GRAPH ATTENTION NETWORKS

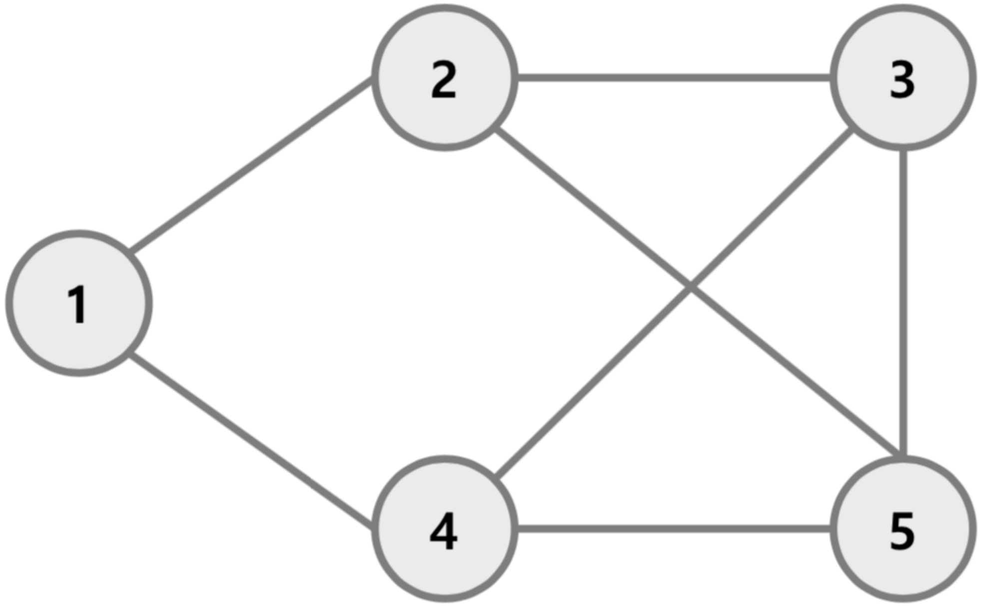
by
Petar Velicˇkovic ́∗, Guillem Cucurull∗, Arantxa Casanova∗,
Adriana Romero, Pietro Lio`, Yoshua Bengio

presented by Sukwon Yun

Contents

1. Background

- Graph Neural Network (GNN)



Notations	Descriptions
$\lvert \cdot \rvert$	The length of a set.
$\odot$	Element-wise product.
G	A graph.
V	The set of nodes in a graph.
v	A node $v \in V$.
E	The set of edges in a graph.
e_{ij}	An edge $e_{ij} \in E$.
$N(v)$	The neighbors of a node v.
$\mathbf{A}$	The graph adjacency matrix.

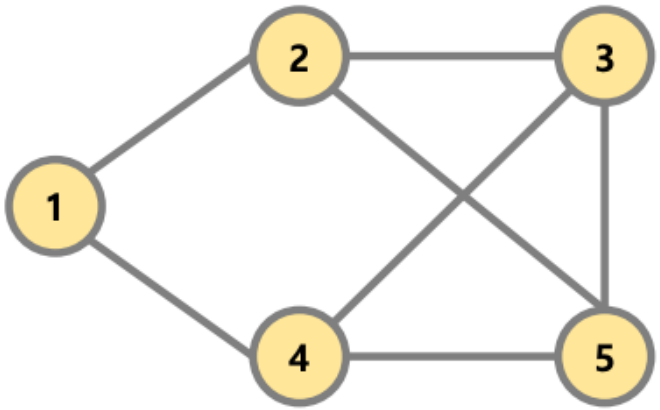
Wu, Zonghan, et al. "A comprehensive survey on graph neural networks." *IEEE transactions on neural networks and learning systems* (2020).

