A Primer on Graph Neural Networks

Emanuele Rossi, Imperial College London

Colt, November 2023

But First, A Little Bit About Myself

It All Started in Imola

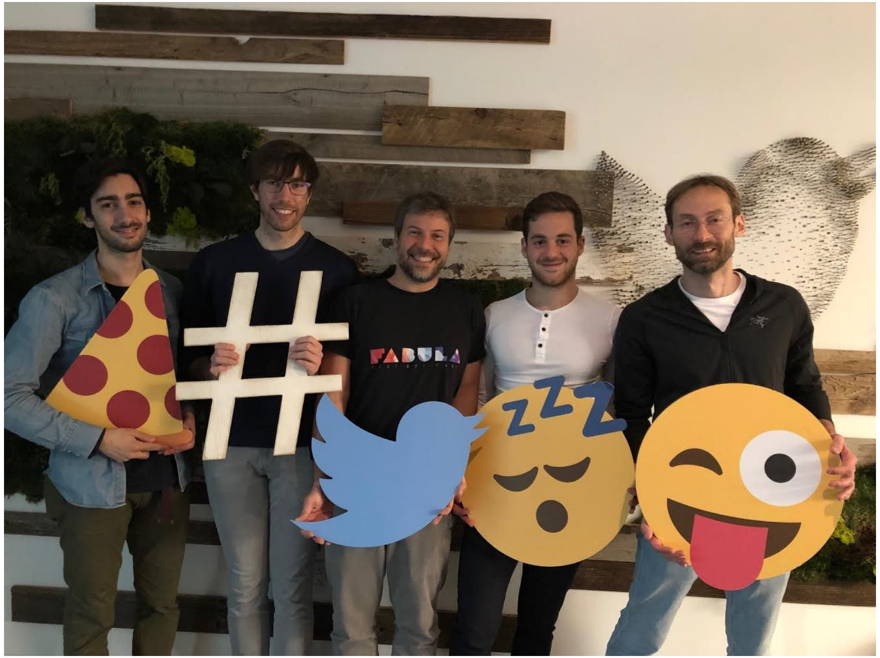
Even though I don't like F1…

2015: I Then Moved to London

2018: MPhil @ Cambridge

2019: Fabula AI

2019: Twitter acquires Fabula



2020: Start PhD with Twitter and Imperial

2021: Remote from Barcelona

2022: What the Heck

So I Took a Break

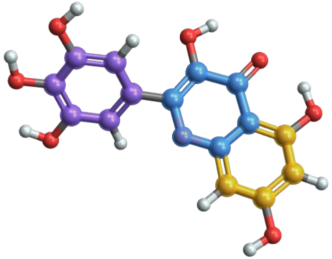
2024: Drug Discovery @ Vant AI

Why Should We Care About Graph Neural Networks?

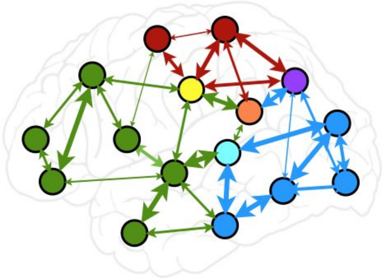
Networks are everywhere

And graphs are a great way to model them

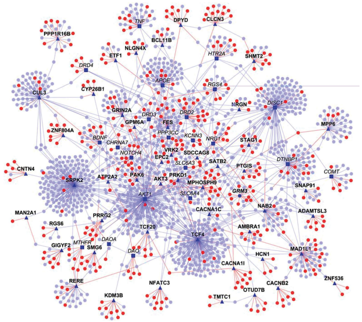
Social Networks



Molecules



Functional Networks



Interaction Networks

Image Credit: https://www.wolfram.com/language/12/molecular-structure-and-computation/molecule-graphs.html.en?product=mathematica
Image Credit: https://gatton.uky.edu/about-us/stay-connected/news/2020/links-center-social-network-analysis-workshop-success
Image Credit: Papo et al., “Reconstructing functional brain networks: have we got the basics right?”, Frontiers in Human Neuroscience 2014
Image Credit: Madhavicmu / Wikimedia Commons / CC-BY-SA-4.0

