What is Geometric Deep Learning?

Antonio Longa[1,2]

MobS[1] Lab, Fondazione Bruno Kessler,Trento, Italy
SML[2] Lab, University of Trento, Italy

Pytorch Geometric tutorials

- Antonio Longa and Gabriele Santin
- Open source project

- Learn how to use Geometric Deep Learning
- Pytorch Geometric

Pytorch Geometric tutorials

- Antonio Longa and Gabriele Santin
- Open source project

- Learn how to use Geometric Deep Learning
- Pytorch Geometric

How it works

- Brief introduction to a GDL model
- Practice!
- Feel free to join, ask and present

Pytorch Geometric tutorials

- Antonio Longa and Gabriele Santin
- Open source project

- Learn how to use Geometric Deep Learning
- Pytorch Geometric

How it works

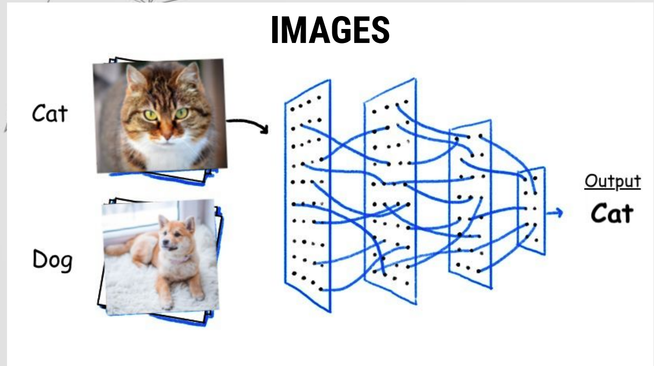
- Brief introduction to a GDL model
- Practice!
- Feel free to join, ask and present

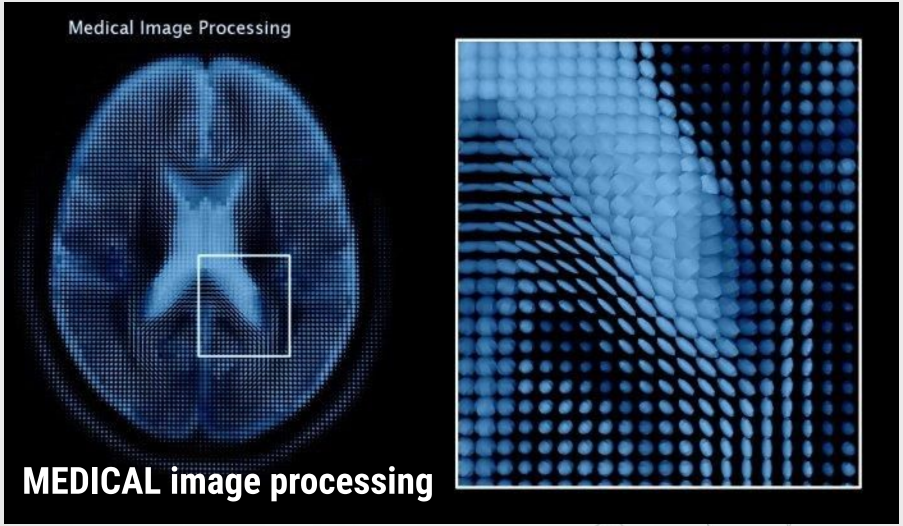
Who are you?

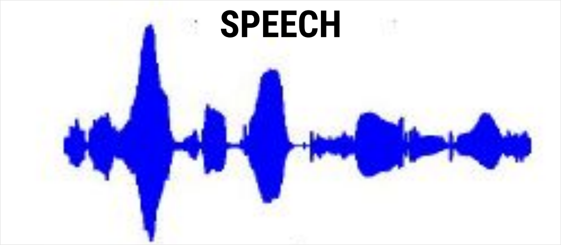
- Researchers
- Students
- Engineers
- ...

TABLE OF CONTENTS

01 Deep Learning

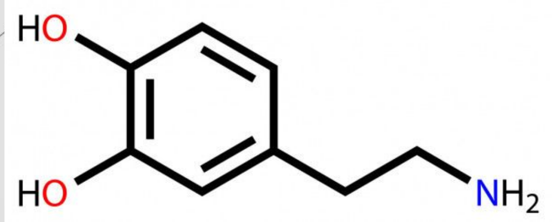




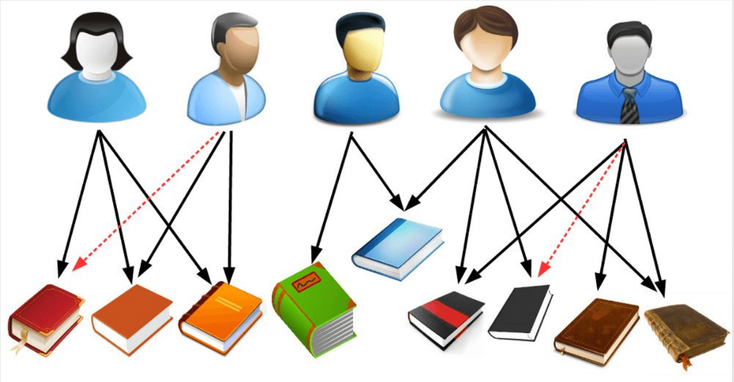


01 Other fields ?

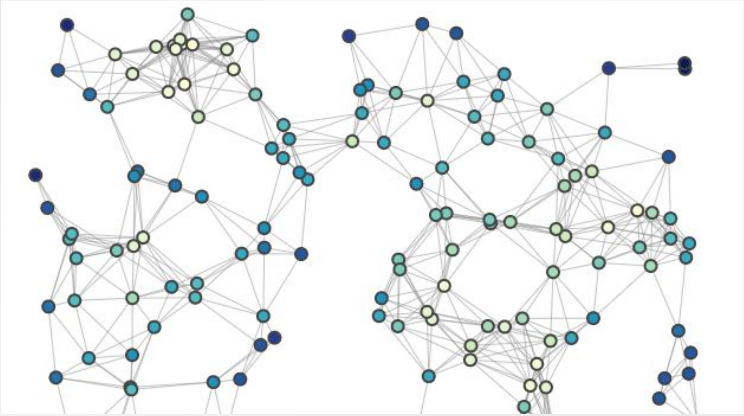
BIOLOGY



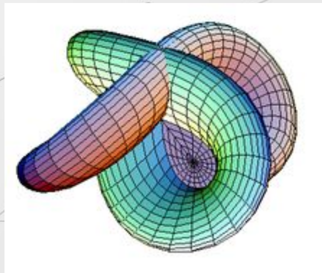
RECOMMENDER SYSTEMS



NETWORK Science



MANIFOLD



01 Other fields ?

DIFFERENCE BETWEEN:

- Images and manifold?
- Speech and molecules?
- RX images and graphs?

