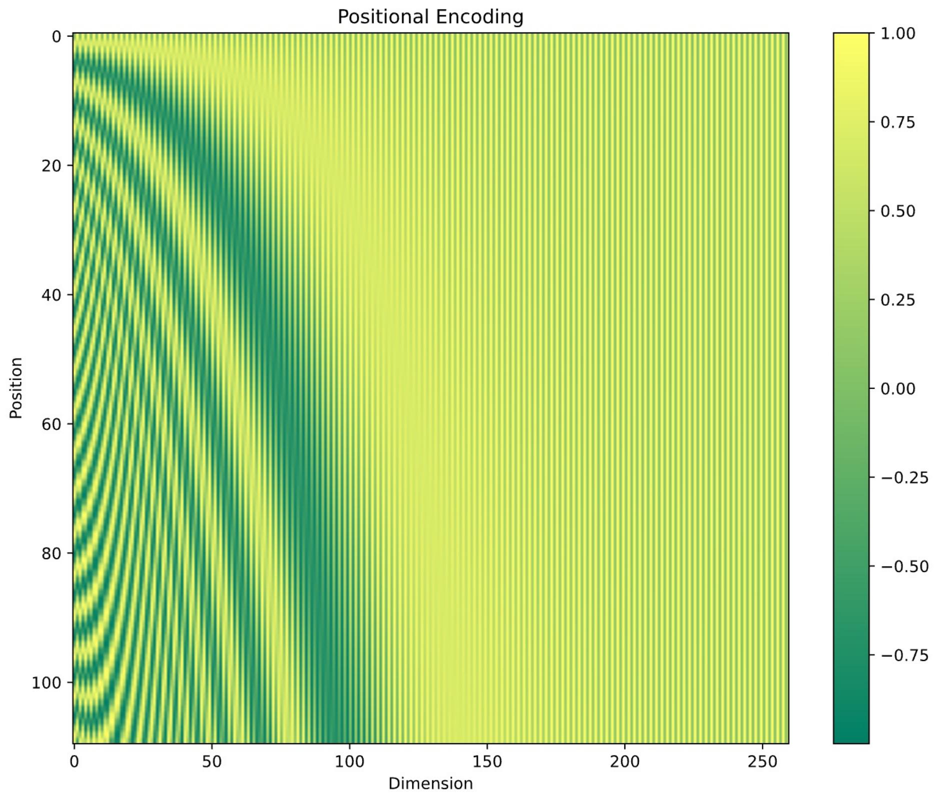
- Input treats the graph as bag of nodes
- Encode structure via positional encoding

HOW CAN WE ADAPT THE TRANSFORMER TO WORK WELL ON GRAPH DATA?



NLP	Graph Learning
❖ Input tokenization	❖ Input treats the graph as bag of nodes
❖ Positional encoding	❖ Encode structure via positional encoding
❖ Transformer layer with self-attention	❖ Transformer layer with self-attention

ABSOLUTE POSITIONAL ENCODING (APE)



PE ON GRAPHS ?

PE ON GRAPHS: LAPLACIAN POSITIONAL ENCODING

- Graph Laplacian: How does a node relate to its neighbours
- U is sorted by increasing order of the corresponding eigenvalue.
- Take the first $1, \dots, k$ eigenvectors of Δ

$$\Delta = I_n - D^{-\frac{1}{2}} A D^{\frac{1}{2}}$$

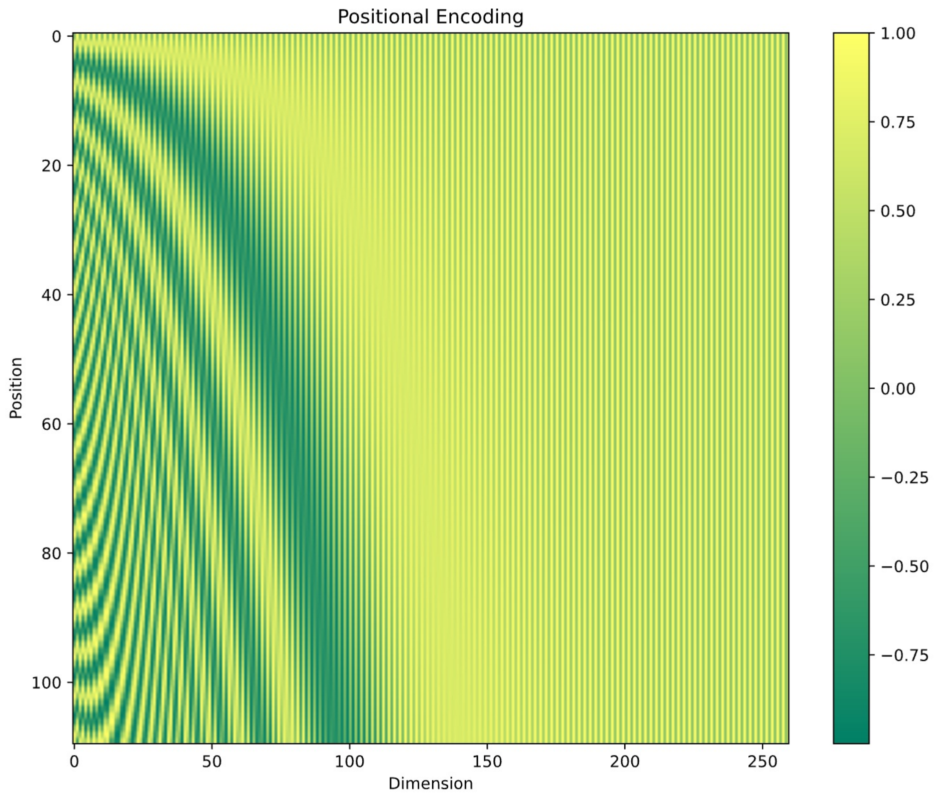
$$PE_{LPE} = [U_1, \dots, U_k]$$

PE ON GRAPHS: LAPLACIAN POSITIONAL ENCODING

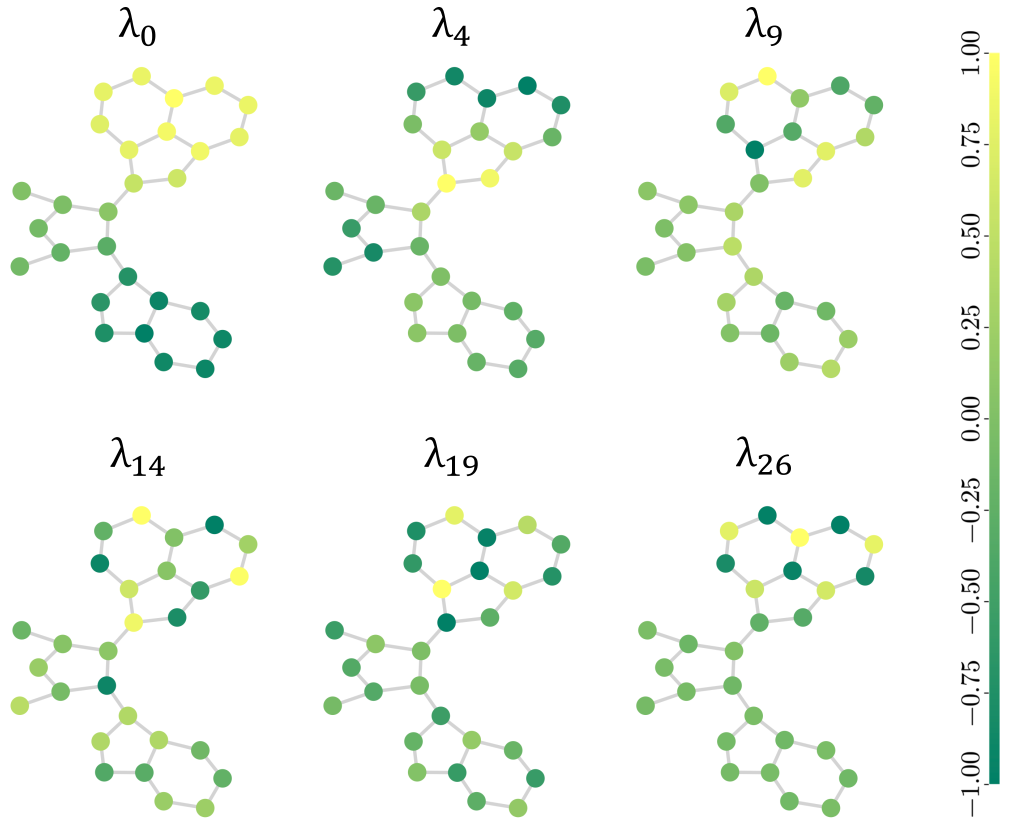
- Graph Laplacian: How does a node relate to its neighbours

