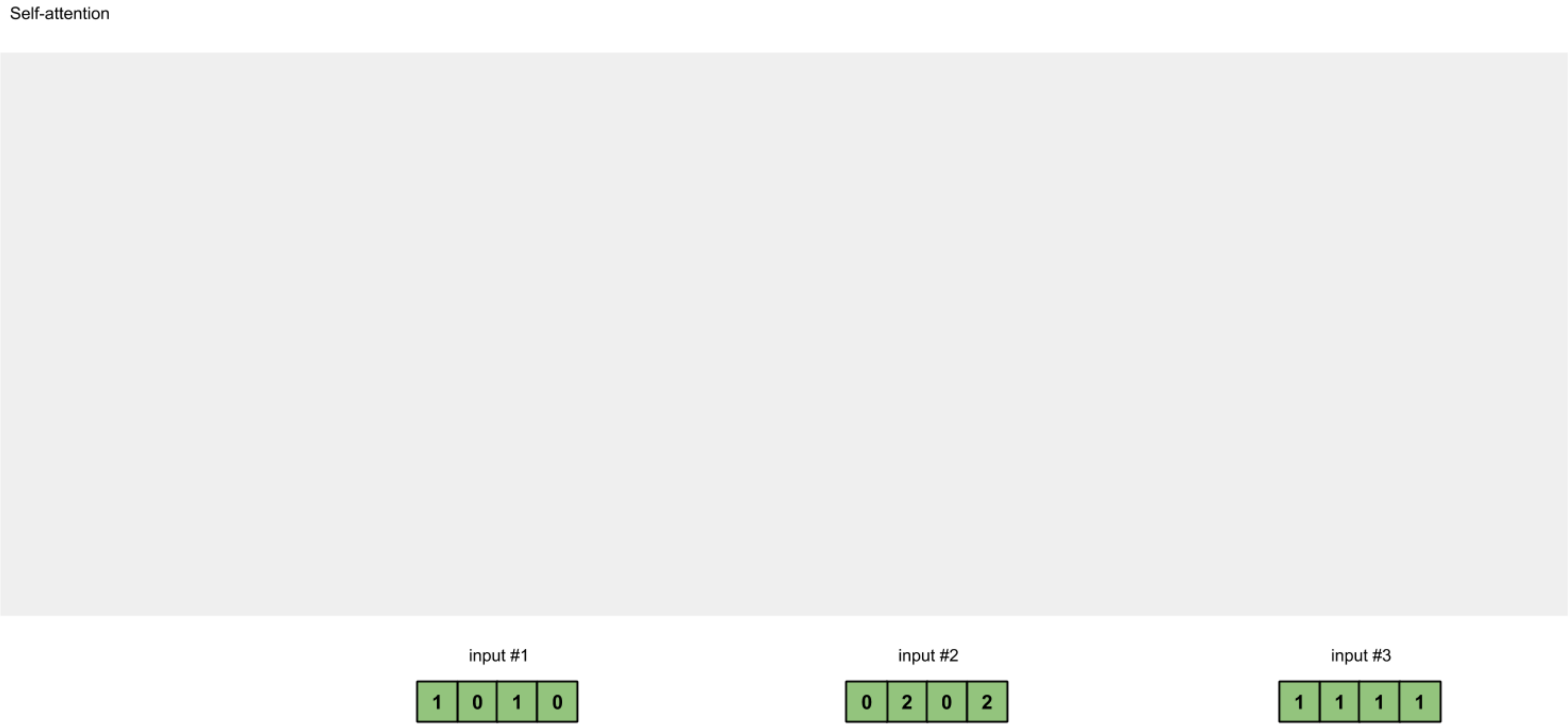
GRAPH ATTENTION NETWORKS(ICLR, 2018)

JuHyeong Kim

SELF-ATTENTION



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

SELF-ATTENTION

Input 1: [1, 0, 1, 0]
Input 2: [0, 2, 0, 2]
Input 3: [1, 1, 1, 1]

Key	Query	Value
[[0, 0, 1],	[[1, 0, 1],	[[0, 2, 0],
[1, 1, 0],	[1, 0, 0],	[0, 3, 0],
[0, 1, 0],	[0, 0, 1],	[1, 0, 3],
[1, 1, 0]]	[0, 1, 1]]	[1, 1, 0]]

these weights are usually small numbers, **initialised randomly** using an appropriate **random distribution like Gaussian, Xavier and Kaiming distributions**. This initialisation is done once before training.

SELF-ATTENTION

Input 1: [1, 0, 1, 0]
Input 2: [0, 2, 0, 2]
Input 3: [1, 1, 1, 1]

×

Dot Product

Key

[[0, 0, 1],
[1, 1, 0],
[0, 1, 0],
[1, 1, 0]]

Query

[[1, 0, 1],
[1, 0, 0],
[0, 0, 1],
[0, 1, 1]]

Value

[[0, 2, 0],
[0, 3, 0],
[1, 0, 3],
[1, 1, 0]]

SELF-ATTENTION

(Input 1: [1, 0, 1, 0] × Query) × (input 1 × Key)
(input 2 × Key)
(input 3 × Key)

score 1

(Input 1: [1, 0, 1, 0] × Query) × (input 1 × Key)
(input 2 × Key)
(input 3 × Key)

score 2

(Input 1: [1, 0, 1, 0] × Query) × (input 1 × Key)
(input 2 × Key)
(input 3 × Key)

score 3

softmax

score1(softmax) ∗
(input 1 × value)

score2(softmax) ∗
(input 2 × value)

score3(softmax) ∗
(input 3 × value)