01 Other fields ?

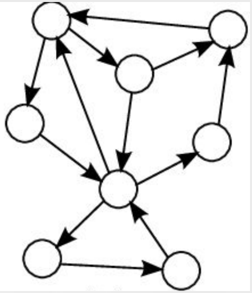
DIFFERENCE BETWEEN:

- Images and manifold?
- Speech and molecules?
- RX images and graphs?

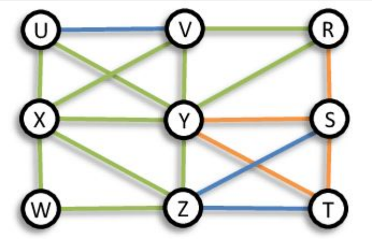
NON-EUCLIDEAN DOMAINS

02 Graphs

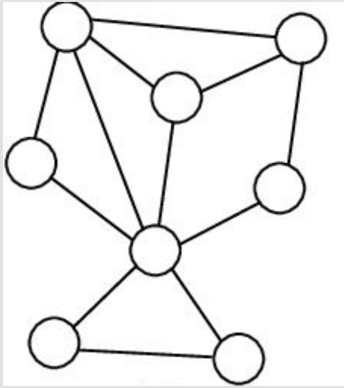
Directed



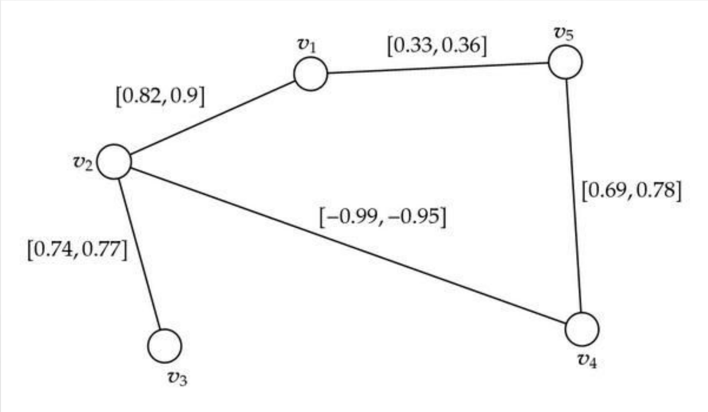
Node labeled graph



Undirected

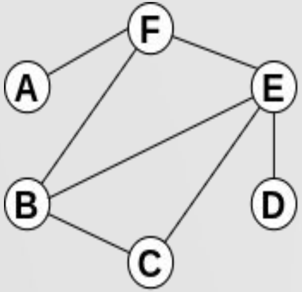


Edge labeled graph



03 Graph representation

GRAPH

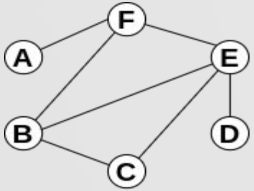


ADJ MATRIX

	A	B	C	D	E	F
A	0	0	0	0	0	1
B		0	1	0	1	1
C			0	0	1	0
D				0	1	0
E					0	1
F						0

03 Deep learning

GRAPH

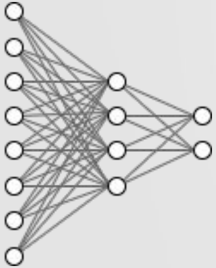


ADJ MATRIX

	A	B	C	D	E	F
A	0	0	0	0	0	1
B		0	1	0	1	1
C			0	0	1	0
D				0	1	0
E					0	1
F						0

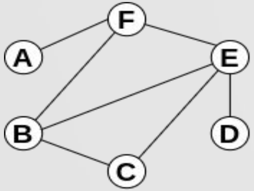
Neural network

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



03 Deep learning

GRAPH

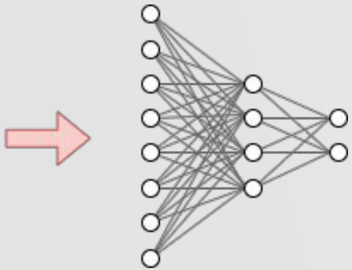


ADJ MATRIX

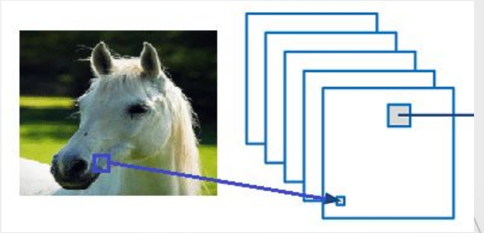
	A	B	C	D	E	F
A	0	0	0	0	0	1
B		0	1	0	1	1
C			0	0	1	0
D				0	1	0
E					0	1
F						0

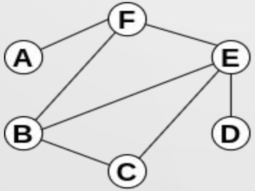
Neural network

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



Convolution Neural network

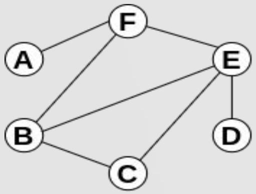




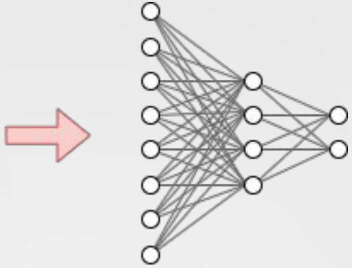
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & \boxed{0 \;\; 1} \\ 1 & 1 & 1 & 0 & \boxed{1 \;\; 0} \end{bmatrix}$$

