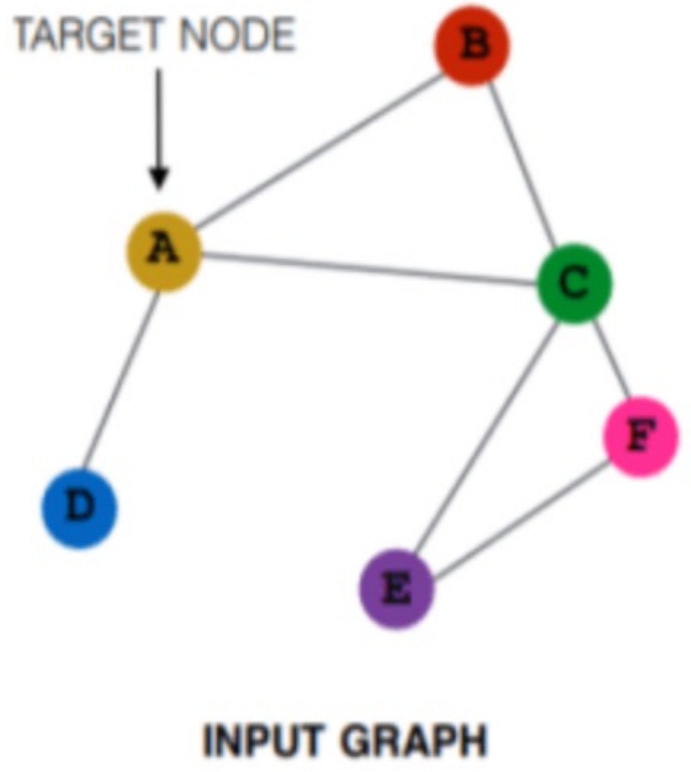
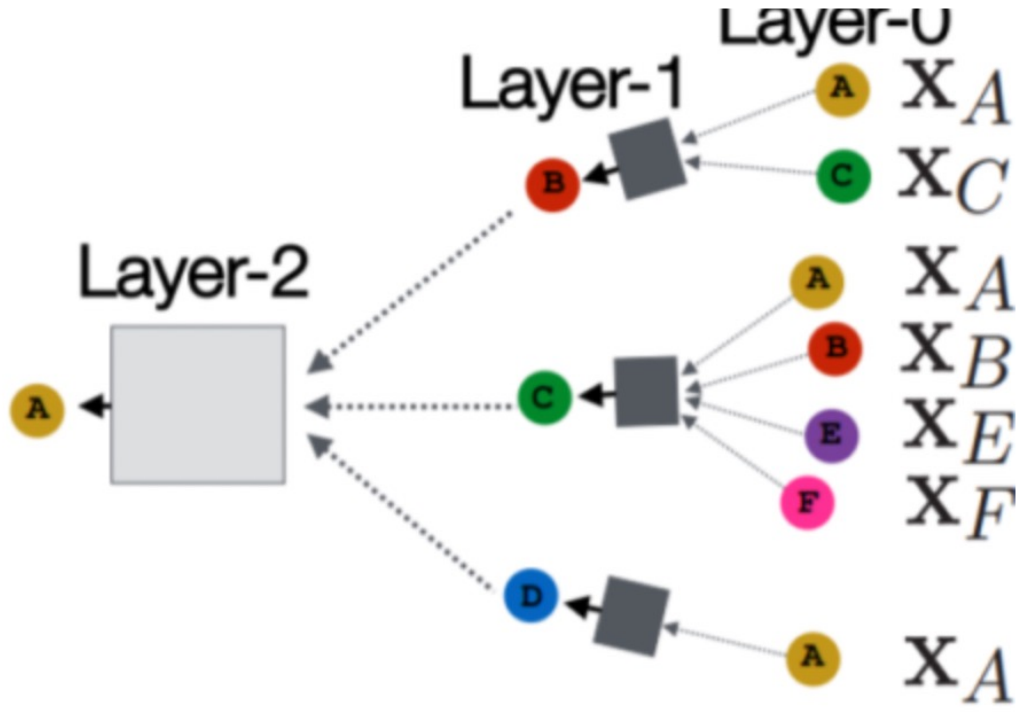
Do Transformers Really Perform Bad for Graph Representation?

