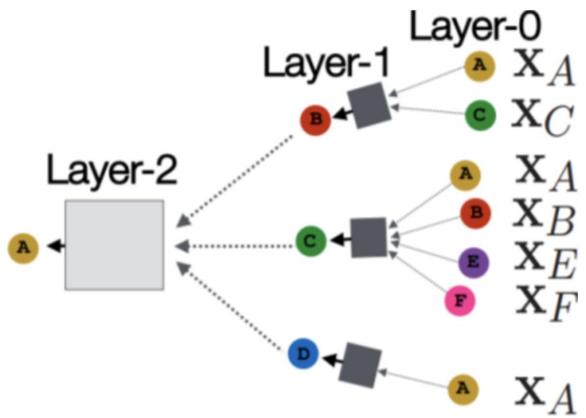
NodeFormer: A Scalable Graph Structure Learning Transformer for Node Classification

Qitian Wu, Wentao Zhao, Zenan Li, David Wipf, Junchi Yan

code

blog

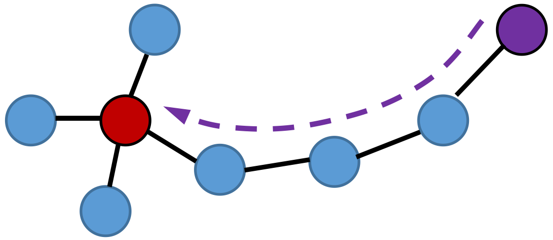
Pitfalls of Graph Neural Networks



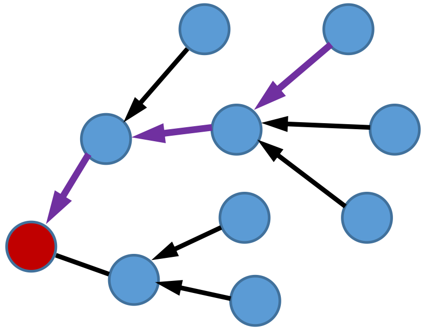
❑ The designs of GNN models:

- Locally aggregate neighbored nodes' features in each layer
- Use other nodes' information for prediction on the target node

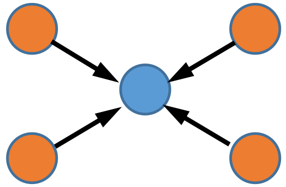
❑ Common scenarios GNNs show deficient power:



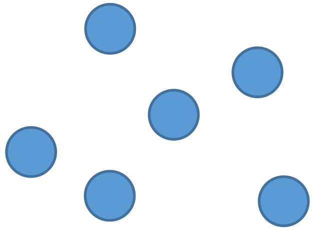
hard to capture long-range dependence [Dai et al., 2018]



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



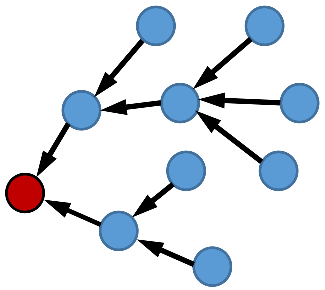
dissimilar linked nodes propagate wrong signals [Zhu et al., 2020]



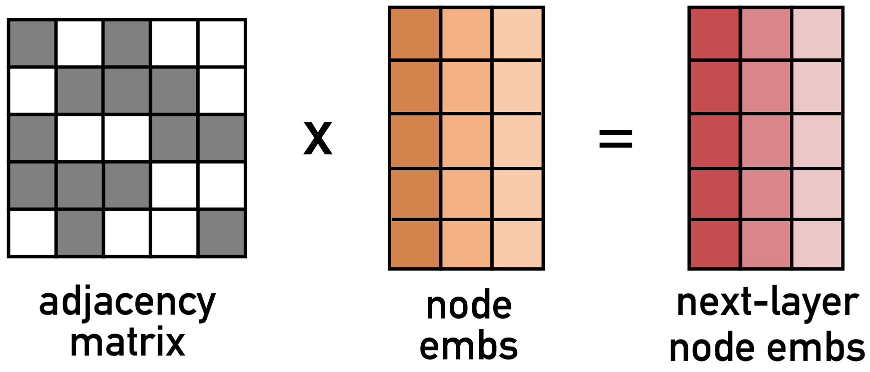
fail to work without input graphs

Message Passing Beyond Input Graphs

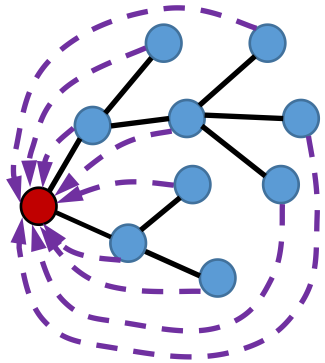
Graph Neural Networks



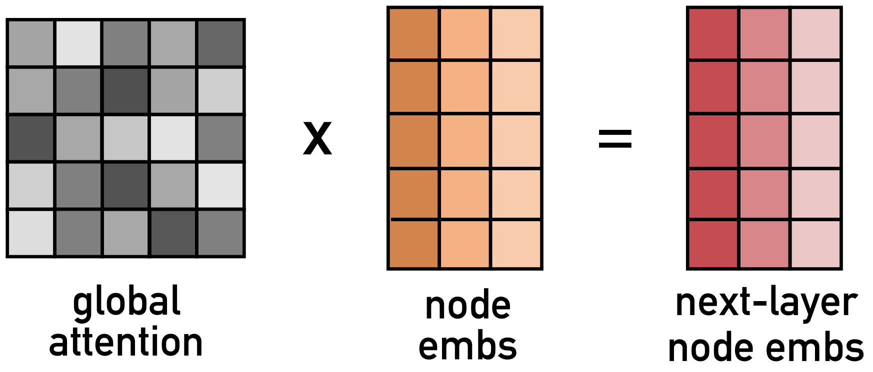
message passing defined over fixed input topology



