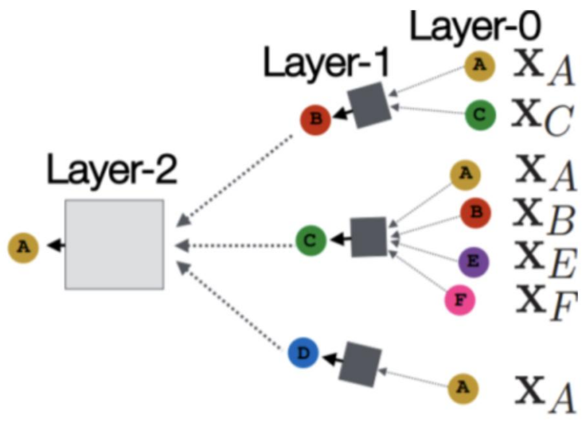
Simplifying and Empowering Transformers for Large-Graph Representations

Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, Junchi Yan

Codes: https://github.com/qitianwu/SGFormer

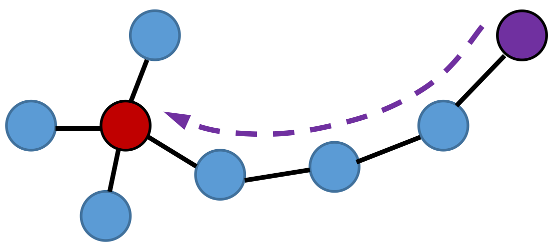
Pitfalls of Graph Neural Networks



❑ The designs of mainstream GNNs:

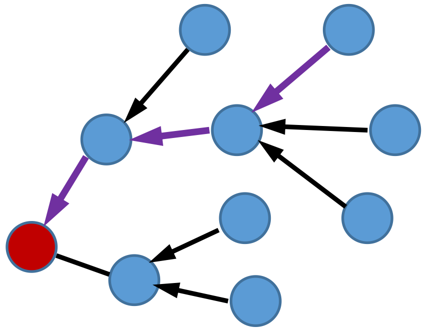
- Locally aggregate neighbored nodes' features in each layer
- Use neighbored nodes' embs for informative represensation

❑ Common scenarios GNNs show deficient capability:



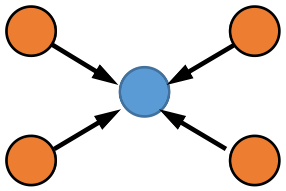
hard to capture long-range dependence [Dai et al., 2018]

long-range reasoning



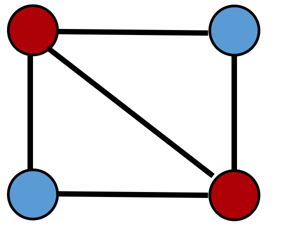
distant signals are overly squashed [Alon et al., 2021]

over-squashing



dissimilar linked nodes propagate wrong signals [Zhu et al., 2020]

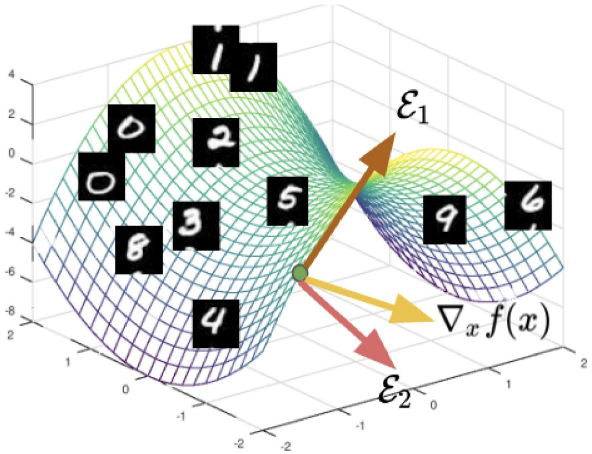
heterophily



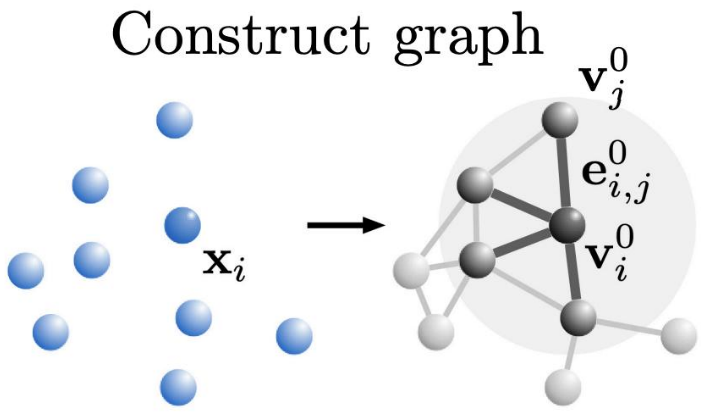
fail to distinguish two similar inputs [Xu et al., 2019]

expressivity

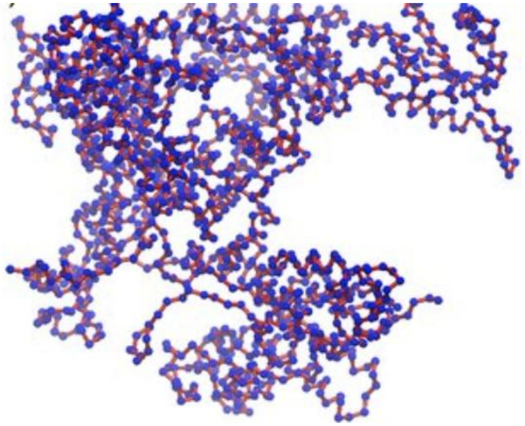
Inter-Dependent Data without Input Graphs



Observed data lies on low-dimensional manifold [Sebastian et al., 2021]



Physical interactions affect data generation yet are not observed [Alvaro et al., 2020]

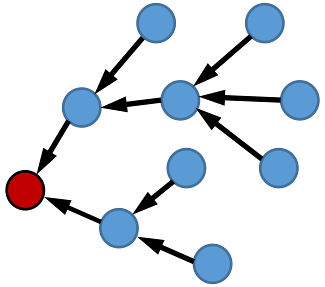


Complex hidden structures beyond observed geometry [Xu et al., 2020]

