GRAPH ATTENTION NETWORKS

Authors: Petar Veličković, Guillem Cucurull,
Arantxa Casanova, Adriana Romero, Pietro Liò, Yoshua Bengio

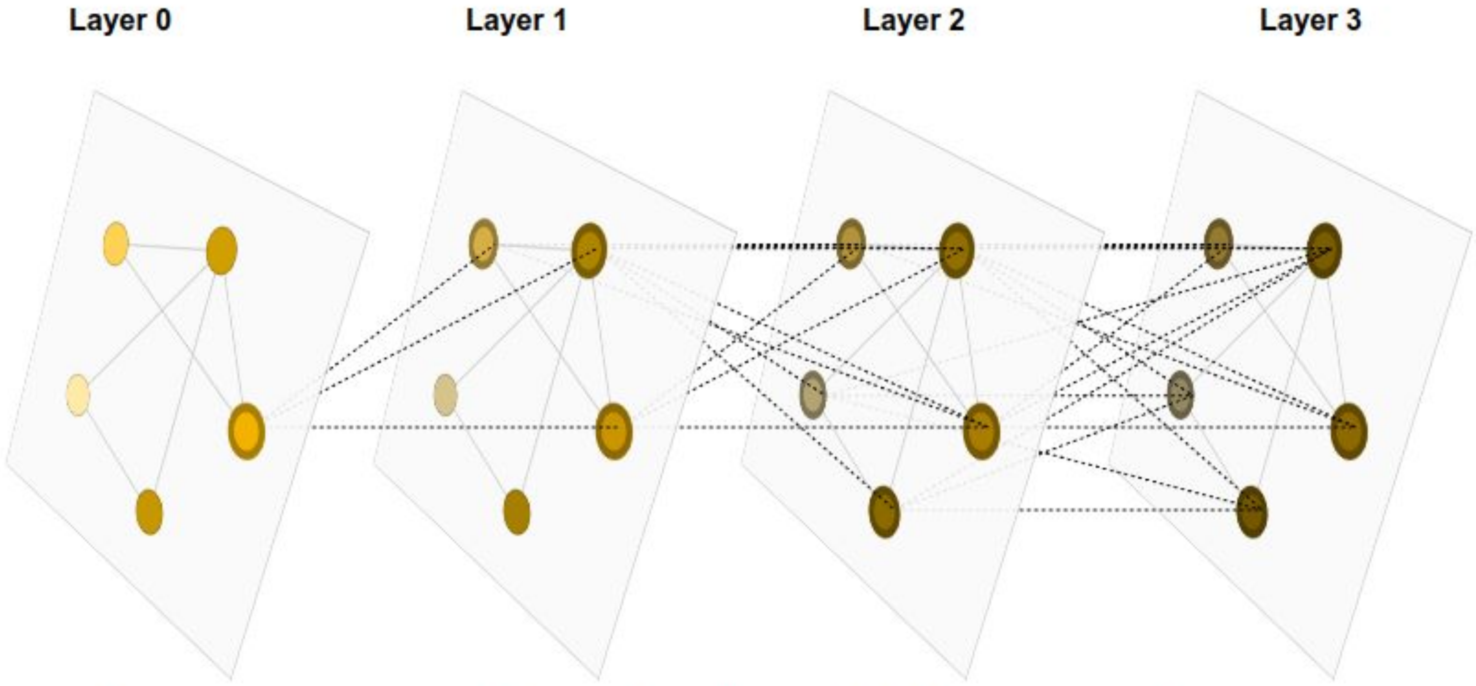
Published as a conference paper at ICLR 2018

Presenter: Yi-Ting Li
Date: Oct. 6, 2022

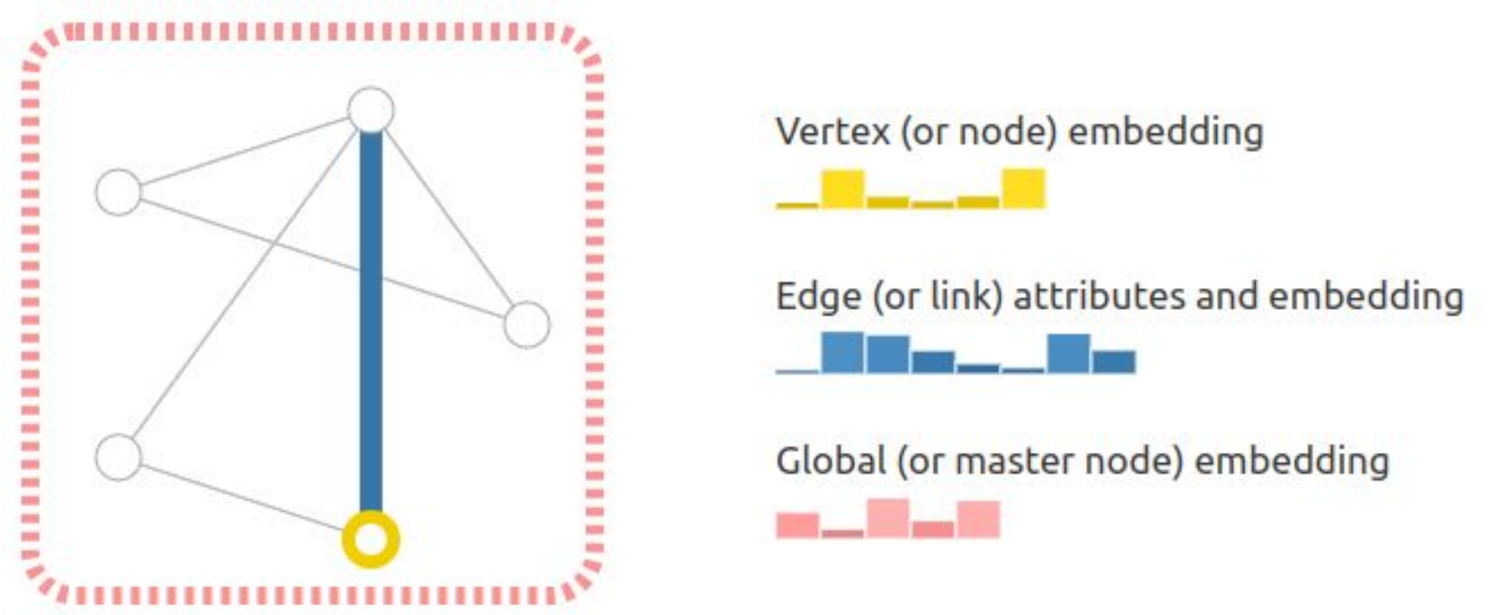
Outline

- Introduction to GNN
  - What is a graph
  - Tasks of graph-structured data
  - Challenges of applying GNNs
- GAT (Graph Attention Networks)
  - GAT Architecture
  - How activation function affects the GAT
  - Experiment

What is a GNN (Graph Neural Network)?



How to represents a graph?



What is a graph? Image is a graph.

