Graph Transformers

Dr. Leslie O'Bray
October 18, 2023

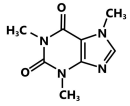
Outline of the lecture

1. (Very brief) Context setting
2. Recap of GNNs & local methods
3. Limitations of modeling local information
4. Transformers & graph transformers
5. Structure-Aware Transformer

Graphs are widely applicable across domains

Social networks

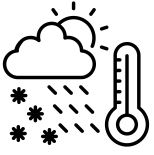
Chemoinformatics



Product recommendations

Traffic prediction

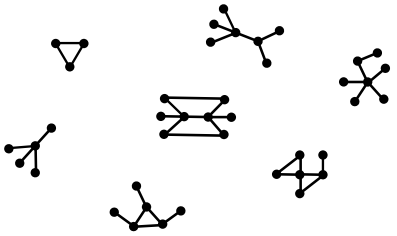
Weather forecasting



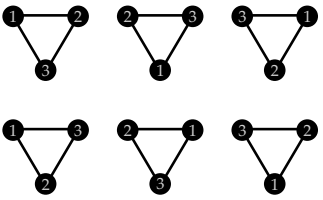
Protein-protein associations

The flexibility of graphs presents unique challenges

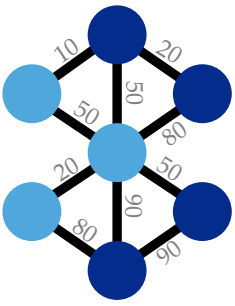
Variable sizes



No canonical ordering



Graph definitions and scope

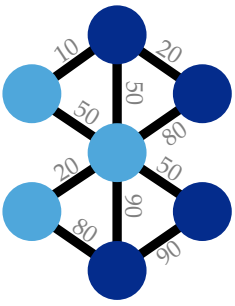


Today's lecture is concerned primarily with:

- Graph classification
- Simple graphs
- Static graphs
- Homogeneous graphs

- $G = (V, E)$
- Node labels or attributes $\mathbf{X}$
- (Optional) edge attributes or weights

Graph definitions and scope



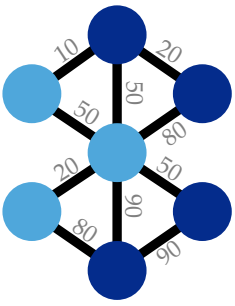
Today's lecture is concerned primarily with:

- Graph classification
- Simple graphs
- Static graphs
- Homogeneous graphs

- $G = (V, E)$
- Node labels or attributes $\mathbf{X}$
- (Optional) edge attributes or weights

...but can be extended to other types of graphs!

Graph definitions and scope



Today's lecture is concerned primarily with:

- ❄ Graph classification
- ❄ Simple graphs
- ❄ Static graphs
- ❄ Homogeneous graphs

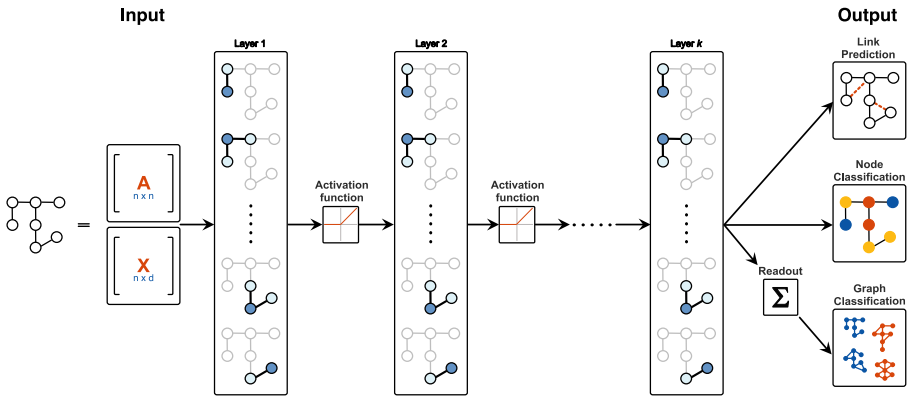
- ❄ $G = (V, E)$
- ❄ Node labels or attributes $\mathbf{X}$
- ❄ (Optional) edge attributes or weights

...but can be extended to other types of graphs!

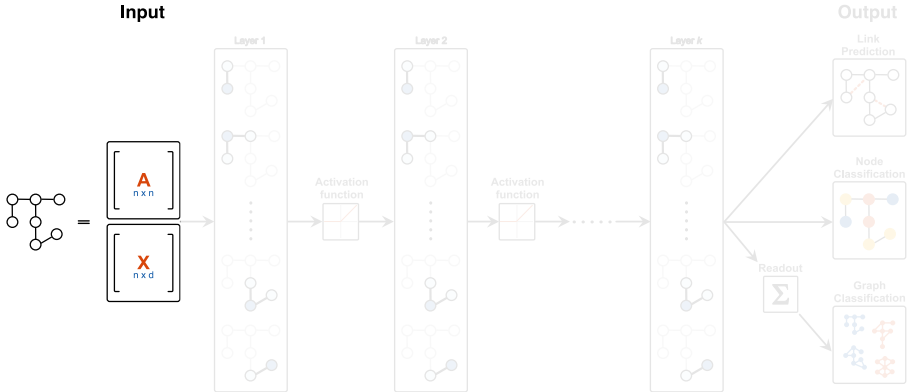
Graph representation learning: how to effectively incorporate the graph structure?

Recap: local message passing GNNs

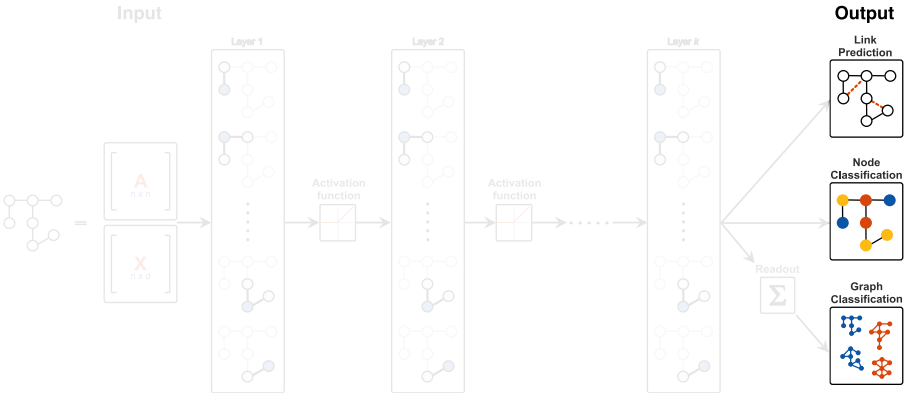
Current paradigm uses neighborhood aggregation



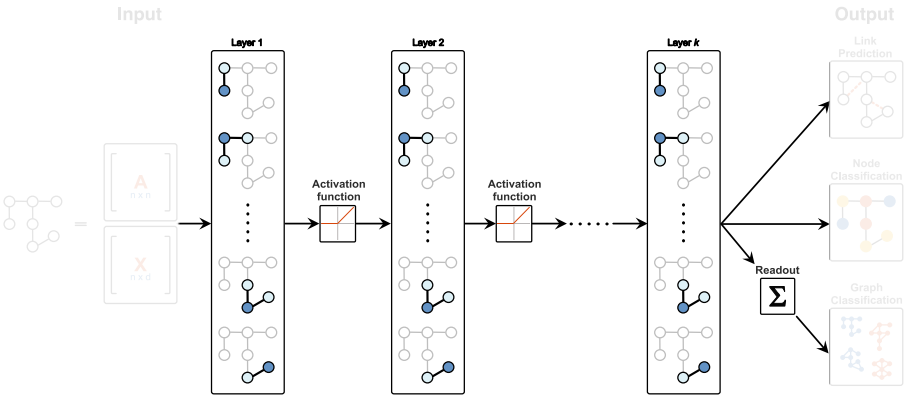
Current paradigm uses neighborhood aggregation