DEMO

ABSOLUTE POSITIONAL ENCODING VS LAPLACIAN POSITIONAL ENCODING

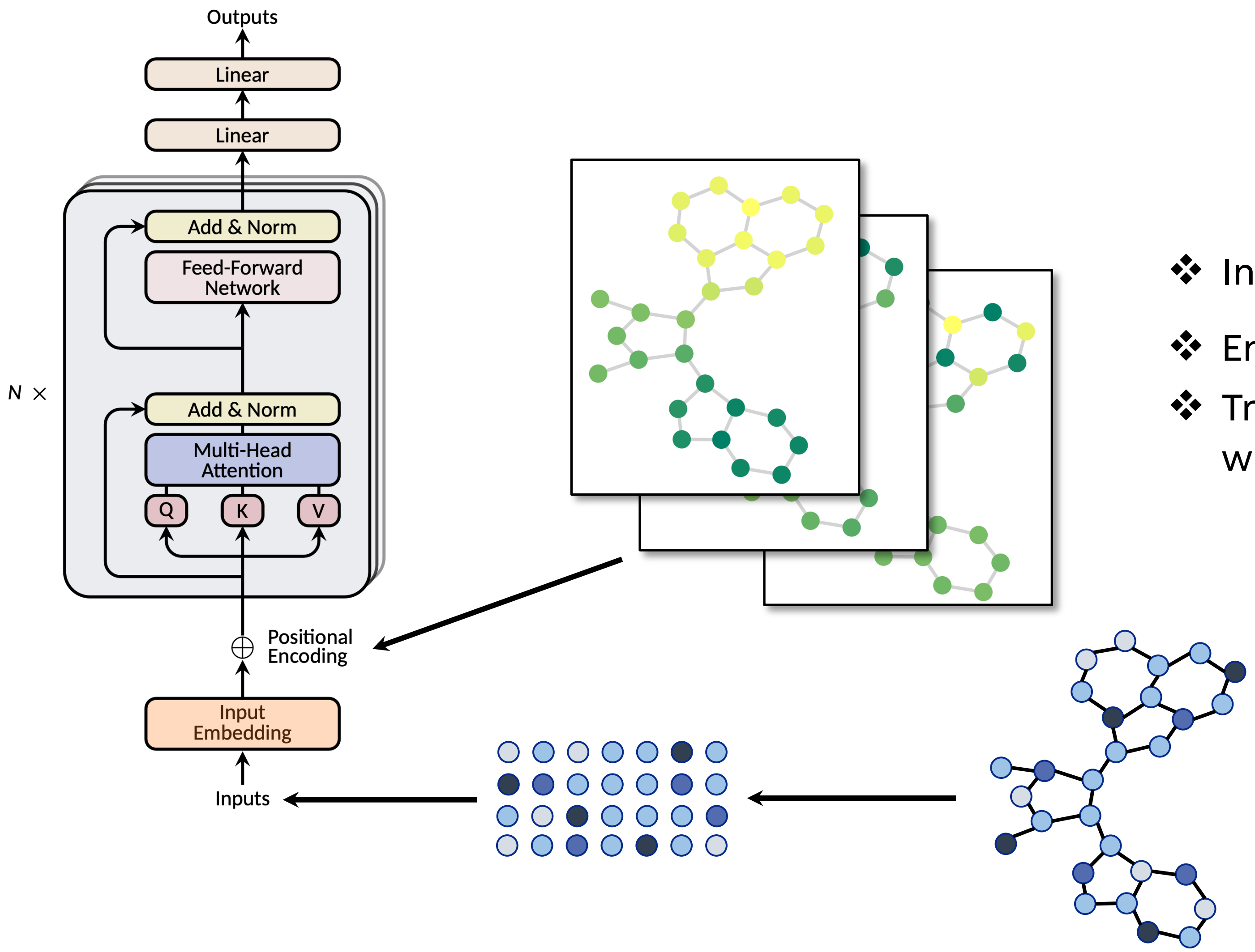


APE of a sequence



LPE of a graph

HOW CAN WE ADAPT THE TRANSFORMER TO WORK WELL ON GRAPH DATA?

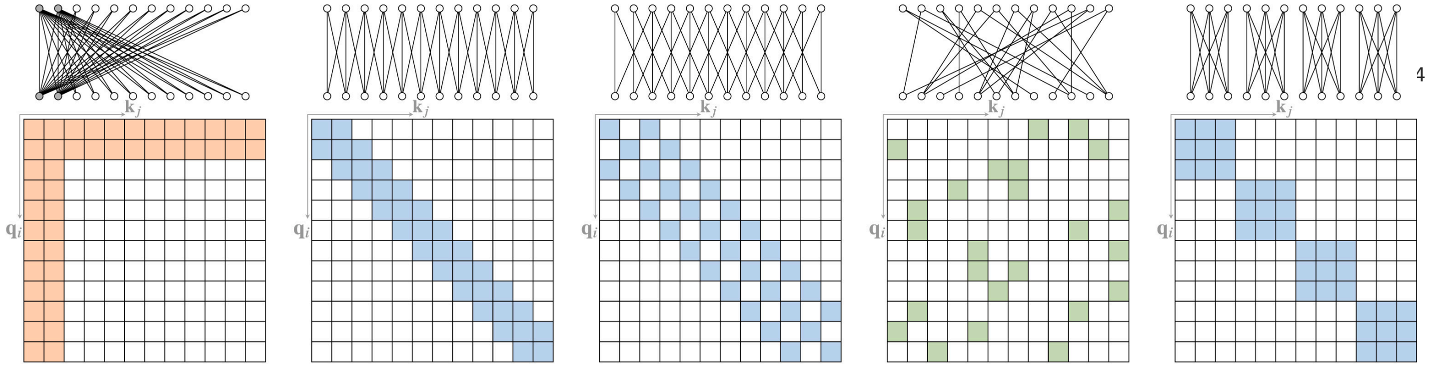


Graph Learning

- Input treats the graph as bag of nodes
- Encode structure via positional encoding
- Transformer layer with self-attention

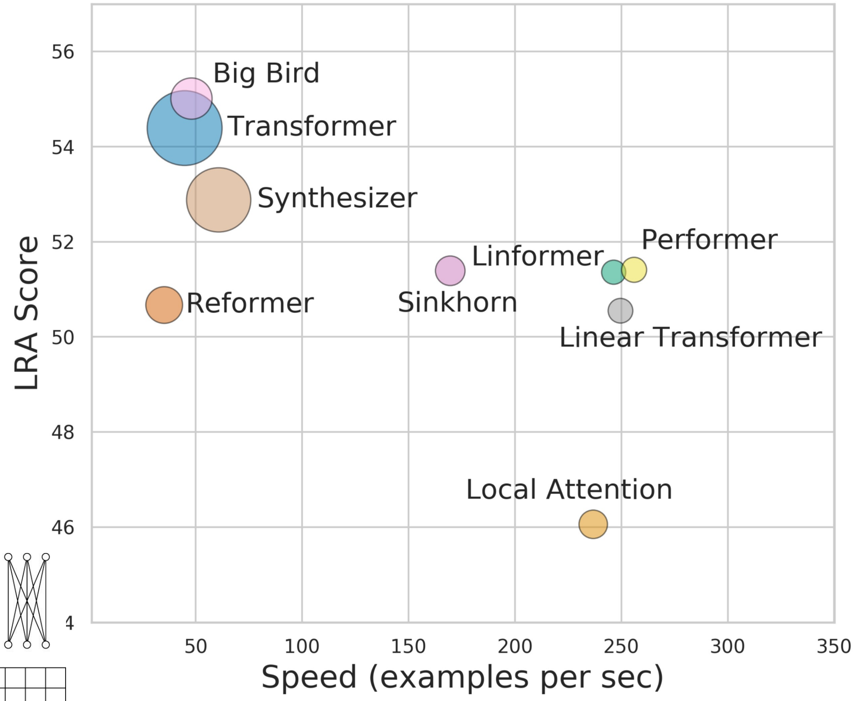
LIMITATIONS OF (GRAPH) TRANSFORMERS

- Loss of inductive bias (locality)
- Self-Attention is $\mathcal{O}(N^2)$!
- A lot of research in developing sparse attention
- Tradeoff between performance and speed



(a) global (b) band (c) dilated (d) random (e) block local

(Lin et al.,2022)



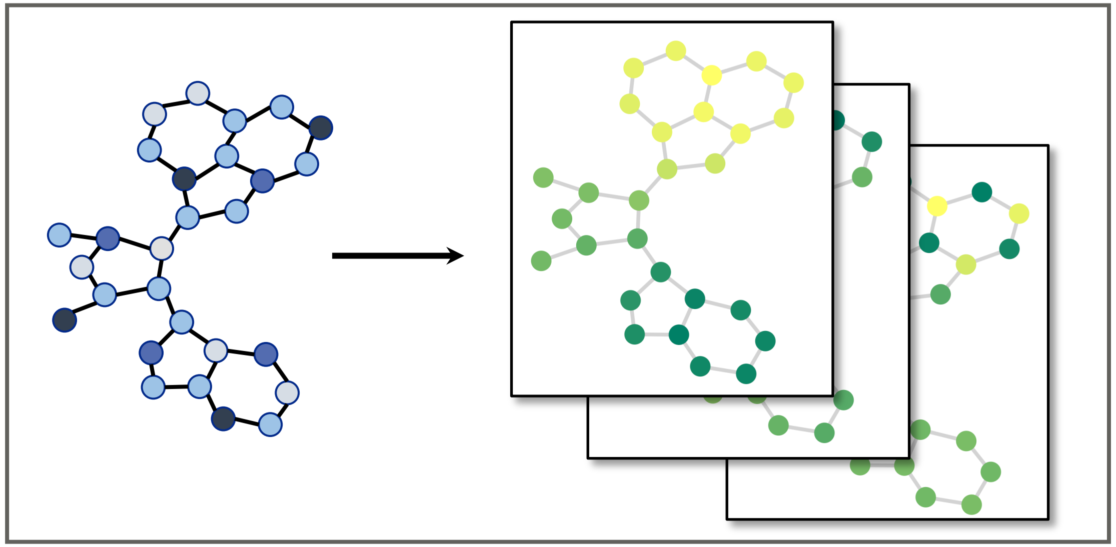
(Tay et al.,2021)