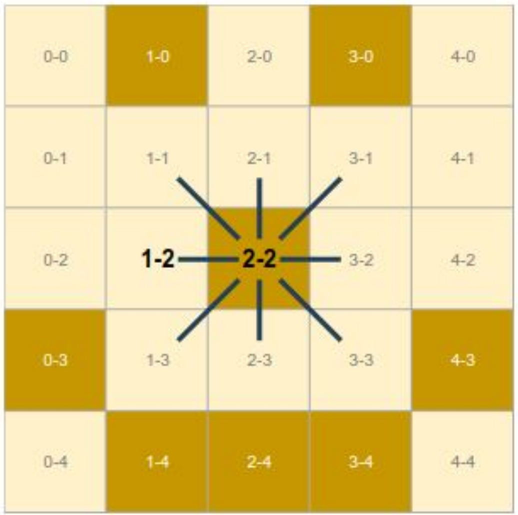
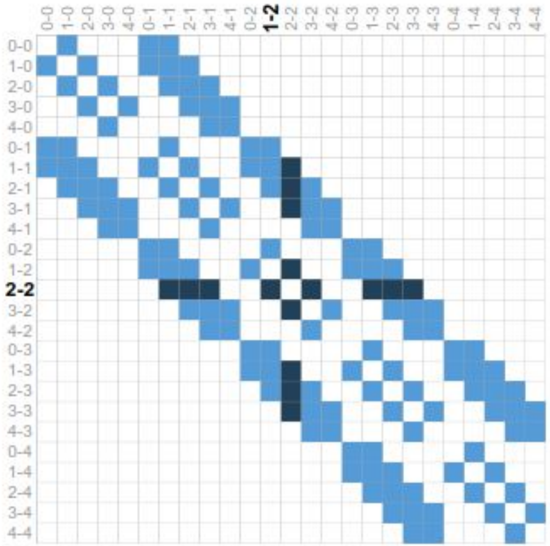
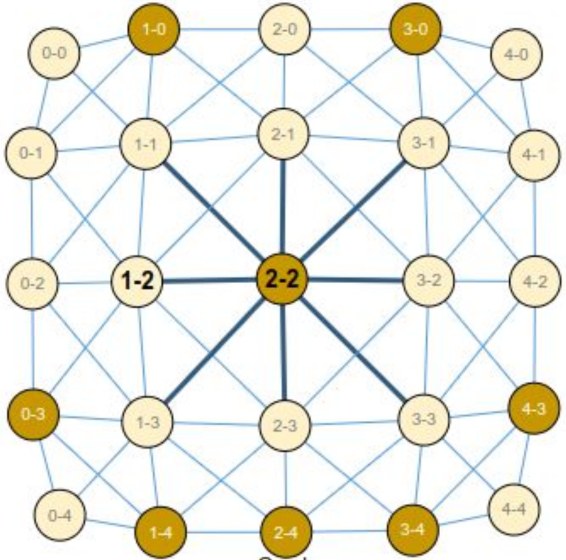


Image Pixels

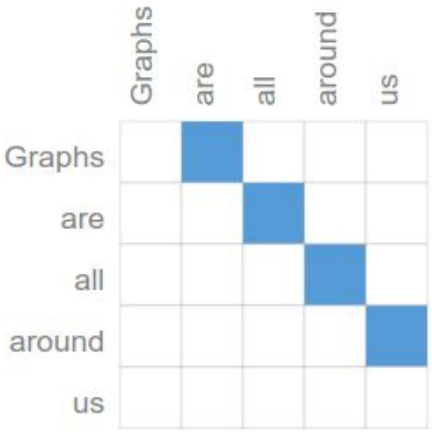


Adjacency Matrix

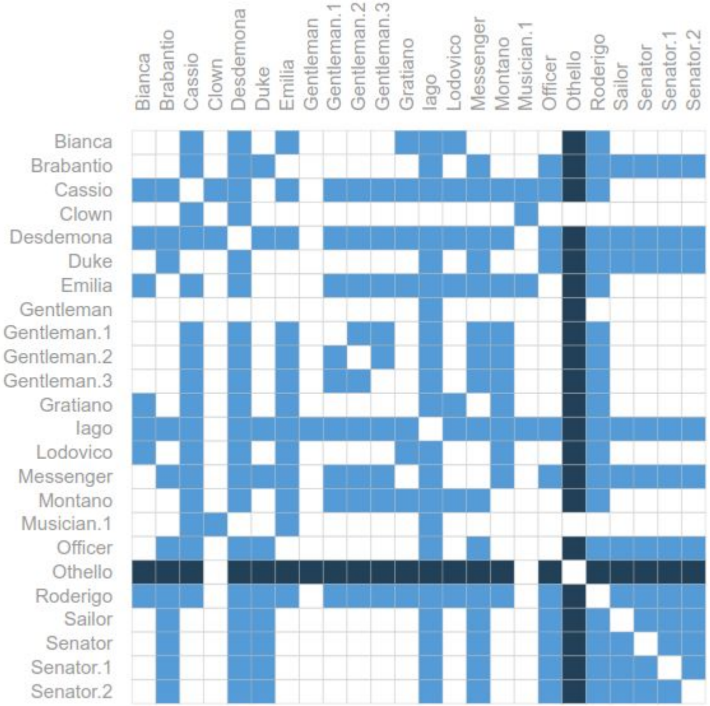


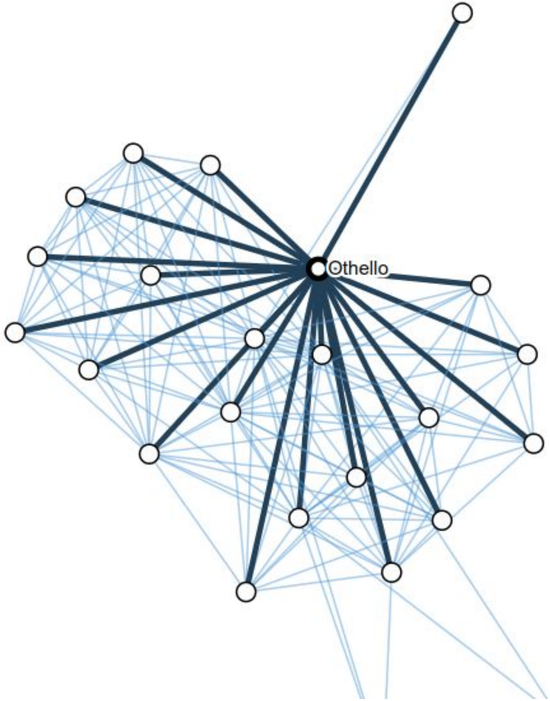
Graph

What is a graph? Text is a graph.



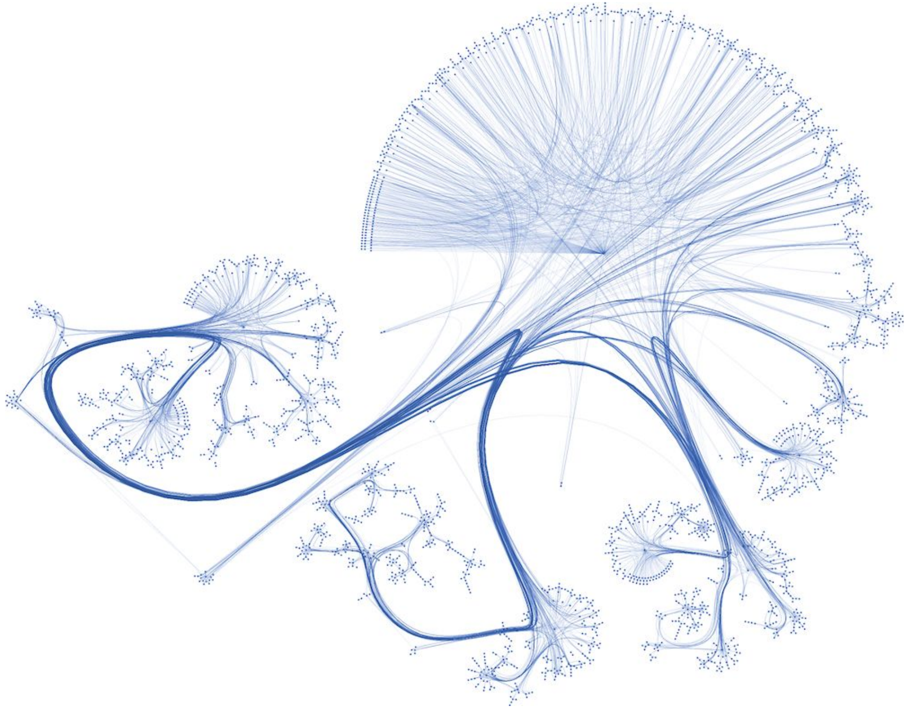
What is a graph? Social network is a graph.