1	0	1
1	1	0
0	1	$\boxed{0}$

03 Deep learning



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

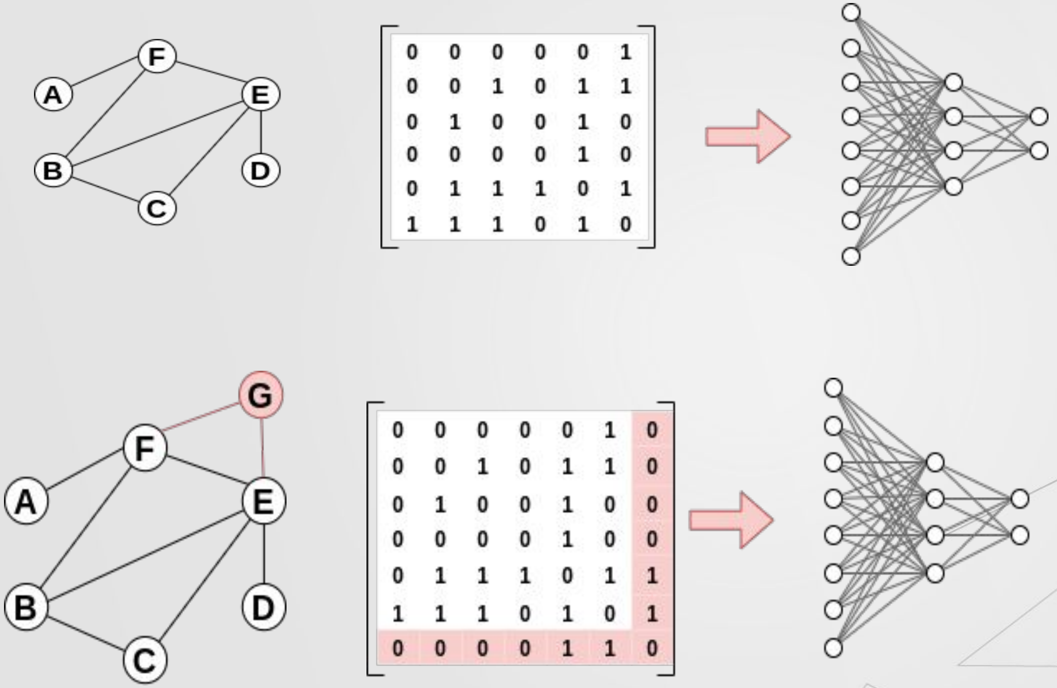


PROBLEMS:

- Different sizes
- NOT invariant to nodes ordering

03 Deep learning: problems

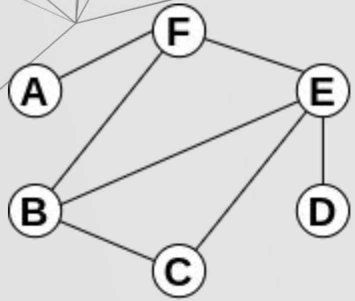
Different sizes



03 Deep learning: problems

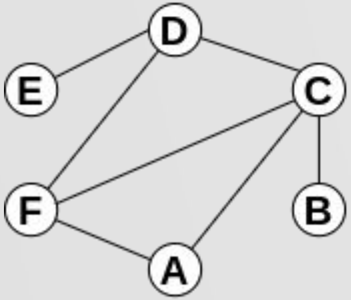
NOT invariant to node ordering

G



$$G = G'$$

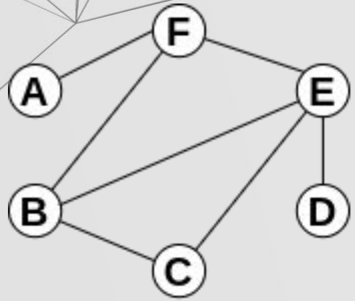
G'



03 Deep learning: problems

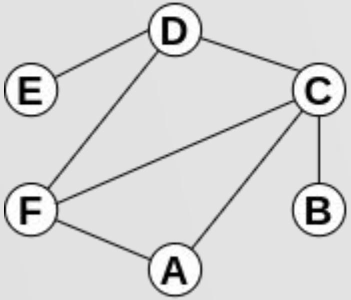
NOT invariant to node ordering

G



$G = G'$

G'



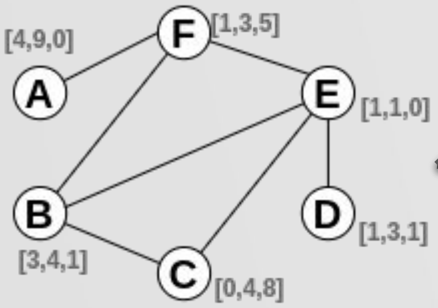
$$\mathrm{Adj}(G) \quad \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

$$\mathrm{Adj}(G) \neq \mathrm{Adj}(G')$$

$$\mathrm{Adj}(G') \quad \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

04 Definitions

$\boldsymbol{G} = (V,E)$



$\mathbf{X} \in \mathcal{R}^{m \times |V|}$

Cols = Features

Rows = Nodes

$$\begin{bmatrix} 4 & 9 & 0 \\ 3 & 4 & 1 \\ 0 & 4 & 8 \\ 1 & 3 & 1 \\ 1 & 1 & 0 \\ 1 & 3 & 5 \end{bmatrix}$$

