Graph Neural Networks

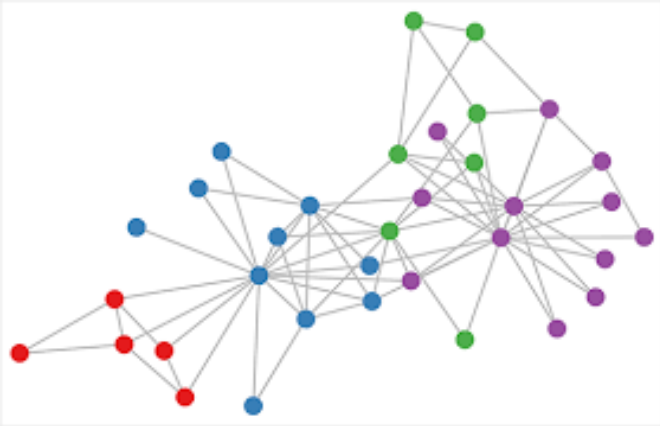
19 Jul 2022

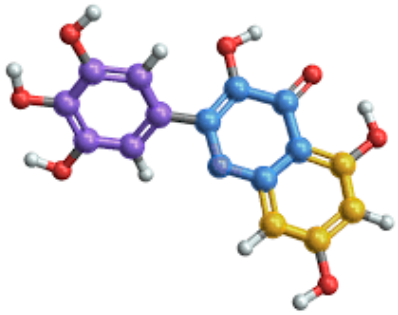
John Tan Chong Min

Background

- Most data is in the form of a graph

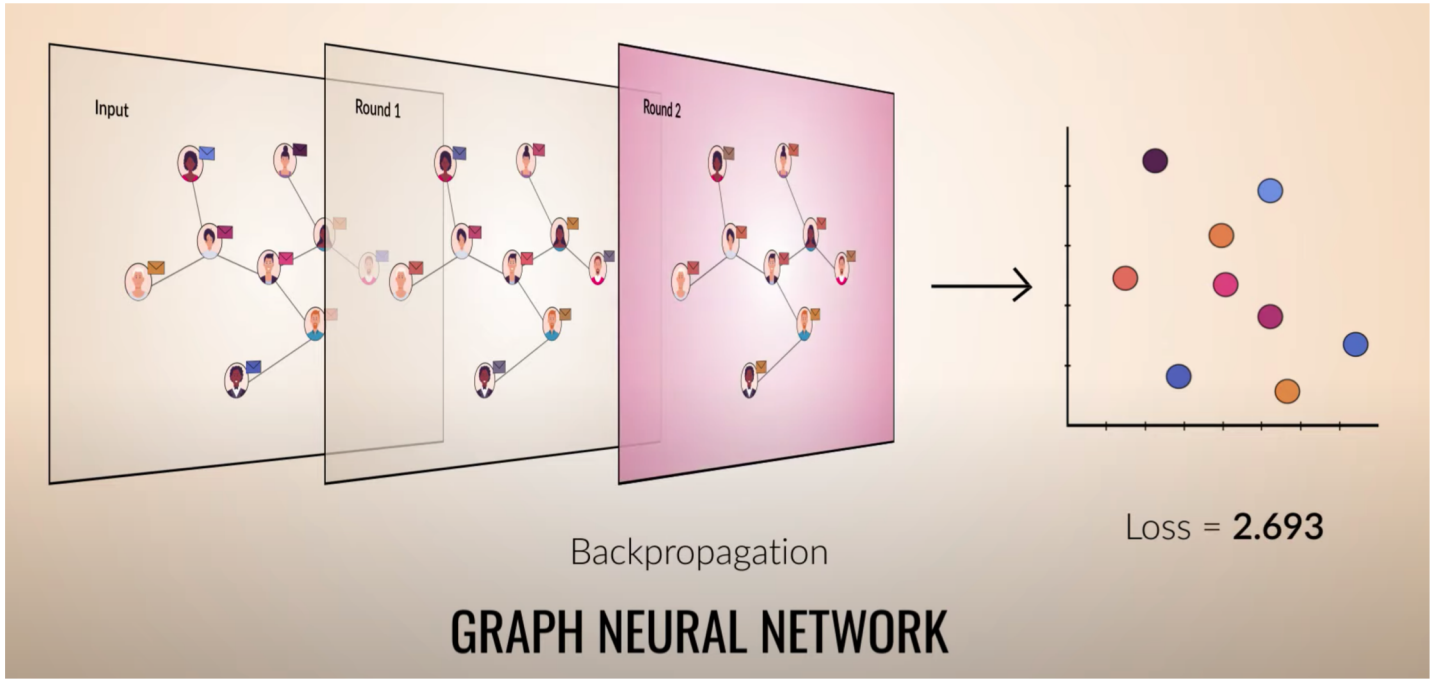
- Would be good if we could iterate over the connections of a graph, to extract the most meaning out of it