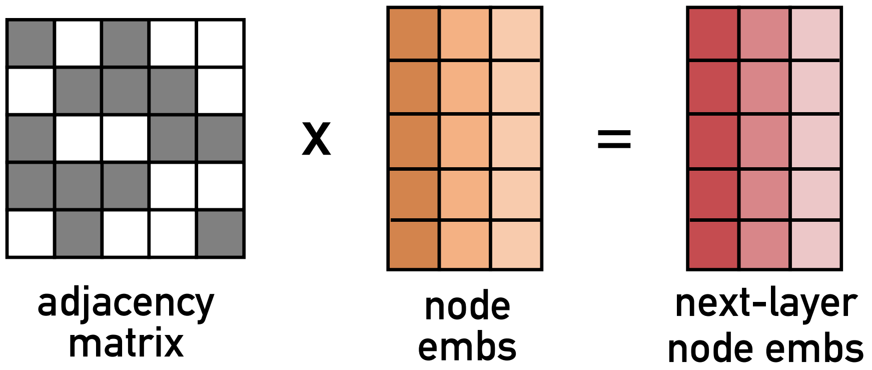
- ❑ GNNs require observed graphs as input:
  - Solution: Pre-define a graph by some rules (e.g., k nearest neighbors)
  - Limitation: the pre-defined graph is independent of downstream tasks

Message Passing Beyond Input Graphs

Graph Neural Networks

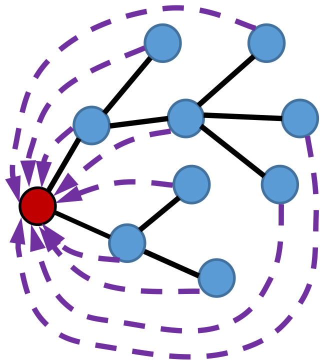


local message passing defined over fixed input topology

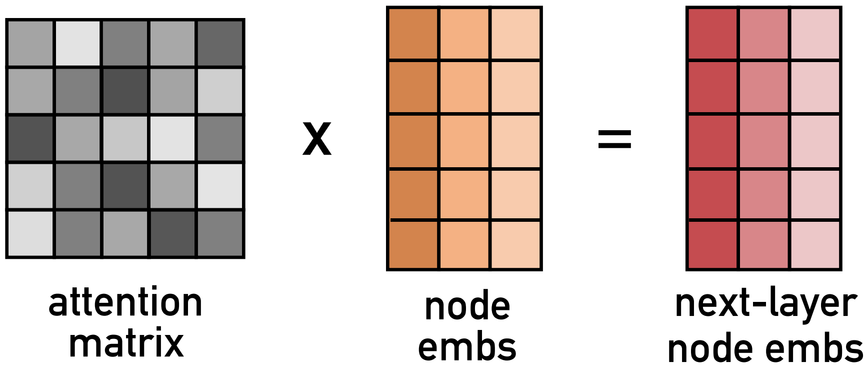


only require $O(E)$ when using sparse matrix computation

Transformers



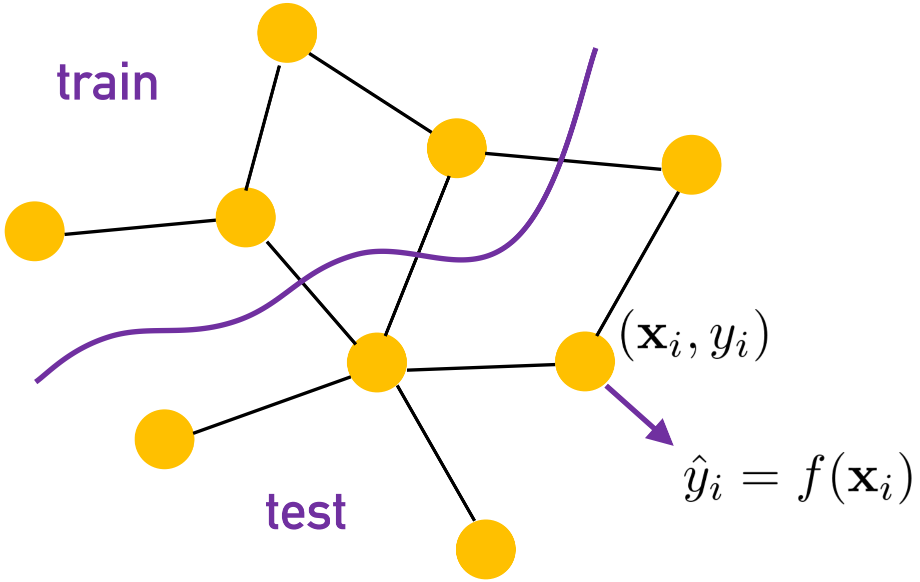
all-pair message passing on layer-specific latent graphs



Q1: computational bottleneck $O(N^2)$

Q2: how to incorporate graph inductive bias

Preliminary: Notations



- Each node is an instance with a label
- Train/test on a dataset of nodes in a graph
- The graph size can be arbitrarily large

Notations for each node

$\mathbf{x}_u$ node (input) feature

y_u node ground-truth label

$\hat{y}_u$ node predicted label

$\mathbf{z}_u^{(l)}$ node embedding at the l-th layer

Notations for the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

$N = |\mathcal{V}|$ node number $E = |\mathcal{E}|$ edge number

$\mathbf{X} = [\mathbf{x}_u]_{u=1}^N$ node feature matrix

$\mathbf{Y} = [y_u]_{u=1}^N$ label vector/matrix

$\mathbf{A} = [a_{uv}]_{u,v\in\mathcal{V}}$ adjacency matrix

$\mathbf{Z}^{(l)} = [\mathbf{z}_u^{(l)}]_{u=1}^N$ node embedding matrix

NodeFormer: All-Pair Attention with O(N)

❑ Kernelized softmax message passing

$$\mathbf{z}_u^{(l+1)} = \sum_{v=1}^{N} \frac{\exp(\mathbf{q}_u^\top \mathbf{k}_v)}{\sum_{w=1}^{N} \exp(\mathbf{q}_u^\top \mathbf{k}_w)} \cdot \mathbf{v}_v$$

where $\mathbf{q}_u = W_Q^{(l)} \mathbf{z}_u^{(l)}, \quad \mathbf{k}_u = W_K^{(l)} \mathbf{z}_u^{(l)}, \quad \mathbf{v}_u = W_V^{(l)} \mathbf{z}_u^{(l)}$

$$\mathbf{z}_u^{(l+1)} = \sum_{v=1}^{N} \frac{\kappa(\mathbf{q}_u, \mathbf{k}_v)}{\sum_{w=1}^{N} \kappa(\mathbf{q}_u, \mathbf{k}_w)} \cdot \mathbf{v}_v$$

where $\kappa(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a positive-definite kernel

