Towards Graph Transformers at Scale

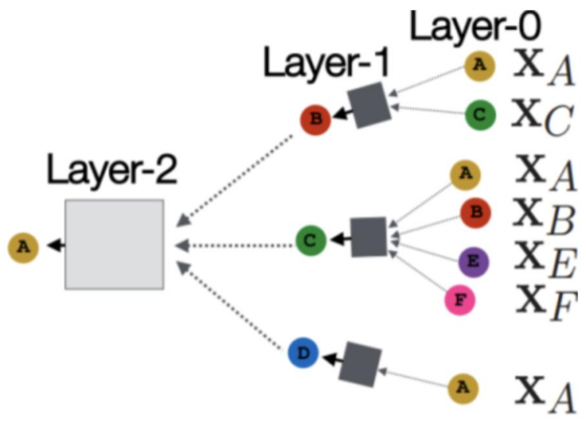
Qitian Wu (吴齐天)
Department of Computer Science and Engineering
Shanghai Jiao Tong University

[1] Qitian Wu et al., *NodeFormer: A Scalable Graph Structure Learning Transformer for Node Classification*, NeurIPS 2022 (spotlight, top 5%)

[2] Qitian Wu et al., *DIFFormer: Scalable (Graph) Transformers Induced by Energy Constrained Diffusion*, ICLR 2023 (spotlight oral, top 0.5%)

[3] Qitian Wu et al., SGFormer: *Simplifying and Empowering Transformers for Large-Graph Representations*, NeurIPS 2023

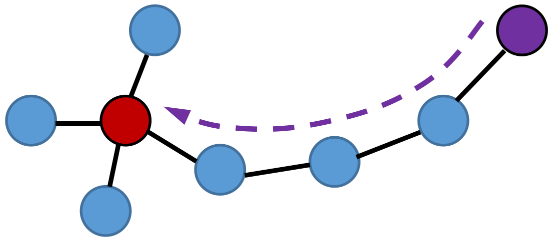
Pitfalls of Graph Neural Networks



❑ The designs of mainstream GNNs:

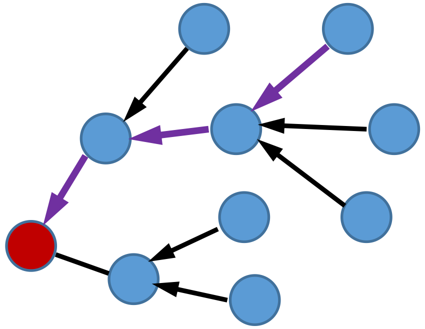
- Locally aggregate neighbored nodes' features in each layer
- Use neighbored nodes' embs for informative represensation

❑ Common scenarios GNNs show deficient capability:



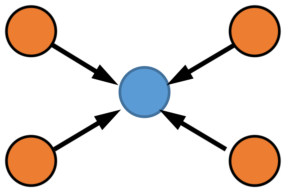
hard to capture long-range dependence [Dai et al., 2018]

long-range reasoning



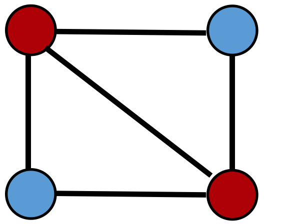
distant signals are overly squashed [Alon et al., 2021]

over-squashing



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

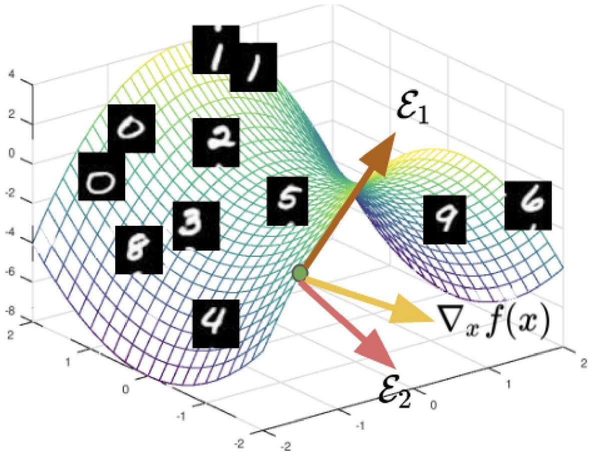
heterophily



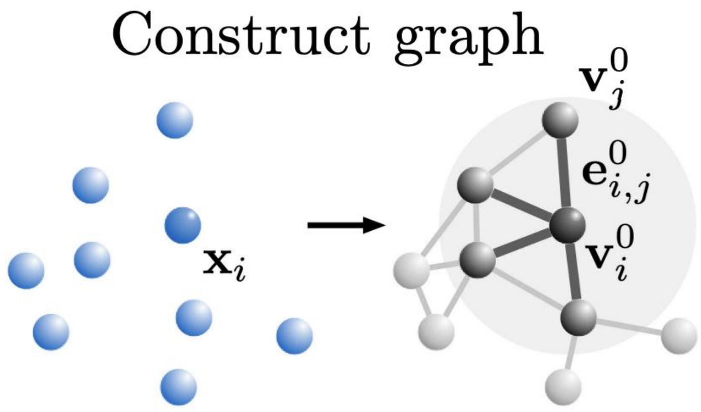
fail to distinguish two similar inputs [Xu et al., 2019]

expressivity

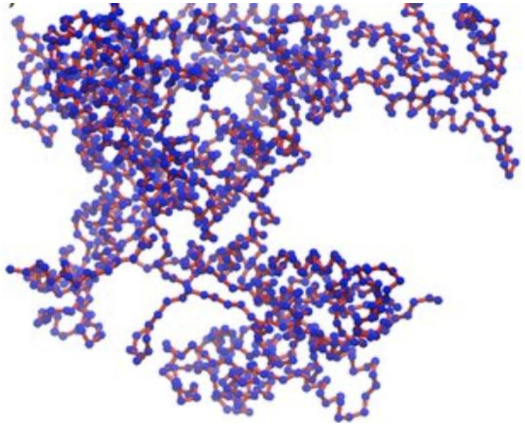
Inter-Dependent Data without Input Graphs



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



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

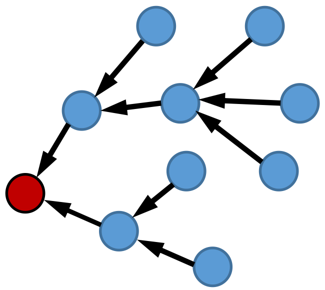


Complex hidden structures beyond observed geometry [Xu et al., 2020]

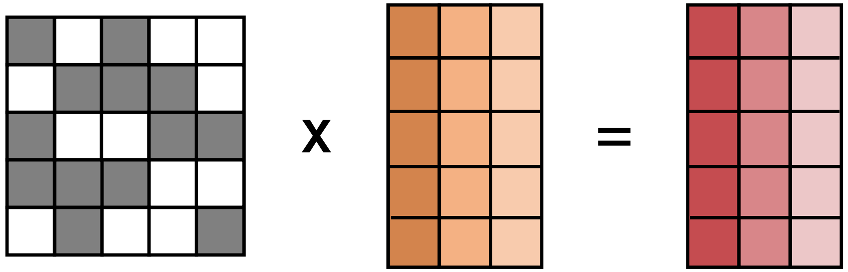
- **GNNs require observed graphs as input:**
  - Solution: Pre-define a graph by some rules (e.g., k nearest neighbors)
  - Limitation: the pre-defined graph is independent of downstream tasks

Message Passing Beyond Input Graphs

Graph Neural Networks

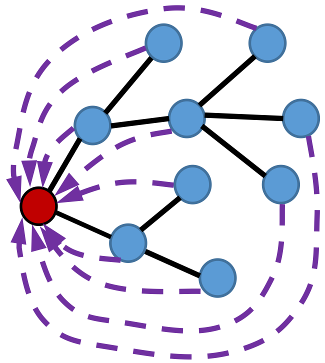


message passing defined over fixed input topology

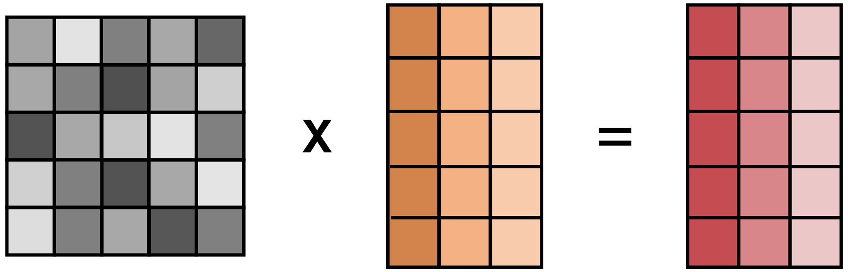


only require $O(E)$ when using sparse matrix computation

Transformers



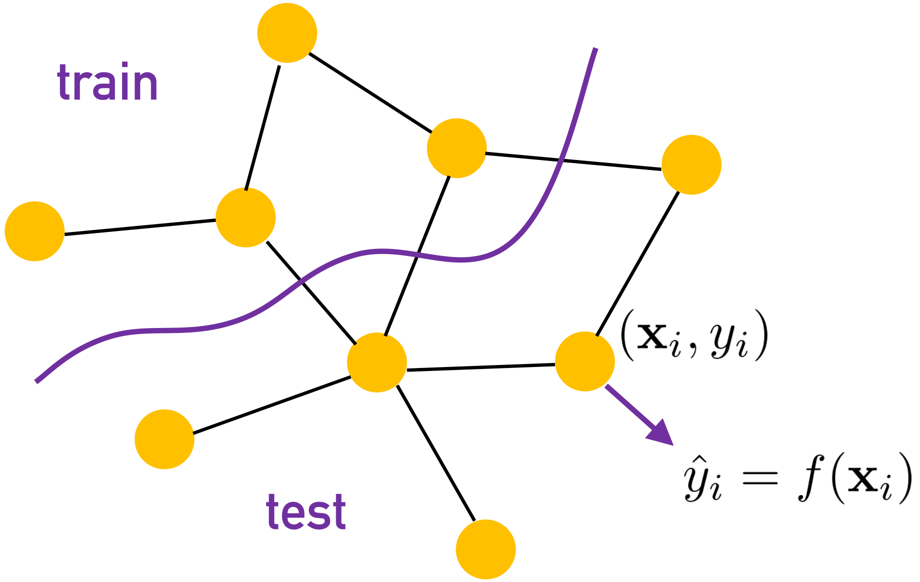
all-pair message passing on layer-specific latent graphs



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

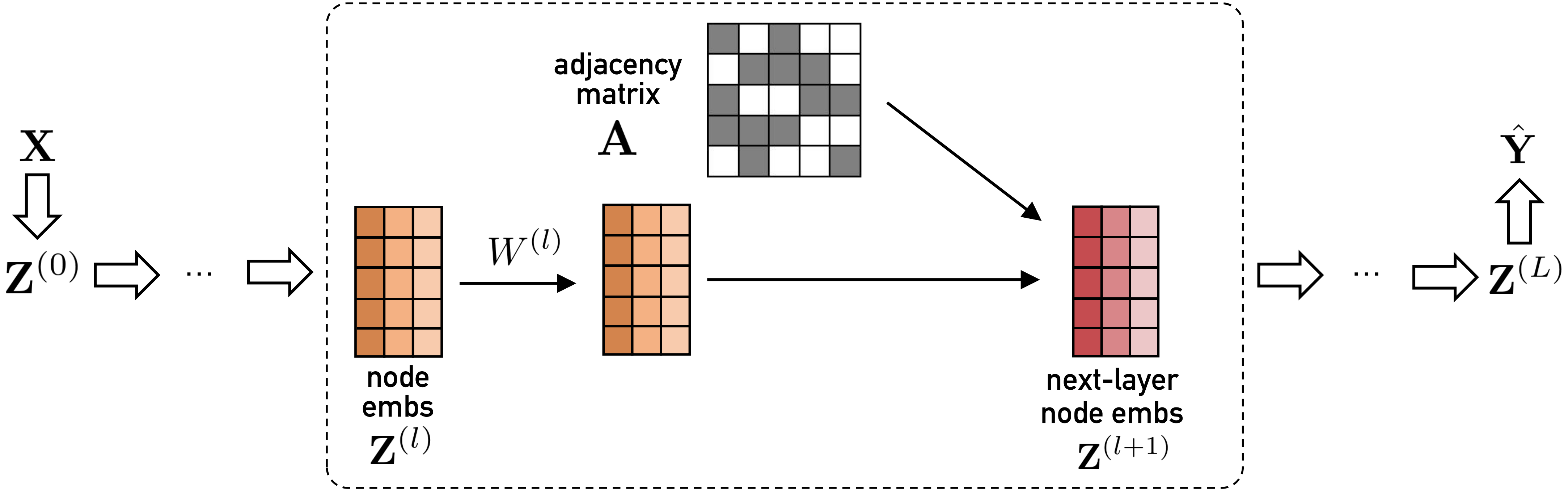
$\mathbf{x}_u$	node (input) feature
y_u	node ground-truth label
$\hat{y}_u$	node predicted label
$\mathbf{z}_u^{(l)}$	node embedding at the l-th layer

