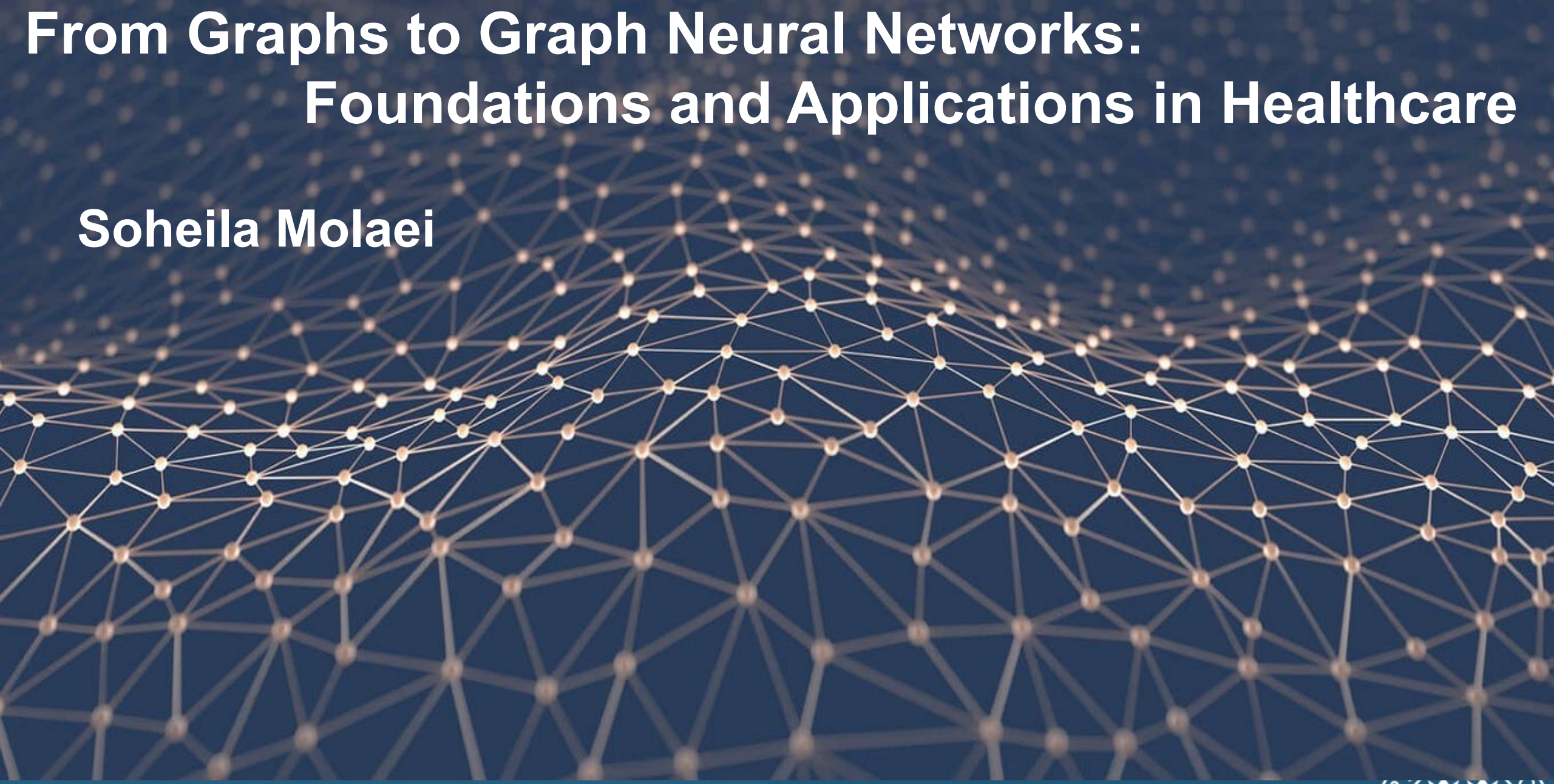
From Graphs to Graph Neural Networks: Foundations and Applications in Healthcare