[Mercer's theorem] $\kappa(\mathbf{a}, \mathbf{b}) = \langle \Phi(\mathbf{a}), \Phi(\mathbf{b}) \rangle_{\mathcal{V}} \approx \phi(\mathbf{a})^\top \phi(\mathbf{b})$

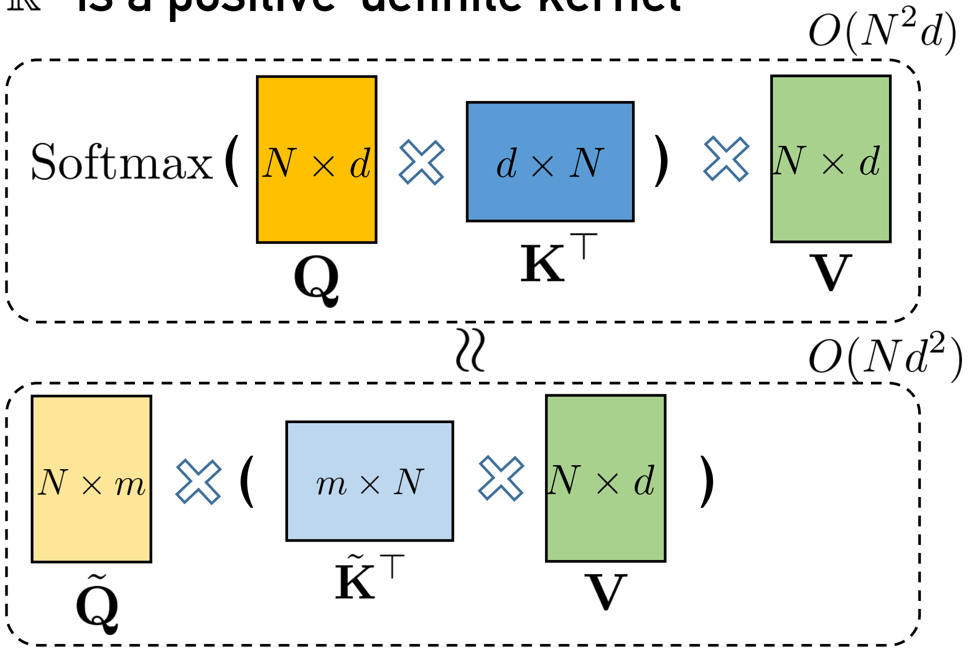
$\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^m$ is a random feature map

$$\mathbf{z}_u^{(l+1)} = \sum_{v=1}^{N} \frac{\phi(\mathbf{q}_u)^\top \phi(\mathbf{k}_v)}{\sum_{w=1}^{N} \phi(\mathbf{q}_u)^\top \phi(\mathbf{k}_w)} \cdot \mathbf{v}_v = \boxed{\frac{\phi(\mathbf{q}_u)^\top \sum_{v=1}^{N} \phi(\mathbf{k}_v) \cdot \mathbf{v}_v^\top}{\phi(\mathbf{q}_u)^\top \sum_{w=1}^{N} \phi(\mathbf{k}_w)}}$$

two summation are shared by all nodes (independent of u)

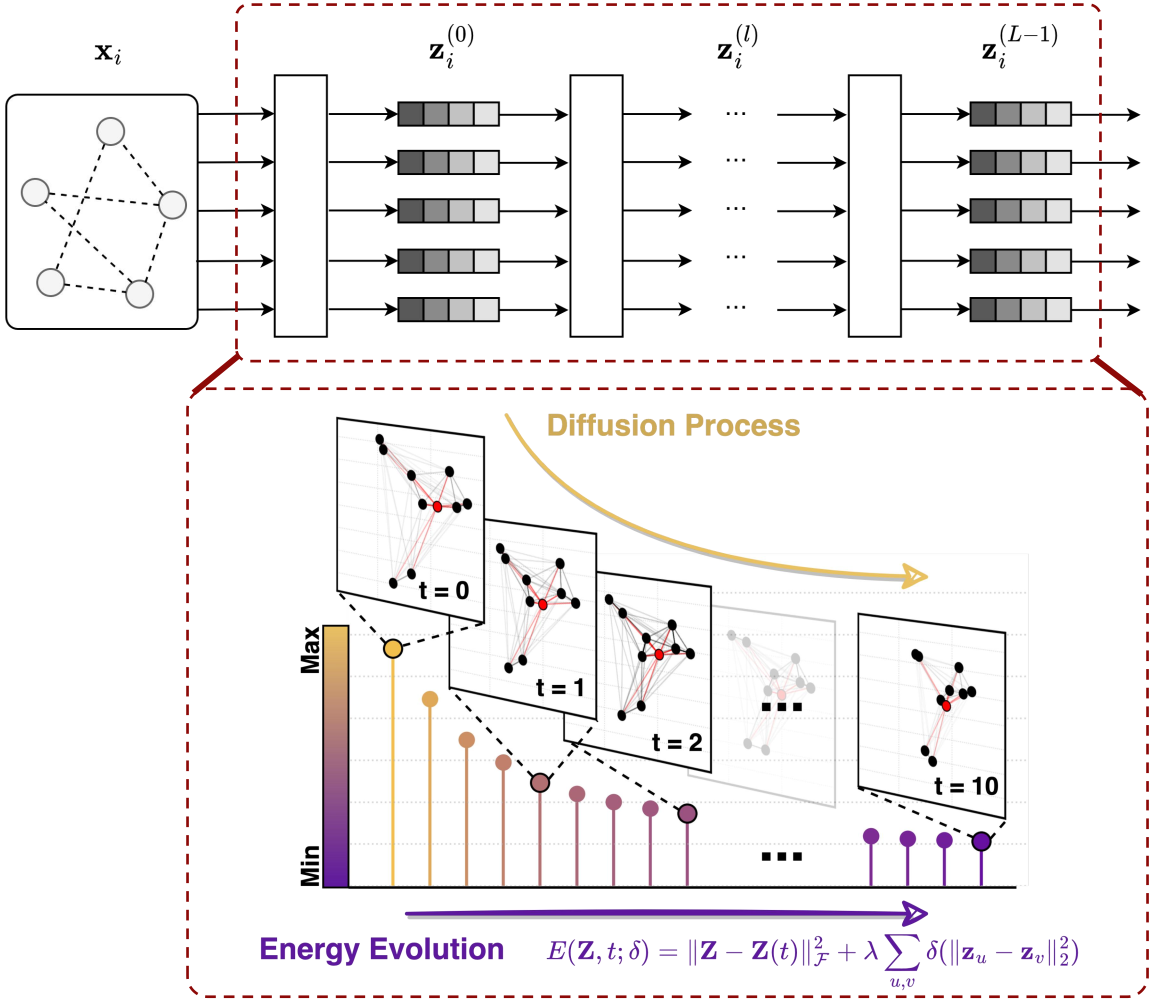
only compute once

computation complexity $O(N) + N \cdot O(1) = O(N)$



Qitian Wu et al., NodeFormer: A Scalable Graph Structure Learning Transformer for Node Classification, NeurIPS 2022