Networks are everywhere

And graphs are a great way to model them

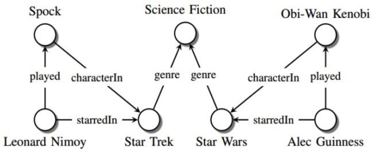


Image credit: Maximilian Nickel et al

Knowledge Graphs

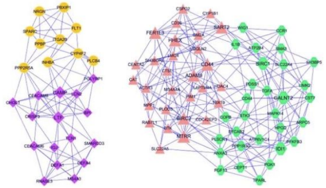


Image credit: ese.wustl.edu

Regulatory Networks

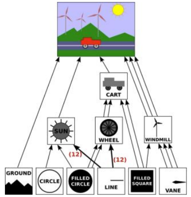


Image credit: math.hws.edu

Scene Graphs

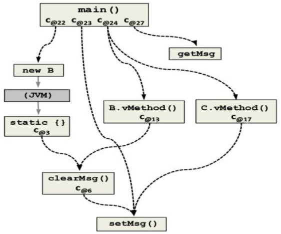


Image credit: ResearchGate

Code Graphs

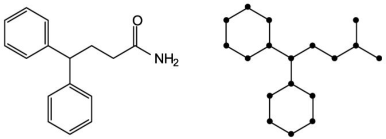


Image credit: MDPI

Molecules

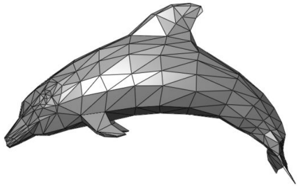
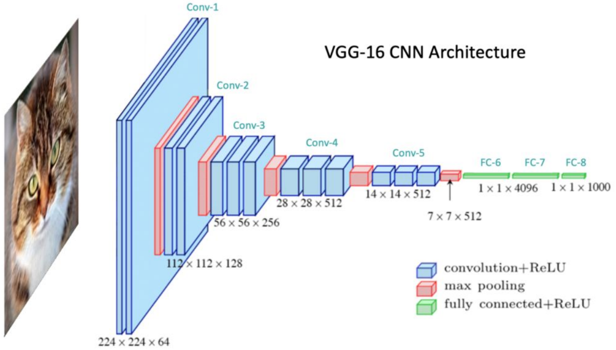
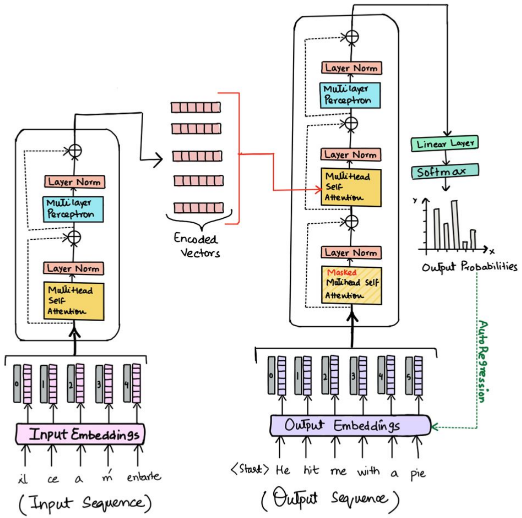


Image credit: Wikipedia

3D Shapes

What Models Should We Use?





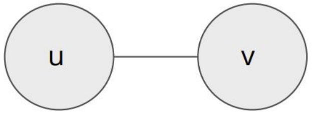
?

Graphs and Graph Tasks

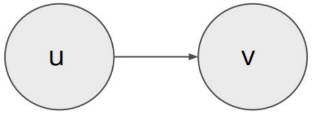
Graph Definition

$G = (V, E)$

- V is a set of nodes
- E is a set of tuples of form (u, v), where there is an edge between u and v
- G is a graph



Undirected edge



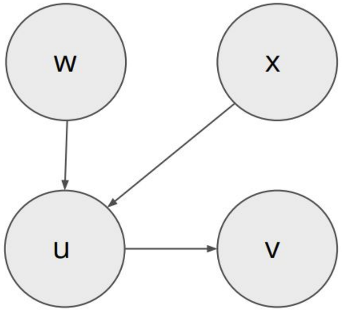
Directed edge

The Adjacency Matrix

Adjacency Matrix: $\mathbf{A} \in \mathbb{R}^{|V| \times |V|}$

- In this example, binary matrix encoding of a unweighted graph
- Rows/columns number the nodes, matrix elements encode edges

V = {u, v, w, x}; E = {(w, u), (x, u), (u, v)}



A =

(to)

	u	v	w	x	(from)
u	0	**1**	0	0	
v	0	0	0	0	
w	**1**	0	0	0	
x	**1**	0	0	0	

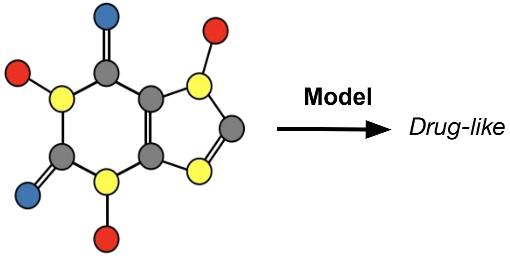
Do the matrices encode the same graph?

0	**1**	0	0
0	0	0	0
1	0	0	0
1	0	0	0

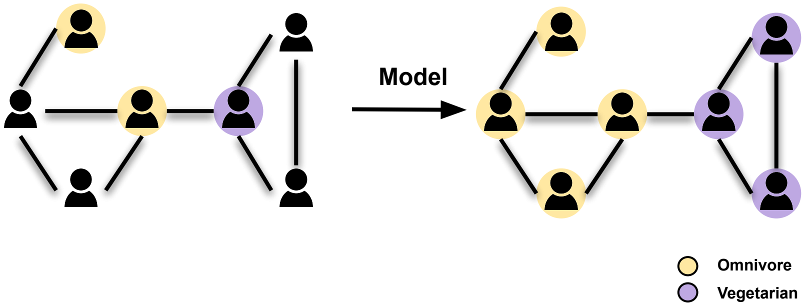
0	0	0	0
0	0	**1**	0
1	0	0	0
0	0	**1**	0

Hint: Have we given you enough information?