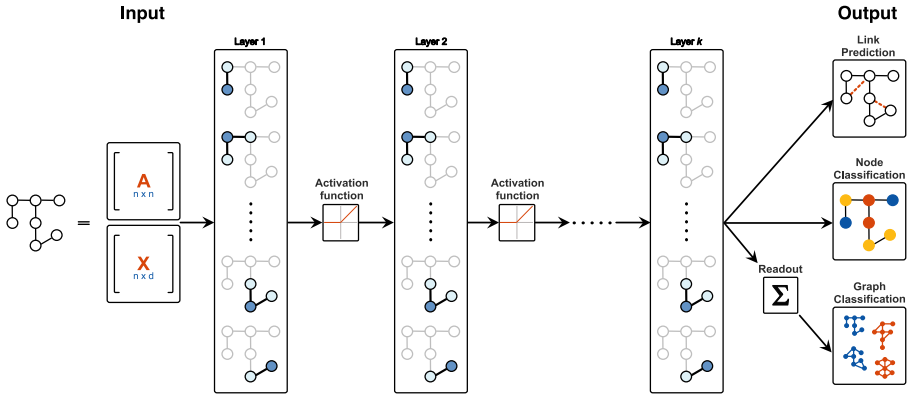
Current paradigm uses neighborhood aggregation



Current paradigm uses neighborhood aggregation



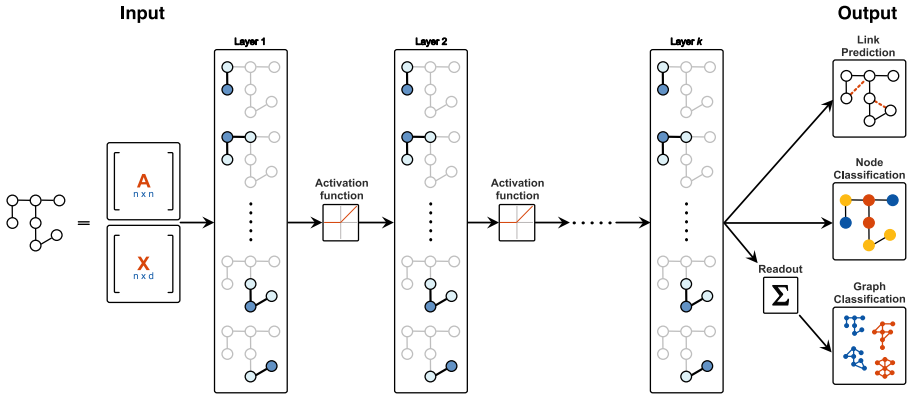
Current paradigm uses neighborhood aggregation



The representation of node u at layer l, h_u^l, is updated by:

$$a_u^l = \texttt{aggregate}\left(\left\{h_v^{l-1}\right\}_{v \in \mathcal{N}_1(u)}\right)$$

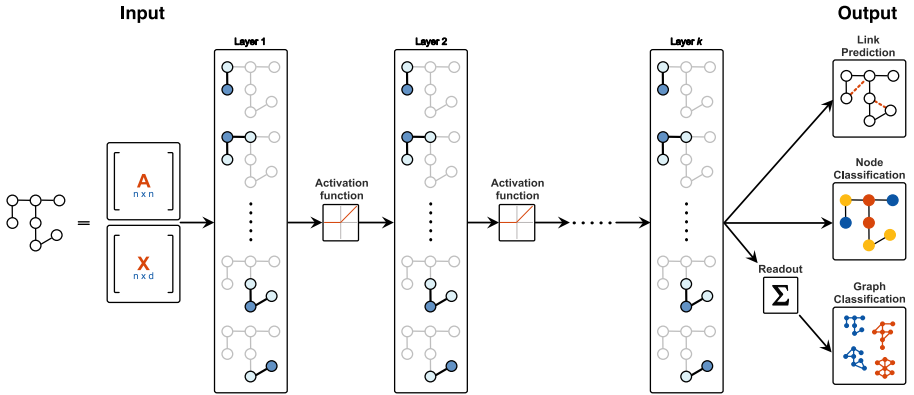
Current paradigm uses neighborhood aggregation



The representation of node u at layer l, h_u^l, is updated by:

$$a_u^l = \texttt{aggregate}\left(\left\{h_v^{l-1}\right\}_{v \in \mathcal{N}_1(u)}\right) \qquad h_u^l = \texttt{combine}\left(a_u^l, h_u^{l-1}\right)$$

Current paradigm uses neighborhood aggregation

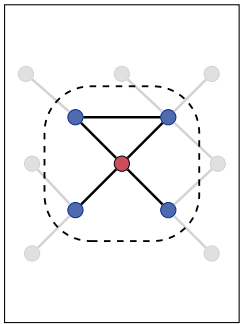


The representation of node u at layer l, h_u^l, is updated by:

$$a_u^l = \texttt{aggregate}\left(\left\{h_v^{l-1}\right\}_{v\in\mathcal{N}_1(u)}\right) \qquad h_u^l = \texttt{combine}\left(a_u^l, h_u^{l-1}\right) \qquad h_G = \texttt{readout}\left(\left\{h_v^k\right\}_{v\in G}\right)$$

GNNs incorporate a local view of the graph, which can cause issues

Under-reaching

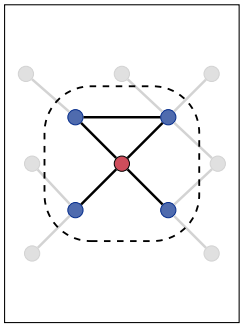


a node cannot *see* some nodes

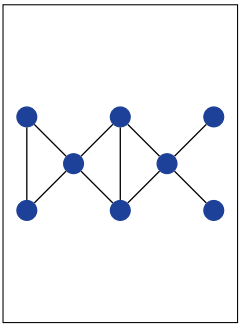
(Barceló et al., 2020)

GNNs incorporate a local view of the graph, which can cause issues

Under-reaching



Over-smoothing

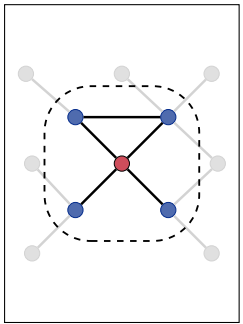


all nodes start to look *the same*

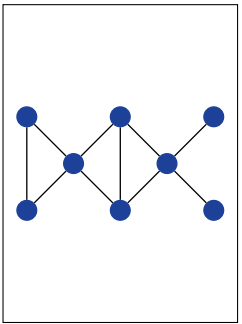
(Oono and Suzuki, 2020)

