Pure Transformers are Powerful Graph Learners

Authors

Pure Transformers are Powerful Graph Learners

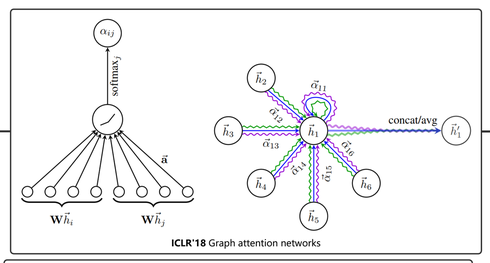
Jinwoo Kim[1*] **Tien Dat Nguyen**[1] **Seonwoo Min**[2] **Sungjun Cho**[2]
Moontae Lee[2,3] **Honglak Lee**[2,4†] **Seunghoon Hong**[1,2†]
[1]KAIST [2]LG AI Research [3]University of Illinois Chicago [4]University of Michigan

Background

Different types of Graph Transformer

Restrcting self-attention to local neighborhoods

GAT

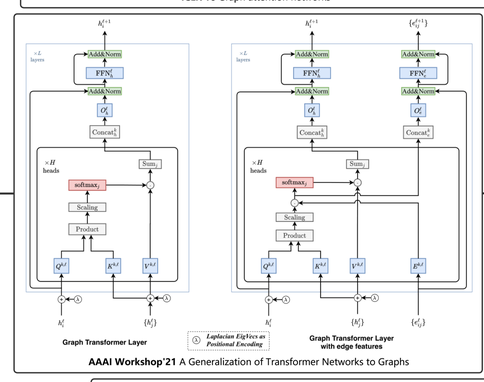


ICLR'18 Graph attention networks

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k\mathbf{W}^k\vec{h}_j\right)$$

GT



AAAI Workshop'21 A Generalization of Transformer Networks to Graphs

$$\hat{h}_i^{\ell+1} = O_h^\ell \Big\|_{k=1}^{H}\Big(\sum_{j\in\mathcal{N}_i} w_{ij}^{k,\ell}V^{k,\ell}h_j^\ell\Big), \quad (4)$$

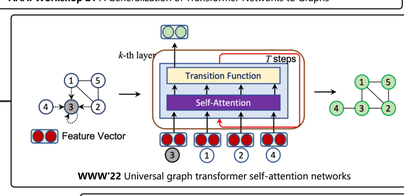
$$\text{where, } w_{ij}^{k,\ell} = \text{softmax}_j\left(\frac{Q^{k,\ell}h_i^\ell \cdot K^{k,\ell}h_j^\ell}{\sqrt{d_k}}\right), \quad (5)$$

$$\hat{h}_i^{\ell+1} = \text{Norm}\left(h_i^\ell + \hat{h}_i^{\ell+1}\right), \quad (6)$$

$$\hat{\hat{h}}_i^{\ell+1} = W_2^\ell\text{ReLU}(W_1^\ell\hat{h}_i^{\ell+1}), \quad (7)$$

$$h_i^{\ell+1} = \text{Norm}\left(\hat{h}_i^{\ell+1} + \hat{\hat{h}}_i^{\ell+1}\right), \quad (8)$$

UGformer Variant 1



WWW'22 Universal graph transformer self-attention networks

$$\mathbf{x}_{t,u}^{(k)} = \text{LayerNormalization}\left(\mathbf{h}_{t-1,u}^{(k)} + \text{ATT}\left(\mathbf{h}_{t-1,u}^{(k)}\right)\right) \quad (1)$$

$$\mathbf{h}_{t,u}^{(k)} = \text{LayerNormalization}\left(\mathbf{x}_{t,u}^{(k)} + \text{Trans}\left(\mathbf{x}_{t,u}^{(k)}\right)\right) \quad (2)$$

$$\text{ATT}\left(\mathbf{h}_{t-1,u}^{(k)}\right) = \sum_{u'\in\mathcal{N}_u\cup\{u\}}\alpha_{u,u'}^{(k)}\left(\boldsymbol{V}^{(k)}\mathbf{h}_{t-1,u'}^{(k)}\right)$$

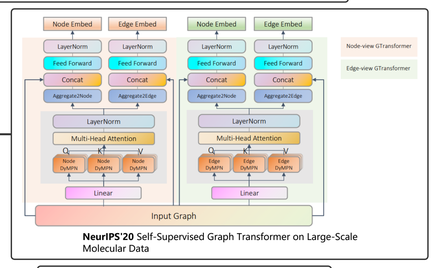
$$\boldsymbol{V}^{(k)} \in \mathbb{R}^{d\times d}$$

$$\alpha_{u,u'}^{(k)} = \text{softmax}\left(\frac{\left(\boldsymbol{Q}^{(k)}\mathbf{h}_{t-1,u}^{(k)}\right)^T\left(\boldsymbol{K}^{(k)}\mathbf{h}_{t-1,u'}^{(k)}\right)}{\sqrt{d}}\right)$$

$$\boldsymbol{Q}^{(k)} \in \mathbb{R}^{d\times d} \text{ and } \boldsymbol{K}^{(k)} \in \mathbb{R}^{d\times d}$$