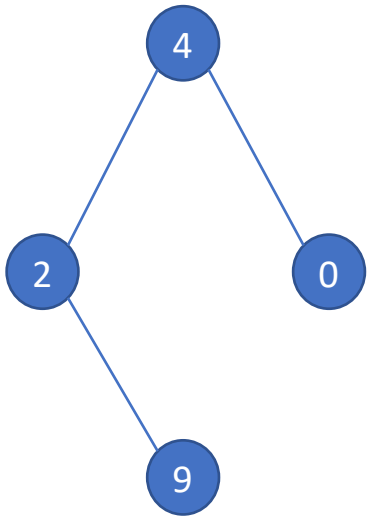
Video

- https://www.youtube.com/watch?v=GXhBEj1ZtE8

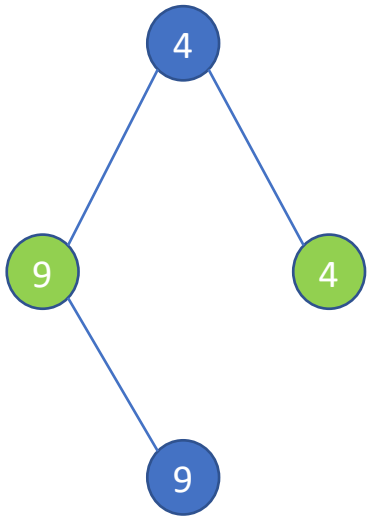


Neural Network to find maximum node value

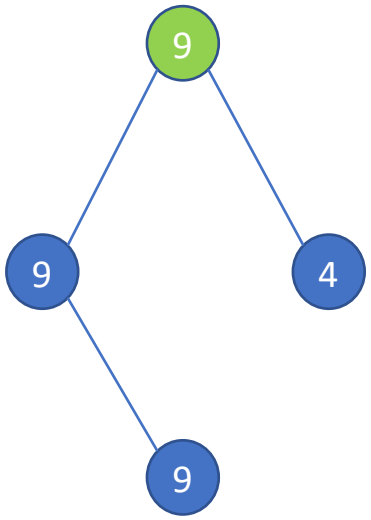
Step 1



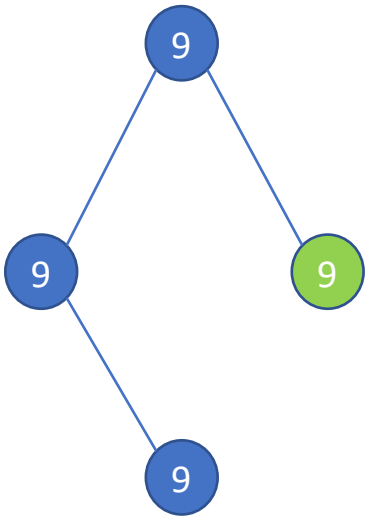
Step 2



Step 3



Step 4



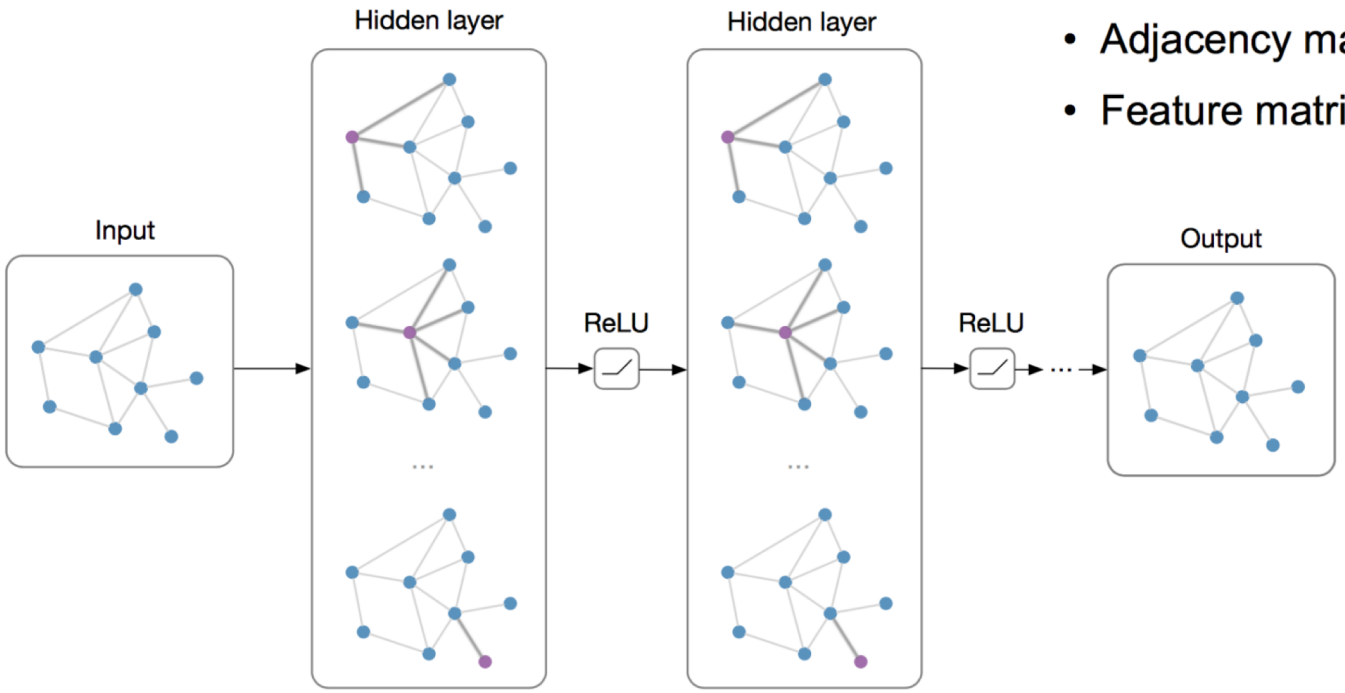
Brief Overview of GNNs

Slides from Thomas Kipf, University of Amsterdam

Graph Neural Network (Overall)

Notation: $\mathcal{G} = (\mathbf{A}, \mathbf{X})$

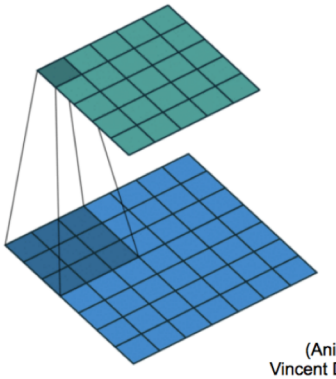
- Adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$
- Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times F}$



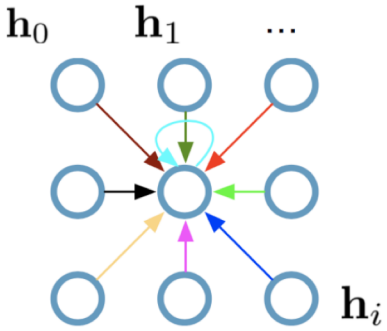
Main idea: Pass messages between nodes and update the node's information

