

Graph 14. Reasoning with Heterogeneous Graphs (I)

- Graph Transformer Networks (GTN) [NeurIPS 2019]
- Graphormer [NeurIPS 2021]

1

Common limitation with conventional GNNs

- Assuming graph structures are **fixed and homogeneous**
 - Such as GCN, GAT, GIN, etc.
- However, real-world graphs are often **heterogeneous**
 - Such as knowledge graphs (previous lectures)
 - Even for citation networks, often the entities (authors, documents, conferences) and the relations (author-paper, paper-conference) are of heterogeneous types

2



Graph Transformer Networks (GTN)

[S Yun et al., NeurIPS 2019]

Key Idea & Contributions

- Automatically learning **useful meta-paths** in a heterogeneous graph instead manually specifying them;
- Constructing a **new graph ("homogeneous")** where the nodes are the same as in the original graph, but the edges are **replaced by the learned meta-paths**;
- Obtaining node embedding with a regular GCN on the newly constructed graph;
 - **Enhanced SOTA results** on evaluation benchmarks for node (multi-class) classification;
 - Offering **interpretable explanation with meta-paths** for the system's predictions.

4/4/2024

@Yiming Yang, 11-741 S24 Graph11 Other CO Solvers (LLMs)

5

5

Notation [S Yun et al., NeurIPS 2019]

- Denote by $G = (V, E)$ the input graph with $|V| = N$ nodes and K types of edges.
- Denote by T^e as the set of edge types and $|T^e| = K$.
- Denote by $A_k \in \mathbb{R}^{N \times N}$ the adjacency matrix with $A_k[i, j] \neq 0$ if there is an edge of type k from j to i .
- Denote by $\mathbf{A} \in \mathbb{R}^{N \times N \times K}$ the tensor of the adjacency matrices for edge type $k = 1, \dots, K$.
- Denote by p a **meta-path** with l heterogeneous edges $v_1 \xrightarrow{t_1} v_2 \xrightarrow{t_2} \dots \xrightarrow{t_l} v_{l+1}$ such that $t_j \in T^e$.
- Denote by $R_p = t_1 \circ t_2 \circ \dots \circ t_l$ the composite relation of meta-path p .
- Denote by $A_p = A_{t_1} A_{t_2} \dots A_{t_l}$ (in a previous work) the adjacency matrix of meta-path p .

4/4/2024

@Yiming Yang, 11-741 S24 Graph11 Other CO Solvers (LLMs)

6

6

Meth-path Generation

- In a previous work [WWW 2019]

$$A_p = A_{t_1} A_{t_2} \dots A_{t_l}$$

multiplying an adj. matrix is like taking k paths

- Proposed new way [S Yun et al., NeurIPS 2019]

$$\begin{aligned} A_p^{(l)} &= \left(\sum_{t_1 \in T^e} \alpha_{t_1}^{(1)} A_{t_1} \right) \left(\sum_{t_2 \in T^e} \alpha_{t_2}^{(2)} A_{t_2} \right) \dots \left(\sum_{t_l \in T^e} \alpha_{t_l}^{(l)} A_{t_l} \right) \\ &= \sum_{t_1, t_2, \dots, t_l \in T^e} \alpha_{t_1}^{(1)} \alpha_{t_2}^{(2)} \dots \alpha_{t_l}^{(l)} A_{t_1} A_{t_2} \dots A_{t_l} \end{aligned}$$

- Resulted $A_p^{(l)} \in R^{N \times N}$ is the matrix of meta-path weights, where each edge in the meta-path has a **learnable weight** (the red parts) .

4/4/2024

@Yiming Yang, 11-741 S24 Graph11 Other CO Solvers (LLMs)

7

7

Graph Transformer with soft selection of edge types

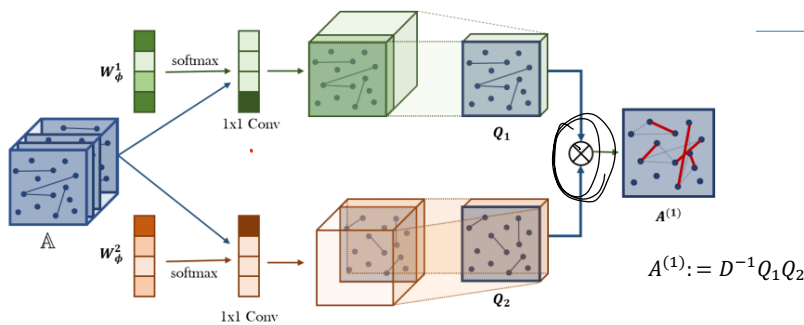


Figure 1: **Graph Transformer Layer** softly selects adjacency matrices (edge types) from the set of adjacency matrices \mathbb{A} of a heterogeneous graph G and learns a new meta-path graph represented by $A^{(1)}$ via the matrix multiplication of two selected adjacency matrices Q_1 and Q_2 . The soft adjacency matrix selection is a weighted sum of candidate adjacency matrices obtained by 1×1 convolution with non-negative weights from $\text{softmax}(W_\phi^1)$.