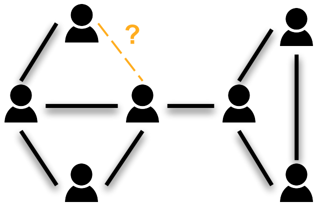
Tasks on Graphs



Graph Classification



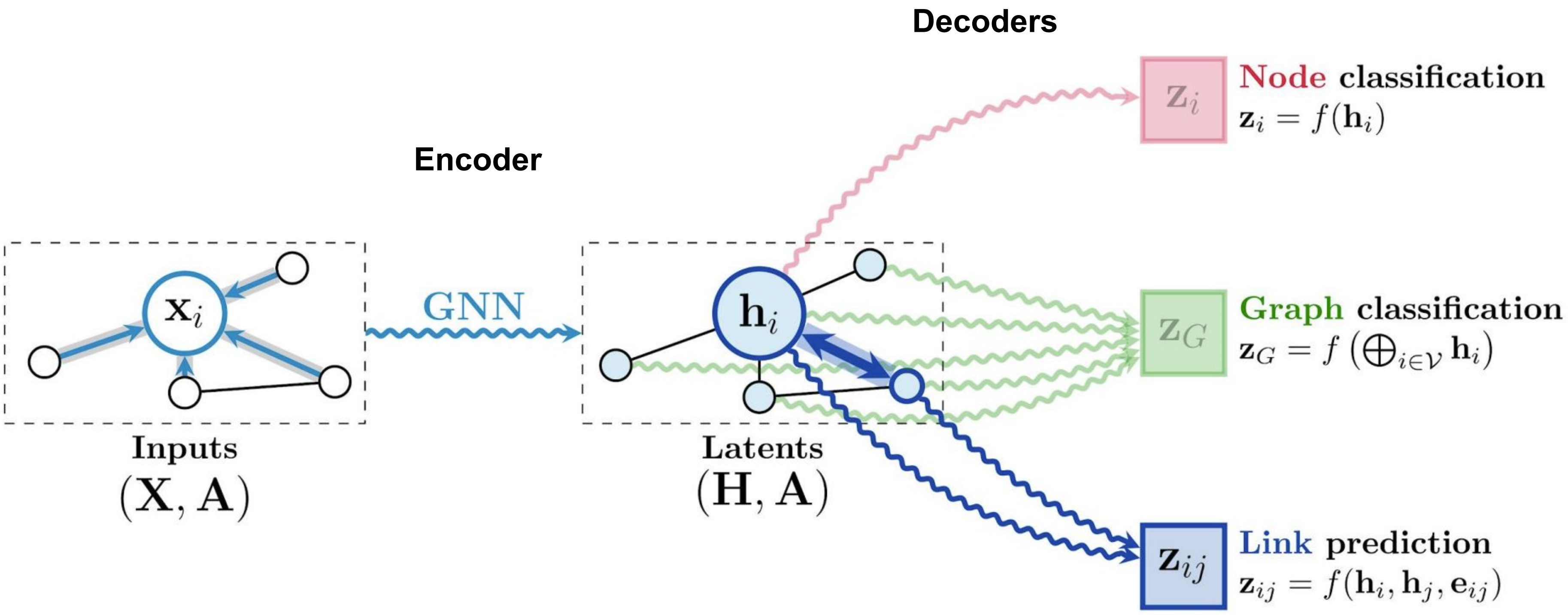
Node Classification



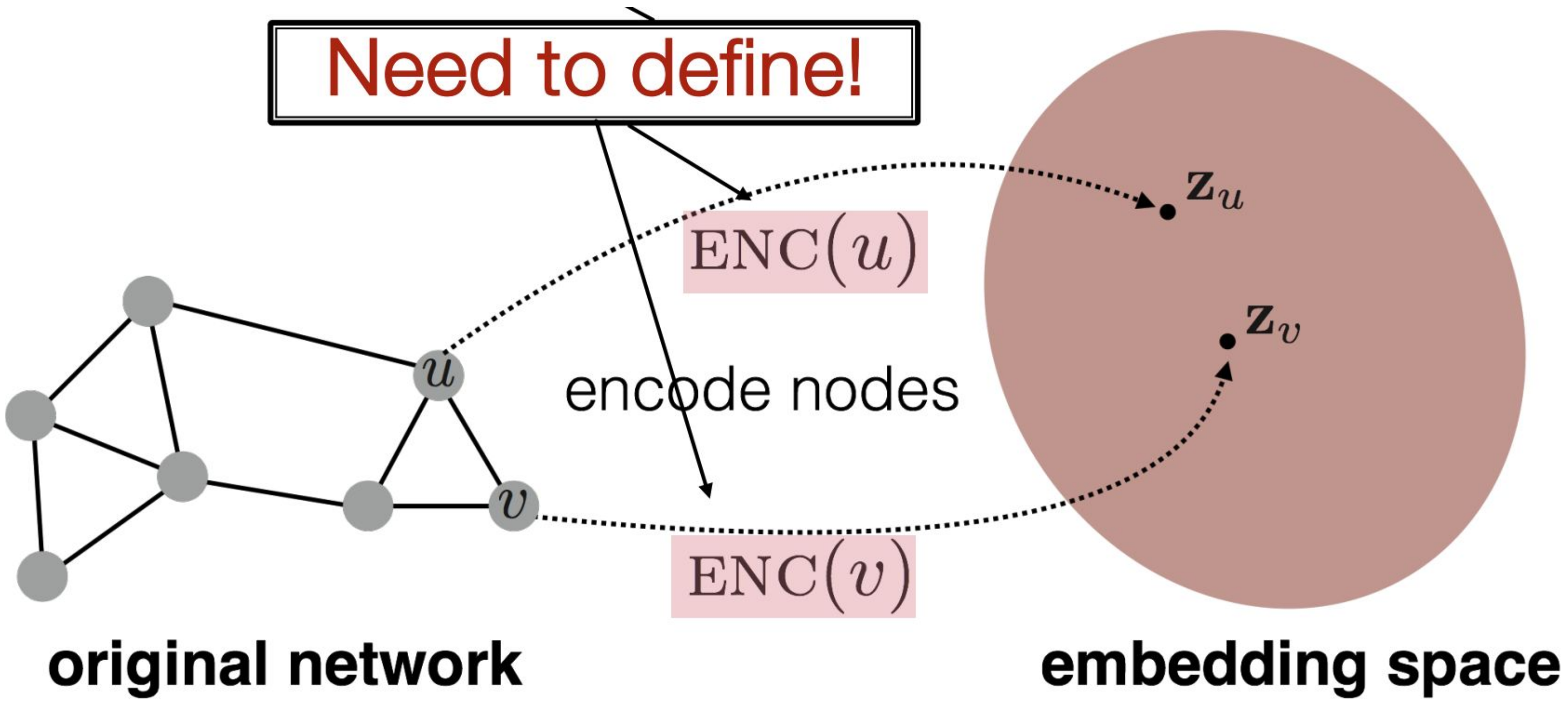
Link Prediction

Encoder-Decoder Framework

Encoder-Decoder Framework

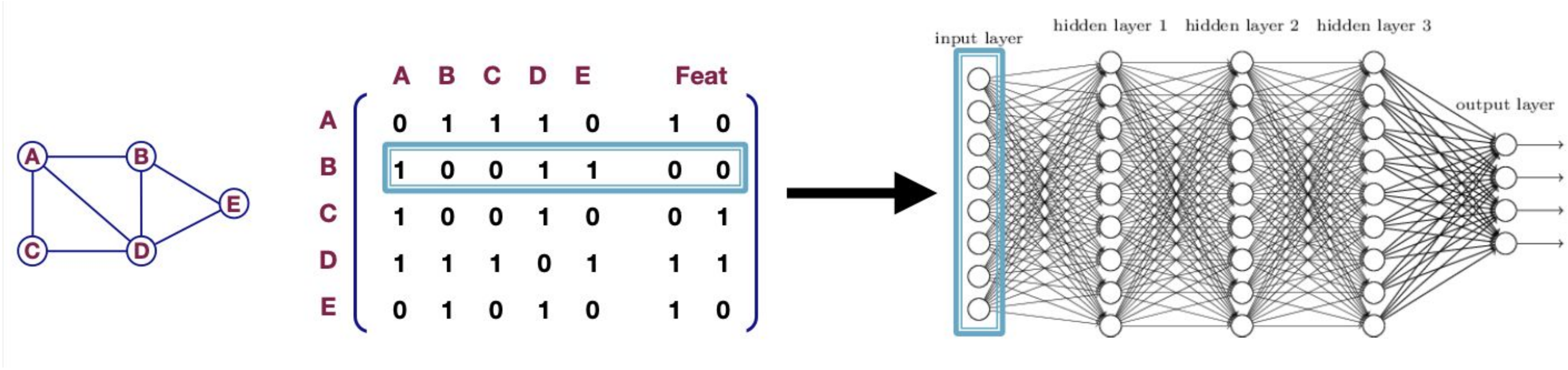