sum

attention
output 1

SELF-ATTENTION

attention output 1

attention output 2

attention output 3

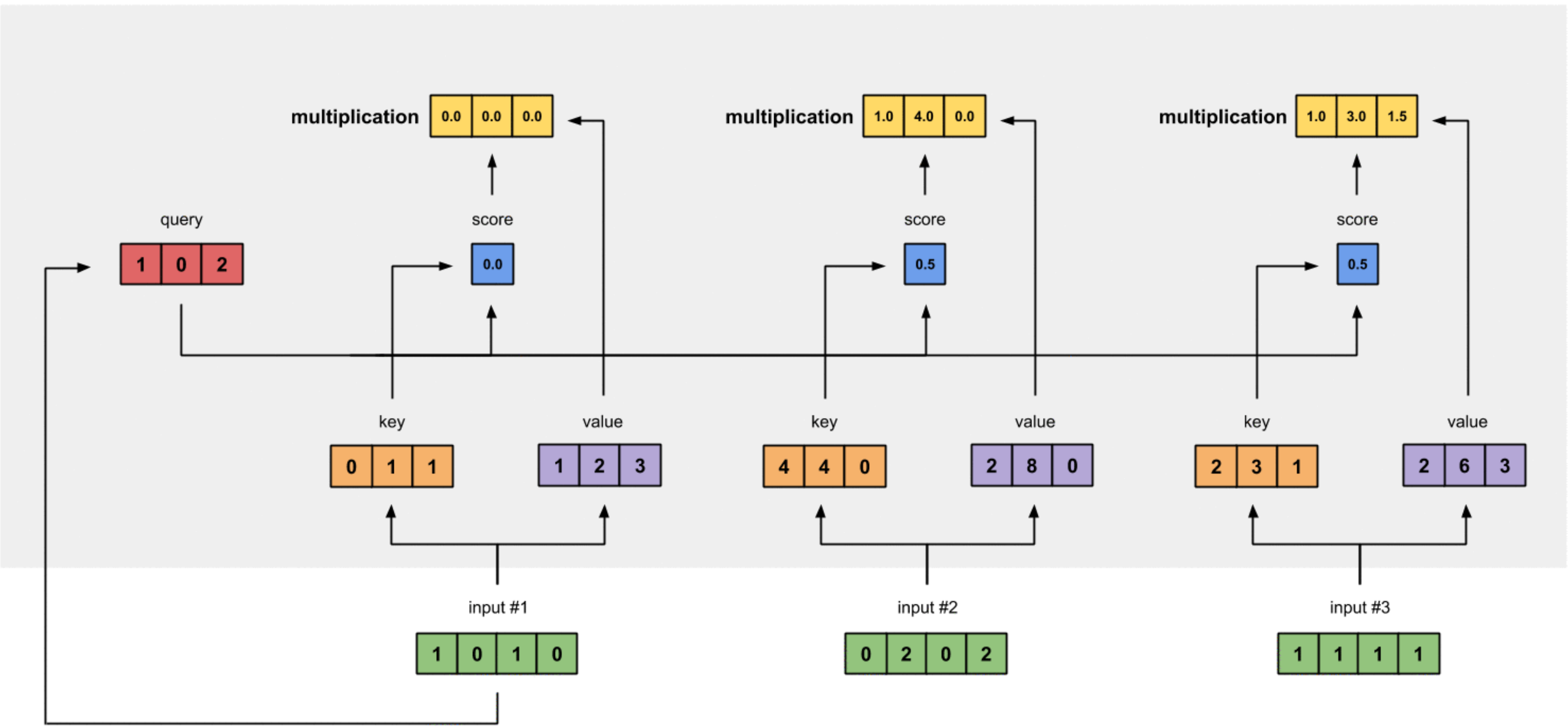
operation

result

https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

SELF-ATTENTION

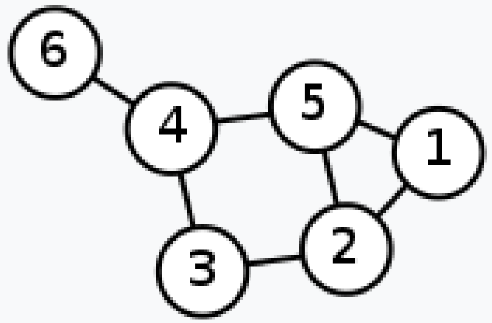
Self-attention



https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

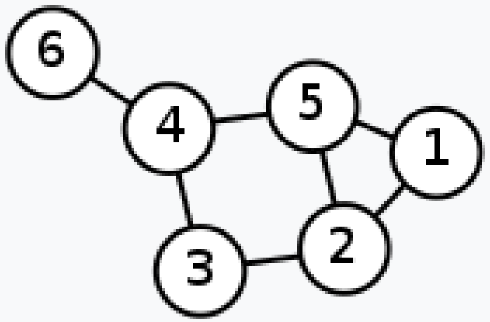
INTRODUCTION

- this paper introduce an **attention-based architecture** to perform node **classification of graph-structured data**.
- The idea is to **compute the hidden representations of each node in the graph**, **by attending** over its neighbors, **following a self-attention strategy.**

Labeled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

INTRODUCTION

- The attention architecture has several interesting properties
  - (1) the operation is **efficient**, since it is **parallelizable across node neighbor pairs**;
  - (2) it can be **applied to graph nodes having different degrees by specifying arbitrary weights** to the neighbors;
  - (3) **the model is directly applicable to inductive learning problems,** including tasks where the model has to generalize to completely unseen graphs.

Labeled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

GAT ARCHITECTURE - METHOD DEFINITION

- input : set of node features

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

  - **N** is the number of nodes, and **F** is the number of features in each node

- output : when produces a new set of node features in layer

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

GAT ARCHITECTURE - METHOD DEFINITION

- as an initial step, a shared linear transformation, parametrized by a weight matrix
  - $\mathbf{W} \in \mathbb{R}^{F' \times F}$ is applied to every node
- attentional mechanism computes attention coefficients

