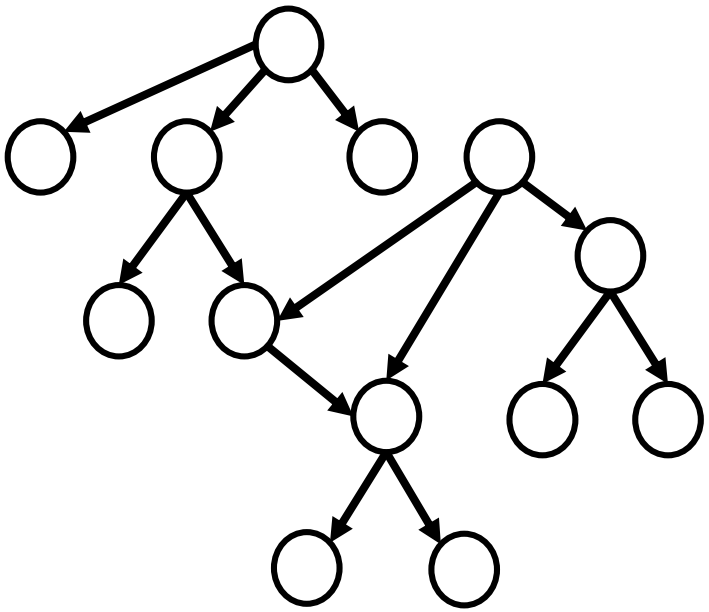
Transformers over Directed Acyclic Graphs

Yuankai Luo, Veronika Thost, Lei Shi

NeurIPS 2023

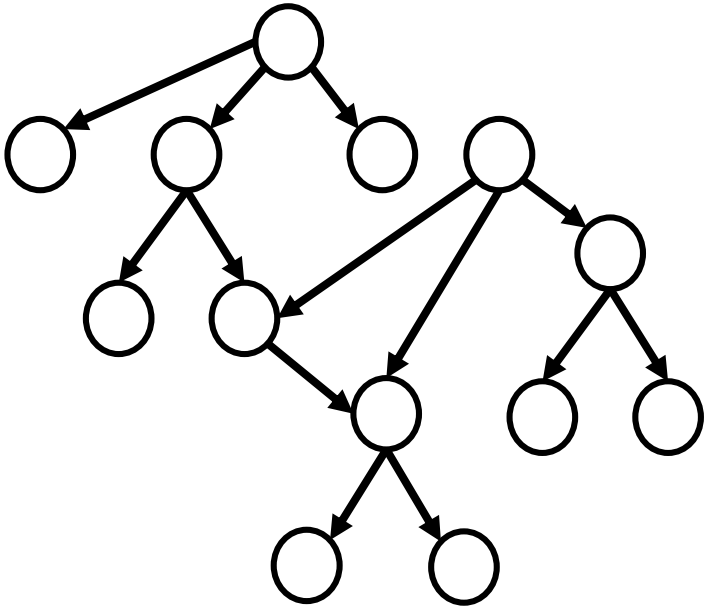
Directed Acyclic Graphs (DAGs)



- Directed acyclic graph (DAG) :

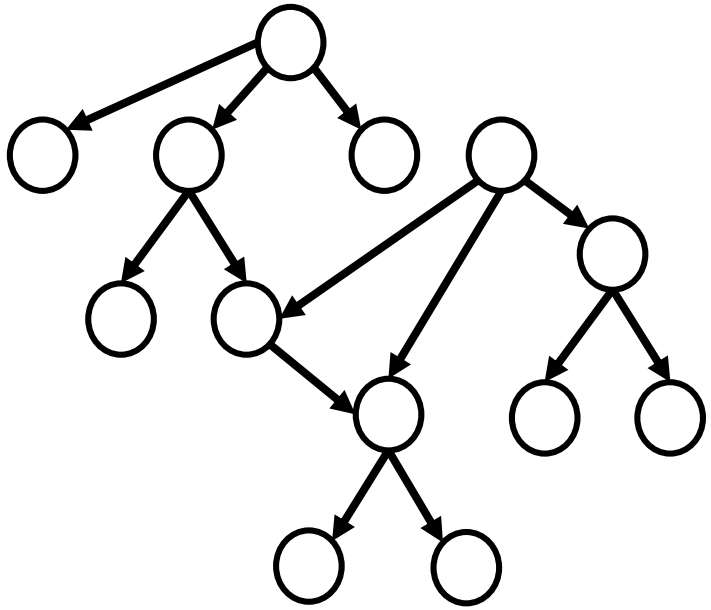
  a directed graph without directed cycles.

