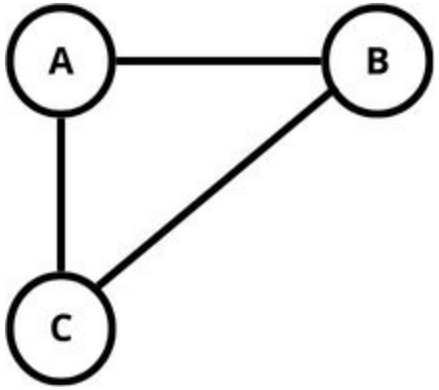
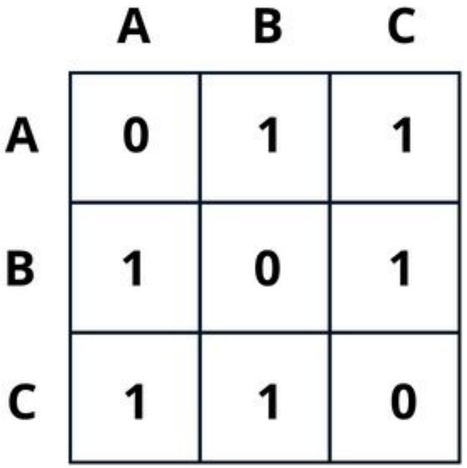
Hands on PyTorch-geometric; an introduction to Graph Neural Networks (GNNs)

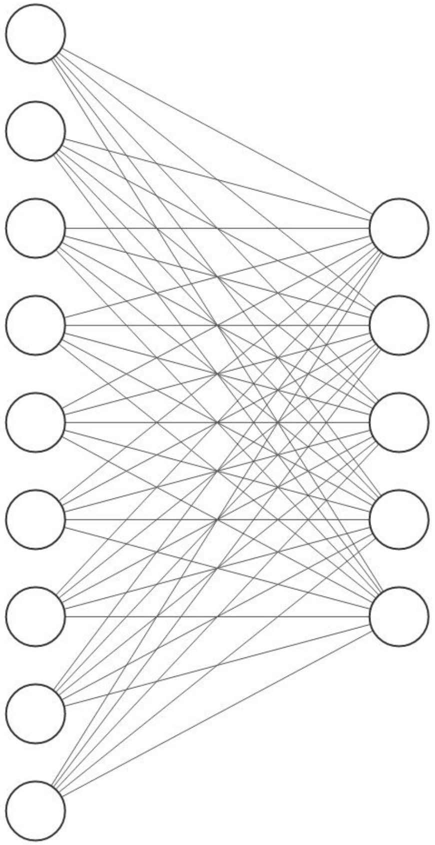
Mohamed Kherraz

Why Graph Neural Networks (GNN) ?

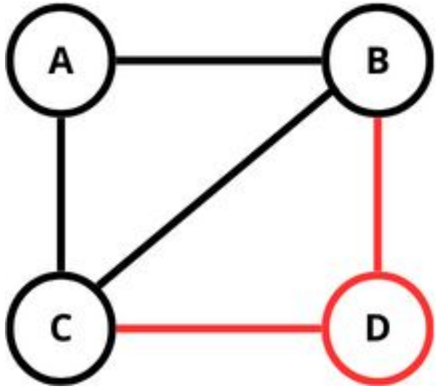


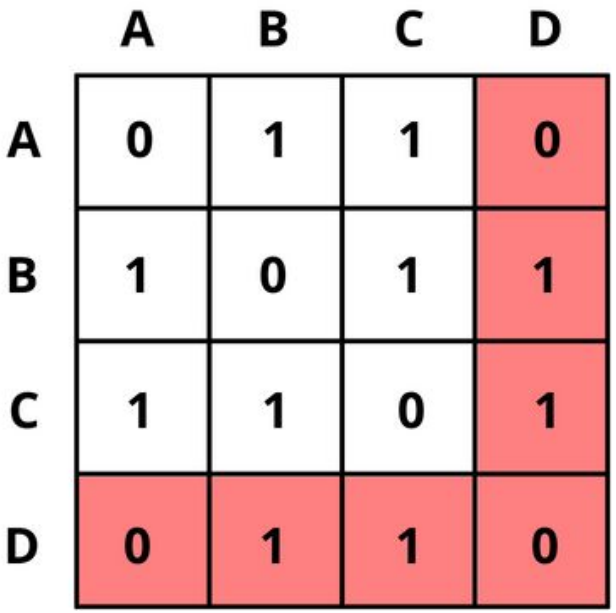


	A	B	C
A	0	1	1
B	1	0	1
C	1	1	0

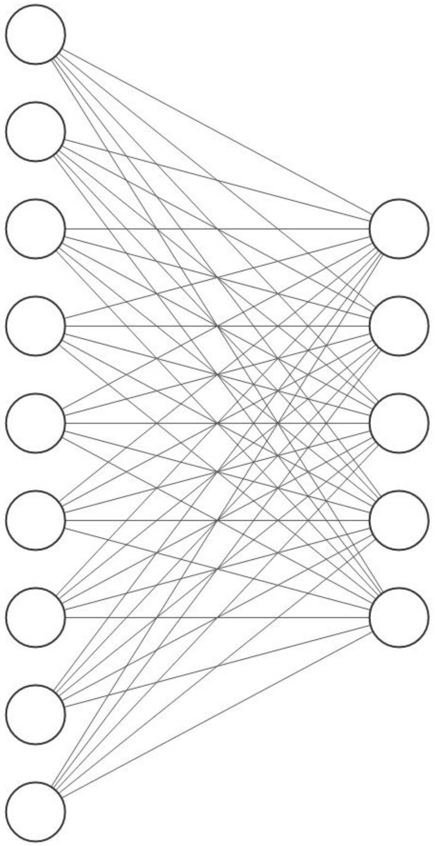


Why Graph Neural Networks (GNN) ?