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

Preliminary: Graph Neural Networks



update for each node (node view)

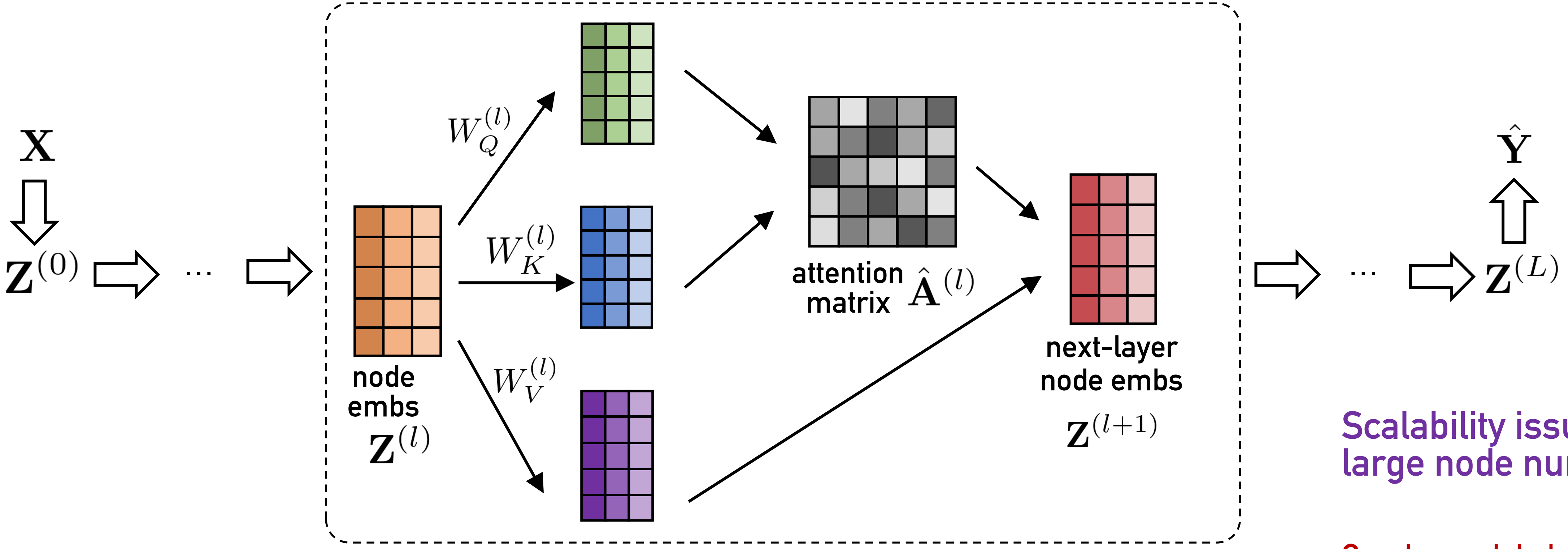
$$\mathbf{z}_u^{(l+1)} = \sum_{u=1}^{N} a_{uv} W^{(l)} \mathbf{z}_u^{(l)} = \sum_{u,(u,v)\in\mathcal{E}} W^{(l)} \mathbf{z}_u^{(l)}$$

update for all nodes (matrix view)

$$\mathbf{Z}^{(l+1)} = \mathbf{A}\mathbf{Z}^{(l)} W^{(l)}$$

At each layer, updating message for each centered node is only dependent on the neighbored nodes within the receptive field

Preliminary: Transformers



Scalability issue for large node numbers

update for each node (node view)

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

update for all nodes (matrix view)

$$\hat{\mathbf{A}}^{(l)} = \mathrm{Softmax}((W_Q^{(l)}\mathbf{Z}^{(l)})^\top (W_K^{(l)}\mathbf{Z}^{(l)})), \quad \mathbf{Z}^{(l+1)} = \hat{\mathbf{A}}^{(l)} W_V^{(l)} \mathbf{Z}^{(l)}$$

One-layer global attention over 10K nodes lead to out-of-memory on a single GPU with 16GB memory

Scalable All-Pair Message Passing with O(N)

Kernelized softmax message passing

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

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

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

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

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

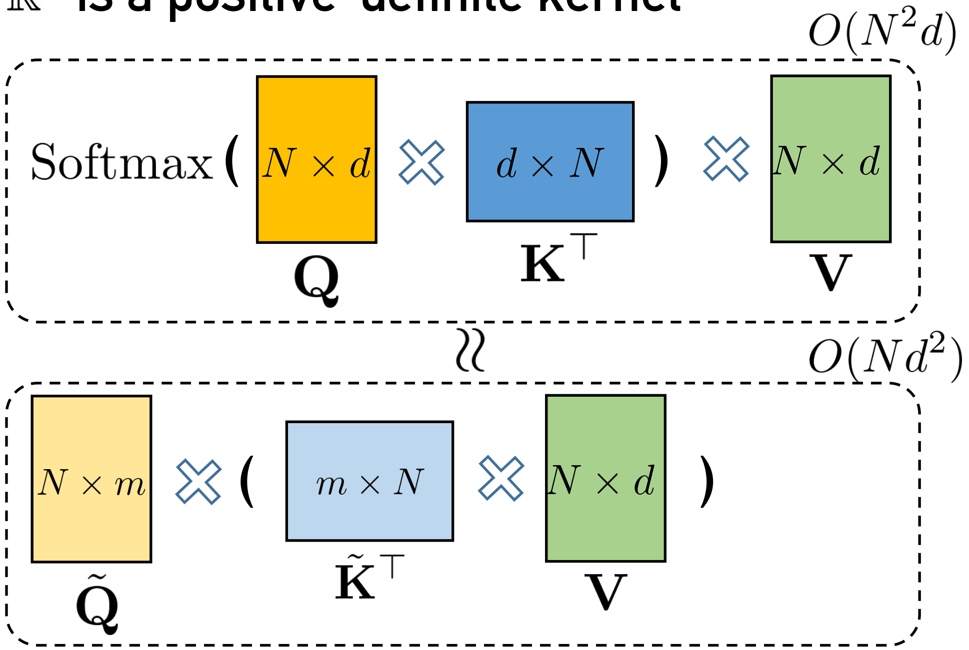
$\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^m$ is a random feature map

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

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

Scalable All-Pair Message Passing with O(N)

❑ Kernelized Gumbel-Softmax

observation: attending on N nodes may lead to over-normalizing (the denominator shrink the attention to zero)
solution: select dominant edges with stochastic sampling

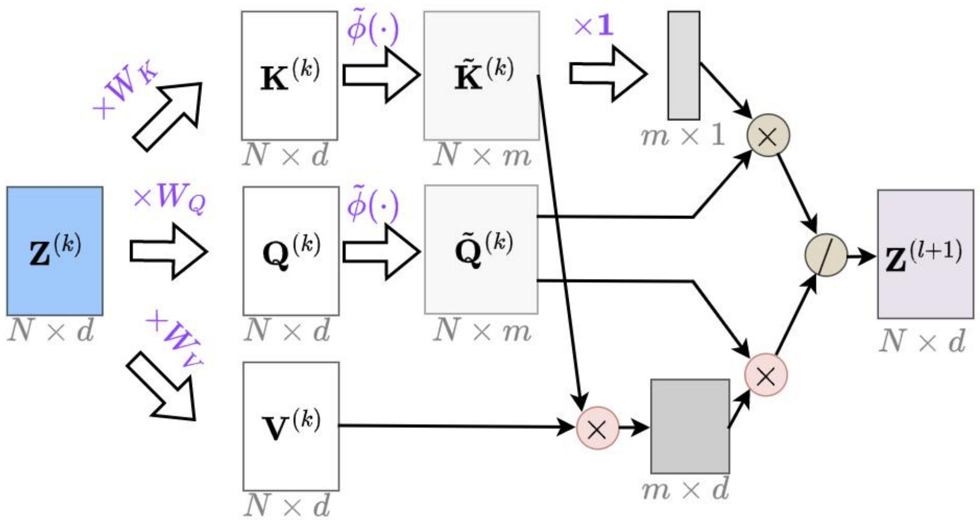
$$\mathbf{z}_u^{(l+1)} = \sum_{v=1}^{N} \frac{\boxed{\exp((\mathbf{q}_u^\top \mathbf{k}_u + g_v)/\tau)}}{\sum_{w=1}^{N} \exp((\mathbf{q}_u^\top \mathbf{k}_w + g_w)/\tau)} \cdot \mathbf{v}_u$$

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

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

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

more details: Appendix A



$$\mathbf{Q}^{(k)} = \mathbf{W}_Q \mathbf{Z}^{(k)}, \quad \mathbf{K}^{(k)} = \mathbf{W}_K \mathbf{Z}^{(k)}, \quad \mathbf{V}^{(k)} = \mathbf{W}_V \mathbf{Z}^{(k)}$$

$$\tilde{\mathbf{Q}}^{(k)} = \phi(\mathbf{Q}^{(k)}), \quad \tilde{\mathbf{K}}^{(k)} = \phi(\mathbf{K}^{(k)})$$

$$\mathbf{D}^{(k)} = \mathrm{diag}^{-1}\left(\tilde{\mathbf{Q}}^{(k)}\left((\tilde{\mathbf{K}}^{(k)})^\top \mathbf{1}\right)\right)$$

$$\mathbf{Z}^{(k+1)} = \mathbf{D}^{(k)}\left[\tilde{\mathbf{Q}}^{(k)}\left((\tilde{\mathbf{K}}^{(k)})^\top \mathbf{V}^{(k)}\right)\right]$$

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

Scalable All-Pair Message Passing with O(N)

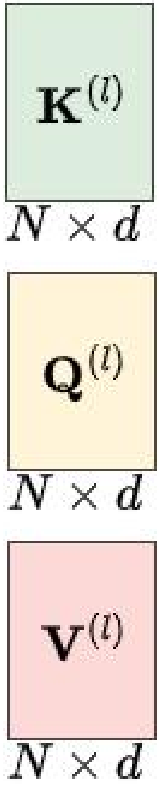
Algorithm 1: Scalable All-Pair Message Passing on Latent Graphs with Linear Complexity ($\mathcal{O}(N)$ or $\mathcal{O}(N+E)$)