Transformers

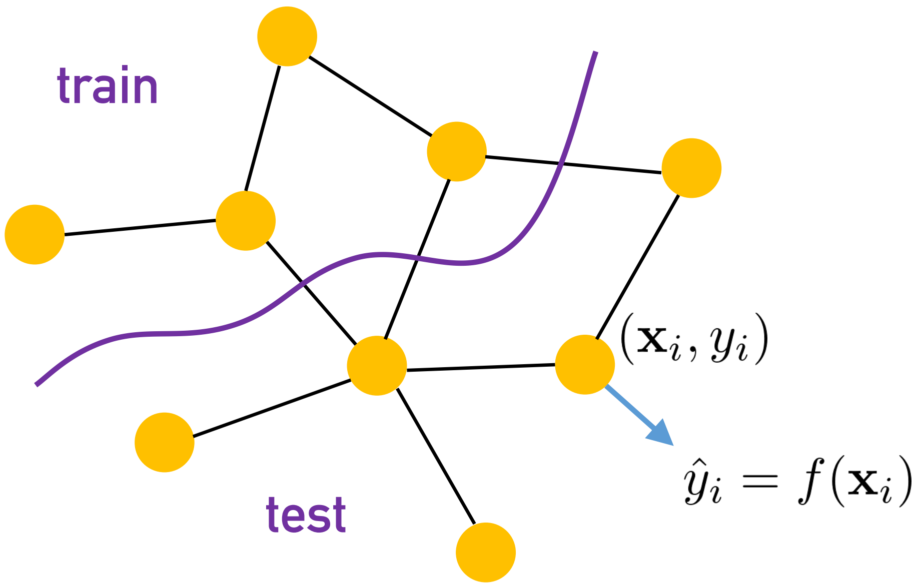


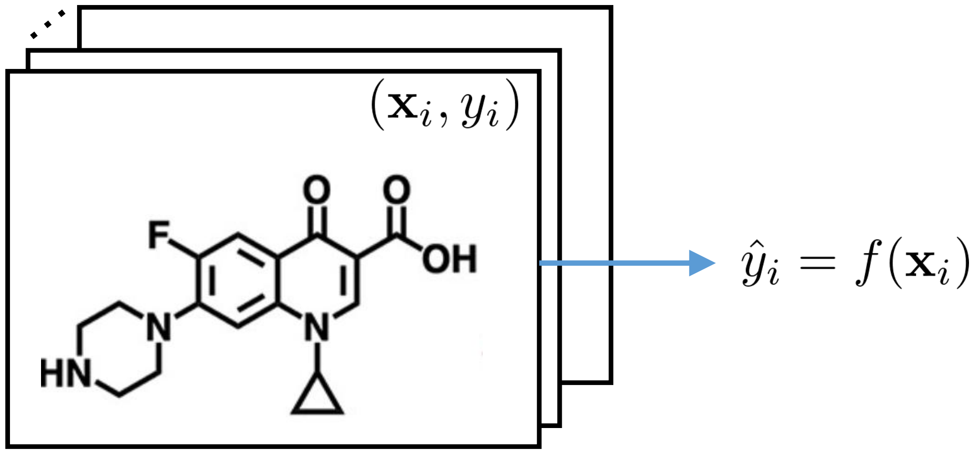
all-pair message passing on layer-specific latent graphs



computational bottleneck $O(N^2)$

Two Problems on Graph Data





Node-Level Prediction/Classification (our focus)

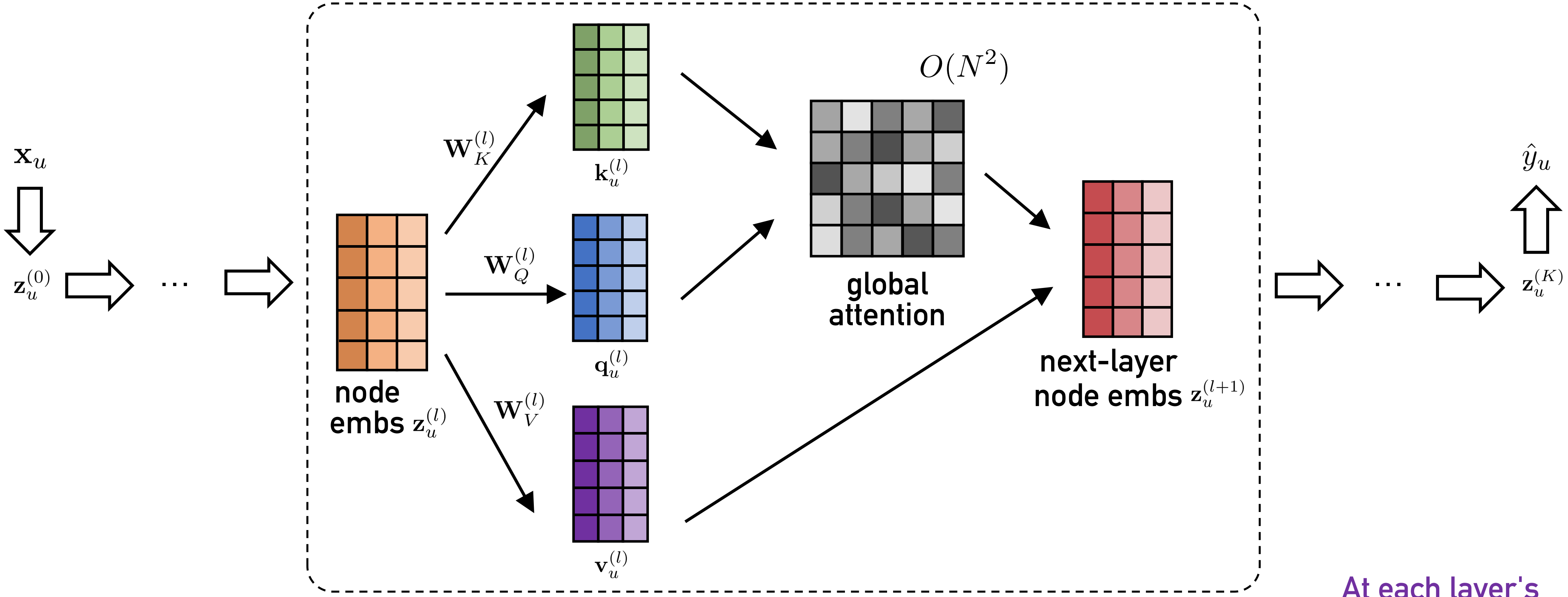
- Each node is an instance with a label
- Train/test on a dataset of nodes in a graph
- The graph is often large (1K-100M nodes)

scalability issue

Graph-Level Prediction/Classification

- Each graph is an instance with a label
- Train/test on a dataset of graphs
- The graphs are often small (e.g., 10-100 nodes)

Transformers for Node Classification



$$\tilde{a}_{uv}^{(l)} = \frac{\exp((W_Q^{(l)}\mathbf{z}_u^{(l)})^\top (W_K^{(l)}\mathbf{z}_v^{(l)}))}{\sum_{w=1}^N \exp((W_Q^{(l)}\mathbf{z}_u^{(l)})^\top (W_K^{(l)}\mathbf{z}_w^{(l)}))}, \quad \mathbf{z}_u^{(l+1)} = \sum_{v=1}^N \tilde{a}_{uv}^{(l)} \cdot (W_V^{(l)}\mathbf{z}_v^{(l)})$$

At each layer's propagation, the model needs to compute attention for each node pair

