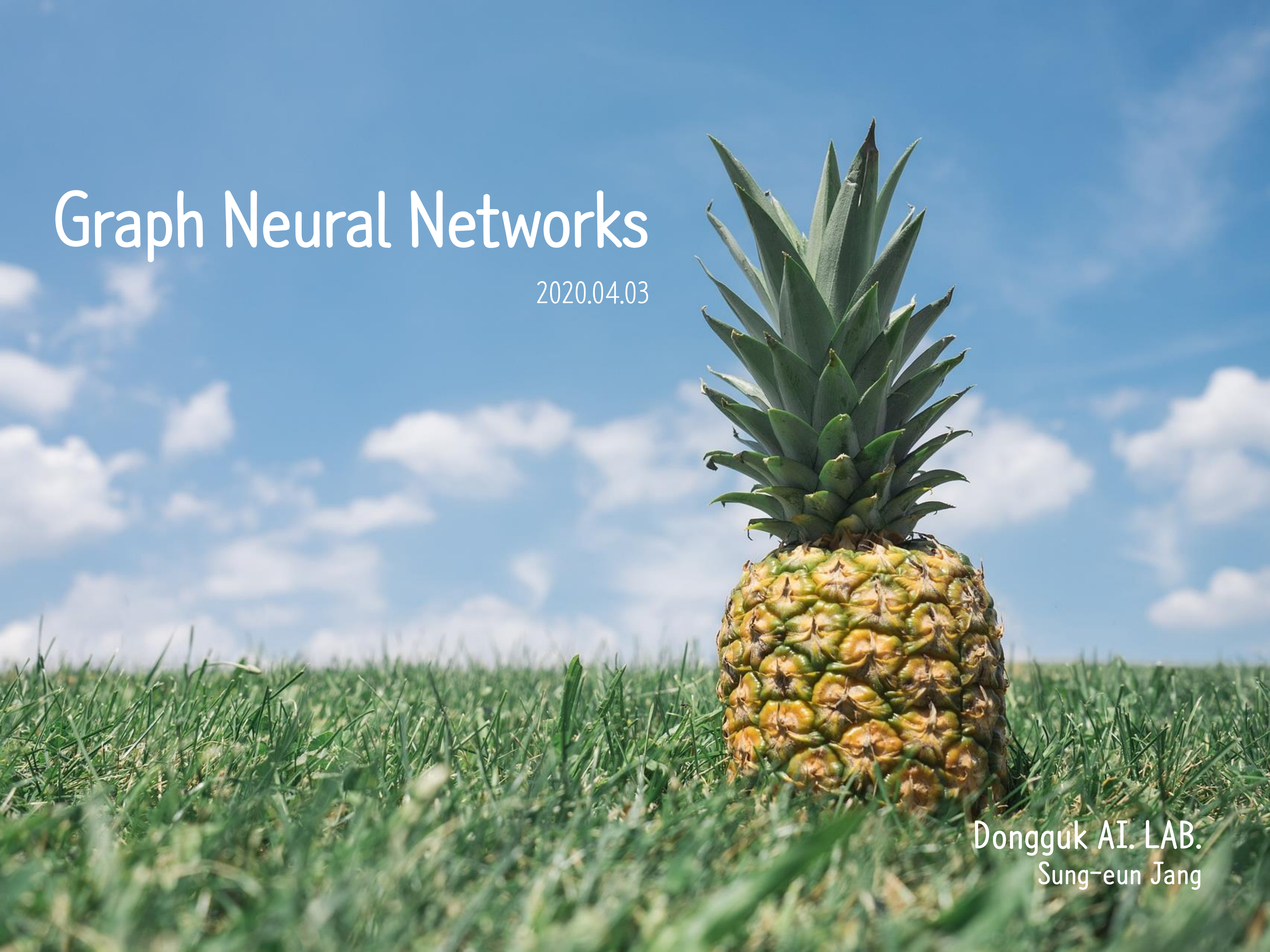


Graph Neural Networks

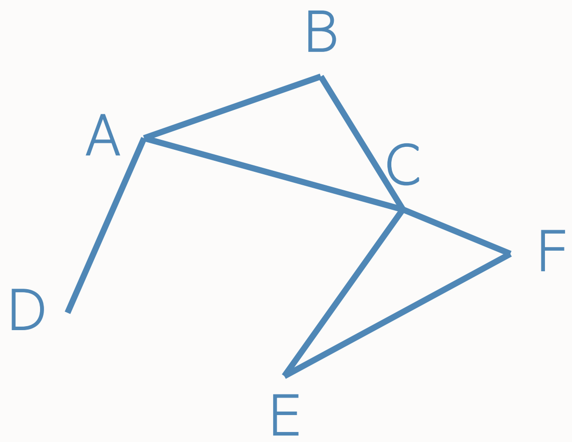
2020.04.03

Dongguk AI. LAB.
Sung-eun Jang

Graph

Definition of Graph

- Graph
  - A collection of objects where some pairs of objects are connected by links

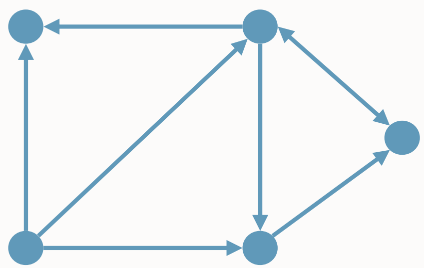


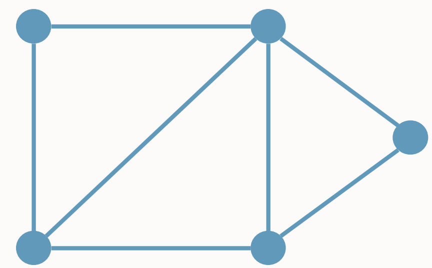
- Components of a Graph
  - Objects nodes, vertices N
  - Interactions links, edges E
  - System network, graph $G(N,E)$