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

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

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

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



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

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**

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

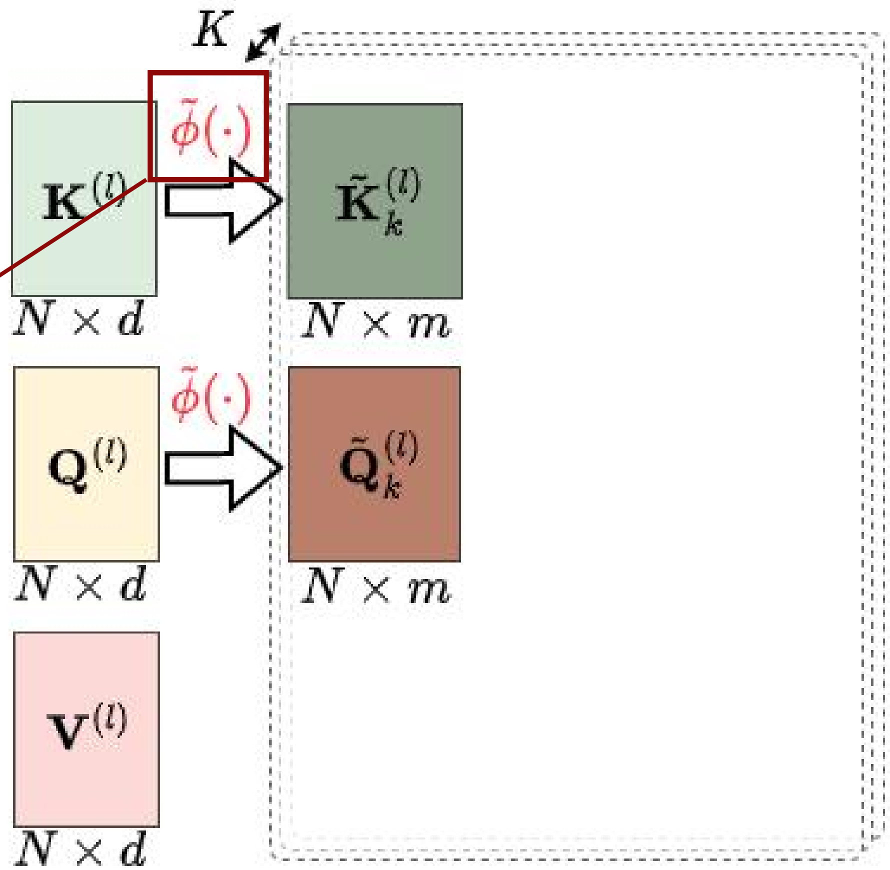
  for $k = 1, 2, \ldots, K$ **do**

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

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

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

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



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

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**

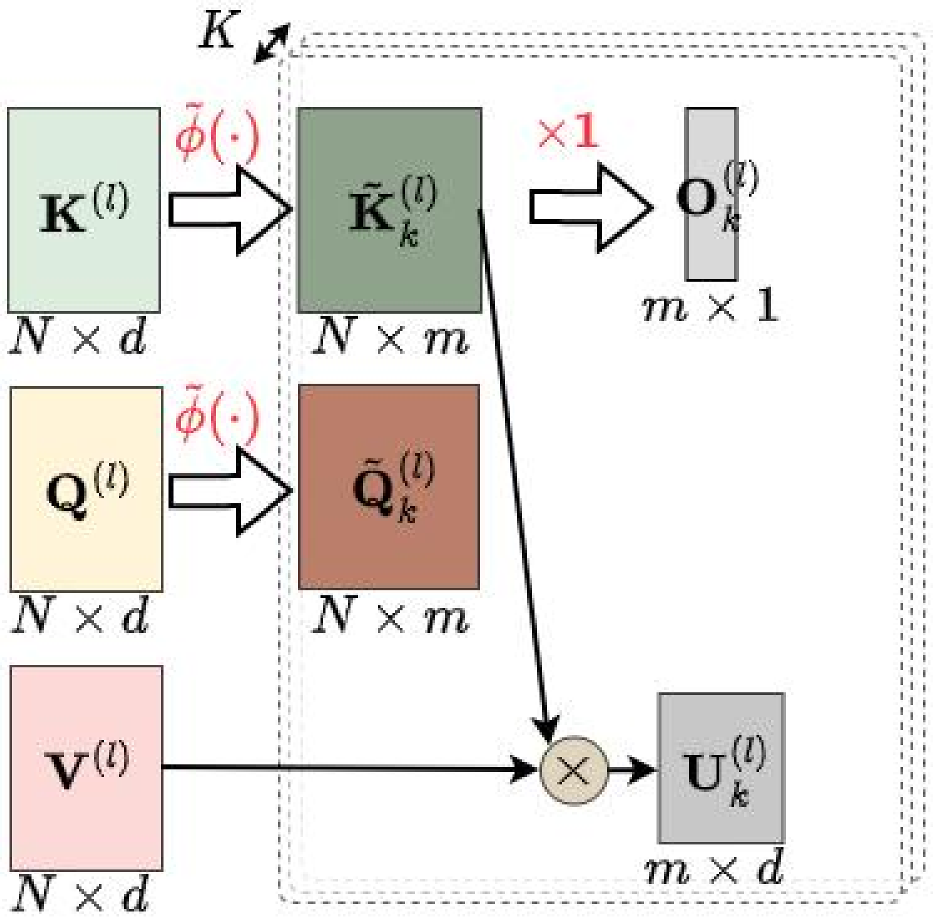
    $G_k = \{e^{g_{ku}/\tau}\}_{u=1}^{N}, \ \ g_{ku} \sim Gumbel(0,1)$;

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

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

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

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



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

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**

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

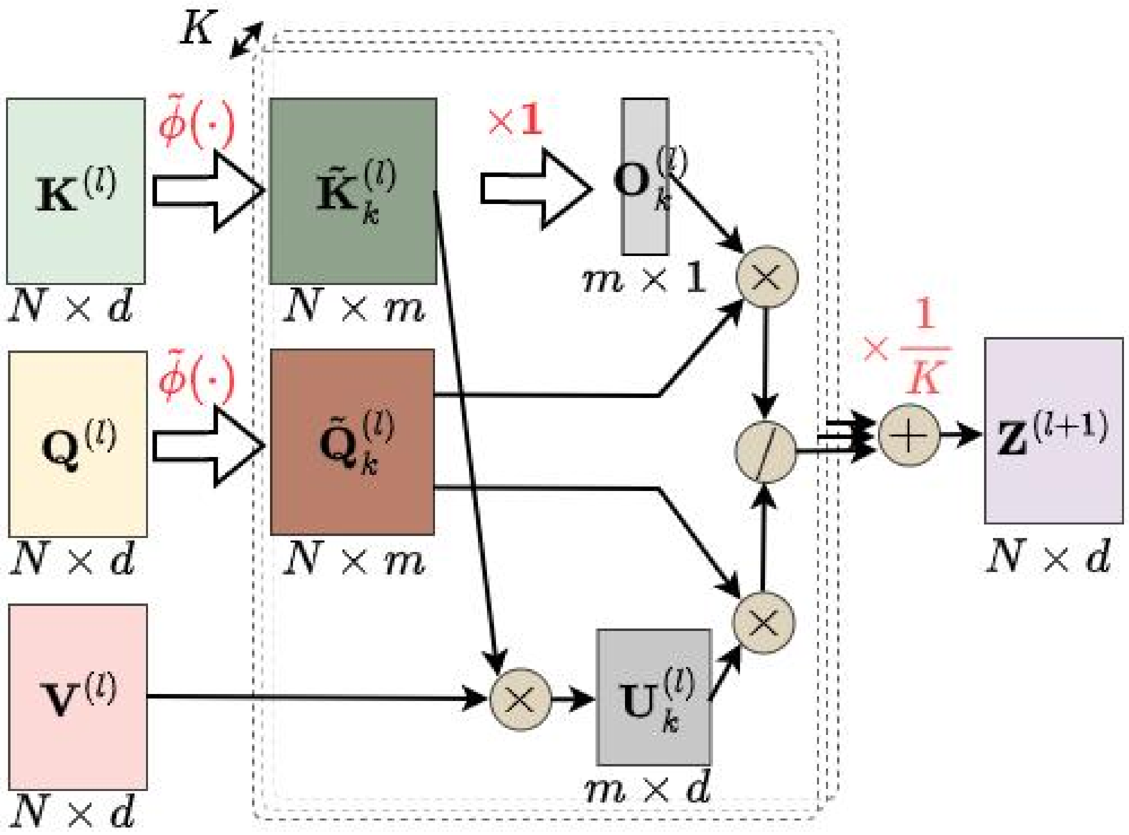
    $\tilde{G}_k = G_k.unsqueeze(1).repeat(1, m)$;

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

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

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

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



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

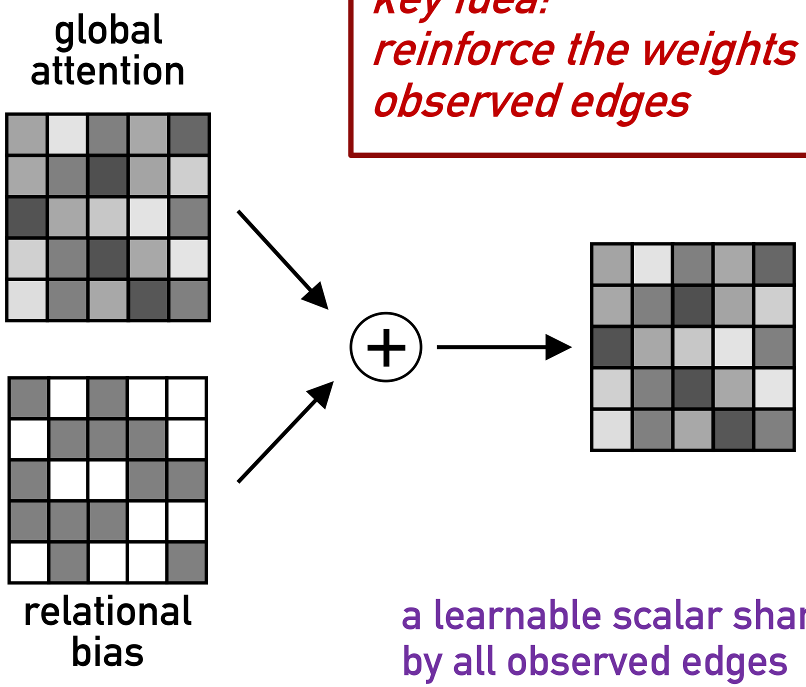
Pytorch Implementation

```
# qs: [N, H, D], ks: [L, H, D], vs: [L, H, D]

qs = softmax_kernel(qs) # [N, H, M]
ks = softmax_kernel(ks) # [L, H, M]

# numerator
kvs = torch.einsum("lhm,lhd->hmd", ks, vs)
attn_num = torch.einsum("nhm,hmd->nhd", qs, kvs) # [N, H, D]

# denominator
all_ones = torch.ones([ks.shape[0]])
ks_sum = torch.einsum("lhm,l->hm", ks, all_ones)
attn_den = torch.einsum("nhm,hm->nh", qs, ks_sum)  # [N, H]

# attentive aggregated results
z_next = attn_num / attn_den # [N, H, D]
```

github repo

tutorial

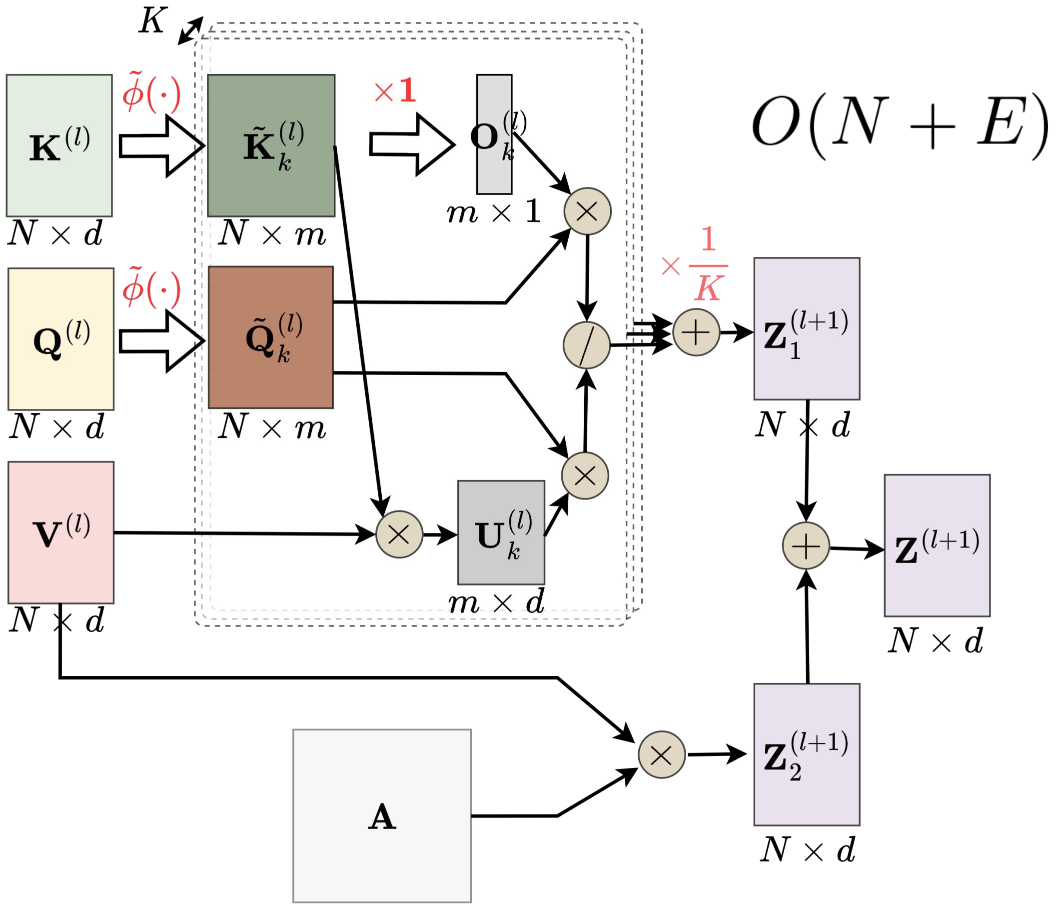
Input Graphs as Relational Bias



key idea:
reinforce the weights for observed edges

a learnable scalar shared by all observed edges

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



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

Input Graphs as Regularization Loss

❑ Supervised classification loss

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

Key observation:

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

❑ Edge-level regularization loss

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

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

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 $\mathcal{O}(1)$

❑ Final loss function

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

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

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 $q(\tilde{\mathbf{A}}|\mathbf{X}, \mathbf{A})$

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

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

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

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

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.

