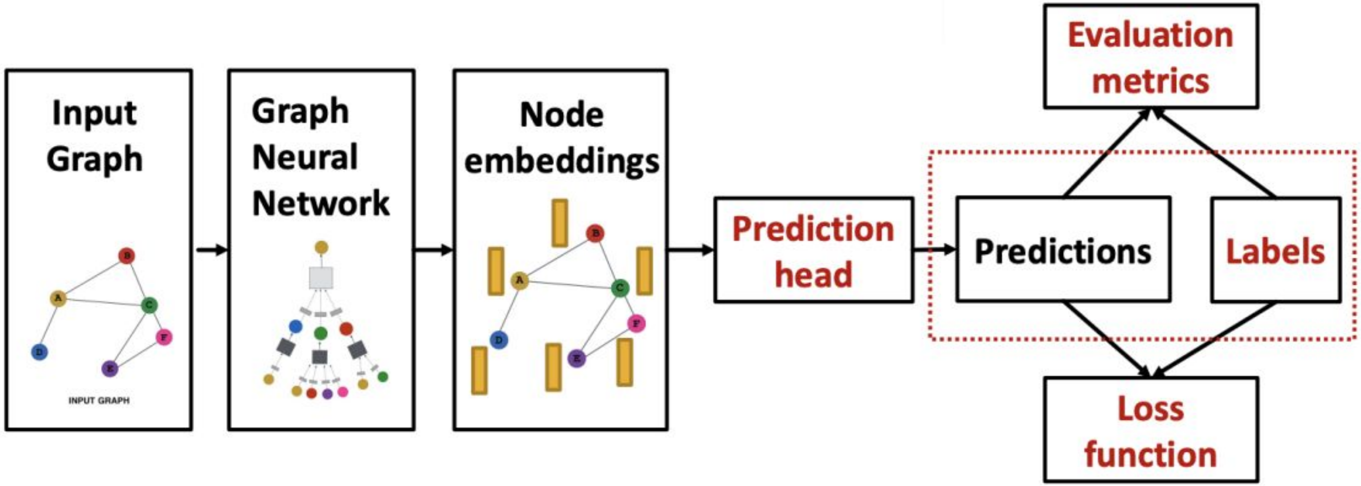
Нейросетевые методы для работы с графами