1. Background

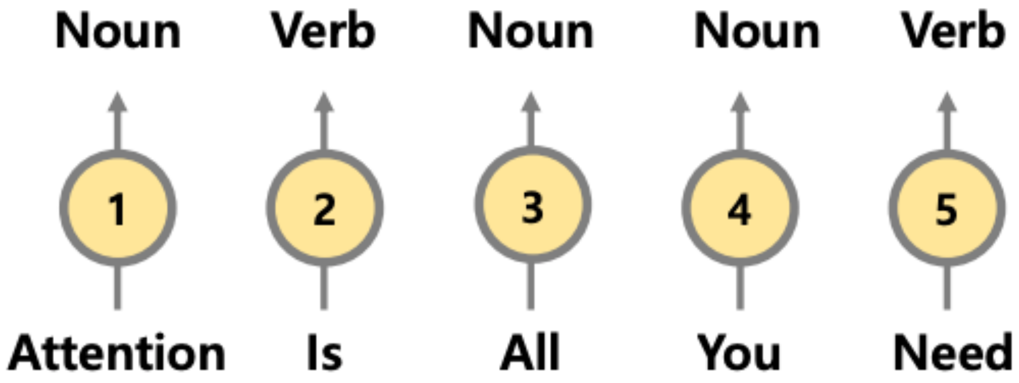
- Task of GNN

  - Node Level
  - Edge Level
  - Graph Level

GAT -> Node level



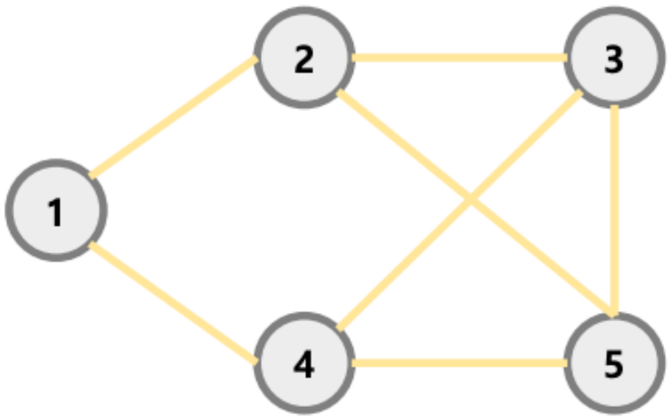
Node Level



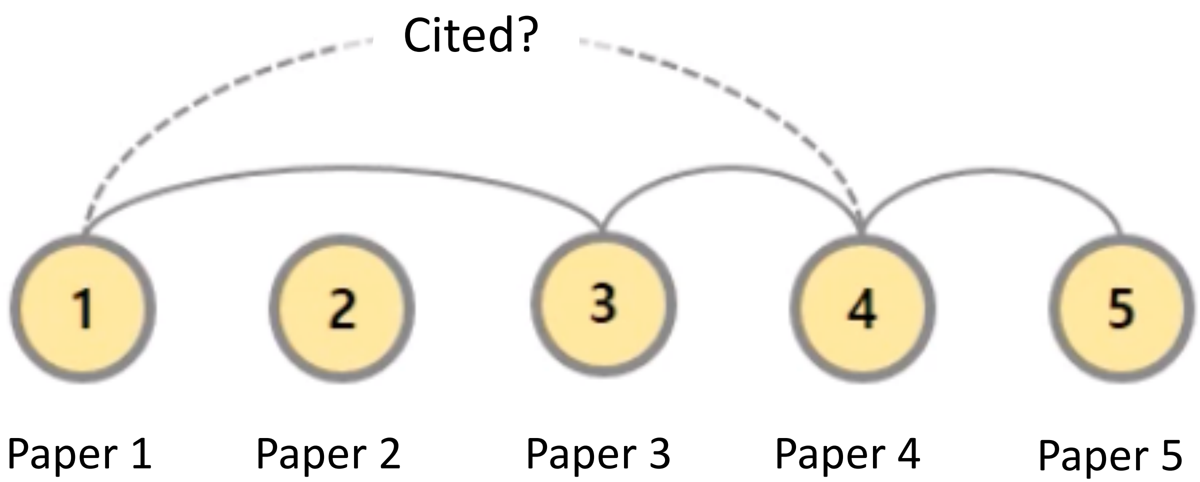
1. Background

- Task of GNN

  - Node Level
  - Edge Level
  - Graph Level



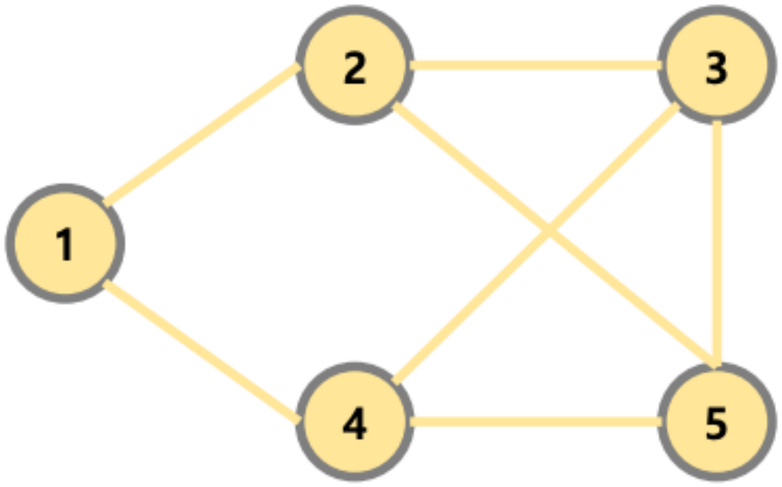
Edge Level



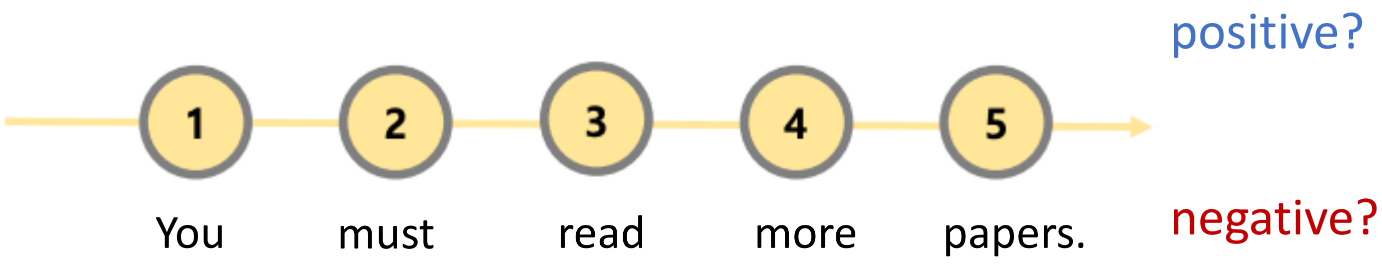
1. Background

- Task of GNN

  - Node Level
  - Edge Level
  - Graph Level



Graph Level



1. Background

Graph **Attention** Networks

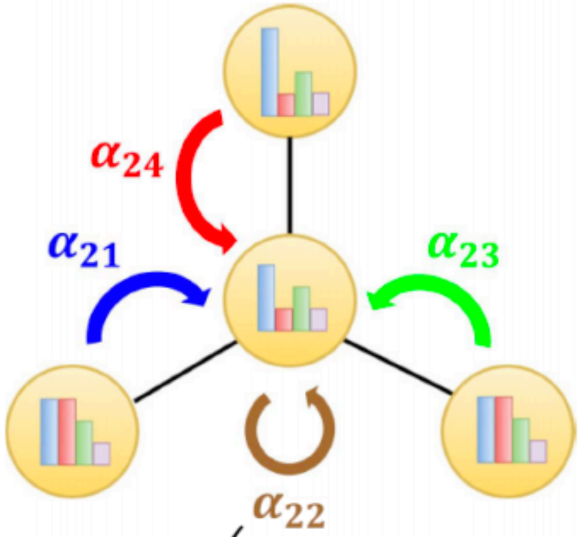
" Give weight to Specific / Interesting Node "

1. Background

Vanilla GCN updates information of neighbor atoms **with same importance**.