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

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

For random feature dimension m and temperature τ, 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

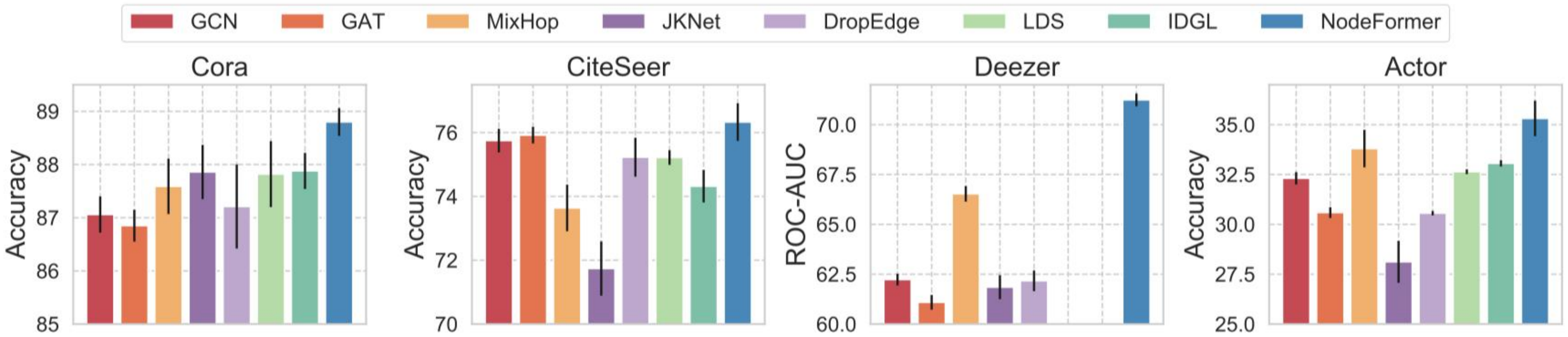
$$\lim_{\tau \to 0} \mathbb{P}(c_{uv} > c_{uv'}, \forall v' \neq v) = \frac{\exp(\mathbf{q}_u^\top \mathbf{k}_v)}{\sum_{w=1}^N \exp(\mathbf{q}_u^\top \mathbf{k}_w)}, \quad \lim_{\tau \to 0} \mathbb{P}(c_{uv} = 1) = \frac{\exp(\mathbf{q}_u^\top \mathbf{k}_v)}{\sum_{w=1}^N \exp(\mathbf{q}_u^\top \mathbf{k}_w)}$$

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

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

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

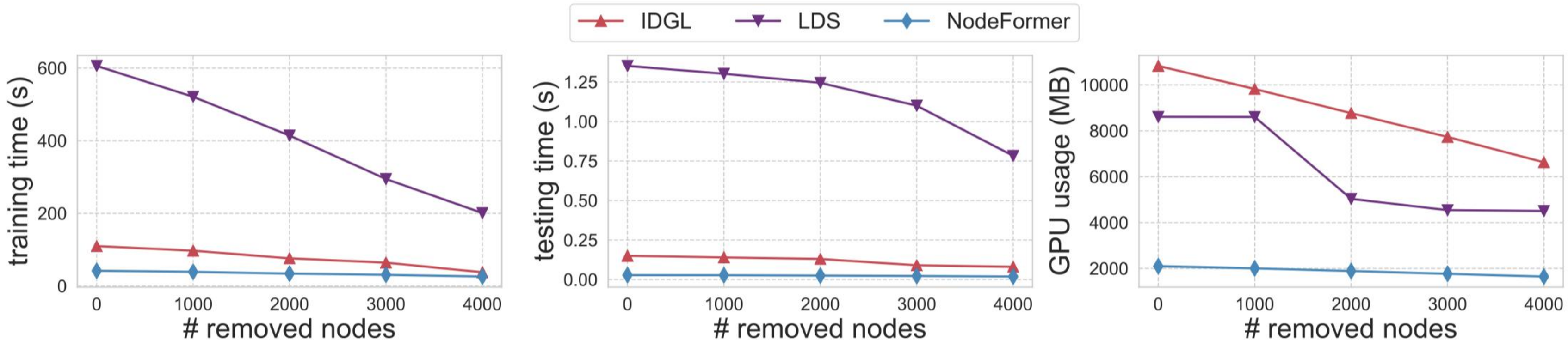
Comparative Experiments

❑ 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			

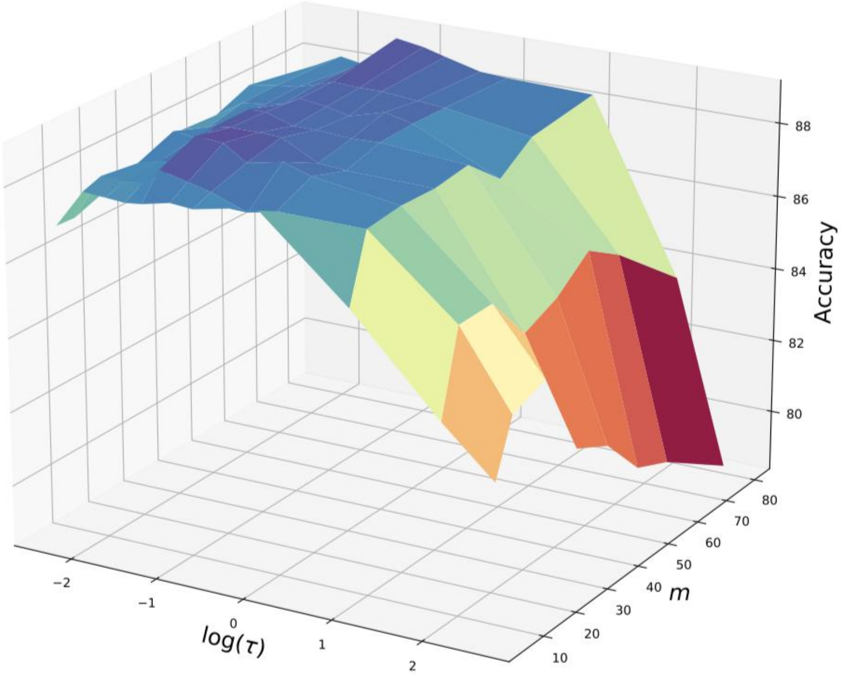
NodeFormer also works with no input graph

❑ Scalability analysis on time/space costs



NodeFormer reduces training time by 93.1%

Ablation Study and Hyper-parameters



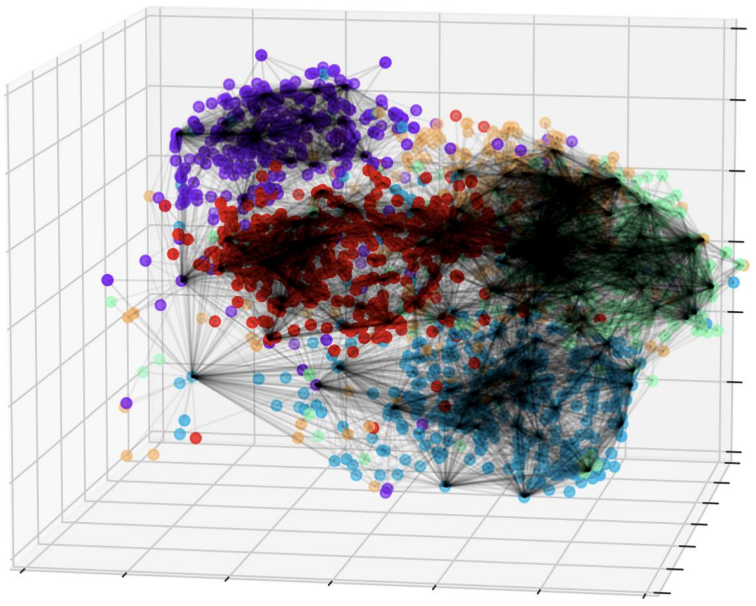
Larger random feature dimension m allows better approximation

Moderate temperature (tau=0.25) yields stably good performance

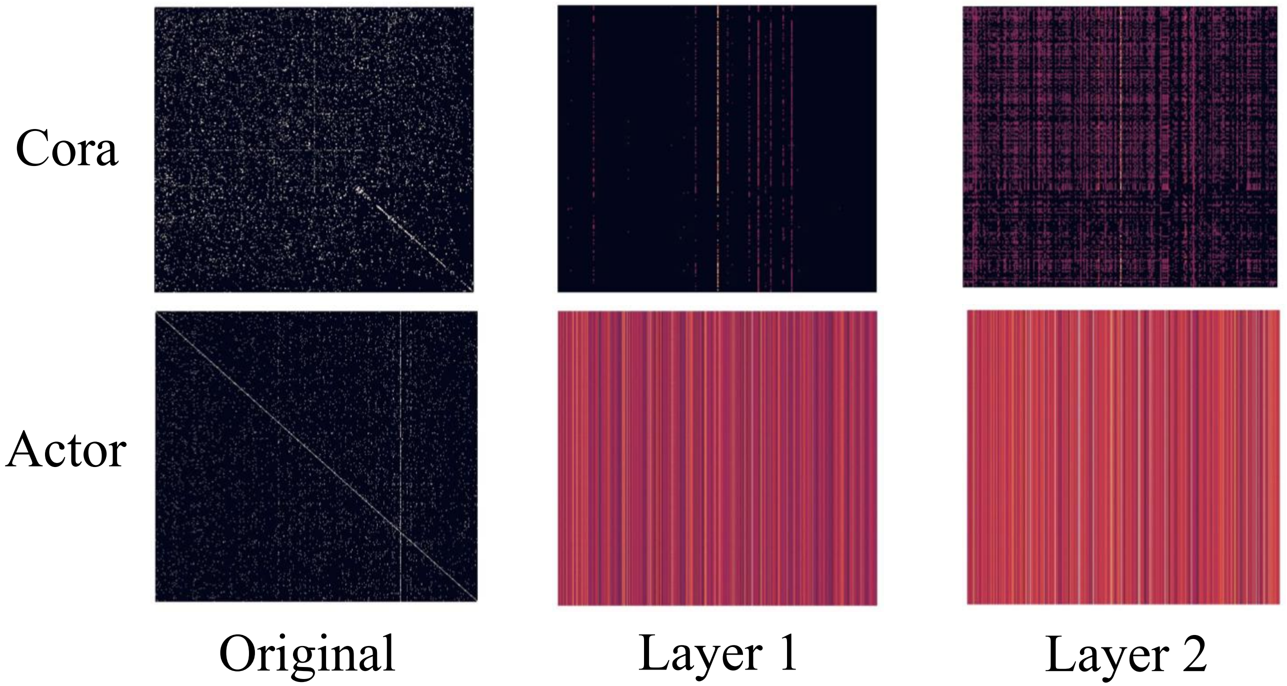
Dataset	NodeFormer	NodeFormer w/o reg	NodeFormer w/o rb
Cora	**88.69** $\pm$ 0.46	81.98 $\pm$ 0.46	88.06 $\pm$ 0.59
Citeseer	**76.33** $\pm$ 0.59	70.60 $\pm$ 1.20	74.12 $\pm$ 0.64
Deezer	**71.24** $\pm$ 0.32	71.22 $\pm$ 0.32	71.10 $\pm$ 0.36
Actor	**35.31** $\pm$ 1.29	35.15 $\pm$ 1.32	34.60 $\pm$ 1.32

Ablation study on edge regularization loss and relational bias

Visualization of Learned Structures



20News-Group



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

Where Are We?