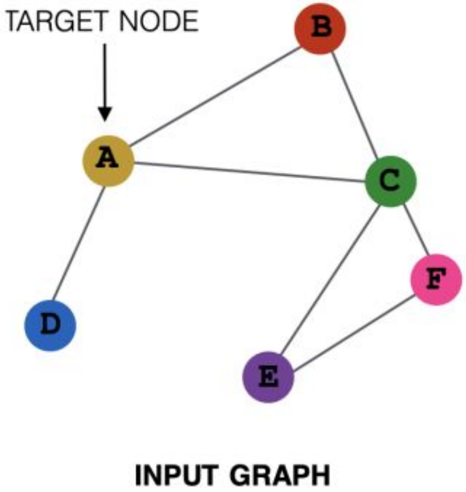
Ознобихин Арсений, БПМИ 213

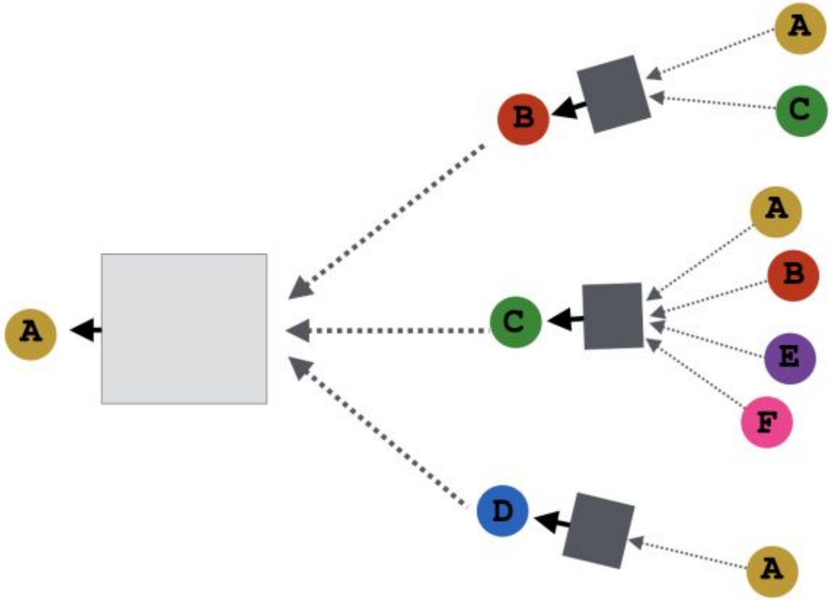
GNN and MPNN as framework

The simple variant of GNN pipeline:



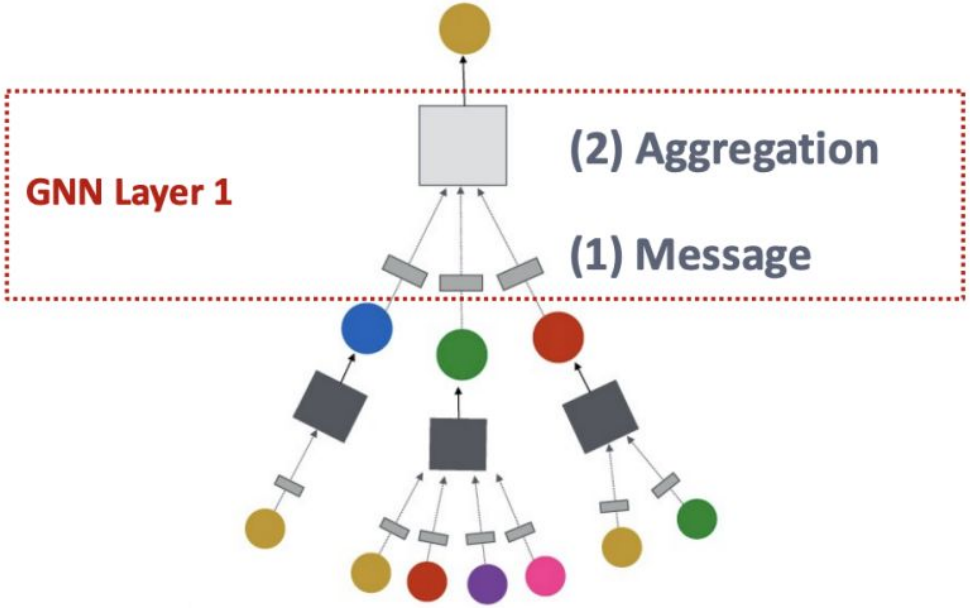
MPNN – neighbours aggregation





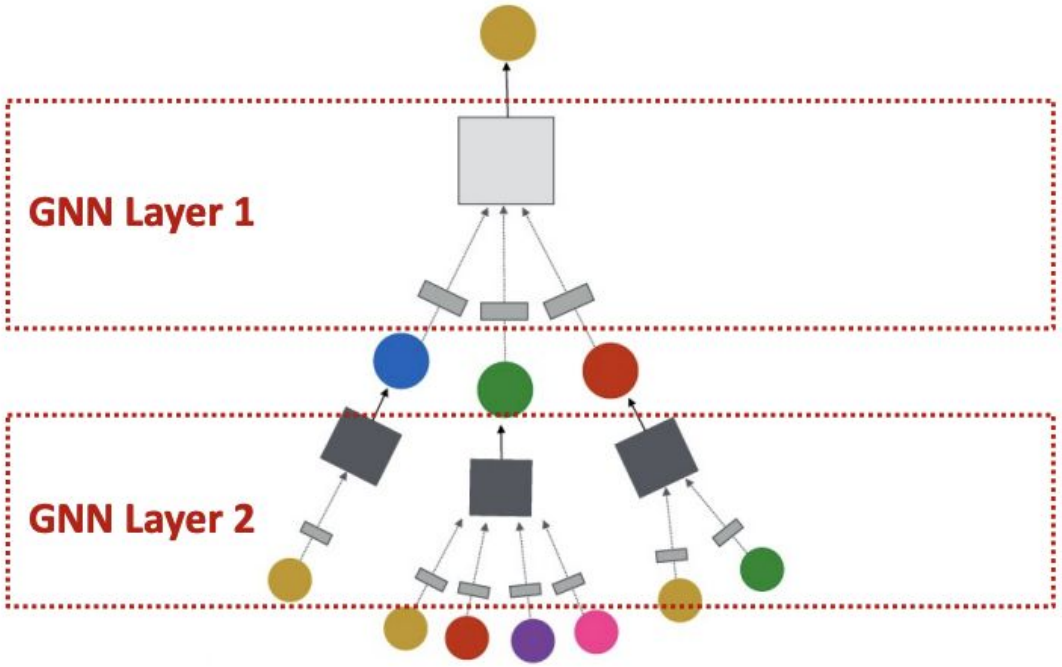
MPNN – GNN layer

- $h_v^{(l)}$ – hidden embedding
- $N(v)$ – neighbours of vertex v
- $m_u^{(l)}$ – message from vertex u

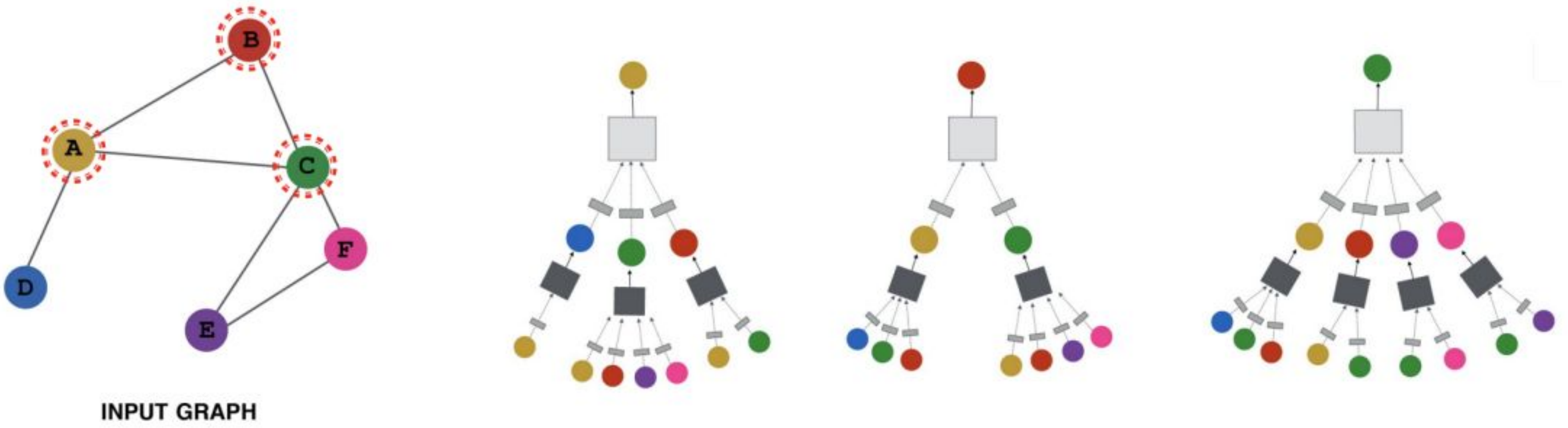


$$h_v^{(l+1)} = UPDATE(h_v^{(l)}, AGGREGATE(\{m_u^{(l)} : \ u \in N(v)\}))$$

MPNN – GNN layers stack



MPNN – batch train



MPNN – message computation

Message function

$$m_u^{(l)} = MSG^{(l)}(h_u^{(l-1)})$$

Example: a linear layer

$$m_u^{(l)} = W^{(l)}(h_u^{(l-1)})$$

MPNN – message aggregation

Aggregate messages from node v neighbours u:

$$h_u^{(l)} = AGG^{(l)}(\{m_u^{(l)},\ u \in N(v)\})$$

Aggregate examples: sum, mean, max.