$$H^{(l+1)} = \sigma\left(\sum_{j \in N(i)} H_j^{(l)} W^{(l)}\right)$$

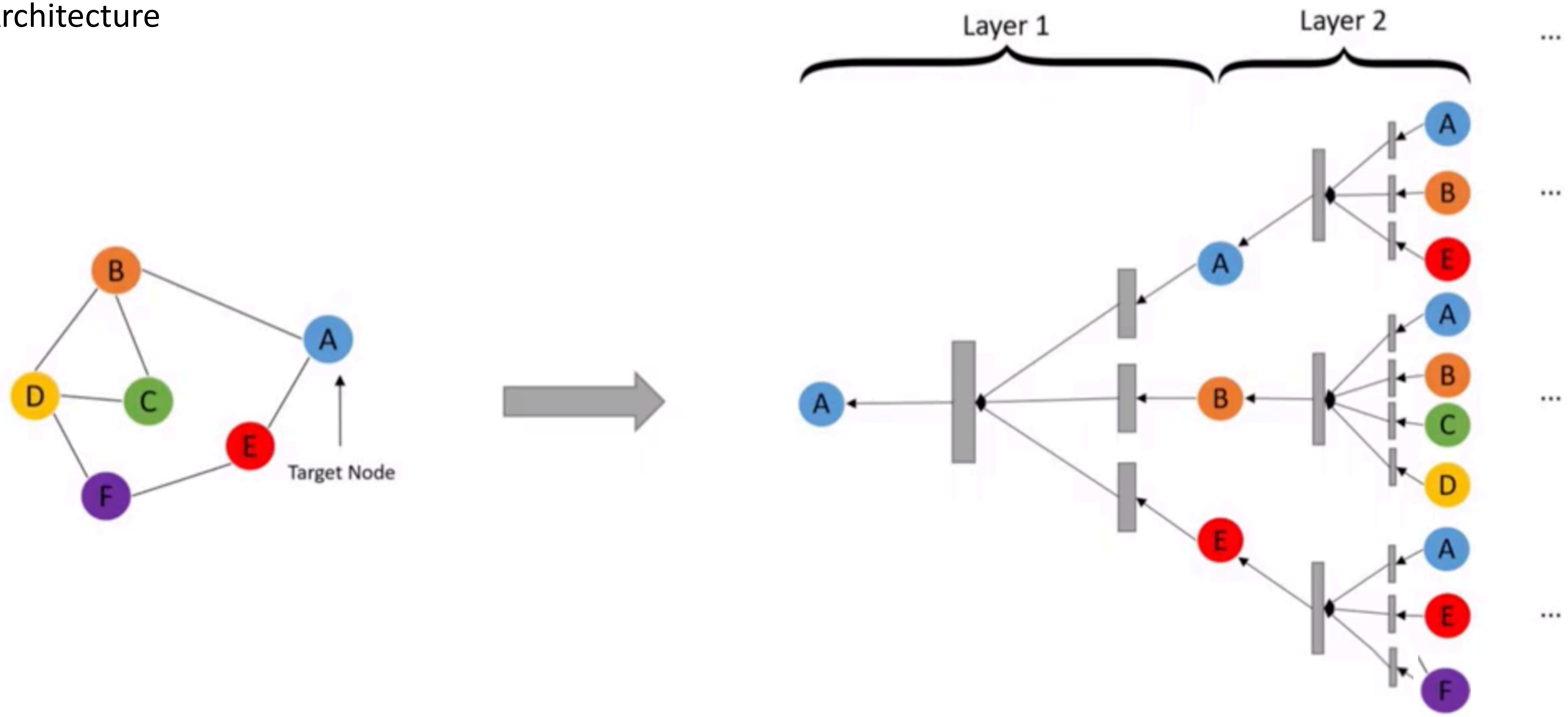
Attention mechanism enables it to update nodes **with different importance**



$$H^{(l+1)} = \sigma\left(\sum_{j \in N(i)} \alpha_{ij} H_j^{(l)} W^{(l)}\right)$$