Prior Art quadratic complexity (hard to scale to 10K nodes)

NodeFormer linear complexity (largest demonstration on 2M nodes)

Follow-up open questions:

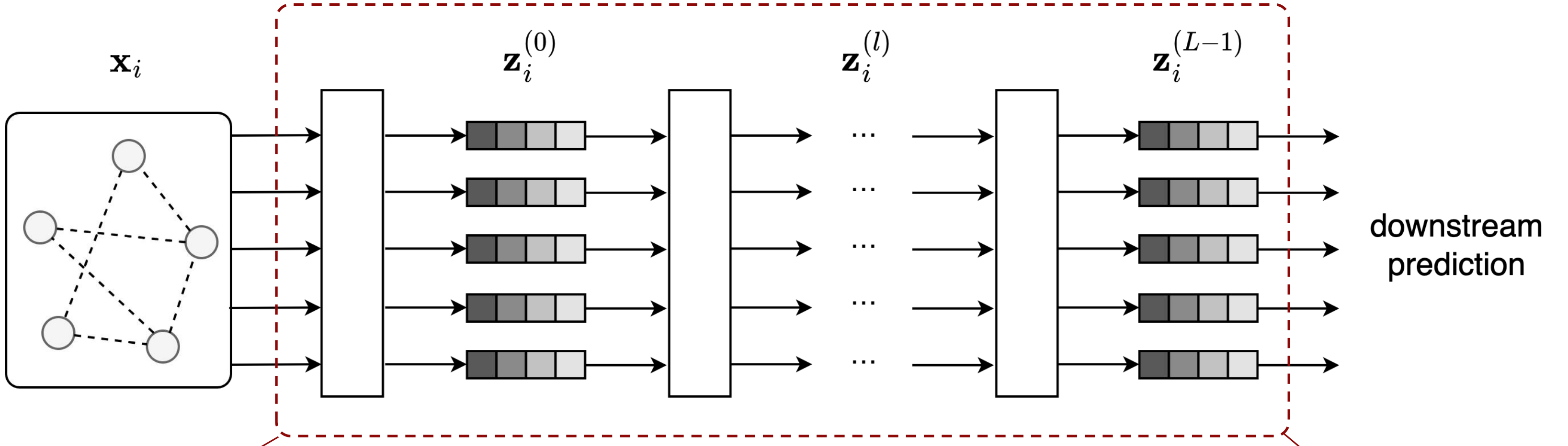
- issue 1: current Transformers mostly stem from heuristic designs

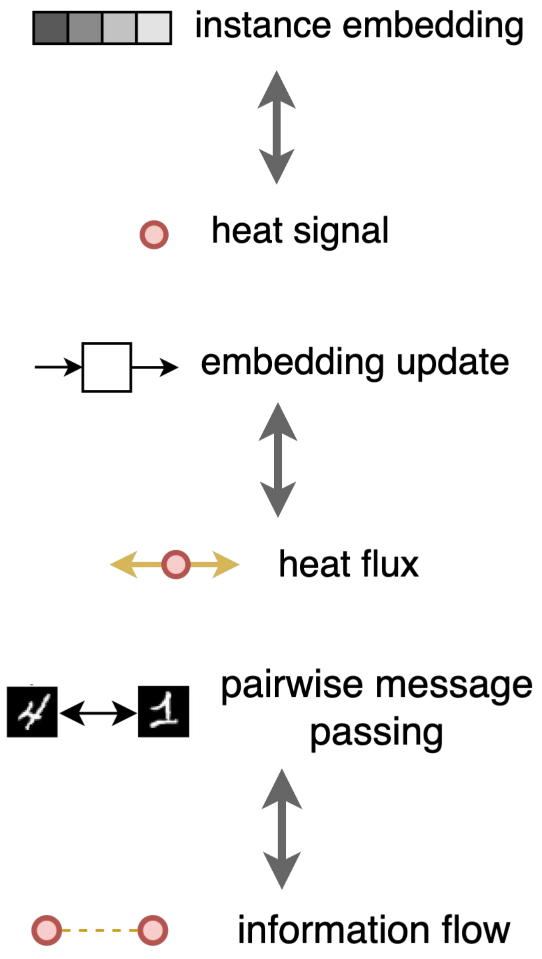
Is there any principled guidance for the design of Transformer attentions?

- issue 2: current Transformers are data-hungry (sufficient supervision)

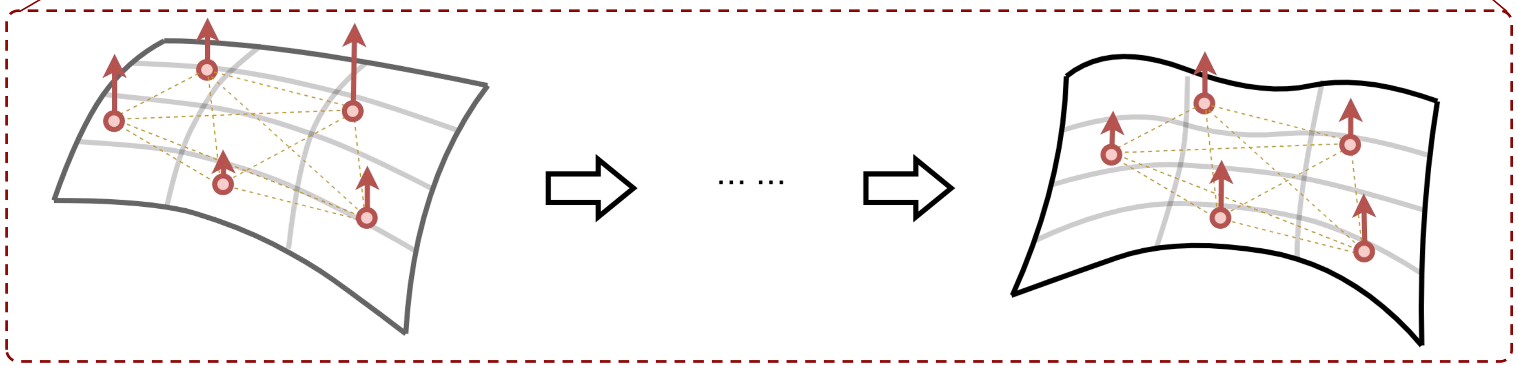
Can graph Transformers handle learning tasks with low labeled rate?

GNN Feed-forward as Diffusion Process





Treat the feed-forward update of embeddings as a diffusion process of heat on locations



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

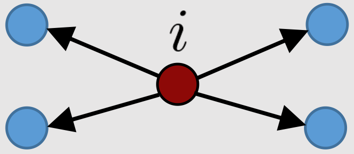
General Formulation of Diffusion Process

The diffusion process of N particles driven by initial states and pairwise interactions:

$$\frac{\partial \mathbf{Z}(t)}{\partial t} = \nabla^* \left(\mathbf{S}(\mathbf{Z}(t), t) \odot \nabla \mathbf{Z}(t) \right), \quad \text{s. t.} \quad \mathbf{Z}(0) = [\mathbf{x}_i]_{i=1}^N, \quad t \geq 0$$

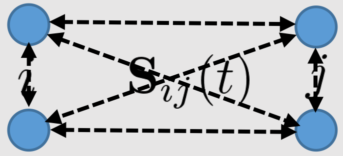
gradient

$$(\nabla \mathbf{Z}(t))_{ij} = \mathbf{z}_j(t) - \mathbf{z}_i(t)$$



divergence

$$(\nabla^*)_i = \sum_{j=1}^N \mathbf{S}_{ij}(\mathbf{Z}(t), t) \, (\nabla \mathbf{Z}(t))_{ij}$$



diffusivity function

$$\mathbf{S}(\mathbf{Z}(t), t) : \mathbb{R}^{N \times d} \times [0, \infty) \to [0, 1]^{N \times N}$$

Diffusion over discrete space composed of N instances with latent structures:

$$\frac{\partial \mathbf{z}_i(t)}{\partial t} = \sum_{j=1}^N \mathbf{S}_{ij}(\mathbf{Z}(t), t)(\mathbf{z}_j(t) - \mathbf{z}_i(t))$$

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

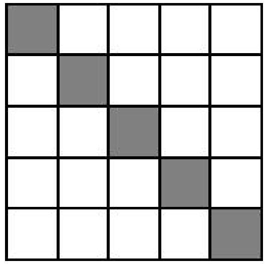
Diffusion with Latent Structures

The iterative dynamics (by explicit scheme) of diffusion induce feed-forward layers:

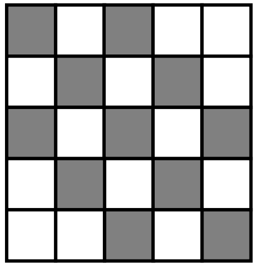
$$\mathbf{z}_i^{(k+1)} = \left(1 - \tau \sum_{j=1}^{N} \mathbf{S}_{ij}^{(k)}\right) \mathbf{z}_i^{(k)} + \tau \sum_{j=1}^{N} \mathbf{S}_{ij}^{(k)} \mathbf{z}_j^{(k)}$$

The $N \times N$ diffusivity $\mathrm{S}^{(k)}$ is a measure of the rate at which the node signals spread

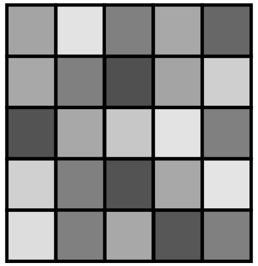
- $\mathrm{S}^{(k)}$ **is an identity matrix: message passing only through self-loops**
- $\mathrm{S}^{(k)}$ **only has non-zero values for observed edges: message passing over a graph**
- $\mathrm{S}^{(k)}$ **can have non-zero values for all entries: all-pair message passing**



MLP



GNN



Transformer

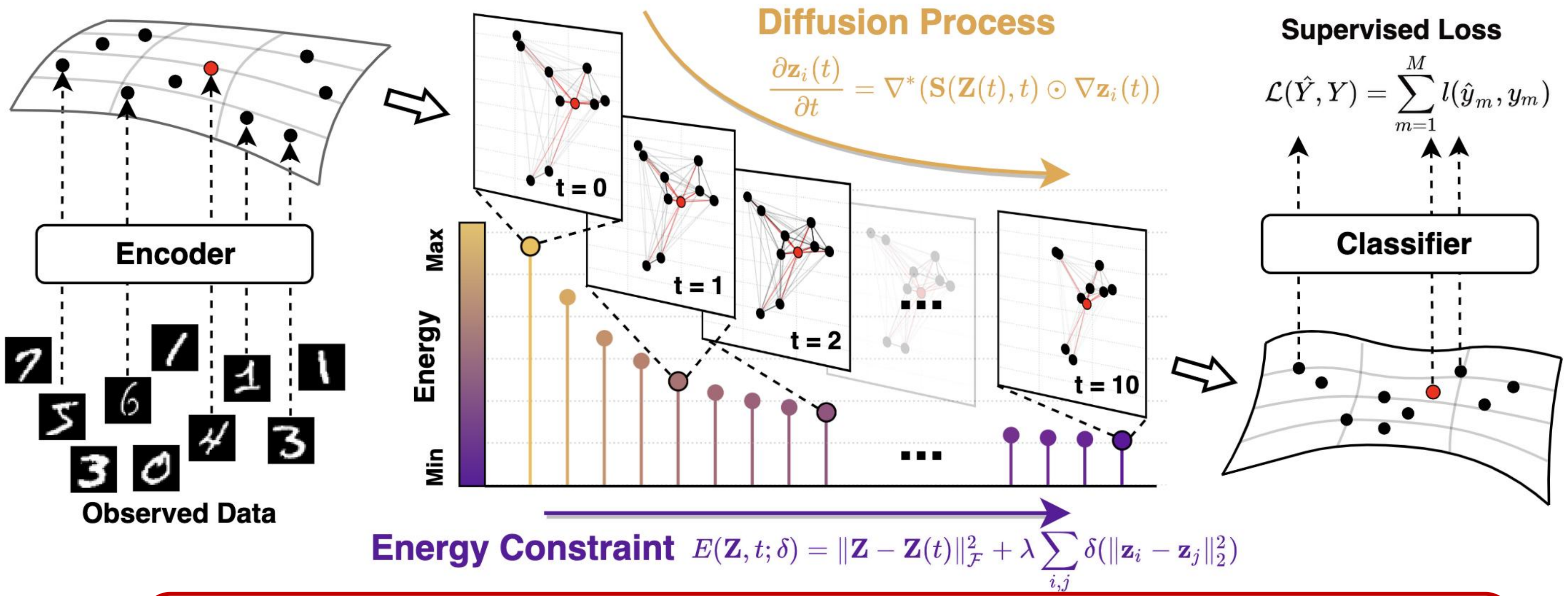
Key question: How to determine a proper diffusivity function for learning desirable node representations?