Convolutional Neural Network (CNN)

Single CNN layer with 3x3 filter:



(Animation by Vincent Dumoulin)



Update for a single pixel:

- Transform messages individually $\mathbf{W}_i \mathbf{h}_i$
- Add everything up $\sum_i \mathbf{W}_i \mathbf{h}_i$

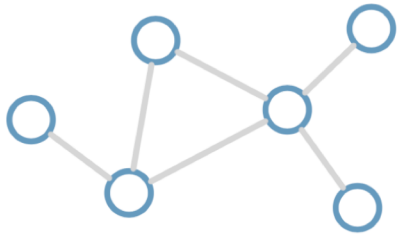
$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

Full update:

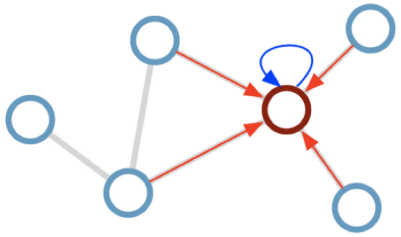
$$\mathbf{h}_4^{(l+1)} = \sigma\left(\mathbf{W}_0^{(l)}\mathbf{h}_0^{(l)} + \mathbf{W}_1^{(l)}\mathbf{h}_1^{(l)} + \cdots + \mathbf{W}_8^{(l)}\mathbf{h}_8^{(l)}\right)$$

Graph Convolutional Networks (GCN)

Consider this undirected graph:



Calculate update for node in red:



Update rule:

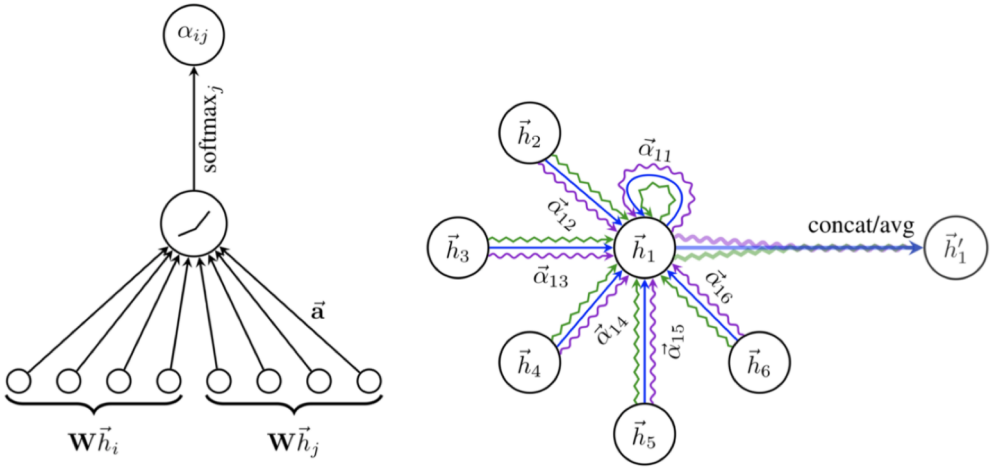
$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

Scalability: subsample messages [Hamilton et al., NIPS 2017]

$\mathcal{N}_i$: neighbor indices

c_{ij} : norm. constant (fixed/trainable)

Graph Attentional Network (GAT)