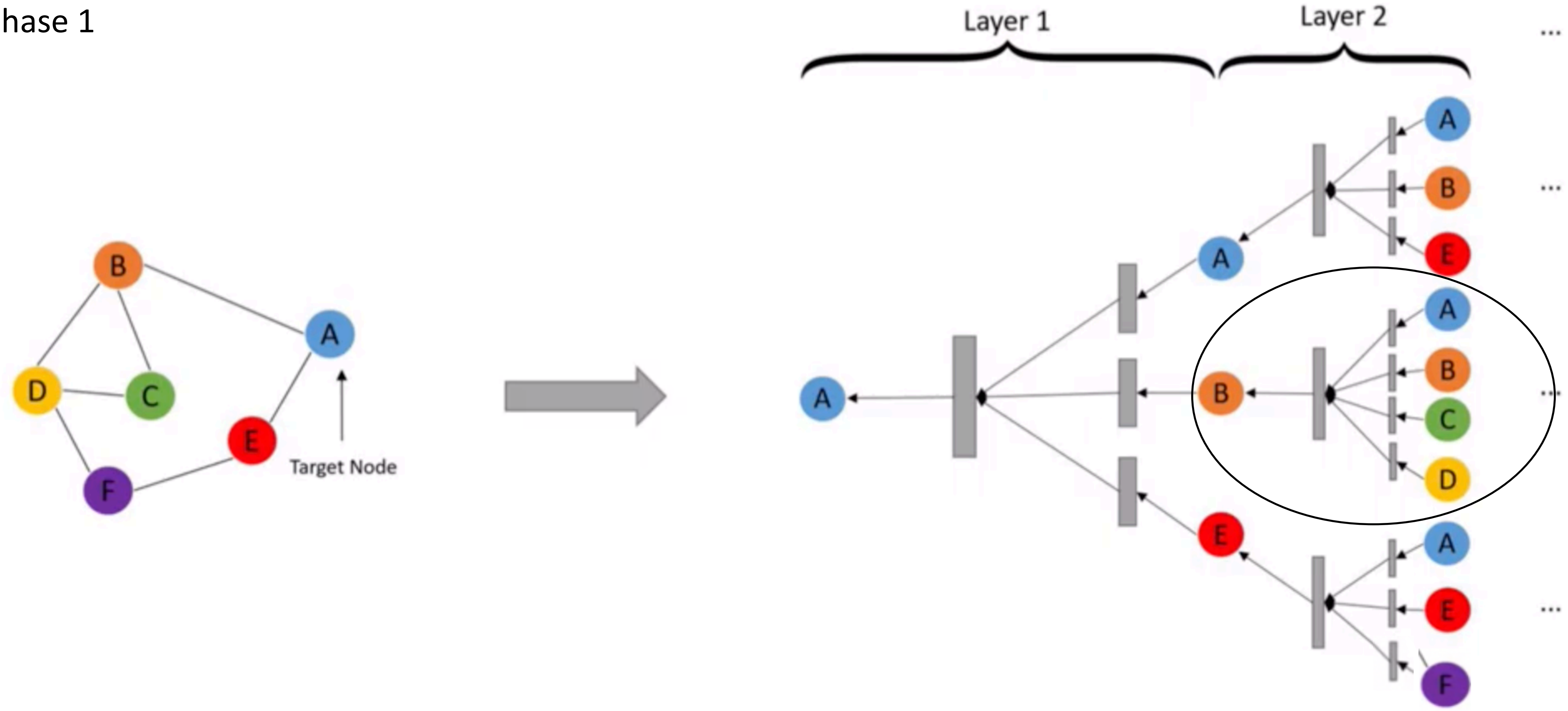
2. Introduction

- Architecture



2. Introduction

- Phase 1



2. Introduction

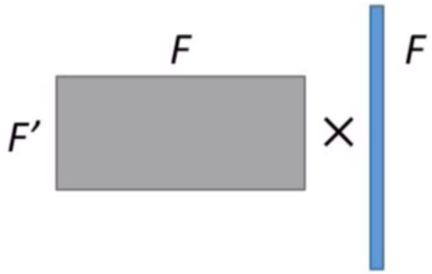
1. Start with original features

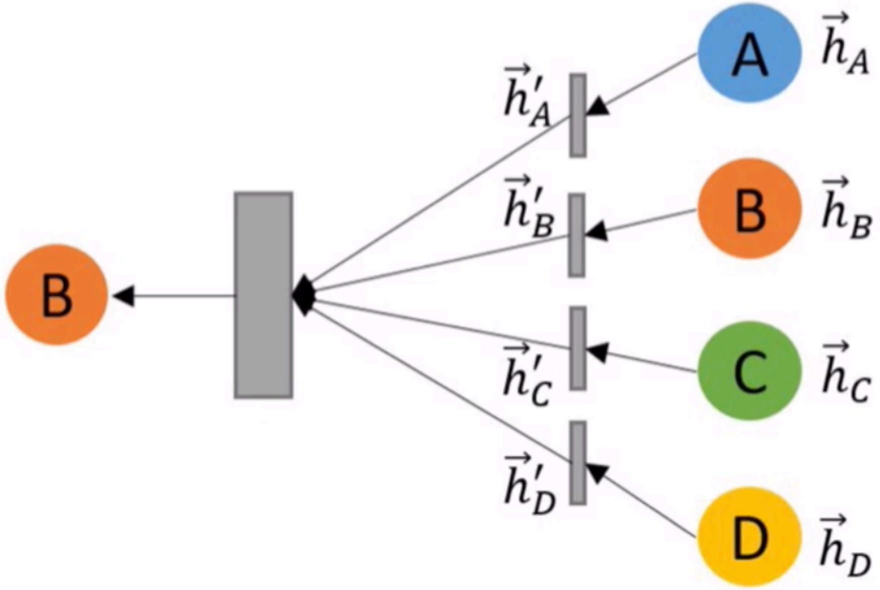
$$\mathbf{h} = \{\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_N\}, \vec{h}_i \in \mathbb{R}^F$$

2. Linear Transformation

$$\vec{h}'_i = \mathbf{W}\vec{h}_i$$

$$\mathbf{h}' = \{\vec{h}'_1, \vec{h}'_2, \ldots, \vec{h}'_N\}, \vec{h}'_i \in \mathbb{R}^{F'}$$