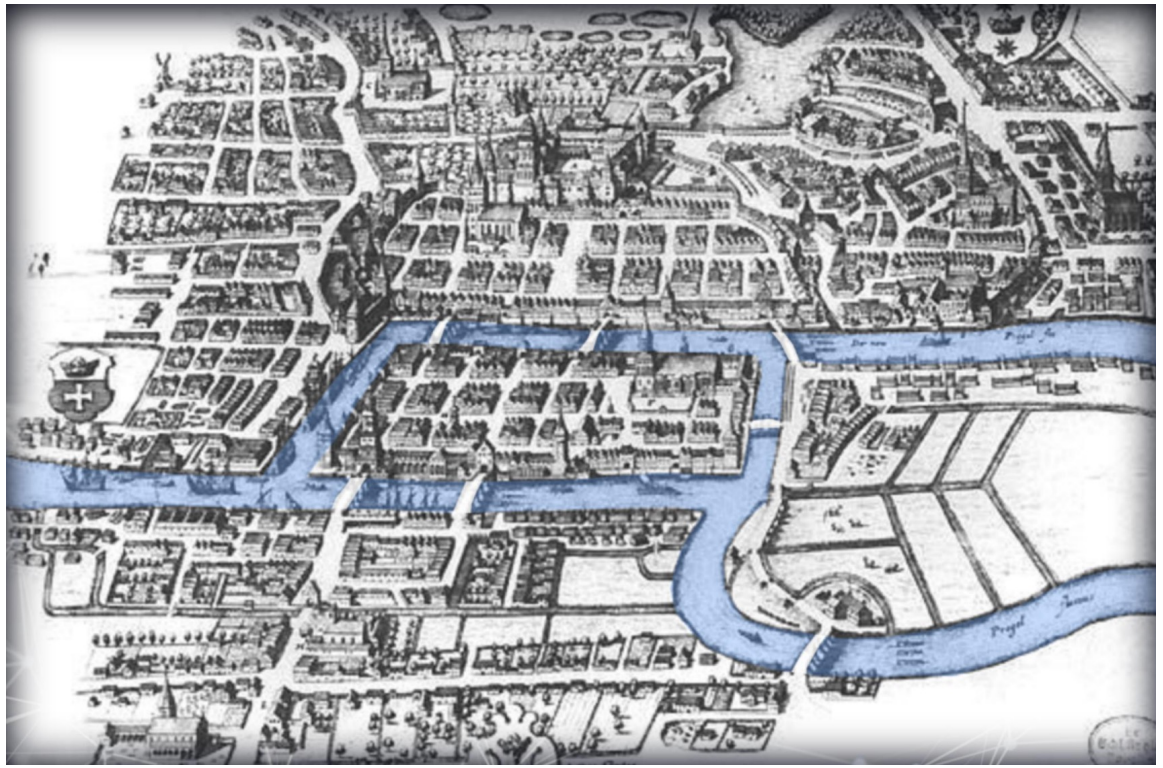
Soheila Molaei

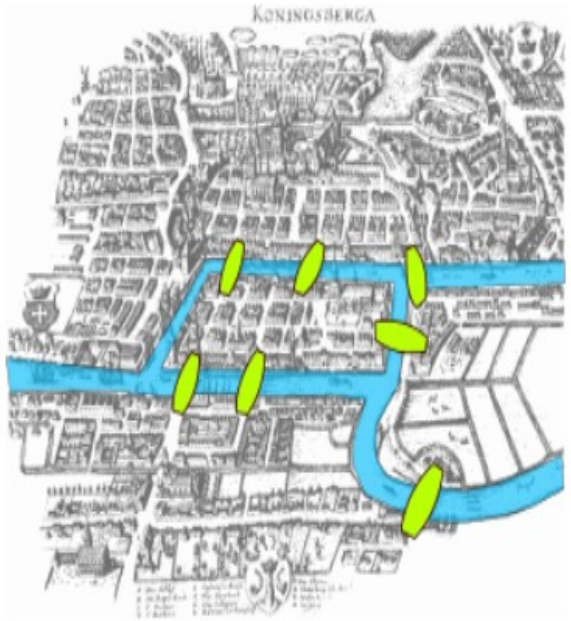


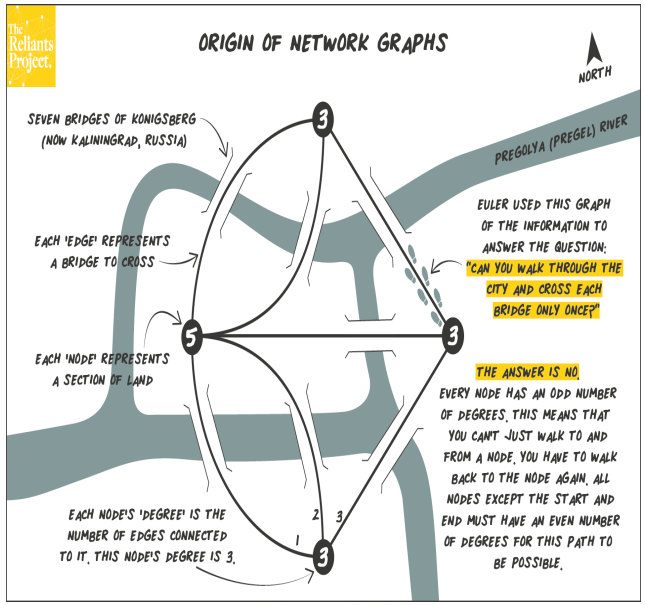
Computational
Health
Informatics

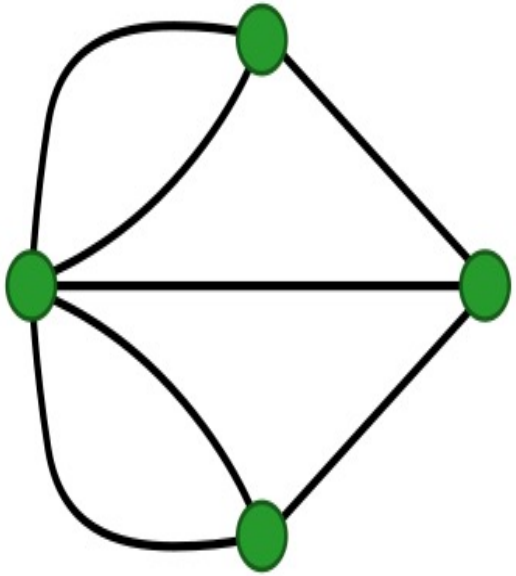
Seven Bridges of Königsberg

Can you walk through the city crossing all bridges only once?