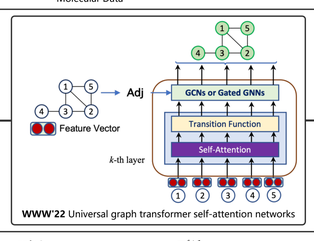
Using global self-attention in conjunction with message-passing GNN

GTransformer



NeurIPS'20 Self-Supervised Graph Transformer on Large-Scale Molecular Data

UGformer Variant 2

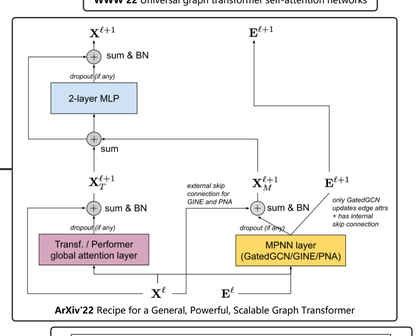


WWW'22 Universal graph transformer self-attention networks

$$\mathbf{H}'^{(k)} = \text{Attention}_{\mathcal{V}}\left(\mathbf{H}^{(k)}\boldsymbol{Q}^{(k)}, \mathbf{H}^{(k)}\boldsymbol{K}^{(k)}, \mathbf{H}^{(k)}\boldsymbol{V}^{(k)}\right) \quad (7)$$

$$\mathbf{H}^{(k+1)} = \text{GNN}\left(\mathbf{A}, \mathbf{H}'^{(k)}\right) \quad (8)$$

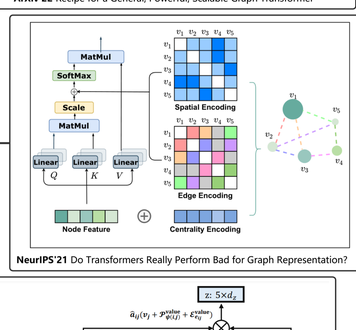
GPS



ArXiv'22 Recipe for a General, Powerful, Scalable Graph Transformer

Injecting edge information into global self-attention via attention bias

Graphormer



NeurIPS'21 Do Transformers Really Perform Bad for Graph Representation?

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

centrality encoding

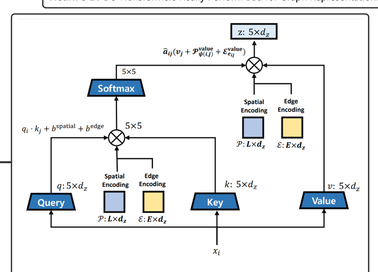
$$A_{ij} = \frac{(h_iW_Q)(h_jW_K)^T}{\sqrt{d}} + b_{\phi(v_i,v_j)}$$

Spatial encoding, ϕ can be connectivity between the nodes in the graph

$$c_{ij} = \frac{1}{N}\sum_{n=1}^{N}x_{e_n}\left(w_n^E\right)^T$$

Edge encoding

GRPE



Graph Relative Positional Encoding

ICLRW'22 GRPE: Relative Positional Encoding for Graph Transformer

Tokenized Graph Transformer (TokenGT)

What is the definition of pure Transformer?

Network architecture compose of

- **Tokenizer:** tokenizes the input data into a set of tokens + **Traditional embedding:** specifies the positional features of each token
- **Multi-head self-attention:** product of the features followed by Softmax
- **Residual feedforward network**

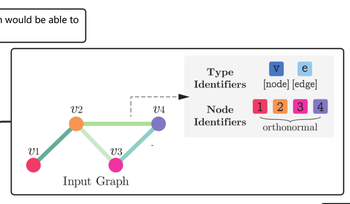
Example: If we add message passing layer in between the self-attention layer, it will not be a pure Transformer

Intuition

- Treat both **node** and **edge** as **token**
- With appropriate **token-wise information**, self-attention would be able to properly recognize the graph structure

TokenGT architecture

Augment token **X** with **node identifiers** and **type identifiers**

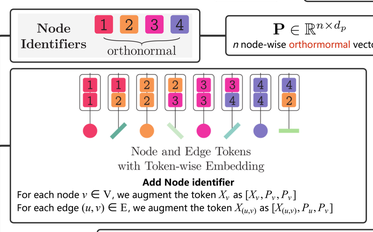


Node and Edge Tokens

- Node Token: $\mathcal{V} = \{v_1, ..., v_n\}$, $\mathbf{X}^{\mathcal{V}} \in \mathbb{R}^{n\times C}$
- Edge Token: $\mathcal{E} = \{e_1, ..., e_m\} \subseteq \mathcal{V}^2$, $\mathbf{X}^{\mathcal{E}} \in \mathbb{R}^{m\times C}$

$$\mathbf{X} = [\mathbf{X}^{\mathcal{V}}; \mathbf{X}^{\mathcal{E}}] \in \mathbb{R}^{(n+m)\times C}$$