Energy-Constrained Diffusion Process

Principle 1: particle states evolution described by a diffusion process

+

Principle 2: the evolutionary directions towards descending the global energy



Key insight: treat diffusivity as latent variables whose optimality is given by descent criteria w.r.t. a principled global energy

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

$$\text{s. t. } \mathbf{z}_i^{(0)} = \mathbf{x}_i, \quad E(\mathbf{Z}^{(k+1)}, k; \delta) \leq E(\mathbf{Z}^{(k)}, k-1; \delta), \quad k \geq 1.$$

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

Closed-Form Solutions for Diffusion Dynamics

Theorem (Optimal Diffusivity Estimates for Energy-Constrained Diffusion)

For any regularized energy over $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^N$ defined by the form

$$E(\mathbf{Z}, k; \delta) = \|\mathbf{Z} - \mathbf{Z}^{(k)}\|_{\mathcal{F}}^2 + \lambda \sum_{i,j} \delta(\|\mathbf{z}_i - \mathbf{z}_j\|_2^2)$$

where $\delta : \mathbb{R}^+ \rightarrow \mathbb{R}$ is a concave, non-decreasing function, the diffusion process with diffusivity

$$\hat{\mathbf{S}}_{ij}^{(k)} = \frac{\omega_{ij}^{(k)}}{\sum_{l=1}^N \omega_{il}^{(k)}}, \quad \omega_{ij}^{(k)} = \left.\frac{\partial \delta(z^2)}{\partial z^2}\right|_{z^2 = \|\mathbf{z}_i^{(k)} - \mathbf{z}_j^{(k)}\|_2^2}$$

yields a descent step on the energy, i.e., $E(\mathbf{Z}^{(k+1)}, k; \delta) \leq E(\mathbf{Z}^{(k)}, k-1; \delta)$

One-layer update of DIFFormer

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

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

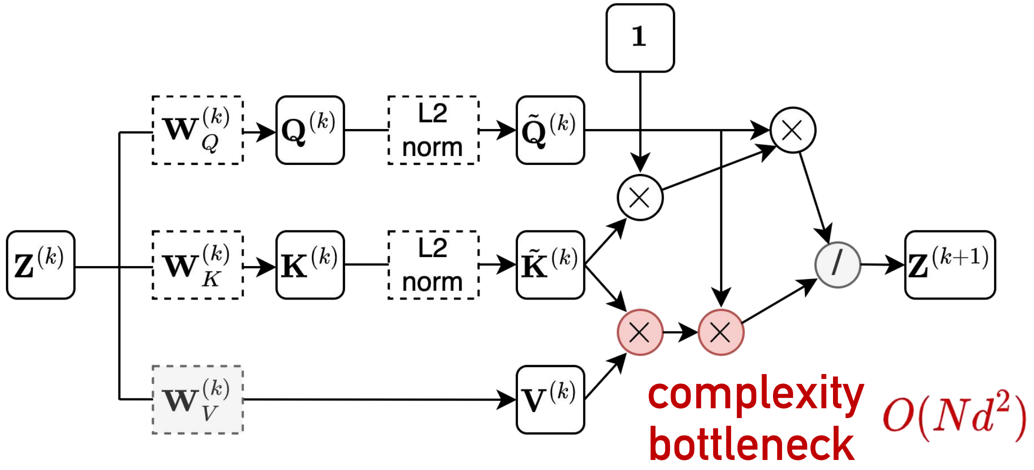
Qitian Wu et al., DIFFormer: Scalable (Graph) Transformers Induced by Energy Constrained Diffusion, ICLR 2023

DIFFormer: Instantiations of Diffusivity

DIFFormer layer with simple diffusivity (DIFFormer-s):

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

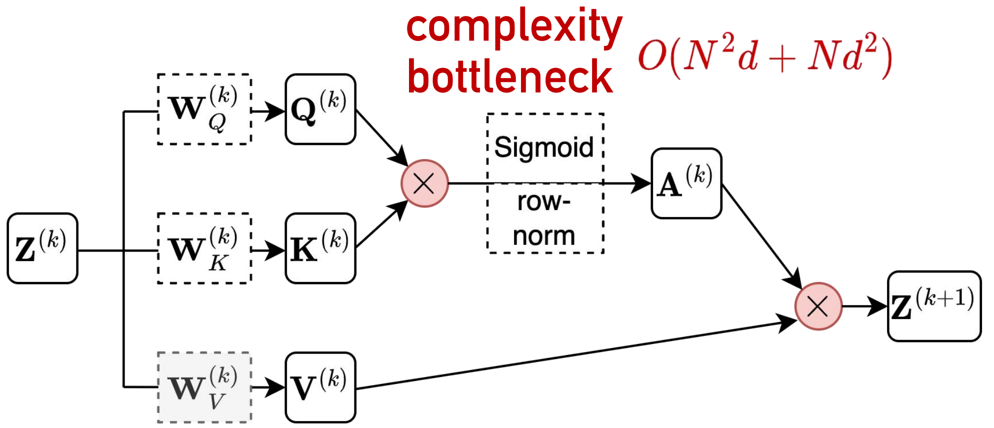
$$\sum_{j=1}^{N} \mathbf{S}_{ij}^{(k)} \mathbf{z}_j^{(k)} = \sum_{j=1}^{N} \frac{1 + (\tilde{\mathbf{z}}_i^{(k)})^\top \tilde{\mathbf{z}}_j^{(k)}}{\sum_{l=1}^{N}\left(1 + (\tilde{\mathbf{z}}_i^{(k)})^\top \tilde{\mathbf{z}}_l^{(k)}\right)} \mathbf{z}_j^{(k)}$$



DIFFormer layer with advanced diffusivity (DIFFormer-a):

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

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



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

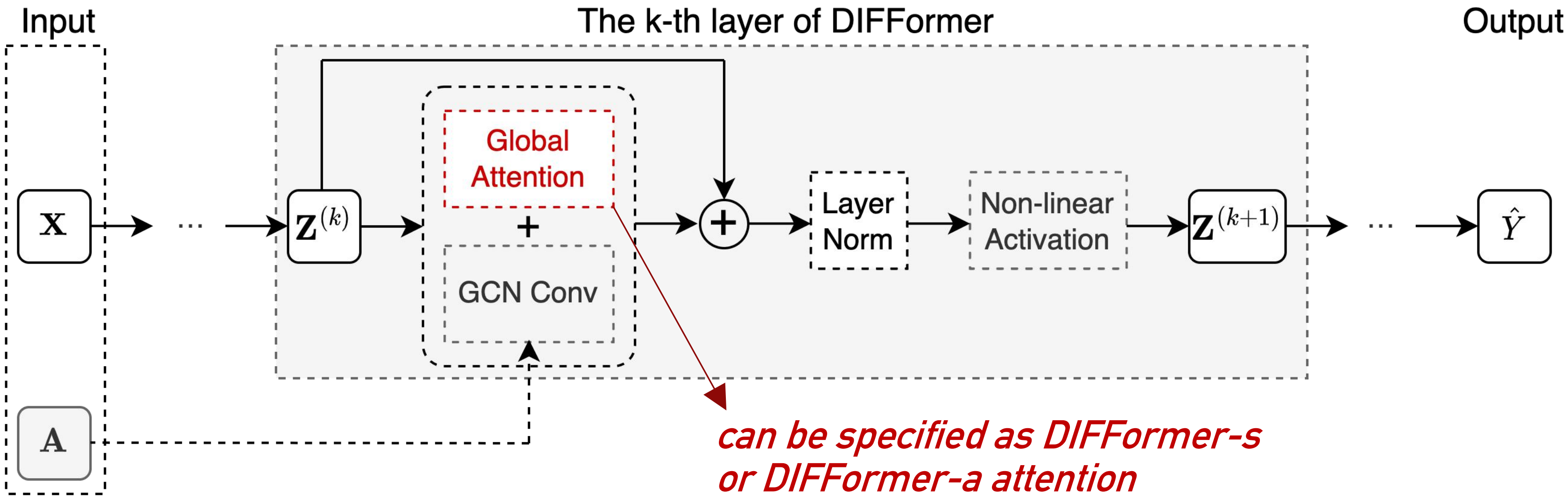
DIFFormer: Extension to a Transformer Layer

Incorporation of input graphs (if available): add graph convolution with global attention

$$\overline{\mathbf{P}}^{(k)} = \frac{1}{2}\left(\hat{\mathbf{S}}^{(k)} + \tilde{\mathbf{A}}\right)\mathbf{Z}^{(k)}$$

DIFFormer layer for updating embedding of the next layer:

$$\mathbf{Z}^{(k+1)} = \sigma'\left(\mathrm{LayerNorm}\left(\tau\overline{\mathbf{P}}^{(k)} + (1-\tau)\mathbf{Z}^{(k)}\right)\right)$$



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

Pytorch Implementation

```
# qs: [N, H, D], ks: [L, H, D], vs: [L, H, D]

qs = qs / torch.norm(qs, p=2) # [N, H, D]
ks = ks / torch.norm(ks, p=2) # [L, H, D]
N = qs.shape[0]

# numerator
kvs = torch.einsum("lhm,lhd->hmd", ks, vs)
attn_num = torch.einsum("nhm,hmd->nhd", qs, kvs) # [N, H, D]
all_ones = torch.ones([vs.shape[0]])
vs_sum = torch.einsum("l,lhd->hd", all_ones, vs) # [H, D]
attn_num += vs_sum.unsqueeze(0).repeat(vs.shape[0], 1, 1) # [N, H, D]

# denominator
all_ones = torch.ones([ks.shape[0]])
ks_sum = torch.einsum("lhm,l->hm", ks, all_ones)
attn_den = torch.einsum("nhm,hm->nh", qs, ks_sum)  # [N, H]

# attentive aggregated results
attn_den = torch.unsqueeze(attn_den, len(attn_den.shape))  # [N, H, 1]
attn_den += torch.ones_like(attn_den) * N
z_next = attn_num / attn_den # [N, H, D]
```

github repo

tutorial

DIFFormer: Scaling to Large-Scale Datasets

Large-scale datasets with massive amount of data, e.g., N instances (N can be arbitrarily large)

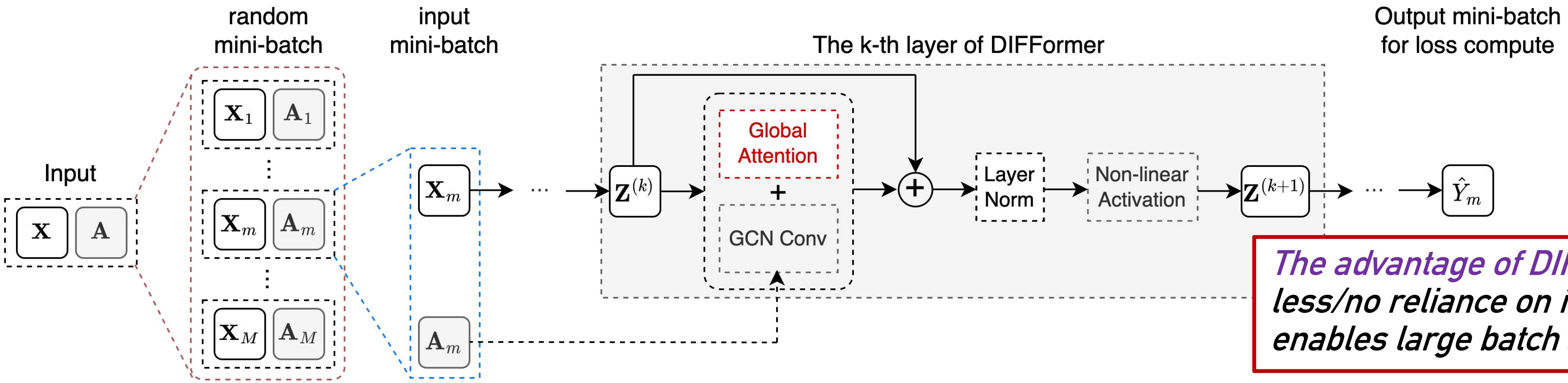
Traditional IID learning enables mini-batch learning with a moderate batch size B << N

How can message passing networks handle large-scale graphs?