Graph

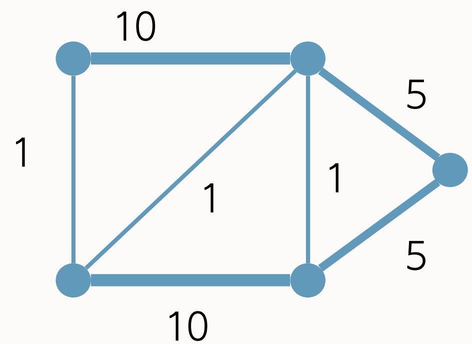
Structure of Graph

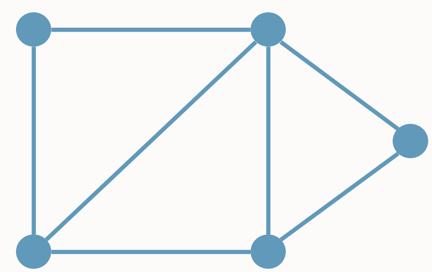
- Directed / Undirected





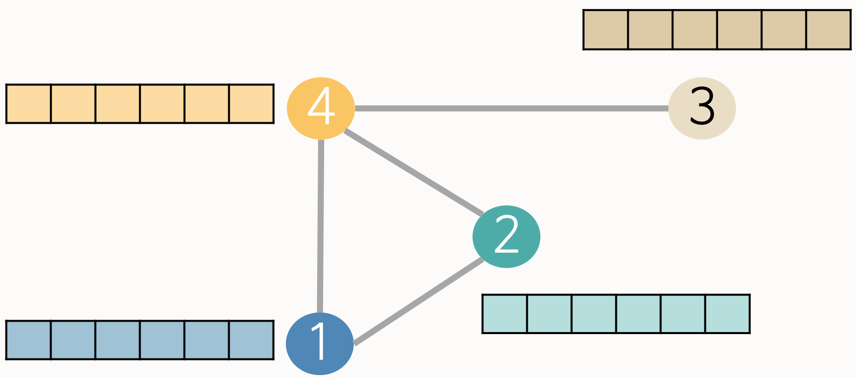
- Weighted / Unweighted



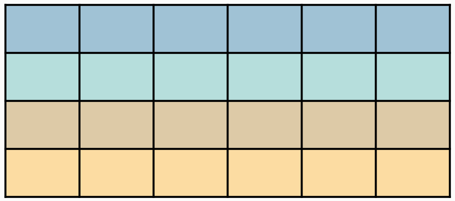


Graph

Graph Representation



Node-feature matrix



Adjacency matrix

0	1	0	1
1	0	0	1
0	0	0	1
1	1	1	0

$$A = \begin{cases} A_{ij} = 1 \; if \; there \; is \; edge \\ A_{ij} = 0 \; if \; there \; is \; no \; edge \end{cases}$$

Degree matrix

2	0	0	0
0	2	0	0
0	0	3	0
0	0	0	1

$$D_{ii} = \sum_{i \sim j} A_{ij}$$

Laplacian matrix

2	−1	0	−1
1	2	0	−1
0	0	3	−1
−1	−1	−1	2

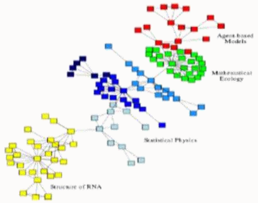
$$L = A - D$$

Graph

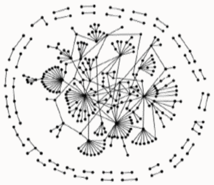
Why Graph?

- Universal Language for describing complex data
- Shared vocabulary between fields
- Data availability & computational challenges

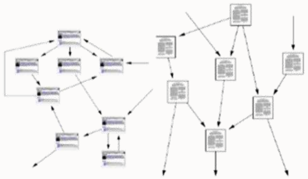
Social networks



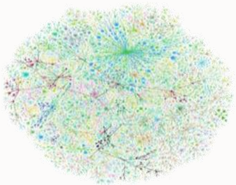
Economic networks



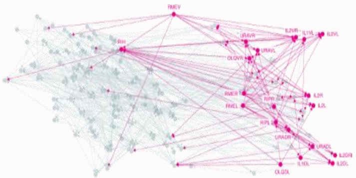
Communication networks



Information networks:
Web & citations



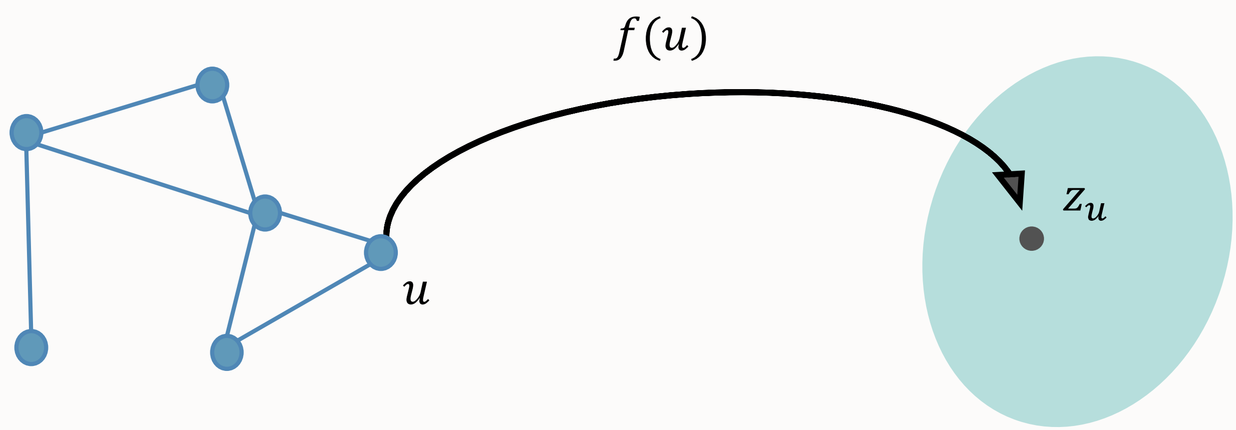
Internet



Networks of neurons

Node Embeddings

Definition.



- Node embedding function (f) map each node in a graph into a low-dimensional space
- similarity in the embedding space approximates similarity in the graph
- Could be helpful with tasks such as…

  ➔ node classification, graph classification, link prediction…

Node Embeddings

Shallow Encoders

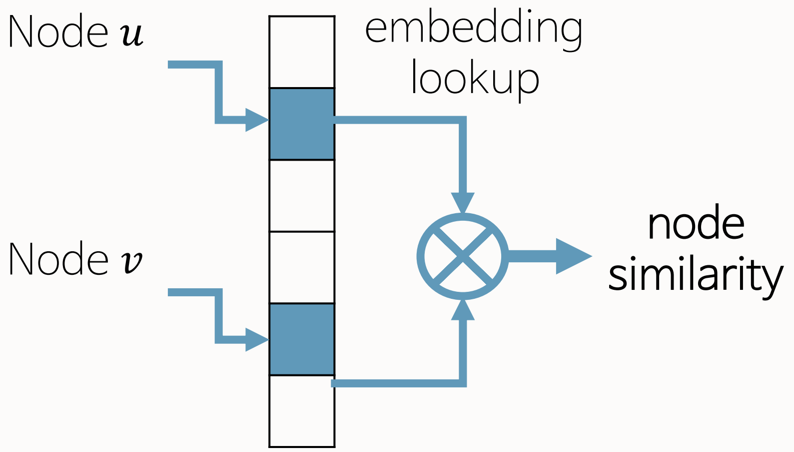
- Use a lookup table consisting of unique embedding of every nodes in graph