Node identifier



$\mathbf{P} \in \mathbb{R}^{n\times d_p}$

n node-wise orthormormal vectors

Add Node identifier

For each node $v \in \mathrm{V}$, we augment the token X_v as $[X_v, P_v, P_v]$
For each edge $(u, v) \in \mathrm{E}$, we augment the token $X_{(u,v)}$ as $[X_{(u,v)}, P_u, P_v]$

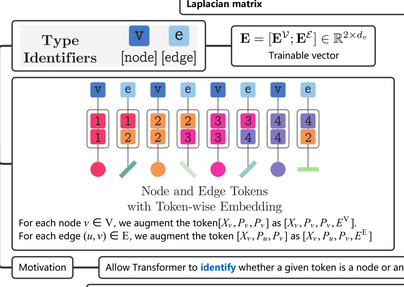
Motivation

- Allows the Transformer to identify and exploit the **connectivity structure** of a graph
- $[\mathbf{P}_u, \mathbf{P}_v][\mathbf{P}_k, \mathbf{P}_k]^\top = 1$ If $k \in (u, v)$. Otherwise the result will be 0

Implementation choices of **P**

- **Orthogonal random features**
- **Laplacian eigenvectors** obtained from eigendecomposition of **graph Laplacian matrix**

Type identifier



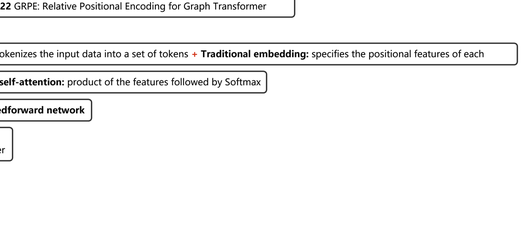
$\mathbf{E} = [\mathbf{E}^{\mathcal{V}}; \mathbf{E}^{\mathcal{E}}] \in \mathbb{R}^{2\times d_e}$

Trainable vector

For each node $v \in \mathrm{V}$, we augment the token$[X_v, P_v, P_v]$ as $[X_v, P_v, P_v, E^{\mathrm{V}}]$.
For each edge $(u, v) \in \mathrm{E}$, we augment the token $[X_v, P_u, P_v]$ as $[X_v, P_u, P_v, E^{\mathrm{E}}]$

Motivation: Allow Transformer to **identify** whether a given token is a node or an edge

Overall architecture of TokenGT



Input

- $\mathbf{X}^{in} \in \mathbb{R}^{(n+m)\times(C+2d_p+d_e)}$
- $w^{in} \in \mathbb{R}^{(C+2d_p+d_e)\times d}$ Projection matrix

$\mathbf{X}^{in}w^{in}$

Similar to BERT and ViT, they prepend a special trainable token [graph] with trainable embedding $\mathbf{v}_{[\text{graph}]} \in \mathbb{R}^d$

$$\mathbf{Z}^{(0)} = [\mathbf{X}_{[\text{graph}]}; \mathbf{X}^{in}w^{in}] \in \mathbb{R}^{(1+n+m)\times d}$$

Final input to the main encoder

Experimental Results

method	# parameters	validate MAE ↓	asymptotics
Message-passing GNNs			
GCN [26]	2.0M	0.1379	$\mathcal{O}(n+m)$
GIN [26]	3.8M	0.1195	$\mathcal{O}(n+m)$
GAT	6.7M	0.1302	$\mathcal{O}(n+m)$
GCN-VN [26]	4.9M	0.1153	$\mathcal{O}(n+m)$
GIN-VN [26]	6.7M	0.1083	$\mathcal{O}(n+m)$
GAT-VN	6.7M	0.1192	$\mathcal{O}(n+m)$
GAT-VN (large)	55.2M	0.1361	$\mathcal{O}(n+m)$
Transformers with strong graph-specific modifications			
Graphormer [52]	48.3M	0.0864	$\mathcal{O}(n^2)$
EGT [28]	89.3M	0.0869	$\mathcal{O}(n^2)$
GRPE [47]	46.2M	0.0890	$\mathcal{O}(n^2)$
Pure Transformers			
Transformer	48.5M	0.2340	$\mathcal{O}((n+m)^2)$
TokenGT (ORF)	48.6M	0.0962	$\mathcal{O}((n+m)^2)$
TokenGT (Lap)	48.5M	0.0910	$\mathcal{O}((n+m)^2)$
TokenGT (Lap) + Performer	48.5M	0.0935	$\mathcal{O}(n+m)$