$$h_u^{(l)} = CONCAT(AGG^{(l)}(\{m_u^{(l)}, u \in N(v)\}), m_v^{(l)})$$

MPNN – L2 normalization

Optionally we can add L2 normalization to each GNN layer:

$$\hat{h}_v^{(l)} := \frac{h_v^{(l)}}{||h_v^{(l)}||_2}$$

This will cause a different scale for final embeddings.

MPNN – attention mechanism

The basic idea is to assign an attention weight (or importance) to each neighbour to set its influence during aggregation steps.

We define importance $\alpha_{v,u}$. For example:

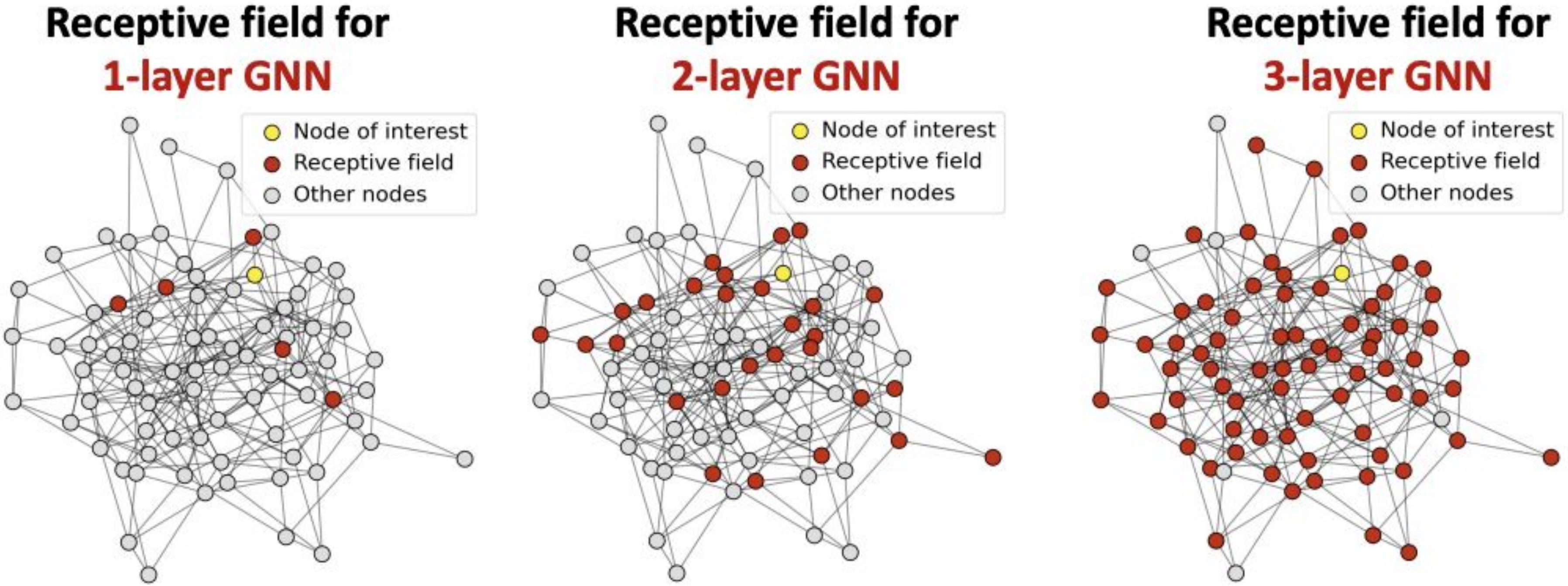
$$\alpha_{v,u} = \frac{\exp(h_u^T W h_v)}{\sum_{v' \in N(u)} exp(h_u^T W h_{v'})}$$

and then:

$$h_v^{(l)} = \sigma\left(\sum_{u \in N(v)} \alpha_{v,u} W^{(l)} h_u^{(l-1)}\right)$$

