Progess of the Project

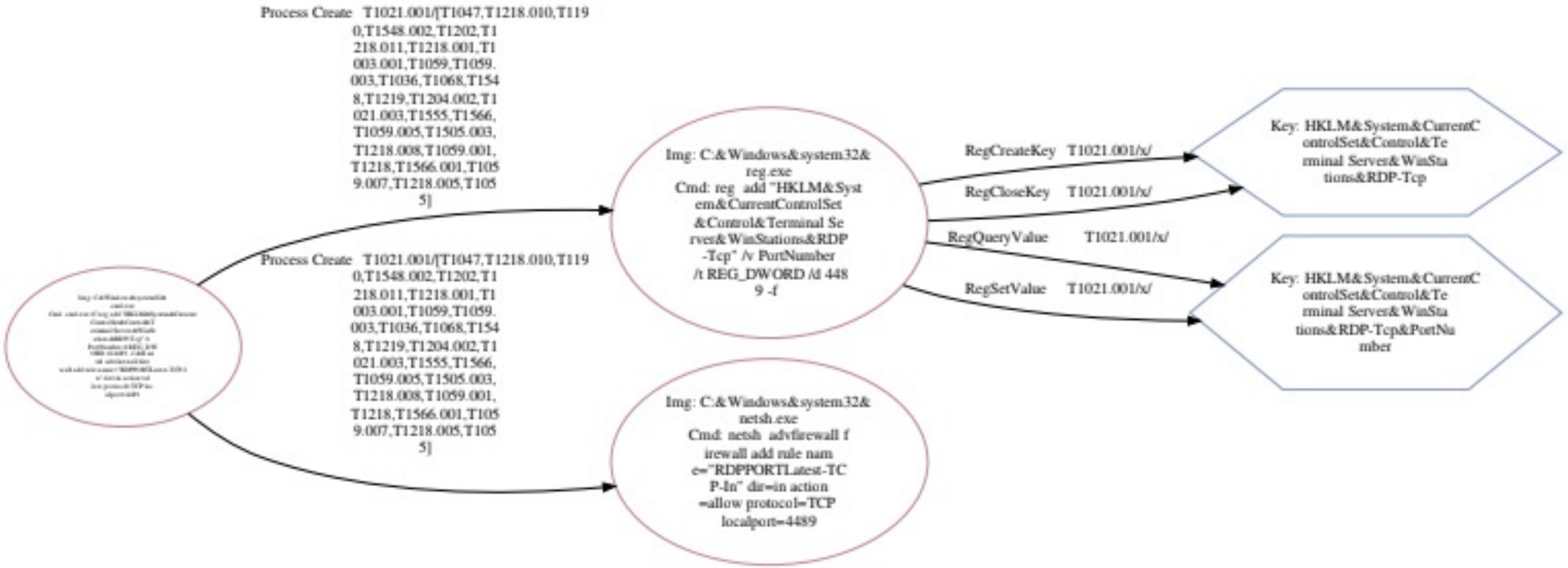
Vincent Pai 2023/7/19

Outline

- **Graph Classification**
    - Model
    - Background
    - Architecture
    - Input Format
    - Possible Issue
- **Future Plan**

Graph Classification

My task



Considering:

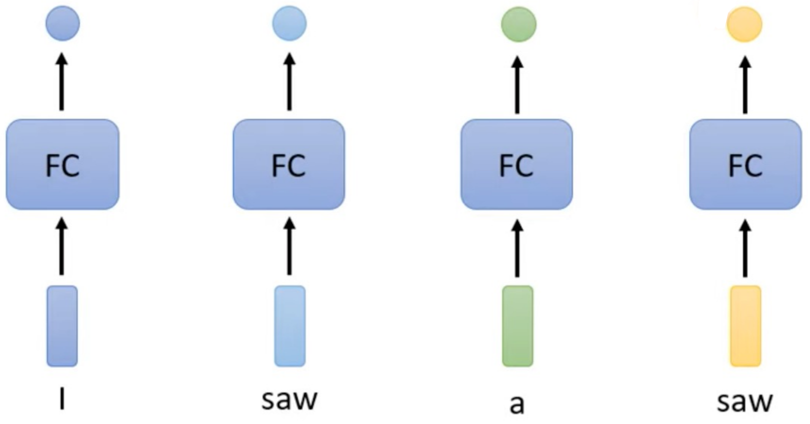
1. Sequence
2. Multi-relation
3. Different destinaiton nodes come from same source node

Model - Graphormer

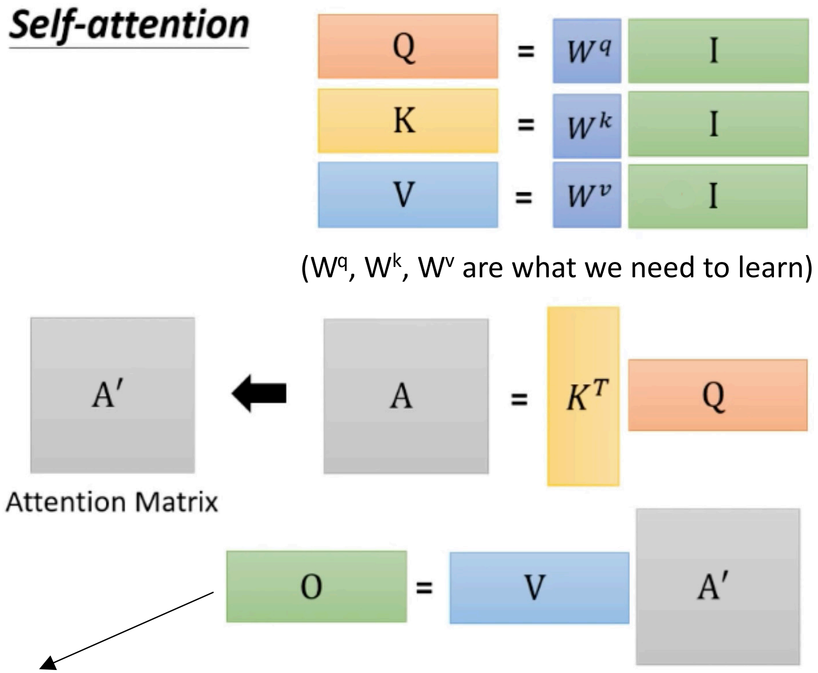
- Published by *Microsoft*
- Paper: *Do Transformers Really Perform Bad for Graph Representation*
- Author: *Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, Tie-Yan Liu from Microsoft Research Asia*
- Published at: 2016 arXiv