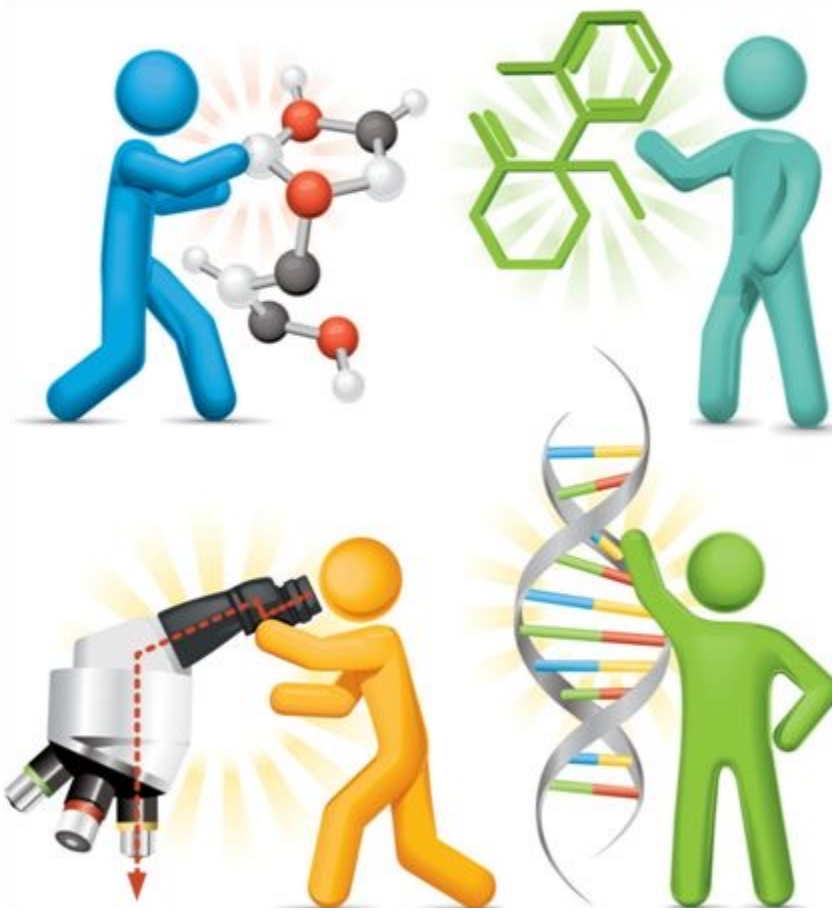






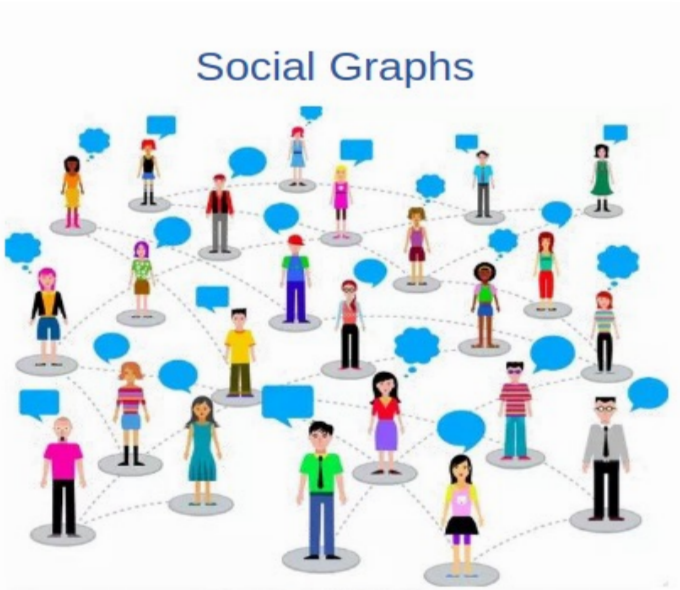
Not possible!

Why Graphs are important?!