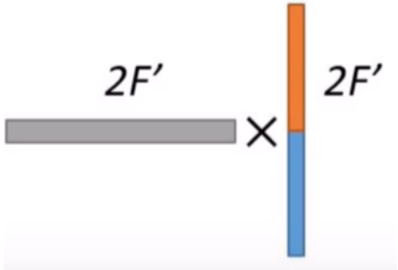


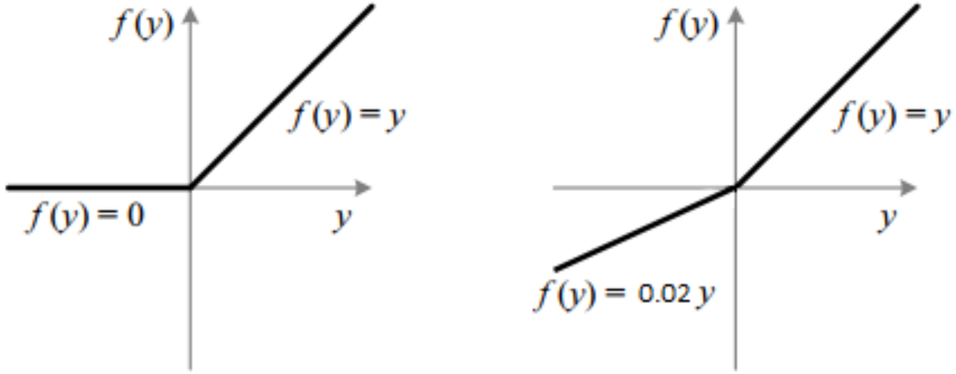


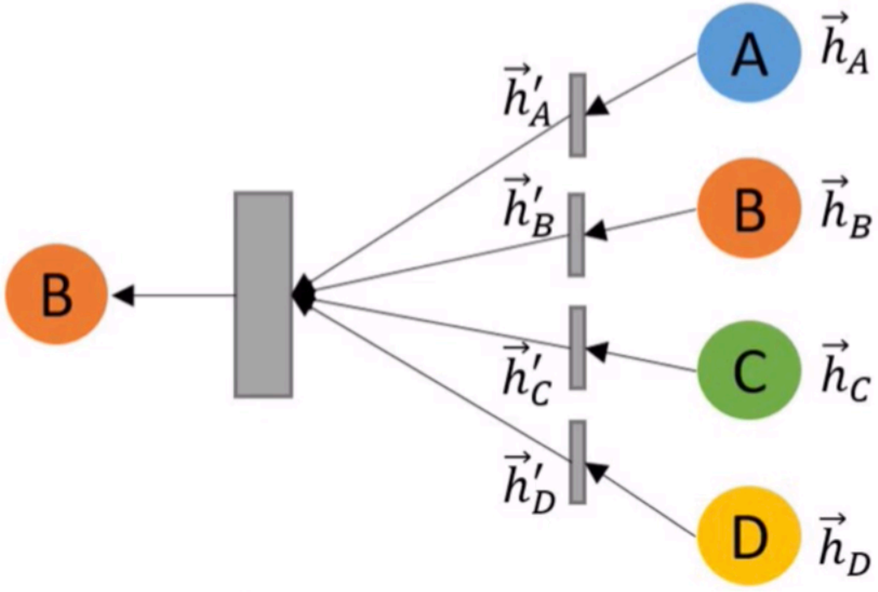
2. Introduction

3. Evaluate Attention

$e_{B,i} = LeakyReLU(\vec{a}^T[\vec{h}'_B || \vec{h}'_i])$, where i = A, B, C, D







2. Introduction

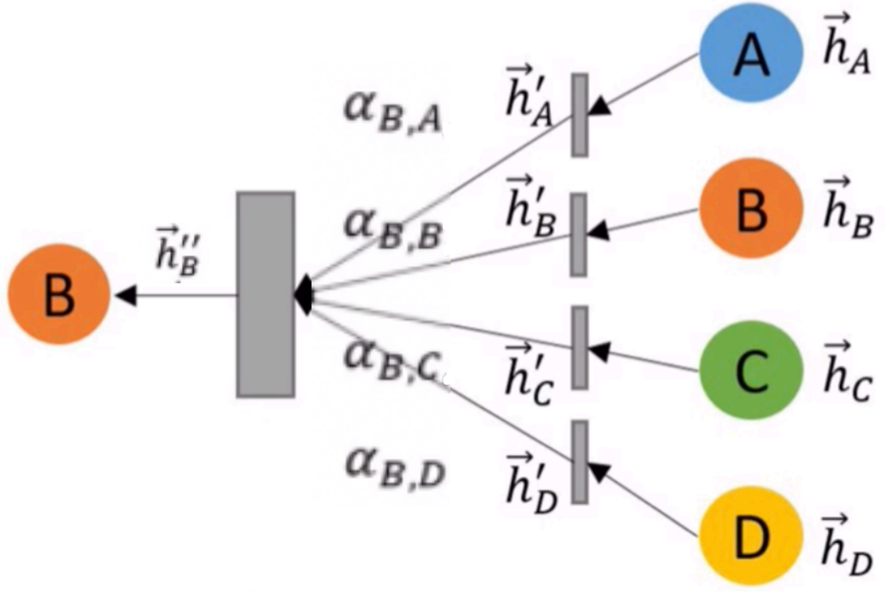
3. Evaluate Attention

$$e_{B,i} = LeakyReLU(\vec{a}^T[\vec{h}'_B||\vec{h}'_i]),\ \text{where } i = A, B, C, D$$

$$\alpha_{B,i} = softmax(e_{B,i}) = \frac{\exp(e_{B,i})}{\sum_i \exp(e_{B,i})},\ \text{where } i = A, B, C, D.$$

4. Summation

$$\vec{h}''_B = \sigma(\textstyle\sum_i \alpha_{B,i}\vec{h}'_i),\ \text{where } i = A, B, C, D.$$

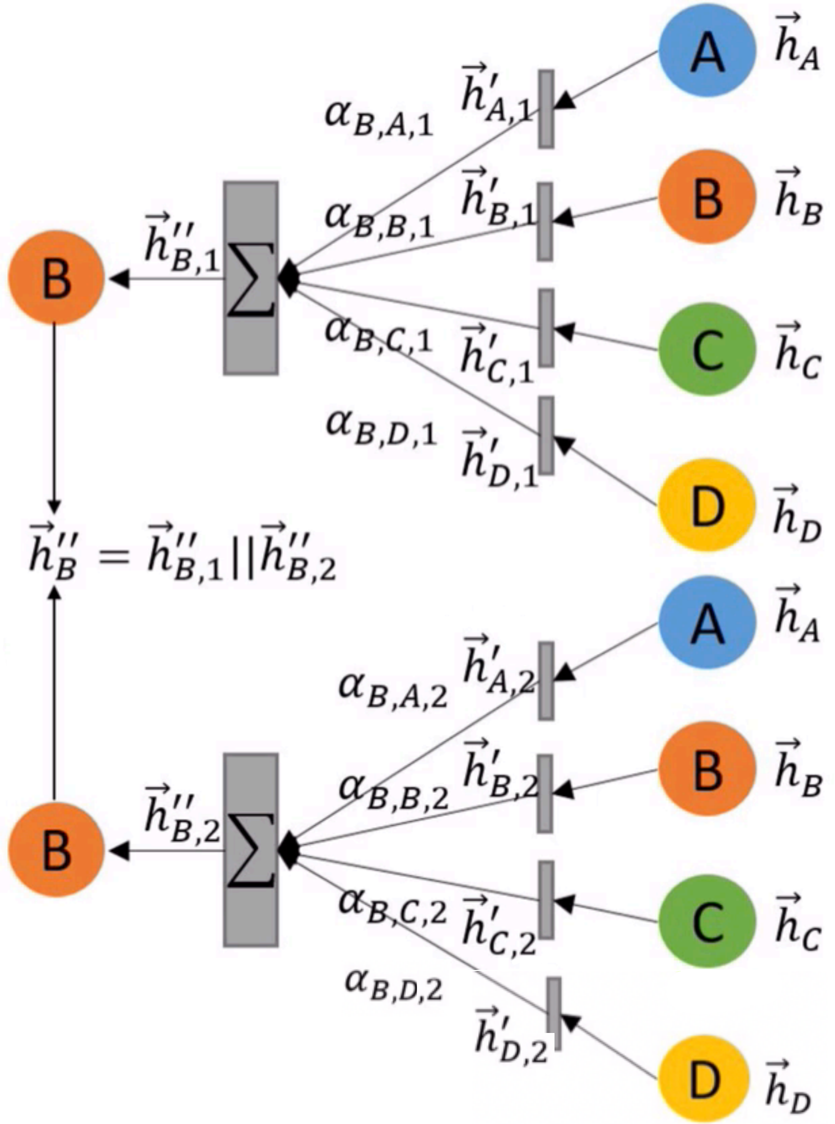


2. Introduction

5. Multi-head attention

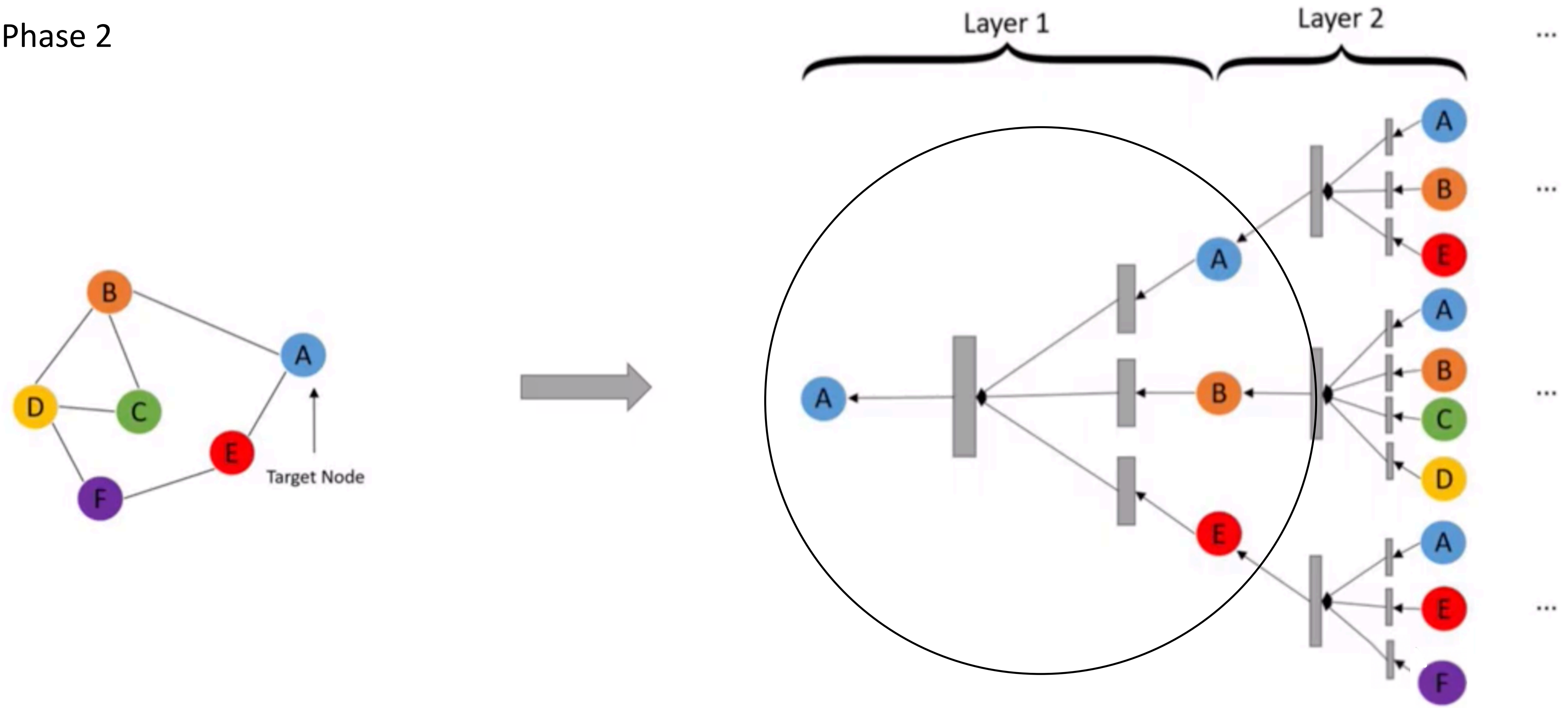
$$\vec{h}''_B = ||_{k=1}^{K} \vec{h}''_{B,k}$$