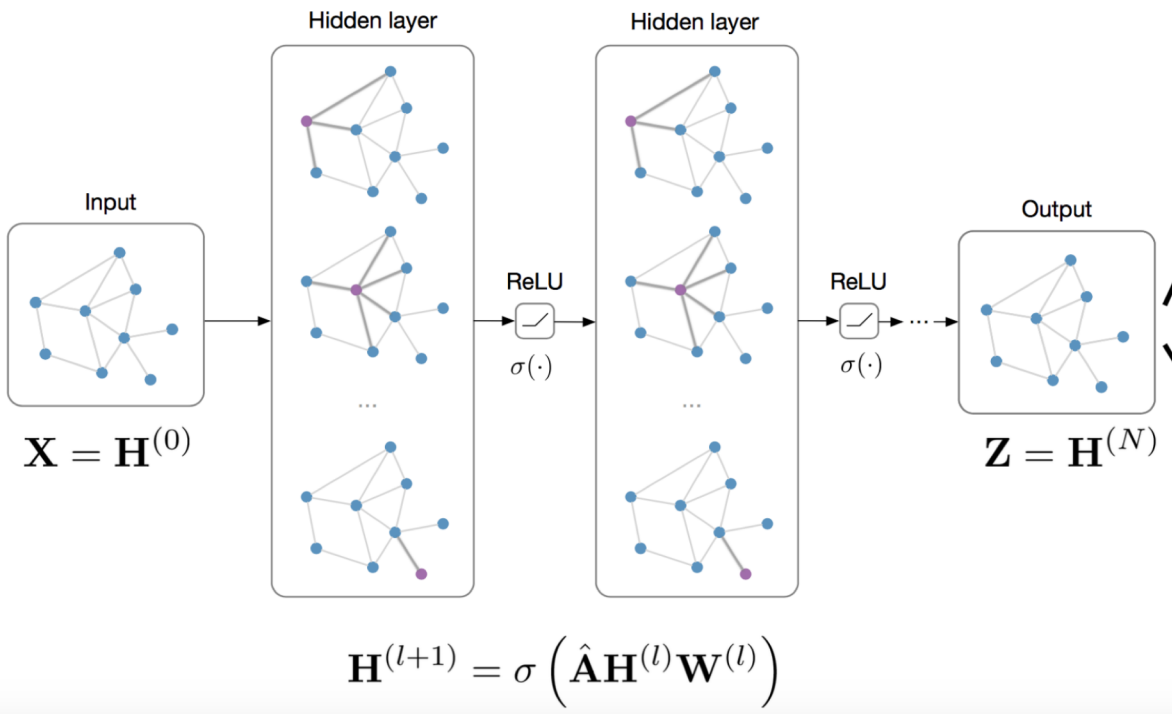


[Figure from Veličković et al. (ICLR 2018)]

$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k\mathbf{W}^k\vec{h}_j\right) \qquad \alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i\|\mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i\|\mathbf{W}\vec{h}_k]\right)\right)}$$

End-to-end GNN

Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



Node classification:

$\mathrm{softmax}(\mathbf{z_n})$

e.g. Kipf & Welling (ICLR 2017)

Graph classification:

$\mathrm{softmax}(\sum_n \mathbf{z_n})$

e.g. Duvenaud et al. (NIPS 2015)

Link prediction:

$p(A_{ij}) = \sigma(\mathbf{z_i^T z_j})$

Kipf & Welling (NIPS BDL 2016)