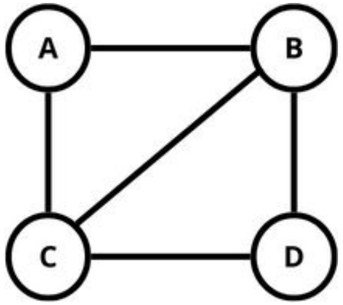


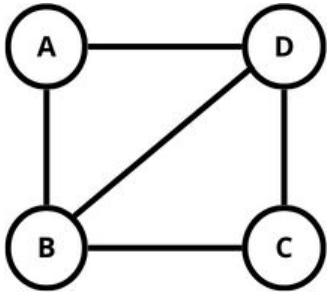


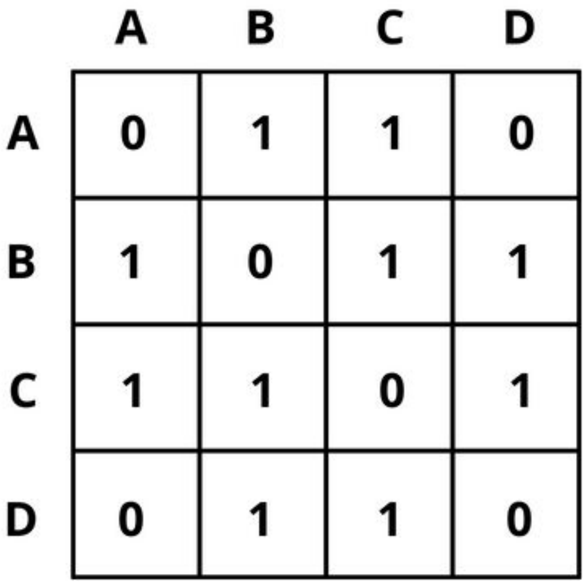
	A	B	C	D
A	0	1	1	0
B	1	0	1	1
C	1	1	0	1
D	0	1	1	0



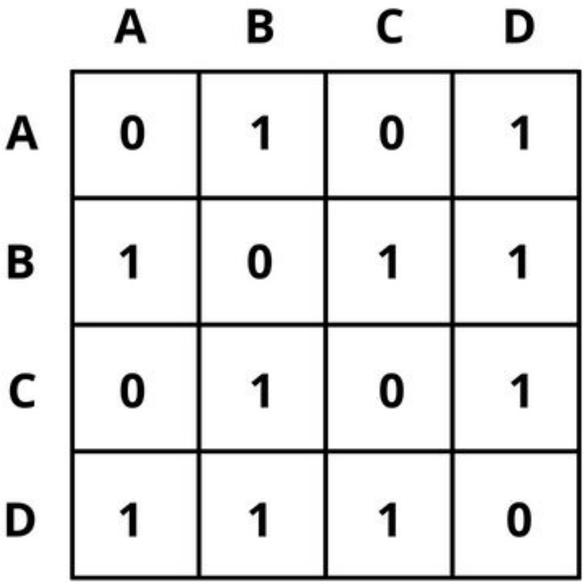
Why Graph Neural Networks (GNN) ?





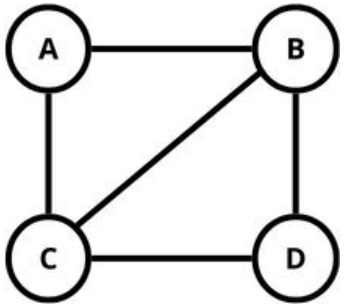


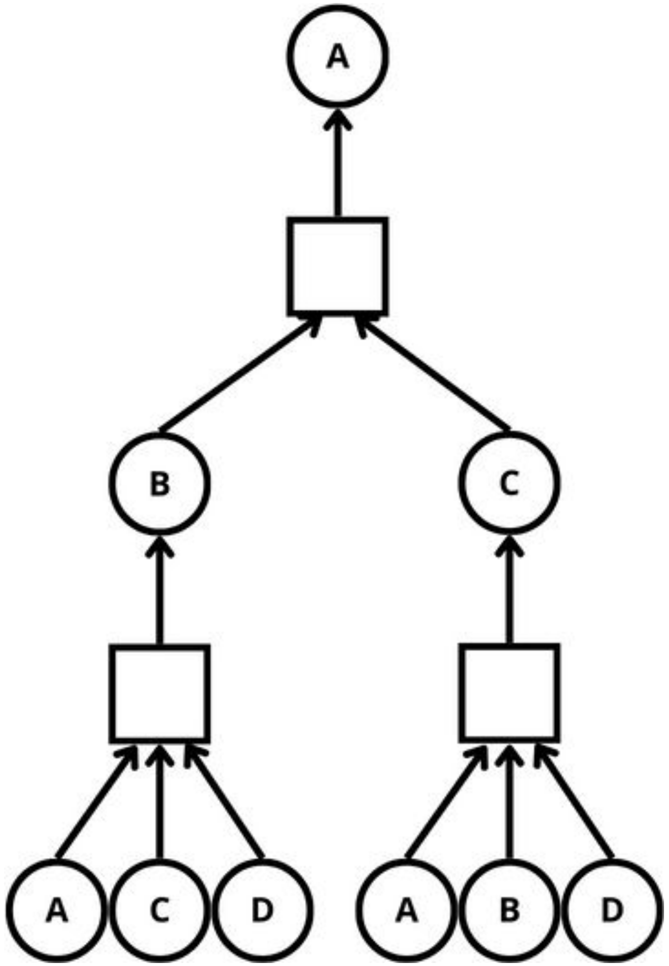
	A	B	C	D
A	0	1	1	0
B	1	0	1	1
C	1	1	0	1
D	0	1	1	0



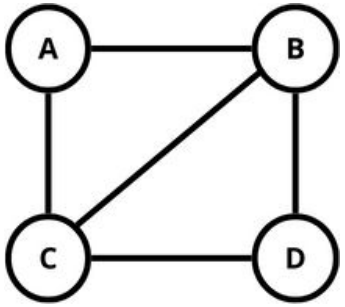
	A	B	C	D
A	0	1	0	1
B	1	0	1	1
C	0	1	0	1
D	1	1	1	0