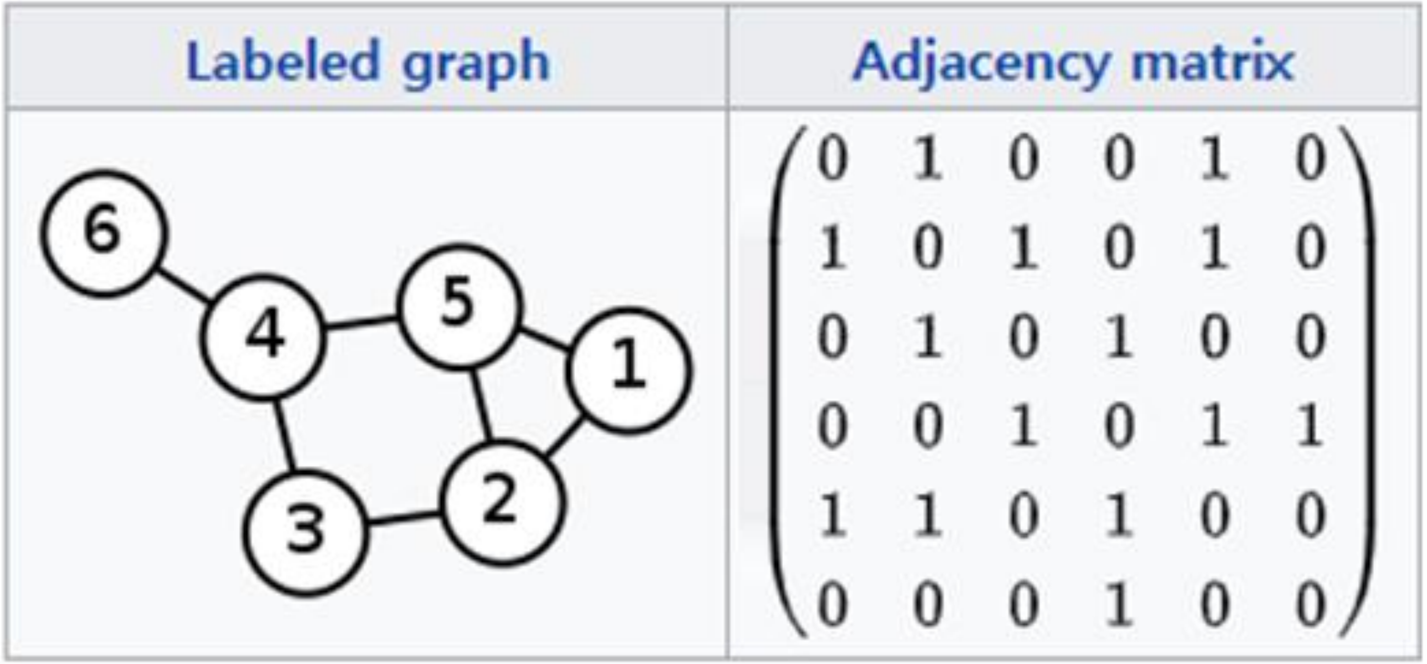
$$e_{ij} = a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$$

masked attention

attention mechanism

GAT ARCHITECTURE - METHOD DEFINITION

- In its most general formulation, the model allows **every node to attend on every other node**, **dropping all structural information**.
- that is masked attention
  - This means that the e_{ij} value is calculated only for edges where connections exist on the Adjacency matrix.