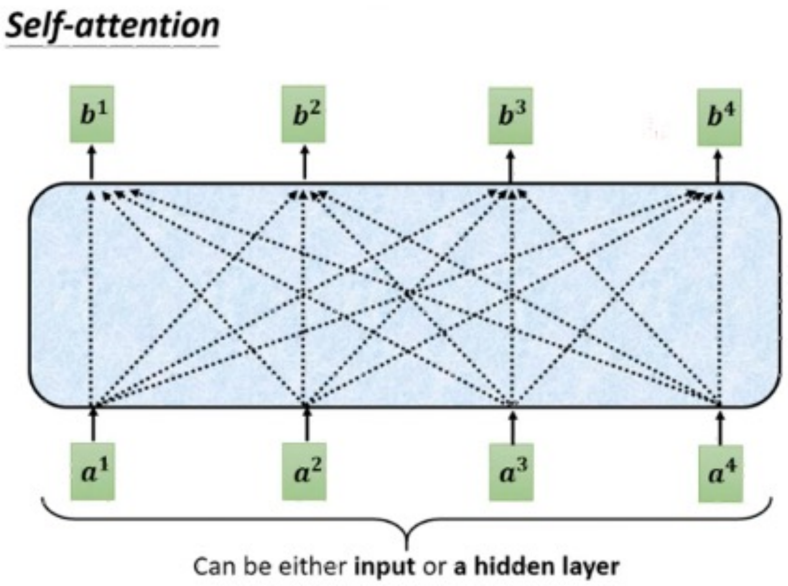
- They want to apply **Transformer** in the realm of the graph, and in the past, the only effective way is to replace some key modules (e.g., feature aggregation) in classic GNN variants by the softmax attention.

Background – Self-attention



- Can do the sequence labeling task:
  - Since considering the context (whole sequence)
  - It can be apply on our task: **considering the whole graph(causility)**
  - Steps:

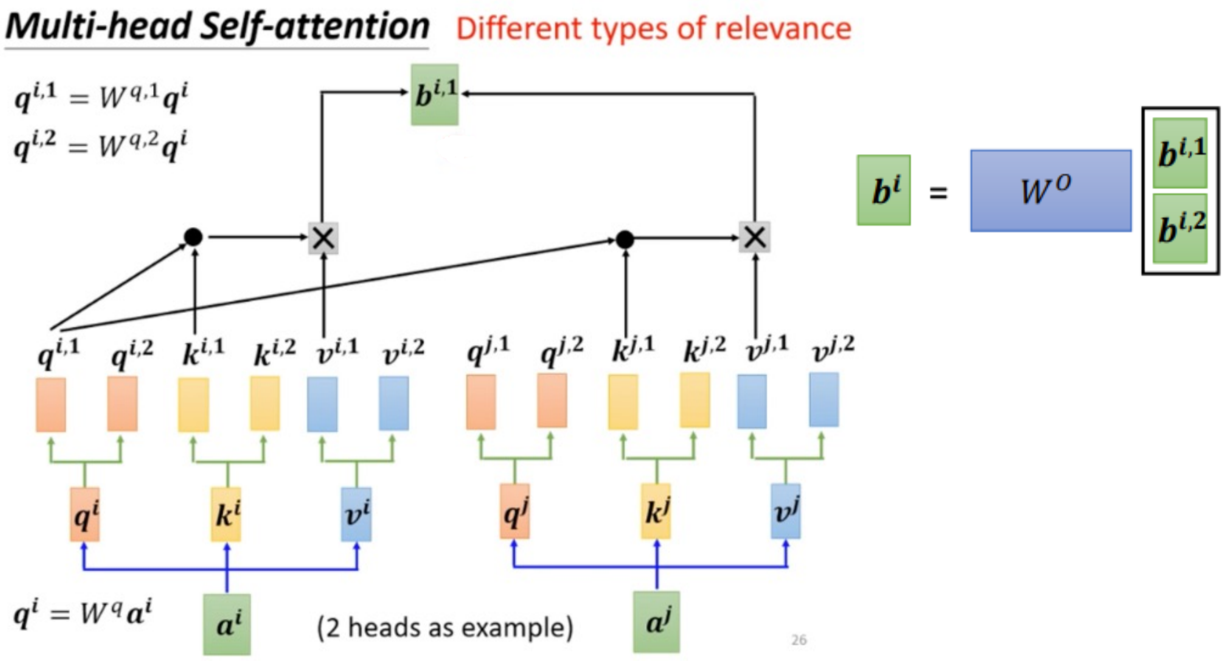


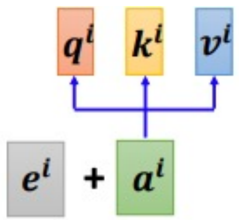


Attention matrix: we'll extract information based on attention scores

Background – Self-attention

- **Multi-head Self-attetnion:**
    - Each head has their own concern
    - They can learn more details
    - Can be applied on our task: **multi-relation**