GNNs incorporate a local view of the graph, which can cause issues

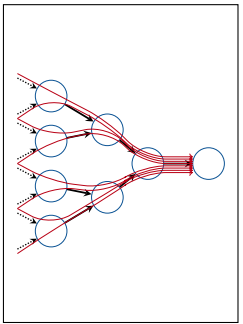
Under-reaching



Over-smoothing



Over-squashing

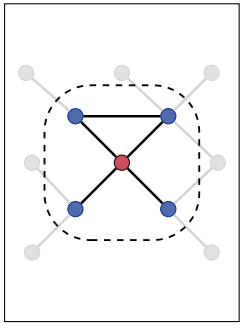


information gets *"lost"* in transit

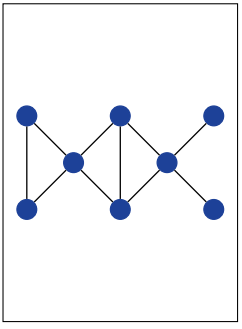
(Topping et al., 2022)

GNNs incorporate a local view of the graph, which can cause issues

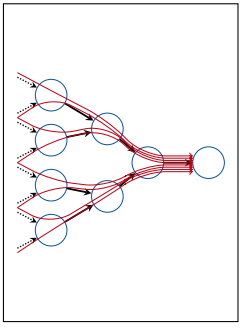
Under-reaching



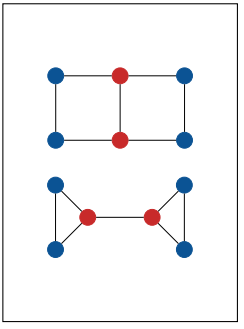
Over-smoothing



Over-squashing



Limited expressivity



at most as expressive as 1-WL

(Morris et al., 2019)

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

Today's focus: 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 modeling (Rives et al., 2021)

Today's focus: 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
  - Models global information by design
  - A transformer is a fully-connected graph
  - Attention enables message passing between all nodes

Today's focus: integrating global information via transformers

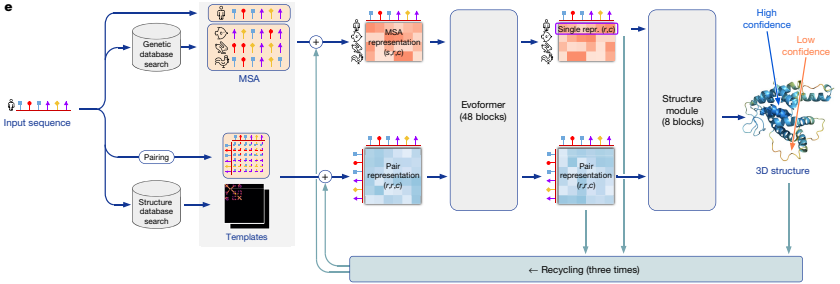
Why use transformers?

- 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

Today's focus: integrating global information via transformers

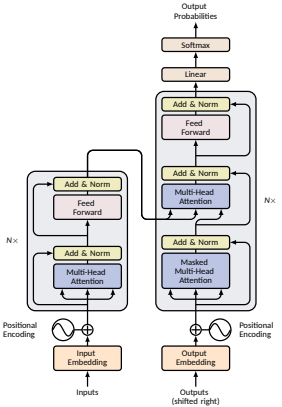
Why use transformers?

- 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
- Seen real world examples where global information is informative

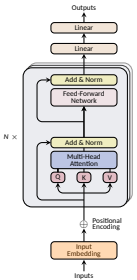


Types of transformer architectures

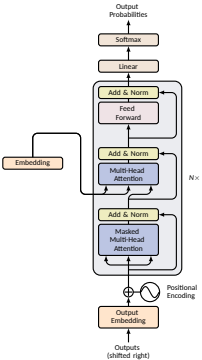
Encoder-Decoder



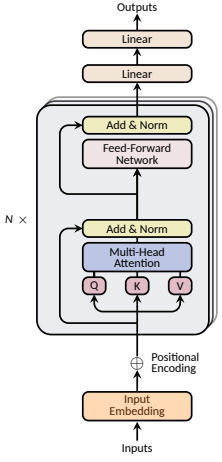
Encoder-only



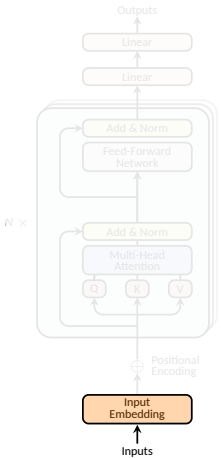
Decoder-only



Components of a transformer



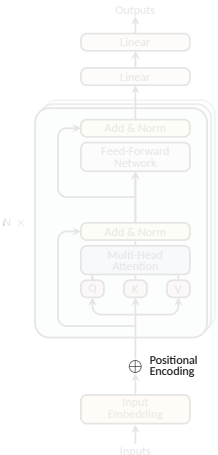
Components of a transformer



Architecture of a transformer

- Input tokenization

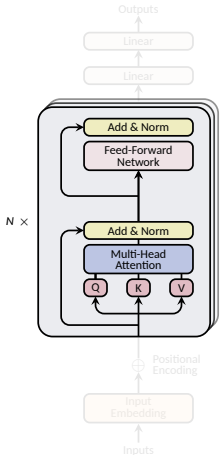
Components of a transformer



Architecture of a transformer

- Input tokenization
- Positional encoding

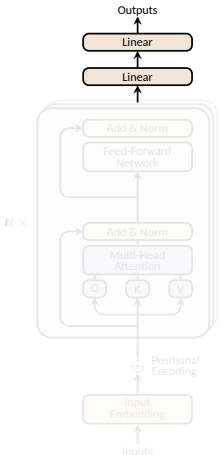
Components of a transformer



Architecture of a transformer

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

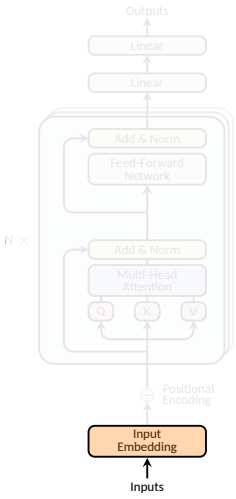
Components of a transformer



Architecture of a transformer

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

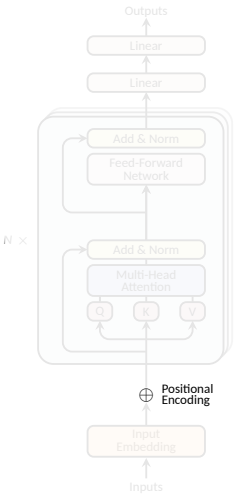
Components of a transformer: Tokenization



Tokenization & embedding

- Uses a dictionary to convert a word to a token
- "The dog sat by the door" $\rightarrow [5\ 129\ 87\ 902\ 5\ 398]$
- Then uses an embedding matrix to convert from tokens to a d-dim vector

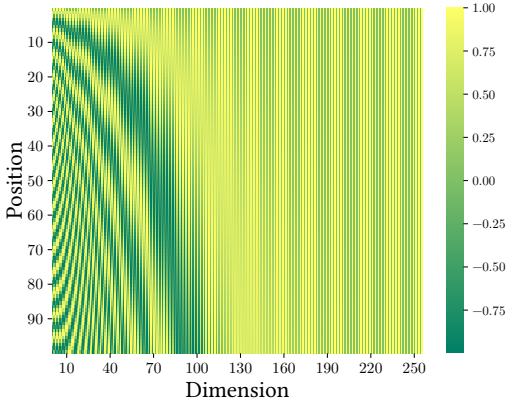
Components of a transformer: Positional encoding