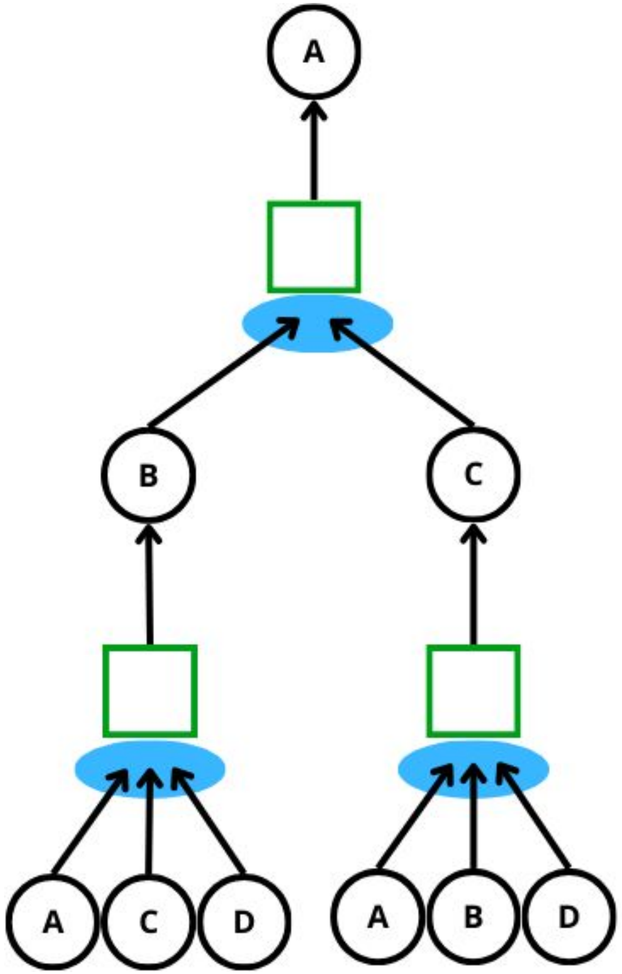
Graph Neural Networks (GNN)





Graph Neural Networks (GNN)

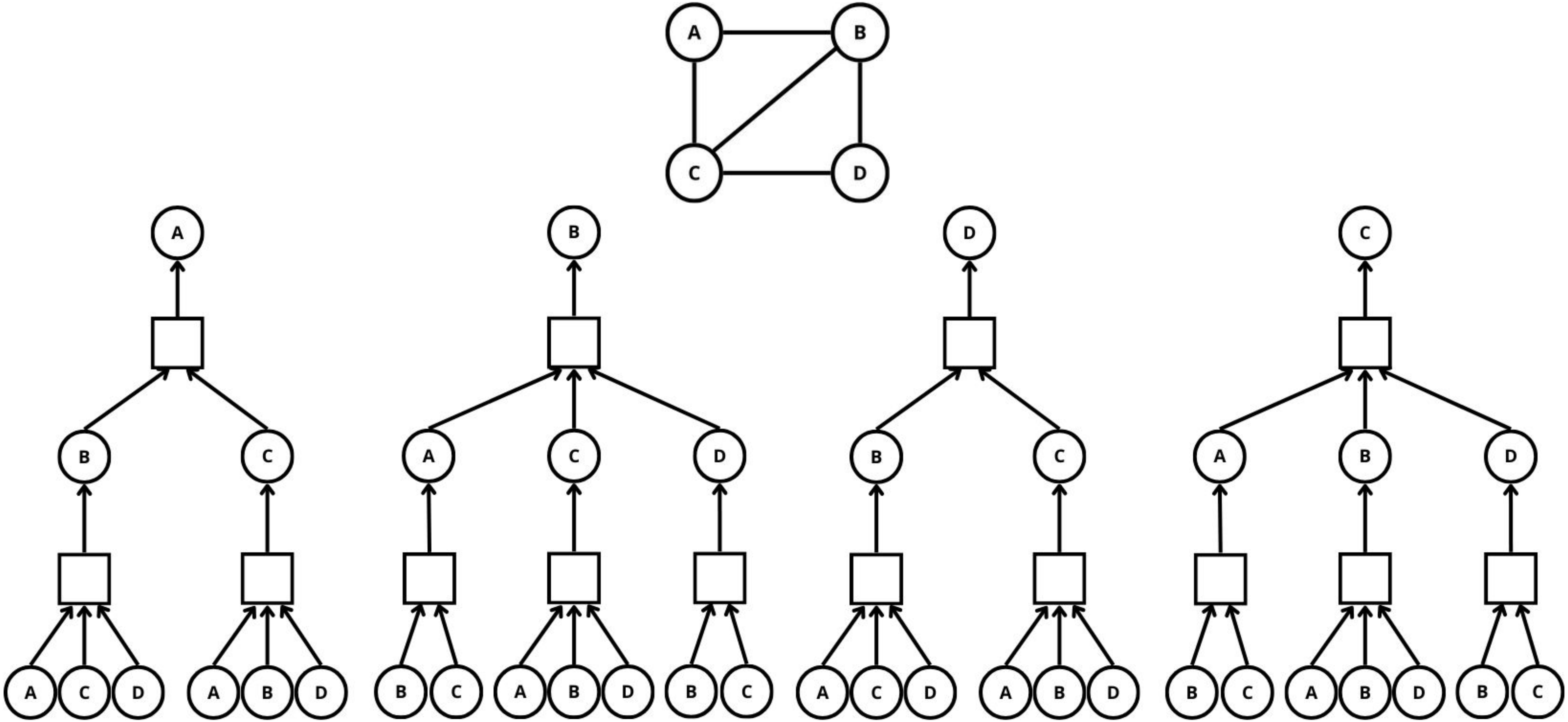




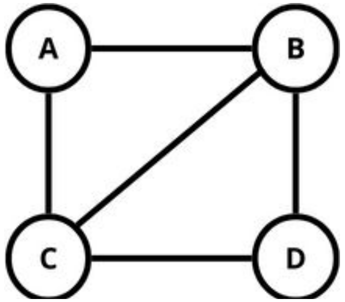
🟦 : *Ordering invariant* agrregation
(Ex : Mean, Sum, ...)