Scalable All-Pair Message Passing with O(N)

❑ Kernelized softmax message passing

$$\tilde{a}_{uv}^{(l)} = \frac{\exp((W_Q^{(l)}\mathbf{z}_u^{(l)})^\top (W_K^{(l)}\mathbf{z}_v^{(l)}))}{\sum_{w=1}^N \exp((W_Q^{(l)}\mathbf{z}_u^{(l)})^\top (W_K^{(l)}\mathbf{z}_w^{(l)}))}, \quad \mathbf{z}_u^{(l+1)} = \sum_{v=1}^N \tilde{a}_{uv}^{(l)} \cdot (W_V^{(l)}\mathbf{z}_v^{(l)})$$

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

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

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

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

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

only require O(N)
compute the sum at once

❑ Kernelized Gumbel-Softmax

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

$$= \sum_{v=1}^N \frac{\kappa(\mathbf{q}_u/\sqrt{\tau}, \mathbf{k}_v/\sqrt{\tau}) e^{g_v/\tau}}{\sum_{w=1}^N \kappa(\mathbf{q}_u/\sqrt{\tau}, \mathbf{k}_w/\sqrt{\tau}) e^{g_w/\tau}} \cdot \mathbf{v}_v$$

$$\approx \sum_{v=1}^N \frac{\phi(\mathbf{q}_u/\sqrt{\tau})^\top \phi(\mathbf{k}_v/\sqrt{\tau}) e^{g_v/\tau}}{\sum_{w=1}^N \phi(\mathbf{q}_u/\sqrt{\tau})^\top \phi(\mathbf{k}_w/\sqrt{\tau}) e^{g_w/\tau}} \cdot \mathbf{v}_v$$

$$= \frac{\phi(\mathbf{q}_u/\sqrt{\tau})^\top \sum_{v=1}^N e^{g_v/\tau} \phi(\mathbf{k}_v/\sqrt{\tau}) \cdot \mathbf{v}_v^\top}{\phi(\mathbf{q}_u/\sqrt{\tau})^\top \sum_{w=1}^N e^{g_w/\tau} \phi(\mathbf{k}_w/\sqrt{\tau})}$$

approximate sampling discrete edges from a potential, large graph that connects ***all nodes***

Approximation Error and Concentration

Theorem 1 (Approximation Error for Softmax-Kernel)

Assume $\|\mathbf{q}_u\|_2$ and $\|\mathbf{k}_v\|_2$ are bounded by r, and ϕ the Positive Random Features, then with probability at least $1-\epsilon$, the approximation error gap will be bounded by

$$\Delta = \left|\phi(\mathbf{q}_u/\sqrt{\tau})^\top \phi(\mathbf{k}_v/\sqrt{\tau}) - \kappa(\mathbf{q}_u/\sqrt{\tau}, \mathbf{k}_v/\sqrt{\tau})\right| \leq \mathcal{O}\left(\sqrt{\frac{\exp(6r/\tau)}{m\epsilon}}\right)$$

m for random feature dimension, τ for temperature

the error is independent of node number N

Theorem 2 (Concentration of Kernelized Gumbel-Softmax Random Variables)

Suppose the random feature dimension m is sufficiently large, we have the convergence property for the kernelized Gumbel-Softmax operator

$$\lim_{\tau \to 0} \mathbb{P}(c_{uv} > c_{uv'}, \forall v' \neq v) = \frac{\exp(\mathbf{q}_u^\top \mathbf{k}_v)}{\sum_{w=1}^N \exp(\mathbf{q}_u^\top \mathbf{k}_w)}, \quad \lim_{\tau \to 0} \mathbb{P}(c_{uv} = 1) = \frac{\exp(\mathbf{q}_u^\top \mathbf{k}_v)}{\sum_{w=1}^N \exp(\mathbf{q}_u^\top \mathbf{k}_w)}$$

The sampled results converge to the ones induced by the Softmax categorical distribution

Scalable All-Pair Message Passing with O(N)

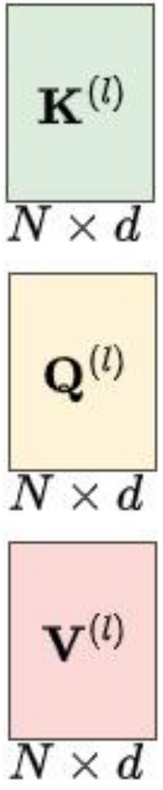
Algorithm 1: Scalable All-Pair Message Passing on Latent Graphs with Linear Complexity ($\mathcal{O}(N)$ or $\mathcal{O}(N+E)$)

Input: Node features $\mathbf{Z}^{(0)} = \mathbf{X}$, input adjacency $\mathbf{A}$.

for $l = 0 \ldots, L-1$ **do**

$\quad \mathbf{Q}^{(l)} \leftarrow W_Q^{(l)} \mathbf{Z}^{(l)}, \mathbf{K}^{(l)} \leftarrow W_K^{(l)} \mathbf{Z}^{(l)}, \mathbf{V}^{(l)} \leftarrow W_V^{(l)} \mathbf{Z}^{(l)};$

Output: Predict node labels $\hat{\mathbf{Y}} = \text{MLP}(\{\mathbf{Z}^{(l)}\}_{l=0}^{L})$.



Scalable All-Pair Message Passing with O(N)

Algorithm 1: Scalable All-Pair Message Passing on Latent Graphs with Linear Complexity ($\mathcal{O}(N)$ or $\mathcal{O}(N+E)$)

Input: Node features $\mathbf{Z}^{(0)} = \mathbf{X}$, input adjacency $\mathbf{A}$.

for $l = 0 \ldots, L-1$ **do**

  $\mathbf{Q}^{(l)} \leftarrow W_Q^{(l)} \mathbf{Z}^{(l)}, \mathbf{K}^{(l)} \leftarrow W_K^{(l)} \mathbf{Z}^{(l)}, \mathbf{V}^{(l)} \leftarrow W_V^{(l)} \mathbf{Z}^{(l)};$