GNN



- aim to learn representation of nodes and graphs
- iteratively updates the representation of a node by aggregating representations of its first or higher-order neighbors

GNN



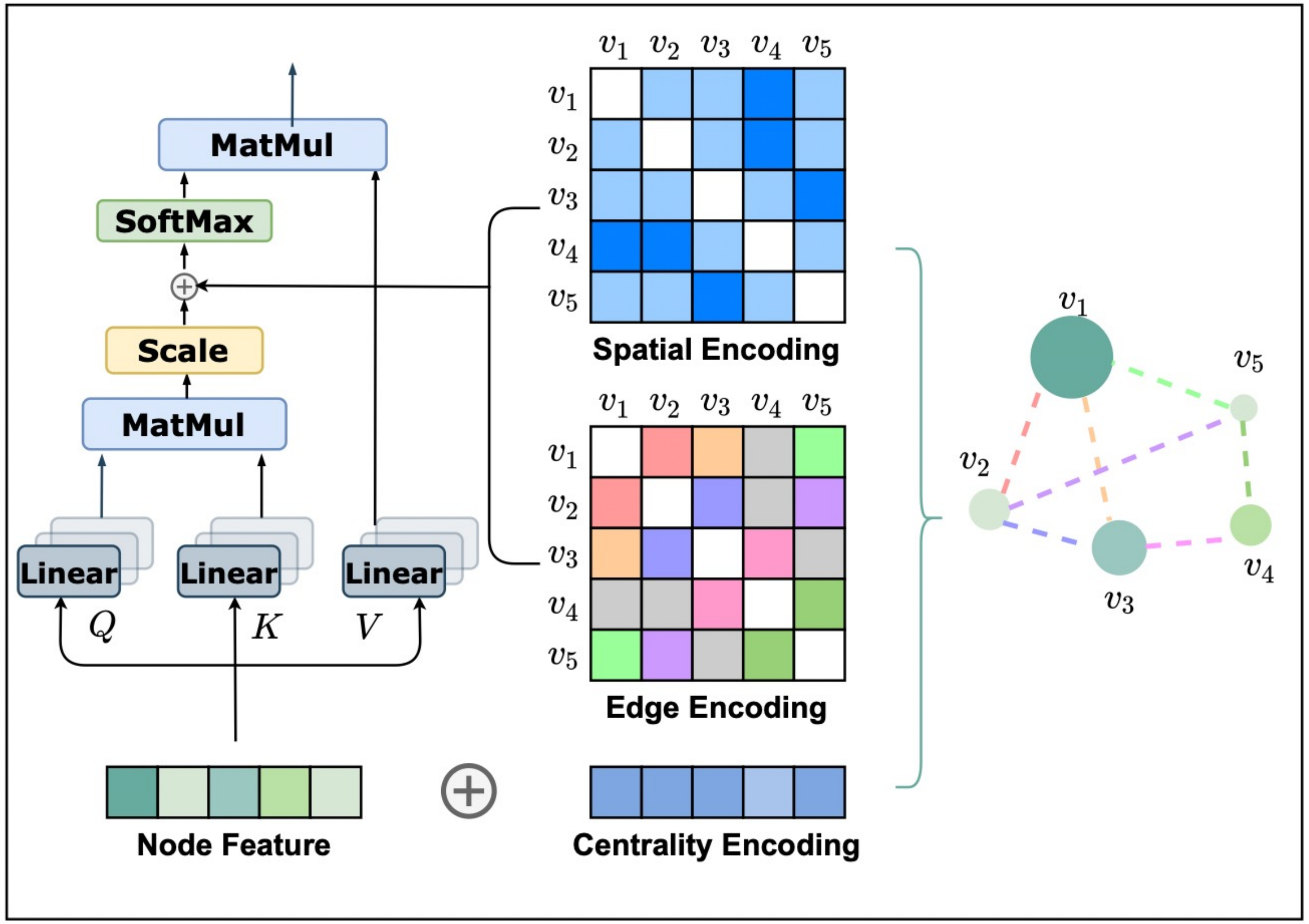
- $h_i^{(l)}$ is the representation of v_i at the l-th layer and define $h_i^{(0)} = x_i$
- l-th iteration of aggregation:

$$a_i^{(l)} = \text{AGGREGATE}^{(l)}\left(\left\{h_j^{(l-1)} : j \in \mathcal{N}(v_i)\right\}\right)$$

$$h_i^{(l)} = \text{COMBINE}^{(l)}\left(h_i^{(l-1)}, a_i^{(l)}\right)$$

Structural Encodings in Graphormer

to leverage the structural information of graphs into the Transformer model



$$Q = HW_Q, \quad K = HW_K, \quad V = HW_V,$$

$$A = \frac{QK^\top}{\sqrt{d_K}}, \quad \mathrm{Attn}\,(H) = \mathrm{softmax}\,(A)\,V,$$

Centrality Encoding

- attention distribution is calculated based on the semantic correlation between nodes

$$A = \frac{QK^\top}{\sqrt{d_K}}, \quad \mathrm{Attn}\,(H) = \mathrm{softmax}\,(A)\,V,$$

without considering the structural information of a graph

Centrality Encoding

- attention distribution is calculated based on the semantic correlation between nodes

$$A = \frac{QK^\top}{\sqrt{d_K}}, \quad \mathrm{Attn}(H) = \mathrm{softmax}(A)\,V,$$

without considering the structural information of a graph

$$h_i^{(0)} = x_i + z^-_{\deg^-(v_i)} + z^+_{\deg^+(v_i)},$$

$z^-, z^+ \in \mathbb{R}^d$ are learnable embedding vectors

Spatial Encoding

- model has to explicitly specify different positions or encode the positional dependency
- nodes are not arranged as a sequence (multi-dimensional spatial space and are linked by edges)

Spatial Encoding

- model has to explicitly specify different positions or encode the positional dependency
- nodes are not arranged as a sequence (multi-dimensional spatial space and are linked by edges)
- $\phi(v_i, v_j) : V \times V \to \mathbb{R}$ measures the spatial relation between v_i and v_j (connectivity between the nodes in the graph)

$$A_{ij} = \frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d}} + b_{\phi(v_i, v_j)},$$

where $b_{\phi(v_i, v_j)}$ is a learnable scalar indexed by $\phi(v_i, v_j)$, and shared across all layers.