What is a graph? More than you think

- **Citation networks**
- Math equations
- Programing code
- Knowledge graph

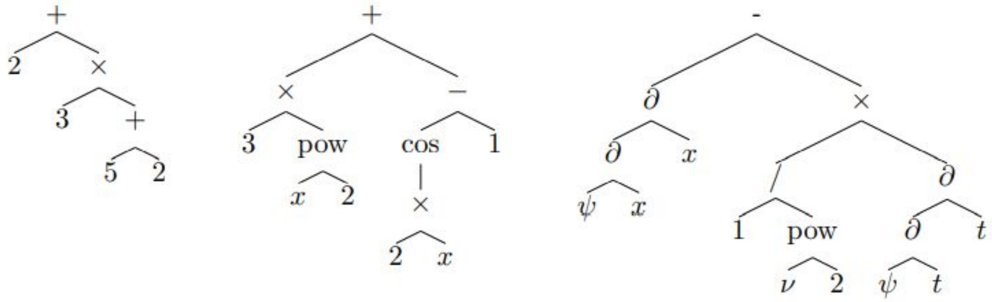


https://relational.fit.cvut.cz/dataset/CORA

What is a graph? More than you think

- Citation networks
- **Math equations**
- Programing code
- Knowledge graph

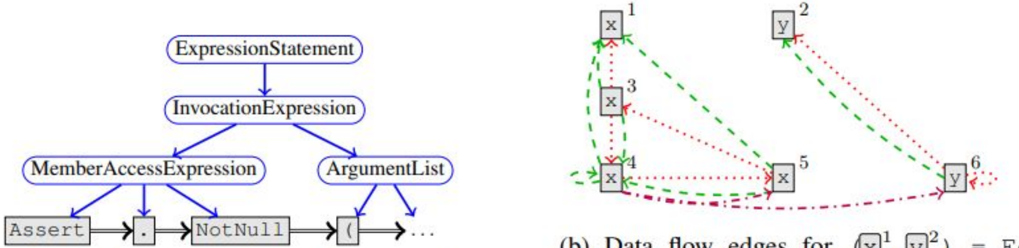
Mathematical expressions can be represented as trees, with operators and functions as internal nodes, operands as children, and numbers, constants and variables as leaves. The following trees represent expressions $2 + 3 \times (5 + 2)$, $3x^2 + \cos(2x) - 1$, and $\frac{\partial^2 \psi}{\partial x^2} - \frac{1}{\nu^2}\frac{\partial^2 \psi}{\partial t^2}$:



Lample, Guillaume, and François Charton. "Deep learning for symbolic mathematics." ICLR 2020

What is a graph? More than you think

- Citation networks
- Math equations
- **Programing code**
- Knowledge graph



(a) Simplified syntax graph for line 2 of Fig. 1, where blue rounded boxes are syntax nodes, black rectangular boxes syntax tokens, blue edges Child edges and double black edges NextToken edges.

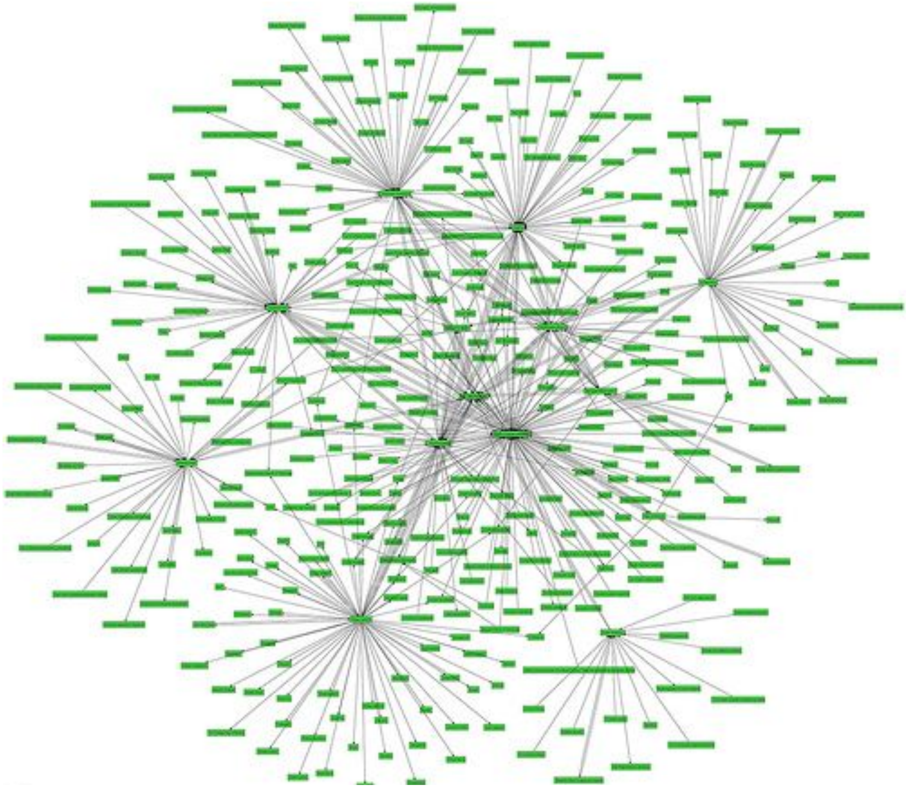
(b) Data flow edges for (x^1, y^2) = Foo(); while (x^3 > 0) x^4 = x^5 + y^6 (indices added for clarity), with red dotted LastUse edges, green dashed LastWrite edges and dashdotted purple ComputedFrom edges.

Figure 2: Examples of graph edges used in program representation.