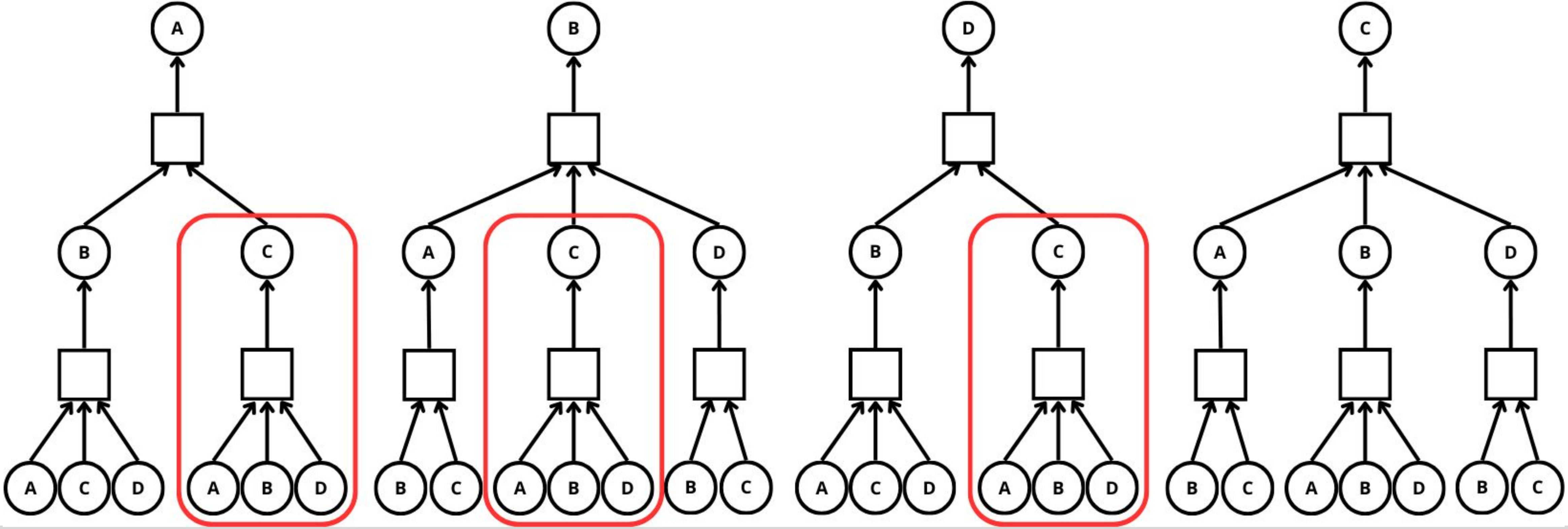
🟩 : Neural Network

Graph Neural Networks (GNN)

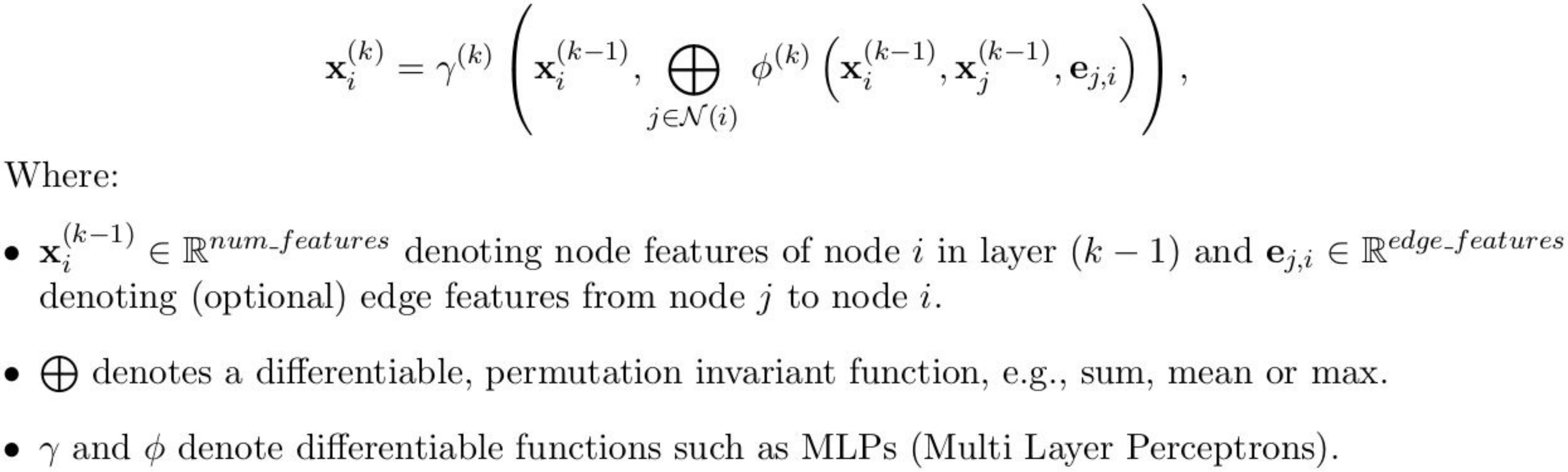


Graph Neural Networks (GNN)





Graph Neural Networks (GNN) - Message passing.



$$\mathbf{x}_i^{(k)} = \gamma^{(k)} \left(\mathbf{x}_i^{(k-1)}, \bigoplus_{j \in \mathcal{N}(i)} \phi^{(k)} \left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i} \right) \right),$$

Where:

- $\mathbf{x}_i^{(k-1)} \in \mathbb{R}^{num\_features}$ denoting node features of node i in layer $(k-1)$ and $\mathbf{e}_{j,i} \in \mathbb{R}^{edge\_features}$ denoting (optional) edge features from node j to node i.
- $\bigoplus$ denotes a differentiable, permutation invariant function, e.g., sum, mean or max.
- γ and ϕ denote differentiable functions such as MLPs (Multi Layer Perceptrons).

Graph Neural Networks (GNN) - GCNConv

$$\mathbf{x}_i^{(k)} = \gamma^{(k)} \left(\mathbf{x}_i^{(k-1)}, \bigoplus_{j \in \mathcal{N}(i)} \phi^{(k)} \left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i} \right) \right)$$

