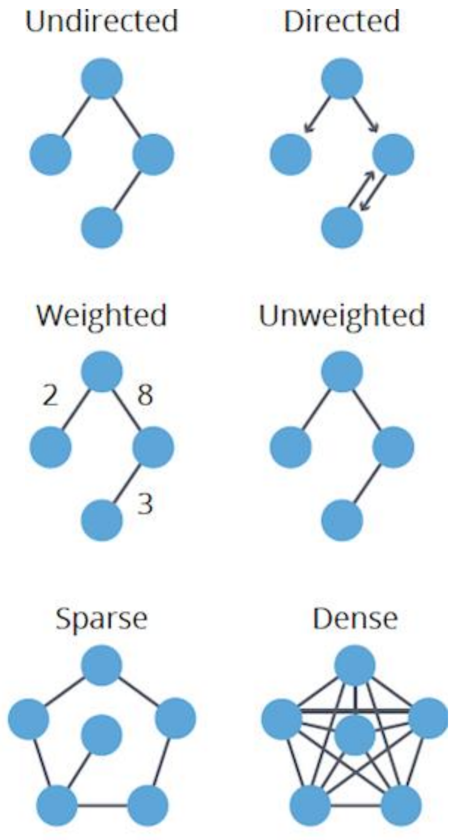
Setup Instructions:

1. Navigate to https://github.com/GageDeZoort/prc_gnn_tutorial
2. Click on gnn_tutorial_3_27_23.ipynb
3. Change "github" to "githubtocolab" in your searchbar, i.e.: https://colab.research.google.com/github/GageDeZoort/prc_gnn_tutorial
4. If you want to use a GPU (none of the cells require one), you can request one on Colab through Edit->Notebook Settings

GRAPHS

- Graphs represent relational data
  - Entities → Nodes: $u \in \mathcal{V}$
    - Node features: $\boldsymbol{x}_u \in \mathbb{R}^{d_{\mathcal{V}}}$
  - Relations → Edges: $(u, v) \in \mathcal{E}$
    - Edge features: $\boldsymbol{e}_{uv} \in \mathbb{R}^{d_{\mathcal{E}}}$



Graph Algorithms in Neo4j: Graph Algorithm Concepts

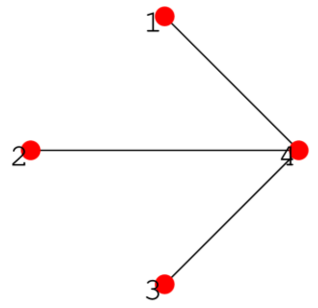
Directed edges specify an incoming and outgoing node

Edge features might be weights or otherwise more complicated attributes

Sparsity is problem-dependent… roughly, sparse if $|\mathcal{E}| \ll |\mathcal{V}|^2$

EDGE REPRESENTATIONS

- Graphs represent relational data
  - Entities → Nodes: $u \in \mathcal{V}$
    - Node features: $\boldsymbol{x}_u \in \mathbb{R}^{d_\mathcal{V}}$
  - Relations → Edges: $(u, v) \in \mathcal{E}$
    - Edge features: $\boldsymbol{e}_{uv} \in \mathbb{R}^{d_\mathcal{E}}$



Edge Representations

Adjacency Matrices
$A_{adjacency} \in \mathbb{R}^{d_\mathcal{V} \times d_\mathcal{V}}$

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

Incidence Matrices
$A_{incidence} \in \mathbb{R}^{d_\mathcal{V} \times d_\mathcal{E}}$