Allamanis, Miltiadis, Marc Brockschmidt, and Mahmoud Khademi. "Learning to represent programs with graphs." ICLR 2018

What is a graph? More than you think

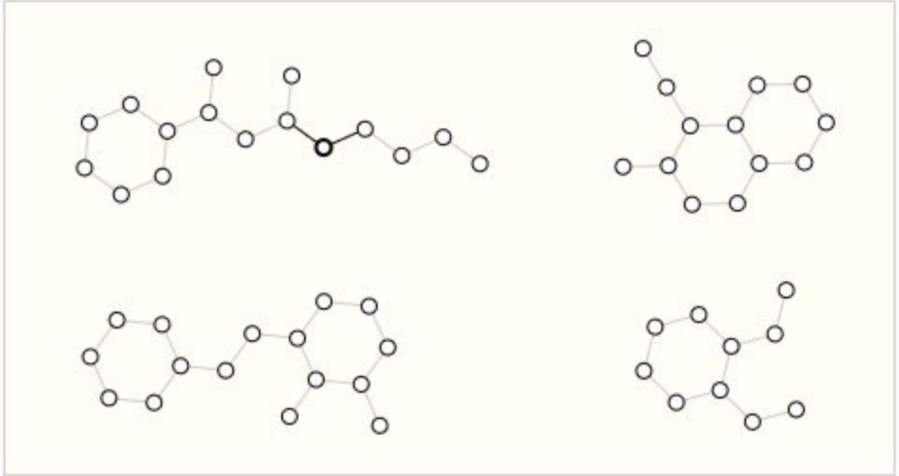
- Citation networks
- Math equations
- Programing code
- **Knowledge graph**



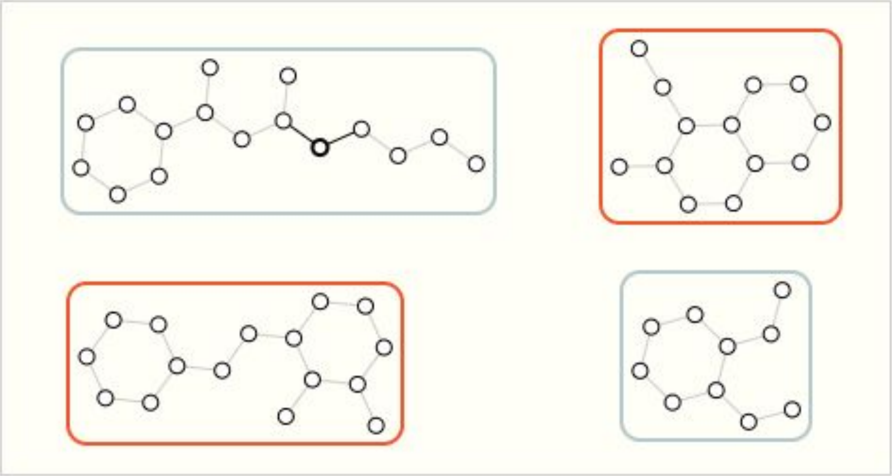
Paolo Boldi, et al. "Layered label propagation: A multiresolution coordinate-free ordering for compressing social networks." WWW 2011

Three task types of graph-structured data: **Graph-level**

- Image: Image classification
- Text: Sentiment analysis
- Molecule: Will it bind to a receptor implicated in a disease?



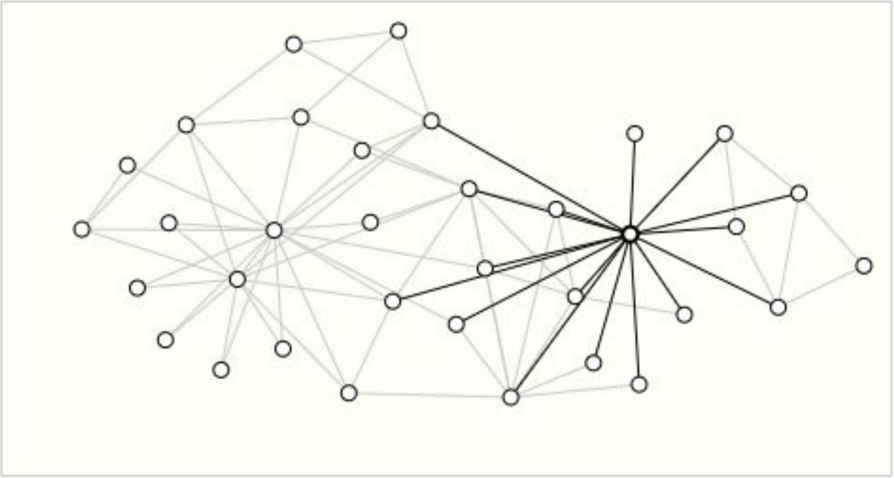
Input: graphs



Output: labels for each graph, (e.g., "does the graph contain two rings?")

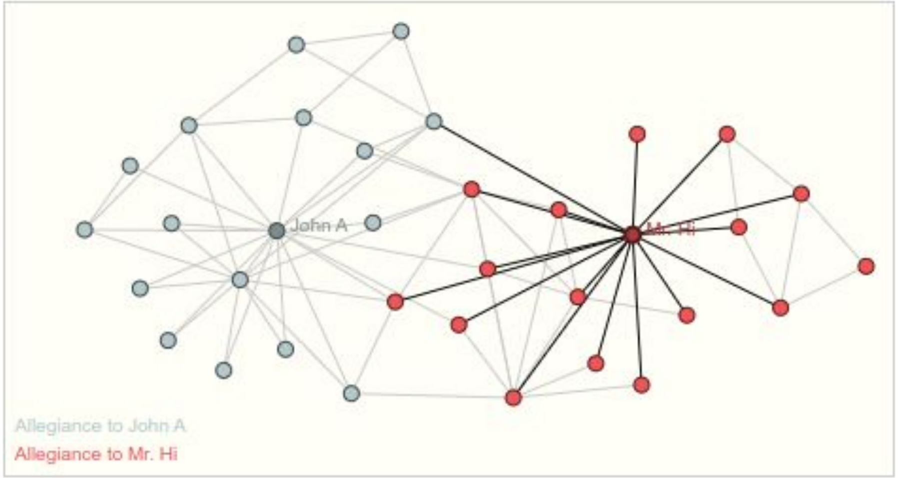
Three task types of graph-structured data: **Node-level**

- Image: Image segmentation
- Text: POS (parts-of-speech)
- Social network: Recommendation system