4/4/2024

@Yiming Yang, 11-741 S24 Graph11 Other CO Solvers (LLMs)

8

8

Multi-channel Convolution Layers + GCN for Learning $A^{(l)}$

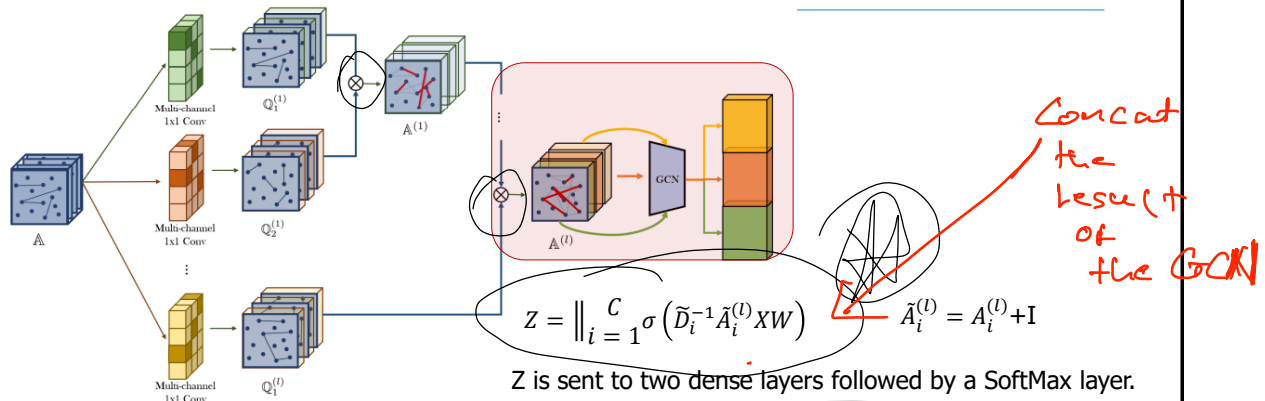


Figure 2: Graph Transformer Networks (GTNs) learn to generate a set of new meta-path adjacency matrices $A^{(l)}$ using GT layers and perform graph convolution as in GCNs on the new graph structures. Multiple node representations from the same GCNs on multiple meta-path graphs are integrated by concatenation and improve the performance of node classification. $Q_1^{(l)}$ and $Q_2^{(l)} \in \mathbf{R}^{N \times N \times C}$ are intermediate adjacency tensors to compute meta-paths at the l th layer.

9

9

Adding Identity matrix in A to learn variable length meta-paths

- Originally

$$A = \{A_1, A_2, \dots, A_k\}$$

- Adding identity matrix in A

$$A = \{A_0, A_1, A_2, \dots, A_k\} \quad \text{where } A_0 = I$$

- Why the latter is better?

- Consider $f(x_1, x_2) = (ax_1 + bx_2)^2 = a^2x_1^2 + 2abx_1x_2 + b^2x_2^2$
- Verses $\phi(x_1, x_2) = (1 + ax_1 + bx_2)^2 = 1 + 2ax_1 + 2bx_2 + a^2x_1^2 + 2abx_1x_2 + b^2x_2^2$
- The former only have quadratic terms and the latter have scaler, linear and quadratic ones.

Evaluation Results on Multi-class Classification of Nodes

[S Yun et al., NeurIPS 2019]

Table 2: Evaluation results on the node classification task (F1 score).