Focus on the GNN Encoder



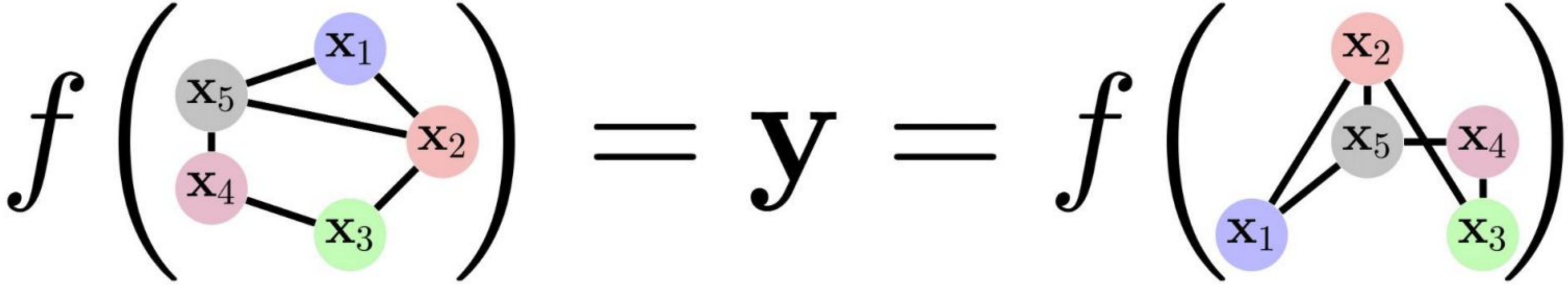
GNN Encoders

MLP for Graphs



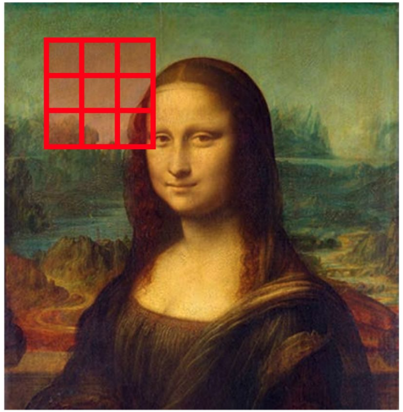
- O(|V|) parameters
- Cannot handle graphs of different size
- **Gives different results for different orderings of the graph**