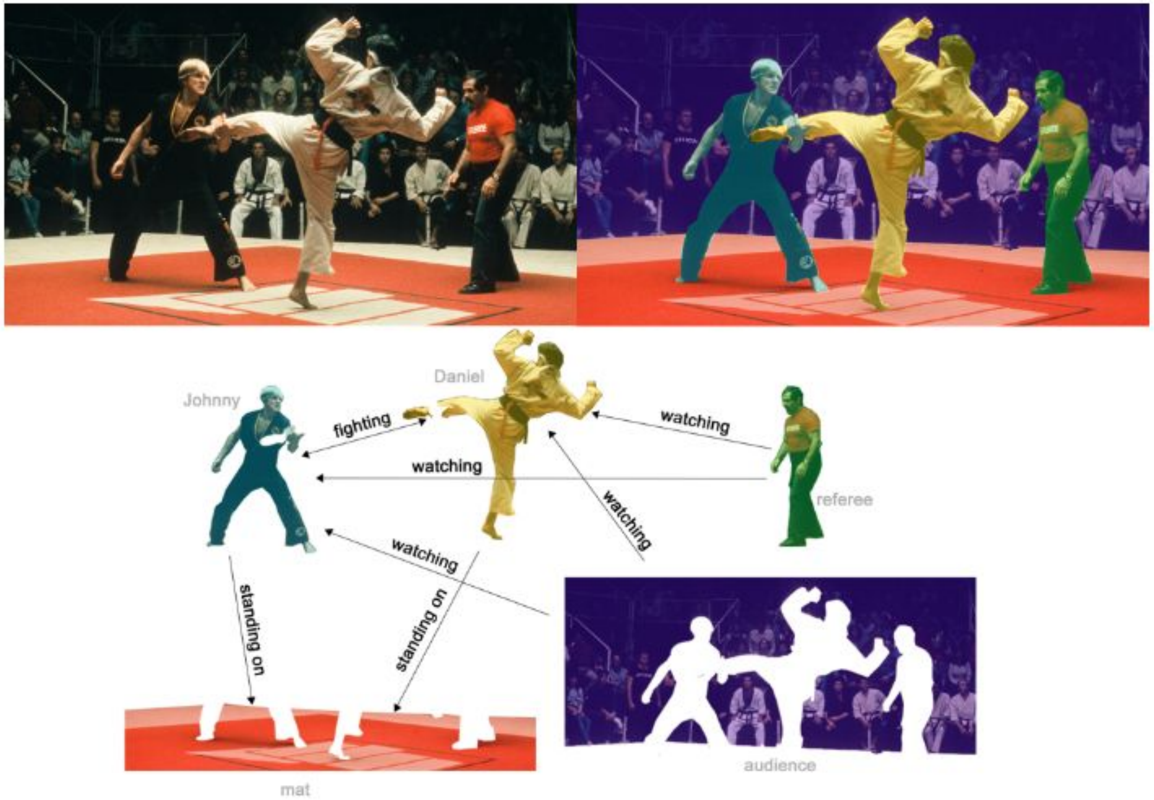
Input: graph with unlabled nodes

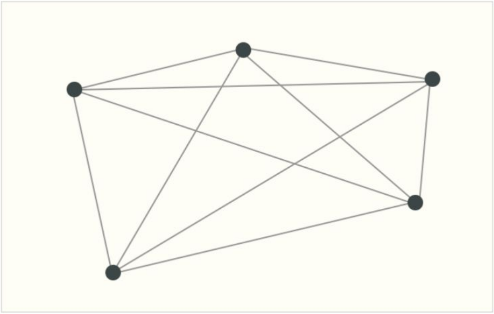
→



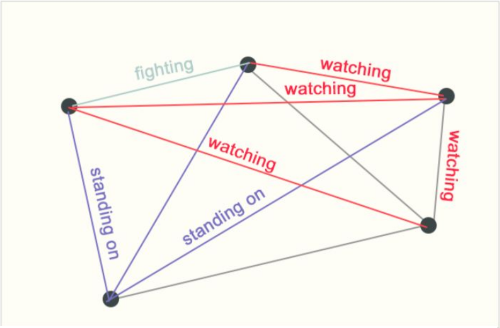
Output: graph node labels

Three task types of graph-structured data: **Edge-level**





Input: fully connected graph, unlabeled edges



Output: labels for edges

Challenges of applying GNNs

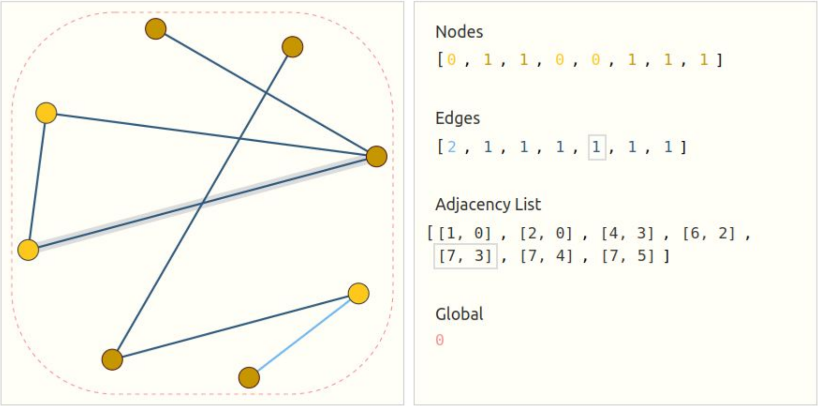
We have to represent 4 types data: nodes, edges, global-context and **connectivity**

Adjacency matrices?

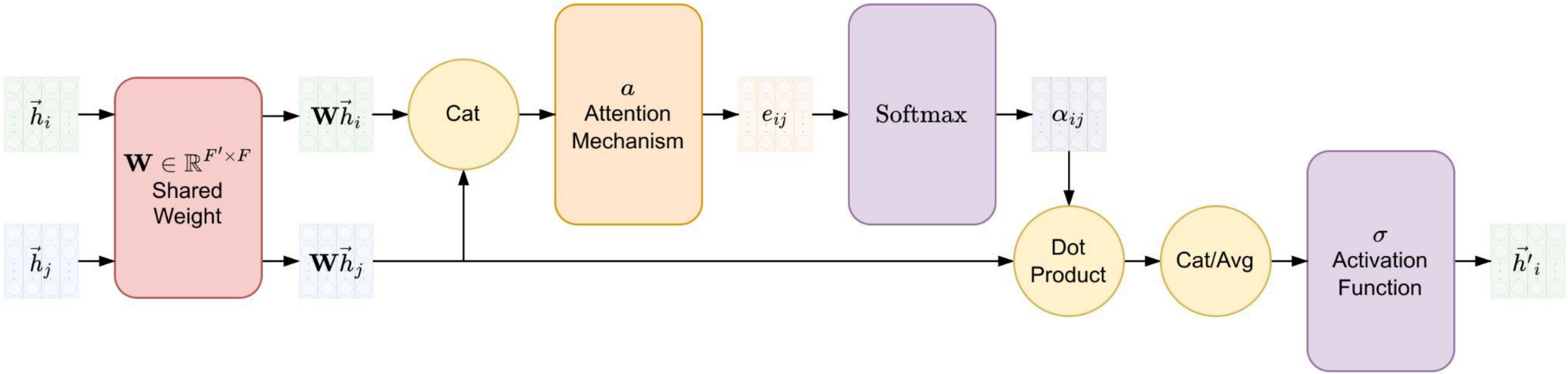
1. Sparse
2. Same graph might have many different adjacency matrices