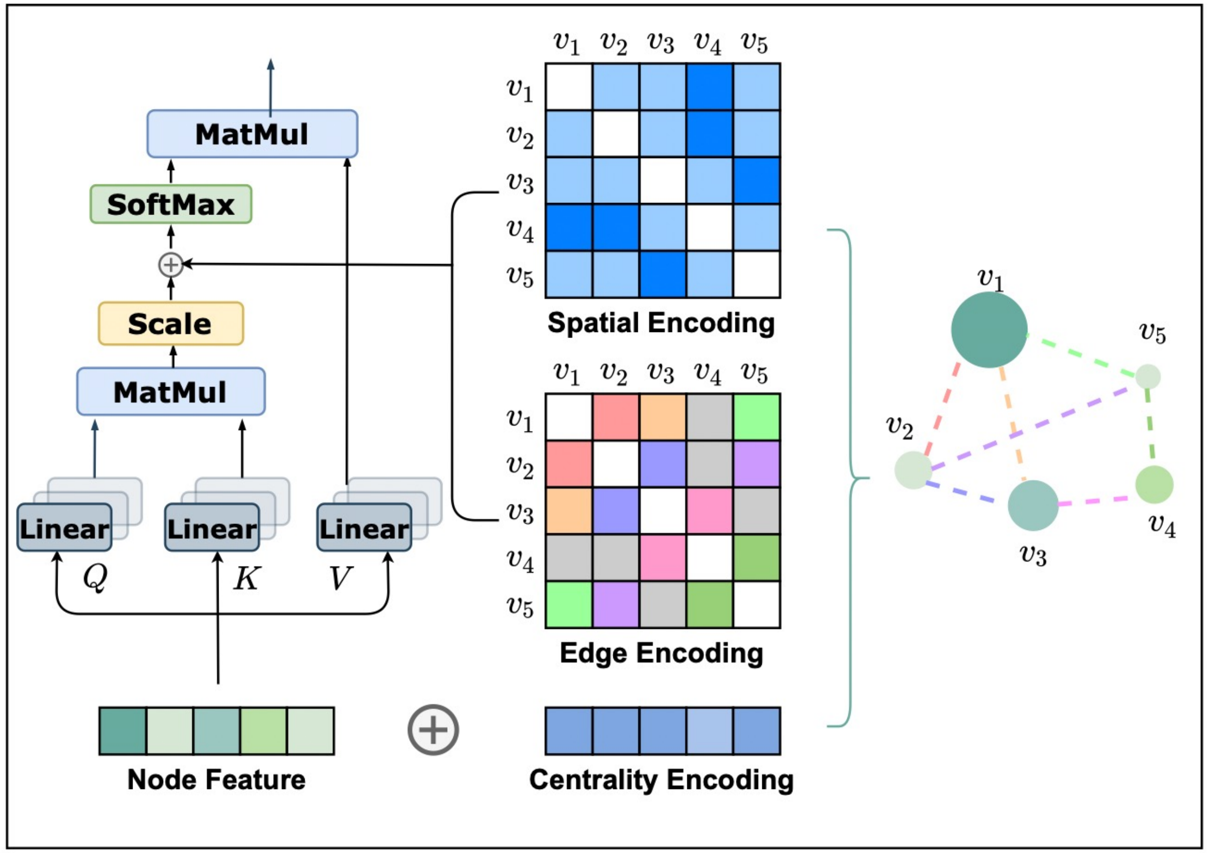
Edge Encoding

$$A_{ij} = \frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d}} + b_{\phi(v_i, v_j)} + c_{ij}, \text{ where } \underline{c_{ij} = \frac{1}{N} \sum_{n=1}^{N} x_{e_n} (w_n^E)^T,} \tag{7}$$

where x_{e_n} is the feature of the n-th edge e_n in SP_{ij}, $w_n^E \in \mathbb{R}^{d_E}$ is the n-th weight embedding, and d_E is the dimensionality of edge feature.

we find (one of) the shortest path $\mathrm{SP}_{ij} = (e_1, e_2, ..., e_N)$ from v_i to v_j,