Directed Acyclic Graphs (DAGs)



- Can define a unique strong partial order $\preccurlyeq$ on the node set V,
- such that, for all pairs of nodes $u, v \in V$, $u \preccurlyeq v$ if and only if there is a directed path from u to v.

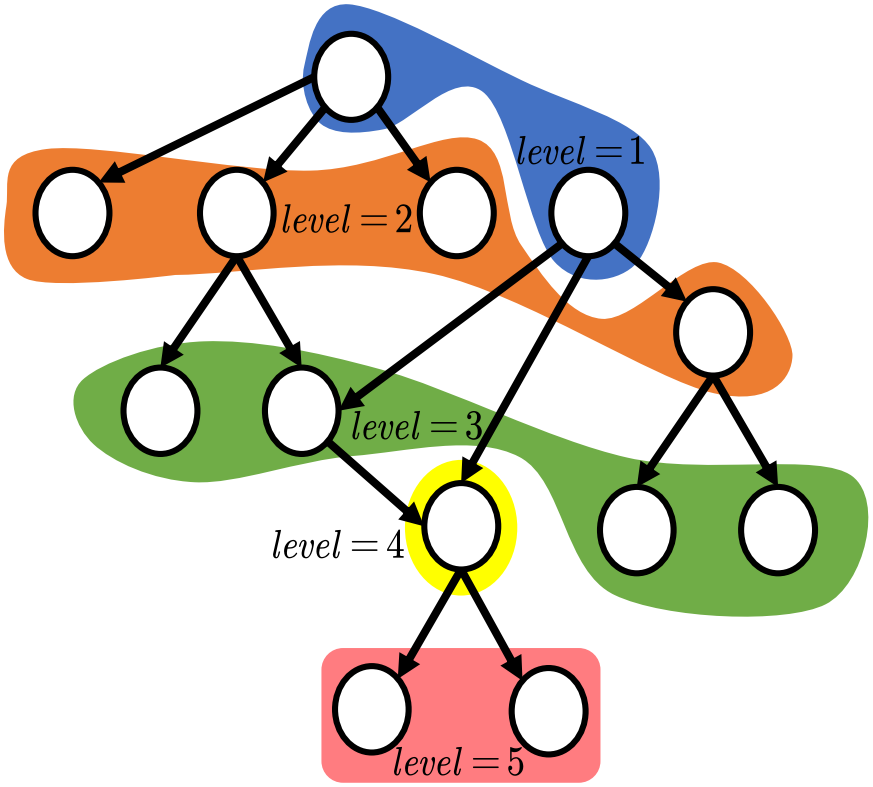
Directed Acyclic Graphs (DAGs)



- DAGs appear across various domains:

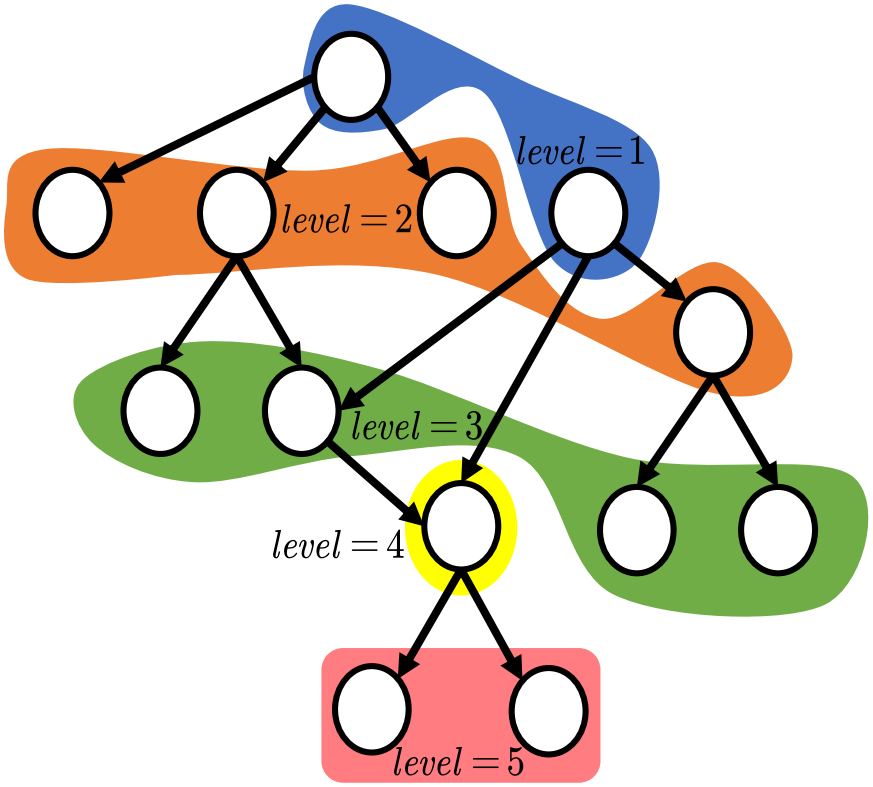
Source code, logical formulas, probabilistic graphical models, neural architectures (NNs), citation networks.

SOTA: Recursive neural networks over DAGs



- The node embedding is computed by iterating over the DAG nodes in an asynchronous way.
- At every node v, information is aggregated from the direct predecessors.