$\mathbf{A} = \mathrm{Adj}(\boldsymbol{G})$

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

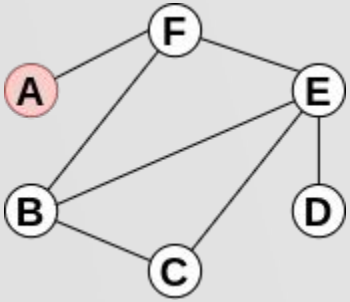
05 Graph neural networks

- Define a **computation graph**
- **Use** the computation graph

COMPUTATION GRAPH

The neighbour of a node define its computation graph

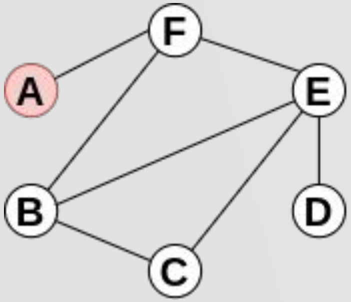
INPUT GRAPH



COMPUTATION GRAPH

The neighbour of a node define its computation graph

INPUT GRAPH

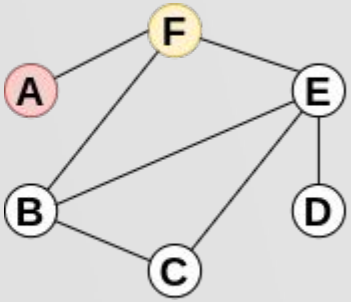


COMPUTATION GRAPH

COMPUTATION GRAPH

The neighbour of a node define its computation graph

INPUT GRAPH

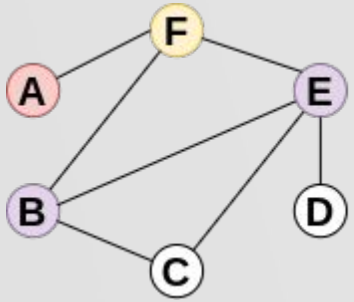


COMPUTATION GRAPH

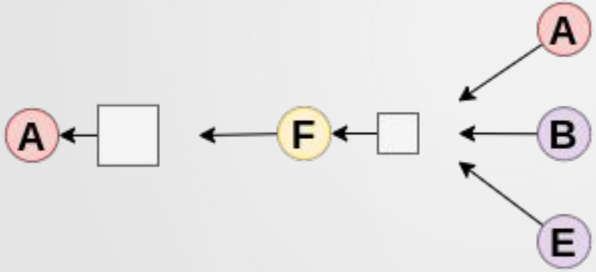
COMPUTATION GRAPH

The neighbour of a node define its computation graph

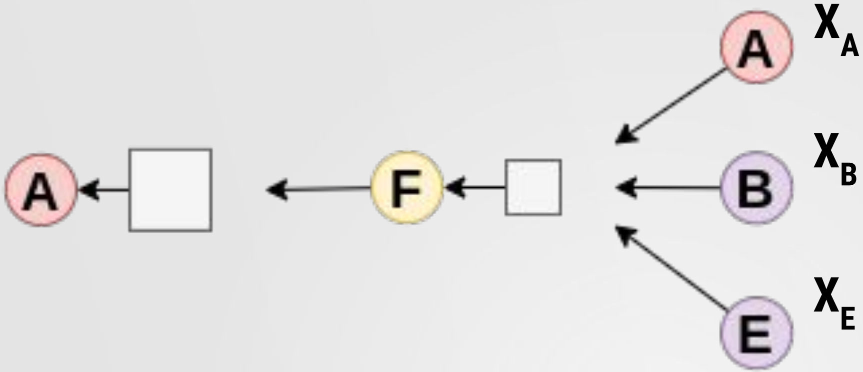
INPUT GRAPH



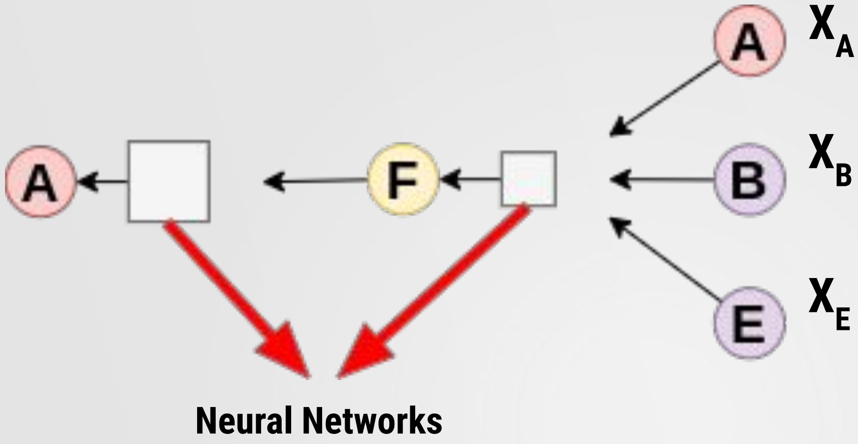
COMPUTATION GRAPH



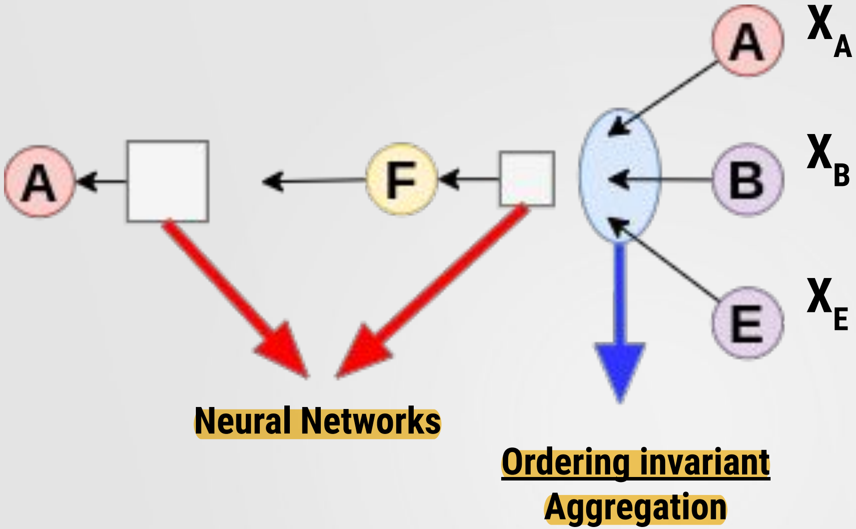
05 Graph neural networks



05 Graph neural networks

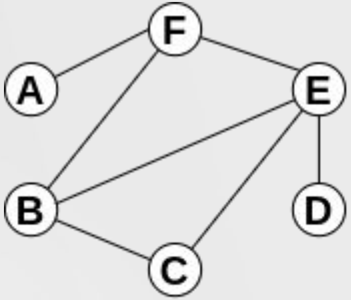


05 Graph neural networks

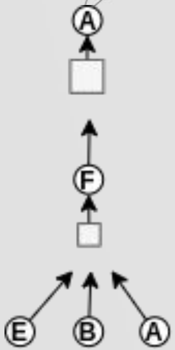


Sum
Average

05 Graph neural networks

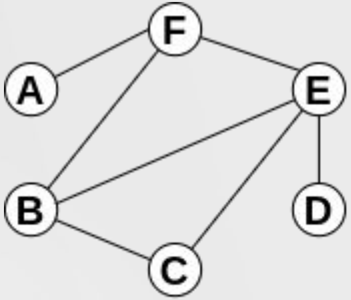


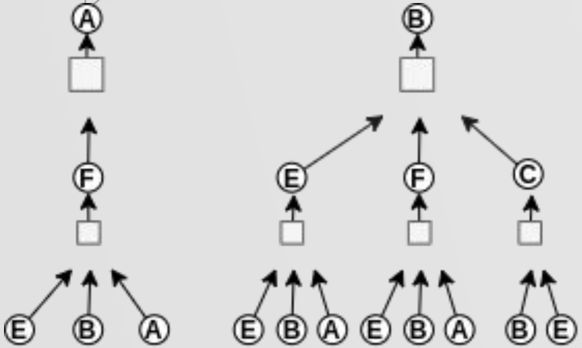
Every node has its own **computation graph**