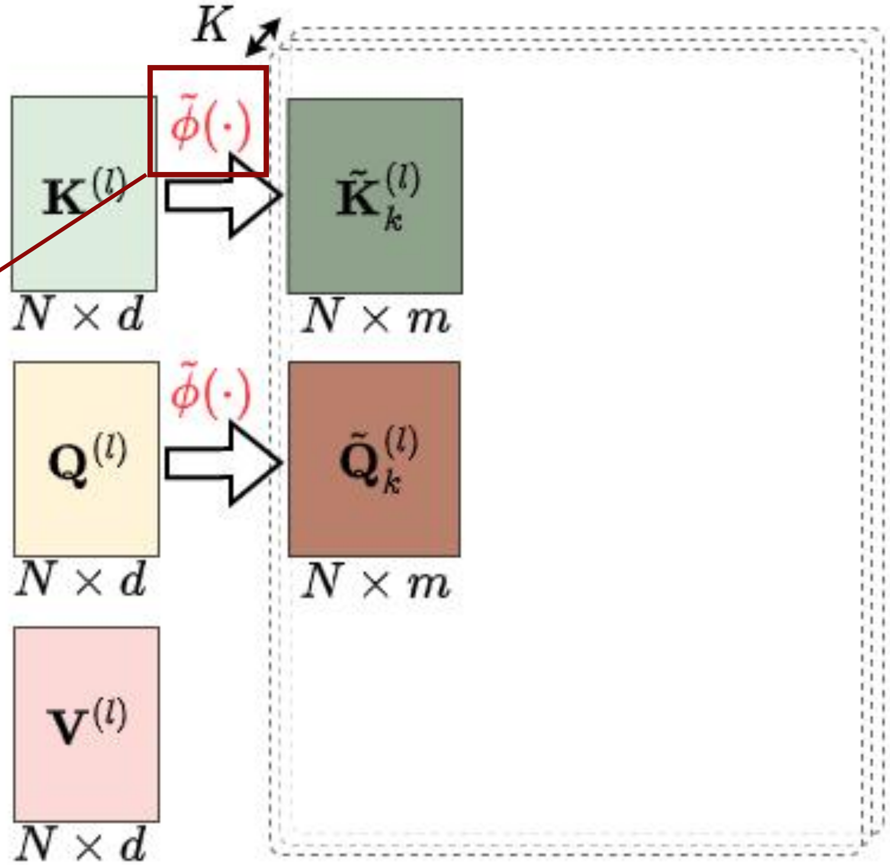
  for $k = 1, 2, \ldots, K$ **do**

    $G_k = \{e^{g_{ku}/\tau}\}_{u=1}^{N}, \;\; g_{ku} \sim Gumbel(0,1);$

    $\tilde{G}_k = G_k.unsqueeze(1).repeat(1, m);$

    $\tilde{\mathbf{K}}_k^{(l)} = \tilde{G}_k \odot \phi(\mathbf{K}^{(l)}/\sqrt{\tau}), \tilde{\mathbf{Q}}_k^{(l)} = \tilde{G}_k \odot \phi(\mathbf{Q}^{(l)}/\sqrt{\tau});$

Output: Predict node labels $\hat{\mathbf{Y}} = \mathrm{MLP}(\{\mathbf{Z}^{(l)}\}_{l=0}^{L})$.



Scalable All-Pair Message Passing with O(N)

Algorithm 1: Scalable All-Pair Message Passing on Latent Graphs with Linear Complexity ($\mathcal{O}(N)$ or $\mathcal{O}(N+E)$)

Input: Node features $\mathbf{Z}^{(0)} = \mathbf{X}$, input adjacency $\mathbf{A}$.

for $l = 0 \ldots, L-1$ **do**

  $\mathbf{Q}^{(l)} \leftarrow W_Q^{(l)}\mathbf{Z}^{(l)}, \mathbf{K}^{(l)} \leftarrow W_K^{(l)}\mathbf{Z}^{(l)}, \mathbf{V}^{(l)} \leftarrow W_V^{(l)}\mathbf{Z}^{(l)}$;

  for $k = 1, 2, \ldots, K$ **do**

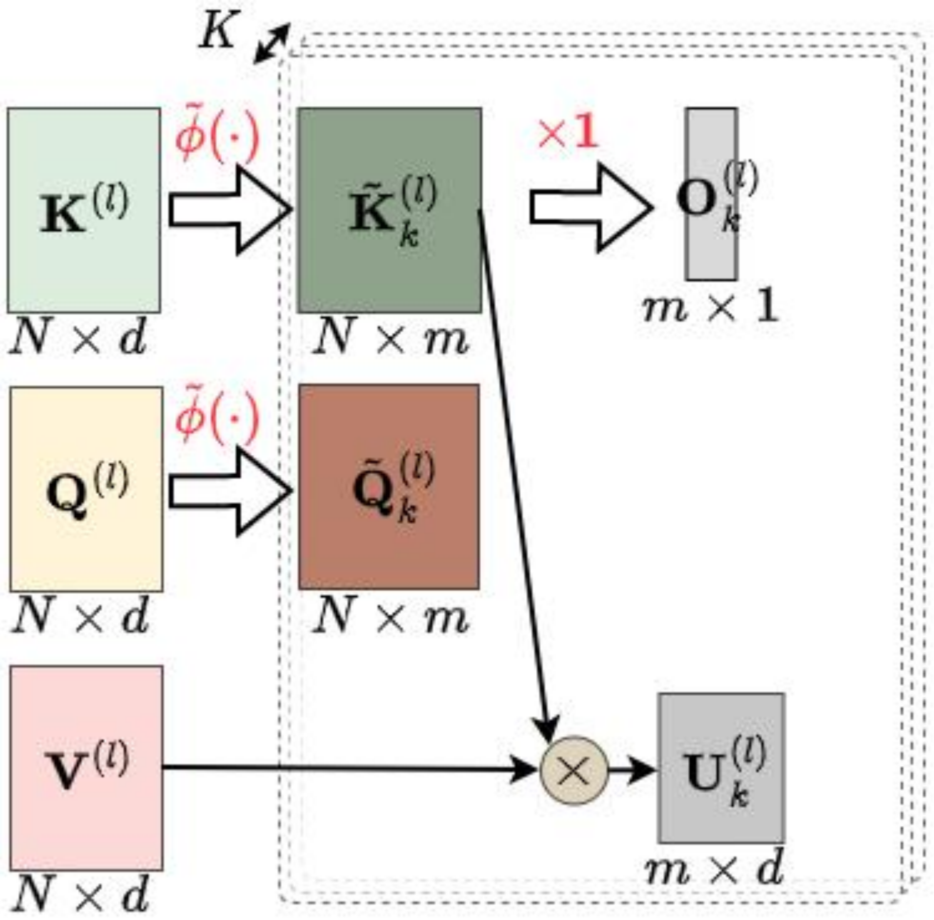
    $G_k = \{e^{g_{ku}/\tau}\}_{u=1}^{N}, \ g_{ku} \sim Gumbel(0,1)$;

    $\tilde{G}_k = G_k.unsqueeze(1).repeat(1, m)$;

    $\tilde{\mathbf{K}}_k^{(l)} = \tilde{G}_k \odot \phi(\mathbf{K}^{(l)}/\sqrt{\tau}), \tilde{\mathbf{Q}}_k^{(l)} = \tilde{G}_k \odot \phi(\mathbf{Q}^{(l)}/\sqrt{\tau})$;

    $\mathbf{U}_k^{(l)} \leftarrow (\tilde{\mathbf{K}}_k^{(l)})^\top \mathbf{V}^{(l)}, \mathbf{O}_k^{(l)} \leftarrow (\tilde{\mathbf{K}}_k^{(l)})^\top \mathbf{1}_{N\times 1}$;

Output: Predict node labels $\hat{\mathbf{Y}} = \mathrm{MLP}(\{\mathbf{Z}^{(l)}\}_{l=0}^{L})$.