Positional encoding

- Captures a word's position in a text
- Is added or concatenated to the input

Absolute positional encoding

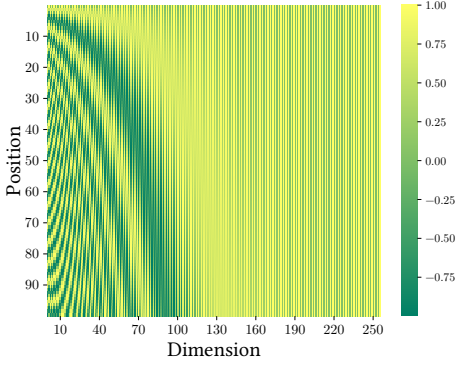


$$\mathrm{PE}_{p,2i} = \sin\left(\frac{\mathrm{p}}{10000^{\frac{2i}{d}}}\right)$$

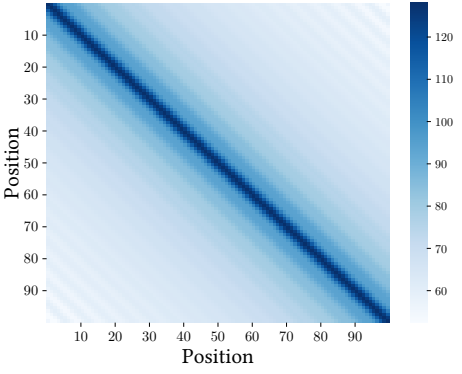
$$\mathrm{PE}_{p,2i+1} = \cos\left(\frac{\mathrm{p}}{10000^{\frac{2i}{d}}}\right)$$

- p: index position of the word
- i: dimension in PE vector
- d: dimension of the model

Absolute positional encoding

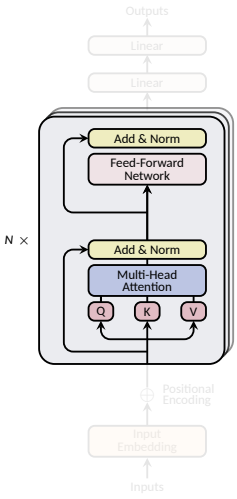


(a) APE



(b) Attention matrix

Components of a transformer: Transformer layer with self-attention

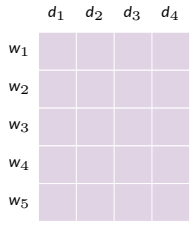


Transformer layer with self-attention

- Input is projected to $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ matrices
- Passes through (multi-head) attention
- Residual connections + normalization
- Passes through MLPs

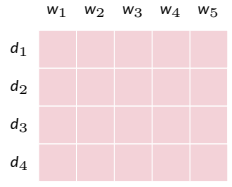
What is self-attention: The components

Q: Query matrix



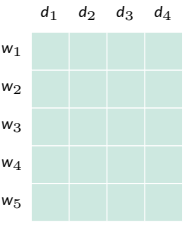
$$\mathbf{Q} = \mathbf{X}\mathbf{W}_\mathbf{Q} \in \mathbb{R}^{n \times d}$$

K: Key matrix



$$\mathbf{K} = \mathbf{X}\mathbf{W}_\mathbf{K} \in \mathbb{R}^{n \times d}$$

V: Value matrix

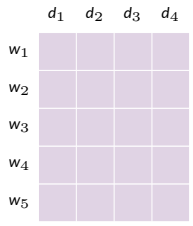


$$\mathbf{V} = \mathbf{X}\mathbf{W}_\mathbf{V} \in \mathbb{R}^{n \times d}$$

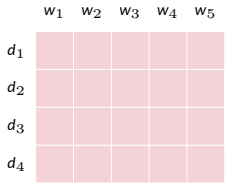
where $|\#\ \textit{words}| = n$, $\mathbf{X} \in \mathbb{R}^{n \times d}$, and $\mathbf{W}_\mathbf{Q}, \mathbf{W}_\mathbf{K}, \mathbf{W}_\mathbf{V} \in \mathbb{R}^{d \times d}$

What is self-attention: The components

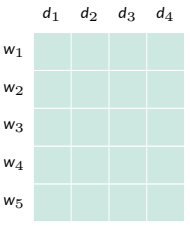
Q: Query matrix



K: Key matrix



V: Value matrix



Interpretation of the different matrices:

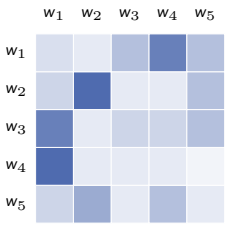
To update an embedding x_i^l at layer l:

- q_i: the embedding of a word when it receives information from the other nodes
- k_j: the embedding of a word when it is sending information to update other nodes
- v_j: the embedding actually used in the update of each word

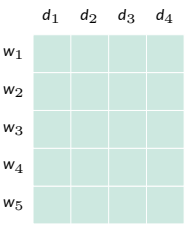
What is self-attention: The intuition

$$\text{Attn}(\mathbf{X}) := \text{softmax}\left(\frac{\mathbf{QK^T}}{\sqrt{\mathbf{d}}}\right)\mathbf{V}$$

Attention matrix

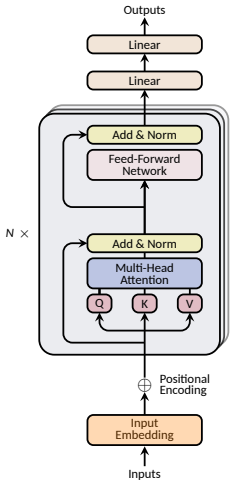


Value matrix



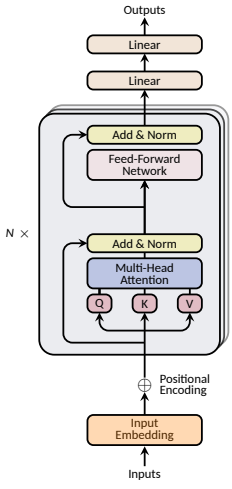
Adapting the transformer to graphs

How can we adapt the transformer to graph data?



Any ideas?

How can we adapt the transformer to graph data?

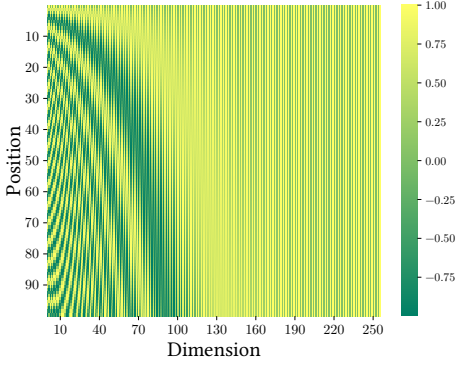


Adapting the transformer for graphs

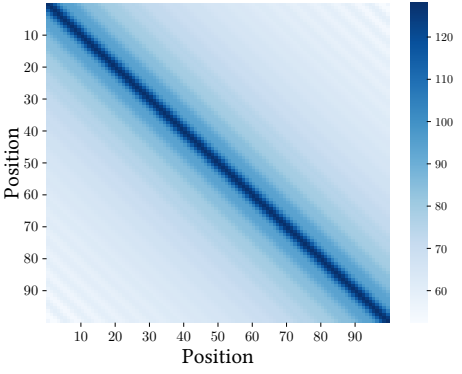
- ❇ Input treats the graph as a bag of nodes
- ❇ Encodes structure via positional encoding