WHAT WE HAVE SEEN SO FAR

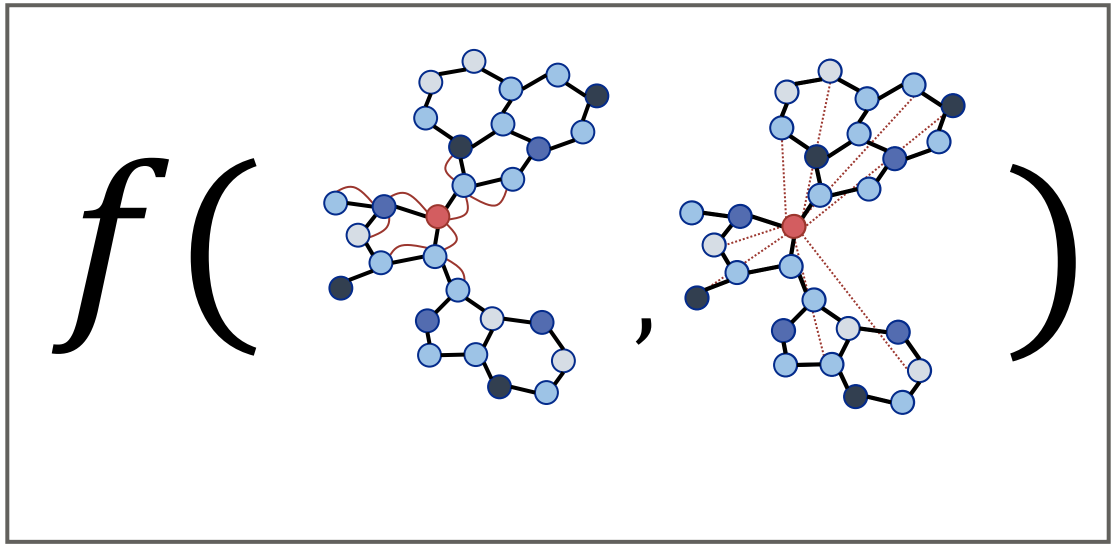
- Graph Representation Learning
- Adapting Transformers for Graph Data
- Limitations of Graph Transformers
- Now: Graph-GPS

GRAPH-GPS: GENERAL POWERFUL SCALABLE GRAPH TRANSFORMER

Positional & Structural Features

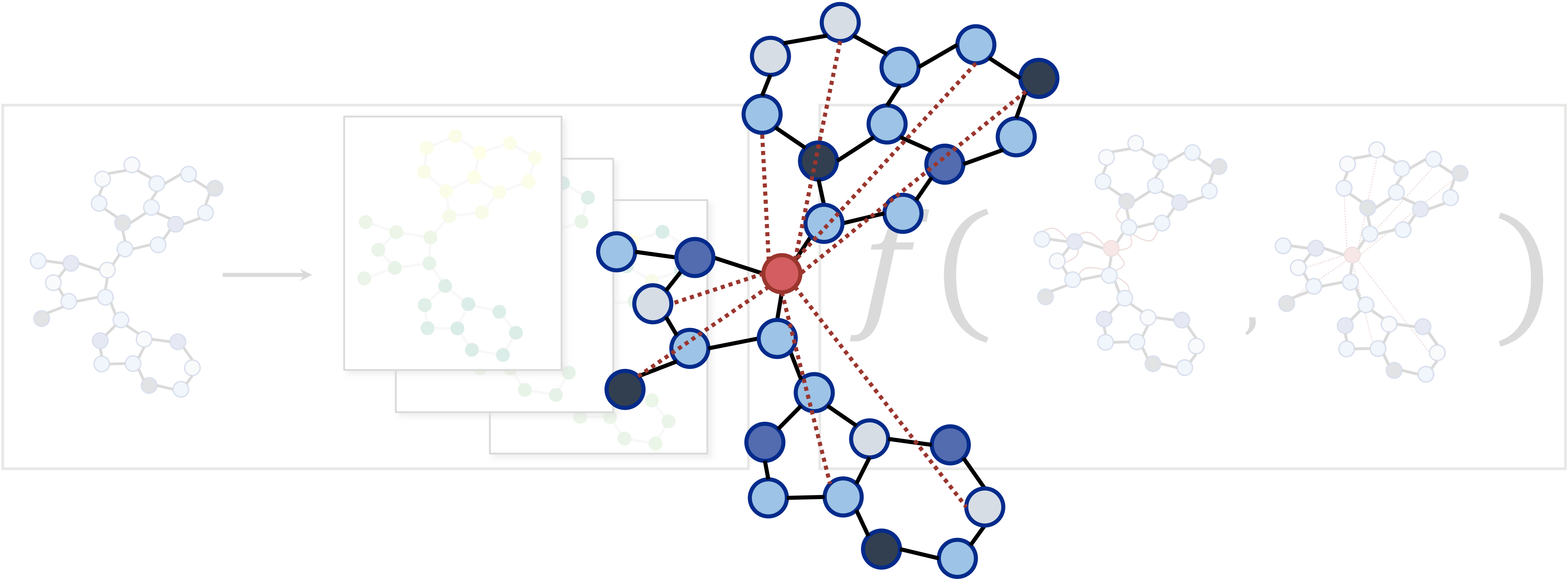


Combine MPNN and Transformer



(Rampášek et al.,2023)

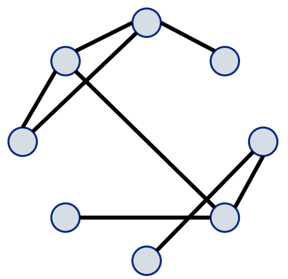
GRAPH-GPS: GENERAL POWERFUL SCALABLE GRAPH TRANSFORMER



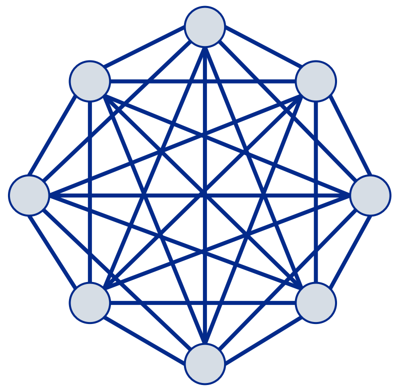
Why not using ***sparse attention*** mechanisms more tailored ***for graphs*** ?

EXPHORMER: SPARSE ATTENTION FOR GRAPHS

Original Graph



Full Attention

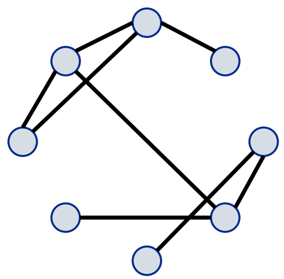


❖ Too memory intensive

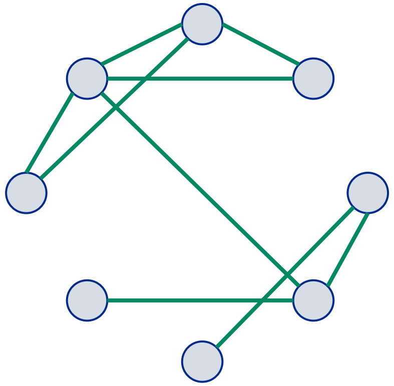
(Schirzad et al.,2023)

EXPHORMER: SPARSE ATTENTION FOR GRAPHS

Original Graph



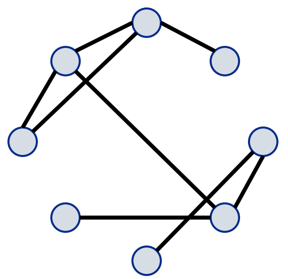
❖ Graphs carry much more topological structure than sequences



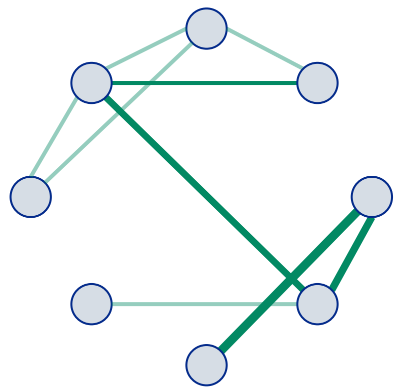
Local Neighbourhood Attention

EXPHORMER: SPARSE ATTENTION FOR GRAPHS

Original Graph



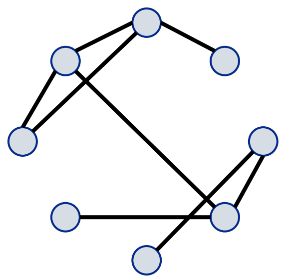
- ❖ Graphs carry much more topological structure than sequences
- ❖ Weak long-range information flow