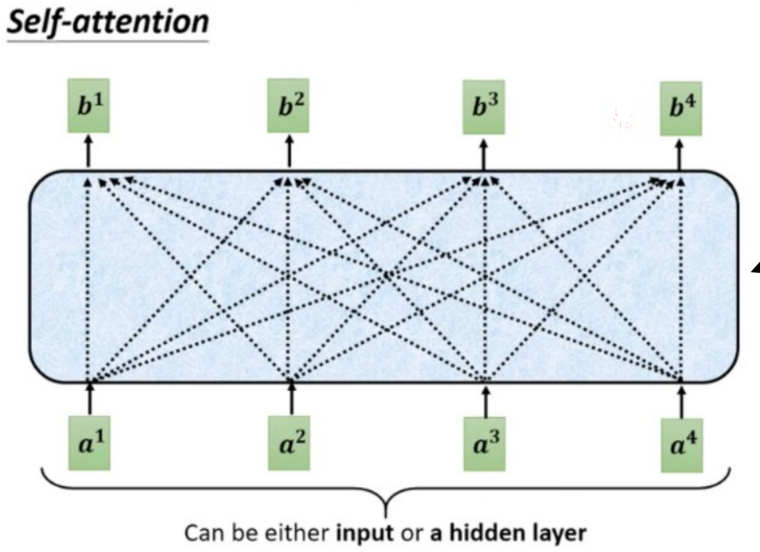


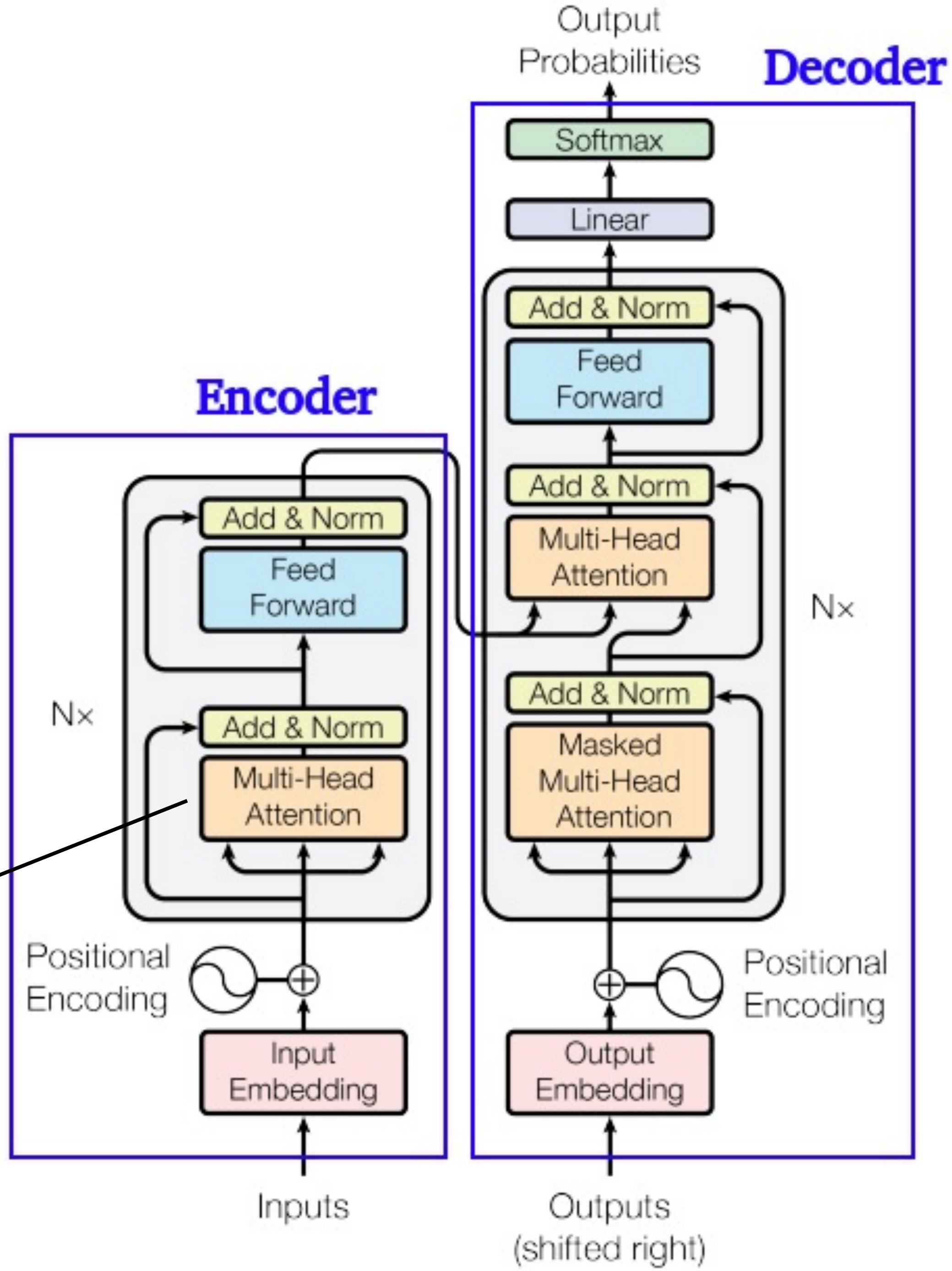


- **Positional Encoding:**
    - Every input is the same position to the self-attention – 天涯若比鄰
    - If the position is important, use positional encoding
    - In *Attention is all you need*: they use sin and cos function to get the positional vector