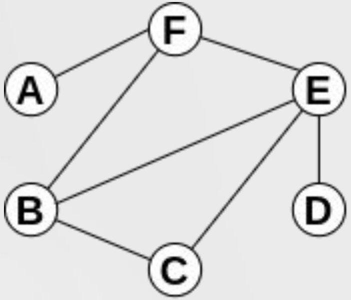
05 Graph neural networks

Every node has its own **computation graph**

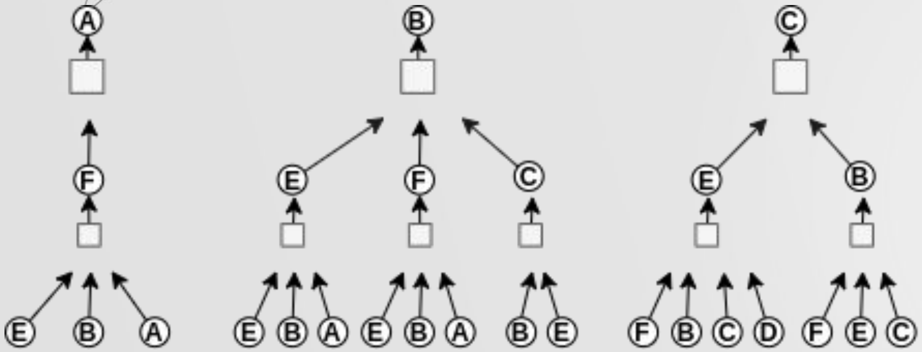




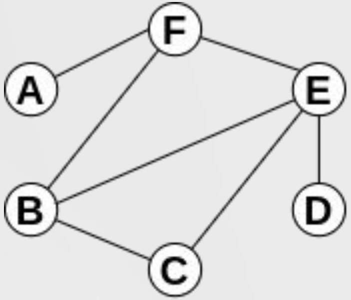
05 Graph neural networks



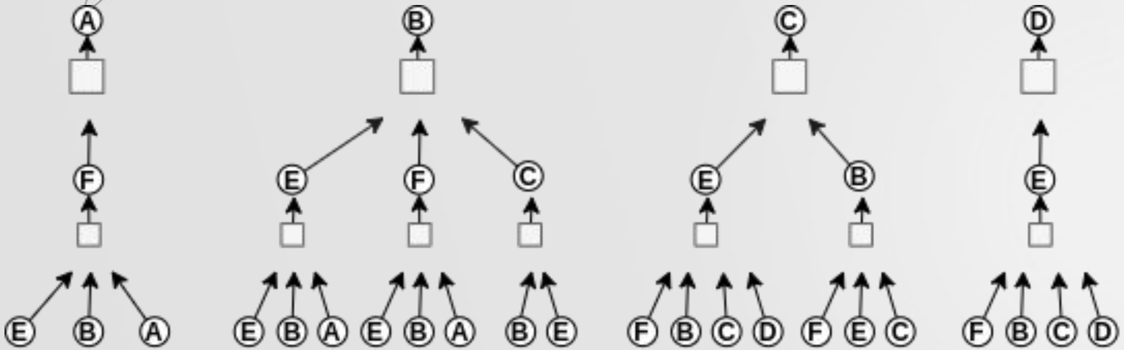
Every node has its own **computation graph**



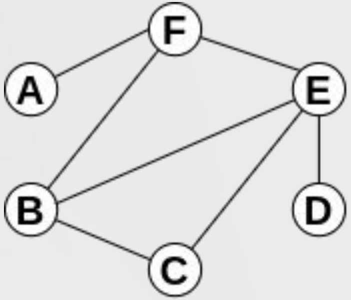
05 Graph neural networks



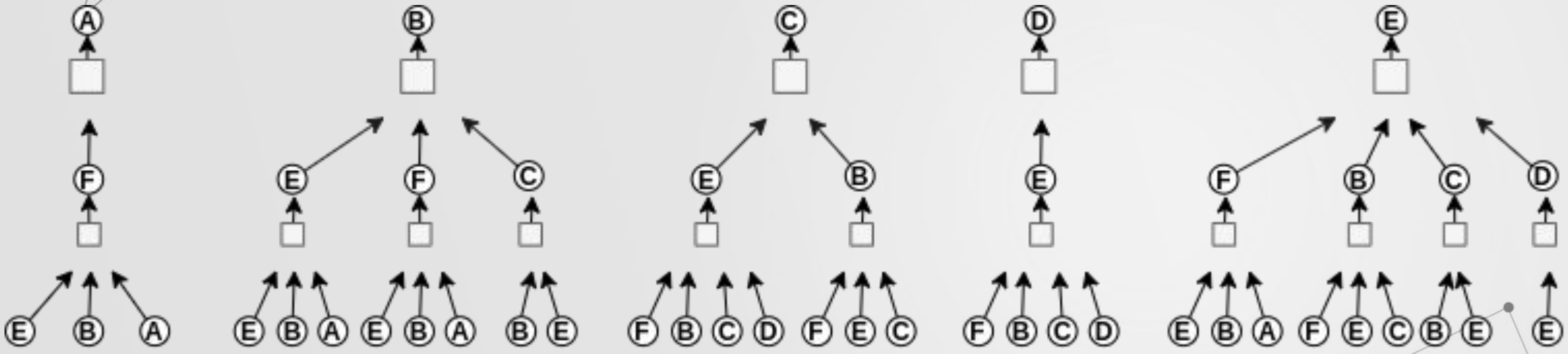
Every node has its own **computation graph**