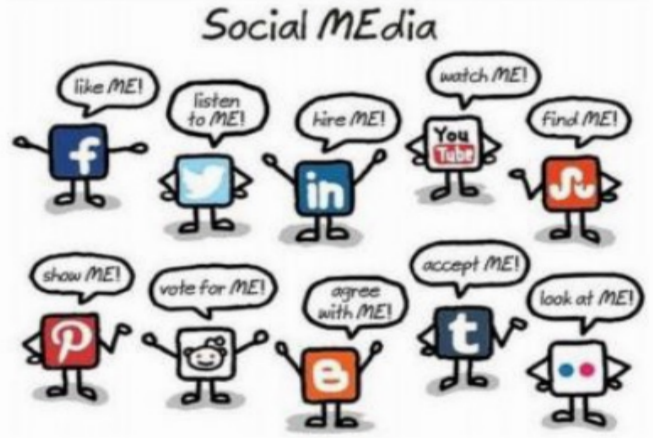


Graphs from social networks

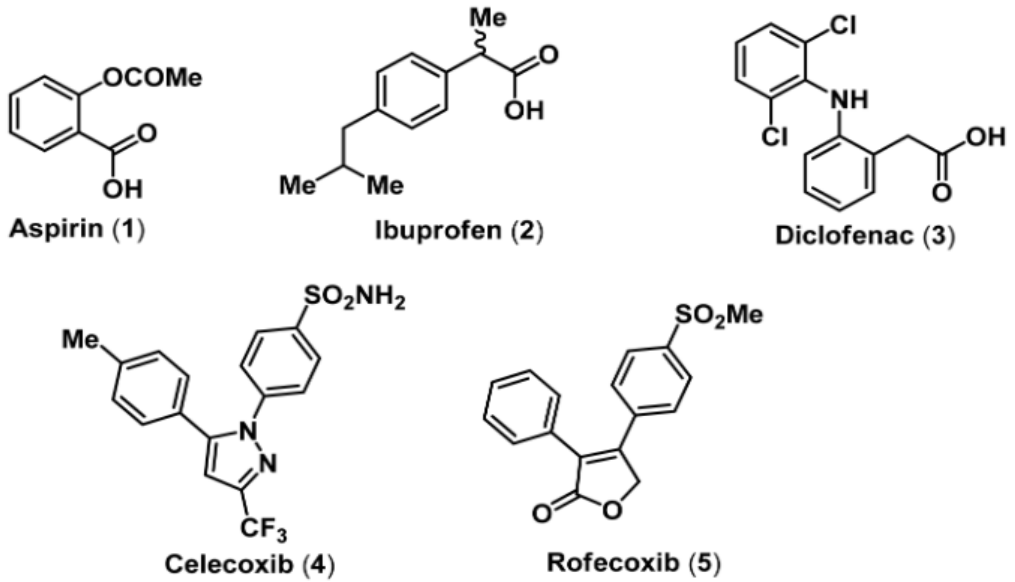
- people and their interactions
  - directed (Twitter) and undirected (Facebook)