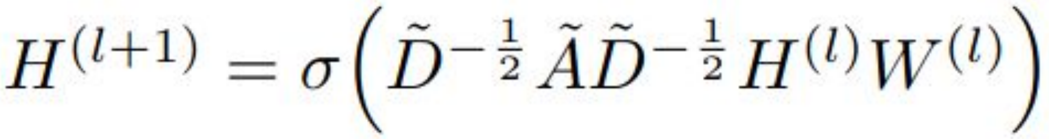
$$\tilde{A} = A + I_N$$

$$\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$$

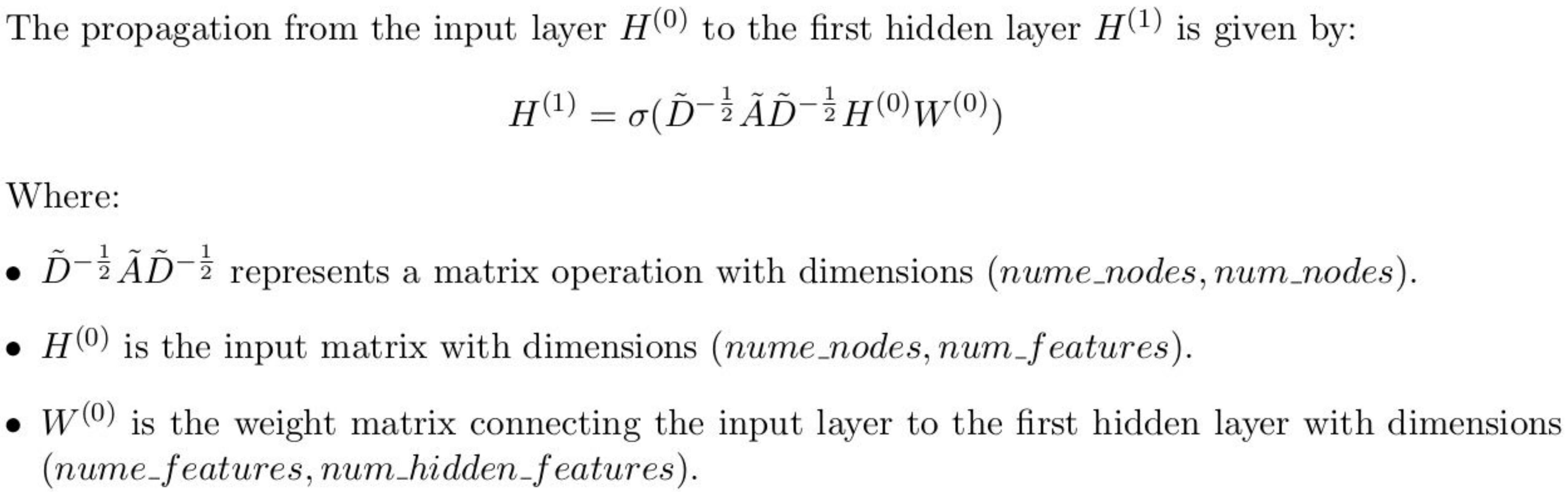
$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)} \right)$$

$$ReLU(.) = max(0, .)$$

Graph Neural Networks (GNN) - GCNConv



$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$



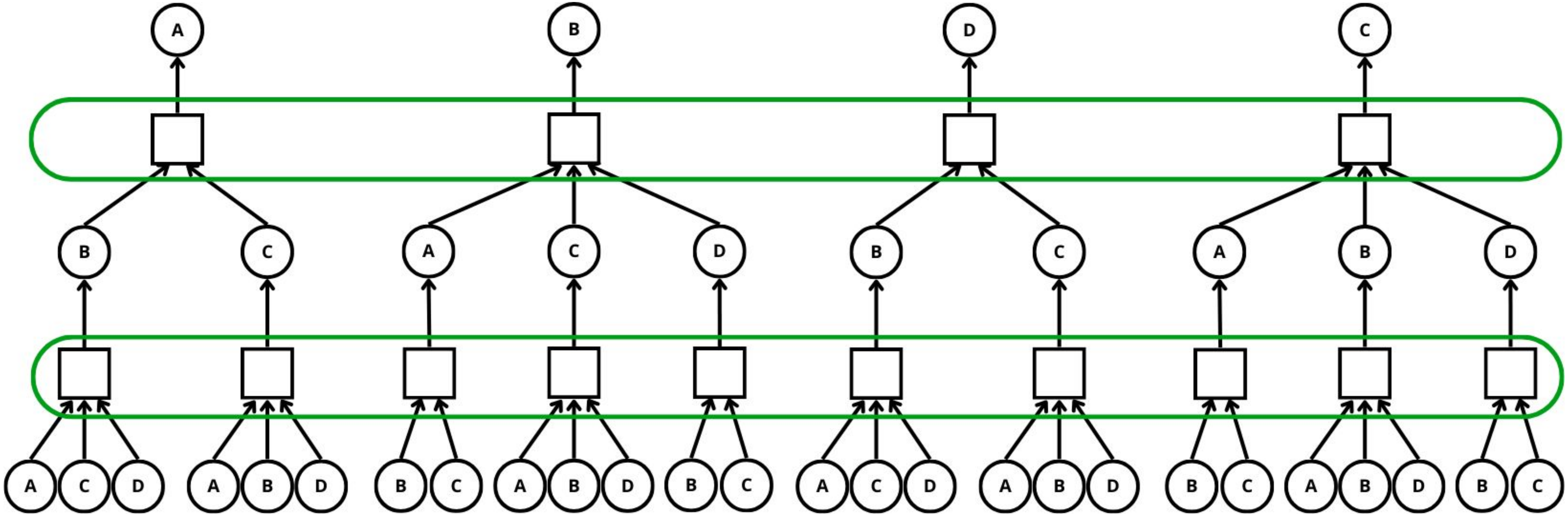
The propagation from the input layer $H^{(0)}$ to the first hidden layer $H^{(1)}$ is given by:

$$H^{(1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(0)}W^{(0)})$$

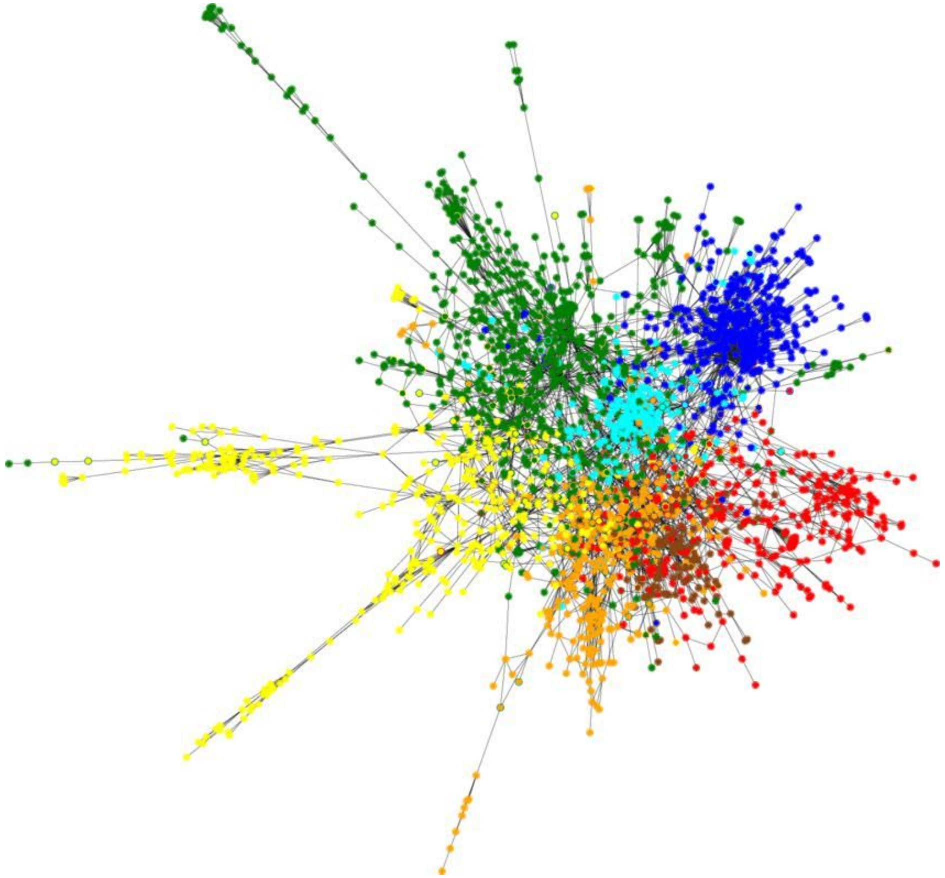
Where:

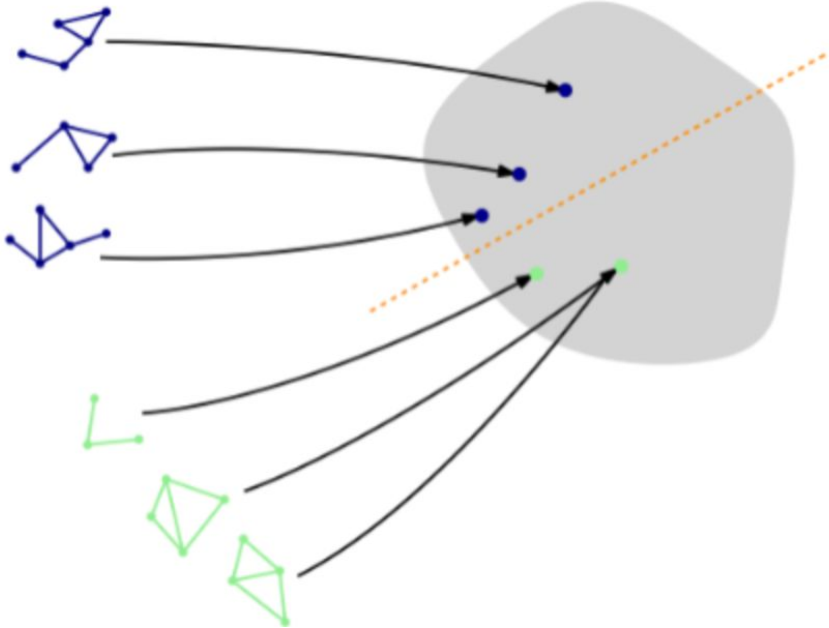
- $\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ represents a matrix operation with dimensions $(nume\_nodes, num\_nodes)$.
- $H^{(0)}$ is the input matrix with dimensions $(nume\_nodes, num\_features)$.
- $W^{(0)}$ is the weight matrix connecting the input layer to the first hidden layer with dimensions $(nume\_features, num\_hidden\_features)$.

Graph Neural Networks (GNN) – shared weights

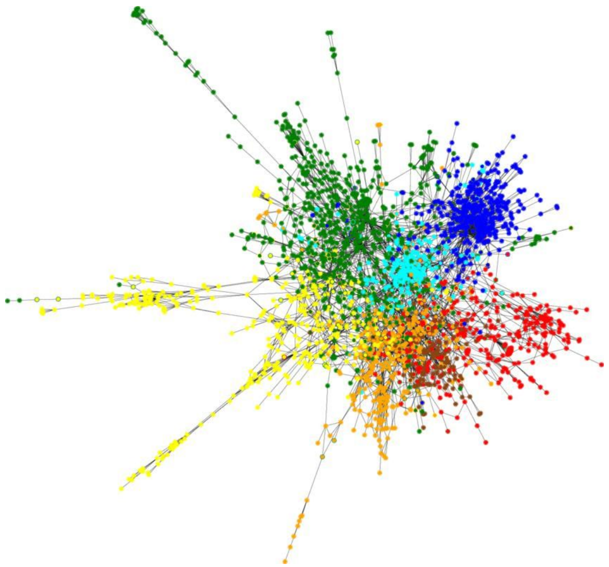


Application of GNNs.

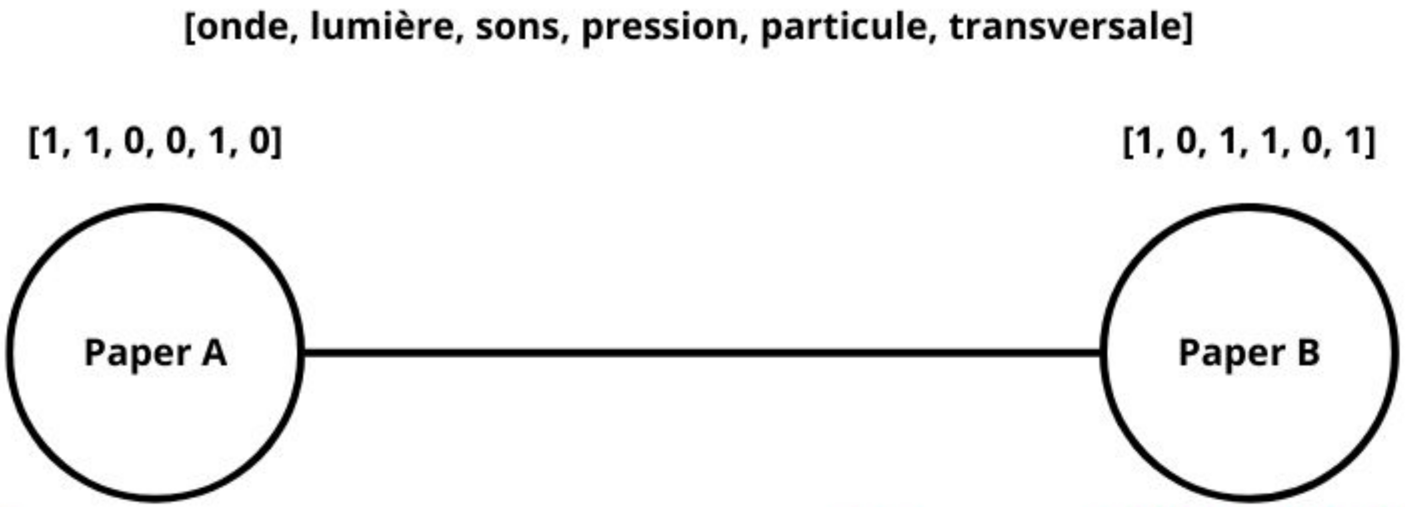




Node classification with GNNs. - Cora Dataset



Train on 5% only



[onde, lumière, sons, pression, particule, transversale]