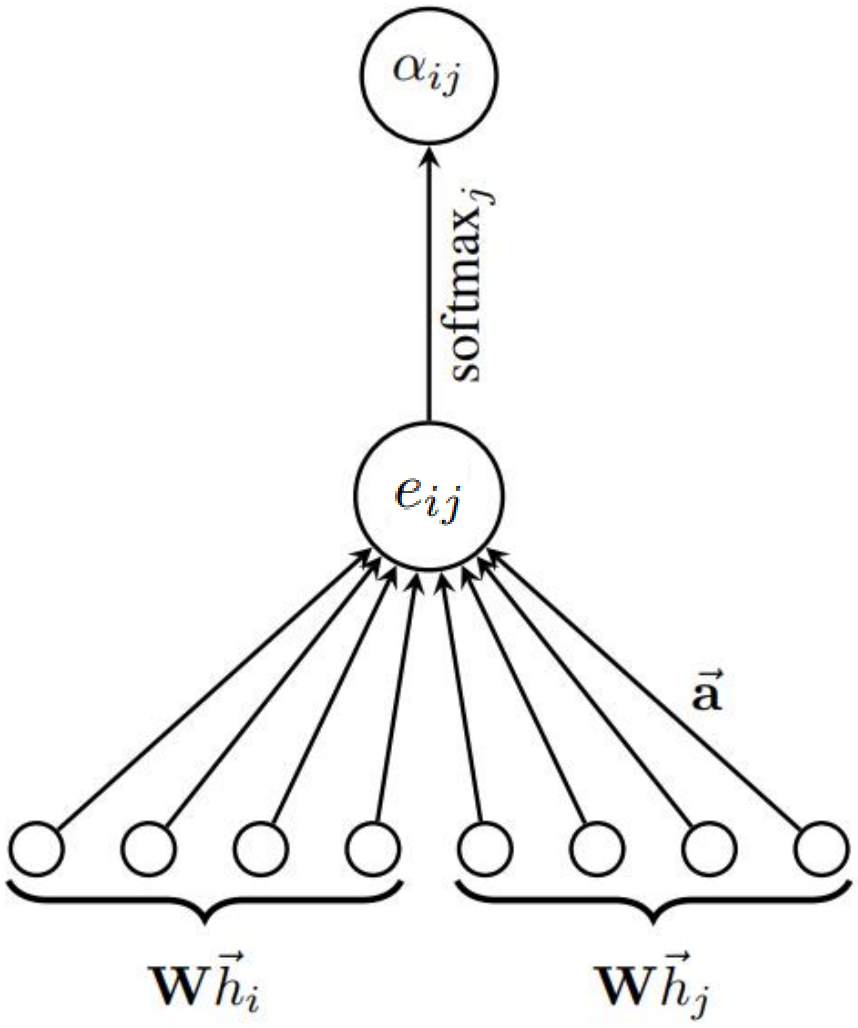


$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

GAT ARCHITECTURE - METHOD DEFINITION

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

normalize



GAT ARCHITECTURE - METHOD DEFINITION

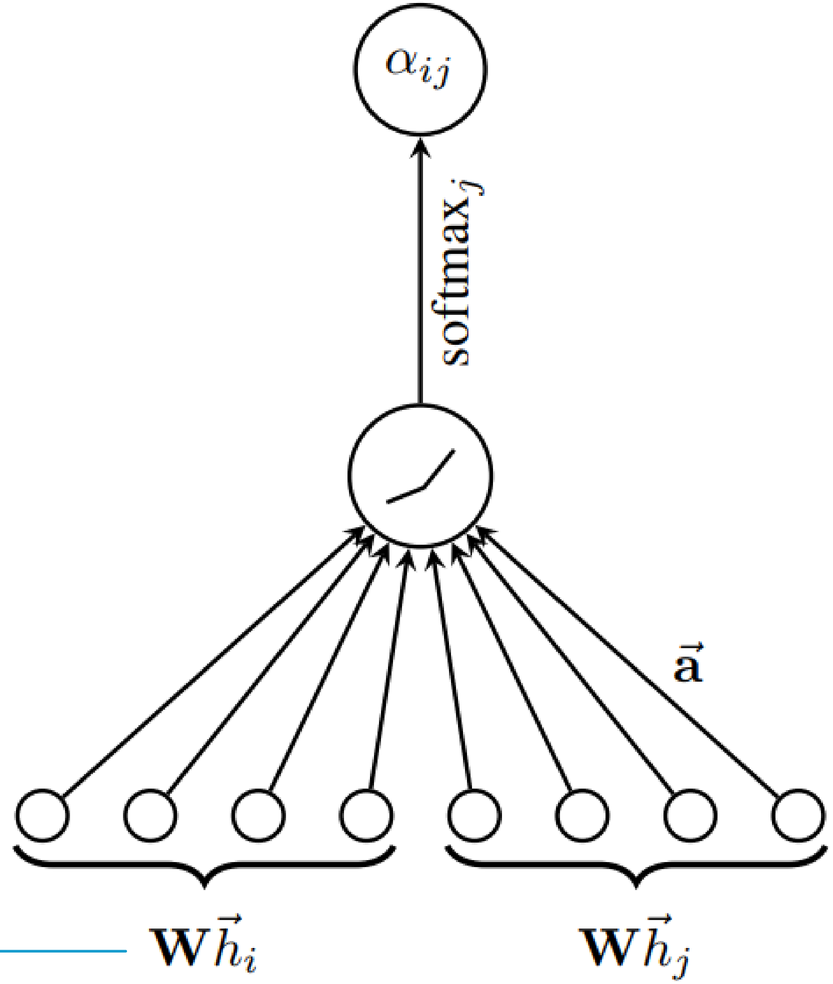
- α_{ij} : normalized attention

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

concatenation operation

nonlinearity (0.2)

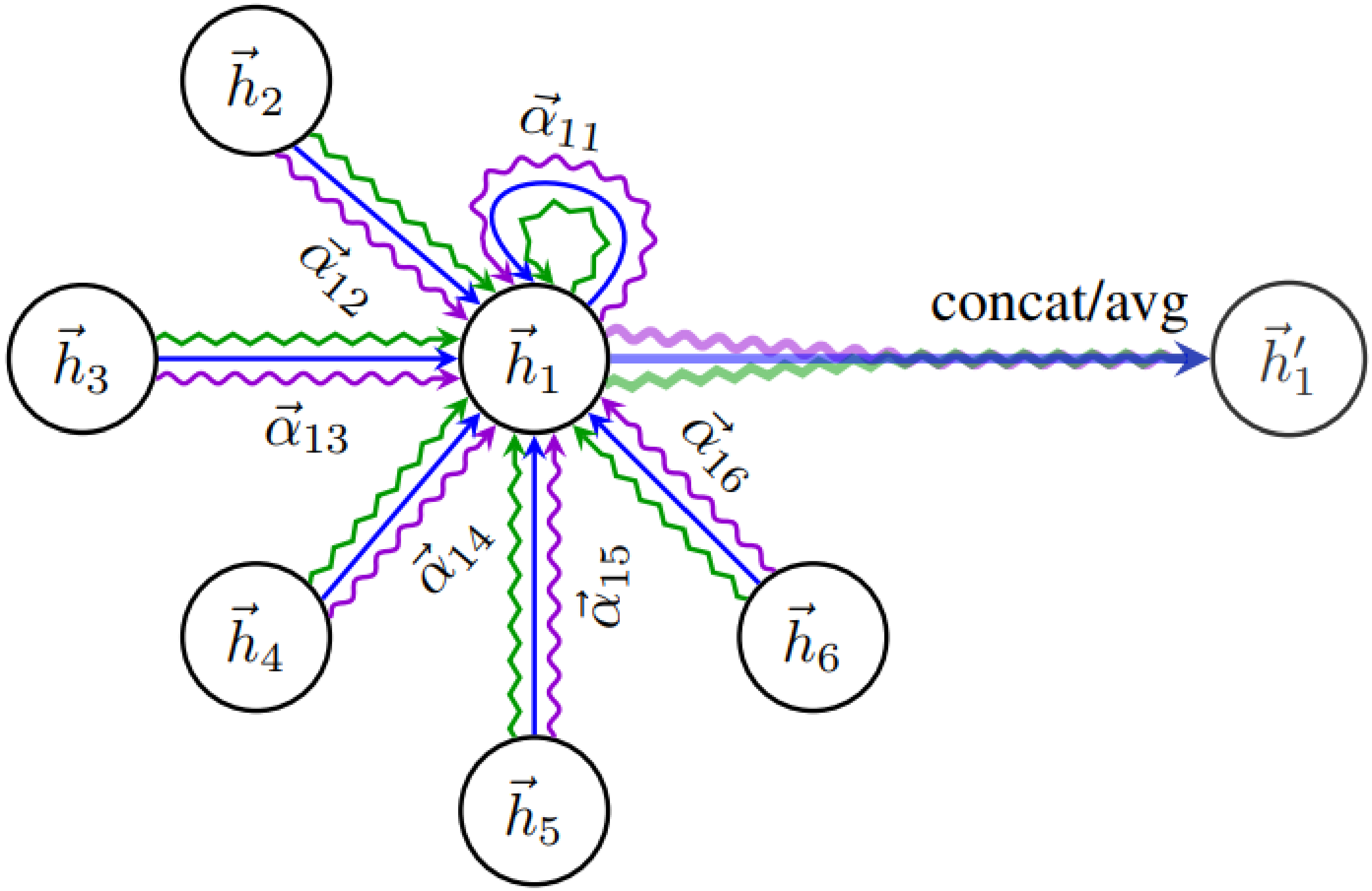
$\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$