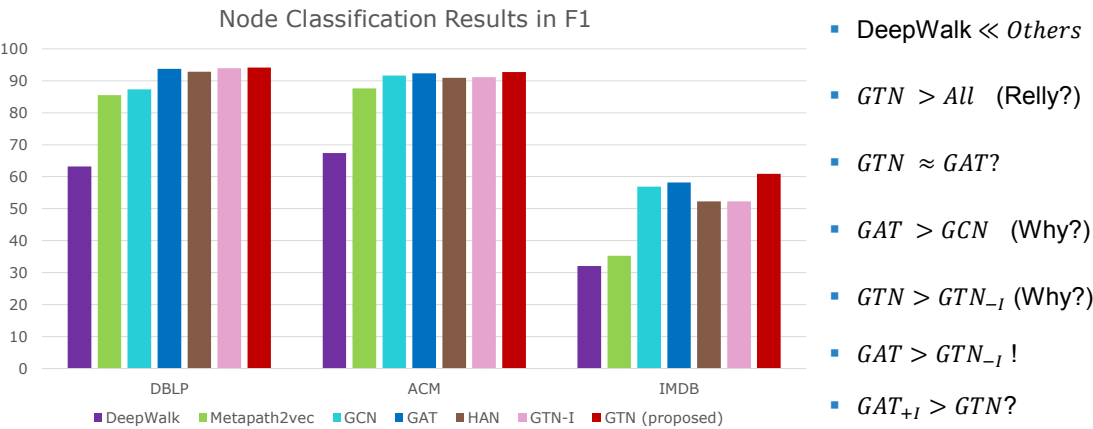
	DeepWalk	metapath2vec	GCN	GAT	HAN	GTN _{-I}	GTN (proposed)
DBLP	63.18	85.53	87.30	93.71	92.83	93.91	94.18
ACM	67.42	87.61	91.60	92.33	90.96	91.13	92.68
IMDB	32.08	35.21	56.89	58.14	56.77	52.33	60.92

- *GTN > All Other Methods (Really?)*

11

Evaluation Results on Multi-class Classification of Nodes

(my graph)



12

Concluding Remarks on GTN

- GTN allows **automated learning of the weights for meta-paths** in various lengths w.r.t. down-stream tasks
- The highly weighted meta-paths provide **understandable interpretation** of the predictions
 - e.g., MAD = (Forest Gump, Tom Hanks, R Zemeckis) → 5-star movie rating
- The **composite adjacency matrix** enables traditional GNNs to be used as plug-in modules for down-stream tasks (instead of designing a new GNN for meth-path based reasoning).
- Evaluation results (on node classification) show similar performance of GTN and GAT.
 - **Can we further improve GAT with simple tricks?**

4/4/2024

@Yiming Yang, 11-741 S24 Graph11 Other CO Solvers (LLMs)

13

13

Do Transformers Relly Perform Bad for Graph Representation? [C Ying et al., NeurIPS 2021]

- Proposed: **Graphomer**
 - Build upon the standard Tranformer architecture
 - Excellent results on many graph-based learning tasks
 - Several **simple encoding tricks** for enhancing effectiveness

4/4/2024

@Yiming Yang, 11-741 S24 Graph11 Other CO Solvers (LLMs)

14

14

The Embedding Tricks

- Centrality Encoding

$$h_i^{(0)} = x_i + z_{\text{in_deg}(v_i)}^- + z_{\text{in_deg}(v_i)}^+ \quad (1)$$

where $z^-, z^+ \in \mathbb{R}^d$ are the embedding of the in/out degrees of a node.

- Spatial Encoding (Extended Attention)

$$A_{ij} = \frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d}} + b_{\phi(v_i, v_j)} \quad (2)$$

where $\phi(v_i, v_j)$ is the distance of the shortest paths between nodes v_i and v_j ;

$b_{\phi(v_i, v_j)} \in \mathbb{R}$ is learnable function of $\phi(v_i, v_j)$.

4/4/2024

@Yiming Yang, 11-741 S24 Graph11 Other CO Solvers (LLMs)

15

15

The Embedding Tricks (cont'd)

- Edge Encoding :

$$c_{ij} = \frac{1}{N} \sum_{n=1}^N x_{e_n}^T w_n \quad (3)$$



where e_n is the n 'th edge in the shortest path of between nodes v_i and v_j ;

$x_{e_n} \in \mathbb{R}^{d_E}$ is the embedding of the input edge features;

$w_n \in \mathbb{R}^{d_E}$ is the embedding of the learnable weight for the n 'th edge in the path.