DIFFormer: Transformers by Diffusion



$$\hat{\mathbf{S}}_{ij}^{(k)} = \frac{f(\|\mathbf{z}_i^{(k)} - \mathbf{z}_j^{(k)}\|_2^2)}{\sum_{l=1}^{N} f(\|\mathbf{z}_i^{(k)} - \mathbf{z}_l^{(k)}\|_2^2)}, \quad 1 \le i, j \le N$$

$$\mathbf{z}_i^{(k+1)} = \left(1 - \tau \sum_{j=1}^{N} \hat{\mathbf{S}}_{ij}^{(k)}\right) \mathbf{z}_i^{(k)} + \tau \sum_{j=1}^{N} \hat{\mathbf{S}}_{ij}^{(k)} \mathbf{z}_j^{(k)}, \quad 1 \le i \le N$$

Global attention inspired by diffusivity function

$$\omega_{ij}^{(k)} = f(\|\tilde{\mathbf{z}}_i^{(k)} - \tilde{\mathbf{z}}_j^{(k)}\|_2^2) = 1 + \left(\frac{\mathbf{z}_i^{(k)}}{\|\mathbf{z}_i^{(k)}\|_2}\right)^{\top} \left(\frac{\mathbf{z}_j^{(k)}}{\|\mathbf{z}_j^{(k)}\|_2}\right)$$

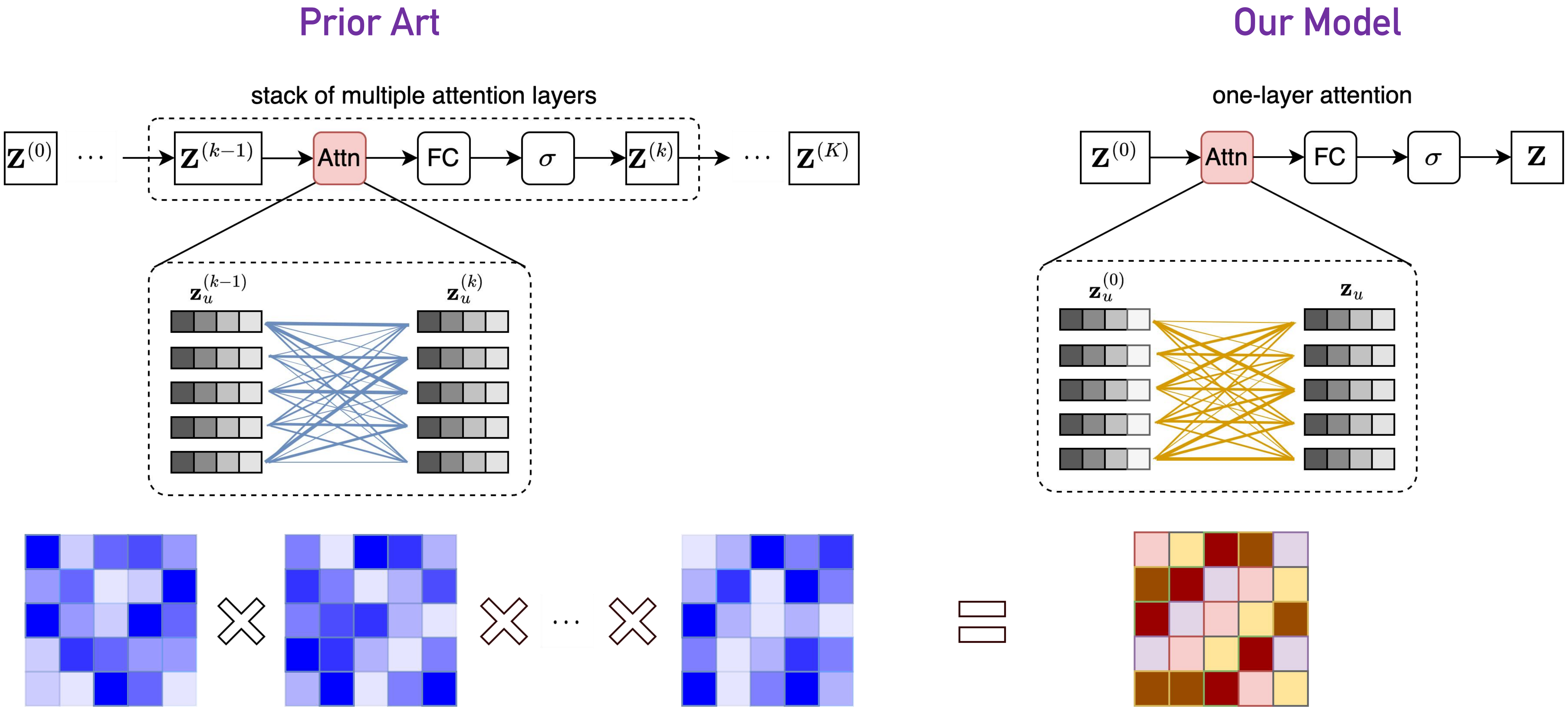
$$\sum_{j=1}^{N} \mathbf{S}_{ij}^{(k)} \mathbf{z}_j^{(k)} = \sum_{j=1}^{N} \frac{1 + (\tilde{\mathbf{z}}_i^{(k)})^{\top} \tilde{\mathbf{z}}_j^{(k)}}{\sum_{l=1}^{N} \left(1 + (\tilde{\mathbf{z}}_i^{(k)})^{\top} \tilde{\mathbf{z}}_l^{(k)}\right)} \mathbf{z}_j^{(k)}$$

$$= \boxed{\frac{\sum_{j=1}^{N} \mathbf{z}_j^{(k)} + \left(\sum_{j=1}^{N} \tilde{\mathbf{z}}_j^{(k)} \cdot (\mathbf{z}_j^{(k)})^{\top}\right) \cdot \tilde{\mathbf{z}}_i^{(k)}}{N + (\tilde{\mathbf{z}}_i^{(k)})^{\top} \sum_{l=1}^{N} \tilde{\mathbf{z}}_l^{(k)}}}$$

$O(N)$

Qitian Wu et al., DIFFormer: Scalable (Graph) Transformers Induced by Energy Constrained Diffusion, ICLR 2023

Do We Really Need Deep Attention Layers?