Implementation details



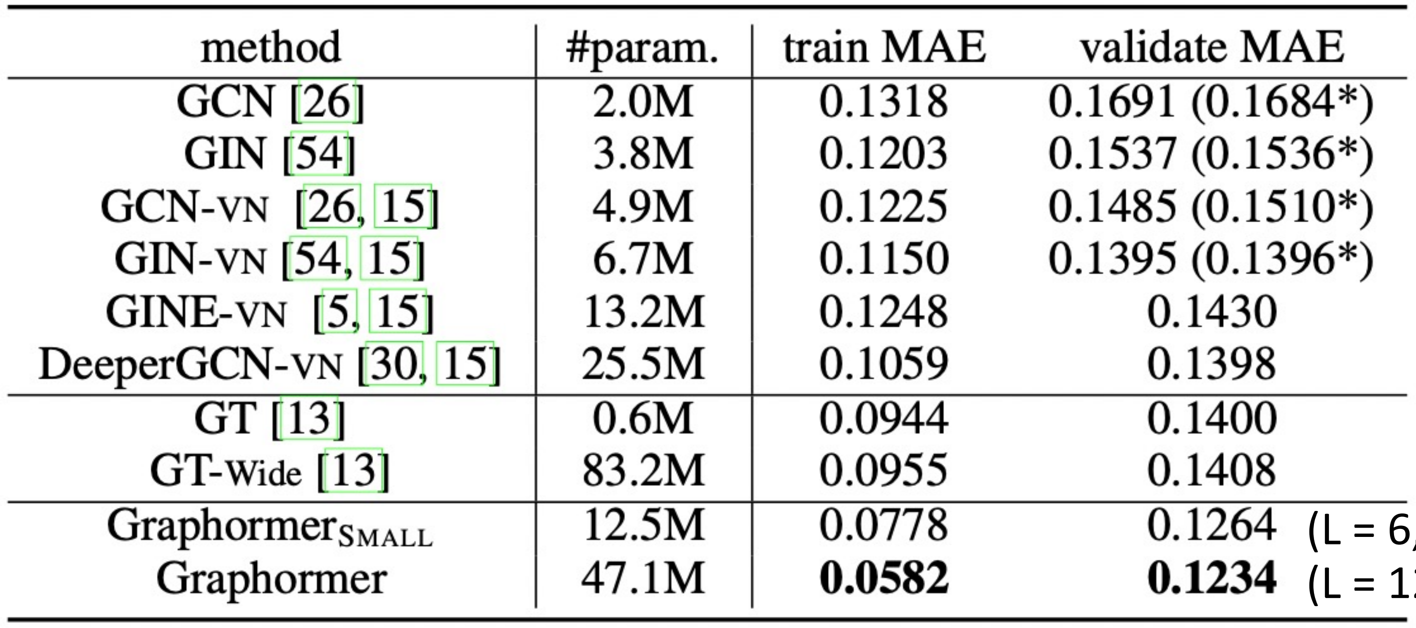
$$h'^{(l)} = \mathrm{MHA}(\mathrm{LN}(h^{(l-1)})) + h^{(l-1)}$$
$$h^{(l)} = \mathrm{FFN}(\mathrm{LN}(h'^{(l)})) + h'^{(l)}$$

add a **special node (VNode)** to the graph and make connection between it and each node:

- the representation of the entire graph G would be the node feature of [VNode] in the final layer
- the connection is not physical

biggest graph-level prediction dataset: 3.8M molecular graphs

Table 1: Results on PCQM4M-LSC. * indicates the results are cited from the official leaderboard [21].