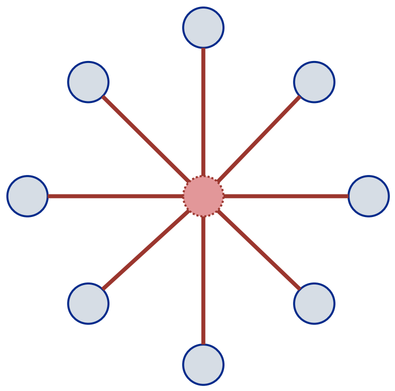
Local Neighbourhood Attention

EXPHORMER: SPARSE ATTENTION FOR GRAPHS

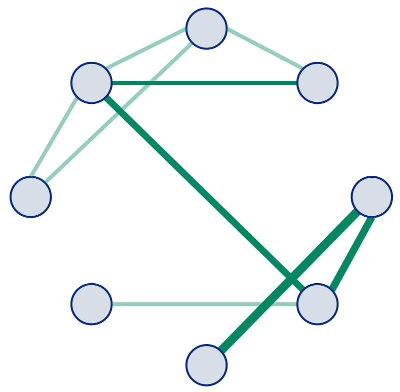
Original Graph



Global Attention



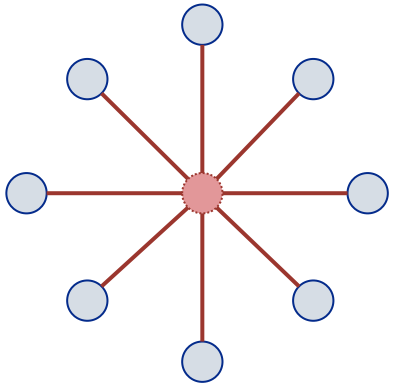
❖ Diameter 2
❖ Information bottleneck



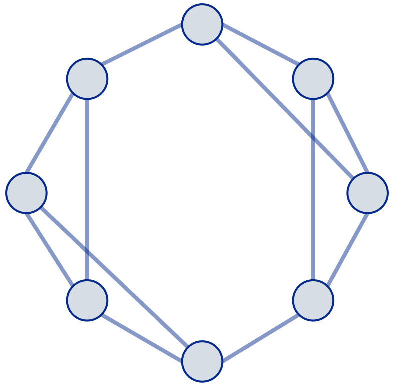
Local Neighbourhood Attention

EXPHORMER: SPARSE ATTENTION FOR GRAPHS

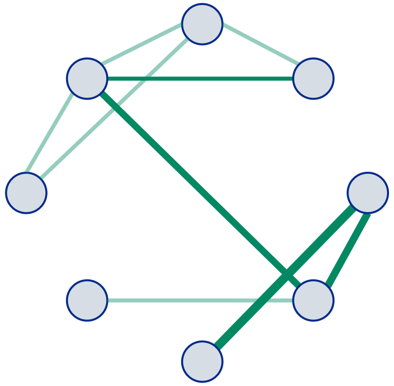
Global Attention



Expander Graph Attention



- Approximate complete graphs
  - Spectral properties
  - Mixing properties
  - Diameter is $\mathcal{O}(\log \mathrm{N})$

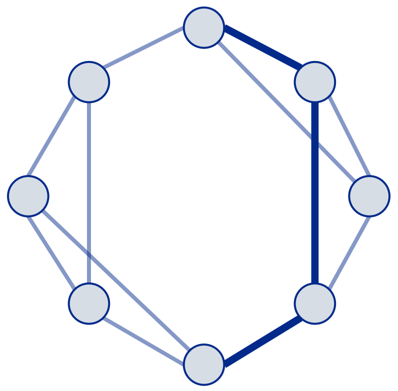


Local Neighbourhood Attention

EXPHORMER: SPARSE ATTENTION FOR GRAPHS – WHY EXPANDER GRAPHS?

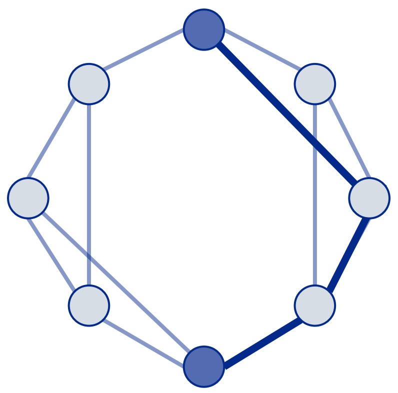
- ❖ Spectral properties:
  - A d-regular expander graph on n vertices approximates the complete graph K_n

Expander Graph Attention



EXPHORMER: SPARSE ATTENTION FOR GRAPHS – WHY EXPANDER GRAPHS?

- Spectral properties:
  - A d-regular expander graph on n vertices approximates the complete graph K_n

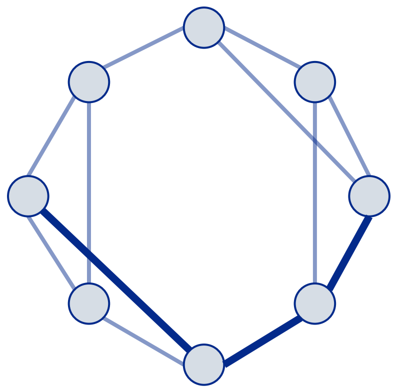


EXPHORMER: SPARSE ATTENTION FOR GRAPHS – WHY EXPANDER GRAPHS?

- ❖ Spectral properties:
  - A d-regular expander graph on n vertices approximates the complete graph K_n
  - There exist simple algorithms to create d-regular expander graphs

- ❖ Compared to K_n it has only $O(n)$ edges

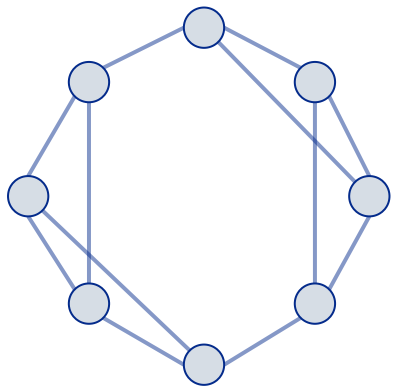
Expander Graph Attention

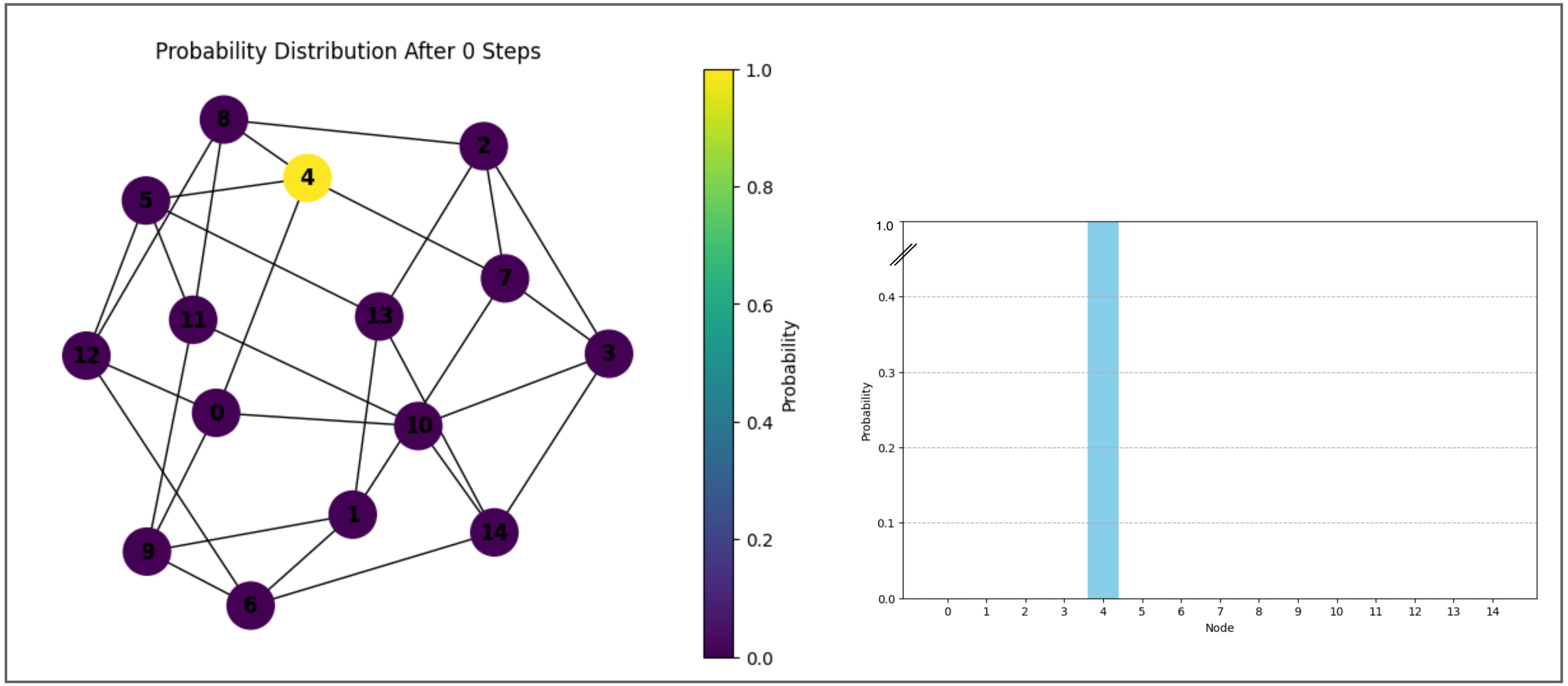


EXPHORMER: SPARSE ATTENTION FOR GRAPHS – WHY EXPANDER GRAPHS?