Background - Transformer

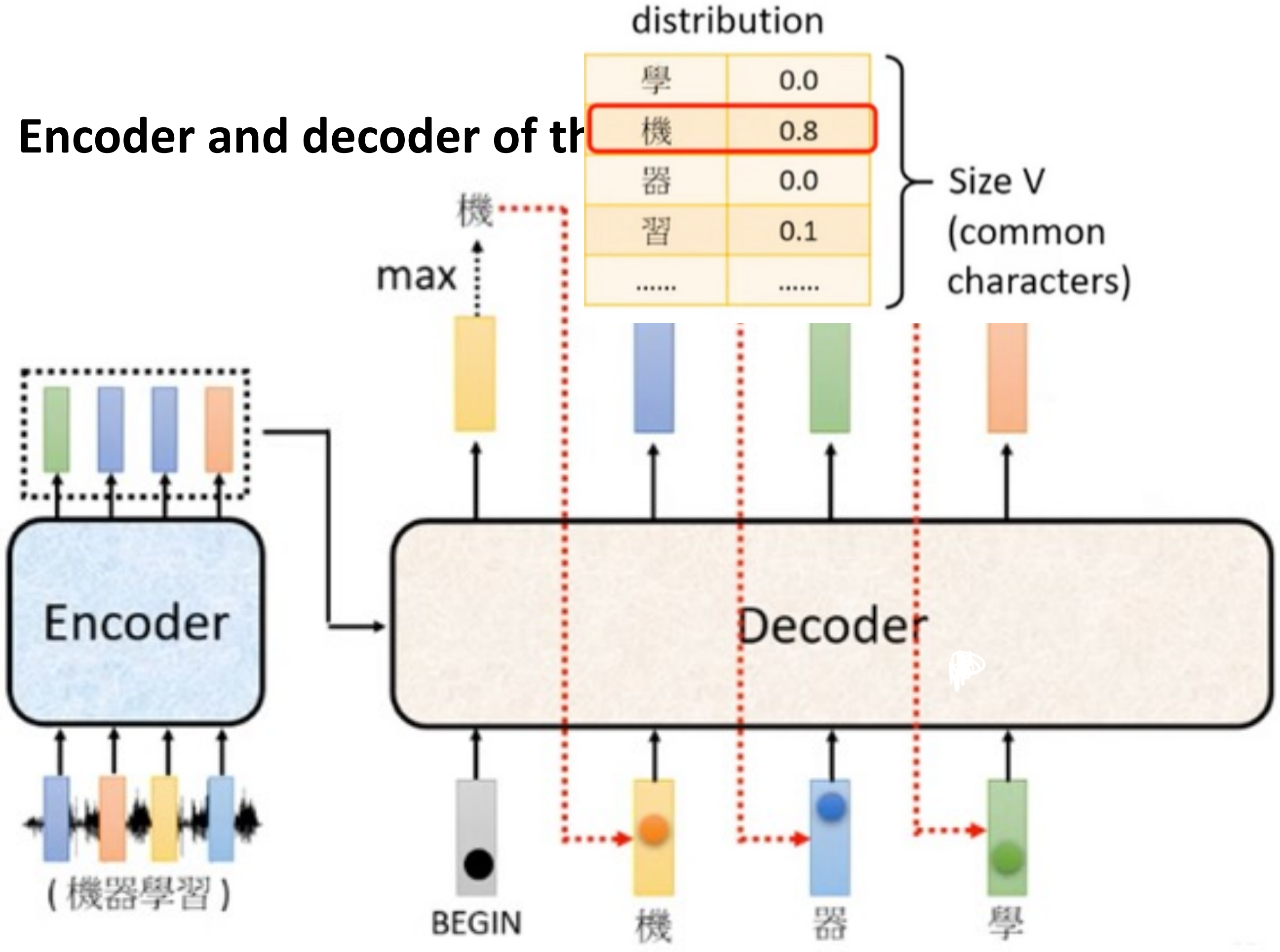
- **Encoder**: input a sequence of vectors and output is a sequence of vectors too.
- **Decoder**: output is a set of probabilities.
- Positional Encoding: positonal information