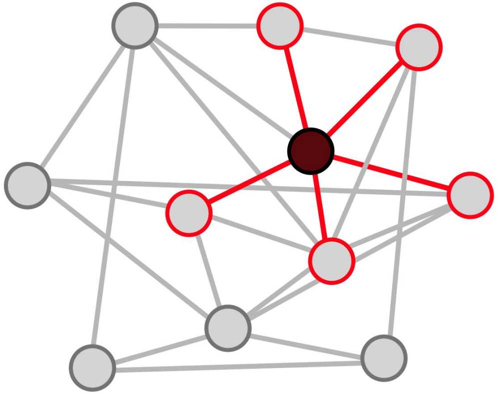
Permutation Invariance



Invariance: $$f(\mathbf{PX}, \mathbf{PAP}^\top) = f(\mathbf{X}, \mathbf{A})$$

Starting from Convolution

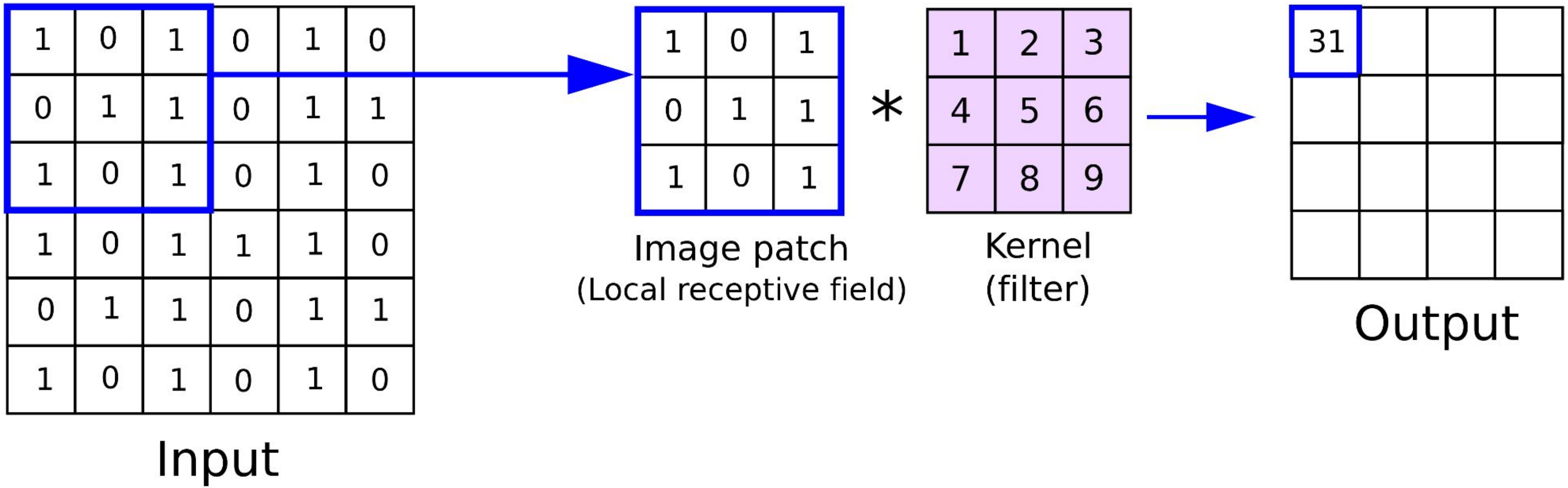




Convolutional Neural Networks

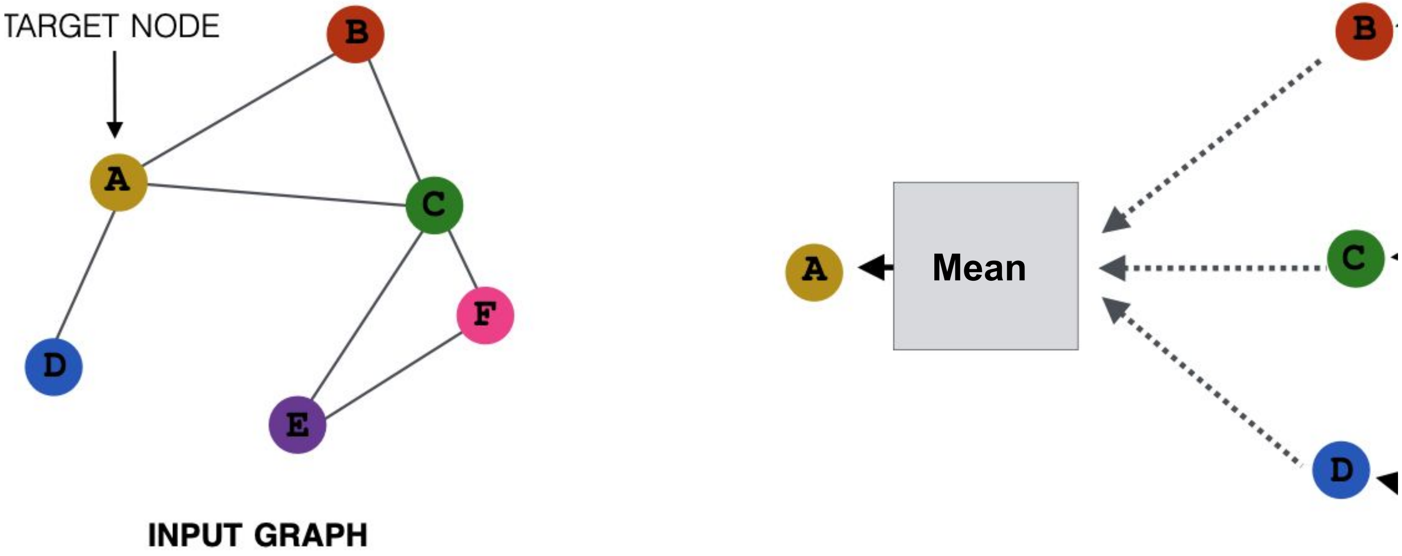
Graph Neural Networks

Convolution is a Weighted Average of Neighboring Pixels



Idea: Take Plain Average of Neighbors

Plus a transformation shared by all neighbors



Simple Graph Convolutional Layer

$$\mathbf{h}_i^{(l)} = \sigma\left(\frac{1}{|\mathcal{N}(i)|} \boxed{\sum_{j \in \mathcal{N}(i)}} \mathbf{h}_j^{(l-1)} \boxed{\mathbf{W}^{(l-1)}}\right)$$

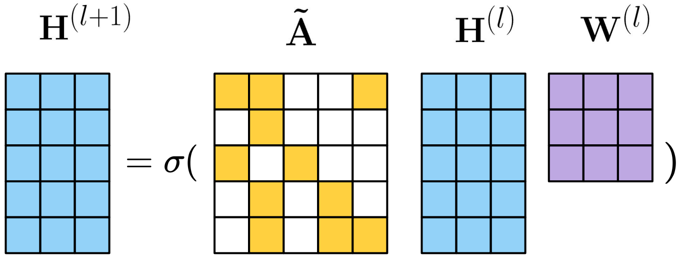
Sum is a permutation invariant operator

Single weight matrix shared by all nodes, compared to convolution where you have kxk matrices