❖ Mixing properties:
- For a d-regular expander graph a random walk mixes well

Expander Graph Attention

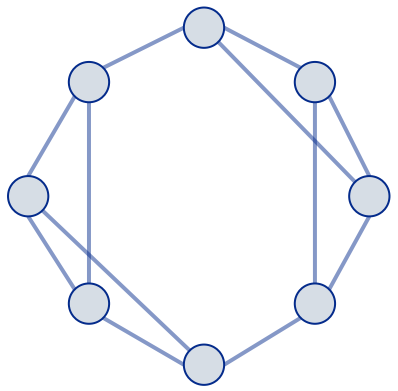


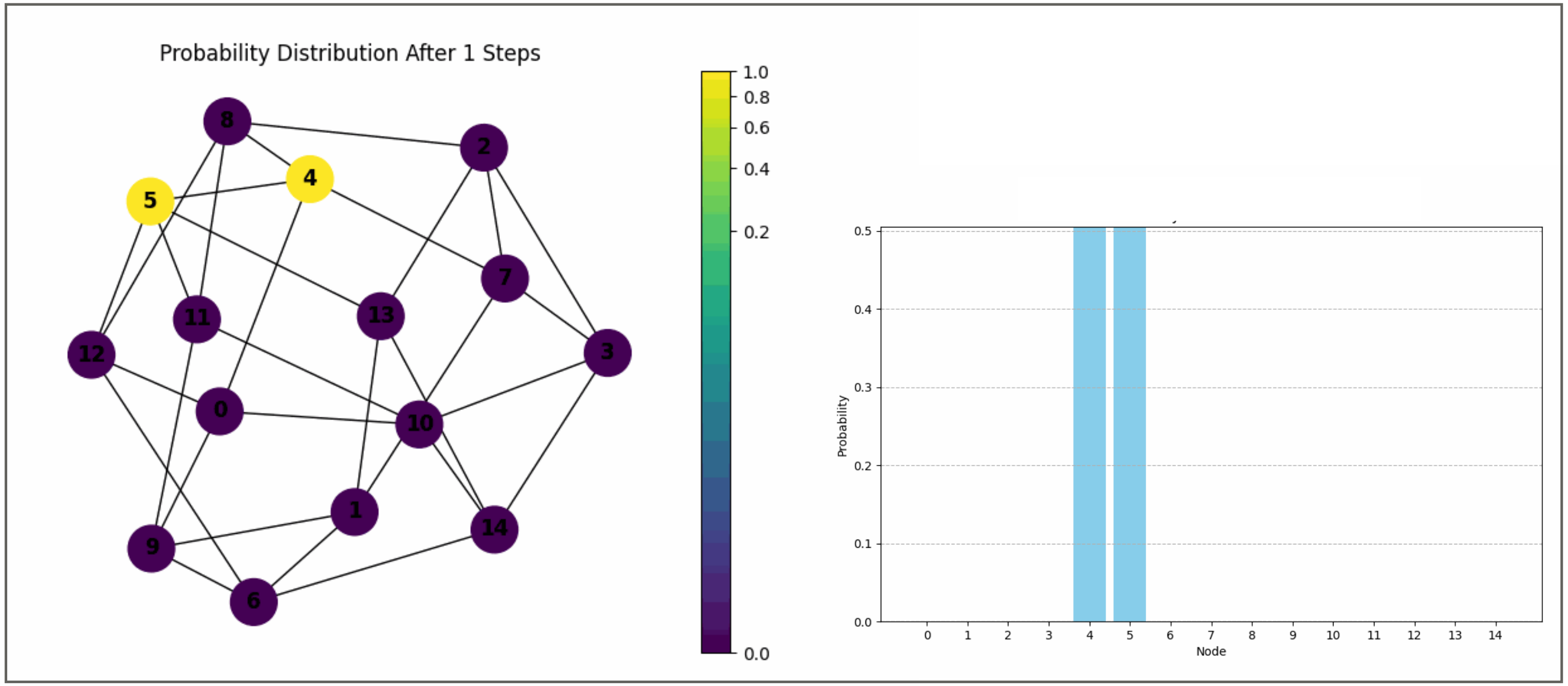


EXPHORMER: SPARSE ATTENTION FOR GRAPHS – WHY EXPANDER GRAPHS?

❖ Mixing properties:
- For a d-regular expander graph a random walk mixes well
- The diameter is $\mathcal{O}_{d,\epsilon}(\log \mathrm{N})$

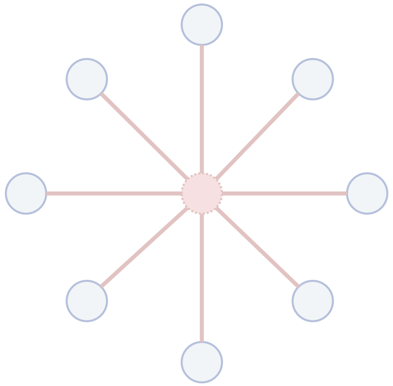
Expander Graph Attention

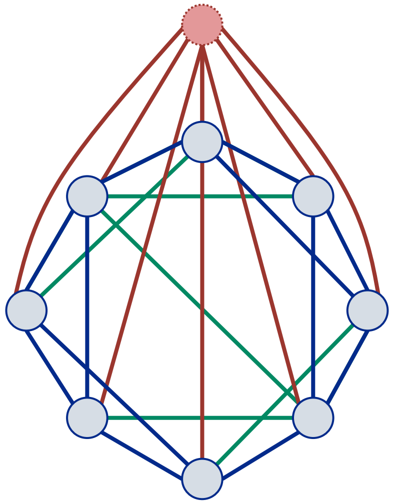




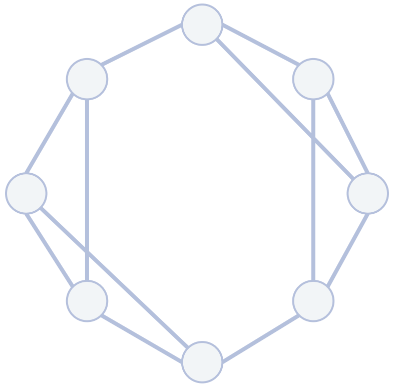
EXPHORMER: SPARSE ATTENTION FOR GRAPHS

Global Attention

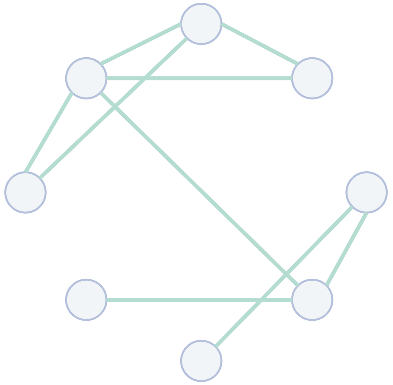




Expander Graph Attention



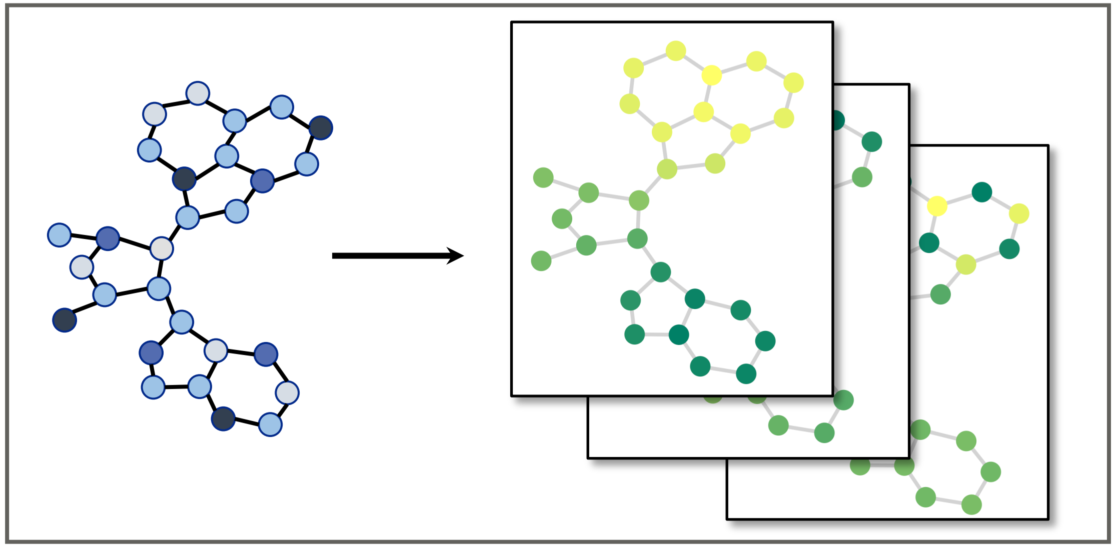
$\mathcal{O}(|V| + |E|)$ edges in the attention graph