- Suppose $d(\vec{h}''_B) = 64, K = 2$, then $d(\vec{h}''_{B,k}) = 32$.
- Each "head" is responsible for 32 dimensions.
- The parameters of different "head" are different



2. Introduction

- Phase 2

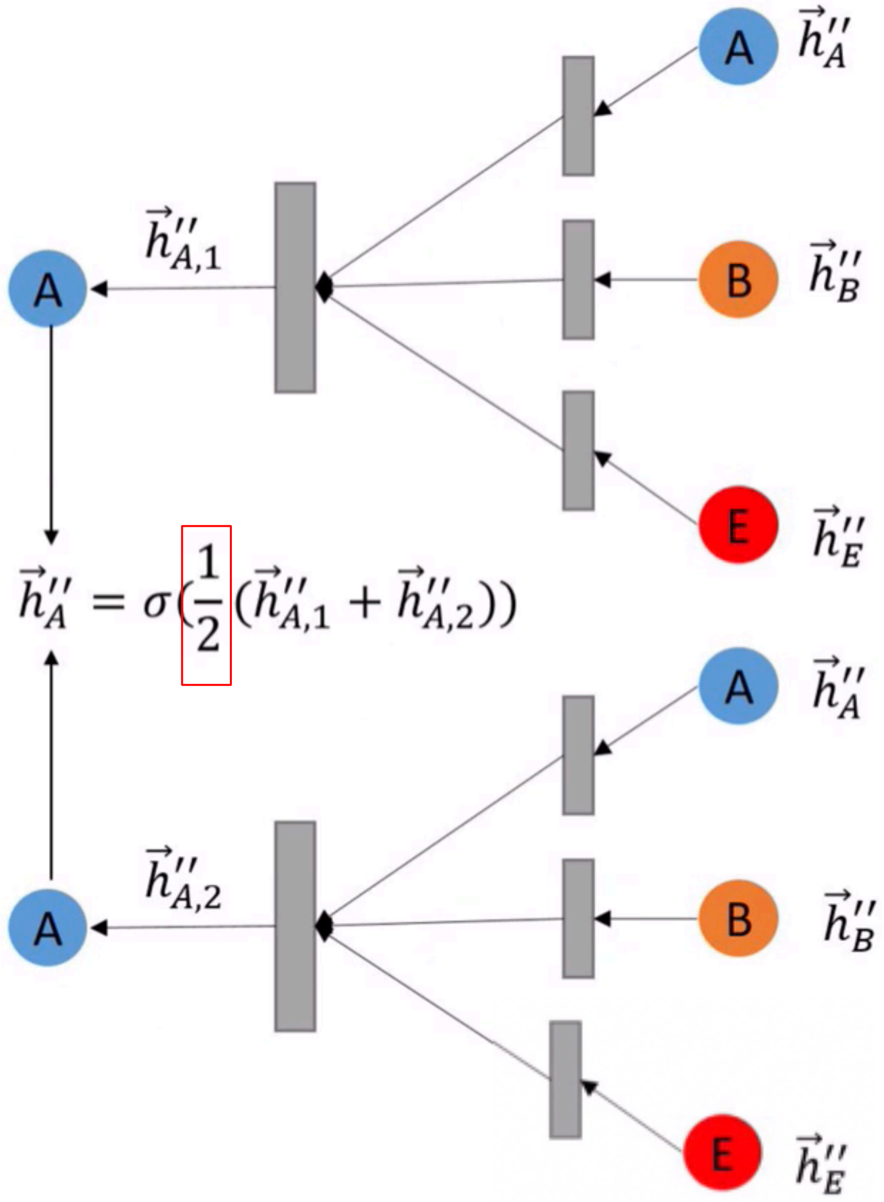


2. Introduction

6. Multi-head attention for final layer

$$\vec{h}''_A = \sigma\left(\boxed{\frac{1}{K}}\sum_k \vec{h}''_{A,k}\right)$$

7. Cross-entropy loss function



3. Results

- Dataset

	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)