⇒ Early work did this (Dwivedi and Bresson, 2021)

Absolute positional encoding



(a) APE



(b) Attention matrix

Positional encodings on graphs: LPE

Laplacian positional encoding

- Positioned as graph analog of the sine/cosine PEs
- $\mathbf{U}$ is sorted by increasing order of the corresponding eigenvalue.
- Take the first $1, \ldots, k$ eigenvectors of $\mathbf{U}$

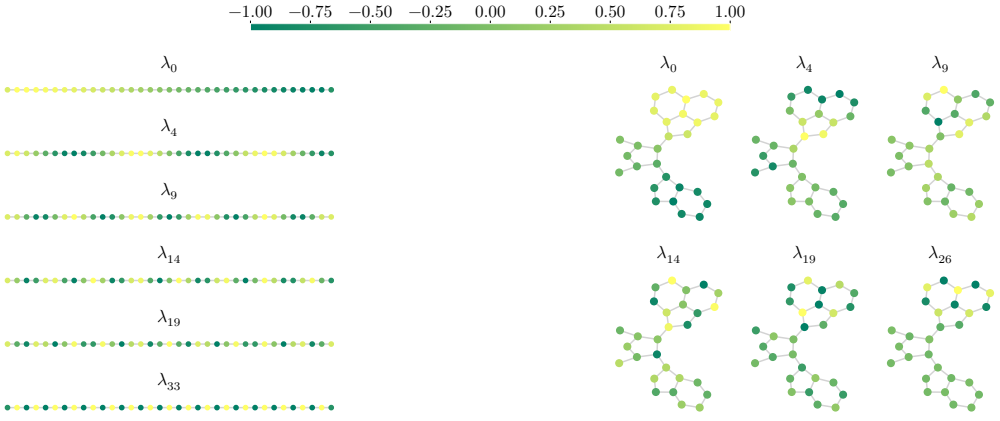
Positional encodings on graphs: LPE

Laplacian positional encoding

- Positioned as graph analog of the sine/cosine PEs
- $\mathbf{U}$ is sorted by increasing order of the corresponding eigenvalue.
- Take the first $1, \ldots, k$ eigenvectors of $\mathbf{U}$
- Limitation: only determined up to sign!

$$\Delta = I_n - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} = \mathbf{U}^T\Lambda\mathbf{U},$$
$$\mathrm{PE}_{\mathrm{LPE}} = \mathbf{U}_k = \left[\mathbf{U}_1, \ldots, \mathbf{U}_k,\right]$$

Laplacian positional encoding (LPE)



(a) LPE of line graph

(b) LPE of a molecule

Positional encodings on graphs: RWPE

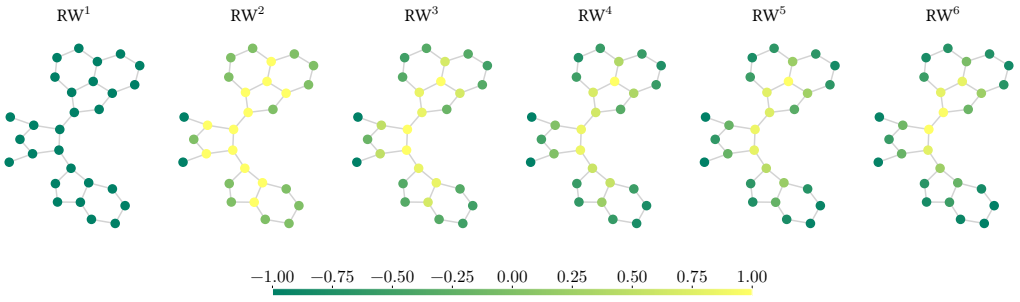
Random walk positional encoding (RWPE)

- Formed by taking powers of the random walk matrix, then taking the diagonal
- Captures some structural information in the graph

$$\mathrm{PE}_{\mathrm{RWPE}} = \begin{bmatrix} \mathrm{RW}^1_{1,1} & \dots & \mathrm{RW}^k_{1,n} \\ \vdots & \ddots & \vdots \\ \mathrm{RW}^1_{n,1} & \dots & \mathrm{RW}^k_{n,n} \end{bmatrix}$$

$$\text{where } \mathrm{RW} = \mathbf{D}^{-1}\mathbf{A}.$$

Random walk positional encoding (RWPE)



Early work on graph transformers showed lackluster results

Early publications tried:

- Different positional embeddings (LPE, RWPE, Weisfeler-Lehman, etc.) (Dwivedi and Bresson, 2021; Zhang et al., 2020)
- Using a GNN as preprocessing before the transformer layer (Rong et al., 2020)
- Limiting self-attention calculation to neighbors (Dwivedi and Bresson, 2021)

Do Transformers Really Perform Bad for Graph Representation?

Chengxuan Ying[*], Tianle Cai[*], Shengjie Luo[*],
Shuxin Zheng[†], Guolin Ke[†], Di He[†], Yanming Shen[†], Tie-Yan Liu[†]
[*]Dalian University of Technology [†]Microsoft Research Asia
ying1bengxuan@mail.dlut.edu.cn, {tianle.cai,shengjie.luo}@stu.pku.edu.cn
{shuxin.zheng, guolin.ke, dihe, tyliu}@microsoft.com, shenmi@dlut.edu.cn

Abstract

The Transformer architecture has become a dominant choice in many domains, such as natural language processing and computer vision. Yet, it has not achieved competitive performance on popular leaderboards of graph-level prediction compared to mainstream GNN variants. Therefore, it remains a mystery how Transformers could perform well for graph representation learning. In this paper, we solve this mystery by presenting Graphormer, which is built upon the standard Transformer architecture, and could attain excellent results on a broad range of graph representation learning tasks, especially on the recent OGB Large-Scale Challenge. Our key insight to utilizing Transformer in the graph is the necessity of effectively encoding the structural information of a graph into the model. To this end, we propose several simple yet effective structural encoding methods to help Graphormer better model graph-structured data. Besides, we mathematically characterize the expressive power of Graphormer and exhibit that with our ways of encoding the structural information of graphs, many popular GNN variants could be covered as the special cases of Graphormer. The code and models of Graphormer will be made publicly available at https://github.com/Microsoft/Graphormer.

1 Introduction

The Transformer [46] is well acknowledged as the most powerful neural network in modelling sequential data, such as natural language [11, 35, 8] and speech [15]. Model variants built upon Transformer have also been shown great performance in computer vision [5, 20, 9] and programming language [19, 47, 41]. However, to the best of our knowledge, Transformer has still not been the de-facto standard on public graph representation leaderboards [21, 14, 23]. There are many attempts of leveraging Transformer into the graph domain, but the only effective way to replacing message-passing GNN backbone (feature aggregation) in classic GNN variants by the softmax attention [40, 25, 30, 50, 43, 7]. Therefore, it is still an open question whether Transformer architecture is suitable to model graphs and how to make it work in graph representation learning.