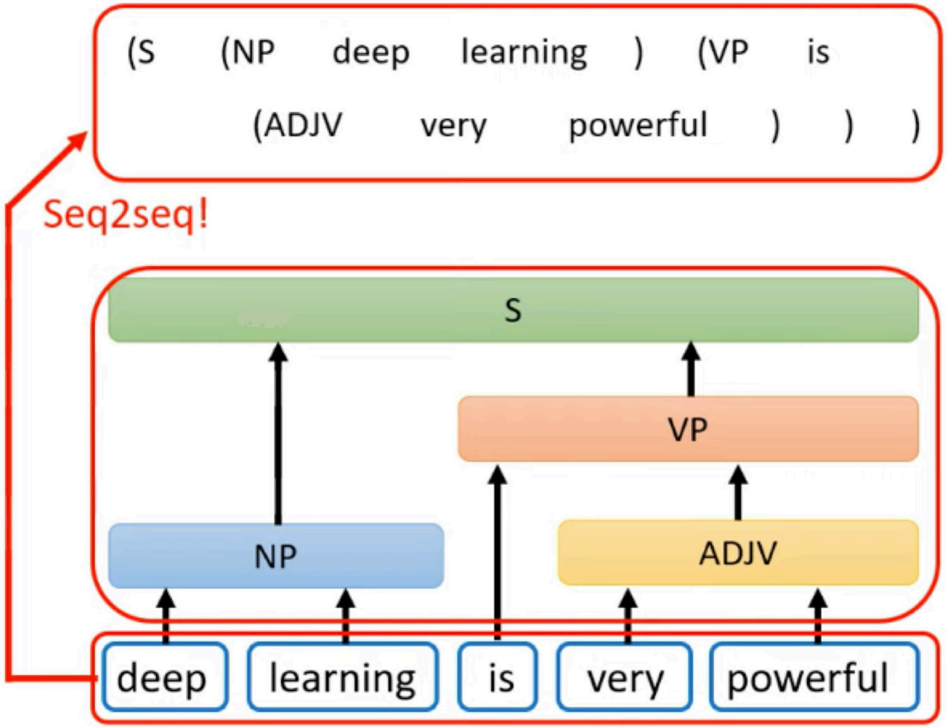


Background - Transformer

Encoder and decoder of th



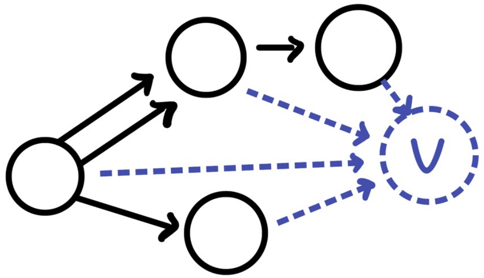
Related task it can do:



Architecture

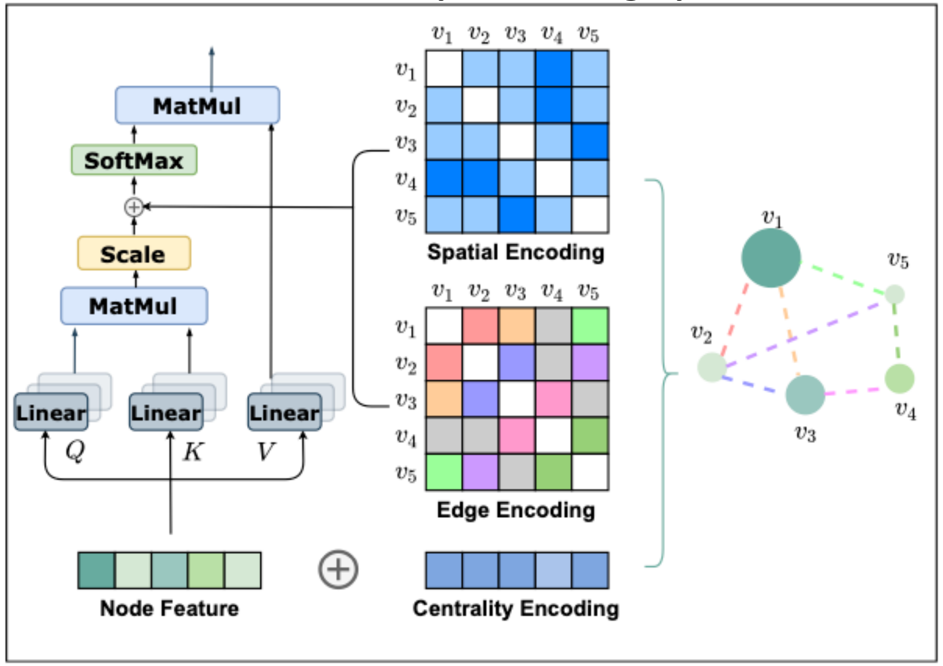
- Directly built on the classic architecture of **Transformer**
- **Graphormer Layer:**
  - Change multi-head self-attention(MHA) part of the Transformer
  - Apply the layer normalization **before** applying **multi-head self-attention** and the feed-forward blocks instead of **after**.
  - Having some special structural encoding

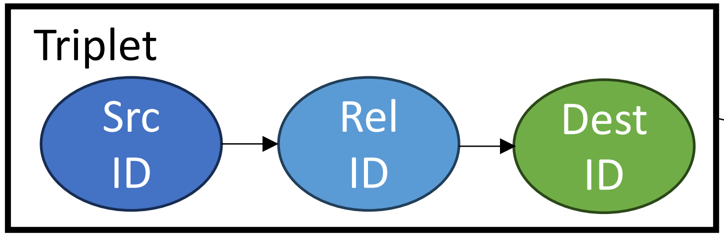
- **Virtual Node:**
  - Connect to **every node** in the graph
  - Representation of the **whole graph** would be the node feature of the **[VNode]** in the final layer