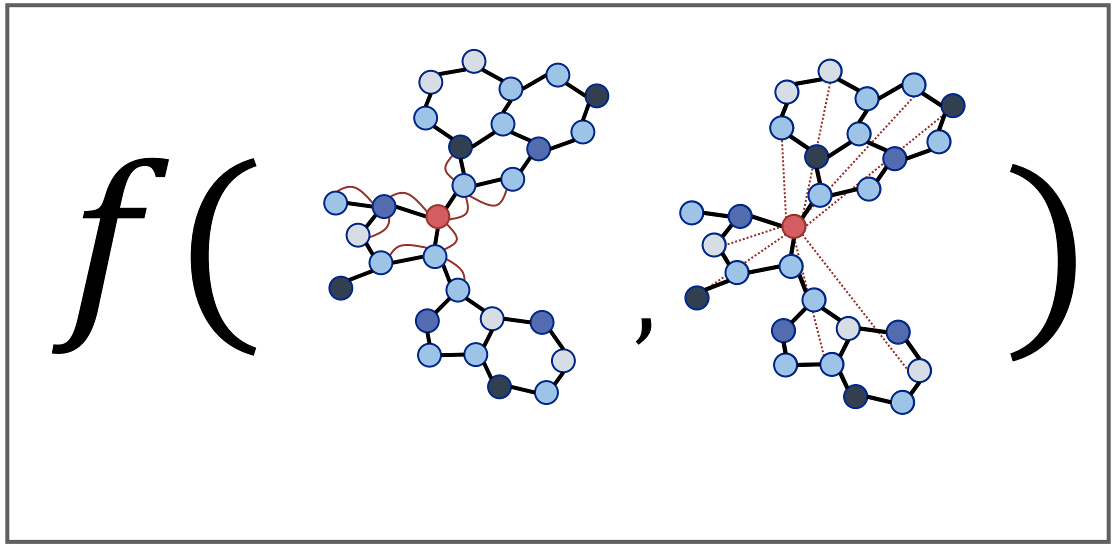
Local Neighbourhood Attention

REMEMBER GRAPH-GPS ?

Positional & Structural Features

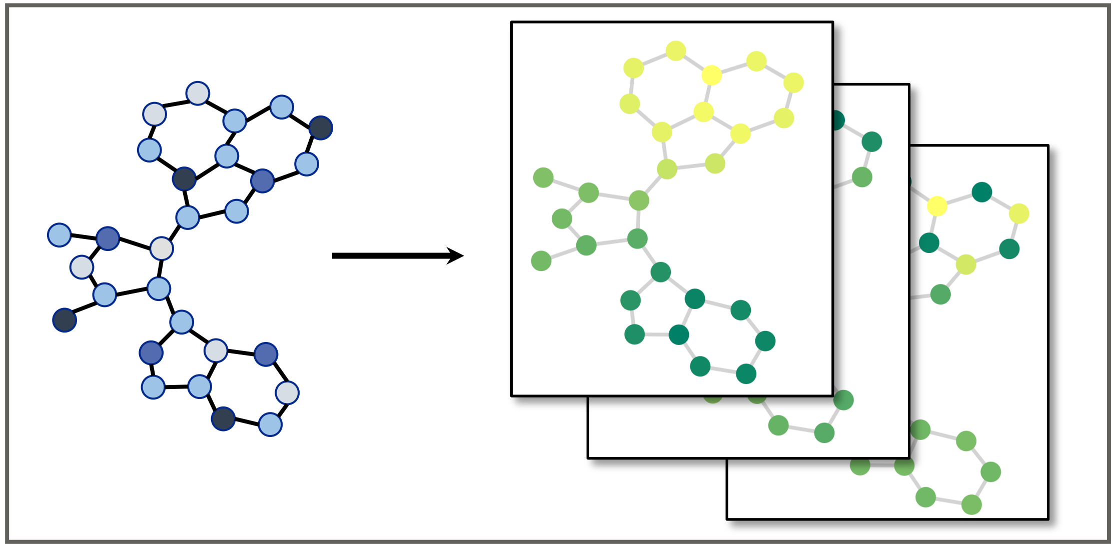


Combine MPNN and Transformer

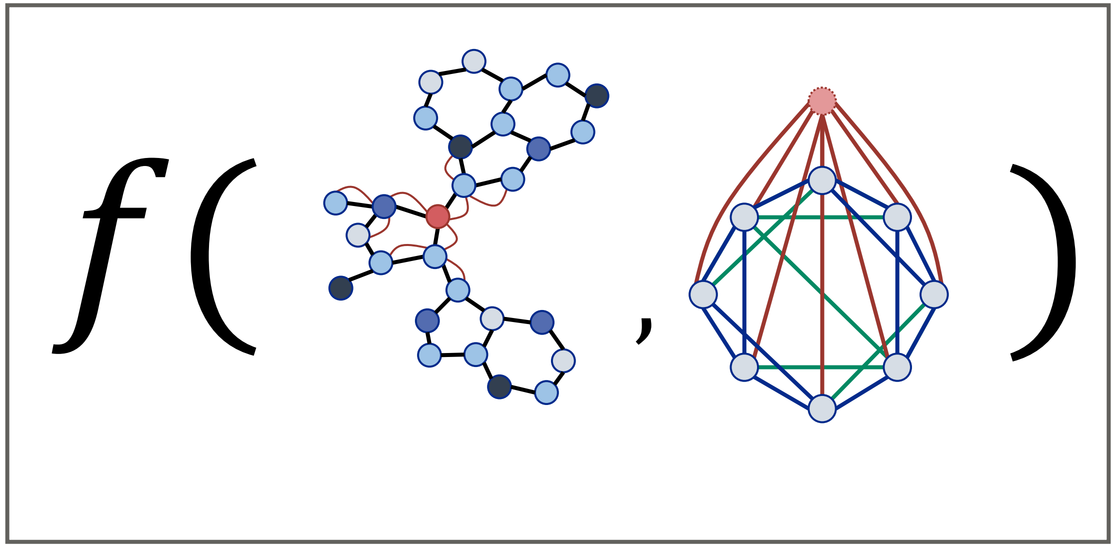


EXPHORMER: GRAPH-GPS WITH EXPANDER ATTENTION MODULE

Positional & Structural Features



Combine MPNN and Transformer



❖ EXPHORMER models have universal approximability !

EXPHORMER: EXPERIMENTS & RESULTS – BASELINES

Model	CIFAR10 Accuracy ↑	MalNet-Tiny Accuracy ↑	MNIST Accuracy ↑	CLUSTER Accuracy ↑	PATTERN Accuracy ↑
GCN (Kipf & Welling, 2017)	55.71±0.381	81.0	90.71±0.218	68.50 ± 0.976	71.89 ± 0.334
GIN (Xu et al., 2018)	55.26±1.527	88.98±0.557	96.49±0.252	64.72 ± 1.553	85.39 ± 0.136
GAT (Veličković et al., 2018)	64.22±0.455	92.1 ±0.242	95.54±0.205	70.59 ± 0.447	78.27 ± 0.186
GatedGCN (Bresson & Laurent, 2017; Dwivedi et al., 2020)	67.31±0.311	**92.23±0.65**	97.34±0.143	73.84 ± 0.326	85.57 ± 0.088
PNA (Corso et al., 2020)	70.35±0.63	–	97.94±0.12	–	–
DGN (Beaini et al., 2021)	**72.84±0.417**	–	–	–	86.68±0.034
CRaWl (Toenshoff et al., 2021)	69.01±0.259	–	97.94±0.050	–	–
GIN-AK+ (Zhao et al., 2022b)	72.19±0.13	–	–	–	**86.85±0.057**
SAN (Kreuzer et al., 2021)	–	–	–	76.69±0.65	86.58±0.037
K-Subgraph SAT (Chen et al., 2022a)	–	–	–	77.86±0.104	**86.85±0.037**
EGT (Hussain et al., 2021)	68.70±0.409		**98.17±0.087**	**79.23±0.348**	**86.82±0.020**
GraphGPS (Rampásek et al., 2022)	**72.30±0.356**	**93.50±0.41**	**98.05±0.126**	**78.02±0.180**	86.69±0.059
EXPHORMER (ours)	**74.69±0.125**	**94.02 ± 0.209**	**98.55 ± 0.039**	**78.07 ± 0.037**	86.74±0.015