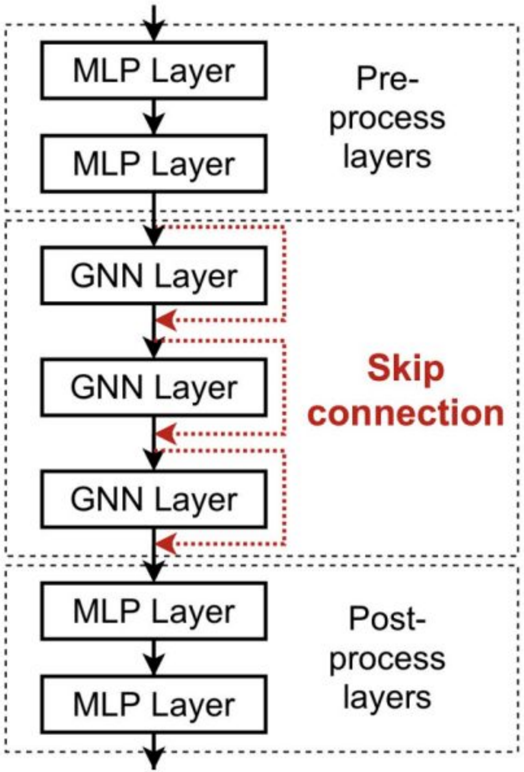
MPNN – over-smoothing problem

All nodes embeddings can converge to the same value



MPNN – possible layer improvements

- Add MPL layers before and after GNN layers
- Add skip connections to GNN layers



GraphSAGE – layer

$$h_v^{(l)} = \sigma\left(W^{(l)} \times CONCAT\left(h_v^{(l-1)}, AGG\left(\{h_u^{(l-1)}, u \in N(v)\}\right)\right)\right)$$

Message is computed inside AGG

$$\mathrm{AGG} = \underbrace{\sum_{u\in N(v)}}_{\text{Aggregation}} \underbrace{\frac{\mathbf{h}_u^{(l-1)}}{|N(v)|}}_{\text{Message computation}}$$

GraphSAGE – results

Table 1: Prediction results for the three datasets (micro-averaged F1 scores). Results for unsupervised and fully supervised GraphSAGE are shown. Analogous trends hold for macro-averaged scores.

	Citation		Reddit		PPI	
Name	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1
Random	0.206	0.206	0.043	0.042	0.396	0.396
Raw features	0.575	0.575	0.585	0.585	0.422	0.422
DeepWalk	0.565	0.565	0.324	0.324	—	—
DeepWalk + features	0.701	0.701	0.691	0.691	—	—
GraphSAGE-GCN	0.742	0.772	**0.908**	0.930	0.465	0.500
GraphSAGE-mean	0.778	0.820	0.897	0.950	0.486	0.598
GraphSAGE-LSTM	0.788	0.832	**0.907**	**0.954**	0.482	**0.612**
GraphSAGE-pool	**0.798**	**0.839**	0.892	0.948	**0.502**	0.600
% gain over feat.	39%	46%	55%	63%	19%	45%

MPNN – examples

Interaction Networks, Battaglia et al. (2016)

- Both node and graph level targets.
- Update function is some NN which takes

  $CONCAT(h_v, x_v, CONCAT(\{m_u, m \in N(v)\}))$,

  where x is some external vector.

- Message function is some NN which takes $CONCAT(h_v, h_u, e_{v,u})$.