3. Results

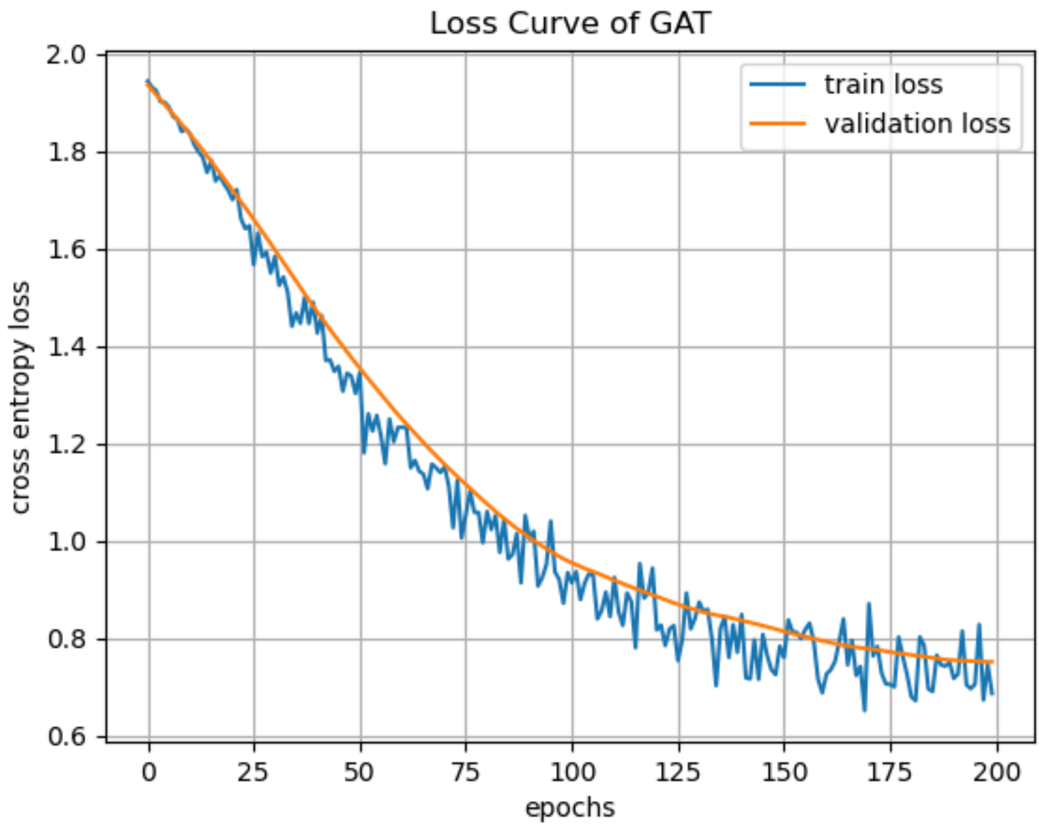
Transductive

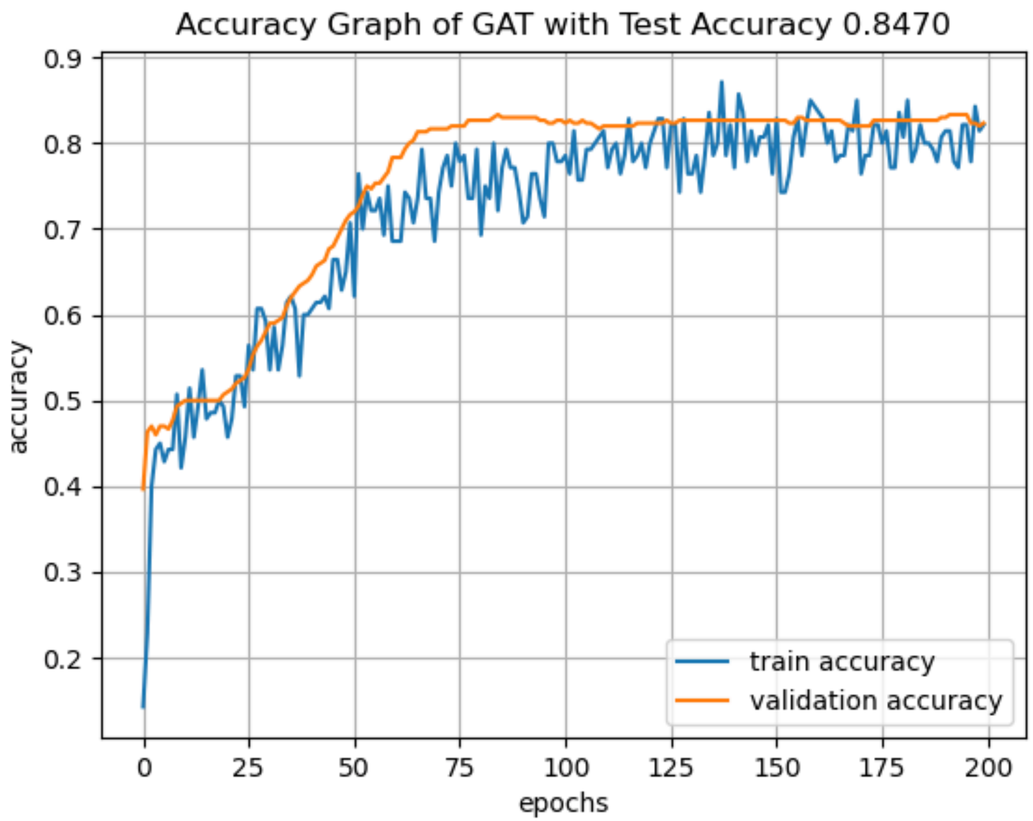
Method	Cora	Citeseer	Pubmed
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 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%	**79.0** $\pm$ 0.3%
GAT (ours)	**83.0** $\pm$ 0.7%	**72.5** $\pm$ 0.7%	**79.0** $\pm$ 0.3%

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 $\pm$ 0.006
GAT (ours)	**0.973** $\pm$ 0.002