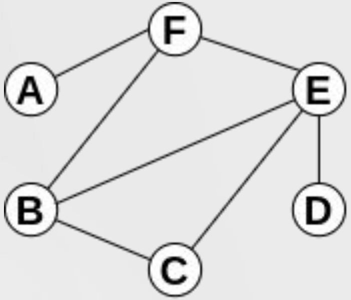
05 Graph neural networks



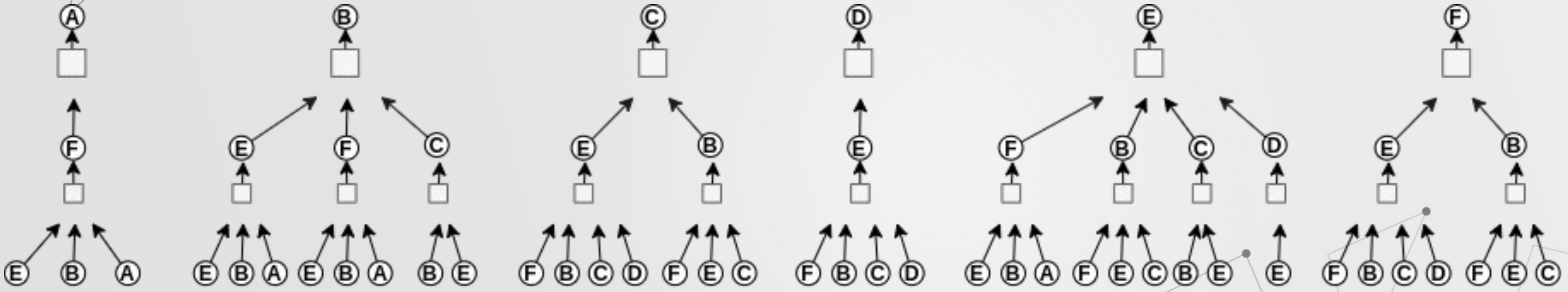
Every node has its own **computation graph**