Existing solutions: 1. neighbor sampling (slow training and limited receptive field)

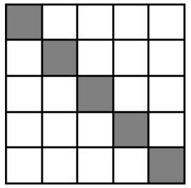
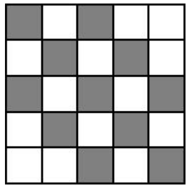
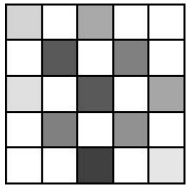
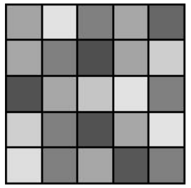
2. graph clustering (time-consuming pre-processing and limited receptive field)

Our solution: partition instances into random mini-batches with a large batch size B



The advantage of DIFFormer: less/no reliance on input graphs enables large batch size

Interpretations of MLP/GNNs as Diffusion

	Energy function	Diffusivity	Illustration
MLP	$\|\mathbf{Z} - \mathbf{Z}^{(k)}\|_2^2$	$\mathbf{S}_{ij}^{(k)} = \begin{cases} 1, & \text{if } i = j \\ 0, & otherwise \end{cases}$	
GCN	$\sum_{(i,j)\in\mathcal{E}} \|\mathbf{z}_i - \mathbf{z}_j\|_2^2$	$\mathbf{S}_{ij}^{(k)} = \begin{cases} \frac{1}{\sqrt{d_i d_j}}, & \text{if } (i,j) \in \mathcal{E} \\ 0, & otherwise \end{cases}$	
GAT	$\sum_{(i,j)\in\mathcal{E}} \delta(\|\mathbf{z}_i - \mathbf{z}_j\|_2^2)$	$\mathbf{S}_{ij}^{(k)} = \begin{cases} \frac{f(\|\mathbf{z}_i^{(k)} - \mathbf{z}_j^{(k)}\|_2^2)}{\sum_{l:(i,l)\in\mathcal{E}} f(\|\mathbf{z}_i^{(k)} - \mathbf{z}_l^{(k)}\|_2^2)}, & \text{if } (i,j) \in \mathcal{E} \\ 0, & otherwise \end{cases}$	
DIFFormer	$\|\mathbf{Z} - \mathbf{Z}^{(k)}\|_2^2 + \lambda \sum_{i,j} \delta(\|\mathbf{z}_i - \mathbf{z}_j\|_2^2)$	$\mathbf{S}_{ij}^{(k)} = \frac{f(\|\mathbf{z}_i^{(k)} - \mathbf{z}_j^{(k)}\|_2^2)}{\sum_{l=1}^{N} f(\|\mathbf{z}_i^{(k)} - \mathbf{z}_l^{(k)}\|_2^2)}, \quad 1 \leq i, j \leq N$	

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

Results on Graph-based Node Classification

Results of testing accuracy on semi-supervised node classification (20 nodes per class for train)

Type	Model	Non-linearity	PDE-solver	Input-G	Cora	Citeseer	Pubmed
Basic models	MLP	R	-	-	56.1 ± 1.6	56.7 ± 1.7	69.8 ± 1.5
	LP	-	-	R	68.2	42.8	65.8
	ManiReg	R	-	R	60.4 ± 0.8	67.2 ± 1.6	71.3 ± 1.4
Standard GNNs	GCN	R	-	R	81.5 ± 1.3	71.9 ± 1.9	77.8 ± 2.9
	GAT	R	-	R	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3
	SGC	-	-	R	81.0 ± 0.0	71.9 ± 0.1	78.9 ± 0.0
	GCN-kNN	R	-	-	72.2 ± 1.8	56.8 ± 3.2	74.5 ± 3.2
	GAT-kNN	R	-	-	73.8 ± 1.7	56.4 ± 3.8	75.4 ± 1.3
	Dense GAT	R	-	-	78.5 ± 2.5	66.4 ± 1.5	66.4 ± 1.5
	LDS	R	-	-	83.9 ± 0.6	**74.8 ± 0.3**	out-of-memory
	GLCN	R	-	-	83.1 ± 0.5	72.5 ± 0.9	78.4 ± 1.5
Diffusion-based models	GRAND-l	-	R	R	83.6 ± 1.0	73.4 ± 0.5	78.8 ± 1.7
	GRAND	R	R	R	83.3 ± 1.3	74.1 ± 1.7	78.1 ± 2.1
	GRAND++	R	R	R	82.2 ± 1.1	73.3 ± 0.9	78.1 ± 0.9
	GDC	R	-	R	83.6 ± 0.2	73.4 ± 0.3	78.7 ± 0.4
	GraphHeat	R	-	R	83.7	72.5	80.5
	DGC-Euler	-	-	R	83.3 ± 0.0	73.3 ± 0.1	80.3 ± 0.1
Graph Transformers	NodeFormer	-	-	-	83.4 ± 0.2	73.0 ± 0.3	**81.5 ± 0.4**
	DIFFORMER-s	-	-	-	**85.9 ± 0.4**	73.5 ± 0.3	**81.8 ± 0.3**
	DIFFORMER-a	-	-	-	**84.1 ± 0.6**	**75.7 ± 0.3**	80.5 ± 1.2

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

Results on Graph-based Node Classification

Proteins: 132,534 nodes, 39,561,252 edges
Pokec: 1,632,803 nodes, 30,622,564 edges

Results of testing accuracy on two large-scale graph datasets

Models	Proteins	Pokec
MLP	72.41 ± 0.10	60.15 ± 0.03
LP	74.73	52.73
SGC	49.03 ± 0.93	52.03 ± 0.84
GCN	74.22 ± 0.49*	62.31 ± 1.13*
GAT	75.11 ± 1.45*	65.57 ± 0.34*
NodeFormer	**77.45 ± 1.15***	**68.32 ± 0.45***
DIFFORMER-s	**79.49 ± 0.44***	**69.24 ± 0.76***

We use batch size 10K/100K for training DIFFormer-s using a single GPU on Proteins/Pokec

Test Acc and memory costs of different batch sizes on Pokec

Batch size	5000	10000	20000	50000	100000	200000
Test Acc (%)	65.24 ± 0.34	67.48 ± 0.81	68.53 ± 0.75	68.96 ± 0.63	69.24 ± 0.76	69.15 ± 0.52
GPU Memory (MB)	1244	1326	1539	2060	2928	4011

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

Results on Image & Text Classification

Results of testing accuracy on semi-supervised image and text classification

Dataset		MLP	LP	ManiReg	GCN-kNN	GAT-kNN	DenseGAT	GLCN	DIFFORMER-s	DIFFORMER-a
CIFAR	100 labels	65.9 ± 1.3	66.2	67.0 ± 1.9	66.7 ± 1.5	66.0 ± 2.1	out-of-memory	66.6 ± 1.4	**69.1 ± 1.1**	**69.3 ± 1.4**
	500 labels	73.2 ± 0.4	70.6	72.6 ± 1.2	72.9 ± 0.4	72.4 ± 0.5	out-of-memory	72.8 ± 0.5	**74.8 ± 0.5**	**74.0 ± 0.6**
	1000 labels	75.4 ± 0.6	71.9	74.3 ± 0.4	74.7 ± 0.5	74.1 ± 0.5	out-of-memory	74.7 ± 0.3	**76.6 ± 0.3**	**75.9 ± 0.3**
STL	100 labels	66.2 ± 1.4	65.2	66.5 ± 1.9	**66.9 ± 0.5**	66.5 ± 0.8	out-of-memory	66.4 ± 0.8	**67.8 ± 1.1**	66.8 ± 1.1
	500 labels	**73.0 ± 0.8**	71.8	72.5 ± 0.5	72.1 ± 0.8	72.0 ± 0.8	out-of-memory	72.4 ± 1.3	**73.7 ± 0.6**	72.9 ± 0.7
	1000 labels	75.0 ± 0.8	72.7	74.2 ± 0.5	73.7 ± 0.4	73.9 ± 0.6	out-of-memory	74.3 ± 0.7	**76.4 ± 0.5**	**75.3 ± 0.6**
20News	1000 labels	54.1 ± 0.9	55.9	56.3 ± 1.2	56.1 ± 0.6	55.2 ± 0.8	54.6 ± 0.2	56.2 ± 0.8	**57.7 ± 0.3**	**57.9 ± 0.7**
	2000 labels	57.8 ± 0.9	57.6	60.0 ± 0.8	60.6 ± 1.3	59.1 ± 2.2	59.3 ± 1.4	60.2 ± 0.7	**61.2 ± 0.6**	**61.3 ± 1.0**
	4000 labels	62.4 ± 0.6	59.5	63.6 ± 0.7	64.3 ± 1.0	62.9 ± 0.7	62.4 ± 1.0	64.1 ± 0.8	**65.9 ± 0.8**	**64.8 ± 1.0**

For image datasets, use a pretrained network to obtain embeddings of images

Use k-nearest-neighbor to construct a graph for baseline methods GCN-kNN and GAT-kNN

DIFFormer-s and DIFFormer-a without using any graph structure outperform the competitors

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

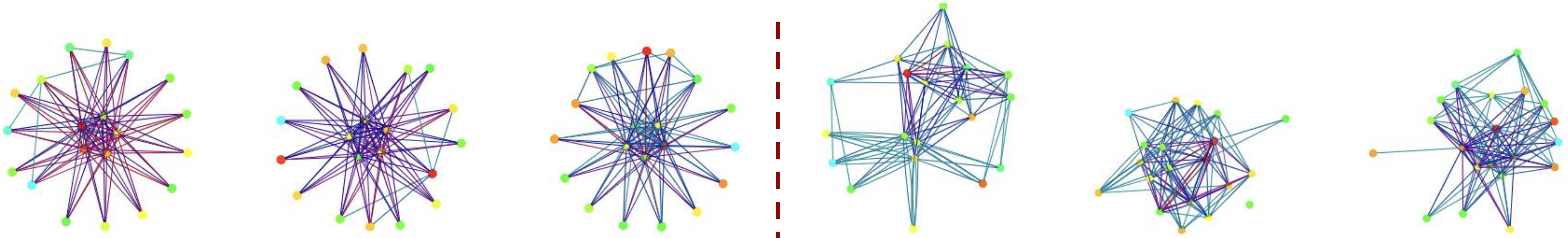
Results on Spatial-Temporal Prediction

Results of testing mean square error for predicting spatial-temporal dynamics based on history

Dataset	MLP	GCN	GAT	Dense GAT	GAT-kNN	GCN-kNN	DIFFormer-s	DIFFormer-a	DIFFormer-s w/o g	DIFFormer-a w/o g
Chickenpox	0.924 (±0.001)	0.923 (±0.001)	0.924 (±0.002)	0.935 (±0.005)	0.926 (±0.004)	0.936 (±0.004)	**0.914 (0.006)**	**0.915 (0.008)**	0.916 (0.006)	0.916 (0.006)
Covid	0.956 (±0.198)	1.080 (±0.162)	1.052 (±0.336)	1.524 (±0.319)	0.861 (±0.123)	1.475 (±0.560)	0.779 (0.037)	**0.757 (0.048)**	0.779 (0.028)	**0.741 (0.052)**
WikiMath	1.073 (±0.042)	1.292 (±0.125)	1.339 (±0.073)	0.826 (±0.070)	0.882 (±0.015)	1.023 (±0.058)	0.731 (0.007)	0.763 (0.020)	**0.727 (0.025)**	**0.716 (0.030)**