GAT ARCHITECTURE - METHOD DEFINITION

- $\overrightarrow{h'}_i$: output

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



GAT ARCHITECTURE - SUMMARY

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

masked attention mechanism

normalization for attention

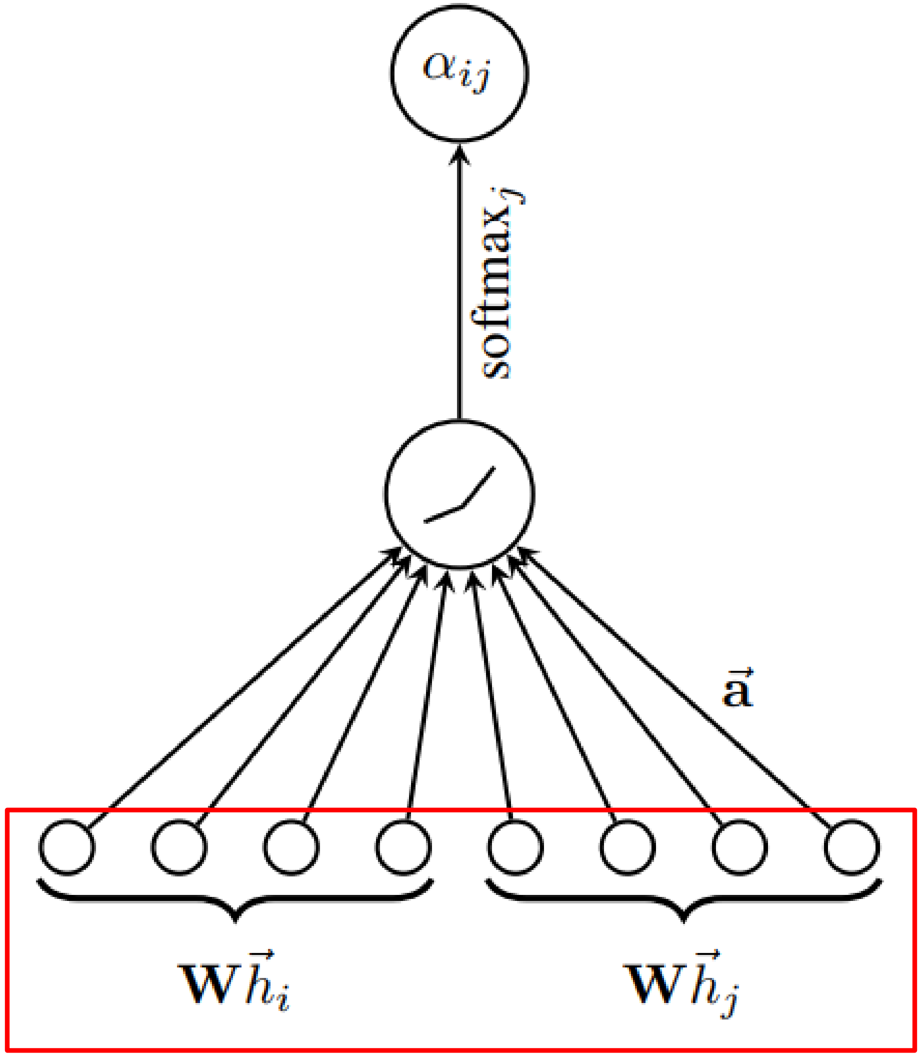
$$\alpha_{ij} = \mathrm{softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$$

Detail

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

GAT ARCHITECTURE - SUMMARY

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



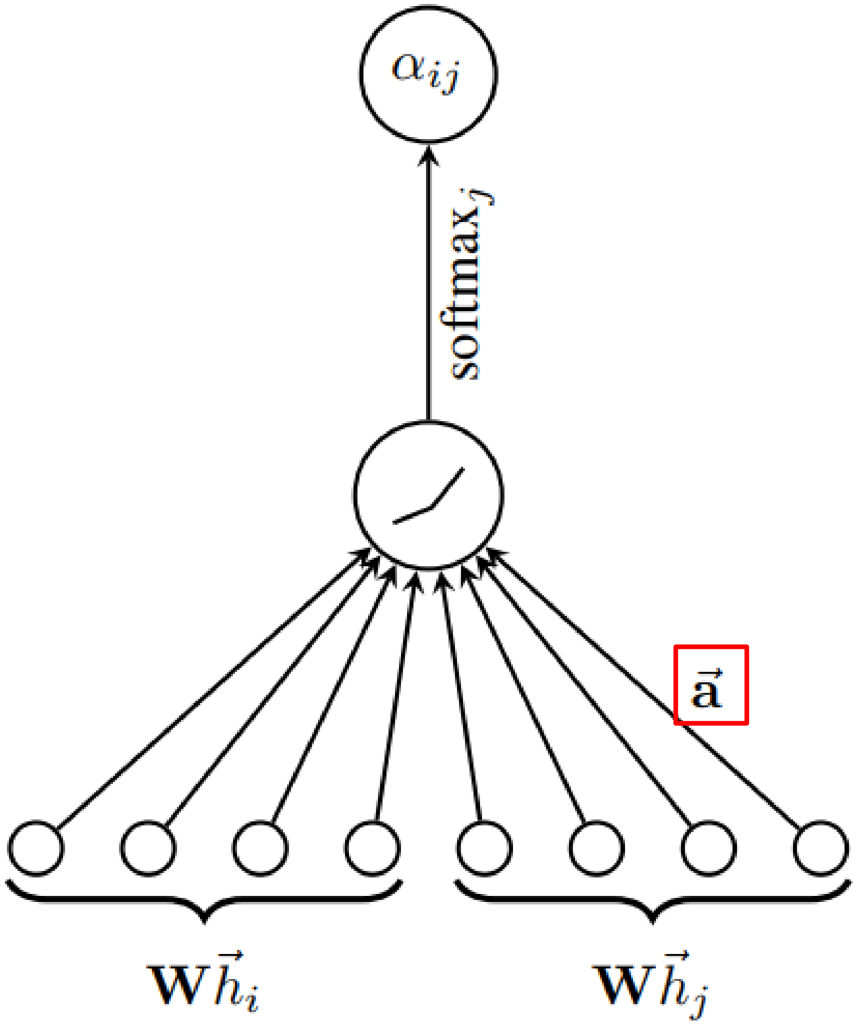
concatenation $\mathbf{W}\overrightarrow{h_i}$ and $\mathbf{W}\overrightarrow{h_j}$