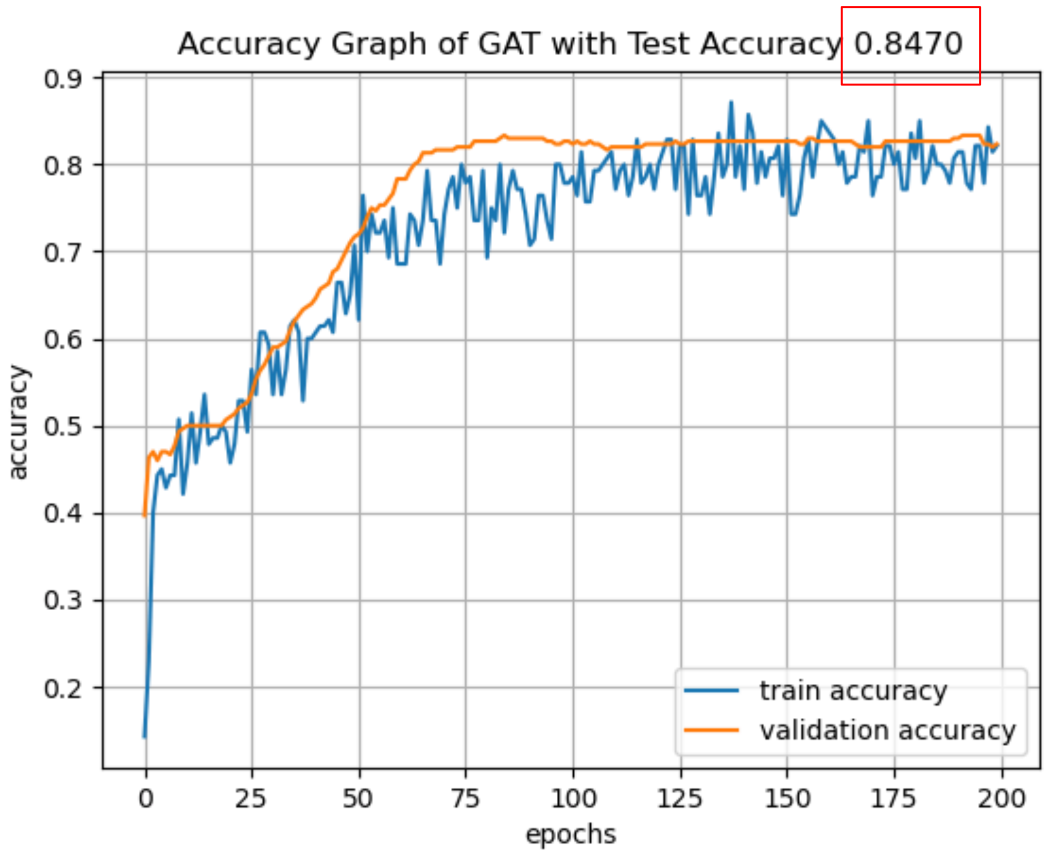
4. Implementation



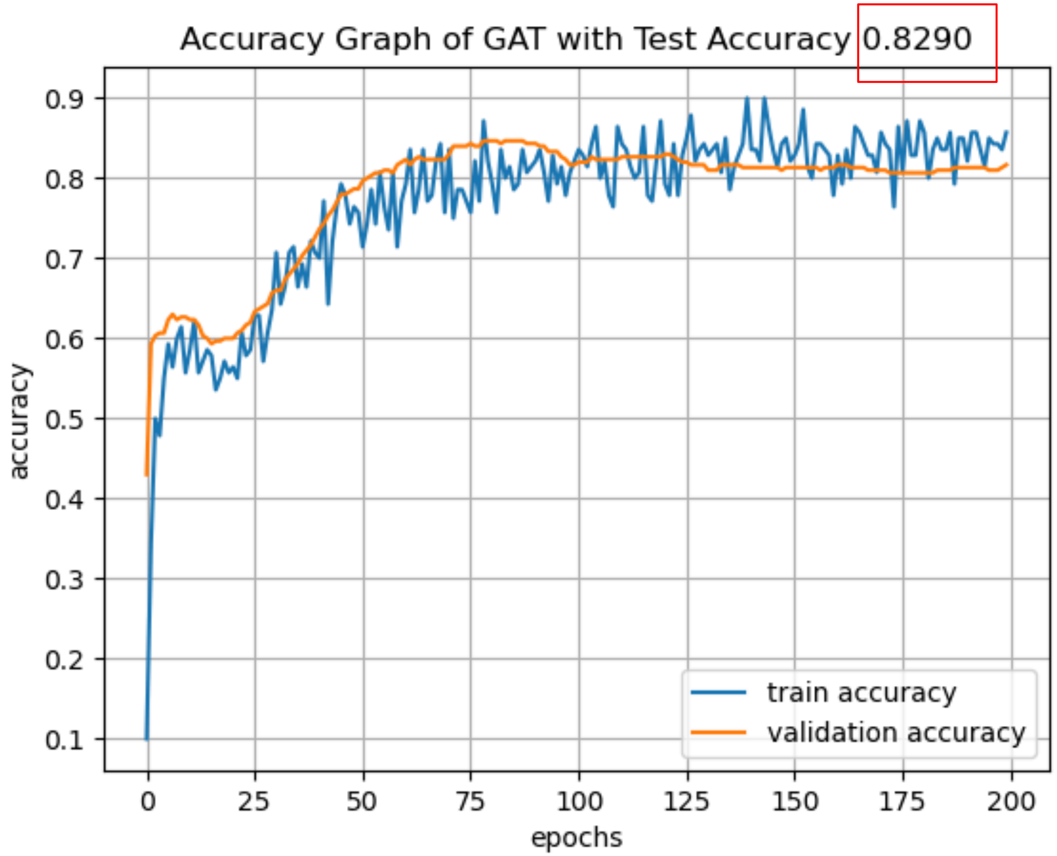


Time per epoch: 0.288s

4. Implementation

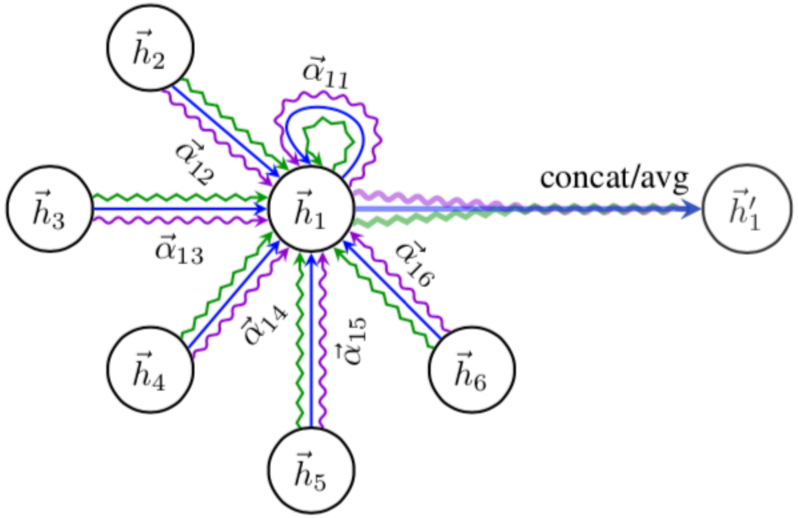


<Including Dropout in Input>



<Excluding Dropout in Input>

5. Key takeaways



- Power of Self-Attention layers, Stacking layers
- Assigning different importances to different nodes
- Computation can be parallelized

5. Discussion

- (-) Handling larger batch size
- (-) Model interpretability
- (-) node classification -> graph classification
- (-) Incorporate edge features
- (-) Rather than Elu?

References

(1) Veličković, Petar, et al. "Graph attention networks." arXiv preprint arXiv:1710.10903 (2017).

(2) Wu, Zonghan, et al. "A comprehensive survey on graph neural networks." IEEE transactions on neural networks and learning systems (2020).

(3) S6 Presenter: Graph Attention Networks by Zhang Ce, https://www.youtube.com/watch?v=6hbWpbi0Z24

(4) Graph Attention Networks, https://www.youtube.com/watch?v=NSjpECvEf0Y&t=2264s