Scalable All-Pair Message Passing with O(N)

Algorithm 1: Scalable All-Pair Message Passing on Latent Graphs with Linear Complexity ($\mathcal{O}(N)$ or $\mathcal{O}(N+E)$)

Input: Node features $\mathbf{Z}^{(0)} = \mathbf{X}$, input adjacency $\mathbf{A}$.

for $l = 0 \dots, L-1$ **do**

  $\mathbf{Q}^{(l)} \leftarrow W_Q^{(l)} \mathbf{Z}^{(l)}$, $\mathbf{K}^{(l)} \leftarrow W_K^{(l)} \mathbf{Z}^{(l)}$, $\mathbf{V}^{(l)} \leftarrow W_V^{(l)} \mathbf{Z}^{(l)}$;

  for $k = 1, 2, \dots, K$ **do**

    $G_k = \{e^{g_{ku}/\tau}\}_{u=1}^{N}, \ \ g_{ku} \sim Gumbel(0,1)$;

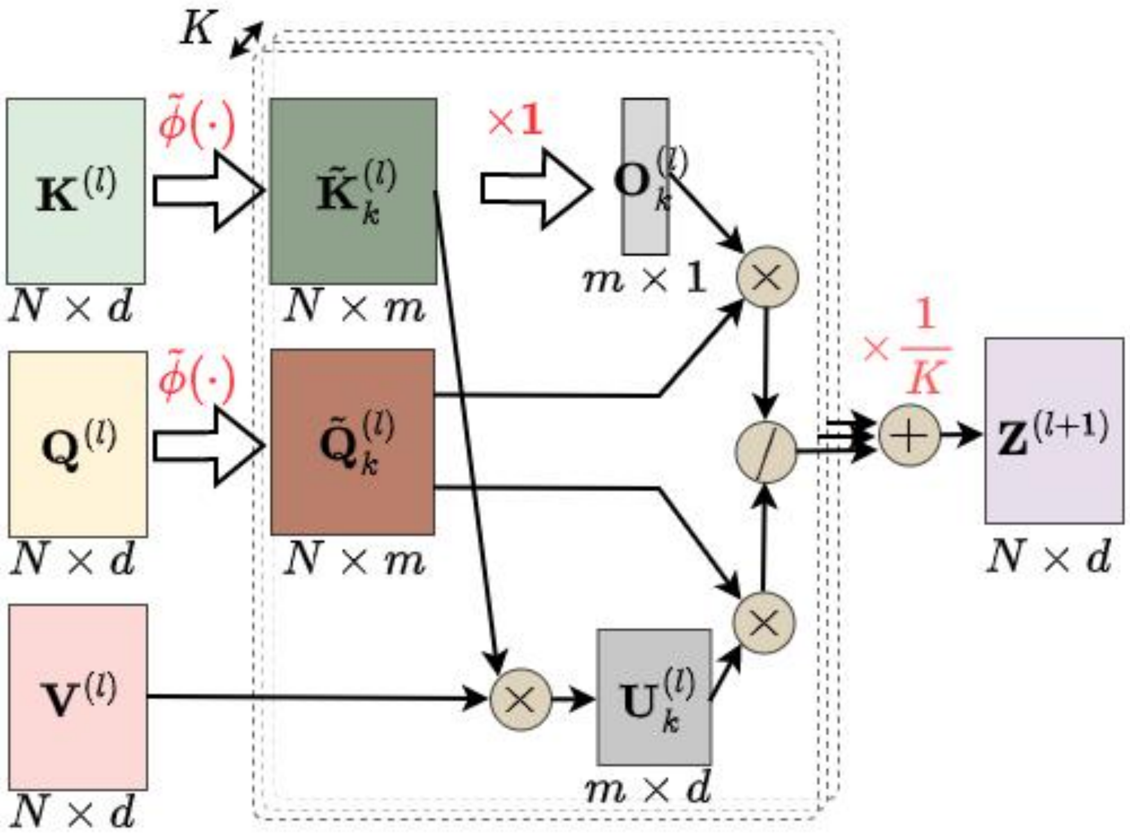
    $\tilde{G}_k = G_k.unsqueeze(1).repeat(1, m)$;

    $\tilde{\mathbf{K}}_k^{(l)} = \tilde{G}_k \odot \phi(\mathbf{K}^{(l)}/\sqrt{\tau})$, $\tilde{\mathbf{Q}}_k^{(l)} = \tilde{G}_k \odot \phi(\mathbf{Q}^{(l)}/\sqrt{\tau})$;

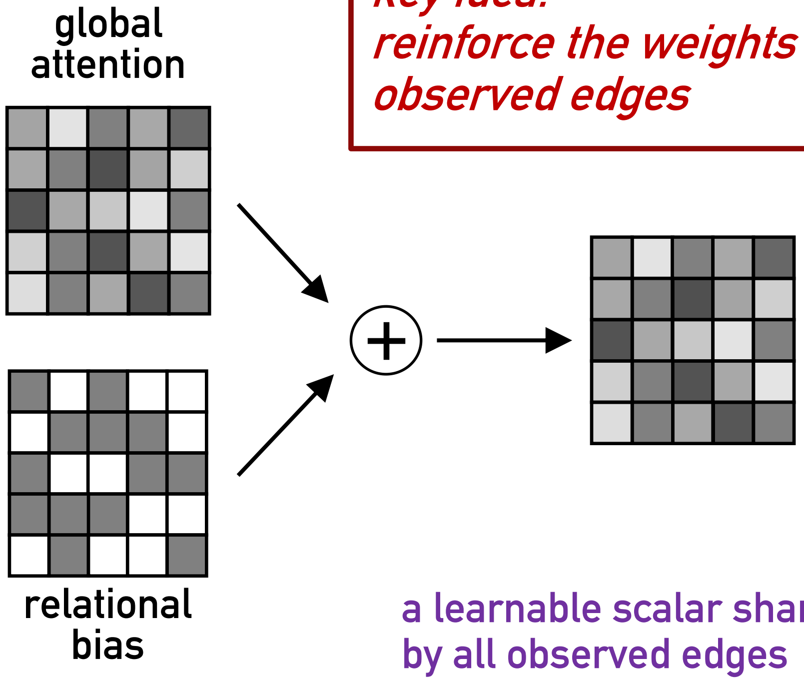
    $\mathbf{U}_k^{(l)} \leftarrow (\tilde{\mathbf{K}}_k^{(l)})^\top \mathbf{V}^{(l)}$, $\mathbf{O}_k^{(l)} \leftarrow (\tilde{\mathbf{K}}_k^{(l)})^\top \mathbf{1}_{N\times 1}$;