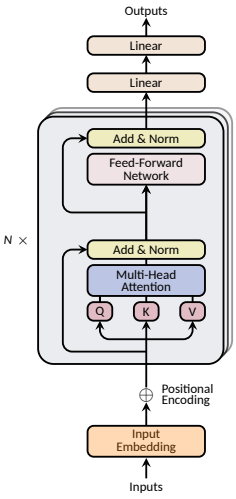
In this paper, we give an affirmative answer by developing Graphormer, which is directly built upon the standard Transformer, and achieves state-of-the-art performance on a wide range of graph-level prediction tasks, including the very recent Open Graph Benchmark Large-Scale Challenge (OGB-LSC) [21], and several popular leaderboards (e.g., OGB [23]). Benchmarking-GNN [14]. The Transformer is originally designed for sequence modelling. To utilize its power in graphs, we believe

[*]Interns at MSRA.
[†]Corresponding authors.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

Can we do better?

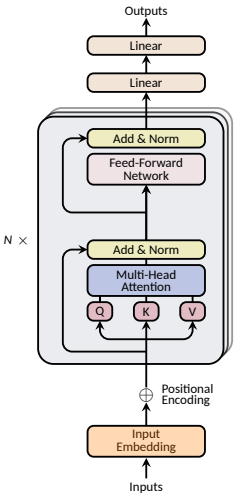
How can we adapt the transformer to work well on graph data?



Adapting the transformer for graphs

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

How can we adapt the transformer to work well on graph data?



Adapting the transformer for graphs

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

What if we *explicitly* incorporate structure into attention?

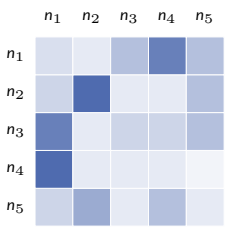
Combining global and local information using transformers

Dexiong Chen,* **Leslie O'Bray*** and Karsten Borgwardt. Structure-aware transformer for graph representation learning. In *International Conference on Machine Learning (ICML)*, 2022.

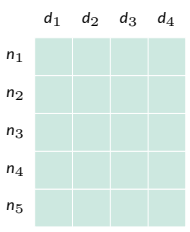
*equal contribution

Self-attention in the context of graphs

Attention matrix



Value matrix

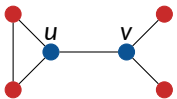


- $G = (V, E, \mathbf{X})$
- $|V| = n$
- $\mathbf{X} \in \mathbb{R}^{n \times d}$

- $\mathbf{Q} = \mathbf{X}\mathbf{W}_\mathbf{Q} \in \mathbb{R}^{n \times d}$
- $\mathbf{K} = \mathbf{X}\mathbf{W}_\mathbf{K} \in \mathbb{R}^{n \times d}$
- $\mathbf{V} = \mathbf{X}\mathbf{W}_\mathbf{V} \in \mathbb{R}^{n \times d}$

$$\text{Attn}(\mathbf{X}) := \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\mathbf{T}}{\sqrt{\mathbf{d}}}\right)\mathbf{V}$$

Encoding structure into self-attention



Existing attention compares nodes on the basis of their node attributes

$\Rightarrow$

Instead compare based on their structural **and** attribute similarity

From attention to structure-aware attention

Reformulate the attention calculation as a kernel smoother (Tsai et al., 2019):

$$\text{Attn}(x_v) = \sum_{u \in V} \frac{\kappa_{\exp}(x_v, x_u)}{\sum_{w \in V} \kappa_{\exp}(x_v, x_w)} f(x_u), \ \forall v \in V,$$

- $f(u) = \mathbf{W_V} x_u$ is the linear value function
- $\kappa_{\exp}$ is an exponential kernel on $\mathbb{R}^d \times \mathbb{R}^d$ parameterized by $\mathbf{W_Q}$ and $\mathbf{W_K}$:

$$\kappa_{\exp}(x, x') := \exp\left(\frac{\langle \mathbf{W_Q} x, \mathbf{W_K} x' \rangle}{\sqrt{d}}\right), \tag{1}$$

- $\langle \cdot, \cdot \rangle$ is the dot product on $\mathbb{R}^d$.

From attention to structure-aware attention

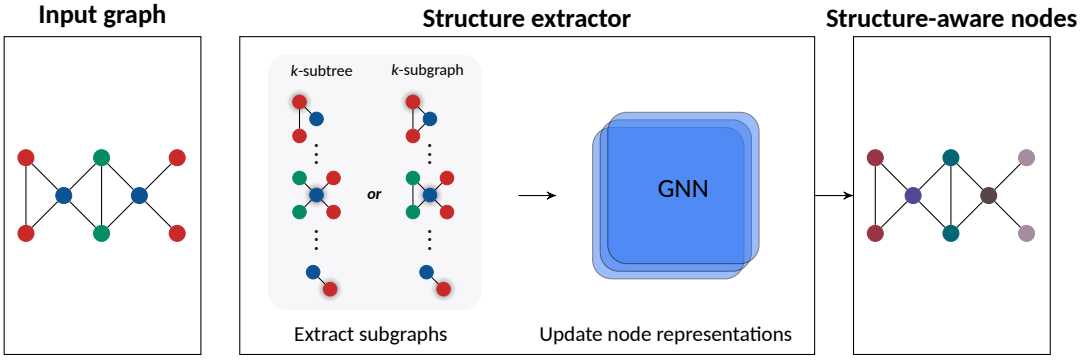
Self-attention **Structure-aware self-attention**

$$\text{Attn}(x_v) = \sum_{u \in V} \frac{\kappa_{\exp}(x_v, x_u)}{\sum_{w \in V} \kappa_{\exp}(x_v, x_w)} f(x_u) \quad \Rightarrow \quad \text{SA-attn}(v) = \sum_{u \in V} \frac{\kappa_{\text{graph}}\Big(S_G(v), S_G(u)\Big)}{\sum_{w \in V} \kappa_{\text{graph}}\Big(S_G(v), S_G(w)\Big)} f(x_u)$$

- κ_{graph} measures similarity between subgraphs
- $S_G(v)$ is a subgraph rooted at node v

$$\kappa_{\text{graph}}\Big(S_G(v), S_G(u)\Big) = \kappa_{\exp}\Big(\underbrace{\varphi(v, G)}_{\text{structure extractor}}, \varphi(u, G)\Big)$$

Example structure extractors: k-subtree and k-subgraph

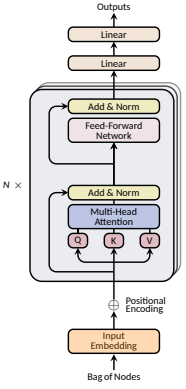


Example structure extractors:

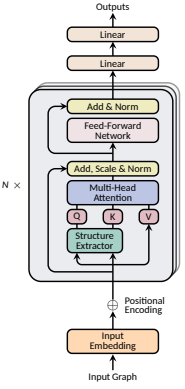
k-subtree: $\varphi(u, G) = \text{GNN}_G^{(k)}(u)$

k-subgraph: $\varphi(u, G) = \sum_{v \in \mathcal{N}_k(u)} \text{GNN}_G^{(k)}(v)$

Structure-Aware Transformer (SAT)

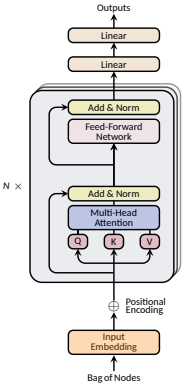


(a) Graph Transformer

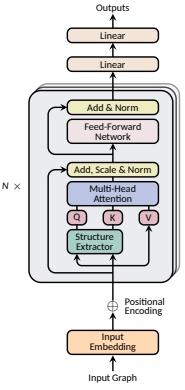


(b) SAT

Structure-Aware Transformer (SAT)



(a) Graph Transformer



(b) SAT

SAT is compatible with linear attention mechanisms, reducing the complexity from $\mathcal{O}\left(n^2\right)$ to $\mathcal{O}(n)$!