Solution

- **Adjacency List**

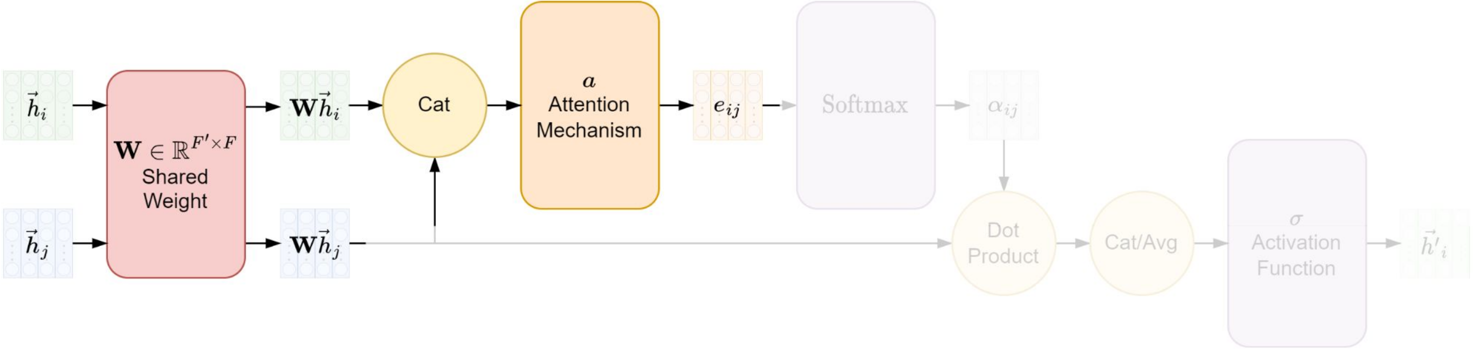


Graph Attention Networks



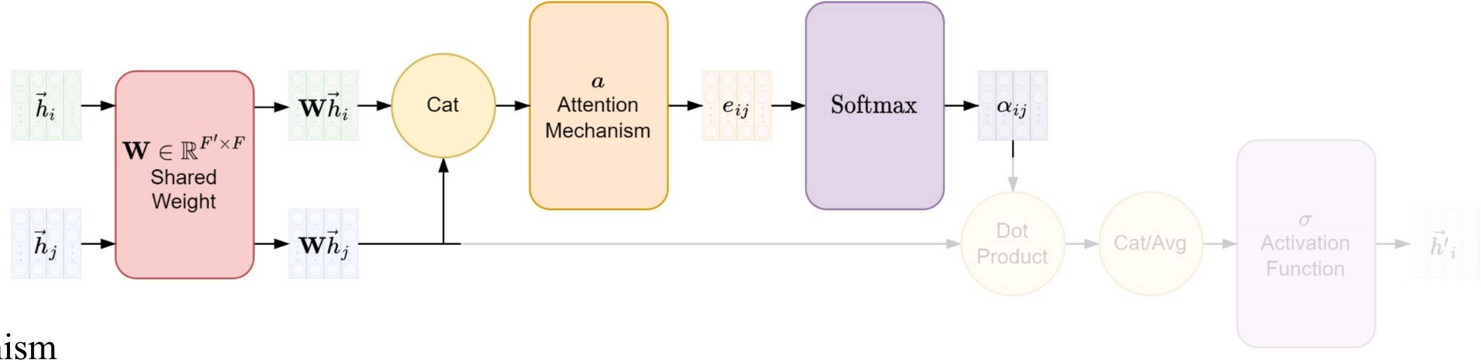
Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." ICLR 2015

GAT Layer



$h = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\}, \vec{h}_i \in \mathbb{R}^F$	Input to layers is a set of **node features**
$h' = \{\vec{h}'_1, \vec{h}'_2, \ldots, \vec{h}'_N\}, \vec{h}_i \in \mathbb{R}^{F'}$	Output from layers also a set of **node features**
$\mathbf{W} \in \mathbb{R}^{F' \times F}$	A shared linear **transformation**, parametrized by a weight matrix
a	A shared **attentional mechanism**
e_{ij}	The **attention coefficient**, here $j \in \mathcal{N}_i$, where $\mathcal{N}_i$ is **neighbor** of node i

GAT Layer



The attention mechanism

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$$

The attention score, where $e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$

Applying attention <u>in experiment</u>

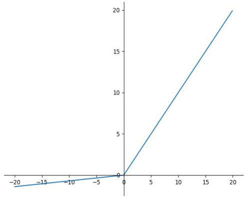
$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

建良:為啥要加LeakyReLU?

The attention mechanism in experiments, here

$\vec{\mathbf{a}}^T \in \mathbb{R}^{2F'}$ is a weight vector of a

LeakyReLU



Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." ICLR 2015

Why LeakyReLU?

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

$$= \frac{\exp[(\vec{a}_1 \| \vec{a}_2)^T (\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j)]}{\sum_{k\in\mathcal{N}_i} \exp[(\vec{a}_1 \| \vec{a}_2)^T (\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k)]}$$

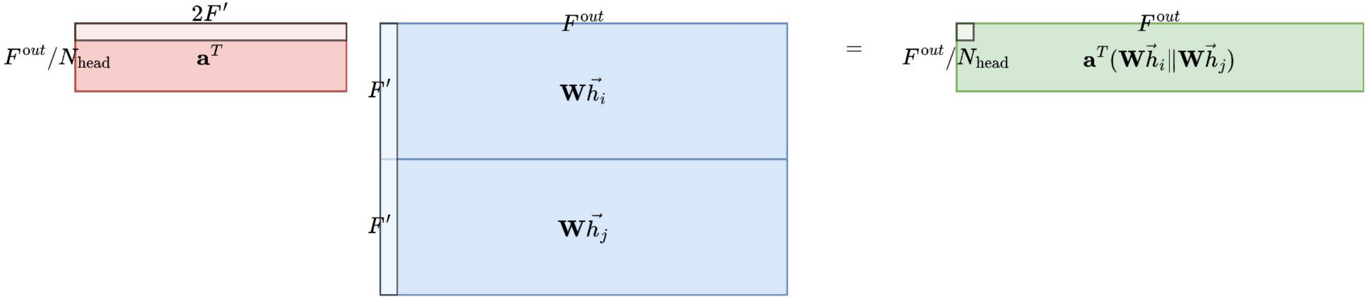
$$= \frac{\exp[\vec{a}_1{}^T\mathbf{W}\vec{h}_i + \vec{a}_2{}^T\mathbf{W}\vec{h}_j]}{\sum_{k\in\mathcal{N}_i} \exp[\vec{a}_1{}^T\mathbf{W}\vec{h}_i + \vec{a}_2{}^T\mathbf{W}\vec{h}_j]}$$

$$= \frac{\cancel{\exp(\vec{a}_1{}^T\mathbf{W}\vec{h}_i)}\ \exp(\vec{a}_2{}^T\mathbf{W}\vec{h}_j)}{\sum_{k\in\mathcal{N}_i} \cancel{\exp(\vec{a}_1{}^T\mathbf{W}\vec{h}_i)}\ \exp(\vec{a}_2{}^T\mathbf{W}\vec{h}_k)}$$

$$= \frac{\exp(\vec{a}_2{}^T\mathbf{W}\vec{h}_j)}{\sum_{k\in\mathcal{N}_i} \exp(\vec{a}_2{}^T\mathbf{W}\vec{h}_k)}$$

峻毅:可以加別的嗎？

Why LeakyReLU?



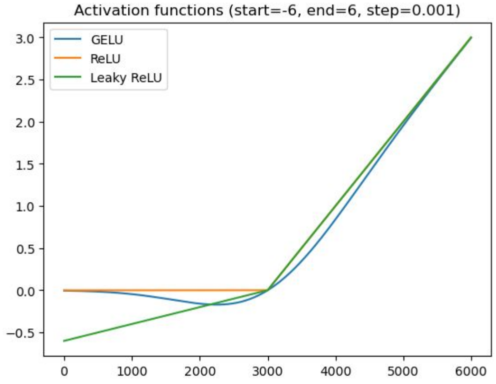
$$\alpha_{ij} = \frac{\exp(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j])}{\sum_{k\in\mathcal{N}_i} \exp(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k])}$$

$$= \frac{\exp[(\vec{a}_1 \| \vec{a}_2)^T \, (\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j)]}{\sum_{k\in\mathcal{N}_i} \exp[(\vec{a}_1 \| \vec{a}_2)^T \, (\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k)]}$$

$$= \frac{\exp[\, \vec{a}_1{}^T\mathbf{W}\vec{h}_i + \vec{a}_2{}^T\mathbf{W}\vec{h}_j]}{\sum_{k\in\mathcal{N}_i} \exp[\, \vec{a}_1{}^T\mathbf{W}\vec{h}_i + \vec{a}_2{}^T\mathbf{W}\vec{h}_j]}$$

$$(\mathbf{a}^T_{0,0}\mathbf{W}\vec{h}_{i\,0,0} + \mathbf{a}^T_{0,1}\mathbf{W}\vec{h}_{i\,1,0} + \cdots + \mathbf{a}^T_{0,F'}\mathbf{W}\vec{h}_{i\,F',0}) + (\mathbf{a}^T_{0,F'+1}\mathbf{W}\vec{h}_{j\,0,0} + \mathbf{a}^T_{0,F'+2}\mathbf{W}\vec{h}_{j\,1,0} + \cdots + \mathbf{a}^T_{0,2F'}\mathbf{W}\vec{h}_{j\,F',0}) = \mathbf{a}^T(\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j)_{0,0}$$