  $\mathbf{Z}^{(l+1)} \leftarrow \frac{1}{K} \sum_{k=1}^{K} \frac{\tilde{\mathbf{Q}}_k^{(l)} \mathbf{U}_k^{(l)}}{\tilde{\mathbf{Q}}_k^{(l)} \mathbf{O}_k^{(l)}}$; % average K samples

Output: Predict node labels $\hat{\mathbf{Y}} = \mathrm{MLP}(\{\mathbf{Z}^{(l)}\}_{l=0}^{L})$.



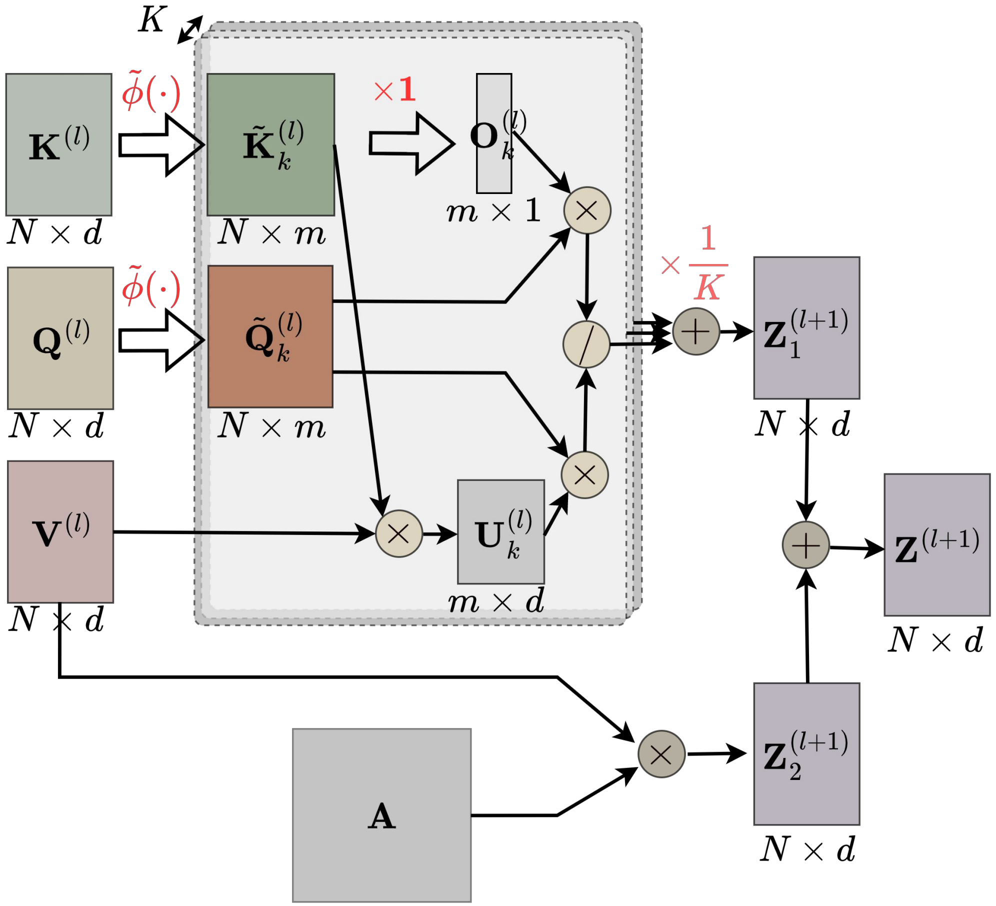
Input Graphs as Relational Bias



key idea:
reinforce the weights for observed edges

a learnable scalar shared by all observed edges

$$\mathbf{z}_u^{(l+1)} \leftarrow \mathbf{z}_u^{(l+1)} + \sum_{v, a_{uv}=1} \sigma(b^{(l)}) \cdot \mathbf{v}_v$$



$O(N + E)$

Input Graphs as Regularization Loss

❑ Supervised classification loss

$$\mathcal{L}_s(\mathbf{Y}, \hat{\mathbf{Y}}) = -\frac{1}{N}\sum_{v=1}^{N}\sum_{c=1}^{C}\mathbb{I}[y_u = c]\log \hat{y}_{u,c}$$

❑ Edge-level regularization loss

$$\mathcal{L}_e(\mathbf{A}, \tilde{\mathbf{A}}) = -\frac{1}{NL}\sum_{l=1}^{L}\sum_{(u,v)\in\mathcal{E}}\frac{1}{d_u}\log \pi_{uv}^{(l)}$$

$$\pi_{uv}^{(l)} = \frac{\phi(W_Q^{(l)}\mathbf{z}_u^{(l)})^\top \phi(W_K^{(l)}\mathbf{z}_v^{(l)})}{\phi(W_Q^{(l)}\mathbf{z}_u^{(l)})^\top \sum_{w=1}^{N}\phi(W_K^{(l)}\mathbf{z}_w^{(l)})}$$

❑ Final loss function

$$\mathcal{L} = \mathcal{L}_s + \lambda\mathcal{L}_e$$

Key observation:

labeled nodes < N << N^2 = # node pairs

The log-likelihood of observed edges, if assuming data distribution as

$$p_0(v|u) = \begin{cases} \frac{1}{d_u}, & a_{uv} = 1 \\ 0, & otherwise. \end{cases}$$

only require O(E)

Since we only need to query the probability for each observed edges, where the complexity of each query is $\mathcal{O}(1)$

Dissecting the Rationale of New Objective

❑ A variational perspective look at the training objective

Key insights:

Treat the latent structure estimation as a variational distribution

The all-pair message passing module induces a predictive distribution $q(\tilde{\mathbf{A}}|\mathbf{X},\mathbf{A})$ $p(\mathbf{Y}|\tilde{\mathbf{A}},\mathbf{X},\mathbf{A})$

$$\mathcal{L}_e(\mathbf{A},\tilde{\mathbf{A}}) = -\frac{1}{NL}\sum_{l=1}^{L}\sum_{(u,v)\in\mathcal{E}}\frac{1}{d_u}\log \pi_{uv}^{(l)}$$

$$\mathcal{L}_s(\mathbf{Y},\hat{\mathbf{Y}}) = -\frac{1}{N}\sum_{v=1}^{N}\sum_{c=1}^{C}\mathbb{I}[y_u = c]\log \hat{y}_{u,c}$$