Simple Graph Convolutional Layer

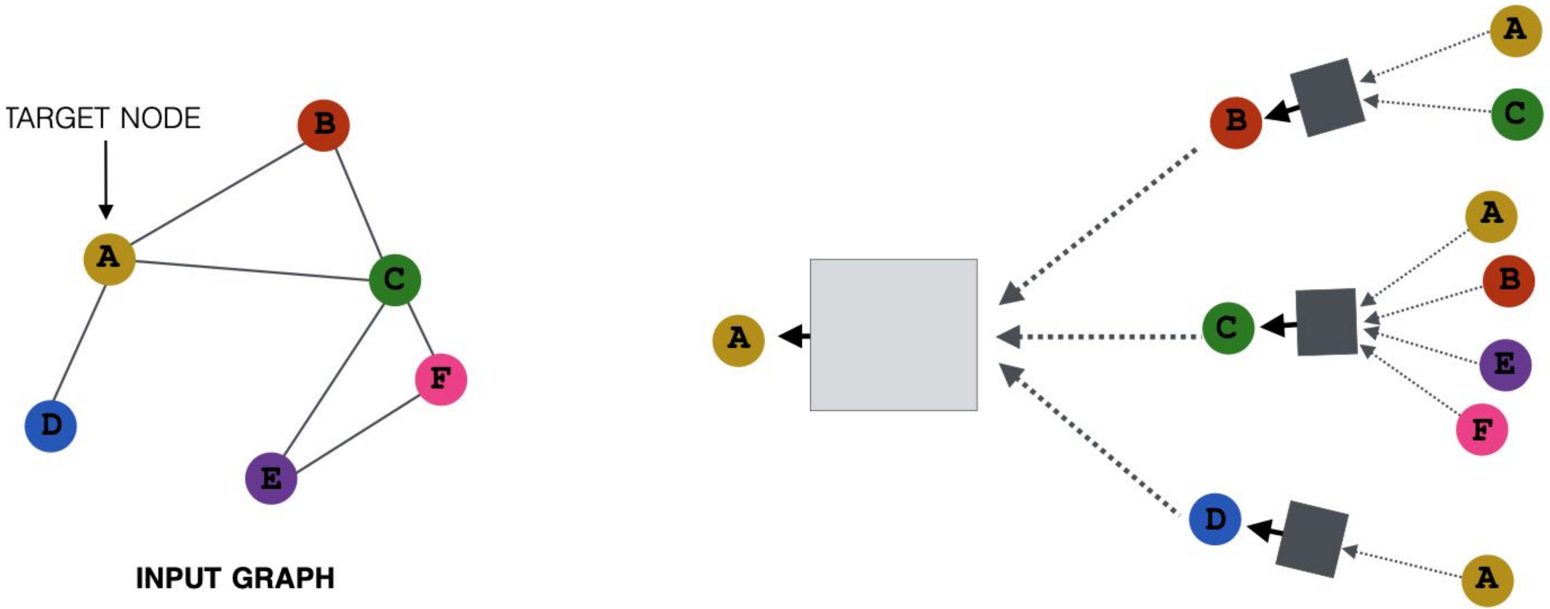
In matrix form

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)})$$
$$\mathbf{H}^{(0)} = \mathbf{X}$$

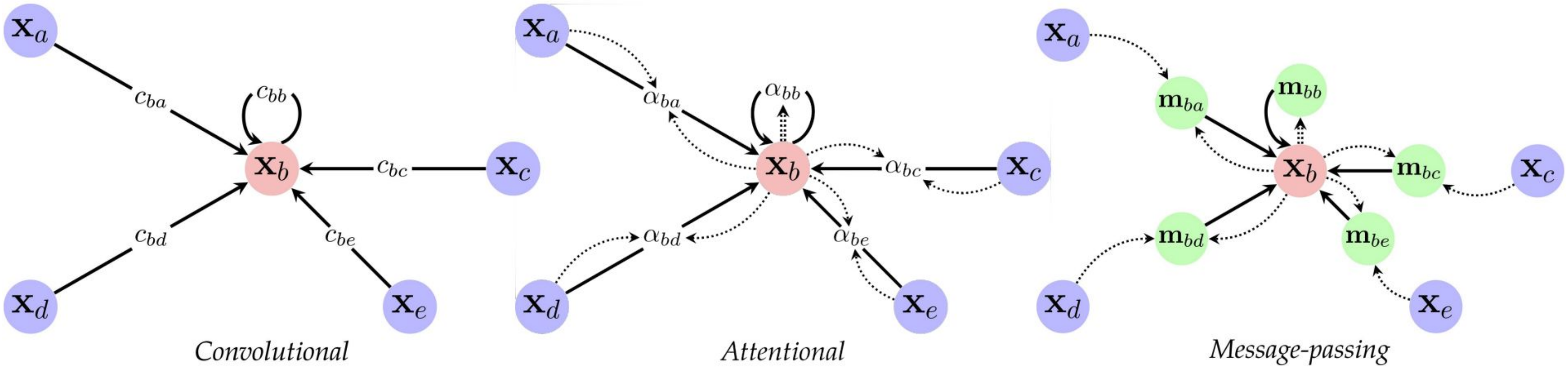


Stacking Multiple Layers

It allows to aggregate from further away in the graph



The three “flavours” of GNN layers



$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij}\psi(\mathbf{x}_j)\right)$$

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j)\psi(\mathbf{x}_j)\right)$$

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

Taken from [Petar Velickovic’s Slides]

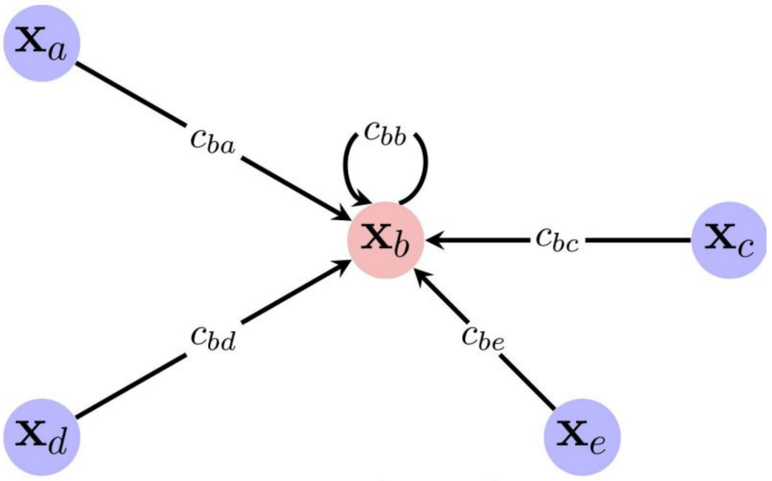
Convolutional GNN

- Features of neighbours aggregated with fixed weights, c_{ij}

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij}\psi(\mathbf{x}_j)\right)$$

- Usually, the weights depend directly on **A**.
  - ChebyNet (Defferrard *et al.*, NeurIPS'16)
  - GCN (Kipf & Welling, ICLR'17)
  - SGC (Wu *et al.*, ICML'19)

- Useful for **homophilous** graphs and **scaling up**
  - When edges encode *label similarity*



Convolutional

Taken from Petar Velickovic's Slides

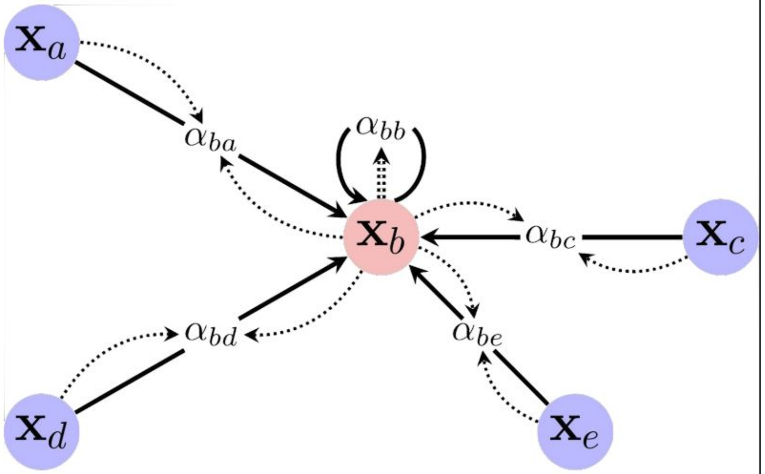
Attentional GNN

- Features of neighbours aggregated with **implicit** weights (via *attention*)