How activation function affects the GAT

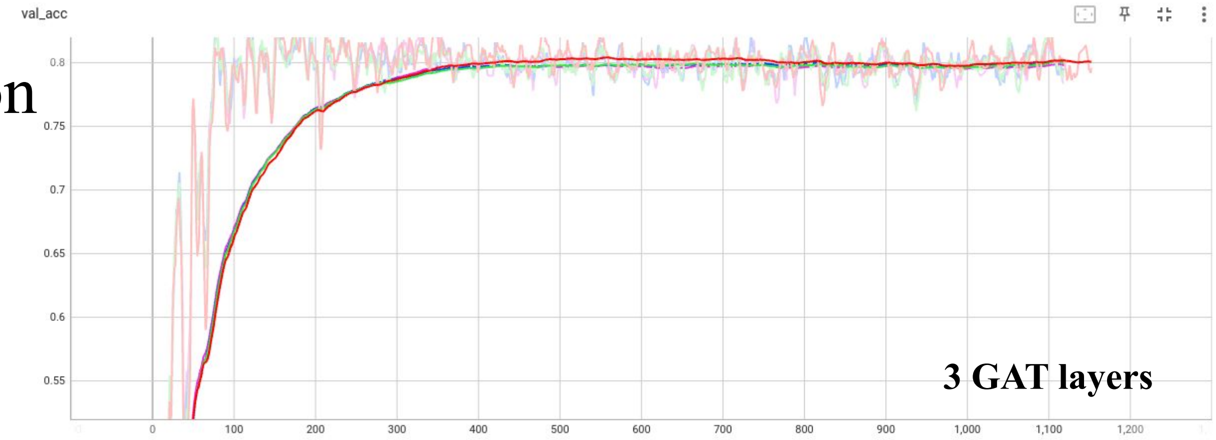


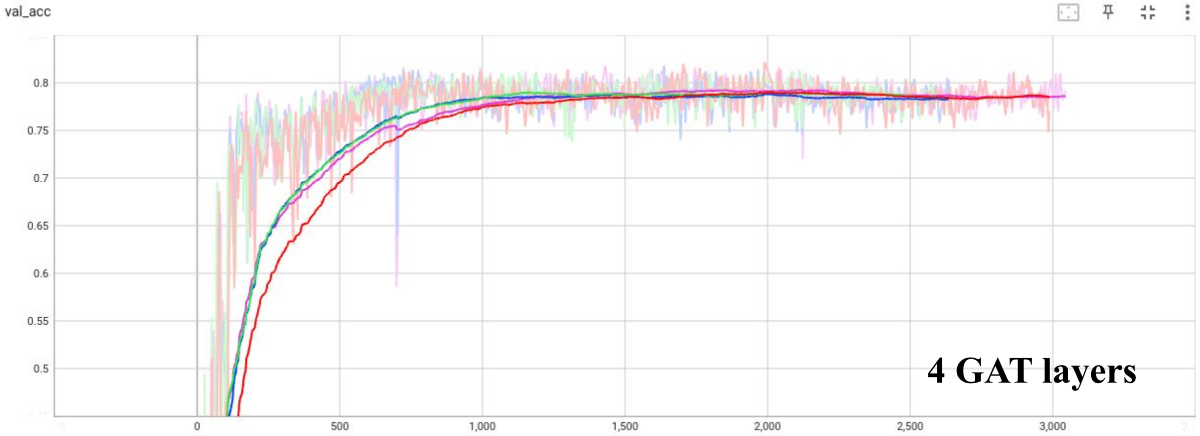
LeakyReLU

w/o activation

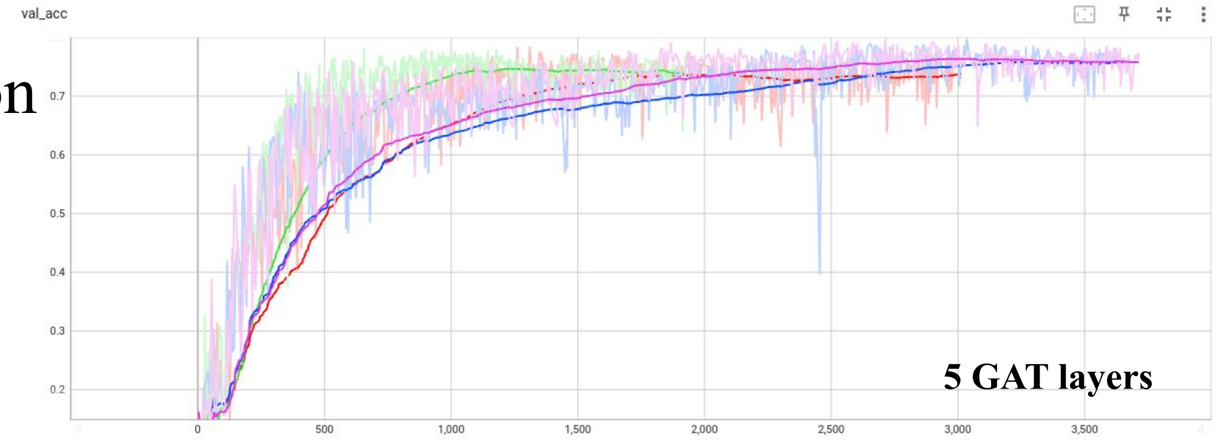
ReLU

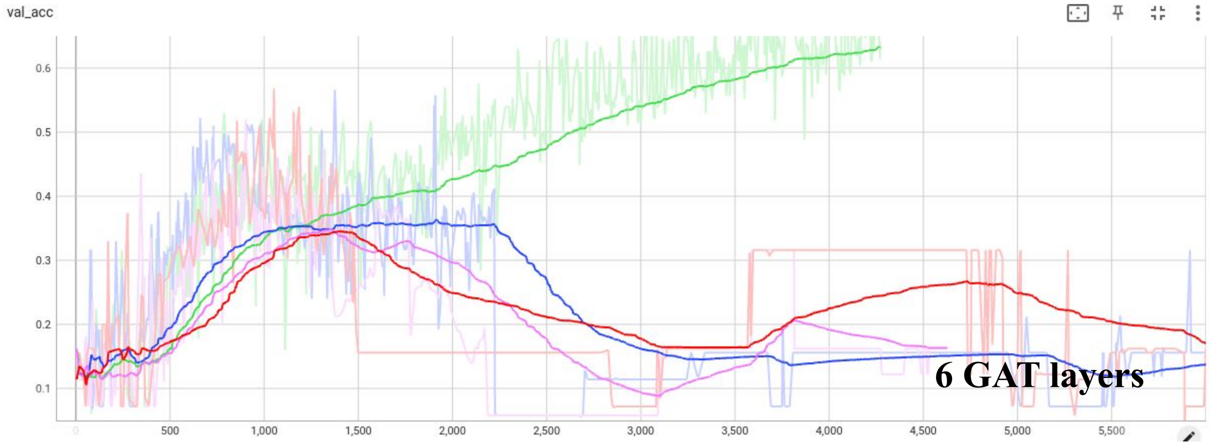
GELU





How activation function affects the GAT





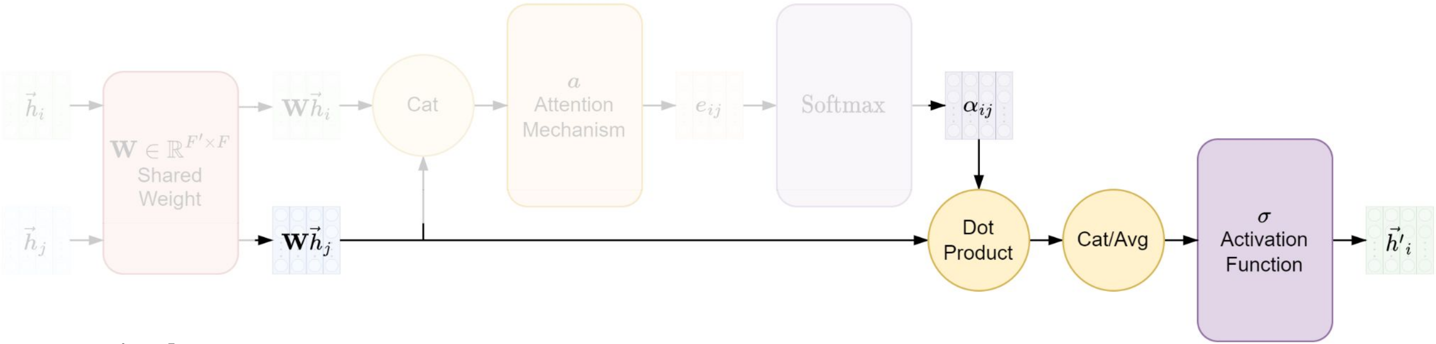
LeakyReLU

w/o activation

ReLU

GELU

GAT Layer



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

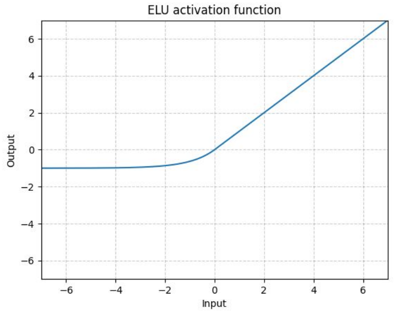
The output features, here the σ is an **activation function**

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

Apply multi-head attention in the first and middle GAT layers, here the K is the number of heads.

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

In the final GAT layer, we apply average instead of concatenation



Vaswani, Ashish, et al. "Attention is all you need." NIPS 2017

Experiment: Datasets

Table 1: Summary of the datasets used in our experiments.

	Cora	**Citeseer**	**Pubmed**	**PPI**
Task	Transductive	Transductive	Transductive	Inductive
# Nodes	2708 (1 graph)	3327 (1 graph)	19717 (1 graph)	56944 (24 graphs)
# Edges	5429	4732	44338	818716
# Features/Node	1433	3703	500	50
# Classes	7	6	3	121 (multilabel)
# Training Nodes	140	120	60	44906 (20 graphs)
# Validation Nodes	500	500	500	6514 (2 graphs)
# Test Nodes	1000	1000	1000	5524 (2 graphs)
	Node-level	**Node-level**	**Node-level**	**Graph-level**

Experiment

Transductive

Method	**Cora**	**Citeseer**	**Pubmed**
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%	74.4%
GCN (Kipf & Welling, 2017)	81.5%	70.3%	**79.0%**
MoNet (Monti et al., 2016)	$81.7 \pm 0.5\%$	—	$78.8 \pm 0.3\%$
GCN-64*	$81.4 \pm 0.5\%$	$70.9 \pm 0.5\%$	$\mathbf{79.0} \pm 0.3\%$