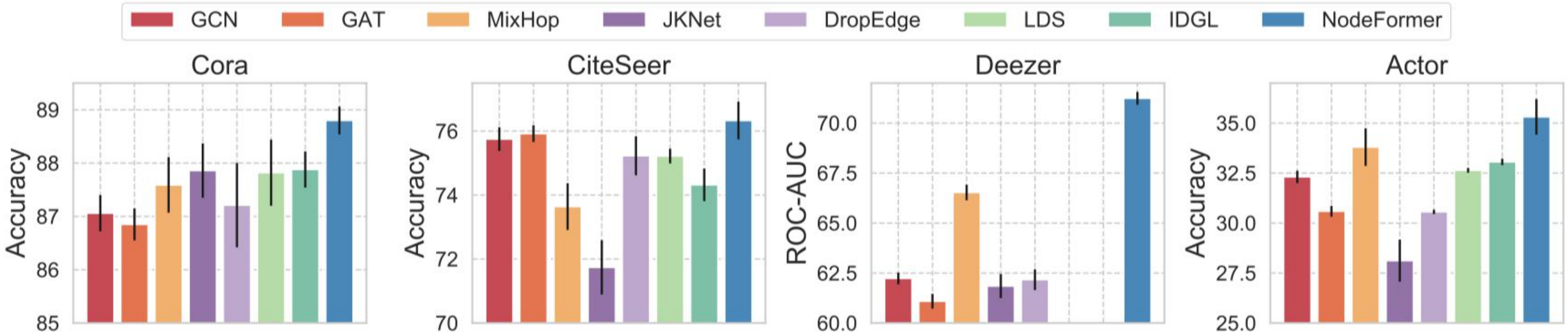
$$p^*, q^* = \arg\min_{p,q} \underbrace{-\mathbb{E}_q[\log p(\mathbf{Y}|\tilde{\mathbf{A}},\mathbf{X},\mathbf{A})]}_{\mathcal{L}_s} + \underbrace{\mathcal{D}(q(\tilde{\mathbf{A}}|\mathbf{X},\mathbf{A})\|p_0(\tilde{\mathbf{A}}|\mathbf{X},\mathbf{A}))}_{\mathcal{L}_e}$$

Proposition (Underlying Effect for Learning Optimal Structures)

Assume q can exploit arbitrary distributions over $\tilde{\mathbf{A}}$. When the objective achieves the optimum, we have 1) $\mathcal{D}(q(\tilde{\mathbf{A}}|\mathbf{X},\mathbf{A})\|p(\tilde{\mathbf{A}}|\mathbf{Y},\mathbf{X},\mathbf{A})) = 0$, and 2) $\log p(\mathbf{Y}|\mathbf{X},\mathbf{A})$ is maximized.

Comparative Experiments

❑ Experiment on small node classification benchmarks



LDS [Franceschi et al., 2020]
IDGL [Chen et al., 2021]

❑ Experiment on large-scale datasets OGB-Proteins and Amazon2M

Method	Accuracy (%)	Train Mem
MLP	63.46 ± 0.10	1.4 GB
GCN	83.90 ± 0.10	5.7 GB
SGC	81.21 ± 0.12	1.7 GB
GraphSAINT-GCN	83.84 ± 0.42	2.1 GB
GraphSAINT-GAT	85.17 ± 0.32	2.2 GB
NodeFormer	**87.85** ± 0.24	4.0 GB
NodeFormer-dt	87.02 ± 0.75	2.9 GB
NodeFormer-tp	87.55 ± 0.11	4.0 GB

NodeFormer successfully scales to graphs with 2M nodes

NodeFormer using batch size 0.1M only requires 4GB memory and hours for training on a single GPU

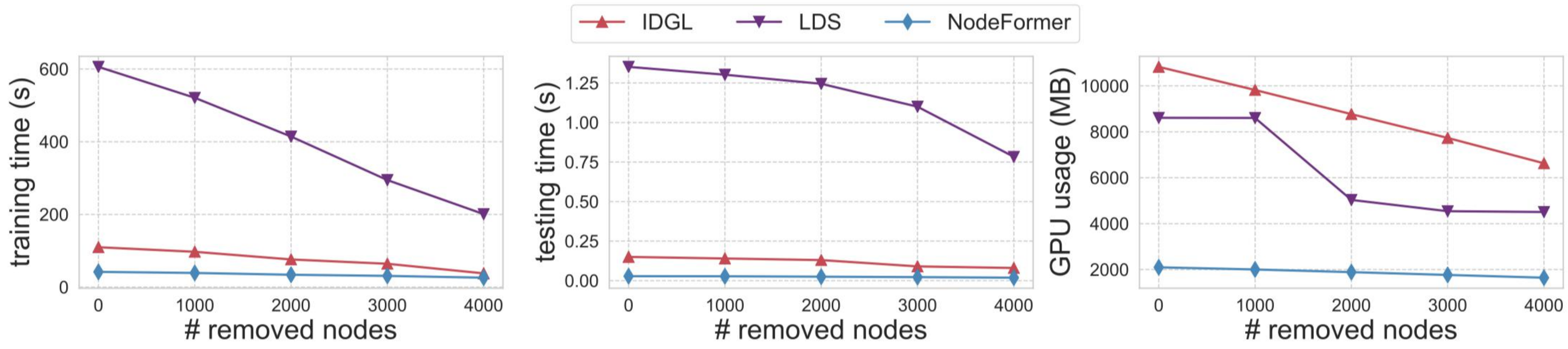
Comparative Experiments

❑ Experiment on image/text classification (no input graph)

Method	Mini-ImageNet $k=5$	Mini-ImageNet $k=10$	Mini-ImageNet $k=15$	Mini-ImageNet $k=20$	20News-Group $k=5$	20News-Group $k=10$	20News-Group $k=15$	20News-Group $k=20$
GCN	84.86 ± 0.42	85.61 ± 0.40	85.93 ± 0.59	85.96 ± 0.66	65.98 ± 0.68	64.13 ± 0.88	62.95 ± 0.70	62.59 ± 0.62
GAT	84.70 ± 0.48	85.24 ± 0.42	85.41 ± 0.43	85.37 ± 0.51	64.06 ± 0.44	62.51 ± 0.71	61.38 ± 0.88	60.80 ± 0.59
DropEdge	83.91 ± 0.24	85.35 ± 0.44	85.25 ± 0.63	85.81 ± 0.65	64.46 ± 0.43	64.01 ± 0.42	62.46 ± 0.51	62.68 ± 0.71
IDGL	83.63 ± 0.32	84.41 ± 0.35	85.50 ± 0.24	85.66 ± 0.42	65.09 ± 1.23	63.41 ± 1.26	61.57 ± 0.52	62.21 ± 0.79
LDS	OOM	OOM	OOM	OOM	**66.15** ± 0.36	64.70 ± 1.07	63.51 ± 0.64	63.51 ± 1.75
NodeFormer	**86.77** ± 0.45	**86.74** ± 0.23	**86.87** ± 0.41	**86.64** ± 0.42	66.01 ± 1.18	**65.21** ± 1.14	**64.69** ± 1.31	**64.55** ± 0.97
NodeFormer w/o graph	**87.46** ± 0.36				**64.71** ± 1.33			