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j)\psi(\mathbf{x}_j)\right)$$

- Attention weight computed as $\alpha_{ij} = a(\mathbf{x}_i, \mathbf{x}_j)$
  - MoNet (Monti *et al.*, CVPR'17)
  - GAT (Veličković *et al.*, ICLR'18)
  - GaAN (Zhang *et al.*, UAI'18)

- Useful as "middle ground" w.r.t. **capacity** and **scale**
  - Edges need not encode homophily
  - But still compute *scalar* value in each edge

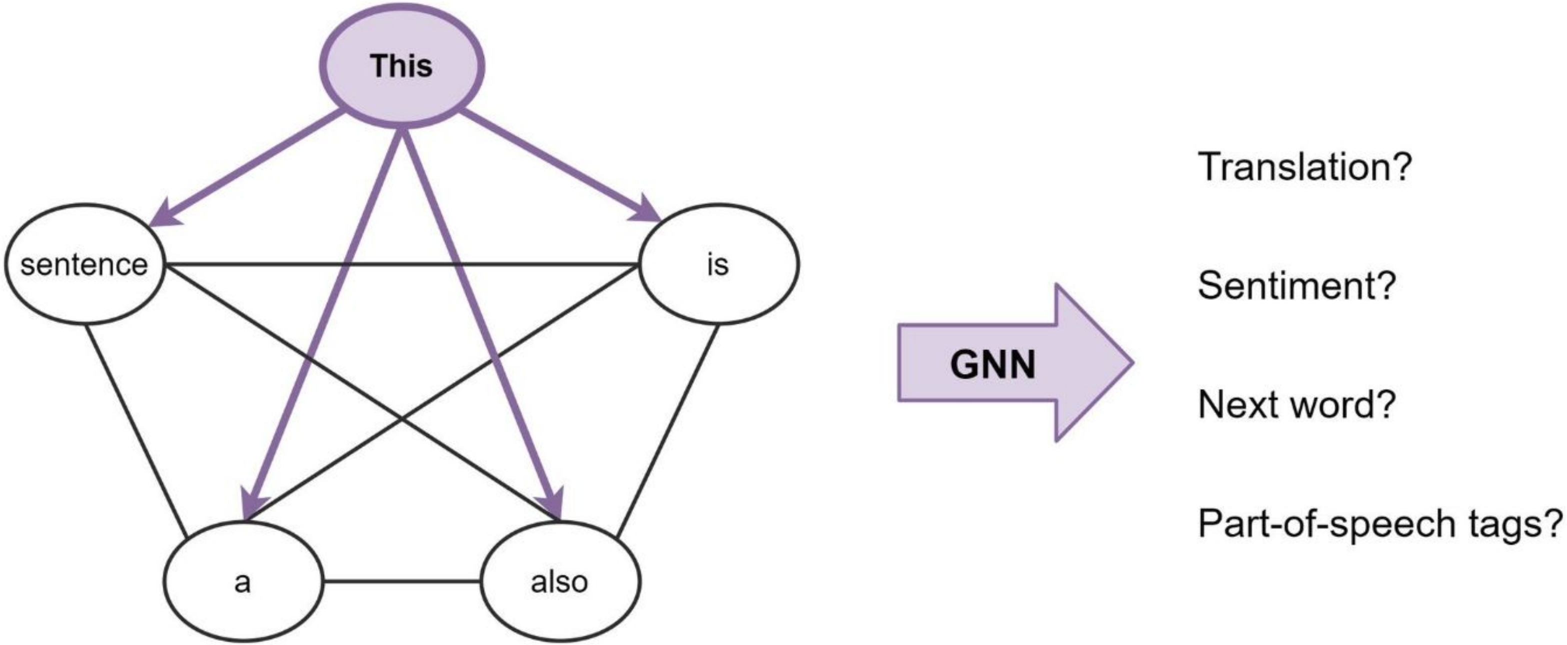


Attentional

Taken from [Petar Velickovic's Slides]

Transformers are GNNs

On the fully connected graphs



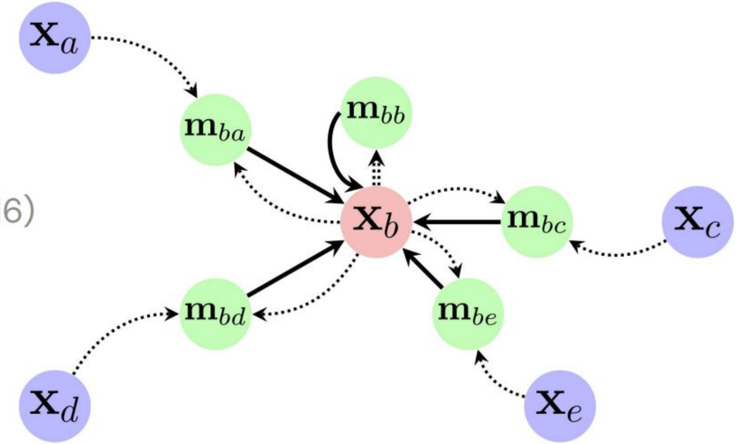
See “Transformers are Graph Neural Networks”

Message-passing GNN

- Compute **arbitrary vectors** (*"messages"*) to be sent across edges

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

- Messages computed as $\mathbf{m}_{ij} = \psi(\mathbf{x}_i, \mathbf{x}_j)$
  - Interaction Networks (Battaglia *et al.*, NeurIPS'16)
  - MPNN (Gilmer *et al.*, ICML'17)
  - GraphNets (Battaglia *et al.*, 2018)



Message-passing

- Most **generic** GNN layer
  - May have *scalability* or *learnability* issues
  - Ideal for *computational chemistry*, *reasoning* and *simulation*