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

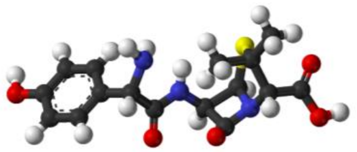
PyTorch geometric

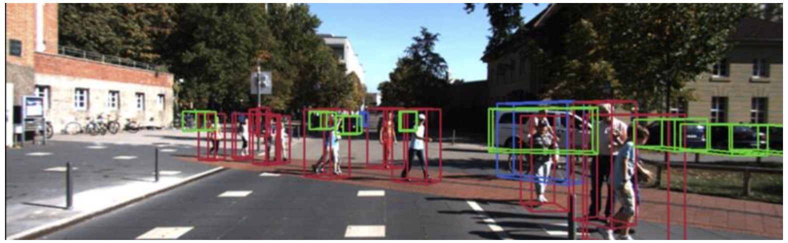
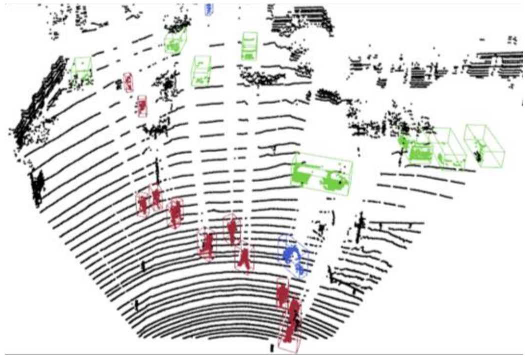
Sparse Index Lists (COO)
$I_{COO} \in \mathbb{R}^{2 \times d_\mathcal{E}}$

$$\begin{bmatrix} [1 & 2 & 3] \\ [4 & 4 & 4] \end{bmatrix}$$

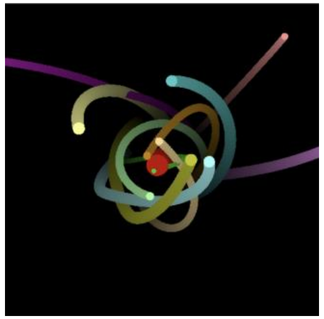
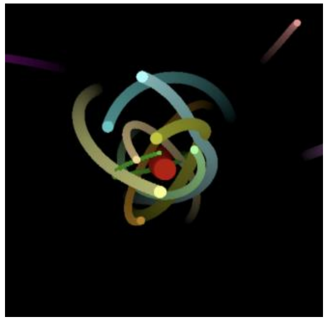
Graph Algorithms in Neo4j: Graph Algorithm Concepts

GRAPH LEARNING TASKS

DRUG DISCOVERY $GNN($  $) \rightarrow$ molecule property

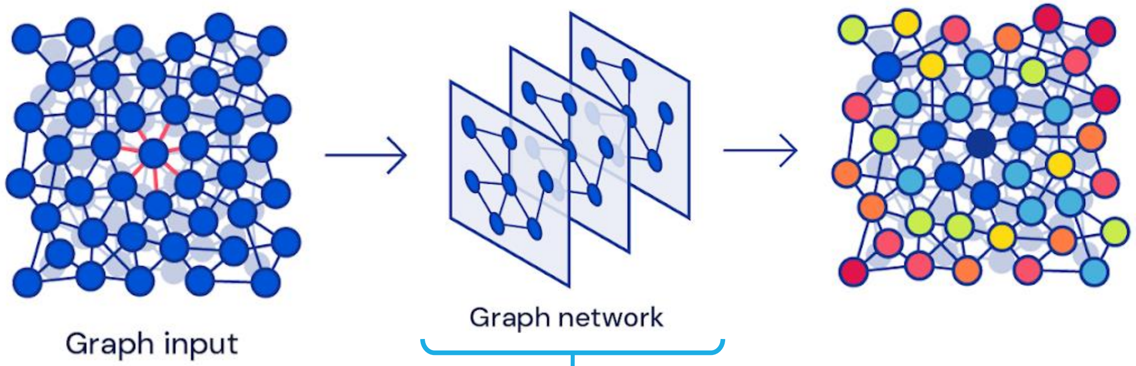
INSTANCE SEGMENTATION $GNN($  $) \rightarrow$ 

2003.01251.pdf (arxiv.org)