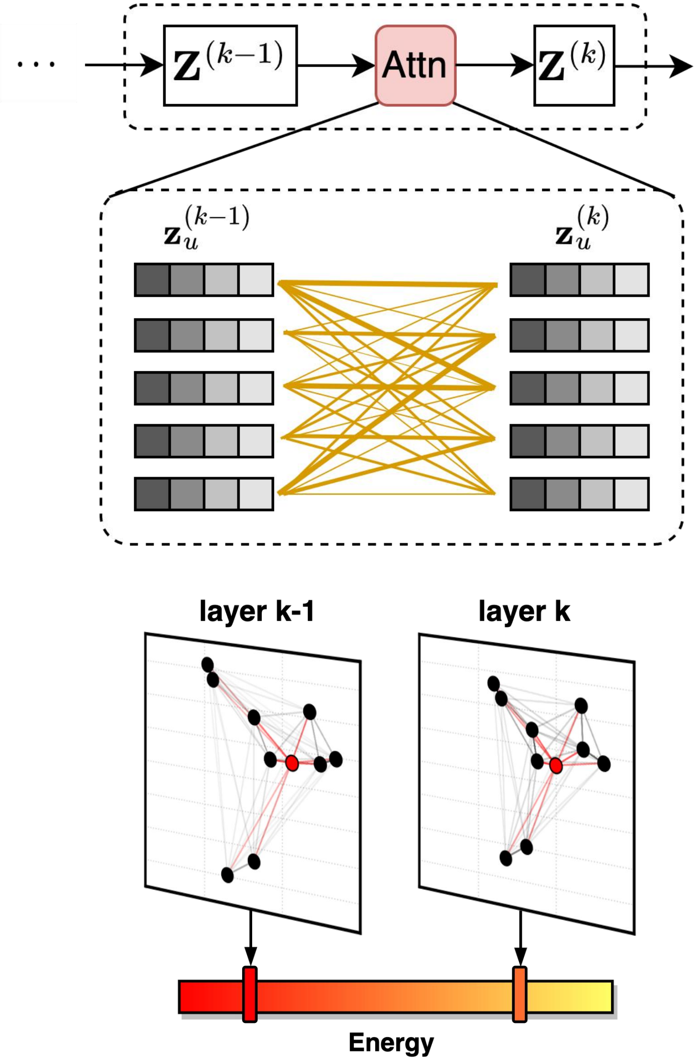
Qitian Wu et al., Simplifying and Empowering Transformers on Large-Graph Representations, NeurIPS 2023

How Powerful Are One-Layer Attentions?

Consider the k-th attention layer in Transformers:

$$\mathbf{z}_u^{(k)} = \boxed{(1-\tau)\mathbf{z}_u^{(k-1)}} + \tau \sum_{v=1}^{N} \boxed{c_{uv}^{(k)}} \mathbf{z}_v^{(k-1)}$$

residual link attention weight



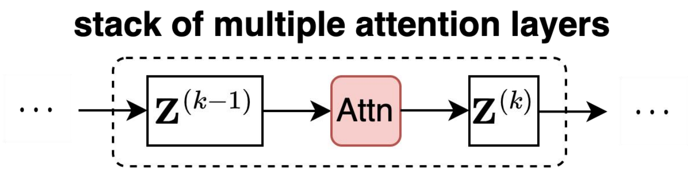
Theorem 1 (Transformers as Graph Signal Denoising)

For any given attention matrix $\mathbf{C}^{(k)} = [c_{uv}^{(k)}]_{N \times N}$, the k-th attention layer is equivalent to a gradient descent operation with step size $\tau/2\lambda$ for an optimization problem with the cost function

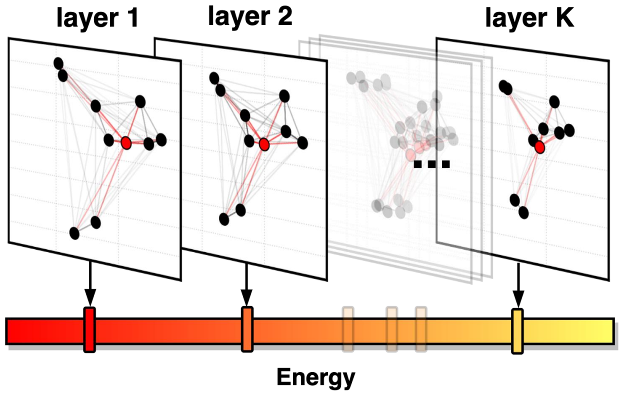
$$\min_{\mathbf{Z}} \sum_u \|\mathbf{z}_u - \mathbf{z}_u^{(k-1)}\|_2^2 + \lambda \sum_{u,v} c_{uv}^{(k)} \|\mathbf{z}_u - \mathbf{z}_v\|_2^2$$

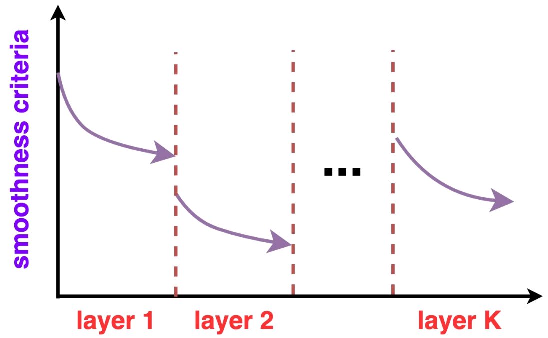
a generalization of Dirichlet energy

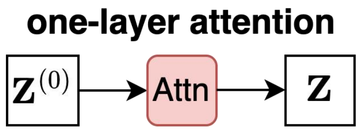
How Powerful Are One-Layer Attentions?