- Limitations

  - O(|V|) parameters are needed

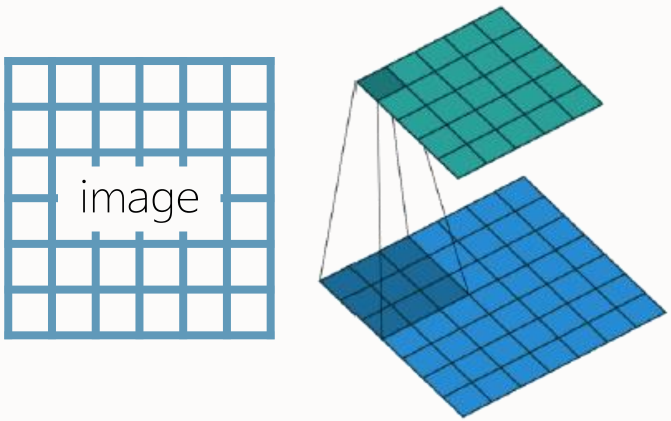
  - Inherently Transductive

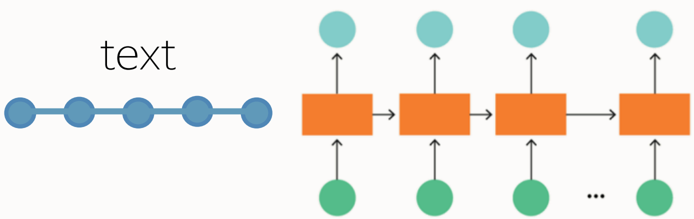
  - Do not incorporate node features



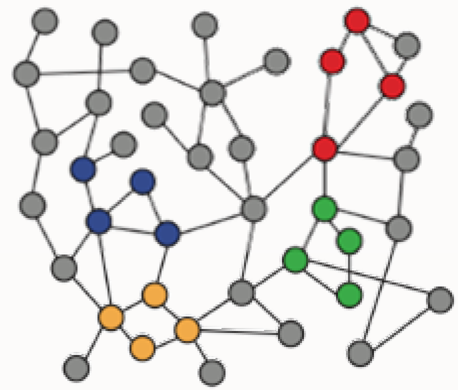
Node Embeddings

Graph and Modern Deep Learning





graph

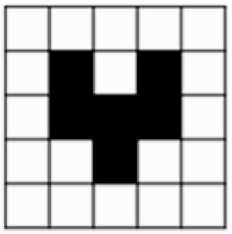


- Arbitrary size
- Complex topological structure
- No fixed node ordering
- Dynamic and multimodal features