05 Graph neural networks

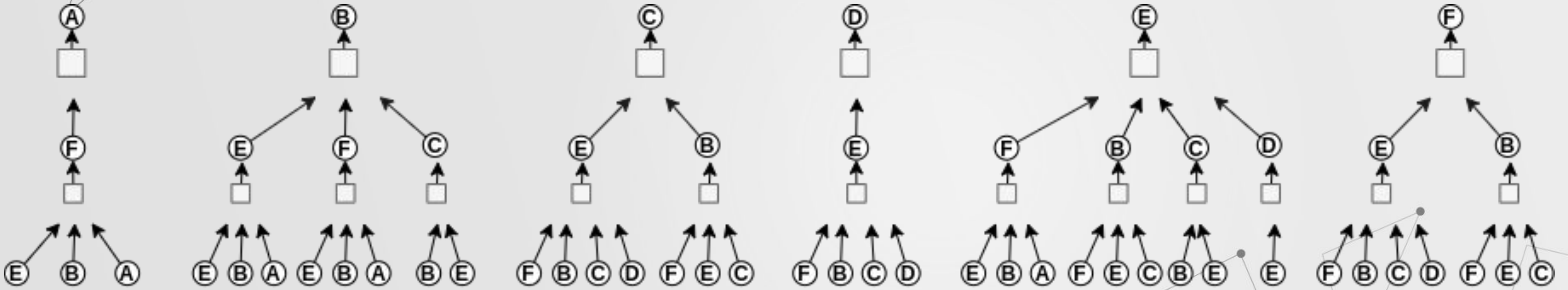


Every node has its own **computation graph**



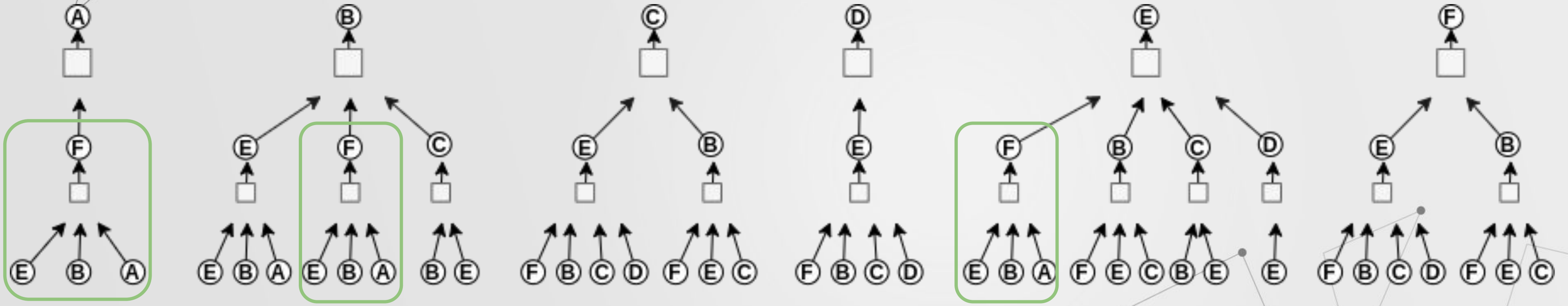
05 Graph neural networks

Can you see **redundancy?**

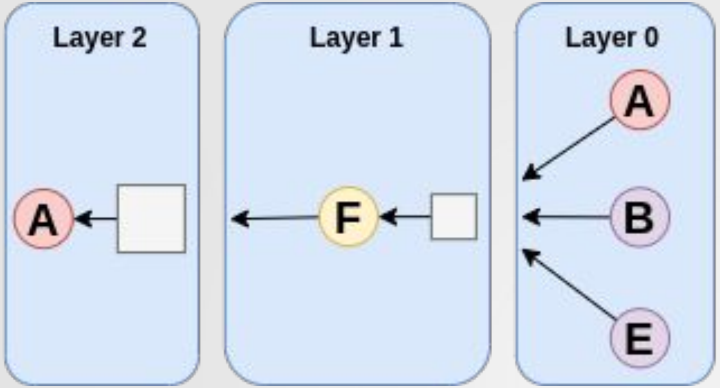


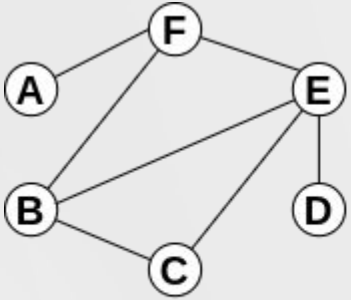
05 Graph neural networks

Can you see **redundancy?**



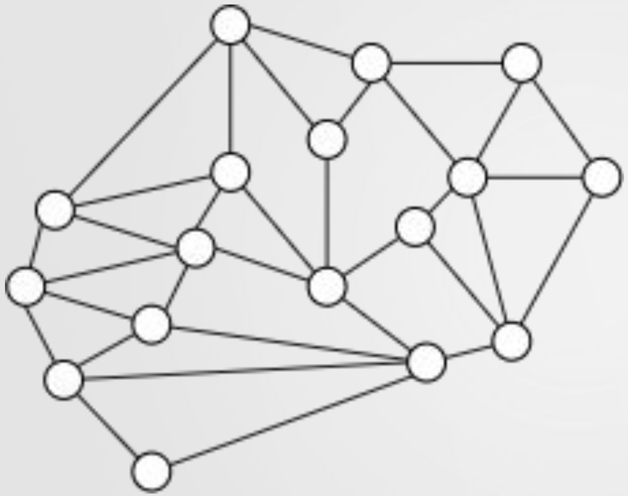
05 Graph neural networks





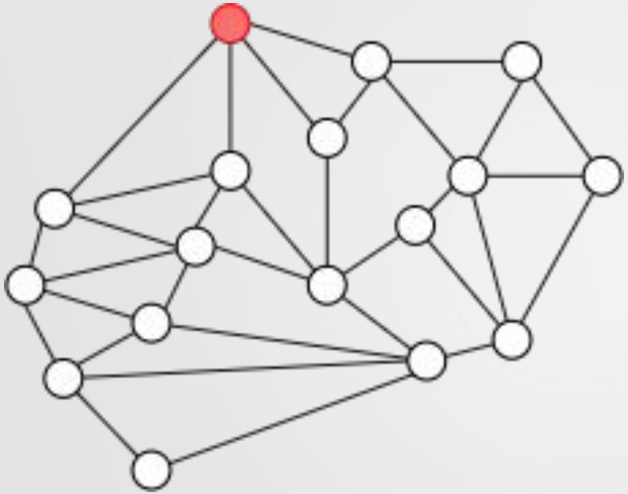
05 Graph neural networks

How much you have to **unroll**?



05 Graph neural networks

How much you have to **unroll**?



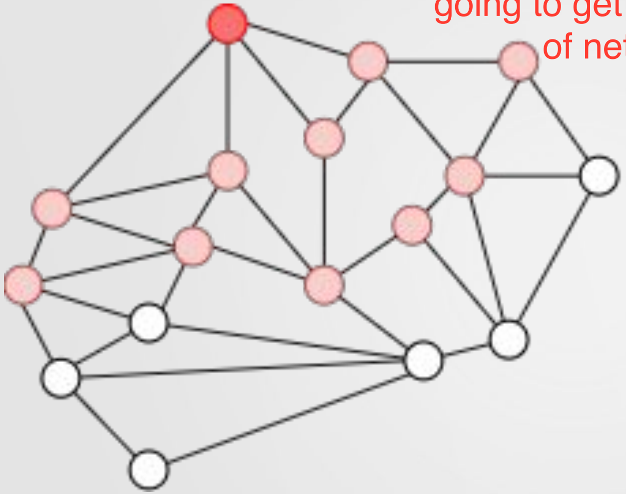
05 Graph neural networks

How much you have to **unroll**?

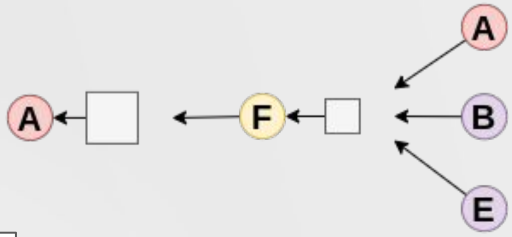
05 Graph neural networks

How much you have to **unroll**?

This will be a key of design consideration. If you unroll too much to do inference the red point, then you are going to get too much information from the wide range of network, which could be unnecessary.



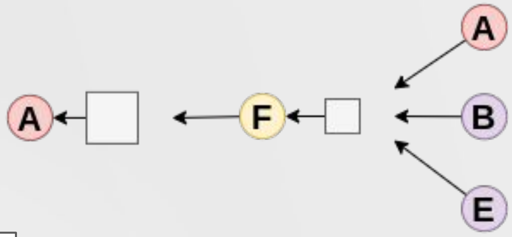
05 Graph neural networks



Math

$$H_v^0 = X_v$$

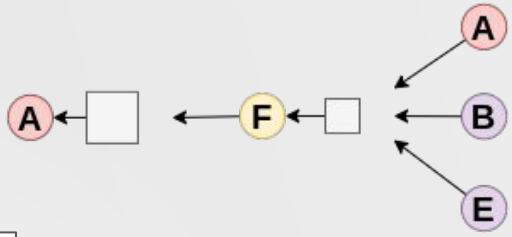
05 Graph neural networks



Math

$$H_v^0 = X_v$$

$$h_v^{k+1} = \sigma(W_k \sum_{u \in N(u)} \frac{h_u^k}{|N(v)|} + B_k h_v^k)$$



Math

$$H_v^0 = X_v$$

$$h_v^{k+1} = \sigma(W_k \sum_{u \in N(u)} \frac{h_u^k}{|N(v)|} + B_k h_v^k)$$

$$Z_v = h_v^K$$ K-th layer of Node v

05 Graph neural networks

$$h_v^{k+1} = \sigma(W_k \sum_{u \in N(u)} \frac{h_u^k}{|N(v)|} + B_k h_v^k)$$

05 Graph neural networks

$$h_v^{k+1} = \sigma(W_k \sum_{u \in N(u)} \frac{h_u^k}{|N(v)|} + B_k h_v^k)$$

It is the k+1 embedding of the node V

05 Graph neural networks

Embedding of node v, at the k-th embedding

$$\boxed{h_v^{k+1}} = \sigma(W_k \sum_{u \in N(u)} \frac{h_u^k}{|N(v)|} + B_k \boxed{h_v^k})$$

It is the k+1 embedding of the node V

05 Graph neural networks

Embedding of node v, at the k-th embedding

$$h_v^{k+1} = \sigma\left(W_k \boxed{\sum_{u \in N(u)} \frac{h_u^k}{|N(v)|}} + B_k \boxed{h_v^k}\right)$$