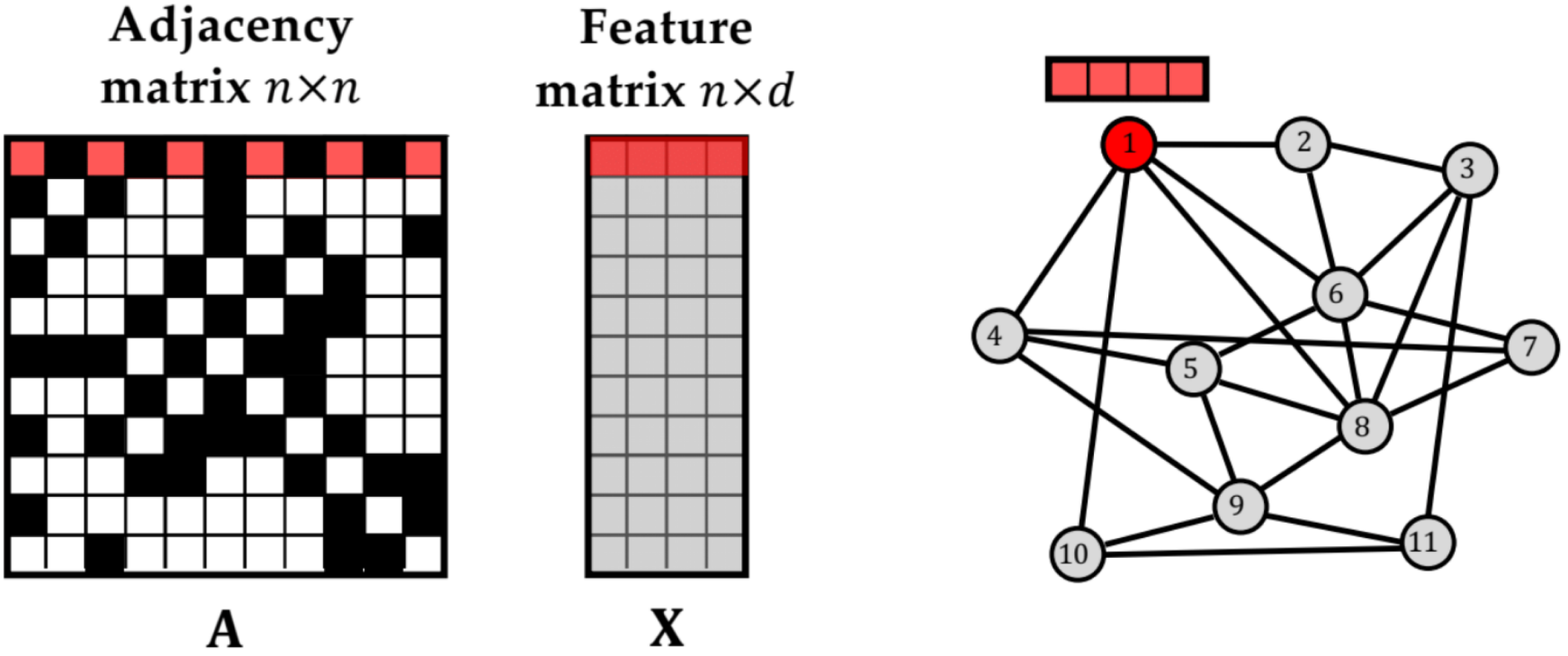
"Graph Auto-Encoders"

Properties of GNN

Permutation Invariance

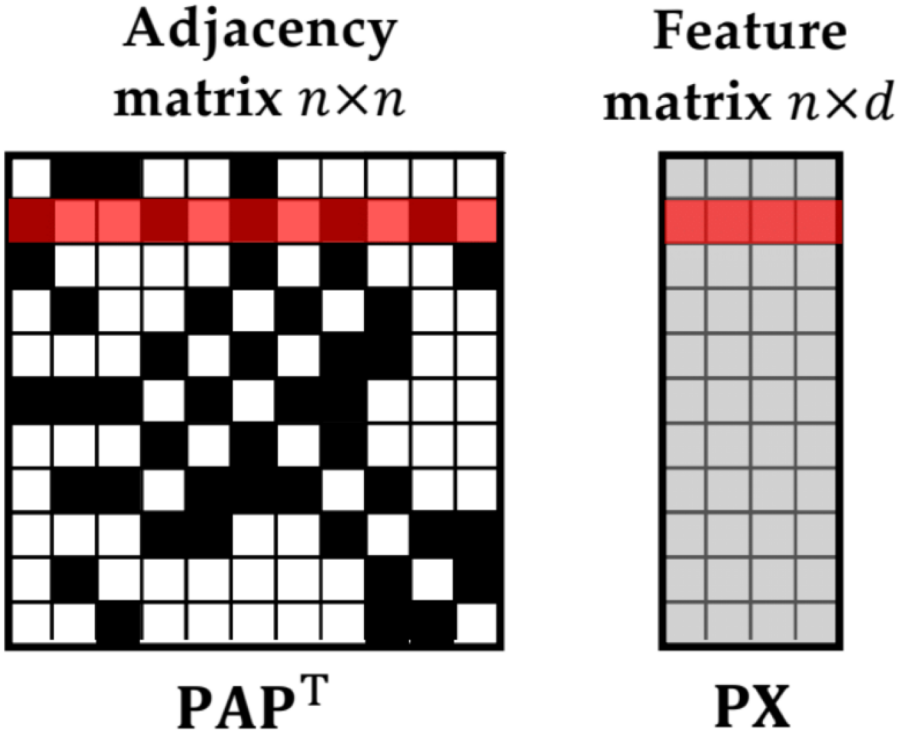
Slides from Geometric Deep Learning Lectures