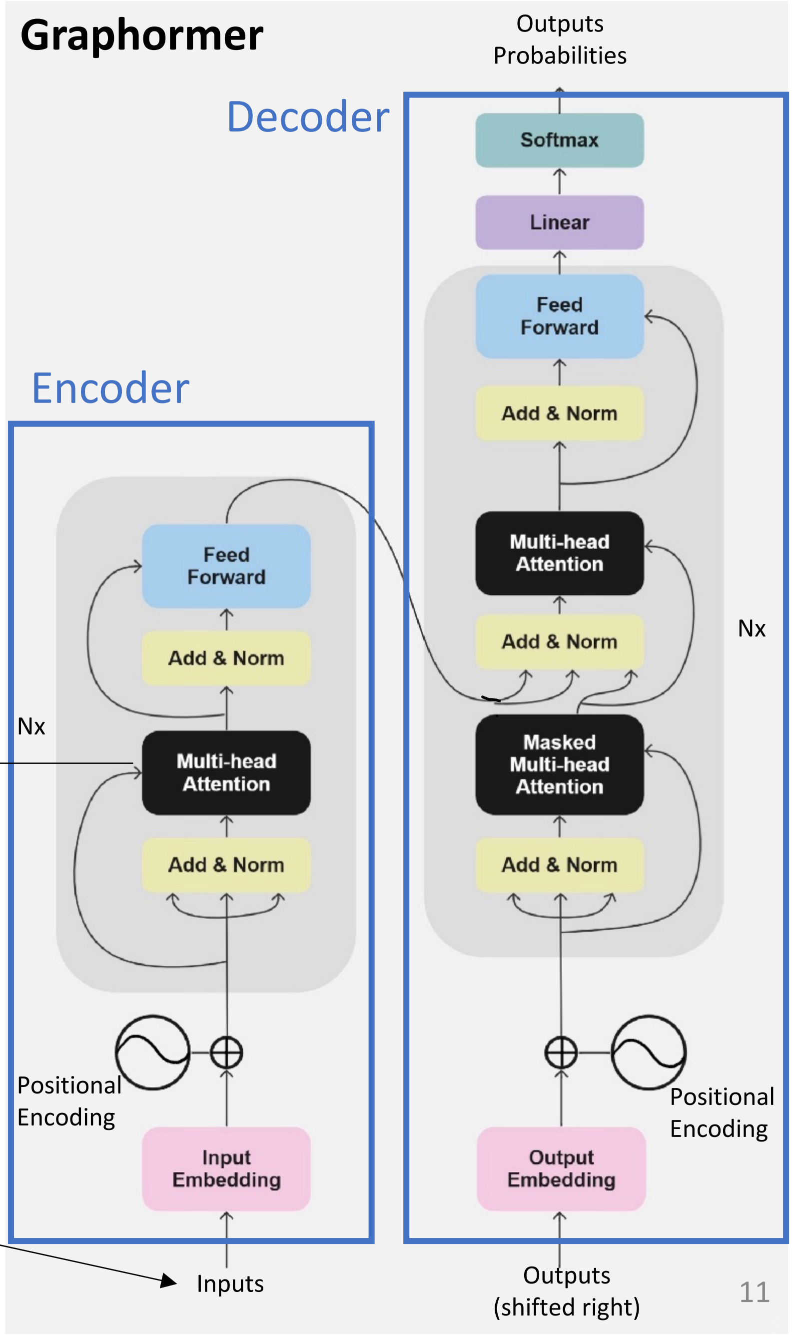
Architecture

Multi-head Self-attention part of the graphormer

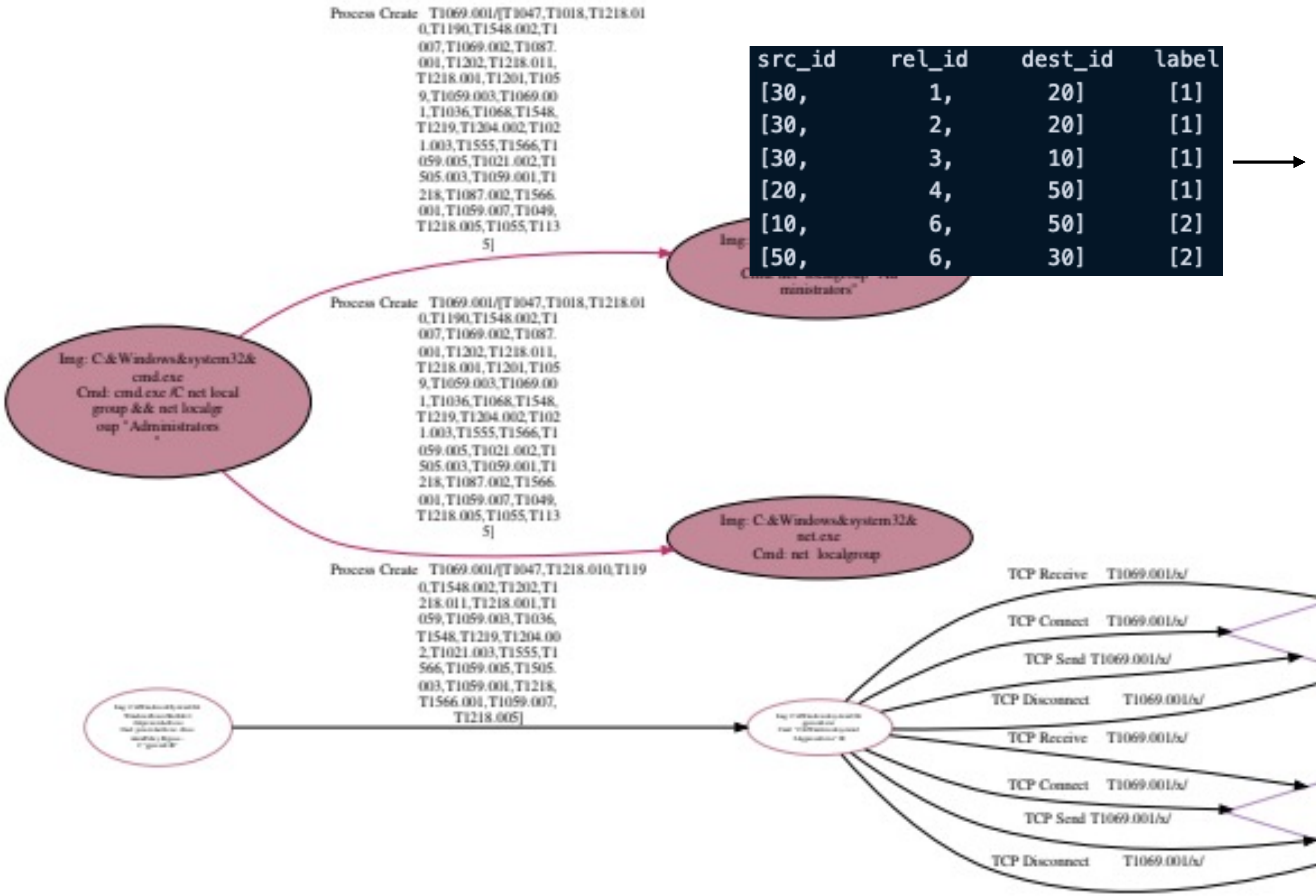
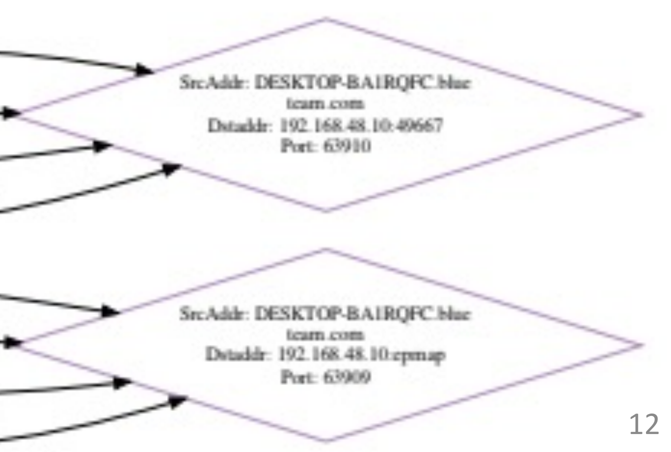