SOTA: Recursive neural networks over DAGs

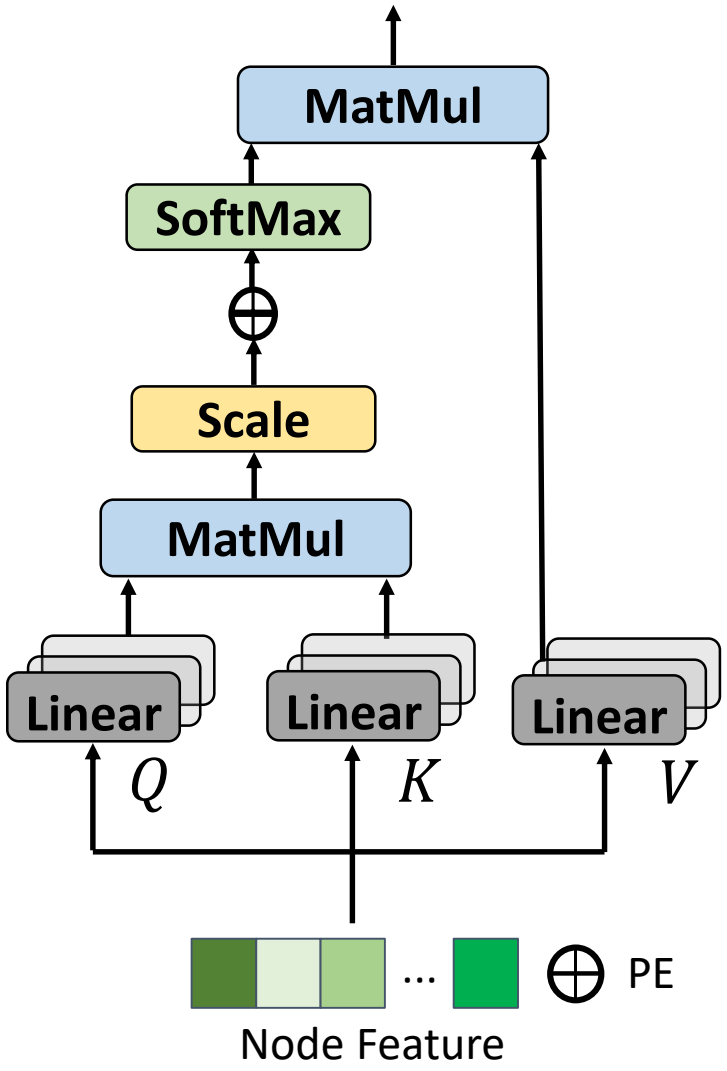


- At every node v, information is aggregated from the direct predecessors.
- The node embedding is computed by iterating over the DAG nodes in an asynchronous way.

These recursive DAG models are yielding SOTA results, but their asynchronous nature leads to very slow performance.

Transformers on Graphs

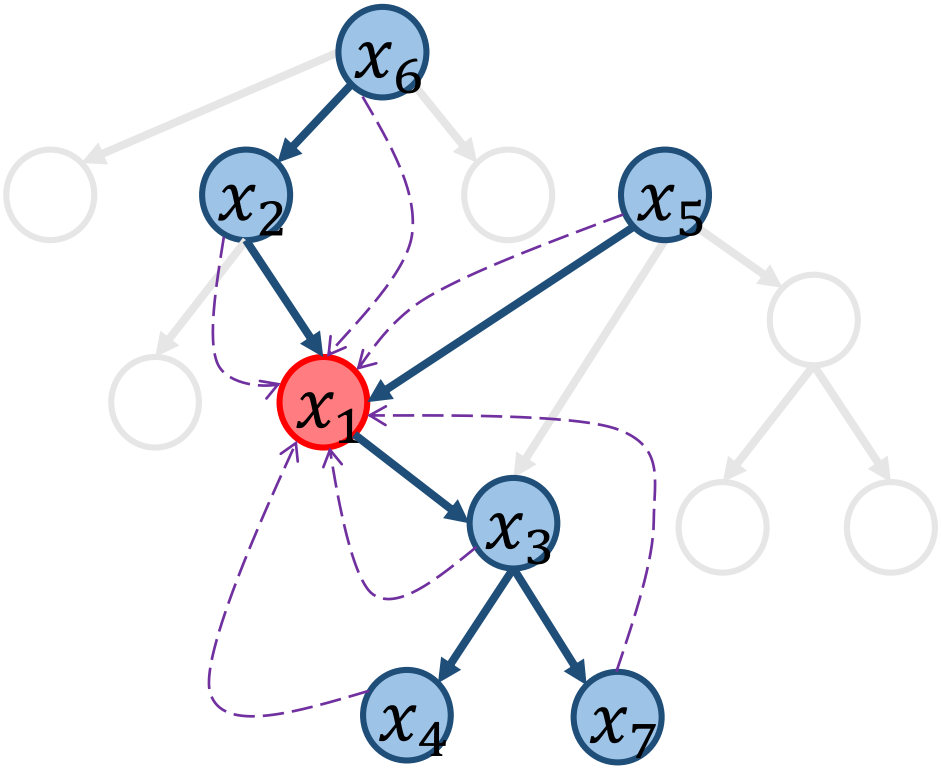
Note: DAGs are sequential in nature! And the parallelism of Transformers actually matches well the shortcomings of DAG neural networks.



Attention based on DAG Reachability (DAGRA)

- We restrict the receptive field of each node based on reachability:

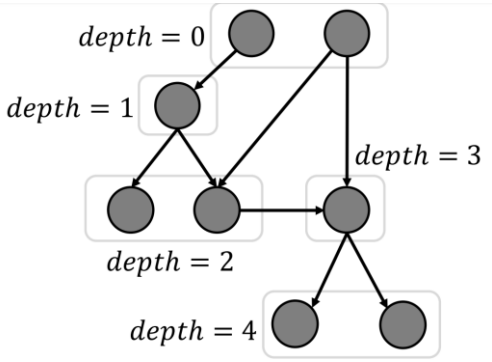
$$N(v) = \{(u,v) \in \leqslant\} \cup \{(v,u) \in \leqslant\}$$