- Put (1), (2) and (3) together



$$A_{ij} = \frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d}} + b_{\phi(v_i, v_j)} + c_{ij} \quad (4)$$

4/4/2024

@Yiming Yang, 11-741 S24 Graph11 Other CO Solvers (LLMs)

16

16

Graphormer Layers

$$h'^{(l)} = MHA(LN(h^{(l-1)}) + h^{(l-1)}) \quad (5)$$

$$h^{(l)} = FFN(LN(h'^{(l)}) + h'^{(l)}) \quad (6)$$

$$h^{(G)} = \text{READOUT}(\{h_i^{(L)} | v_i \in G\}) \quad (7)$$

where LN stands for Layer Normalization,;

MHA stands for multi-head self-attention;

FFN stands for point-wise feed-forward neural network.

4/4/2024

@Yiming Yang, 11-741 S24 Graph11 Other CO Solvers (LLMs)

17

17

Benchmark Evaluation Datasets

[C Ying et al., NeurIPS 2021]

Table 6: Statistics of the datasets.

Dataset	Scale	# Graphs	# Nodes	# Edges	Task Type
PCQM4M-LSC	Large	3,803,453	53,814,542	55,399,880	Regression
OGBG-MolPCBA	Medium	437,929	11,386,154	12,305,805	Binary classification
OGBG-MolHIV	Small	41,127	1,048,738	1,130,993	Binary classification
ZINC (sub-set)	Small	12,000	277,920	597,960	Regression

The task of PCQM4M-LSC is to predict DFT(density functional theory)-calculated HOMO-LUMO energy gap of molecules given their 2D molecular graphs, which is one of the most practically-relevant quantum chemical properties of molecule science. PCQM4M-LSC is unprecedentedly large in scale comparing to other labeled graph-level prediction datasets, which contains more than 3.8M graphs. Besides, we conduct experiments on two molecular graph datasets in popular OGB leaderboards, i.e., OGBG-MolPCBA and OGBG-MolHIV. They are two molecular property prediction datasets with different sizes. The pre-trained knowledge of molecular graph on PCQM4M-LSC could be easily leveraged on these two datasets. We adopt official scaffold split on three datasets following [21, 22]. In addition, we employ another popular leaderboard, i.e., benchmarking-gnn [14]. We use the ZINC datasets, which is the most popular real-world molecular dataset to predict graph property regression for constrained solubility, an important chemical property for designing generative GNNs for molecules. Different from the scaffold splitting in OGB, uniform sampling is adopted in ZINC for data splitting.

4/4/2024

@Yiming Yang, 11-741 S24 Graph11 Other CO Solvers (LLMs)

18

18

Benchmark Evaluation Results

[C Ying et al., NeurIPS 2021]

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

-VN: virtual node

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
12-layer → 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 _{SMALL}	12.5M	0.0778	0.1264
Graphormer	47.1M	0.0582	0.1234

19

Benchmark Evaluation Results

[C Ying et al., NeurIPS 2021]

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 4: Results on ZINC.

method	#param.	test MAE
GIN [54]	509,549	0.526±0.051
GraphSage [18]	505,341	0.398±0.002
GAT [50]	531,345	0.384±0.007
GCN [26]	505,079	0.367±0.011
GatedGCN-PE [4]	505,011	0.214±0.006
MPNN (sum) [15]	480,805	0.145±0.007
PNA [10]	387,155	0.142±0.010
GT [13]	588,929	0.226±0.014
SAN [28]	508,577	0.139±0.006
Graphormer _{SLIM}	489,321	0.122±0.006

20

Concluding Remarks on Graphormer

- 
▪ Graphormer captures the strengths of edges in graph-based information propagation via **enriched attention**, which incorporates node embedding, edge-feature embedding, and learnable edge weights in multi-hop (shortest) paths between node pairs.
- 
▪ Compared to the **Graph Transformer Network (GTN)** [NeurIPS 2019], which relies on **a final set of edge types** and **explicit calculation of meta-paths**, Graphormer is more flexible or expressive.
- 
▪ Evaluation results on a broad range of regression/classification tasks on the largest evaluation benchmarks show superior performance of Graphormer over GAT, GCN and GIN (where GTN cannot be compared due to the lack of available results.)

4/4/2024

@Yiming Yang, 11-741 S24 Graph11 Other CO Solvers (LLMs)

21

21

References

- [NeurIPS 2019] [Graph Transformer Networks](#) Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, Hyunwoo J. Kim
- [NeurIPS 2021] [Do Transformers Really Perform Bad for Graph Representation?](#) Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, Tie-Yan Liu.

4/4/2024

@Yiming Yang, 11-741 S24 Graph11 Other CO Solvers (LLMs)

22

22