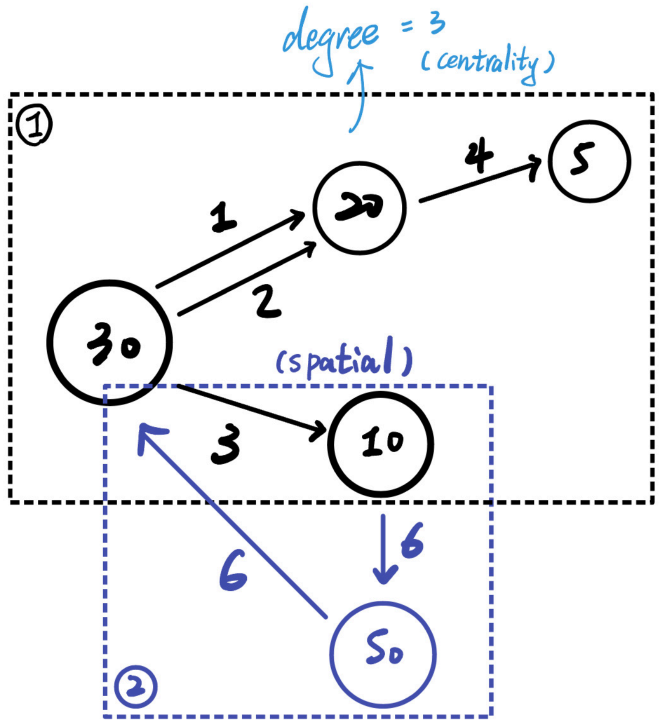
Triplet x 4932605



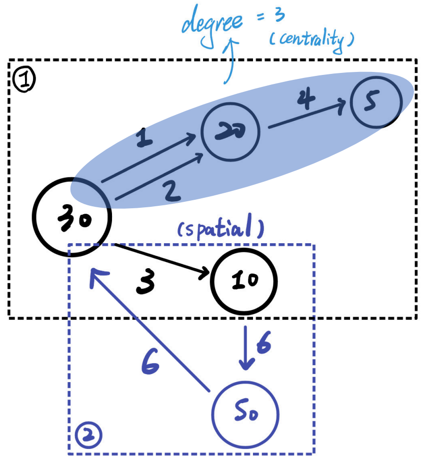
Architecture – Graph

src_id	rel_id	dest_id	label
[30,	1,	20]	[1]
[30,	2,	20]	[1]
[30,	3,	10]	[1]
[20,	4,	50]	[1]
[10,	6,	50]	[2]
[50,	6,	30]	[2]



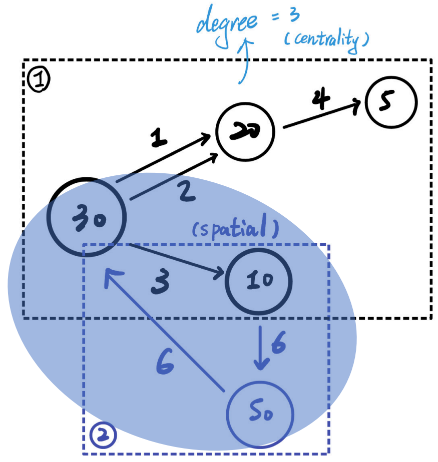
Architecture – Centrality Encoding



- Based on the **degree** of each node and add them to the node inputs
- Capture both **semantic correlation** and the **node importance**.

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

Architecture – Spatial Encoding

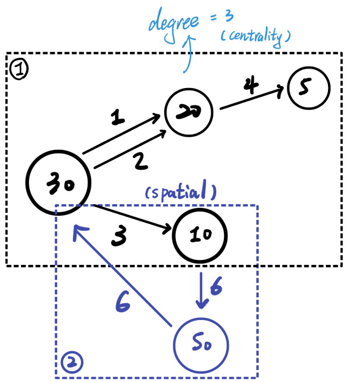


- Consider the multi-dimension or non-sequence case → **Graph**
- Ø(vi, vj) is defined as the SPD(shortest path distance) of vi and vj
  - If not connected → set to -1
  - Be a **bias term** of the attention module
- **Adaptively** attend to all other nodes according to the graph structure

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

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

Architecture – Edge Encoding



- In many case, edges also have structural features
  - E.g., in molecular graph, atom pairs may have some features
- Compute the **average** of the **dot-products** of the edge features and a learnable embedding along the path
- A **bias tram** of the attention module

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

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.