MPNN – results

Table 2. Comparison of Previous Approaches (left) with MPNN baselines (middle) and our methods (right)

Target	BAML	BOB	CM	ECFP4	HDAD	GC	GG-NN	DTNN	enn-s2s	enn-s2s-ens5
mu	4.34	4.23	4.49	4.82	3.34	0.70	1.22	-	**0.30**	0.20
alpha	3.01	2.98	4.33	34.54	1.75	2.27	1.55	-	**0.92**	0.68
HOMO	2.20	2.20	3.09	2.89	1.54	1.18	1.17	-	**0.99**	0.74
LUMO	2.76	2.74	4.26	3.10	1.96	1.10	1.08	-	**0.87**	0.65
gap	3.28	3.41	5.32	3.86	2.49	1.78	1.70	-	**1.60**	1.23
R2	3.25	0.80	2.83	90.68	1.35	4.73	3.99	-	**0.15**	0.14
ZPVE	3.31	3.40	4.80	241.58	1.91	9.75	2.52	-	**1.27**	1.10
U0	1.21	1.43	2.98	85.01	0.58	3.02	0.83	-	**0.45**	0.33
U	1.22	1.44	2.99	85.59	0.59	3.16	0.86	-	**0.45**	0.34
H	1.22	1.44	2.99	86.21	0.59	3.19	0.81	-	**0.39**	0.30
G	1.20	1.42	2.97	78.36	0.59	2.95	0.78	.84[2]	**0.44**	0.34
Cv	1.64	1.83	2.36	30.29	0.88	1.45	1.19	-	**0.80**	0.62
Omega	0.27	0.35	1.32	1.47	0.34	0.32	0.53	-	**0.19**	0.15
Average	2.17	2.08	3.37	53.97	1.35	2.59	1.36	-	**0.68**	0.52

MPNN – conclusion

Advantages

- very flexible design
- taking into account the graph structure
- local and global perception

Disadvantages

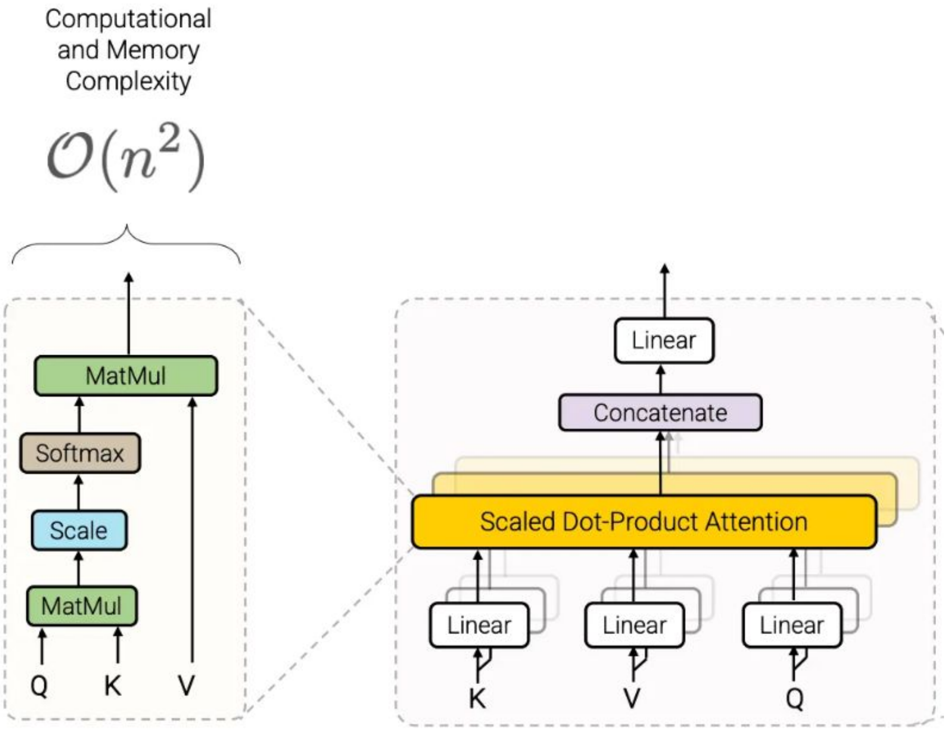
- huge computational cost
- insufficient on big graphs

Graphormer – motivation

- Transformers show state-of-the-art results in many other domains.
- We want to find alternative for GNN in the case of large graphs.