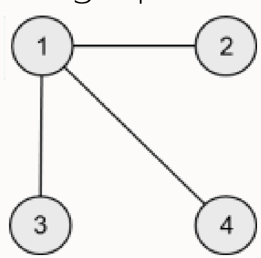
Graph Convolutional Network

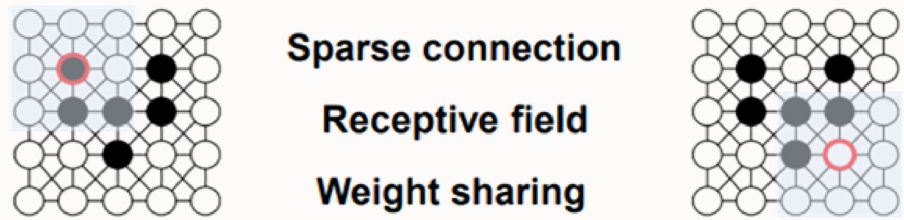
From Image to Graph

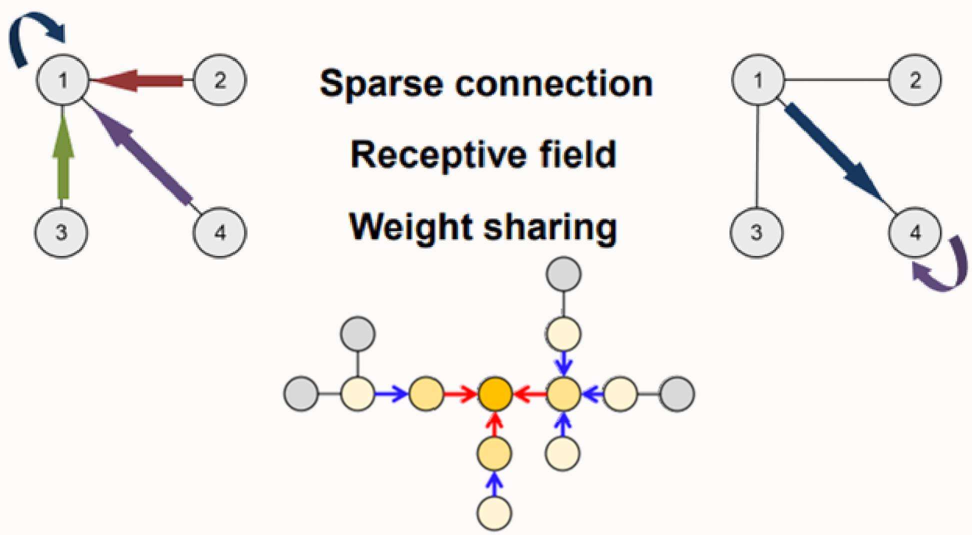
image



graph



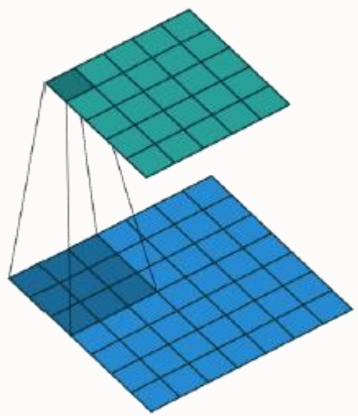




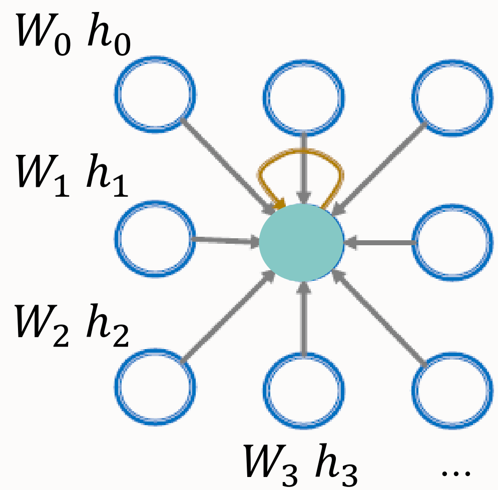
Graph Convolutional Network

GCN

image



graph



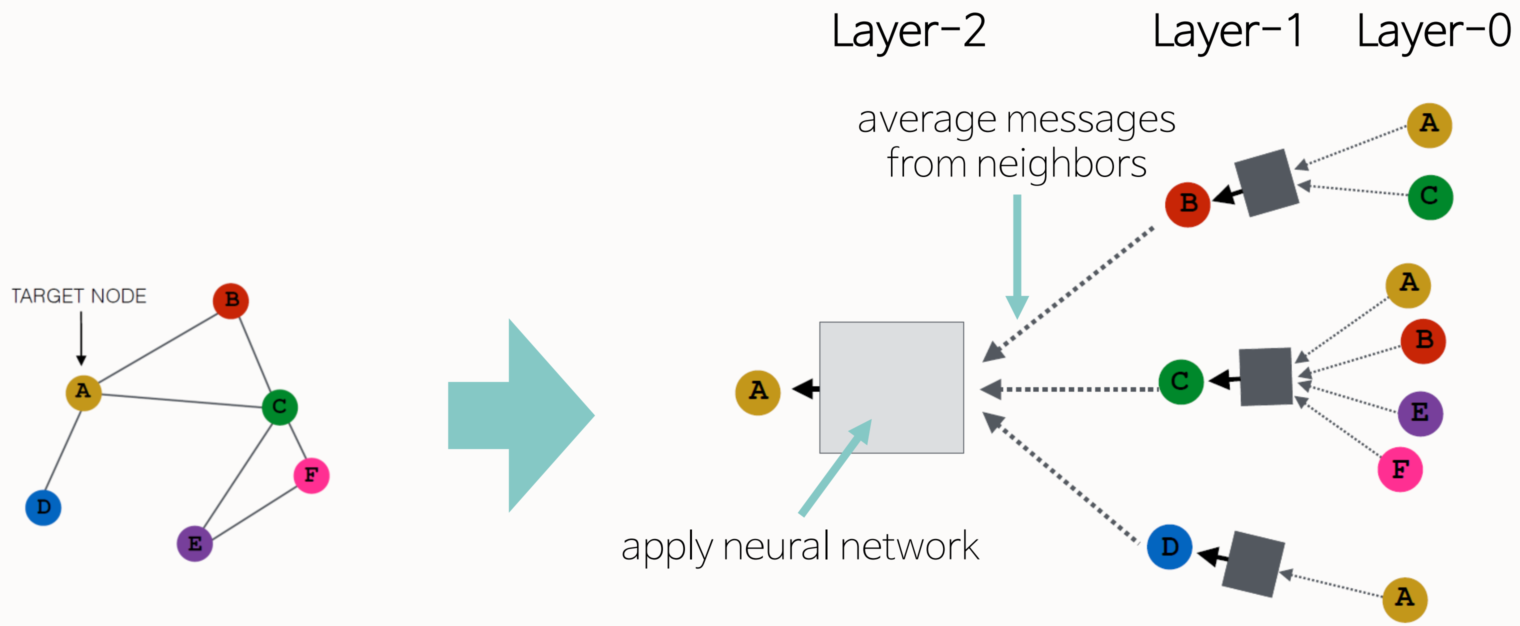
$$\sum_i W_i\ h_i$$

- Transform information at the neighbors and combine it

  1) Transform "messages" h_i from neighbors:
  2) Add them up

Graph Convolutional Network

Compute Node Features

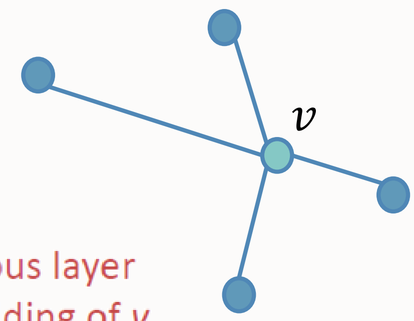


- Generate node embeddings based on local network neighborhoods

- Nodes have embeddings at each layer, repeating combine messages from their neighbor using neural networks

Graph Convolutional Network

Compute Node Features



Initial 0-th layer embeddings are equal to node features

$$\boxed{\mathbf{h}_v^0 = \mathbf{x}_v}$$

Previous layer embedding of *v*

$$\mathbf{h}_v^k = \boxed{\sigma}\left(\mathbf{W}_k \boxed{\sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|}} + \mathbf{B}_k \boxed{\mathbf{h}_v^{k-1}}\right), \quad \forall k \in \{1, \ldots, K\}$$

$$\boxed{\mathbf{z}_v} = \mathbf{h}_v^K$$

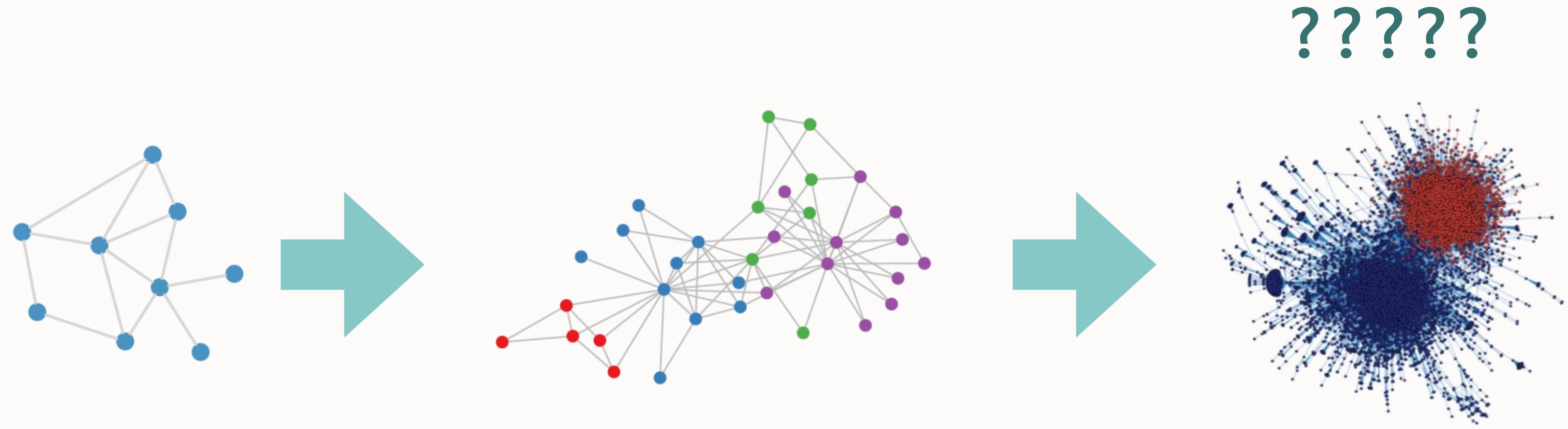
Average of neighbor's previous layer embeddings