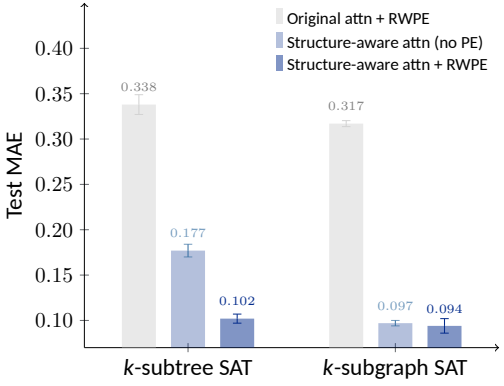
Experiments: SAT achieves SOTA results on several datasets

		ZINC ↓	CLUSTER ↑	PATTERN ↑
	# graphs	12,000	12,000	14,000
	Avg. # nodes	23.2	117.2	118.9
	Avg. # edges	49.8	4,303.9	6,098.9
	Metric	MAE	Accuracy	Accuracy
GNNs	GIN	0.387±0.015	64.716±1.553	85.590±0.011
GNNs	GAT	0.384±0.007	70.587±0.447	78.271±0.186
GNNs	PNA	0.188±0.004	67.077±0.977	86.567±0.075
GTs	Transformer+RWPE	0.310±0.005	29.622±0.176	86.183±0.019
GTs	SAN	0.139±0.006	76.691±0.650	86.581±0.037
GTs	Graphormer	0.122±0.006	–	–
	k-subtree SAT	**0.102±0.005**	**77.751±0.121**	**86.865±0.043**
	k-subgraph SAT	**0.094±0.008**	**77.856±0.104**	**86.848±0.037**

	OGBG-PPA ↑	OGBG-CODE2 ↑
# graphs	158,100	452,741
Avg. # nodes	243.4	125.2
Avg. # edges	2,266.1	124.2
Metric	Accuracy	F1 score
GCN-Virtual Node	0.6857±0.0061	0.1595±0.0018
GIN-Virtual Node	**0.7037±0.0107**	0.1581±0.0026
Transformer	0.6454±0.0033	0.1670±0.0015
GraphTrans	–	**0.1830±0.0024**
k-subtree SAT	**0.7522±0.0056**	**0.1937±0.0028**

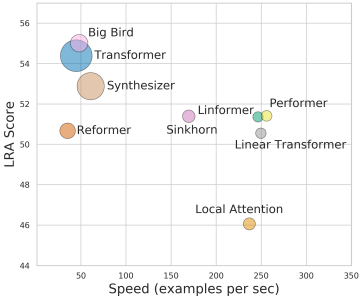
First | **Second** | Third

Benefit of the structure-aware attention on ZINC



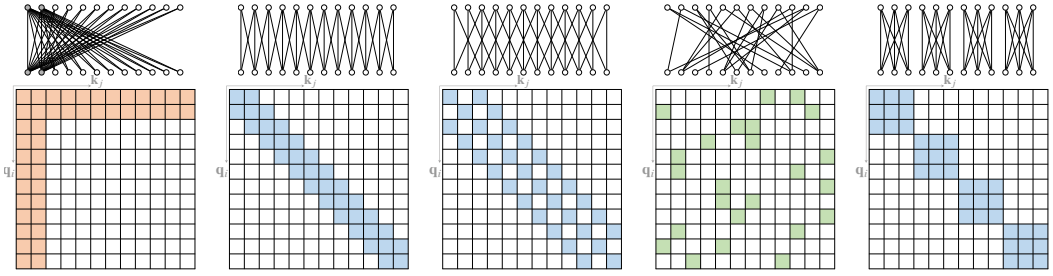
Limitations of transformers and graph transformers

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



(Tay et al., 2021)

Example of different ideas for sparse attention



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

(Lin et al., 2022)

A lot of exciting recent work in graph transformers

- Rethinking Graph Transformers with Spectral Attention (Kreuzer et al., 2021)
- Do Transformers Really Perform Badly for Graph Representation? (Ying et al., 2021)
- Recipe for a General, Powerful, Scalable Graph Transformer (Rampasek et al., 2022)
- Pure Transformers are Powerful Graph Learners (Kim et al., 2022)

Summary & take-home

- GNNs are SOTA but have known issues given that they only model local structural information
- Transformers offer a way to counteract some of these issues by incorporating global information in the graph
- Global information alone may not be enough, but the combination of local and global is very promising

References I

P. Barceló, E. V. Kostylev, M. Monet, J. Pérez, J. Reutter, and J. P. Silva. The logical expressiveness of graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2020.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2020.

V. P. Dwivedi and X. Bresson. A generalization of transformer networks to graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications*, 2021.

A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. 2020.

J. Kim, D. T. Nguyen, S. Min, S. Cho, M. Lee, H. Lee, and S. Hong. Pure transformers are powerful graph learners. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=um2BxfgkT2_`.

D. Kreuzer, D. Beaini, W. L. Hamilton, V. Létourneau, and P. Tossou. Rethinking graph transformers with spectral attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

T. Lin, Y. Wang, X. Liu, and X. Qiu. A survey of transformers. *AI Open*, 3:111–132, 2022. ISSN 2666-6510. doi: https://doi.org/10.1016/j.aiopen.2022.10.001.

C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

References II

K. Oono and T. Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations (ICLR)*, 2020.

OpenAI. Gpt-4 technical report, 2023.

L. Rampasek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini. Recipe for a general, powerful, scalable graph transformer. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL `https://openreview.net/forum?id=lMMaNf6oxKM`.

A. Rives, J. Meier, T. Sercu, S. Goyal, Z. Lin, J. Liu, D. Guo, M. Ott, C. L. Zitnick, J. Ma, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15), 2021.

Y. Rong, Y. Bian, T. Xu, W. Xie, Y. Wei, W. Huang, and J. Huang. Self-supervised graph transformer on large-scale molecular data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Y. Tay, M. Dehghani, S. Abnar, Y. Shen, D. Bahri, P. Pham, J. Rao, L. Yang, S. Ruder, and D. Metzler. Long range arena : A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=qVyeW-grC2k`.

J. Topping, F. D. Giovanni, B. P. Chamberlain, X. Dong, and M. M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations*, 2022.