Graphormer – structural graph encoding

The main challenge is to develop the way to leverage the statical information of graph into Transformer.



Graphormer – centrality encoding

We want to show importance of some nodes in graph. In practice it`s enough to modify node features

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

Graphormer – spatial encoding

Let us A is a matrix with size $V \times V$ with spatial relations between graph nodes

$$A_{ij} = \frac{(h_i W_Q)(h_j W_k)^T}{\sqrt{d}} + b_{i,j}$$

Graphormer – edge encoding

In some tasks edges also have structural features.

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

$$c_{i,j} = \frac{1}{N_{i,j}} \sum_{n=1}^{N_{i,j}} x_{e_n} (w_n^E)^T$$

Graphormer – design

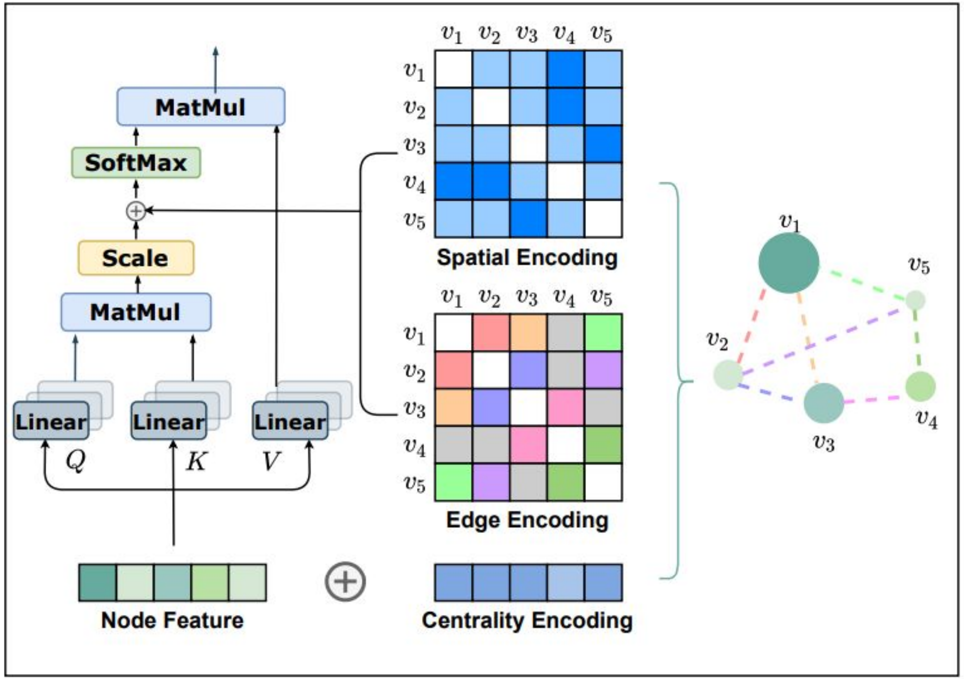


Figure 1: An illustration of our proposed centrality encoding, spatial encoding, and edge encoding in Graphormer.

Graphormer – results

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 [50]	3.8M	0.1203	0.1537 (0.1536*)
GCN-VN [26, 15]	4.9M	0.1225	0.1485 (0.1510*)
GIN-VN [50, 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
$\text{Graphormer}_{\text{SMALL}}$	12.5M	0.0778	0.1264
Graphormer	47.1M	**0.0582**	**0.1234**

Graphormer – conclusion

Advantages

- shows better results than other popular models
- by choosing proper parameters can replace AGGREGATE and UPDATE stages of some GNN models

Disadvantages

- takes a long time to learn (about 2 days and 1M learning steps)