- typical ML tasks
  - Node prediction
  - Link(edge) prediction

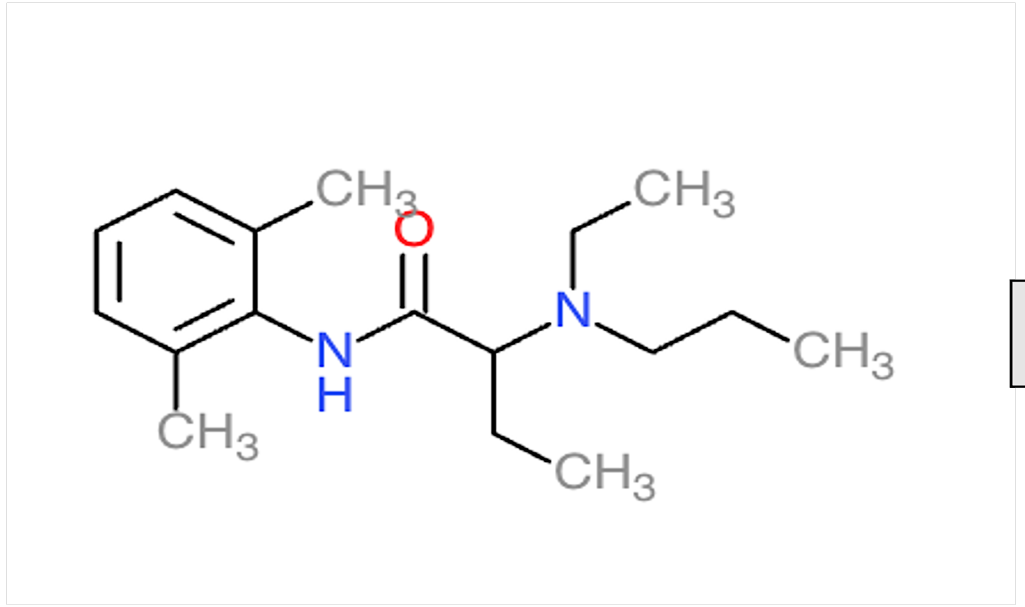


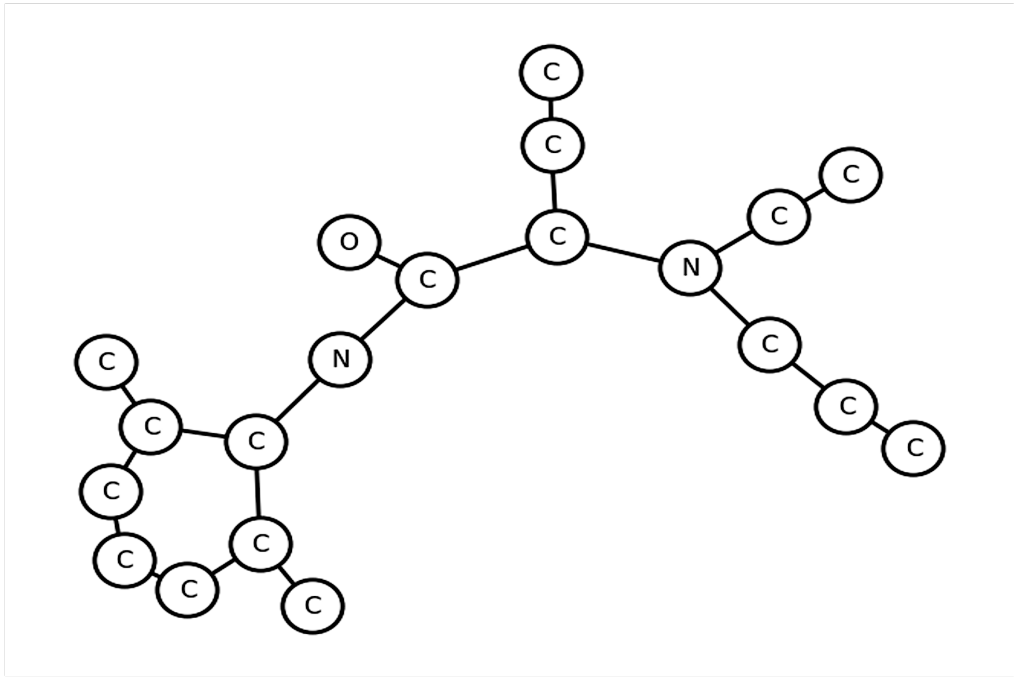
Biomedical Data: Molecular Scale



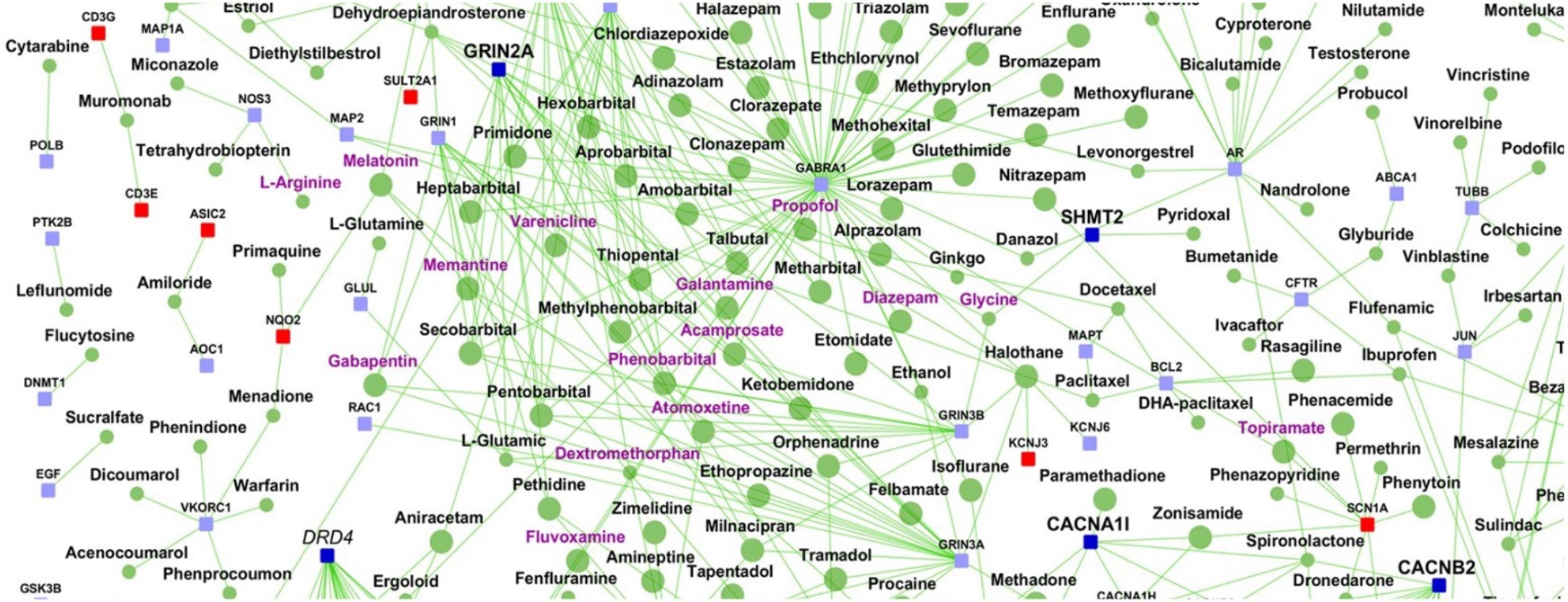
Molecular scale: Small molecule drugs can be represented as graphs relating their **constituent atoms** and **chemical bonding** structure. Complex molecules, such as proteins, can be represented as graphs capturing spatial and structural relationships between their amino acid residues.