Embedding after K layers of neighborhood aggregation

Non-linearity (e.g., ReLU)

Graph Convolutional Network

Limitation of GCN



- For training of the embeddings, model requires that all nodes in the graph are present during training time

- As the number of nodes increases, the matrix becomes larger and computational cost increases accordingly.

GraphSAGE

Definition

- GraphSAGE (**SA**mple and aggre**G**at**E**)

  - Instead of training individual embeddings for each node, generates embeddings by **sampling** features from neighborhoods

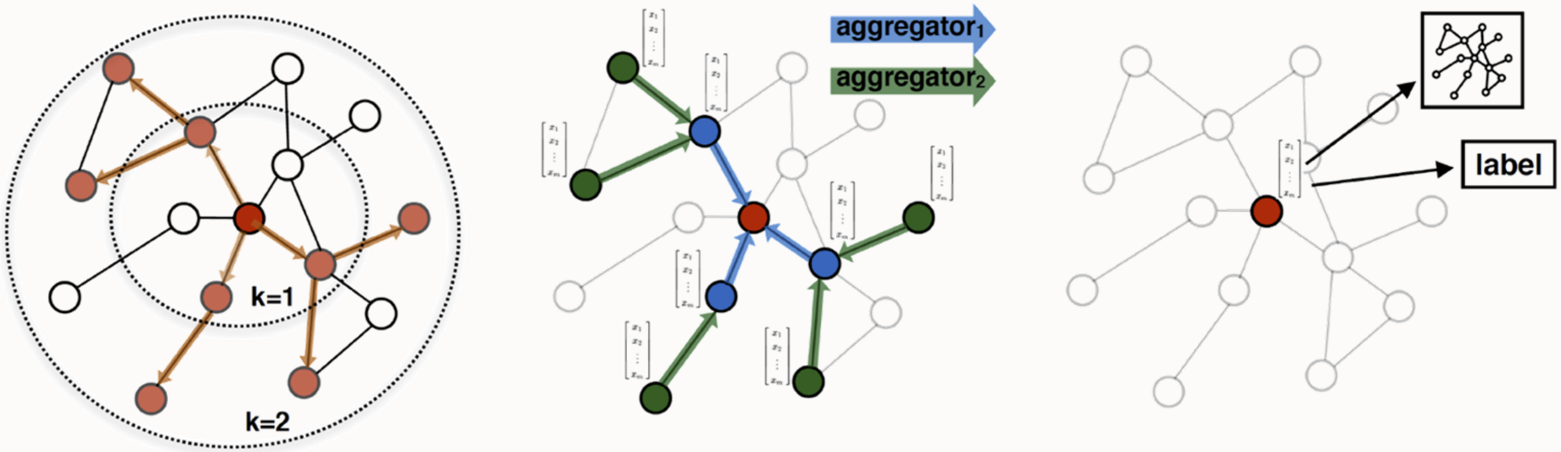
  ➔ **mini batch**

  - Train a set of aggregator functions that learn to **aggregate** feature information a node's local neighborhood

  ➔ **aggregating**

GraphSAGE

Mini batch



1. Sample neighborhood

2. Aggregate feature information from neighbors

3. Predict graph context and label using aggregated information

GraphSAGE

Aggregating

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right)$$

Concatenate neighbor embedding and self embedding

$$\mathbf{h}_v^k = \sigma\left(\left[\mathbf{W}_k \cdot \boxed{\mathrm{AGG}\left(\{\mathbf{h}_u^{k-1}, \forall u \in N(v)\}\right)}, \mathbf{B}_k \mathbf{h}_v^{k-1}\right]\right)$$

Generalized aggregation

GraphSAGE

Aggregating

- Mean

$$\text{AGG} = \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|}$$

- Pool

Element-wise mean/max

$$\text{AGG} = \boxed{\gamma}\left(\{\mathbf{Q}\mathbf{h}_u^{k-1}, \forall u \in N(v)\}\right)$$

- LSTM

$$\text{AGG} = \text{LSTM}\left([\mathbf{h}_u^{k-1}, \forall u \in \pi(N(v))]\right)$$

GraphSAGE

Algorithm

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

---

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \to 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

GraphSAGE

Weighting factor in GraphSAGE

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \boxed{\frac{\mathbf{h}_u^{k-1}}{|N(v)|}} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right)$$

Weighting factor

$$\alpha_{vu} = \frac{1}{|N(v)|}$$

- α_{vu} (importance) is defined explicitly based on the structure properties of graph
- All neighbors $u \in N(v)$ are equally important to node v

Graph Attention Network

Idea

- Specify arbitrary importances to different neighbors of each node in the graph

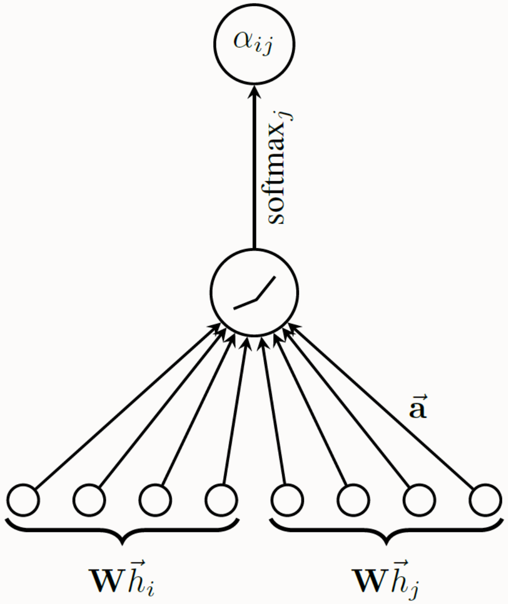
- Compute embedding h_v^k of each node in the graph following an **attention strategy**:

➔ Nodes attend over their neighborhoods' message

➔ Implicitly specifying different weights to different nodes

Graph Attention Network

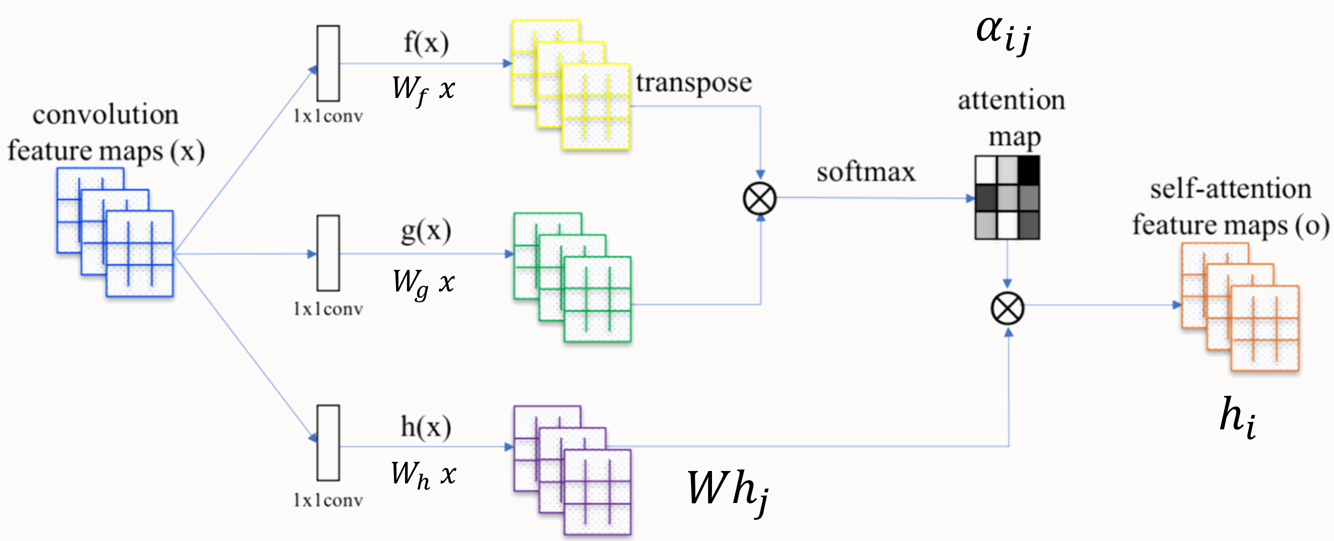
Attention



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

Graph Attention Network

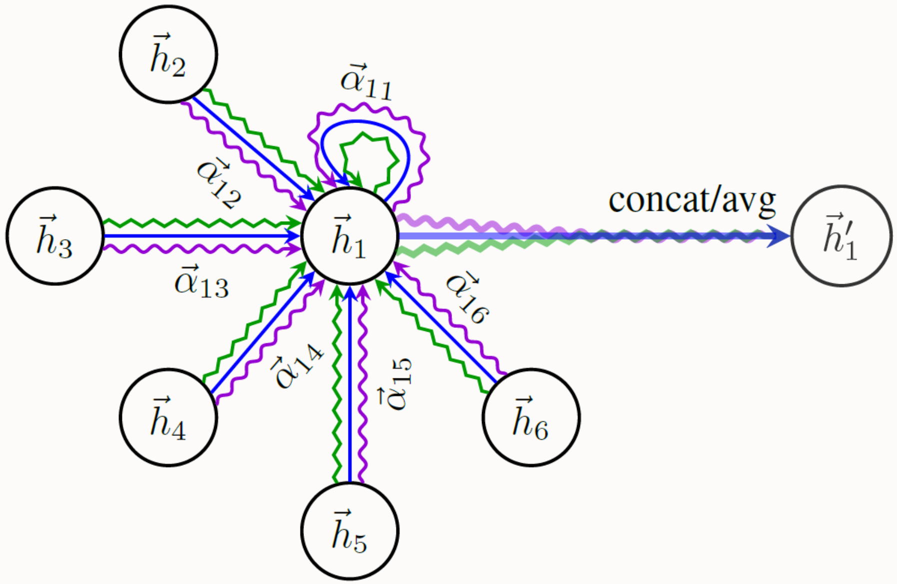
Self attention



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

Graph Attention Network

Multi-head attention.



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

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

Graph Attention Network

Result

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)	$81.7 \pm 0.5\%$	—	$78.8 \pm 0.3\%$
GCN-64*	$81.4 \pm 0.5\%$	$70.9 \pm 0.5\%$	$\mathbf{79.0} \pm 0.3\%$
GAT (ours)	$\mathbf{83.0} \pm 0.7\%$	$\mathbf{72.5} \pm 0.7\%$	$\mathbf{79.0} \pm 0.3\%$