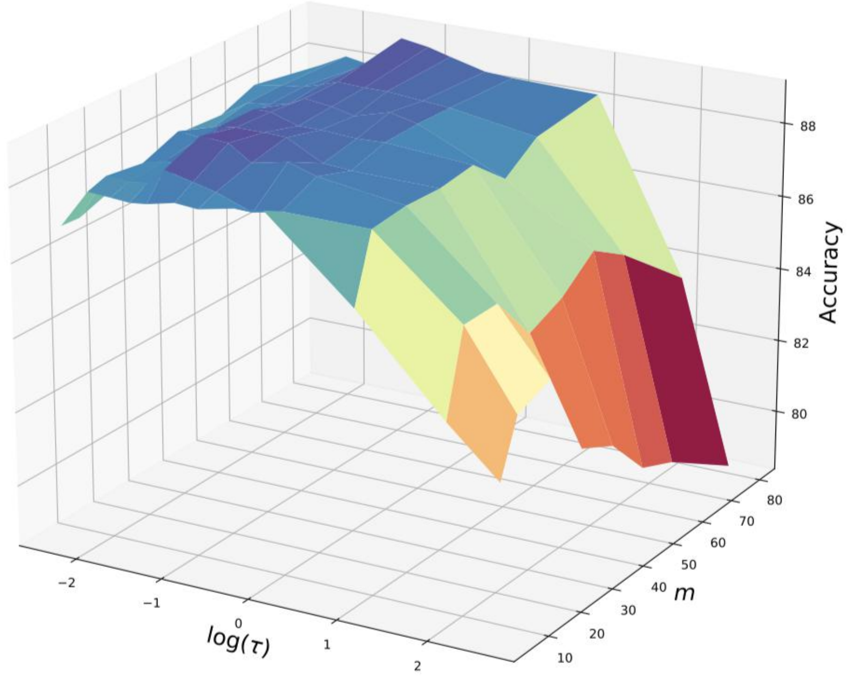
NodeFormer also works with no input graph

❑ Scalability analysis on time/space costs



NodeFormer reduces training time by 93.1%

Ablation Study and Hyper-parameters



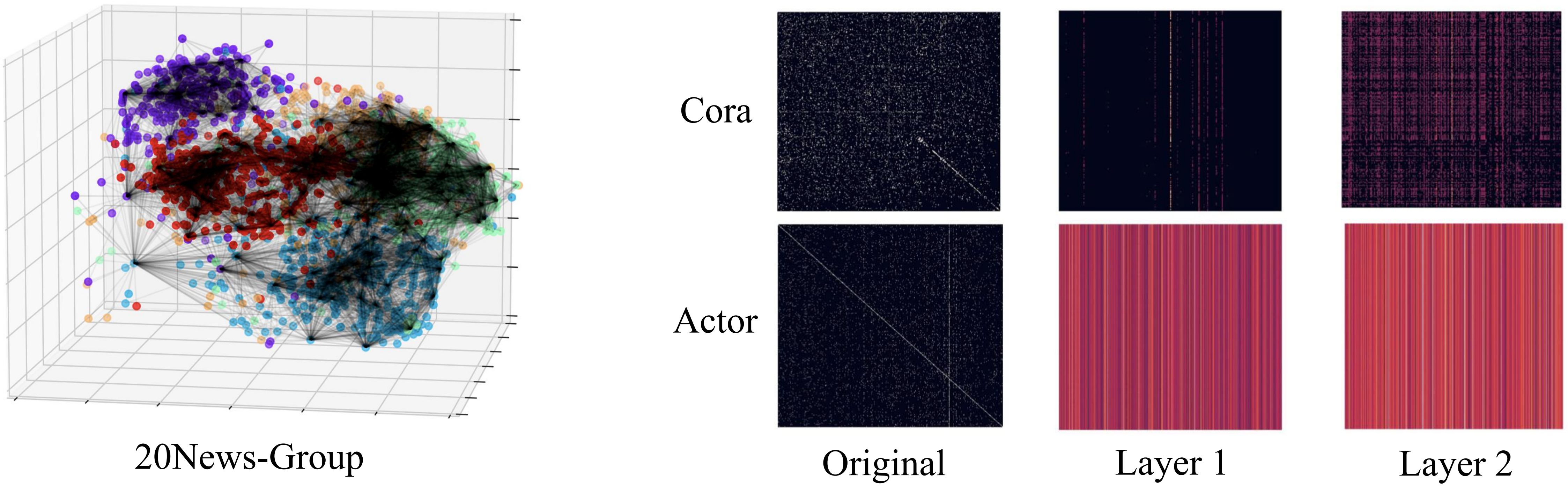
Larger random feature dimension m allows better approximation

Moderate temperature (tau=0.25) yields stably good performance

Dataset	NodeFormer	NodeFormer w/o reg	NodeFormer w/o rb
Cora	**88.69** ± 0.46	81.98 ± 0.46	88.06 ± 0.59
Citeseer	**76.33** ± 0.59	70.60 ± 1.20	74.12 ± 0.64
Deezer	**71.24** ± 0.32	71.22 ± 0.32	71.10 ± 0.36
Actor	**35.31** ± 1.29	35.15 ± 1.32	34.60 ± 1.32

Ablation study on edge regularization loss and relational bias

Visualization of Learned Structures



The latent structures produced by NodeFormer tend to connect nodes within the same class and increase the overall connectivity of the whole graph

Comparison with Existing Graph Transformers

Prior Art	NodeFormer
quadratic complexity (hard to scale to 10K nodes)	linear complexity (largest demonstration on 2M nodes)
most desgined for graph classification (a dataset of small graphs)	desgined for node classification (a dataset of nodes with inter-connection)
require postional embedding (preprocess node/edge features)	use relational bias and edge regularization loss for using input graph information

Resources and Related Materials

code

blog

https://github.com/qitianwu/NodeFormer

https://zhuanlan.zhihu.com/p/587086593

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

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

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