$\mathbf{F} + \mathbf{F}$

GAT ARCHITECTURE - SUMMARY

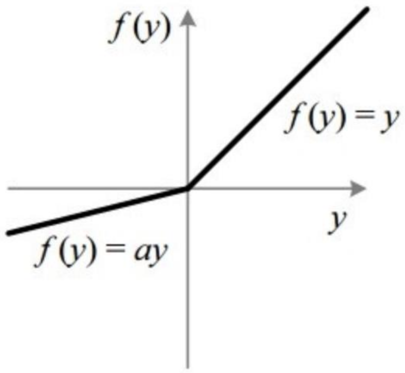
$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\boxed{\vec{\mathbf{a}}^T}[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\boxed{\vec{\mathbf{a}}^T}[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$

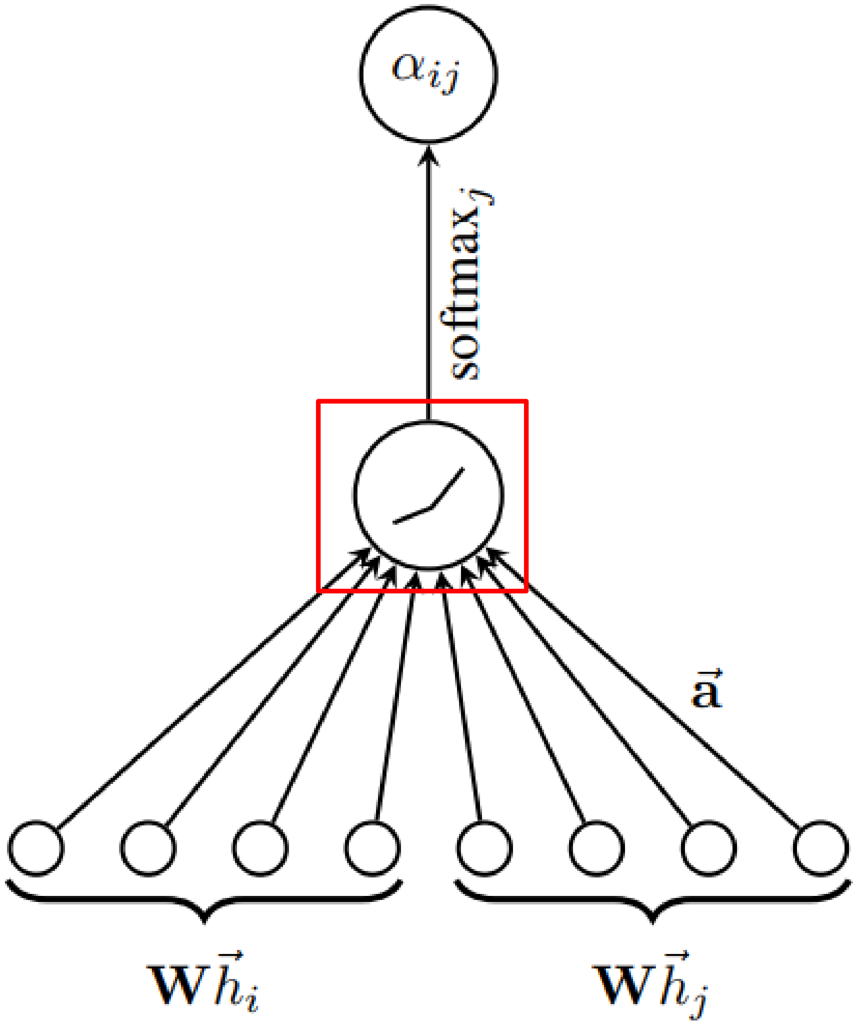


GAT ARCHITECTURE - SUMMARY

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



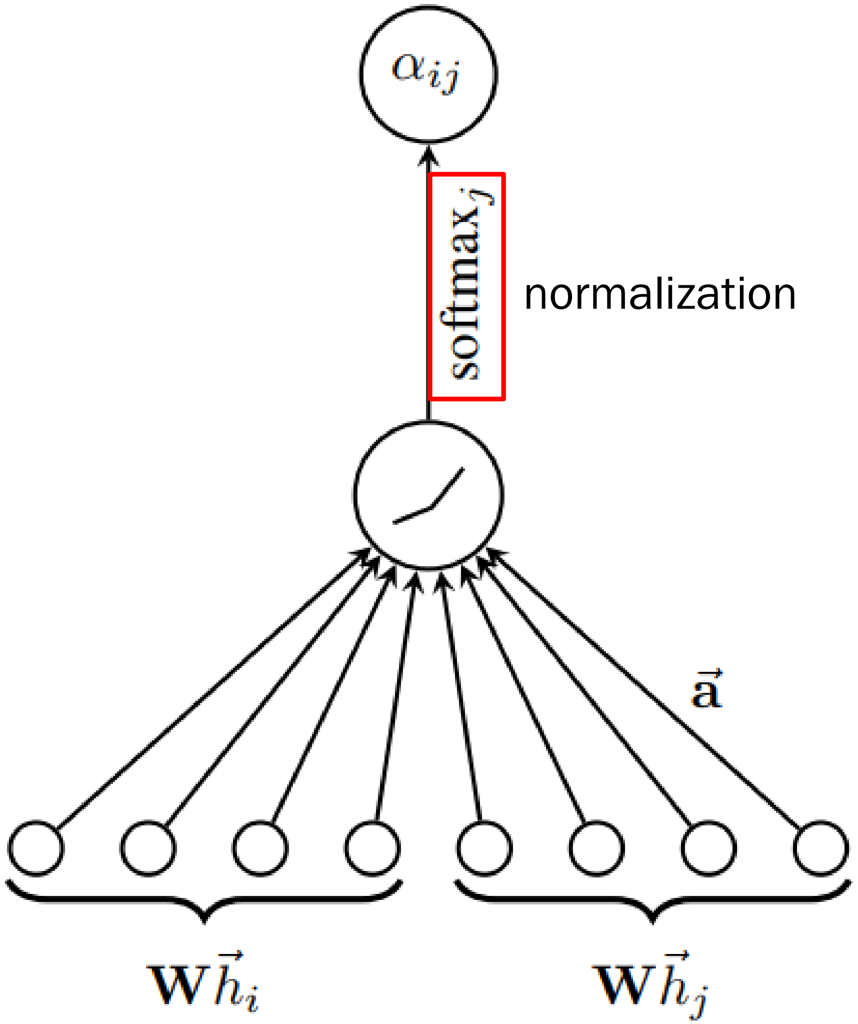
LeakyReLU $f(x) = \begin{cases} x, & x \geq 0 \\ 0.2x, & x < 0 \end{cases}$