- first
- second
- third

EXPHORMER: PROMISING RESULTS ON LONG-RANGE GRAPH BENCHMARKS

Model	PascalVOC-SP F1 score ↑	COCO-SP F1 score ↑	Peptides-Func AP ↑	Peptides-Struct MAE ↓	PCQM-Contact MRR ↑
GCN	0.1268 ± 0.0060	0.0841 ± 0.0010	0.5930 ± 0.0023	0.3496 ± 0.0013	0.3234 ± 0.0006
GINE	0.1265 ± 0.0076	0.1339 ± 0.0044	0.5498 ± 0.0079	0.3547 ± 0.0045	0.3180 ± 0.0027
GatedGCN	0.2873 ± 0.0219	**0.2641 ± 0.0045**	0.5864 ± 0.0077	0.3420 ± 0.0013	0.3218 ± 0.0011
GatedGCN+RWSE	0.2860 ± 0.0085	0.2574 ± 0.0034	0.6069 ± 0.0035	0.3357 ± 0.0006	0.3242 ± 0.0008
Transformer+LapPE	0.2694 ± 0.0098	0.2618 ± 0.0031	0.6326 ± 0.0126	**0.2529 ± 0.0016**	0.3174 ± 0.0020
SAN+LapPE	**0.3230 ± 0.0039**	0.2592 ± 0.0158*	0.6384 ± 0.0121	0.2683 ± 0.0043	**0.3350 ± 0.0003**
SAN+RWSE	0.3216 ± 0.0027	0.2434 ± 0.0156*	**0.6439 ± 0.0075**	0.2545 ± 0.0012	**0.3341 ± 0.0006**
GraphGPS	**0.3748 ± 0.0109**	**0.3412 ± 0.0044**	**0.6535 ± 0.0041**	**0.2500 ± 0.0005**	0.3337 ± 0.0006
Exphormer (ours)	**0.3975 ± 0.0037**	**0.3455 ± 0.0009**	**0.6527 ± 0.0043**	**0.2481 ± 0.0007**	**0.3637 ± 0.0020**

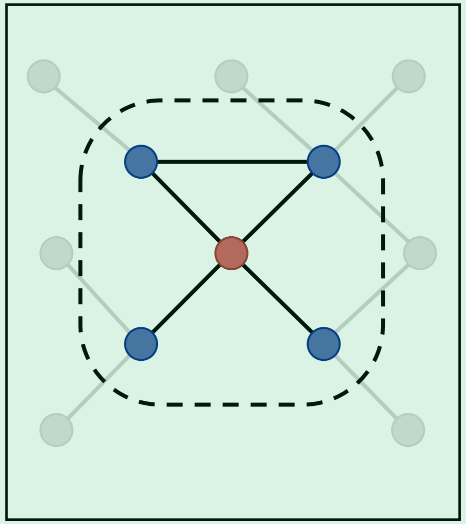
- first
- second
- third

EXPHORMER: NO ONE-FITS-ALL SOLUTION!

Dataset	No Local Edges	No Expander Edges	No Global Nodes	All Components
Cifar10	74.62 ± 0.12	74.53 ± 0.19	74.68 ± 0.19	$\mathbf{74.69 \pm 0.13}$
Malnet-Tiny	92.64 ± 0.55	$\mathbf{94.02 \pm 0.21}$	92.48 ± 0.33	92.06 ± 0.18
Pattern	86.59 ± 0.03	86.70 ± 0.02	86.14 ± 0.08	$\mathbf{86.74 \pm 0.02}$
PascalVOC-SP	0.3708 ± 0.0039	0.3588 ± 0.0013	$\mathbf{0.3975 \pm 0.0037}$	0.3682 ± 0.0042
Peptides-Struct	0.2631 ± 0.0007	$\mathbf{0.2481 \pm 0.0007}$	0.2655 ± 0.0003	0.2643 ± 0.0008
Computer	90.34 ± 0.45	91.48 ± 0.41	$\mathbf{91.59 \pm 0.31}$	91.43 ± 0.53

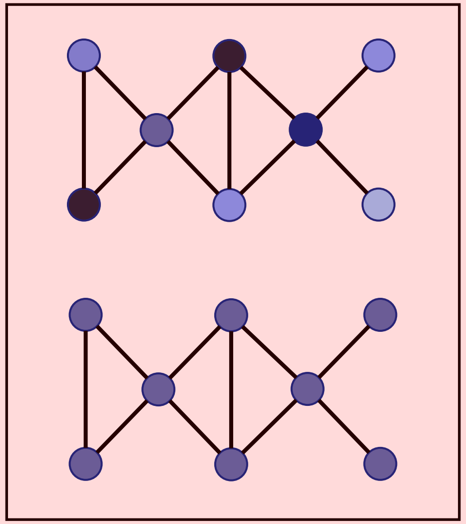
EXPHORMER: SOME IMPROVEMENTS BUT NO ONE-FITS-ALL SOLUTION!

Under-reaching



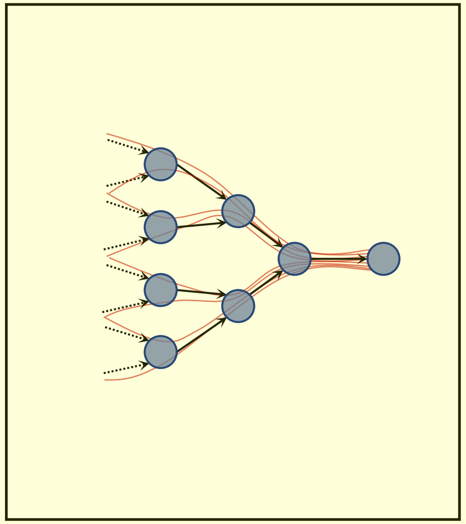
a node cannot see some nodes
(Barceló et al., 2020)

Over-smoothing



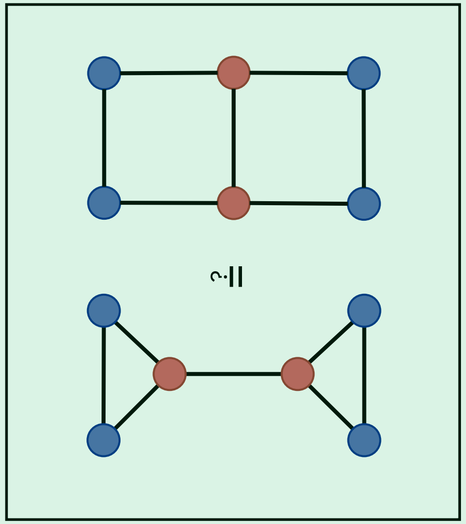
all nodes start to look *the same*
(Oono and Suzuki, 2020)

Over-squashing



information gets *"lost"* in transit
(Topping et al., 2022)

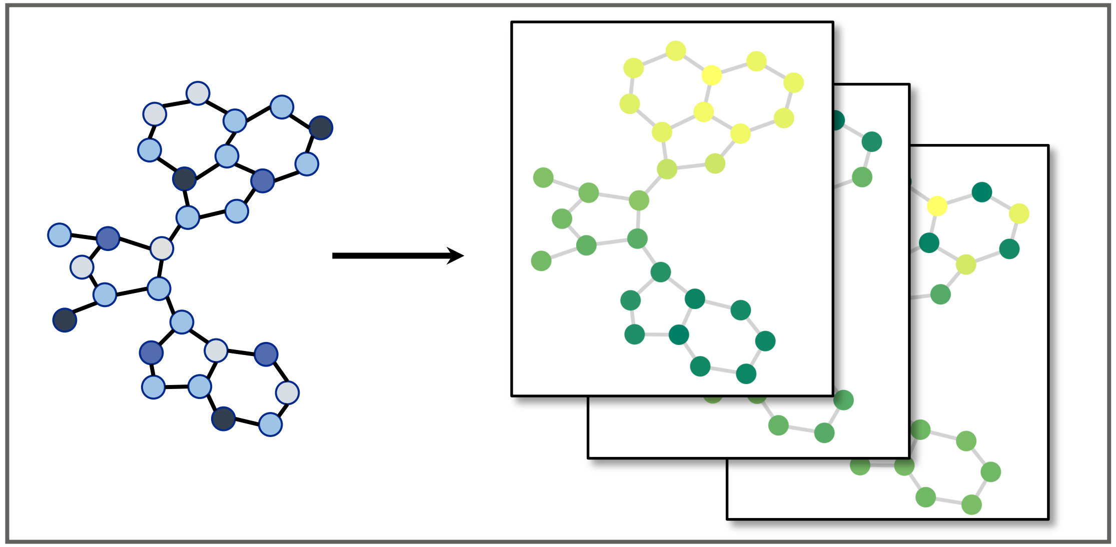
Limited Expressivity

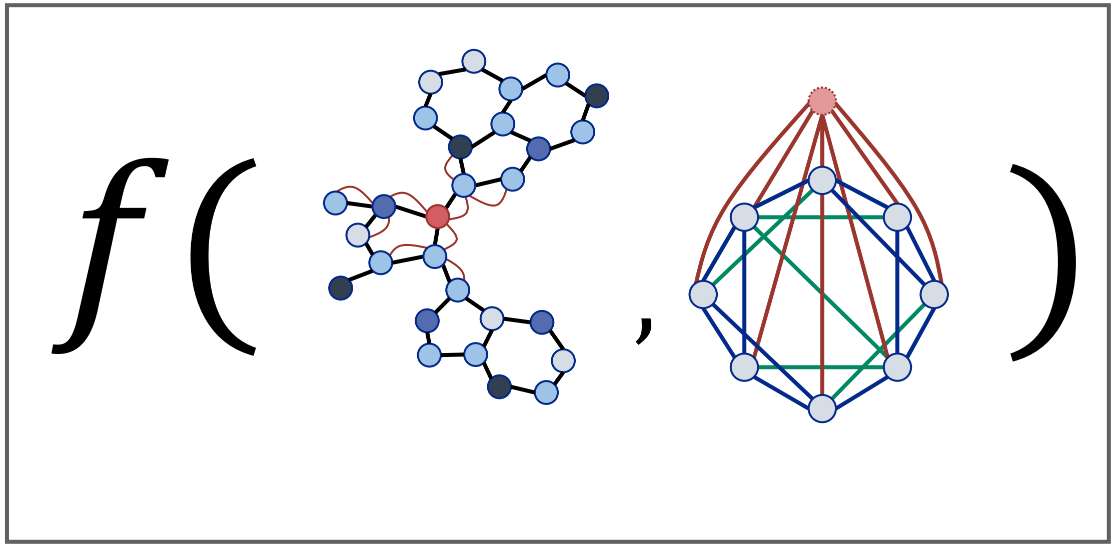


at most as expressive as 1-WL
(Morris et al., 2019)

EXPHORMER: SOME IMPROVEMENTS BUT NO ONE-FITS-ALL SOLUTION!