PEs based on DAG Depth (DAGPE)

- Our positional encoding encodes (only) node depth.

$$depth(v) = \begin{cases} 0 & \text{if } v \text{ is a source node} \\ 1 + \max\limits_{(u,v)\in E} depth(u) & \text{otherwise} \end{cases}$$



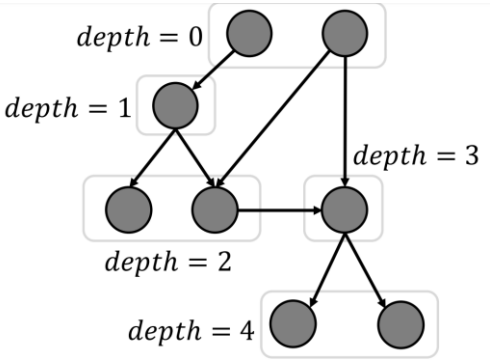
$$PE_{(v,2i)} = \sin\left(\frac{depth(v)}{10000^{\frac{2i}{d}}}\right)$$

$$PE_{(v,2i+1)} = \cos\left(\frac{depth(v)}{10000^{\frac{2i}{d}}}\right)$$

PEs based on DAG Depth (DAGPE)

- Our positional encoding encodes (only) node depth.

$$depth(v) = \begin{cases} 0 & \text{if } v \text{ is a source node} \\ 1 + \max\limits_{(u,v)\in E} depth(u) & \text{otherwise} \end{cases}$$



$$PE_{(v,\,2i)} = \sin\left(\frac{depth(v)}{10000^{\frac{2i}{d}}}\right)$$

$$PE_{(v,\,2i+1)} = \cos\left(\frac{depth(v)}{10000^{\frac{2i}{d}}}\right)$$

- Combining DAGRA and DAGPE, we obtain DAG Attention:

$$\text{Attention}(x_v) = \sum_{u\in N(v)} \frac{\kappa(x_v + PE_v, x_u + PE_u)}{\sum\limits_{w\in N(v)} \kappa(x_v + PE_v, x_w + PE_w)} f(x_u)$$

DAG Attention

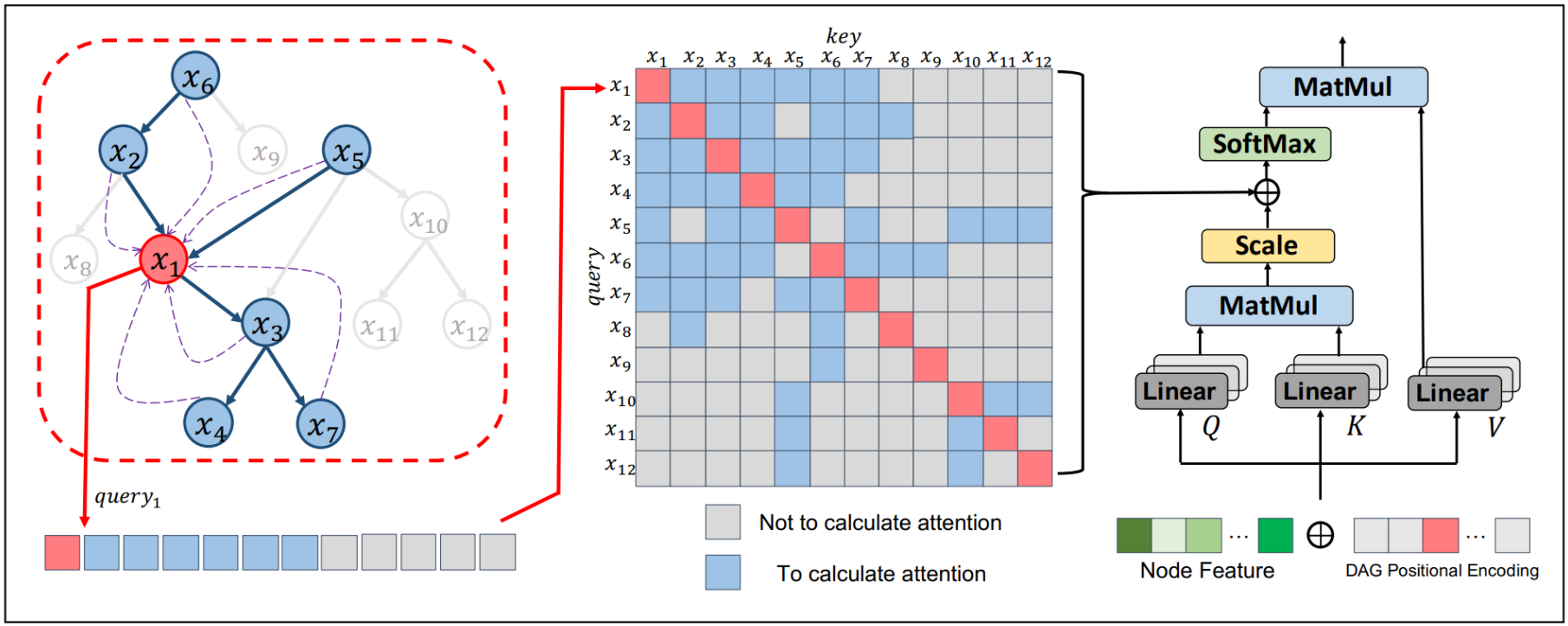
- Combining DAGRA and DAGPE, we obtain DAG Attention:

$$\text{Attention}(x_v) = \sum_{u \in N(v)} \frac{\kappa(x_v + PE_v, x_u + PE_u)}{\sum_{w \in N(v)} \kappa(x_v + PE_v, x_w + PE_w)} f(x_u)$$

This can be flexibly applied on top of existing graph transformers and tailor them to DAGs.

DAG Attention

- DAG attention using a mask that masks out node pairs based on the DAG reachability relation. $O(|V|^2 d)$



- DAG Attention using Message Passing $O(|V| \times N \times d)$

Aggregating messages using the *torch* <u>*scatter add*</u> method

Evaluation

Table 2: **Code graph classification** on ogbg-code2. The baseline results were taken from the OGB leaderboard.

Model	Valid F1 (%)	Test F1 (%)	Time(epoch)
GIN	13.7 ± 0.2	14.9 ± 0.2	181s
GCN	14.0 ± 0.2	15.1 ± 0.2	127s
GIN-Virtual	14.4 ± 0.3	15.8 ± 0.2	155s
GCN-Virtual	14.6 ± 0.1	16.0 ± 0.2	198s
GAT	14.4 ± 0.2	15.7 ± 0.2	134s
PNA	14.5 ± 0.3	15.7 ± 0.3	427s
DAGNN	16.1 ± 0.4	17.5 ± 0.5	6018s
PACE	16.3 ± 0.3	17.8 ± 0.2	2410s
Transformer	15.5 ± 0.2	16.7 ± 0.2	1817s
DAG+Transformer	**17.4** ± 0.1	**18.8** ± 0.2	591s
GraphTrans	16.6 ± 0.1	18.3 ± 0.2	1117s
DAG+GraphTrans	**17.0** ± 0.2	**18.7** ± 0.2	526s
GraphGPS	17.4 ± 0.2	18.9 ± 0.2	1919s
DAG+GraphGPS	**17.6** ± 0.1	**19.3** ± 0.2	608s
SAT (SOTA)	17.7 ± 0.2	19.4 ± 0.3	2437s
DAG+SAT	**18.5** ± 0.1	**20.2** ± 0.2	681s

Table 3: **Node classification** results for the self-citation dataset; AP (%) and ROC-AUC (%).