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

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

Paper A

Paper B

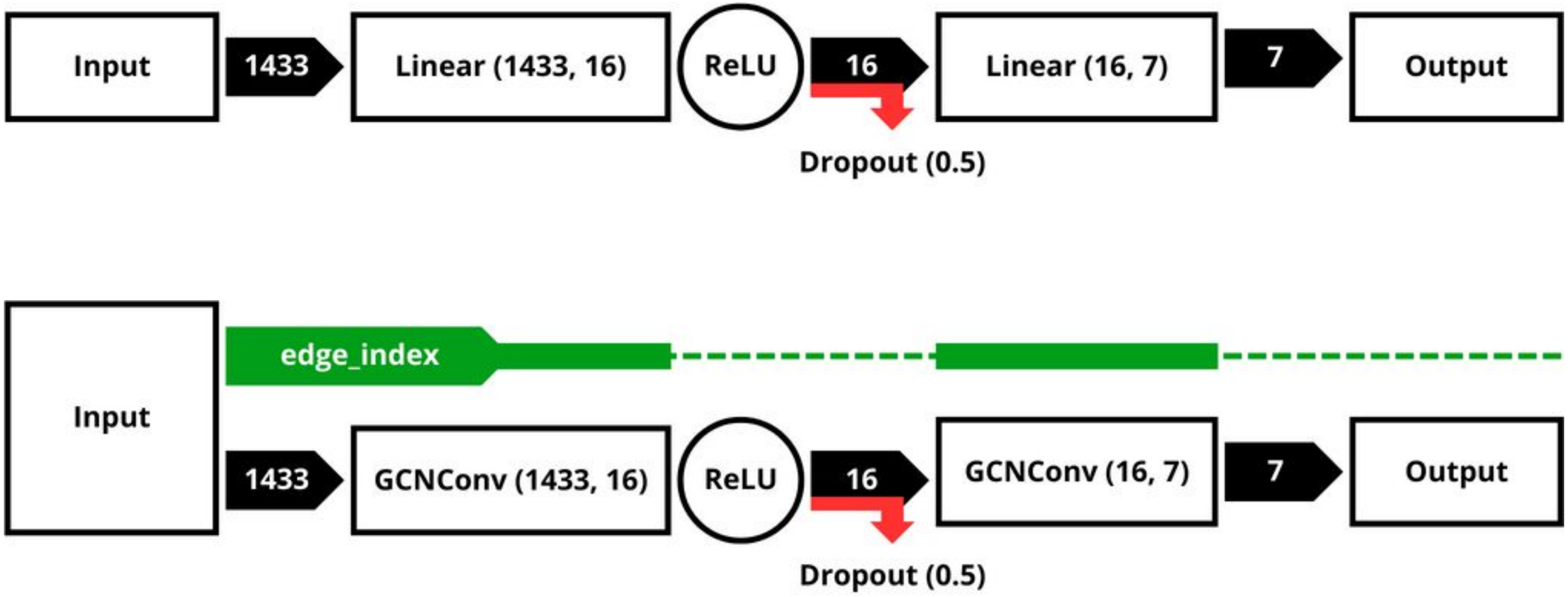
La lumière prends le comportement d'onde et particule en même temps.

Le sons est une onde transversale qui se propage dans l'air sous forme de différence de pression.

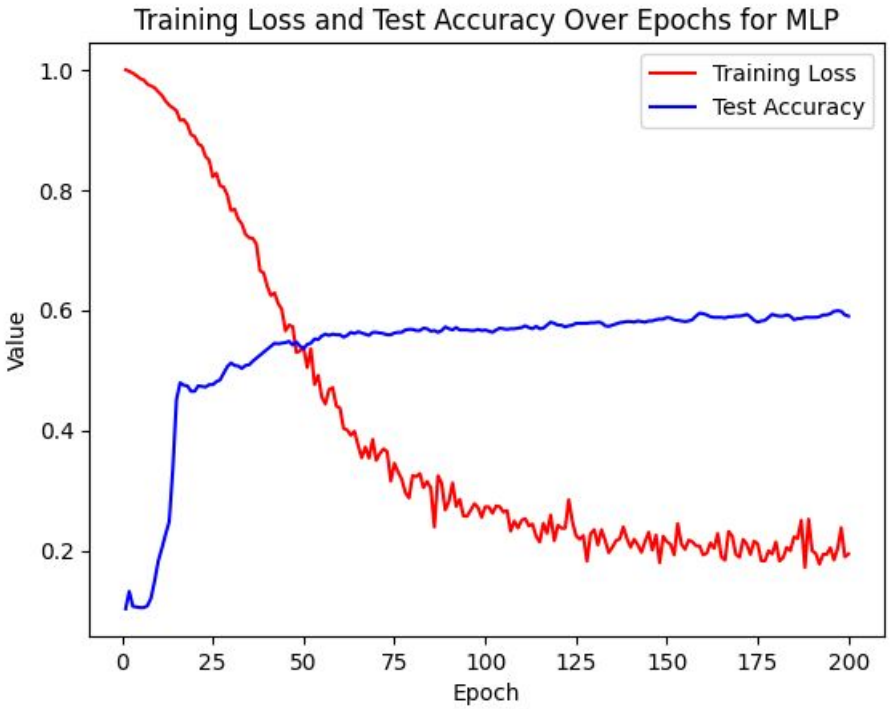
Time for practice !

https://colab.research.google.com/drive/1_fB3-rOURz0LldskC2TTd87U5Uu0aYzf?usp=sharing

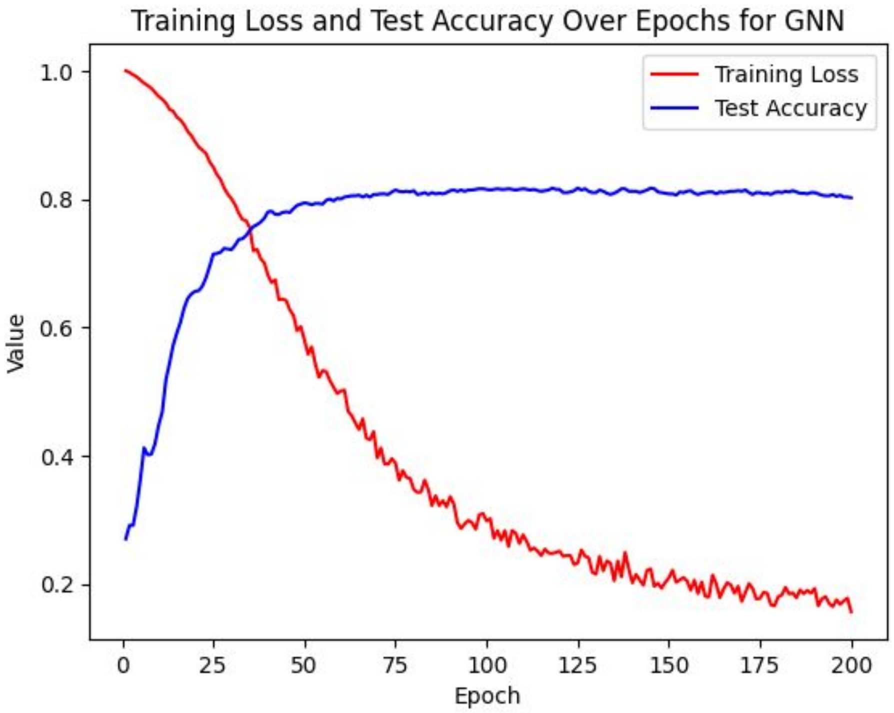
Node classification with GNNs.