It is the k+1 embedding of the node V

For each neighbour of v (**u = N(v)**), we average the embedding of the embedding at the k-th layer

05 Graph neural networks

Weights

Embedding of node v, at the k-th embedding

$$h_v^{k+1} = \sigma\left(W_k \boxed{\sum_{u \in N(u)} \frac{h_u^k}{|N(v)|}} + B_k \boxed{h_v^k}\right)$$

It is the k+1 embedding of the node V

For each neighbour of v (**u = N(v)**), we average the embedding of the embedding at the k-th layer

05 Graph neural networks

Weights

Embedding of node v, at the k-th embedding

It is the k+1 embedding of the node V

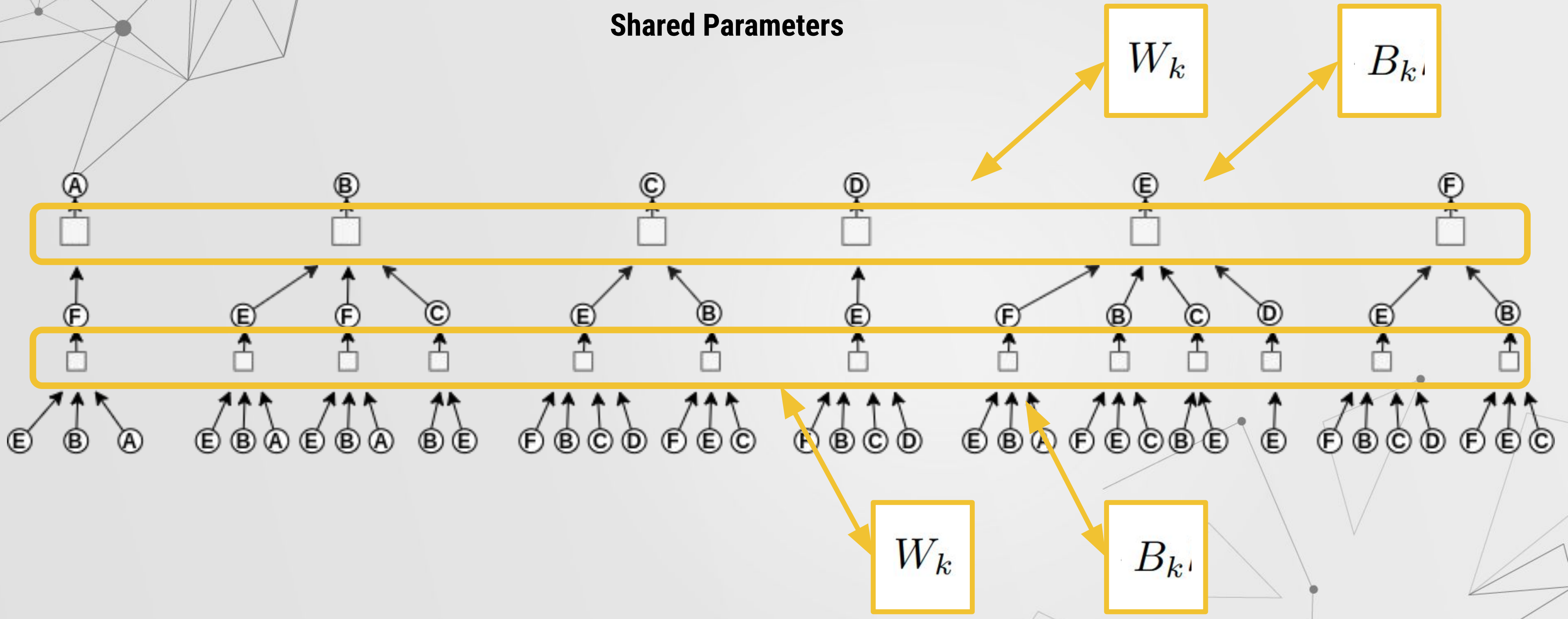
$$h_v^{k+1} = \sigma\left(W_k \sum_{u \in N(u)} \frac{h_u^k}{|N(v)|} + B_k h_v^k\right)$$

Activation function (**ReLU**)

For each neighbour of v (**u = N(v)**), we average the embedding of the embedding at the k-th layer

05 Graph neural networks

Shared Parameters



06 Graph SAGE

Inductive Representation Learning on Large Graphs

William L. Hamilton* wleif@stanford.edu
Rex Ying* rexying@stanford.edu
Jure Leskovec jure@cs.stanford.edu

Department of Computer Science
Stanford University
Stanford, CA, 94305

$$H_v^0 = X_v$$

$$h_v^{k+1} = \sigma(W_k \sum_{u \in N(u)} \frac{h_u^k}{|N(v)|} + B_k h_v^k)$$

$$Z_v = h_v^K$$

06 Graph SAGE

$$h_v^{k+1} = \sigma(W_k \sum_{u \in N(u)} \frac{h_u^k}{|N(v)|} + B_k h_v^k)$$

$$h_v^{k+1} = \sigma([W_k \cdot AGG(\{h_u^{k-1}, \forall u \in N(v)\}), B_k h_v^k])$$

06 Graph SAGE

$$h_v^{k+1} = \sigma(W_k \boxed{\sum_{u \in N(u)} \frac{h_u^k}{|N(v)|}} + B_k h_v^k)$$

$$h_v^{k+1} = \sigma([W_k \boxed{\cdot AGG(\{h_u^{k-1}, \forall u \in N(v)\}}), B_k h_v^k])$$

06 Graph SAGE

$$h_v^{k+1} = \sigma(W_k \boxed{\sum_{u \in N(u)} \frac{h_u^k}{|N(v)|}} + B_k h_v^k)$$

$$h_v^{k+1} = \sigma([W_k \cdot \boxed{AGG(\{h_u^{k-1}, \forall u \in N(v)\})}, B_k h_v^k])$$

06 Graph SAGE

$$h_v^{k+1} = \sigma([W_k \cdot \boxed{AGG}(\{h_u^{k-1}, \forall u \in N(v)\}), B_k h_v^k])$$

AGG:

- **AGG → POOL**: es: element-wise min/max
- **AGG → LSTM**: (note not order invariant)

07 Practice

Jupyter-notebook