Goal: Given the historical graph snapshot, one needs to predict node labels at the next step

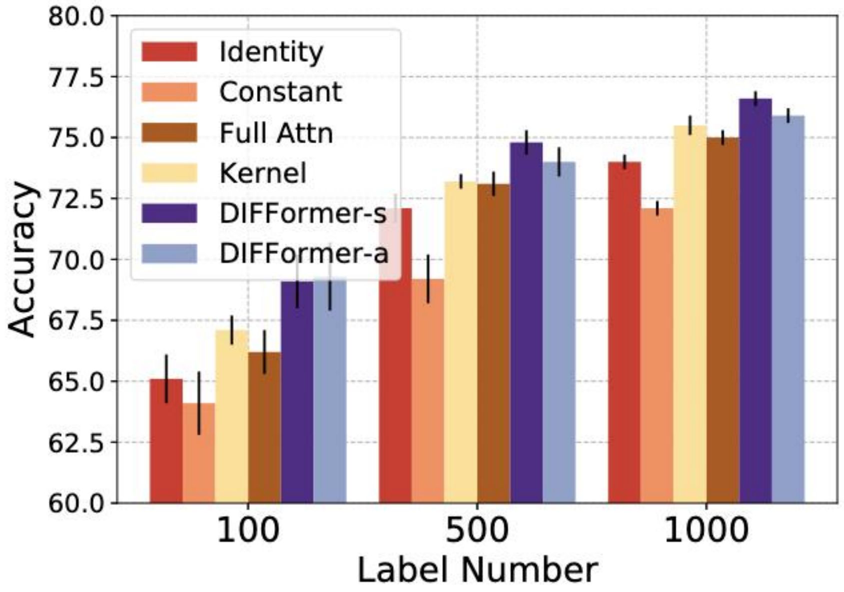
DIFFormer without using graph structure (w/o g) can sometimes yield better prediction



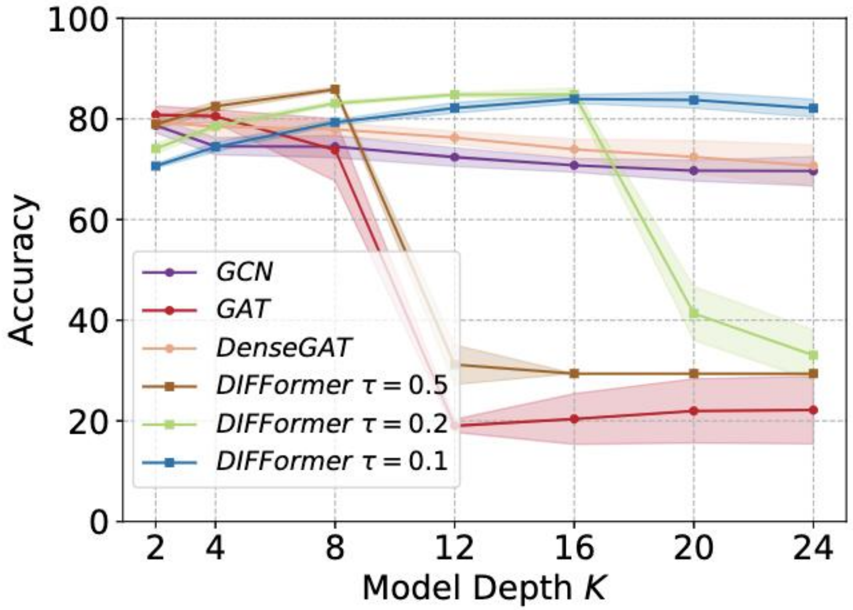
Diffusivity estimates of DIFFormer-s

Diffusivity estimates of DIFFormer-a

Ablation Study and Hyperparameters

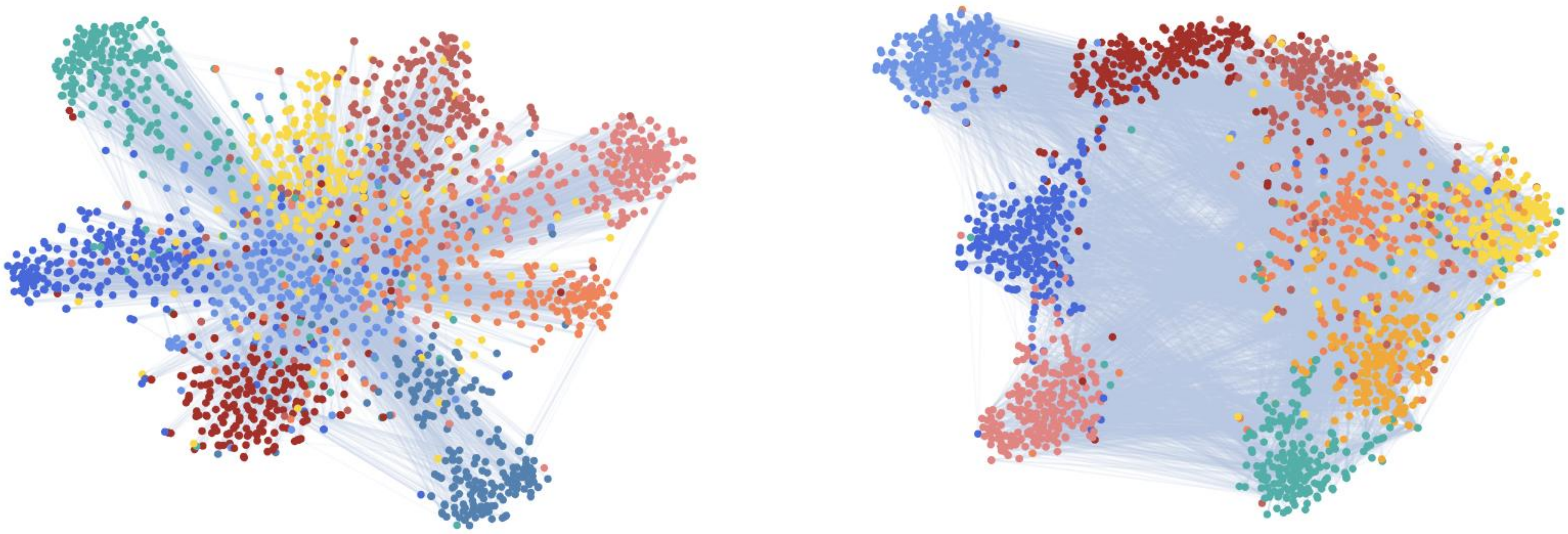


Ablation study on attention functions (i.e., diffusivity parameterization)



Impact of model depth K and step size τ for diffusion iteration

Visualization of Representations



Instance embeddings (colored by different classes) and attention weights (edges with different strengths) on 20News (the left) and STL-10 (the right)

Where Are We?

Prior Art	quadratic complexity (hard to scale to 10K nodes) data-hungry (require abundant labeled information)
NodeFormer	linear complexity (largest demonstration on 2M nodes)
DIFFormer	capable of learning with limited labeled rate

Follow-up open questions:

(issue: the complicated architectures limit the efficiency and scalability)

Can Transformer architectures be simplified and scale to web-scale graphs?

SGFormer: Simplified Graph Transformers

Observation: one-layer all-pair attention can propagate information among arbitrary node pairs

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

- Simple attention with linear complexity:

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

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

- 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., SGFormer: *Simplifying and Empowering Transformers for 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
GraphGPS [Rampáse et al. 2022]	R	R	R	yes	$O(N^2)$	1.0K
ANS-GT [Zhang et al. 2022]	R	R	R	no	$O(Nsm^2)$	20K
NodeFormer [Wu et al. 2022]	R	R	-	yes	$O(N+E)$	2M
SGFormer	-	-	-	yes	$O(N+E)$	0.1B

Experiment Results

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

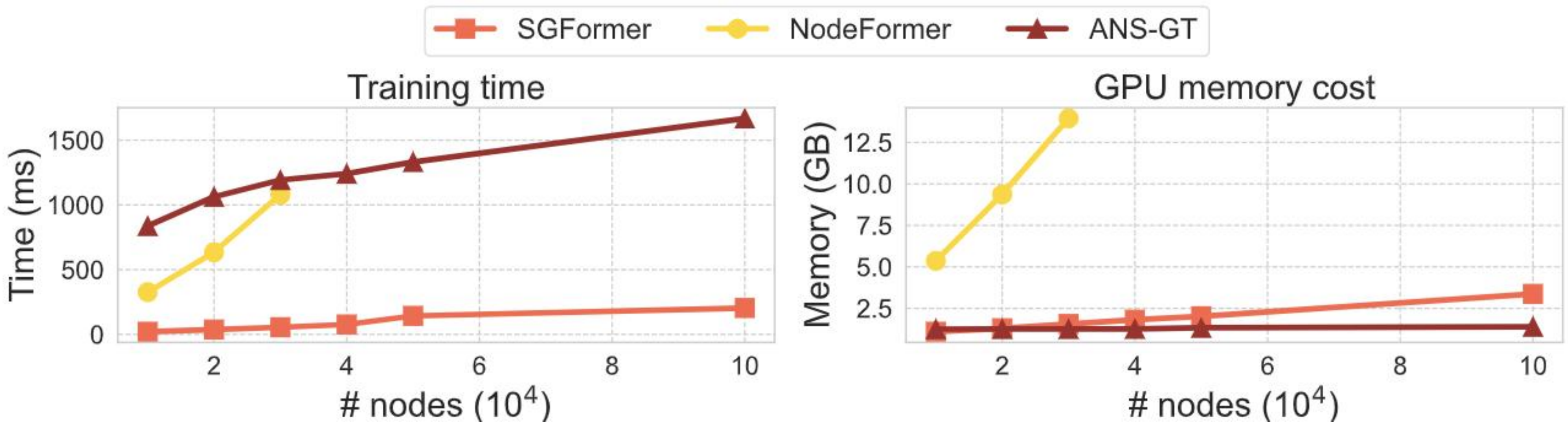
Mini-batch training for proteins, Amazon2M, pokec with batch size 10K/100K

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

Experiment Results

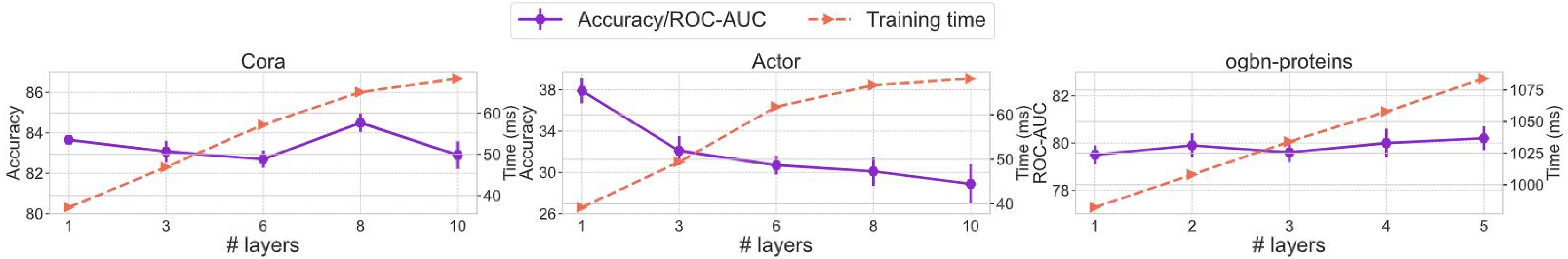
Comparison of training/inference time per epoch and memory cost

Method	Cora			PubMed			Amazon2M		
	Tr (ms)	Inf (ms)	Mem (GB)	Tr (ms)	Inf (ms)	Mem (GB)	Tr (ms)	Inf (ms)	Mem (GB)
Graphormer	215.8	63.6	5.0	-	-	-	-	-	-
GraphTrans	160.4	40.2	3.8	-	-	-	-	-	-
ANS-GT	570.1	539.2	1.0	511.9	461.0	2.1	-	-	-
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 can overcome several limitations of GNNs

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

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

principled global attention designs   superiority for low labeled rates

[3] SGFormer: 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

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