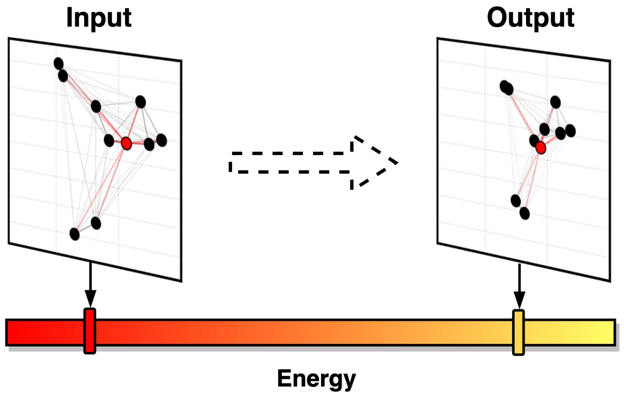
$$\mathbf{z}_u^{(k)} = (1-\tau)\mathbf{z}_u^{(k-1)} + \tau \sum_{v=1}^{N} c_{uv}^{(k)} \mathbf{z}_v^{(k-1)}$$

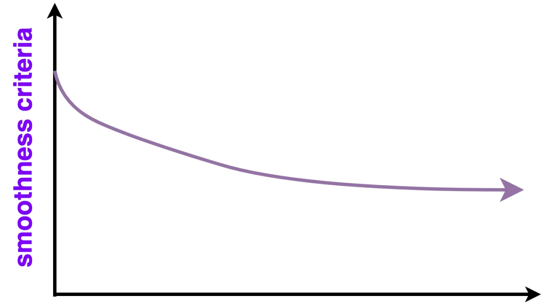






$$\mathbf{z}_u = (1-\tau)\mathbf{z}_u^{(0)} + \tau \sum_{v=1}^{N} c_{uv} \mathbf{z}_v^{(0)}$$





Theorem 2 (Equivalence between Multi-Layer Attentions and One-Layer Attention)

For any K-layer attention, there exists a one-layer model that induces the same denoising effect.

SGFormer: Simplified Graph Transformers

Observation: one-layer all-pair attention is expressive enough for propagating global information among arbitrary node pairs

SGFormer: one-layer single-head global attention + auxiliary GNN

- Simple attention with linear complexity: $\mathbf{Z}^{(0)} = f_I(\mathbf{X})$

$$\mathbf{Q} = f_Q(\mathbf{Z}^{(0)}), \quad \tilde{\mathbf{Q}} = \frac{\mathbf{Q}}{\|\mathbf{Q}\|_{\mathcal{F}}}, \quad \mathbf{K} = f_K(\mathbf{Z}^{(0)}), \quad \tilde{\mathbf{K}} = \frac{\mathbf{K}}{\|\mathbf{K}\|_{\mathcal{F}}}, \quad \mathbf{V} = f_V(\mathbf{Z}^{(0)})$$

$$\mathbf{D} = \operatorname{diag}\left(\mathbf{1} + \frac{1}{N}\tilde{\mathbf{Q}}(\tilde{\mathbf{K}}^\top \mathbf{1})\right), \quad \mathbf{Z} = \beta \mathbf{D}^{-1}\left[\mathbf{V} + \frac{1}{N}\tilde{\mathbf{Q}}(\tilde{\mathbf{K}}^\top \mathbf{V})\right] + (1-\beta)\mathbf{Z}^{(0)}$$

- Add an auxiliary GNN at the output layer:

$$\mathbf{Z}_O = (1-\alpha)\mathbf{Z} + \alpha \mathrm{GN}(\mathbf{Z}^{(0)}, \mathbf{A}), \quad \hat{Y} = f_O(\mathbf{Z}_O)$$

Qitian Wu et al., Simplifying and Empowering Transformers on Large-Graph Representations, NeurIPS 2023

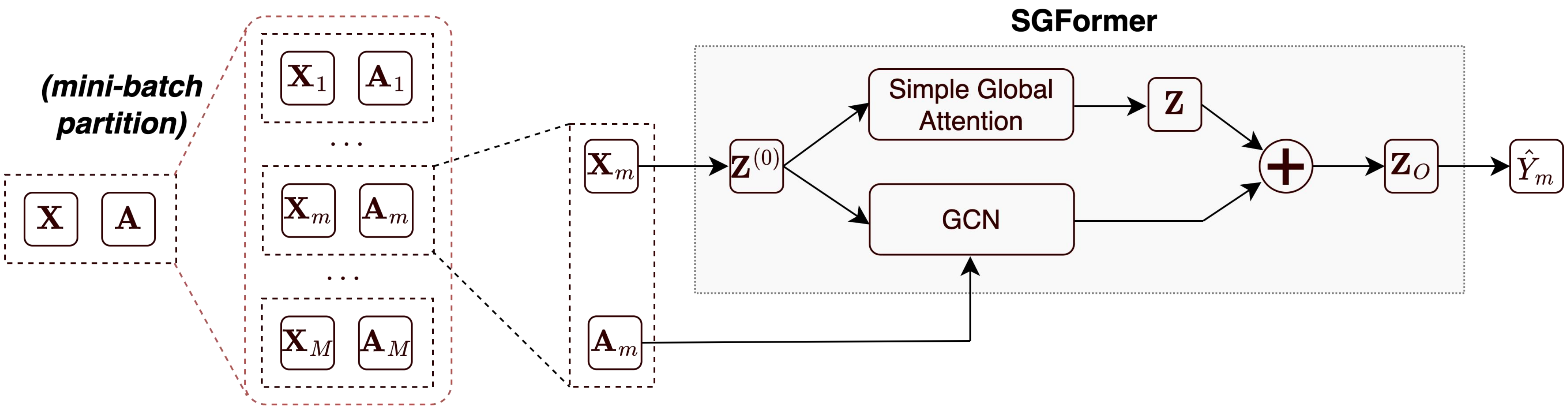
SGFormer: Scaling to Large Graphs

Challenge of training on large graphs:

The graph data cannot be loaded as a whole into a single GPU for training

Mini-batch sampling strategies:

1) using local graph adjacency 2) neighbor sampling



Qitian Wu et al., Simplifying and Empowering Transformers on Large-Graph Representations, NeurIPS 2023

Comparison of Existing Graph Transformers

	pos emb	multi-head	pre-processing	all-pair expressivity	complexity	largest demo of #nodes
GraphTransformer [Dwivedi et al. 2020]	R	R	R	yes	$O(N^2)$	0.2K
Graphormer [Ying et al. 2021]	R	R	R	yes	$O(N^2)$	0.3K
SAT [Chen et al. 2022]	R	R	R	yes	$O(N^2)$	0.2K
EGT [Hussain et al. 2022]	R	R	R	yes	$O(N^2)$	0.5K
GraphGPS [Rampáse et al. 2022]	R	R	R	yes	$O(N^2)$	1.0K
NodeFormer [Wu et al. 2022]	R	R	-	yes	$O(N+E)$	2M
SGFormer	-	-	-	yes	$O(N+E)$	0.1B