GAT ARCHITECTURE - SUMMARY

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

normalization

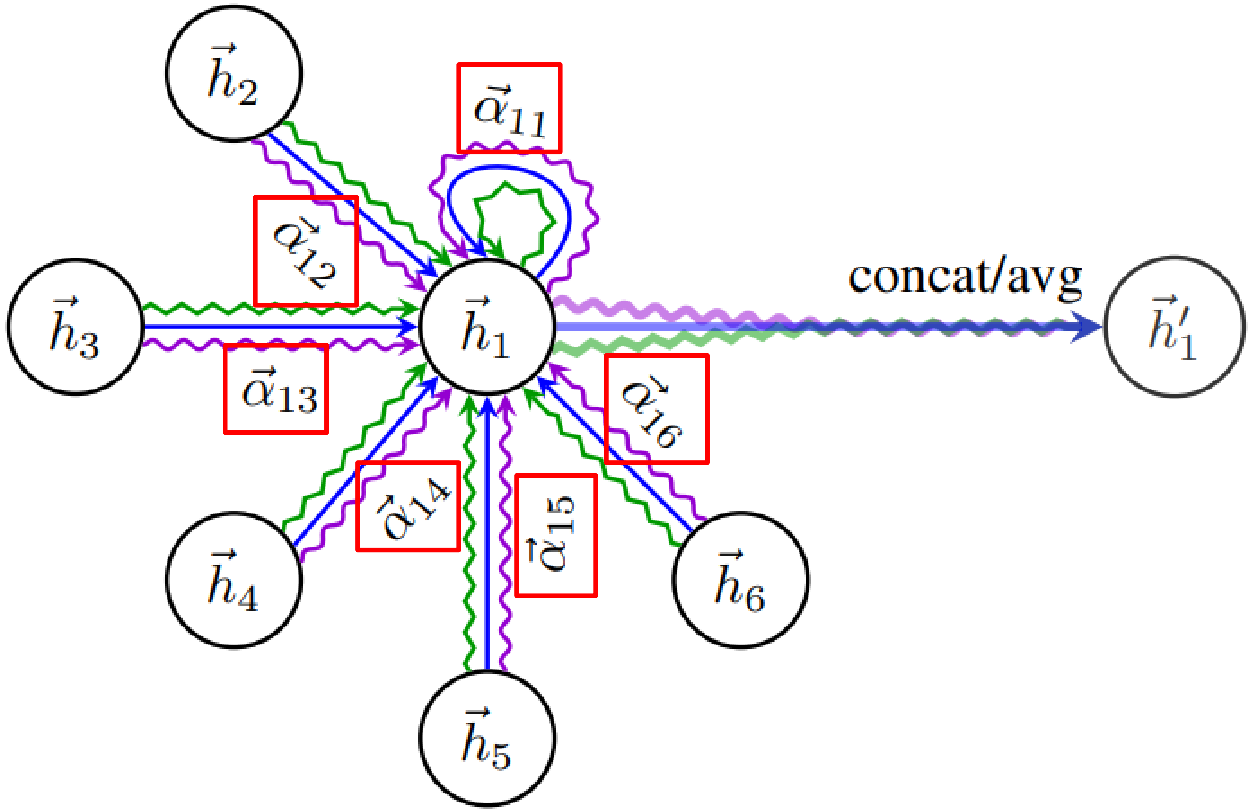


GAT ARCHITECTURE - SUMMARY

- $\overrightarrow{h'}_i$: output

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

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

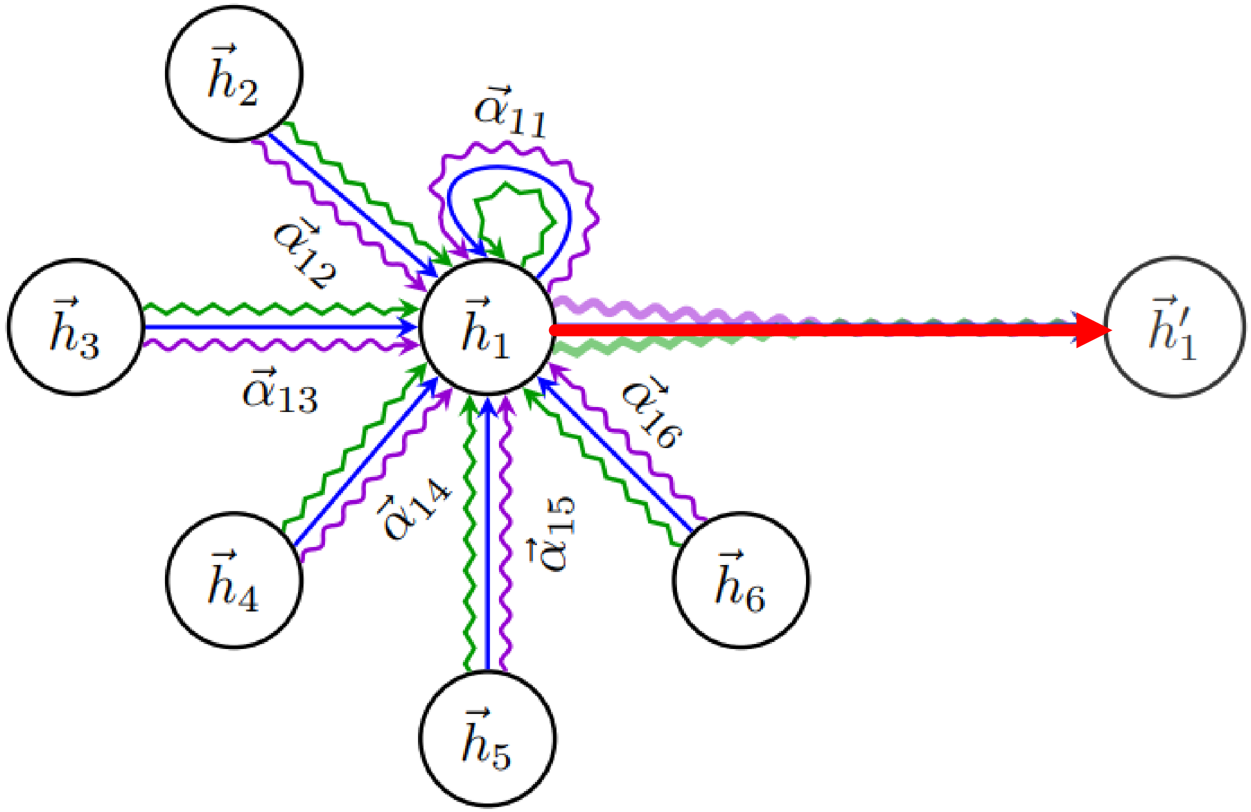


GAT ARCHITECTURE - SUMMARY

- $\overrightarrow{h'}_i$: output

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

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



GAT ARCHITECTURE – MULTI-HEAD ATTENTION

- Multi-head attention
  - **Query, Key** and **Value** are different attention values(independent).
  - **K** is the number of Multi-head attention
- Then, output is follows:

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

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

averaging and
delay applying the final nonlinearity

GAT ARCHITECTURE – RESULT

- Dataset – Cora, Citeseer, Pubmed, PPI

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)

GAT ARCHITECTURE – RESULT