Changing Molecule to Graph



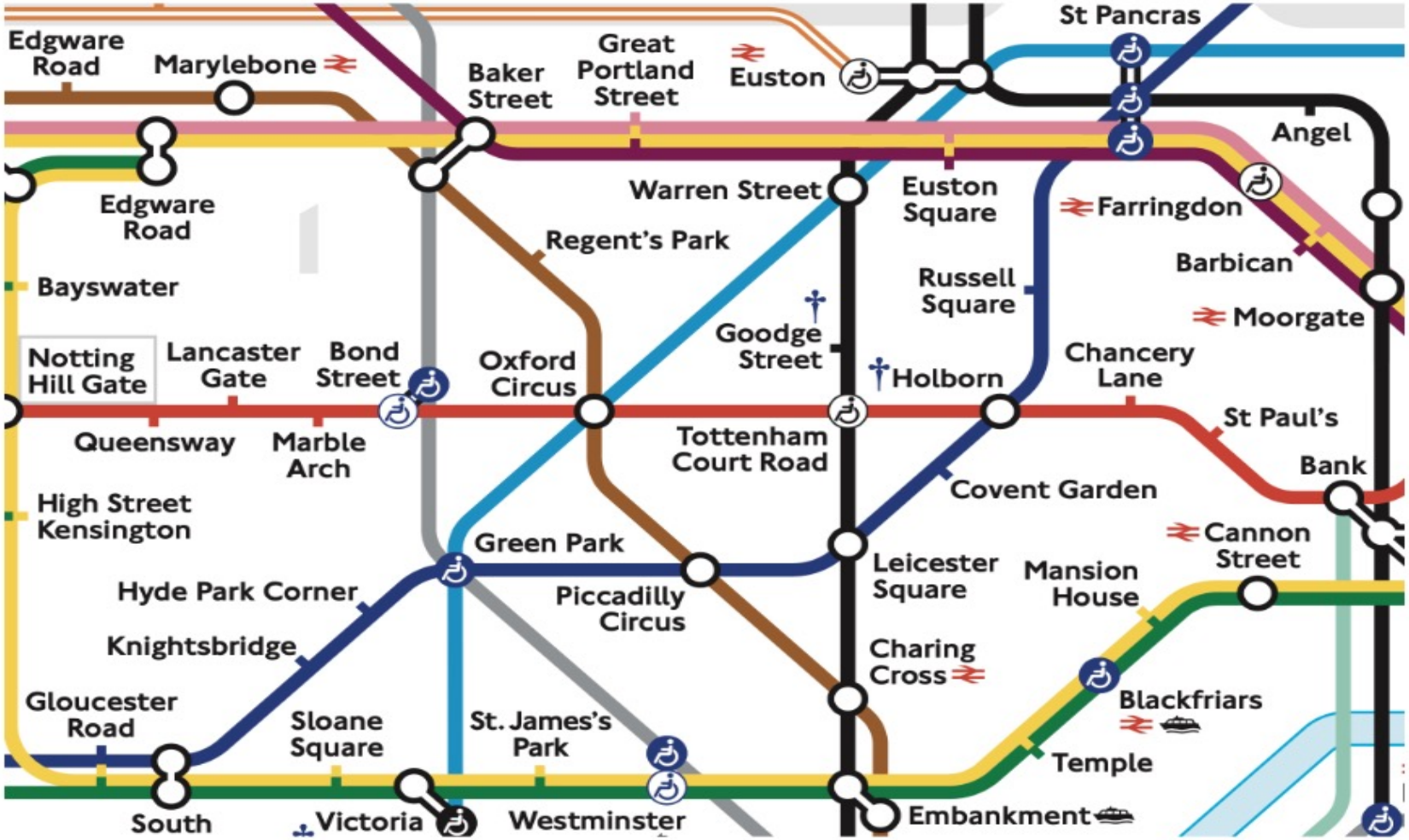


Biomedical Data: Intermediary Scale



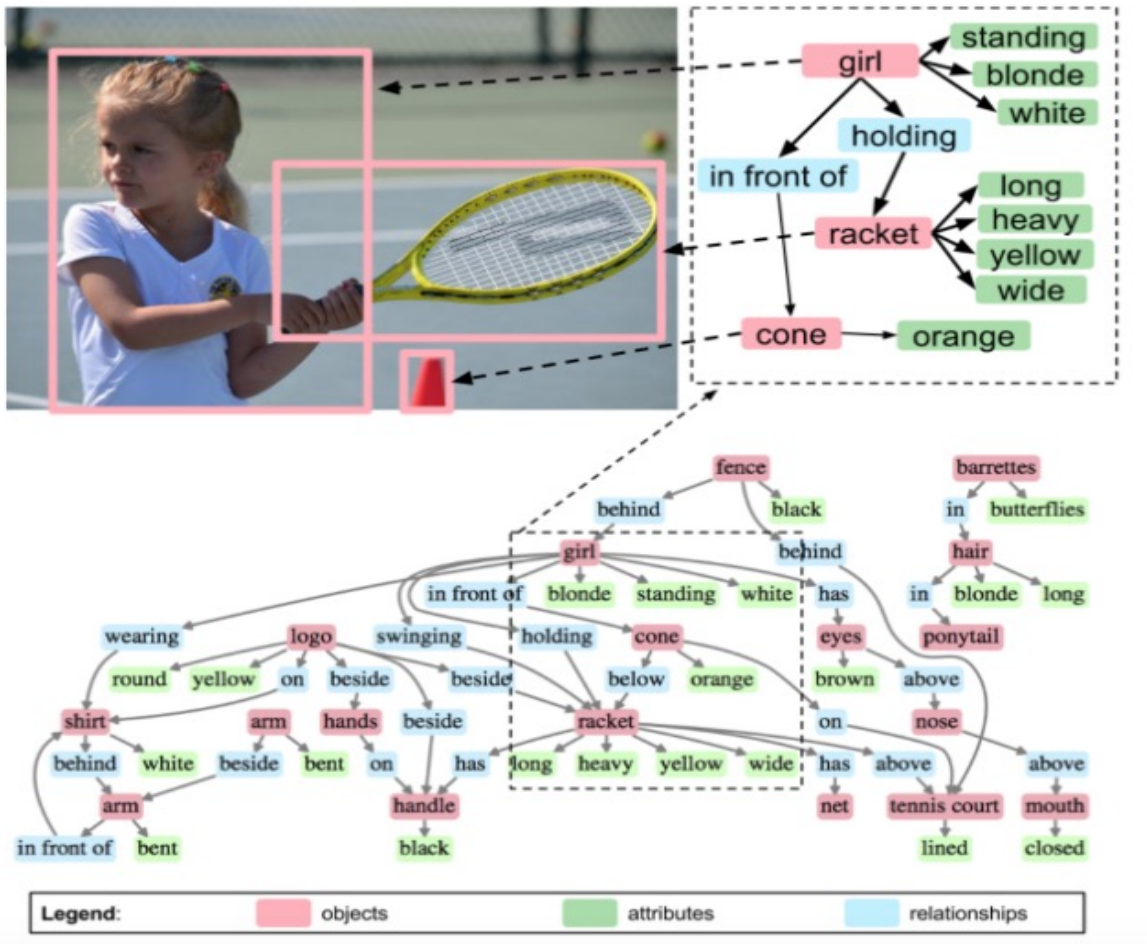
Excerpt from gene–drug interactome: The network shows the drugs that target genes from the schizophrenia interactome. **Drugs** are shown as **round nodes** coloured in green and **genes** as **square nodes** coloured in **dark blue (schizophrenia genes)**, **light blue (known interactors)**, and **red (novel interactors)**.