Experiment on Medium-Sized Graphs

Results on medium-sized node classification graphs

Dataset	Cora	CiteSeer	PubMed	Actor	Squirrel	Chameleon	Deezer
# nodes	2,708	3,327	19,717	7,600	2223	890	28,281
# edges	5,278	4,552	44,324	29,926	46,998	8,854	92,752
GCN	81.6 ± 0.4	71.6 ± 0.4	78.8 ± 0.6	30.1 ± 0.2	38.6 ± 1.8	41.3 ± 3.0	62.7 ± 0.7
GAT	83.0 ± 0.7	72.1 ± 1.1	79.0 ± 0.4	29.8 ± 0.6	35.6 ± 2.1	39.2 ± 3.1	61.7 ± 0.8
SGC	80.1 ± 0.2	71.9 ± 0.1	78.7 ± 0.1	27.0 ± 0.9	39.3 ± 2.3	39.0 ± 3.3	62.3 ± 0.4
JKNet	81.8 ± 0.5	70.7 ± 0.7	78.8 ± 0.7	30.8 ± 0.7	39.4 ± 1.6	39.4 ± 3.8	61.5 ± 0.4
APPNP	**83.3 ± 0.5**	71.8 ± 0.5	**80.1 ± 0.2**	31.3 ± 1.5	35.3 ± 1.9	38.4 ± 3.5	66.1 ± 0.6
H_2GCN	82.5 ± 0.8	71.4 ± 0.7	79.4 ± 0.4	34.4 ± 1.7	35.1 ± 1.2	38.1 ± 4.0	66.2 ± 0.8
SIGN	82.1 ± 0.3	72.4 ± 0.8	79.5 ± 0.5	36.5 ± 1.0	40.7 ± 2.5	41.7 ± 2.2	66.3 ± 0.3
CPGNN	80.8 ± 0.4	71.6 ± 0.4	78.5 ± 0.7	34.5 ± 0.7	38.9 ± 1.2	40.8 ± 2.0	65.8 ± 0.3
GloGNN	81.9 ± 0.4	72.1 ± 0.6	78.9 ± 0.4	36.4 ± 1.6	35.7 ± 1.3	40.2 ± 3.9	65.8 ± 0.8
$\text{Graphormer}_{\text{SMALL}}$	OOM	OOM	OOM	OOM	OOM	OOM	OOM
$\text{Graphormer}_{\text{SMALLER}}$	75.8 ± 1.1	65.6 ± 0.6	OOM	OOM	40.9 ± 2.5	41.9 ± 2.8	OOM
$\text{Graphormer}_{\text{ULTRASSMALL}}$	74.2 ± 0.9	63.6 ± 1.0	OOM	33.9 ± 1.4	39.9 ± 2.4	41.3 ± 2.8	OOM
$\text{GraphTrans}_{\text{SMALL}}$	80.7 ± 0.9	69.5 ± 0.7	OOM	32.6 ± 0.7	**41.0 ± 2.8**	42.8 ± 3.3	OOM
$\text{GraphTrans}_{\text{ULTRASSMALL}}$	81.7 ± 0.6	70.2 ± 0.8	77.4 ± 0.5	32.1 ± 0.8	40.6 ± 2.4	42.2 ± 2.9	OOM
NodeFormer	82.2 ± 0.9	**72.5 ± 1.1**	79.9 ± 1.0	**36.9 ± 1.0**	38.5 ± 1.5	34.7 ± 4.1	66.4 ± 0.7
SGFormer	**84.5 ± 0.8**	**72.6 ± 0.2**	**80.3 ± 0.6**	**37.9 ± 1.1**	**41.8 ± 2.2**	**44.9 ± 3.9**	**67.1 ± 1.1**

Experiment on Large-Sized Graphs

Results on large node classification graphs

Method	ogbn-proteins	Amazon2m	pokec	ogbn-arxiv	ogbn-papers100M
# nodes	132,534	2,449,029	1,632,803	169,343	111,059,956
# edges	39,561,252	61,859,140	30,622,564	1,166,243	1,615,685,872
MLP	72.04 ± 0.48	63.46 ± 0.10	60.15 ± 0.03	55.50 ± 0.23	47.24 ± 0.31
GCN	72.51 ± 0.35	83.90 ± 0.10	62.31 ± 1.13	**71.74 ± 0.29**	OOM
SGC	70.31 ± 0.23	81.21 ± 0.12	52.03 ± 0.84	67.79 ± 0.27	63.29 ± 0.19
GCN-NSampler	73.51 ± 1.31	83.84 ± 0.42	63.75 ± 0.77	68.50 ± 0.23	62.04 ± 0.27
GAT-NSampler	74.63 ± 1.24	85.17 ± 0.32	62.32 ± 0.65	67.63 ± 0.23	63.47 ± 0.39
SIGN	71.24 ± 0.46	80.98 ± 0.31	68.01 ± 0.25	70.28 ± 0.25	**65.11 ± 0.14**
NodeFormer	**77.45 ± 1.15**	**87.85 ± 0.24**	**70.32 ± 0.45**	59.90 ± 0.42	-
SGFormer	**79.53 ± 0.38**	**89.09 ± 0.10**	**73.76 ± 0.24**	**72.63 ± 0.13**	**66.01 ± 0.37**