- Summary of results in terms of **classification accuracy**

Table 2: Summary of results in terms of classification accuracies, for Cora, Citeseer and Pubmed. GCN-64* corresponds to the best GCN result computing 64 hidden features (using ReLU or ELU).

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 ± 0.5%	—	78.8 ± 0.3%
GCN-64*	81.4 ± 0.5%	70.9 ± 0.5%	**79.0** ± 0.3%
GAT (ours)	**83.0** ± 0.7%	**72.5** ± 0.7%	**79.0** ± 0.3%

GAT ARCHITECTURE – RESULT

- Summary of results in terms of **micro-averaged F1-scores**

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)	$\mathbf{0.973} \pm 0.002$

GAT ARCHITECTURE – RESULT

- it's pre-trained GAT model for Cora dataset

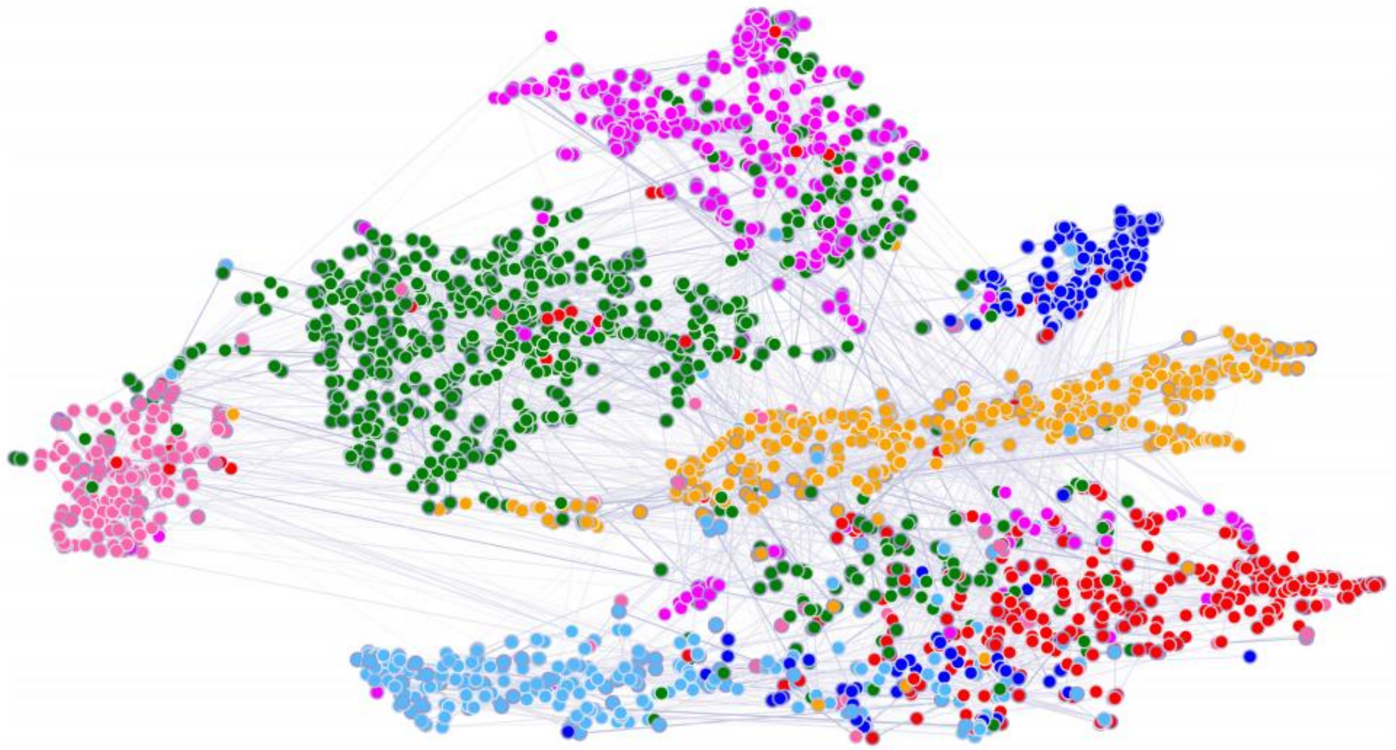


Figure 2: A t-SNE plot of the computed feature representations of a pre-trained GAT model's first hidden layer on the Cora dataset. Node colors denote classes. Edge thickness indicates aggregated normalized attention coefficients between nodes i and j, across all eight attention heads $(\sum_{k=1}^{K} \alpha_{ij}^{k} + \alpha_{ji}^{k})$.