Node classification with GNNs.

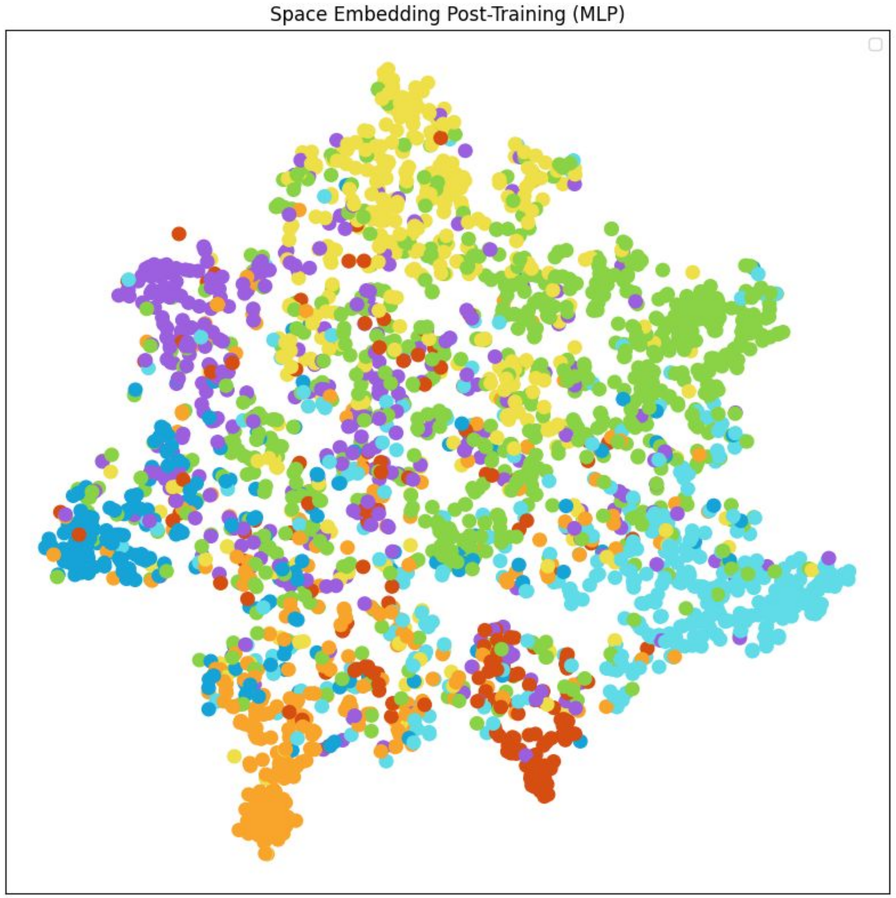


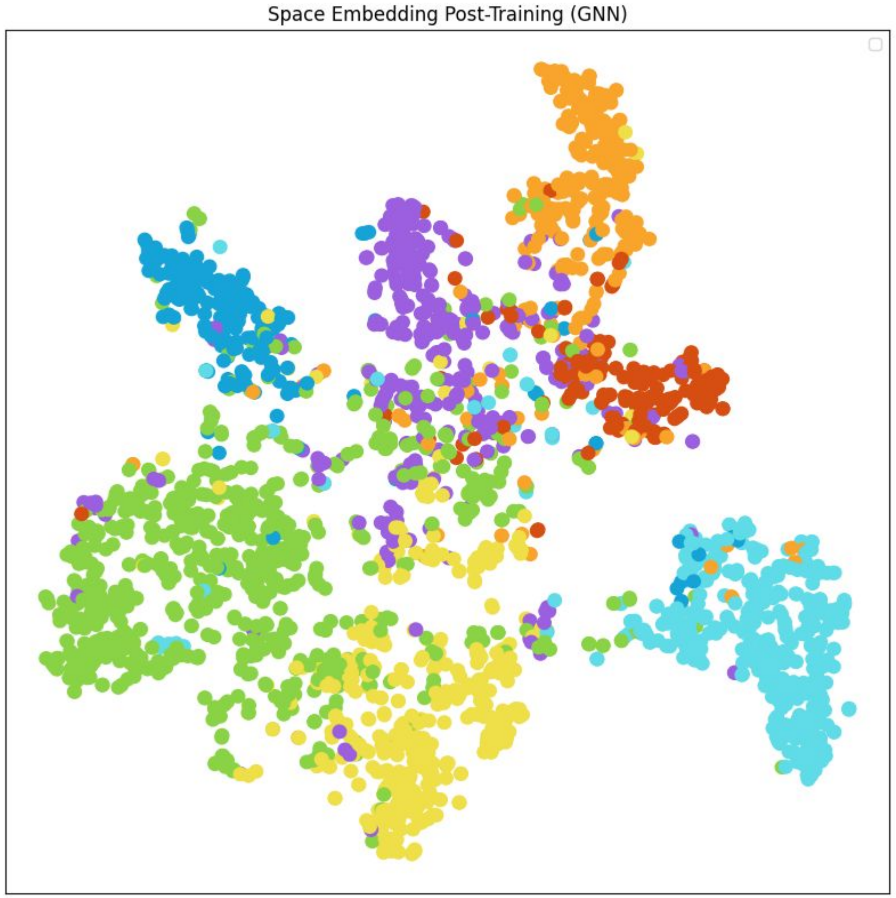
Training Loss and Test Accuracy Over Epochs for MLP



Training Loss and Test Accuracy Over Epochs for GNN

Node classification with GNNs.





Thank you!