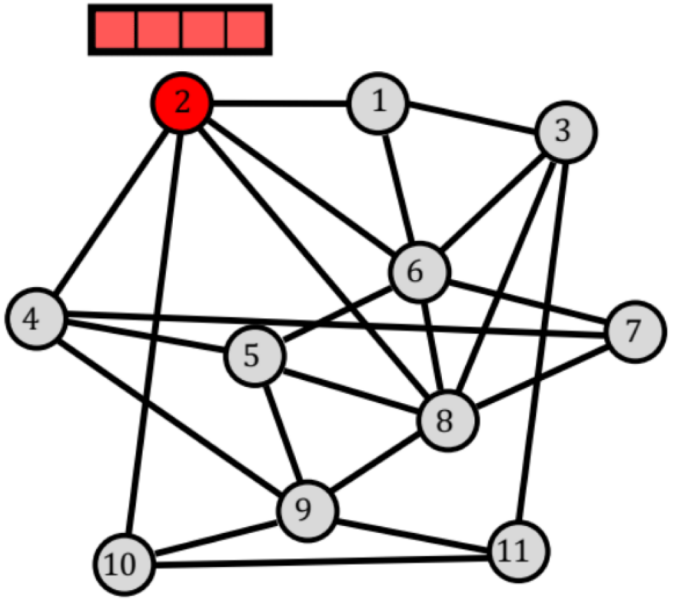
Graph Illustrated



Note: Numbering of nodes is arbitrary.

Graphs can be permuted arbitrarily





P: Permutation Matrix

Transformers

As a form of Attentional GNN

Complete Edge Set (aka Transformers)

- Links between each element dependent on the element itself
- Transformers can be viewed as inferring soft adjacency
  - Letting GNN choose its own edges

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

Self-attention, typically constrained in the range [0, 1] using softmax

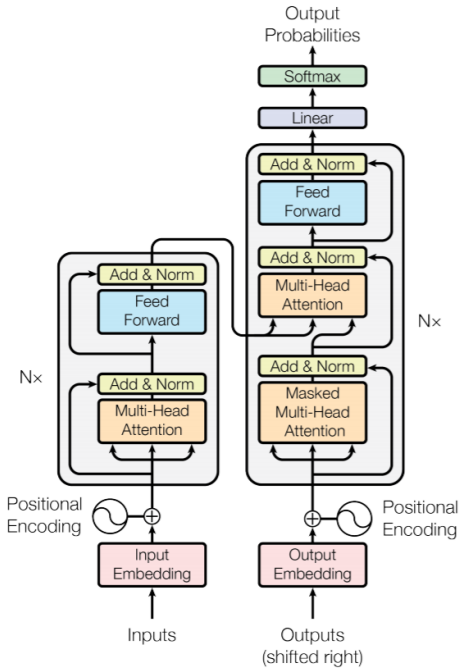
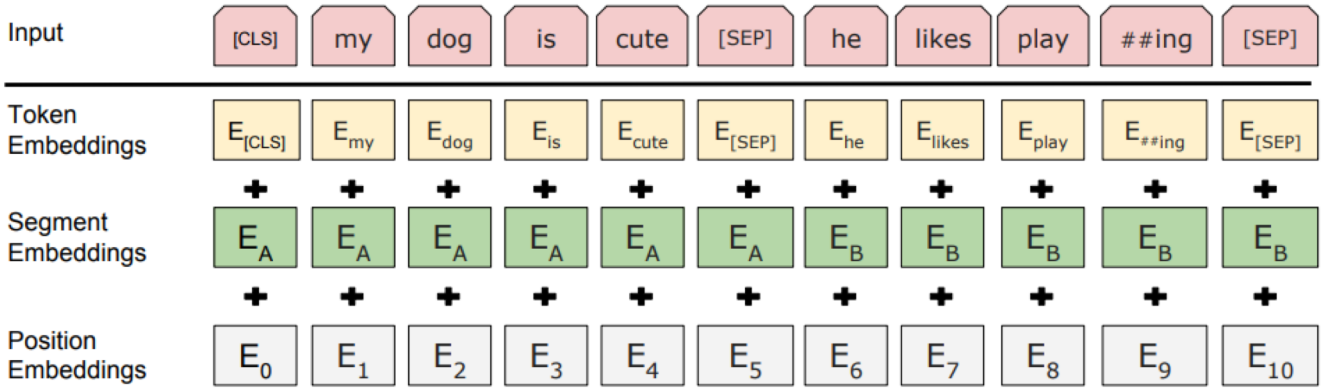