GAT (ours)	$\mathbf{83.0} \pm 0.7\%$	$\mathbf{72.5} \pm 0.7\%$	$\mathbf{79.0} \pm 0.3\%$

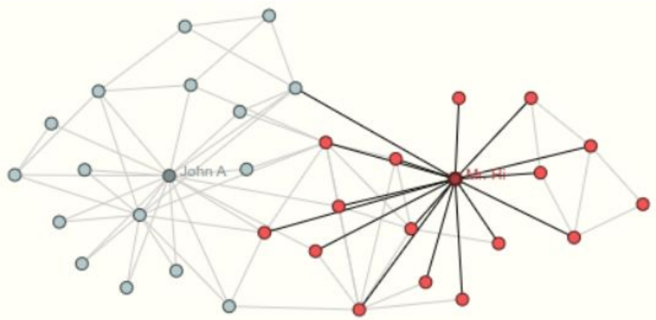
Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." ICLR 2017

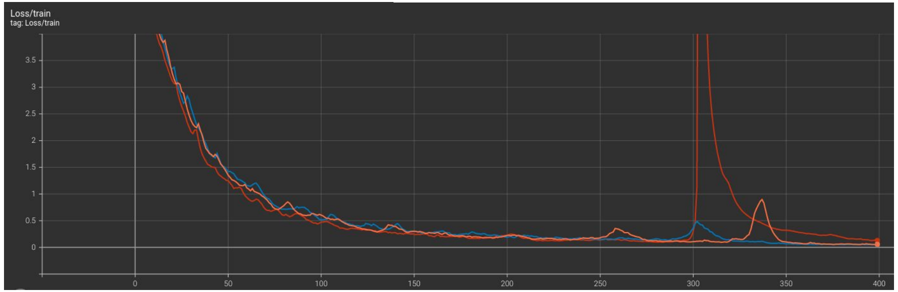
Experiment

Table 3: Summary of results in terms of micro-averaged F_1 scores, for the PPI dataset. GraphSAGE* corresponds to the best GraphSAGE result we were able to obtain by just modifying its architecture. Const-GAT corresponds to a model with the same architecture as GAT, but with a constant attention mechanism (assigning same importance to each neighbor; GCN-like inductive operator).

Inductive

Method	**PPI**
Random	0.396
MLP	0.422
GraphSAGE-GCN (Hamilton et al., 2017)	0.500
GraphSAGE-mean (Hamilton et al., 2017)	0.598
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612
GraphSAGE-pool (Hamilton et al., 2017)	0.600
GraphSAGE*	0.768
Const-GAT (ours)	0.934 ± 0.006
GAT (ours)	$\mathbf{0.973} \pm 0.002$





Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." NIPS 2017

SWOT

Strength

- **Easy** to implement
- **Learnable** aggregator
- Could be used for **inductive tasks**

Weakness

- **Sensitive** to parameter initialization
- **LeakyReLU** is indispensable

Opportunity

- **Edge-level** task by node attention
- Combine the **Reformer**: dismiss low attention neighbor

Threat

- **Over-smoothing problem**: feature with over-smoothing when model too depth

Kitaev, Nikita, Łukasz Kaiser, and Anselm Levskaya. "Reformer: The efficient transformer." ICLR 2020

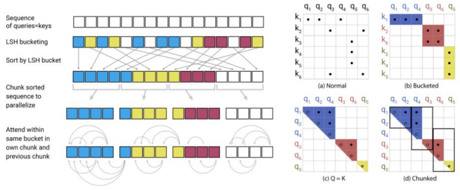


Figure 2: Simplified depiction of LSH Attention showing the hash-bucketing, sorting, and chunking steps and the resulting causal attentions. (a-d) Attention matrices for these varieties of attention.