Model	AP ↑	ROC-AUC ↑
GIN	57.7 ± 1.8	79.7 ± 0.2
GCN	58.8 ± 0.4	79.9 ± 0.2
GIN-Virtual	57.4 ± 1.2	79.5 ± 0.4
GCN-Virtual	58.9 ± 0.2	80.0 ± 0.1
GAT	55.3 ± 3.7	77.9 ± 1.4
PNA	62.4 ± 0.7	81.0 ± 0.4
DAGNN	61.2 ± 0.6	81.0 ± 0.3
PACE	52.1 ± 1.8	75.9 ± 0.7
Transformer	56.8 ± 1.8	78.7 ± 0.3
DAG+Transformer	**63.8** ± 0.8	**82.2** ± 0.5
GraphGPS	61.6 ± 2.6	**81.3** ± 0.6
DAG+GraphGPS	**63.5** ± 1.2	80.8 ± 0.5
SAT	59.8 ± 1.7	79.8 ± 0.7
DAG+SAT	**62.7** ± 1.5	**80.6** ± 0.7
NodeFormer	39.6 ± 0.6	69.4 ± 0.3
DAG+NodeFormer	**64.9** ± 0.8	**81.7** ± 0.8

Evaluation

Table 4: **Node classification accuracy** (%).The baseline results were taken from [Wu et al., 2022].

Model	Cora	Citeseer	Pubmed
GCN	87.06 ± 0.34	75.75 ± 0.37	88.16 ± 0.14
GAT	86.85 ± 0.30	75.92 ± 0.26	86.90 ± 0.22
MixHop	87.59 ± 0.52	73.64 ± 0.73	89.32 ± 0.25
IDGL	87.88 ± 0.34	74.32 ± 0.51	89.22 ± 0.14
LDS-GNN	87.82 ± 0.62	75.22 ± 0.23	OOM
PACE	79.47 ± 0.63	73.65 ± 1.23	OOM
Transformer	75.92 ± 0.86	72.23 ± 1.06	OOM
DAG+Transformer	**87.80** ± 0.53	**74.42** ± 0.22	**89.01** ± 0.13
SAT	75.18 ± 0.62	74.88 ± 0.73	OOM
DAG+SAT	**87.48** ± 0.37	**76.64** ± 0.26	**89.17** ± 0.15
NodeFormer	88.80 ± 0.26	76.33 ± 0.59	89.32 ± 0.25
DAG+NodeFormer	**90.49** ± 0.17	**78.24** ± 0.33	**89.44** ± 0.24

Table 5: **Regression.** Predictive performance of latent representations over NA.

Model	RMSE ↓	Pearson's r ↑
GCN	0.482 ± 0.003	0.871 ± 0.001
S-VAE	0.521 ± 0.002	0.847 ± 0.001
D-VAE	0.375 ± 0.003	0.924 ± 0.001
DAGNN	0.264 ± 0.004	0.964 ± 0.001
PACE	0.254 ± 0.002	0.964 ± 0.001
Transformer	0.285 ± 0.004	0.957 ± 0.001
GT	0.275 ± 0.003	0.961 ± 0.001
DAG+Transformer	**0.253** ± 0.002	**0.966** ± 0.001
GraphGPS	0.306 ± 0.004	0.950 ± 0.001
DAG+GraphGPS	**0.267** ± 0.005	**0.964** ± 0.001
SAT	0.298 ± 0.003	0.952 ± 0.001
DAG+SAT	**0.262** ± 0.004	**0.964** ± 0.001

- Over all datasets, our DAG attention makes the transformers outperform

  (1) the original transformers and

  (2) the neural networks tailored to DAGs.

Conclusions

Our proposal proves effective in:

- Making graph transformers generally outperform neural networks tailored to DAGs
- Improving SOTA graph transformer performance in terms of both quality and efficiency.

https://github.com/LUOyk1999/DAGformer

Conclusions

Our proposal proves effective in:

- Making graph transformers generally outperform neural networks tailored to DAGs
- Improving SOTA graph transformer performance in terms of both quality and efficiency.

Thanks for listening!

https://github.com/LUOyk1999/DAGformer