Figure 1: The Transformer - model architecture.

Where is the sequence info in Transformers?

- If Transformers are purely attentional GNNs over complete graph, there would be no information about sequence position

- This was explicitly given as an input to the model via positional embeddings



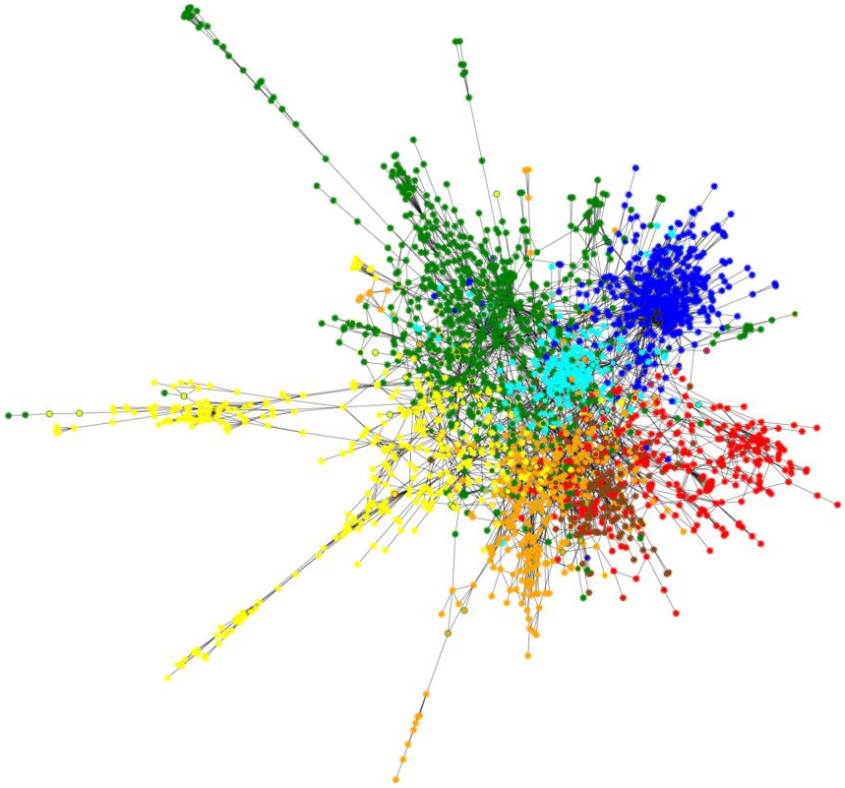
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Devlin et al. (2018).

Graph Attention Network

On Cora dataset

Cora Dataset

- 2708 scientific publication
  - Grouped into 7 classes
- 5429 citation links total
  - Paper A (source) citing Paper B (target)
- Features of each publication is 1433 unique 0/1-valued word vector
  - Indicates presence of a particular word



Discussion