Reference

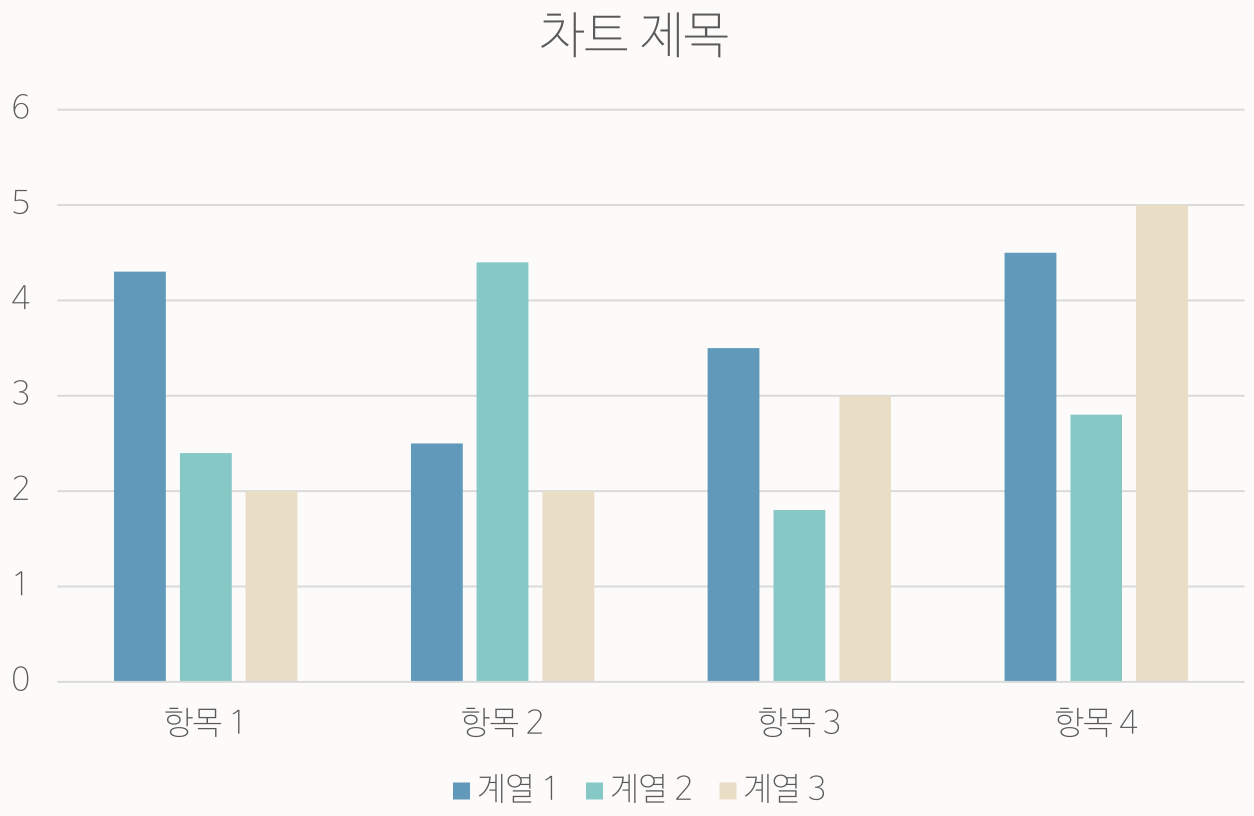
Lorem Ipsum is simply dummy text of the printing and typesetting industry.

- Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.
- Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems (pp. 1024-1034).
- Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.
- http://web.stanford.edu/class/cs224w/
- http://dmqm.korea.ac.kr/activity/seminar/267
- https://www.cs.ubc.ca/~lsigal/532S_2018W2/Lecture18a.pdf

Thank You

제목을 입력하세요

Lorem Ipsum is simply dummy text of the printing and typesetting industry.

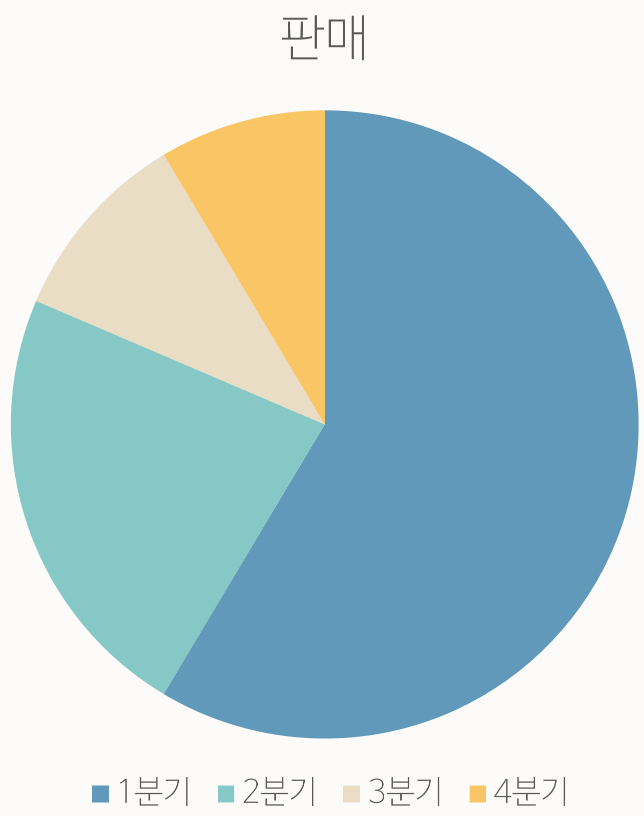


Graph Attention Network

Lorem Ipsum is simply dummy text of the printing and typesetting industry.