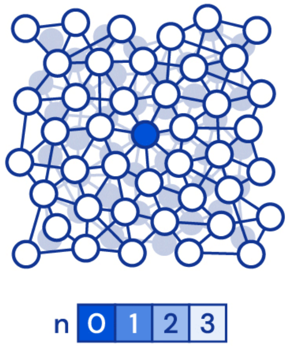
PHYSICS SIMULATION $GNN($  $) \rightarrow$ 

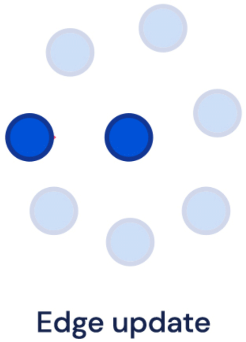
[1612.00222] Interaction Networks for Learning about Objects, Relations and Physics (arxiv.org)

GNNs: High-Level View



GNN Layers, Neural Message Passing

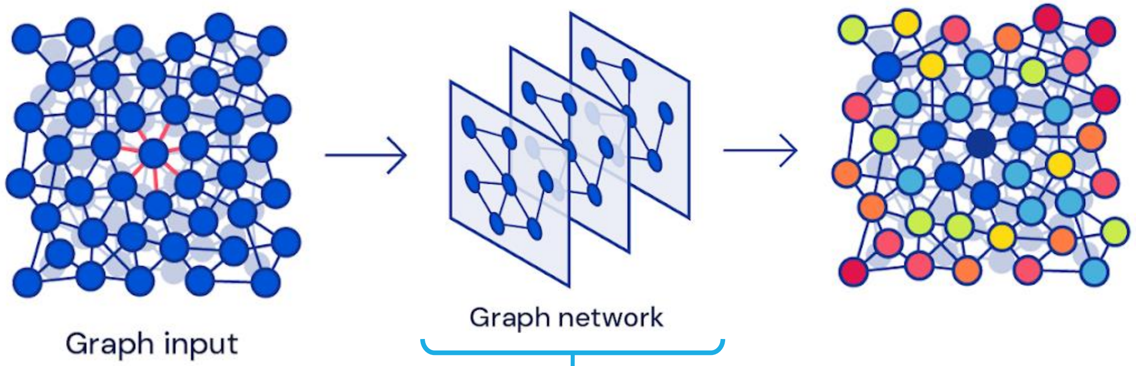




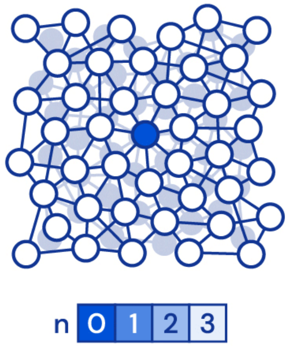
"Messages" computed from each node's neighborhood are used to update graph features… *k* iterations → info from *k*-hop neighborhoood

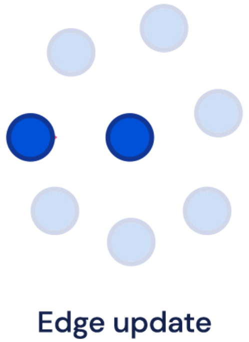
Figure Source: https://deepmind.com/blog/article/Towards-understanding-glasses-with-graph-neural-networks

GNNs: High-Level View



GNN Layers, Neural Message Passing





"Messages" computed from each node's neighborhood are used to update graph features… k iterations → info from k-hop neighborhoood

Message Passing (MPNN) Layers:

Framework for many equivariant graph updates

At each layer k, compute messages in each node's neighborhood:

$$\boldsymbol{m}_{uv}^{(k)} = \psi^{(k)}\left(\boldsymbol{h}_u^{(k-1)}, \boldsymbol{h}_v^{(k-1)}, \boldsymbol{e}_{uv}^{(k-1)}\right)$$

Aggregate messages in a permutation-invariant way:

Messages passed only from u's direct neighbors

$$\boldsymbol{a}_u^{(k)} = \bigoplus_{v \in N(u)} \boldsymbol{m}_{uv}^{(k)}$$

Any permutation invariant operation (e.g. sum, mean, max)

Update the node's state based on the messages it received:

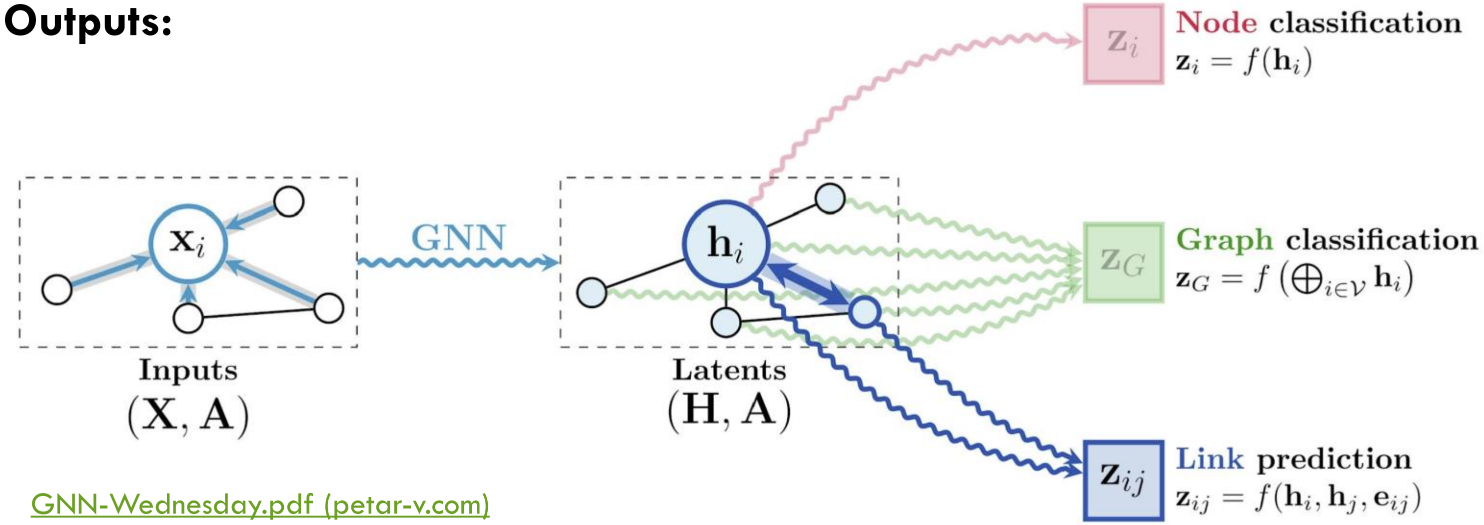
$$\boldsymbol{h}_u^{(k)} = \phi^{(k)}(\boldsymbol{h}_u^{(k-1)}, \boldsymbol{a}_u^{(k)})$$

Figure Source: https://deepmind.com/blog/article/Towards-understanding-glasses-with-graph-neural-networks

Generic MPNN Layers:

$$\boldsymbol{h}_u^{(k)} = \phi^{(k)}\left[\boldsymbol{h}_u^{(k-1)}, \bigoplus_{v \in N(u)} \psi^{(k)}\left(\boldsymbol{h}_u^{(k-1)}, \boldsymbol{h}_v^{(k-1)}, \boldsymbol{e}_{uv}^{(k-1)}\right)\right]$$

Outputs:



[GNN-Wednesday.pdf (petar-v.com)]()

Tutorial Agenda

0. Software Setup
1. Graph-Structured Data
    1. Karate Club (node classification)
    2. ENZYMES (graph classification)
2. Graph Neural Networks
    1. GCN Math
    2. Training on Cora
3. Long Exercise: graph-level training with ENZYMES (Optional)
4. Message Passing Under the Hood (Optional)