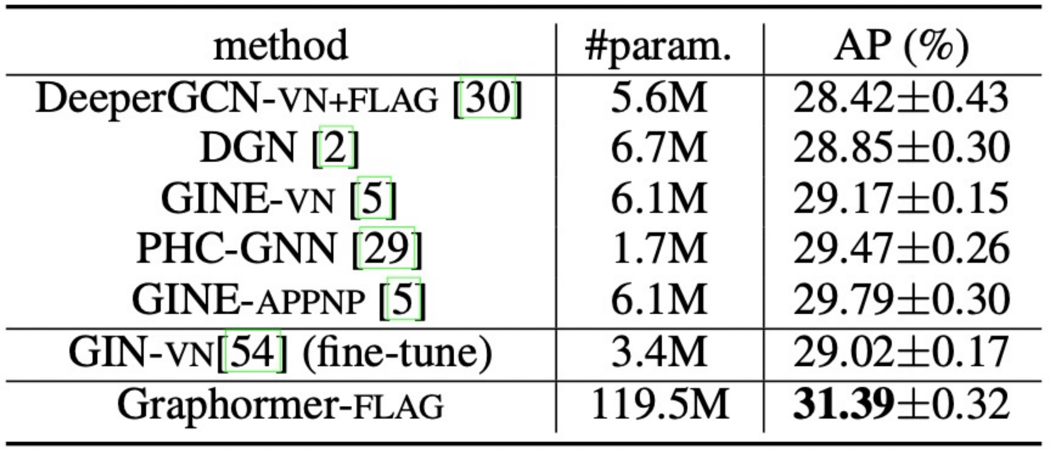


method	#param.	train MAE	validate MAE	
GCN [26]	2.0M	0.1318	0.1691 (0.1684*)	
GIN [54]	3.8M	0.1203	0.1537 (0.1536*)	
GCN-VN [26, 15]	4.9M	0.1225	0.1485 (0.1510*)	
GIN-VN [54, 15]	6.7M	0.1150	0.1395 (0.1396*)	
GINE-VN [5, 15]	13.2M	0.1248	0.1430	
DeeperGCN-VN [30, 15]	25.5M	0.1059	0.1398	
GT [13]	0.6M	0.0944	0.1400	
GT-Wide [13]	83.2M	0.0955	0.1408	
Graphormer$_{\text{SMALL}}$	12.5M	0.0778	0.1264	(L = 6, dimension = 512)
Graphormer	47.1M	**0.0582**	**0.1234**	(L = 12, dimension = 768)

AdamW, 60k-step warm-up stage followed by a linear decay learning rate scheduler
number of attention heads: 32
total training steps are 1M
batch size is set to 1024
8 NVIDIA V100 GPUS for about 2 days

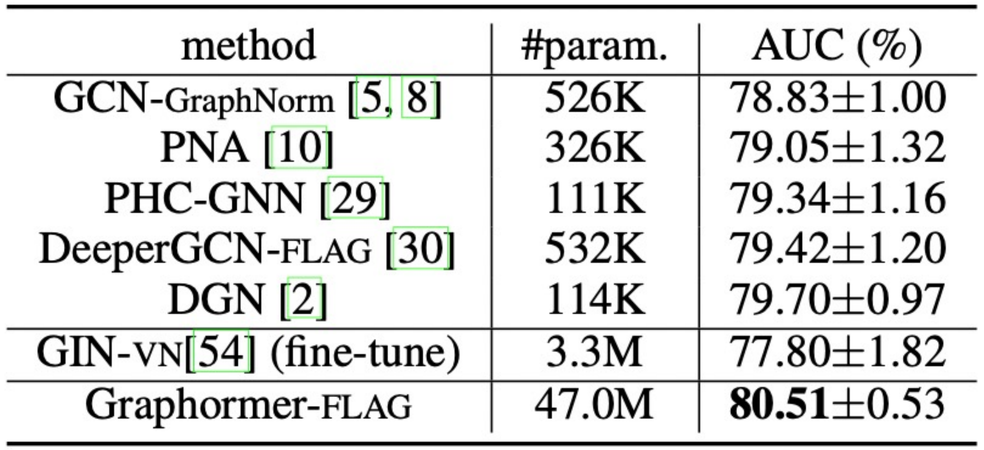
Transferable capability

Table 2: Results on MolPCBA.



method	#param.	AP (%)
DeeperGCN-VN+FLAG [30]	5.6M	28.42±0.43
DGN [2]	6.7M	28.85±0.30
GINE-VN [5]	6.1M	29.17±0.15
PHC-GNN [29]	1.7M	29.47±0.26
GINE-APPNP [5]	6.1M	29.79±0.30
GIN-VN[54] (fine-tune)	3.4M	29.02±0.17
Graphormer-FLAG	119.5M	**31.39**±0.32

Table 3: Results on MolHIV.



method	#param.	AUC (%)
GCN-GraphNorm [5, 8]	526K	78.83±1.00
PNA [10]	326K	79.05±1.32
PHC-GNN [29]	111K	79.34±1.16
DeeperGCN-FLAG [30]	532K	79.42±1.20
DGN [2]	114K	79.70±0.97
GIN-VN[54] (fine-tune)	3.3M	77.80±1.82
Graphormer-FLAG	47.0M	**80.51**±0.53

Graphormer model pre-trained on OGB-LSC (i.e., PCQM4M-LSC)
MolHIV: 40.000 molecular graphs
PCBA: 440.000 molecular graphs

Conclusions

- Simple and intuitive modifications to the regular Transformerc
- Complexity. Similar to regular Transformer, the attention mechanism in Graphormer scales quadratically with the number of nodes n
- Mainly evaluate general centrality and distance metric in graph theory