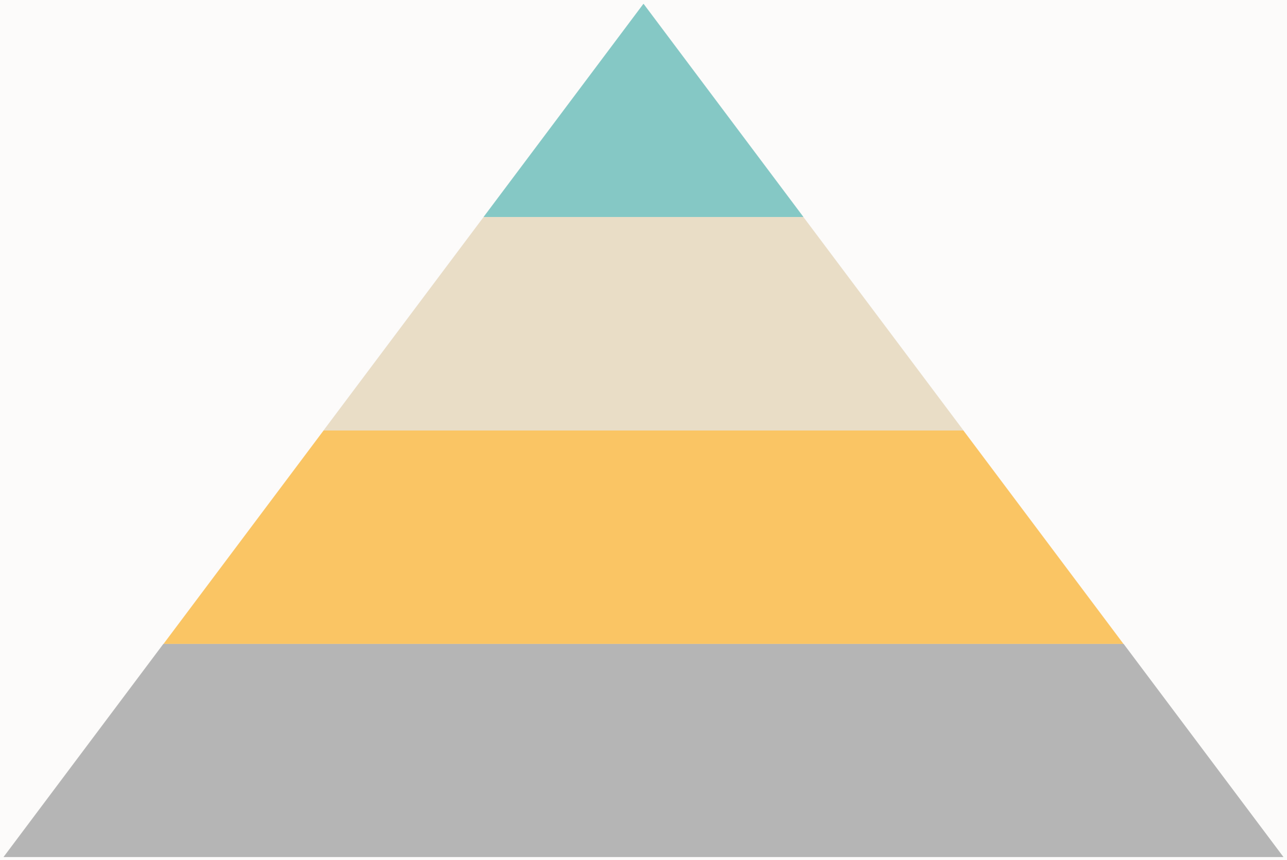
제목을 입력하세요

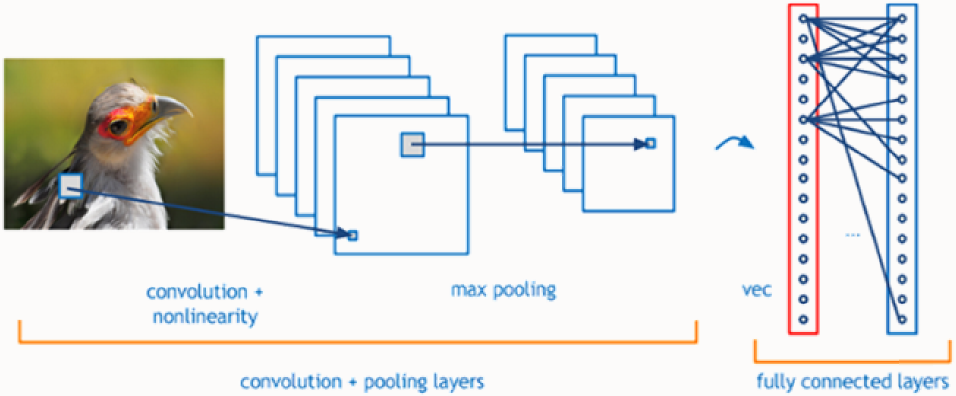
Lorem Ipsum is simply dummy text of the printing and typesetting industry.



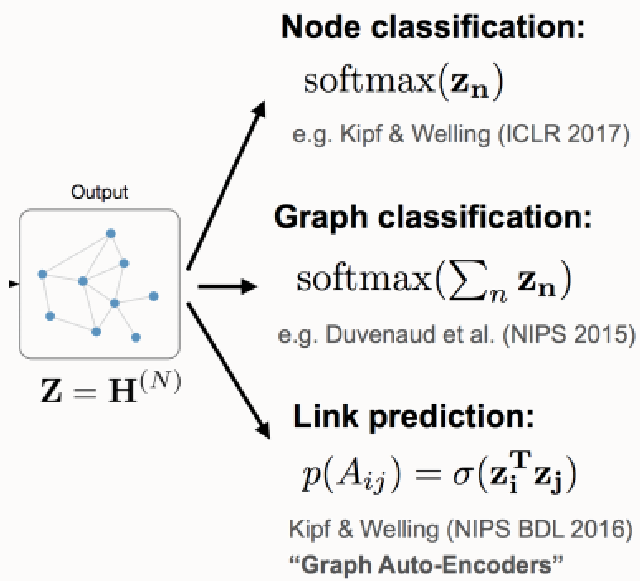
제목을 입력하세요

Lorem Ipsum is simply dummy text of the printing and typesetting industry.



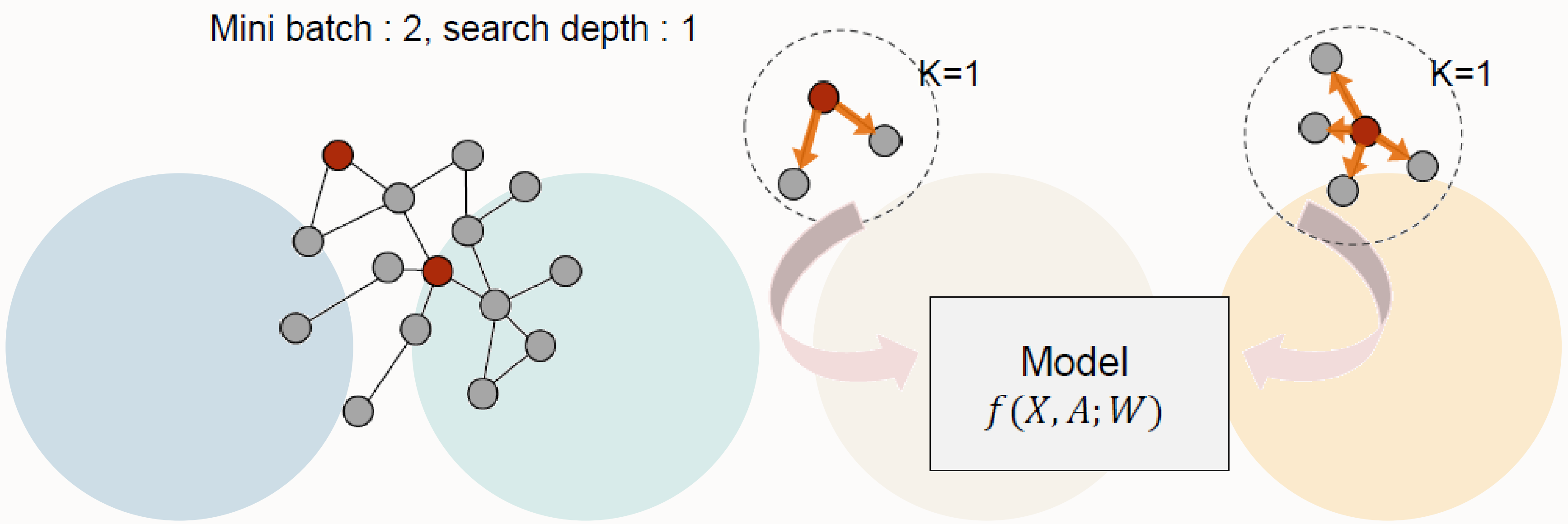


Sparse connection
Weight sharing
Receptive field



Graph Convolutional Network

Lorem Ipsum is simply dummy text of the printing and typesetting industry.



GraphSAGE

Mini batch

Mini batch : 2, search depth : 2