SGFormer can be trained in full-graph manner on obgn-arxiv

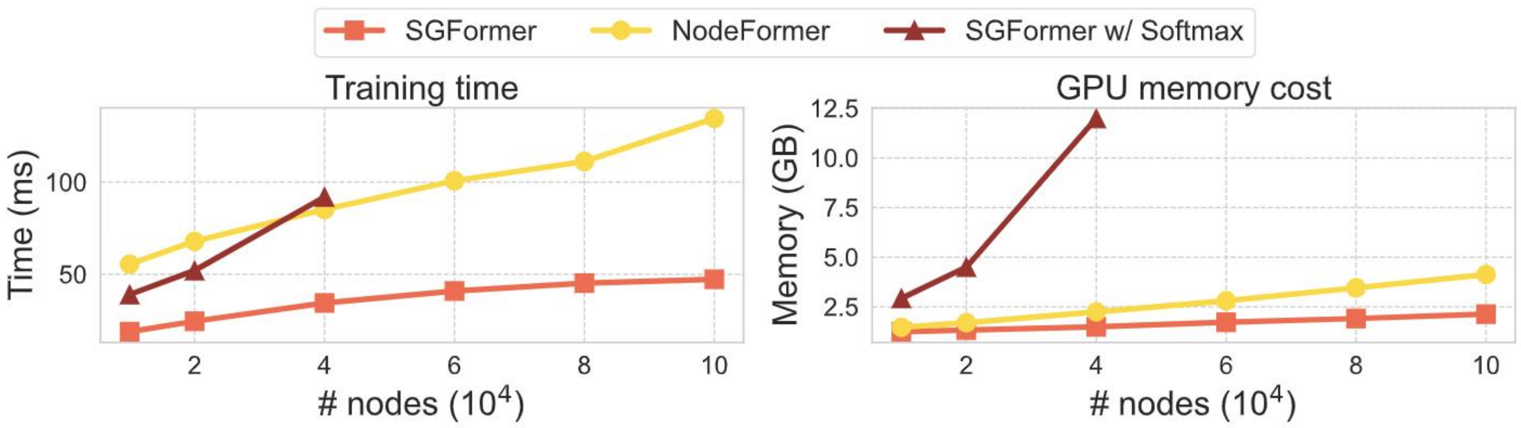
Mini-batch training for proteins, Amazon2M, pokec with batch size 10K/100K

For Papers100M, using batch size 0.4M only requires 3.5 hours on a 24GB GPU

Experiment Results

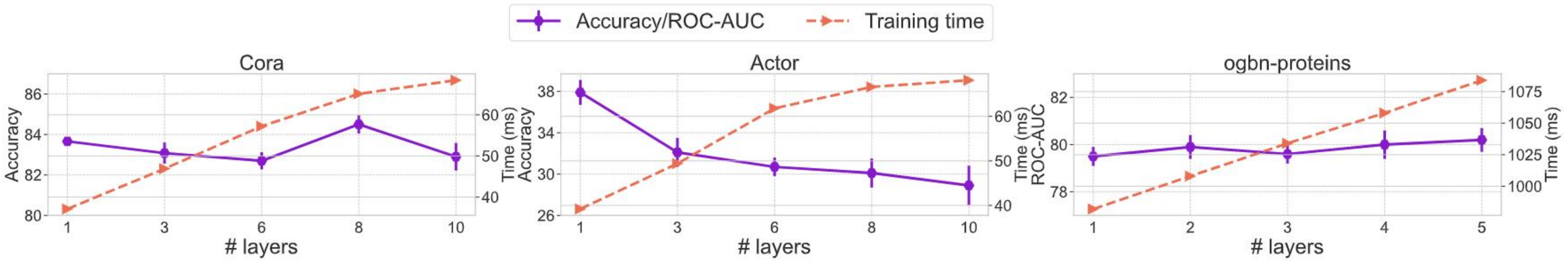
Comparison of training/inference time per epoch and memory cost

Method	Cora			PubMed			Amazon2M		
	Tr (ms)	Inf (ms)	Mem (GB)	Tr (ms)	Inf (ms)	Mem (GB)	Tr (ms)	Inf (ms)	Mem (GB)
Graphormer	563.5	537.1	5.0	-	-	-	-	-	-
GraphTrans	160.4	40.2	3.8	-	-	-	-	-	-
NodeFormer	68.5	30.2	1.2	321.4	135.5	2.9	5369.5	1410.0	4.6
SGFormer	15.0	3.8	0.9	15.4	4.4	1.0	2481.4	382.5	2.7

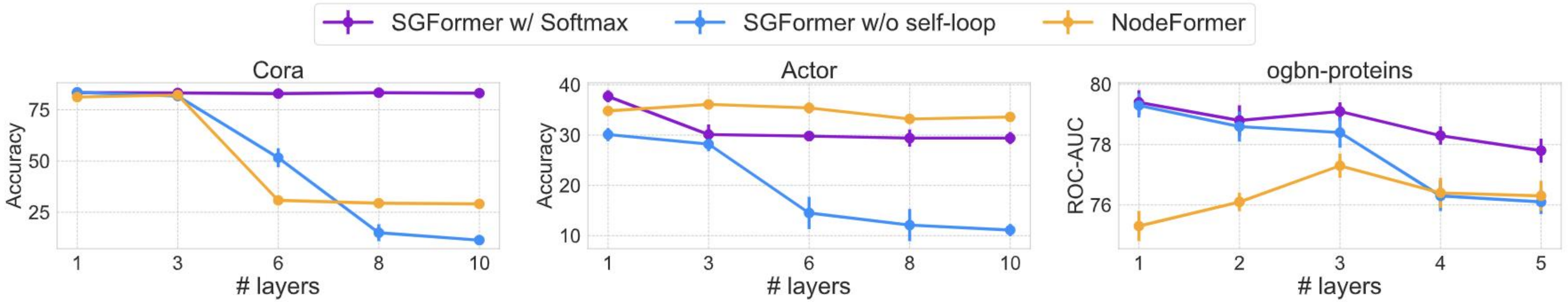


Scalability test of training time/memory costs w.r.t. number of nodes

Experiment Results



Obs 1: one-layer attention of SGFormer is highly competitive and efficient as well



Obs 2: one-layer attention of other (all-pair) models can also yield promising acc

Conclusions

Graph Transformers have become a popular research topic in GNN community

Some open problems: 1) poor scalability (quadratic complexity)

2) lack of principled guidance for attention designs

3) inefficiency, complicated model

[1] NodeFormer: A Scalable Graph Structure Learning Transformer for Node Classification, in NeurIPS 2022

all-pair message passing with linear complexity   scale to 2M nodes   handle no-graph tasks

Codes: https://github.com/qitianwu/NodeFormer

[2] DIFFormer: Scalable (Graph) Transformers Induced by Energy Constrained Diffusion, in ICLR 2023

principled global attention designs   superiority for low labeled rates

Codes: https://github.com/qitianwu/DIFFormer

[3] Simplifying and Empowering Transformers for Large-Graph Representations, in NeurIPS 2023

simple attention (one-layer single-head)   30x inference speed-up   scale to 0.1B nodes

Codes: https://github.com/qitianwu/SGFormer

Email: echo740@sjtu.edu.cn   WeChat: myronwqt228