- Idea to use expanders as attention graphs is intuitive
- Exphormer promotes its ideas well, but in the end is just a small extension to Graph-GPS
- Need for careful tuning of the components !
- Benchmarking is hard to compare





THANK YOU FOR YOUR ATTENTION !

Johannes Herter

johannes.herter@inf.ethz.ch

BACKUP: EXPANDER GRAPH CONSTRUCTION IS SIMPLE AND FAST!

Generating a Random Regular Expander We now describe how we generate a random regular expander. Let $G = (V, E)$ be the original graph, where $V = \{1, 2, \ldots, n\}$. For the purposes of experimentation (in Tables 1 to 5), we use the random graph process analyzed in Friedman (2003) (see Theorem C.2 in Appendix C) to generate a random d-regular graph $G' = (V, E')$ on the same node set V:

- Pick $d/2$ permutations $\pi_1, \pi_2, \ldots, \pi_{d/2}$ on V, each π_i chosen independently and uniformly among all possible permutations of n elements.
- Then, letting $[k]$ denote $\{1, 2, \ldots, k\}$, choose

$$E' = \left\{(i, \pi_j(i)), (i, \pi_j^{-1}(i)) : j \in [d/2], i \in [n]\right\}.$$

Theorem C.2. *(Friedman, 2003, Theorem 1.1) Fix $\epsilon > 0$ and an even integer $d > 2$. Then, suppose $G = (V, E)$ is a random graph generated by taking $d/2$ independent uniformly random permutations $\pi_1, \pi_2, \ldots, \pi_{d/2}$ on $V = \{1, 2, \ldots, n\}$ and then choosing the edge set as*

$$E = \left\{(i, \pi_j(i)), (i, \pi_j^{-1}(i)) : 1 \le j \le d, 1 \le i \le n\right\}.$$

Then, with probability $1 - O(n^{-\Omega(\sqrt{d})})$, G satisfies $\lambda_j(G) \le 2\sqrt{d-1} + \epsilon$ for $j = 2, \ldots, n$, where $d = \lambda_1(G) \ge \lambda_2(G) \ge \cdots \ge \lambda_n(G) \ge -d$ are the eigenvalues of the adjacency matrix of G.

BACKUP: EXPANDER HAVE NICE SPECTRAL AND MIXING PROPERTIES

Theorem 4.1. *(Spielman, 2019, Section 27.2) A d-regular ϵ-expander G on n vertices spectrally approximates the complete graph K_n on n vertices:*[2]

$$(1-\epsilon)\frac{1}{n}L_K \preceq \frac{1}{d}L_G \preceq (1+\epsilon)\frac{1}{n}L_K.$$

Lemma 4.2. *(Hoory et al., 2006, Theorem 3.2) Let $G = (V, E)$ be a d-regular ϵ-expander graph on $n = |V|$ nodes. For any initial distribution $\pi^{(0)} : V \to \mathbb{R}^+$ and any $\delta > 0$, $\pi^{(t)}$ satisfies*

$$\|\pi^{(t)} - \tfrac{1}{n}\|_1 \leq \delta$$

as long as $t \geq \frac{1}{2(1-\varepsilon)} \log(n/\delta^2)$.

BACKUP: RESULTS GRAPHGPS VS EXPHORMER

Model/Dataset	**Cifar10** Accuracy ↑	**MalNet-Tiny** Accuracy ↑	**PascalVOC-SP** F1 score ↑	**Peptides-Func** AP ↑
GPS (MPNN-only)	69.948 ± 0.499	92.23 ± 0.65	0.3016 ± 0.0031	0.6159 ± 0.0048
GPS-BigBird	70.480 ± 0.106	92.34 ± 0.34	0.2762 ± 0.0069	0.5854 ± 0.0079
GPS-Performer	**70.670 ± 0.338**	**92.64 ± 0.78**	**0.3724 ± 0.0131**	**0.6475 ± 0.0056**
GPS-Transformer	**72.305 ± 0.344**	**93.50 ± 0.41**	**0.3736 ± 0.0158**	**0.6535 ± 0.0041**
EXPHORMER	**74.69±0.125**	**94.02 ± 0.21**	**0.3975 ± 0.0037**	**0.6527 ± 0.0043**

BACKUP: NO COMPARISON TO GRIT ??

Model	ZINC	MNIST	CIFAR10	PATTERN	CLUSTER
	MAE↓	Accuracy↑	Accuracy↑	Accuracy↑	Accuracy↑
GCN	0.367 ± 0.011	90.705 ± 0.218	55.710 ± 0.381	71.892 ± 0.334	68.498 ± 0.976
GIN	0.526 ± 0.051	96.485 ± 0.252	55.255 ± 1.527	85.387 ± 0.136	64.716 ± 1.553
GAT	0.384 ± 0.007	95.535 ± 0.205	64.223 ± 0.455	78.271 ± 0.186	70.587 ± 0.447
GatedGCN	0.282 ± 0.015	97.340 ± 0.143	67.312 ± 0.311	85.568 ± 0.088	73.840 ± 0.326
GatedGCN-LSPE	0.090 ± 0.001	–	–	–	–
PNA	0.188 ± 0.004	97.94 ± 0.12	70.35 ± 0.63	–	–
DGN	0.168 ± 0.003	–	72.838 ± 0.417	86.680 ± 0.034	–
GSN	0.101 ± 0.010	–	–	–	–
CIN	0.079 ± 0.006	–	–	–	–
CRaW1	0.085 ± 0.004	97.944 ± 0.050	69.013 ± 0.259	–	–
GIN-AK+	0.080 ± 0.001	–	72.19 ± 0.13	86.850 ± 0.057	–
SAN	0.139 ± 0.006	–	–	86.581 ± 0.037	76.691 ± 0.65
Graphormer	0.122 ± 0.006	–	–	–	–
K-Subgraph SAT	0.094 ± 0.008	–	–	86.848 ± 0.037	77.856 ± 0.104
EGT	0.108 ± 0.009	98.173 ± 0.087	68.702 ± 0.409	86.821 ± 0.020	79.232 ± 0.348
Graphormer-URPE	0.086 ± 0.007	–	–	–	–
Graphormer-GD	0.081 ± 0.009	–	–	–	–
GPS	0.070 ± 0.004	98.051 ± 0.126	72.298 ± 0.356	86.685 ± 0.059	78.016 ± 0.180
GRIT (ours)	0.059 ± 0.002*	98.108 ± 0.111	76.468 ± 0.881*	87.196 ± 0.076*	80.026 ± 0.277*

Model	CIFAR10 Accuracy ↑	MalNet-Tiny Accuracy ↑	MNIST Accuracy ↑	CLUSTER Accuracy ↑	PATTERN Accuracy ↑
GCN (Kipf & Welling, 2017)	55.71±0.381	81.0	90.71±0.218	68.50 ± 0.976	71.89 ± 0.334
GIN (Xu et al., 2018)	55.26±1.527	88.98±0.557	96.49±0.252	64.72 ± 1.553	85.39 ± 0.136
GAT (Veličković et al., 2018)	64.22±0.455	92.1 ±0.242	95.54±0.205	70.59 ± 0.447	78.27 ± 0.186
GatedGCN (Bresson & Laurent, 2017; Dwivedi et al., 2020)	67.31±0.311	92.23±0.65	97.34±0.143	73.84 ± 0.326	85.57 ± 0.088
PNA (Corso et al., 2020)	70.35±0.63	–	97.94±0.12	–	–
DGN (Beaini et al., 2021)	72.84±0.417	–	–	–	86.68±0.034
CRaWl (Toenshoff et al., 2021)	69.01±0.259	–	97.94±0.050	–	–
GIN-AK+ (Zhao et al., 2022b)	72.19±0.13	–	–	–	86.85±0.057
SAN (Kreuzer et al., 2021)	–	–	–	76.69±0.65	86.58±0.037
K-Subgraph SAT (Chen et al., 2022a)	–	–	–	77.86±0.104	86.85±0.037
EGT (Hussain et al., 2021)	68.70±0.409		98.17±0.087	79.23±0.348	86.82±0.020
GraphGPS (Rampásek et al., 2022)	72.30±0.356	93.50±0.41	98.05±0.126	78.02±0.180	86.69±0.059
EXPHORMER (ours)	74.69±0.125	94.02 ± 0.209	98.55 ± 0.039	78.07 ± 0.037	86.74±0.015