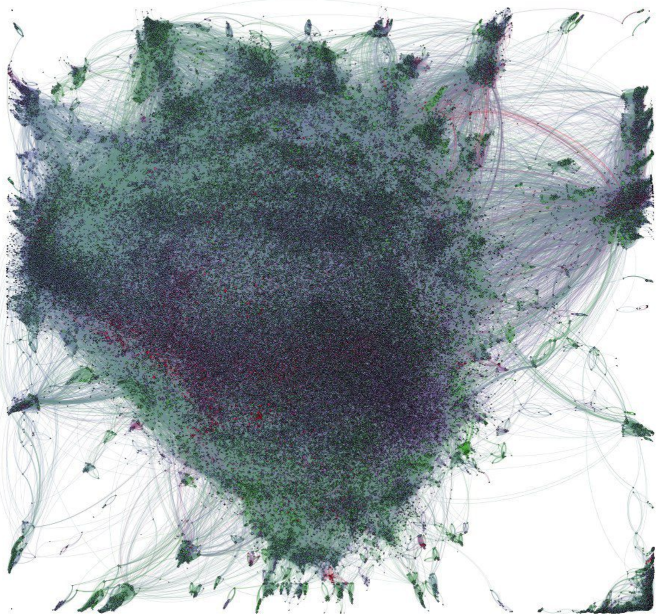
Taken from [Petar Velickovic's Slides]

My Research: Challenges in the Real World

Scalability, Temporality, Missing Data, Directed Graphs

Scalability [1, 2]

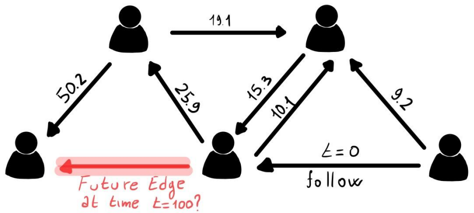
Learning on web-scale graphs



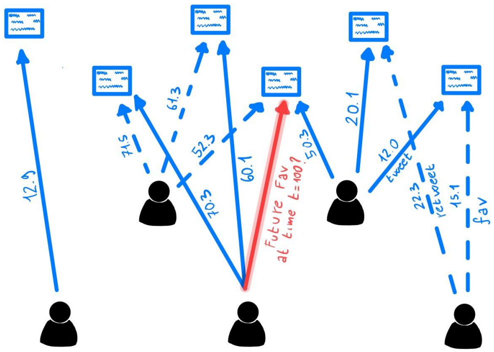
[1] Rossi et al., “SIGN: Scalable Inception Graph Neural Networks”, ICML 2020 GRL Workshop;
[2] Chamberlain et al., “Link Prediction with Subgraph Sketching”, ICLR 2023;

Dynamic Graphs [3, 4]

Graphs changing over time



Social Networks

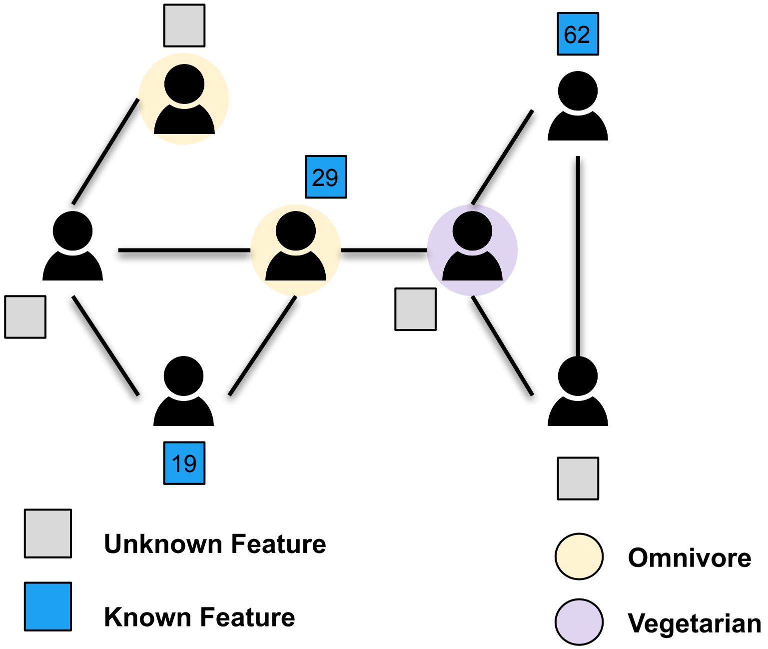


Interaction Networks

[3] Rossi et al., "Temporal Graph Networks For Deep Learning On Dynamic Graphs", ICML 2020 GRL Workshop;
[4] S. Huang et al., "Temporal Graph Benchmark for Machine Learning on Temporal Graphs", NeurIPS 2023 Datasets and Benchmarks;

Missing Node Features [5]

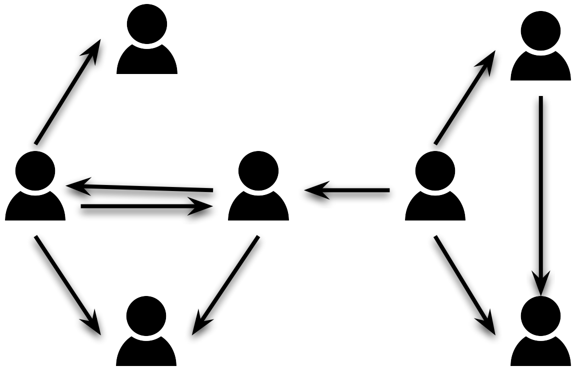
Think of user demographics (eg. age) in a social network



[5] Rossi et al., "On the Unreasonable Effectiveness of Feature propagation in Learning on Graphs with Missing Node Features", LoG 2022;

Directed Graph Neural Networks [6]

When edges have a direction



[6] Rossi et al., “Edge Directionality Improves Learning on Heterophilic Graphs“, LoG 2023

Resources and Tools

Great resources to learn more

- A Gentle Introduction to Graph Neural Networks (Distill Blog Post)
- Stanford CS224W: Machine Learning with Graphs
- Graph Neural Networks: Foundations, Frontiers, and Applications
- PyG: PyTorch best library for GNNs

Questions?

@emaros96
www.emanuelerossi.co.uk