Input Format

- A **jsonl** file:

edge_index (sequence)	edge_attr (sequence)	y (sequence)	num_nodes (int64)	node_feat (sequence)
[[0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7,…	[[0, 0, 1], [0, 0, 1], [3, 0, 1], [3, 0, …	[0]	24	[[6, 0, 3, 5, 2, 0, 1, 0, 0], [5,…
[[0, 1, 1, 2, 1, 3, 1, 4, 4, 5, 5, 6, 6, 7, 6,…	[[1, 0, 0], [1, 0, 0], [1, 0, 0], [1, 0, …	[0]	10	[[7, 0, 1, 5, 0, 0, 1, 0, 0], [15…

- **Edge_index:** contains the indices of nodes in edges, stored as a list containing two parallel lists of edge indices `edge_index = [[1,2,1], [2,3,3]]`
- **Labels:** list or an integer contain the corresponding techniques
- **Nodes_nums:** total number of the nodes
- **Node_feat:** contains the available features of each node (if present)
- **Edge_feat:** contains the available features of each edge (if present)

Possible Issue

- Graphormer is more easily trapped in the **over-fitting** problem due to the **large** size of the model and the **small** size of the dataset.
- Therefore, we may need to employ a widely used **data augmentation** for graph - **FLAG** to mitigate the over-fitting problem on OGB datasets.
  - Paper: *Adversarial Data Augmentation for Graph Neural Networks*

Future Plan

Plan of Next Week

- For **TRAM**
  - Try to use the real dataset to upload and then labeled them

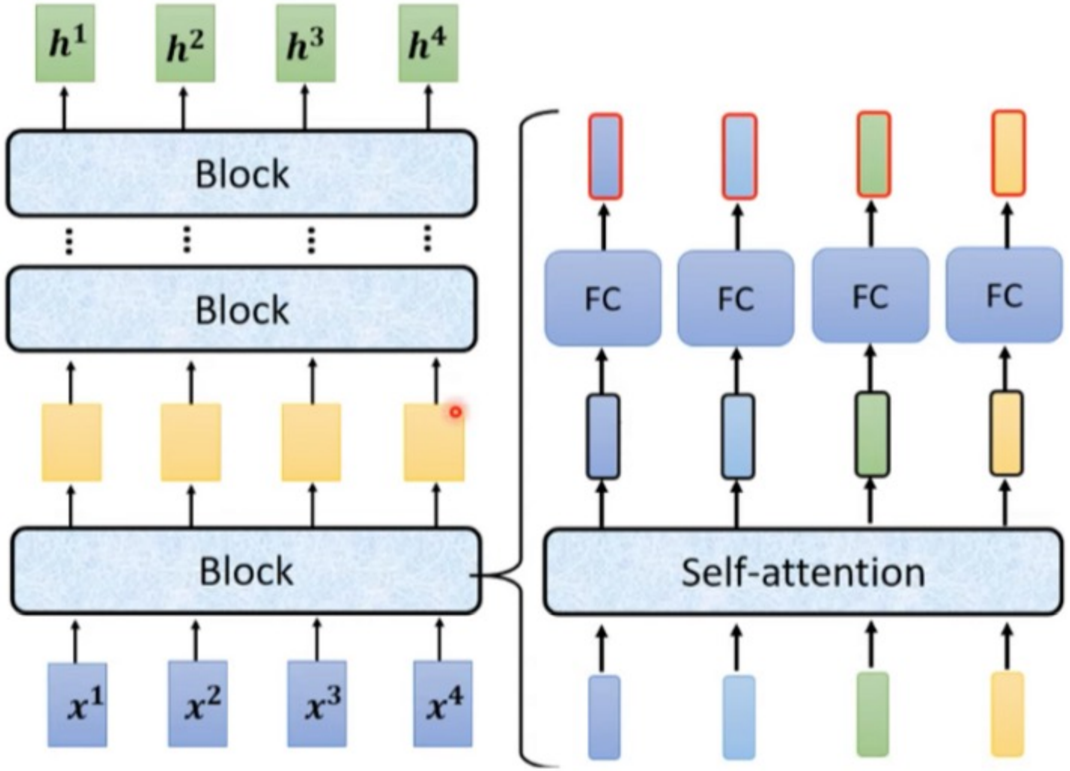
- For **Graphormer**
  - Try to successfully input the data(jsonl format)
  - Try to implement or use the simplest model to train
  - If needed, try the data augmentation - FLAG

  - if Graphormer is not feasible, try some more models

Appendix

Transformer -encoder

3. Add & Norm（[3]殘差連接residual connection）：把Multi-head attention的input a和output b加起來得到b'，再做[1] Layer Normalization
4. 計算完後丟到前向傳播，再經過一個Add & Norm



Others may be useful

- **Graph Transformer** – improvement of the GNN
- **Graph attention network (GAT)** for node classification
- **Multilabel graph classification** using GAT

- MULTIHEADATTENTION
- Self-attention does not need O(n^2) memory

Self-attention does not need O(n^2) memory

- Only need O(log n) space complexity (usually considered to be O(n^2))

Sequence length	$n = 2^8$	2^{10}	2^{12}	2^{14}	2^{16}	2^{18}	2^{20}
Size of inputs and outputs	160KB	640KB	2.5MB	10MB	40MB	160MB	640MB
Memory overhead of standard attention	270KB	4.0MB	64MB	1GB	OOM	OOM	OOM
Memory overhead of memory-eff. attn.	270KB	4.0MB	16MB	17MB	21MB	64MB	256MB
Compute time on TPUv3	0.06ms	0.11ms	0.7ms	11.3ms	177ms	2.82s	45.2s
Relative compute speed	$\pm 5\%$	$\pm 5\%$	$-8 \pm 2\%$	$-13 \pm 2\%$	-	-	-

Table 2: Memory and time requirements of self-attention during **inference**.

Self-attention does not need O(n^2) memory

- https://arxiv.org/pdf/2112.05682.pdf
- https://github.com/google-research/google-research/blob/master/memory_efficient_attention/memory_efficient_attention.ipynb