Y.-H. H. Tsai, S. Bai, M. Yamada, L.-P. Morency, and R. Salakhutdinov. Transformer dissection: A unified understanding of transformer's attention via the lens of kernel. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

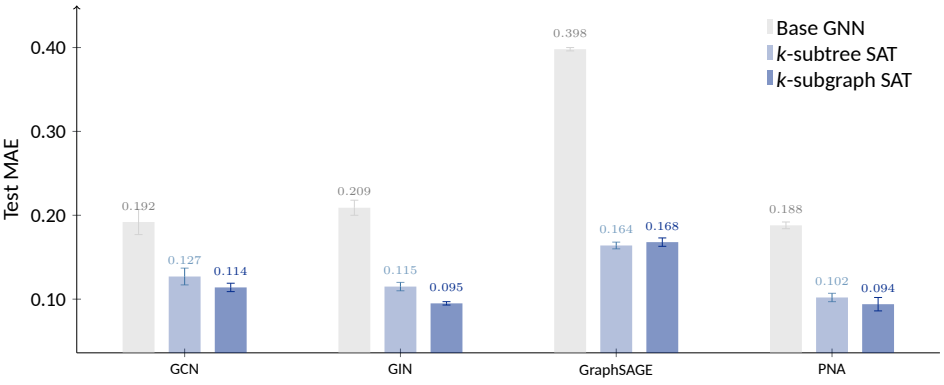
References III

C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu. Do transformers really perform badly for graph representation? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

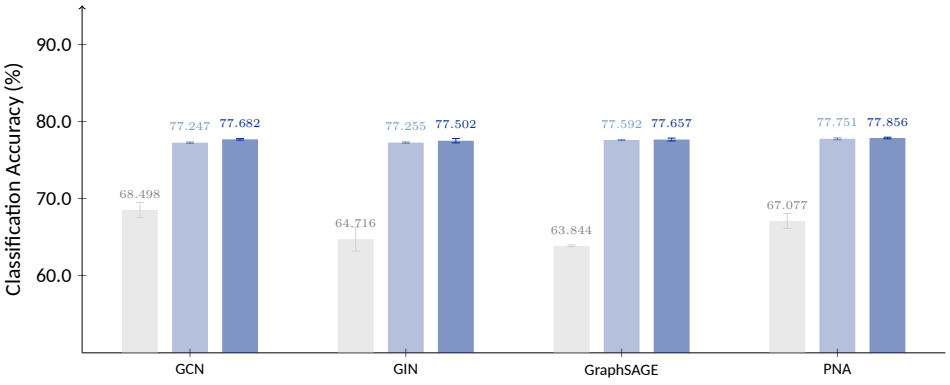
J. Zhang, H. Zhang, C. Xia, and L. Sun. Graph-Bert: Only attention is needed for learning graph representations. 2020.

Backup slides: SAT

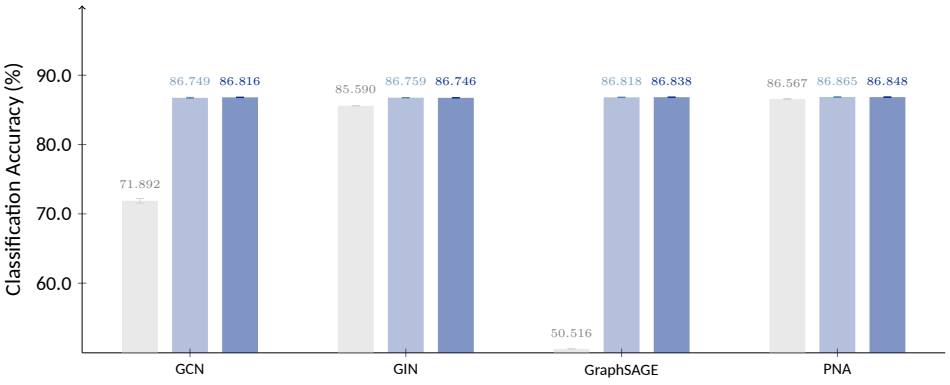
Uplift of integrating global information with local information on ZINC



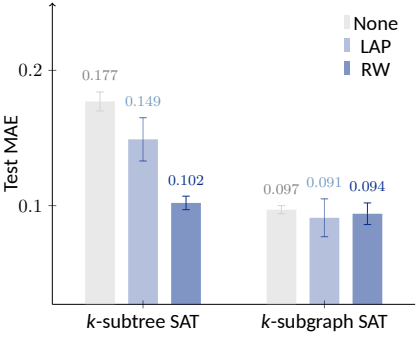
Uplift of integrating global information with local information, CLUSTER



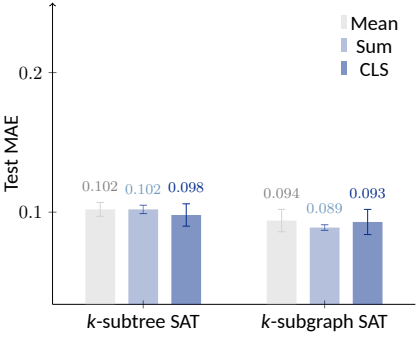
Uplift of integrating global information with local information, PATTERN



Effect of positional encoding and readout on ZINC

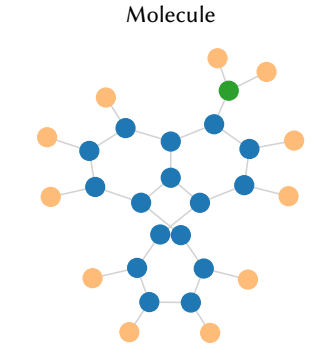


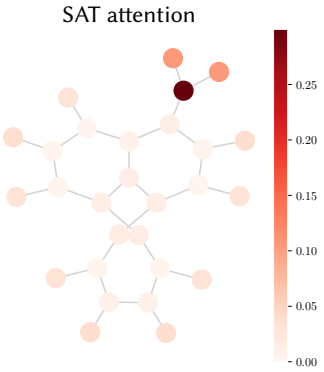
(a) Effect of positional encoding

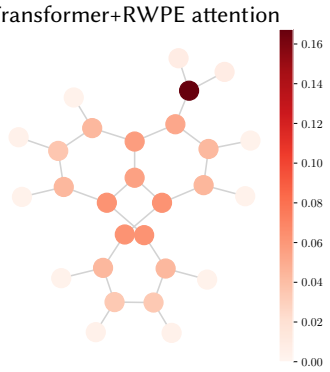


(b) Effect of readout

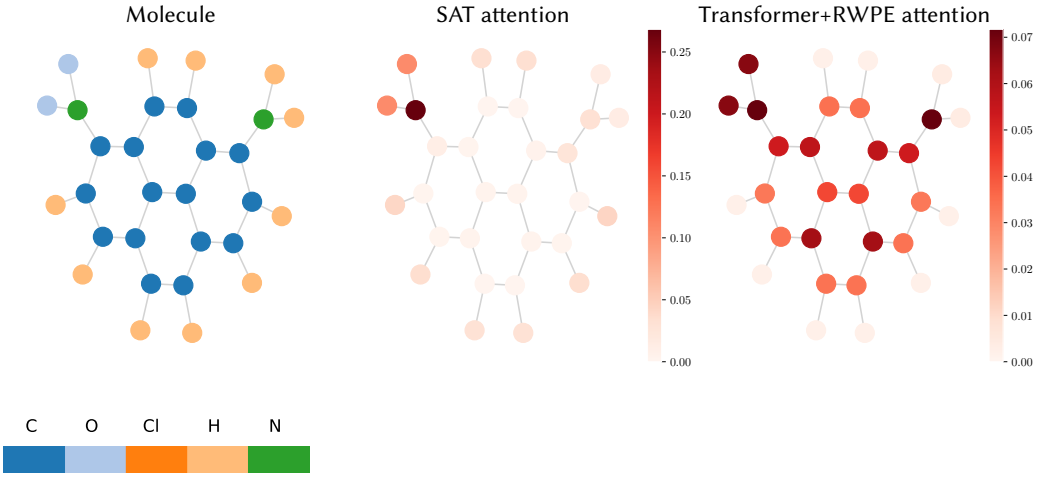
Structure-aware attention scores on Mutagenicity







Structure-aware attention scores on Mutagenicity



Structure-aware attention scores on Mutagenicity