Road Network

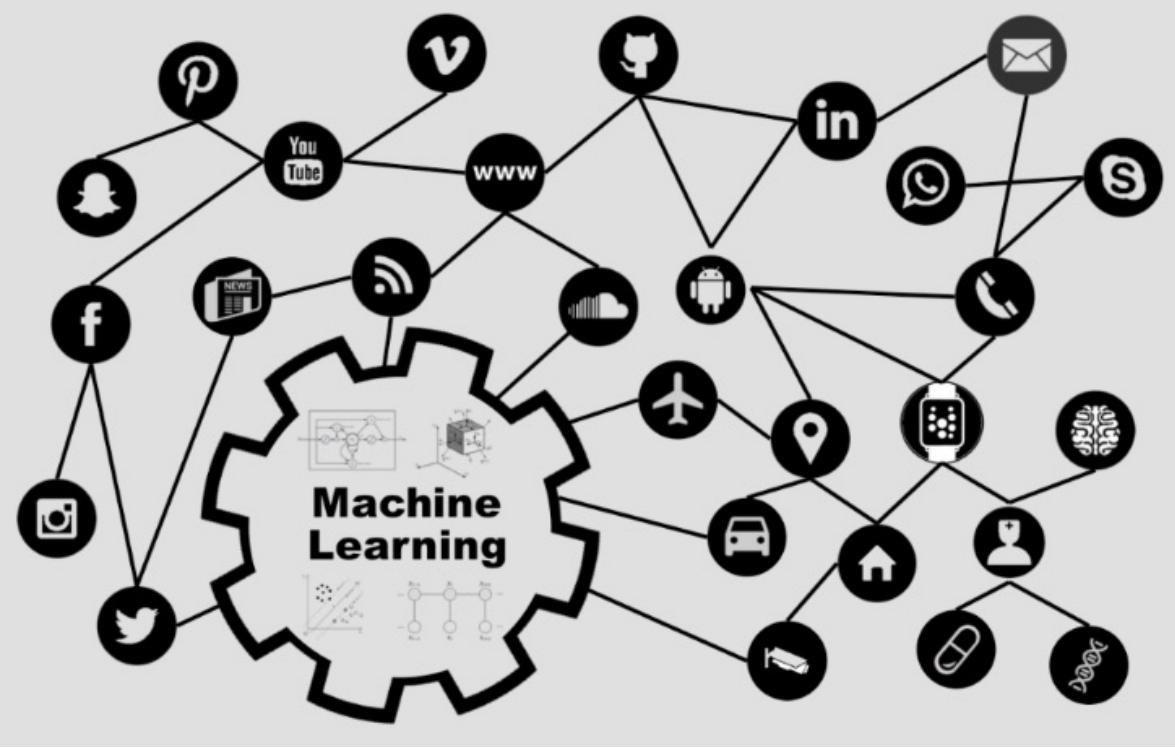


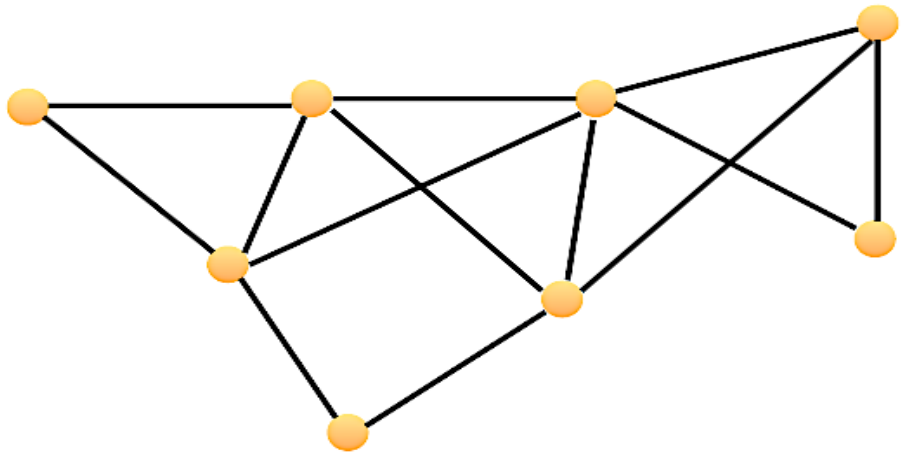
Traffic networks: An excerpt of the London Tube of Zone 1, showing different lines.

Computer Vision: Scene Graphs



What is special about graph machine learning?

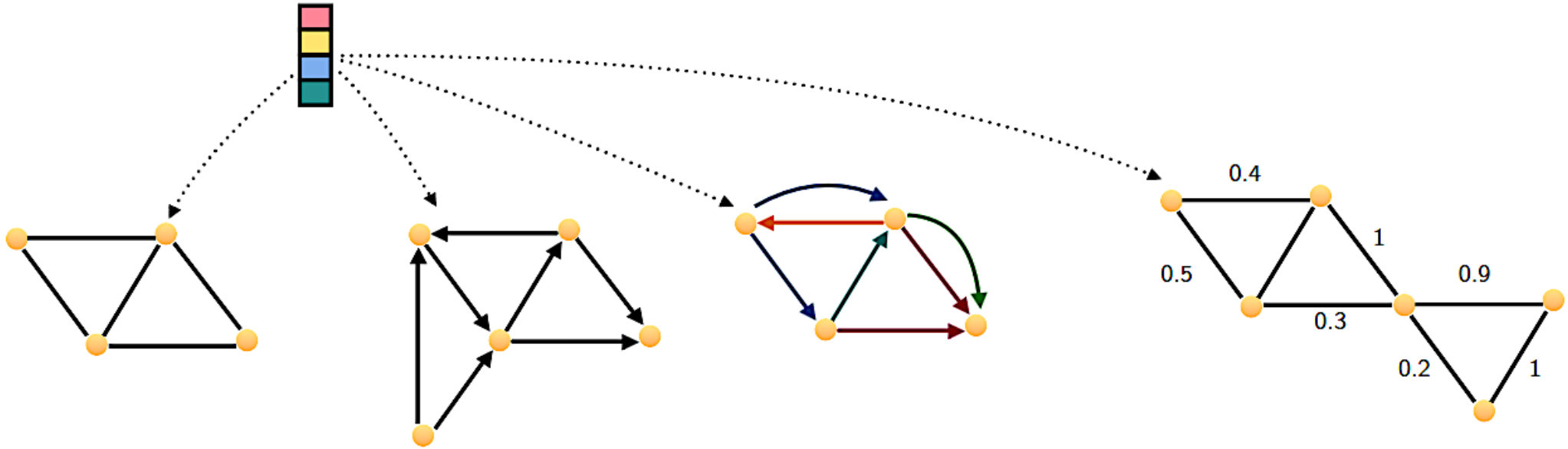




ML algorithms make assumptions about the data, e.g., the data points are **independent and identically distributed (i.i.d.)**:

- independence: no need to model the **dependencies**,
- identical distribution: **generalization** guarantees possible to new/unseen data points.

These assumptions are unrealistic in the context of graphs.



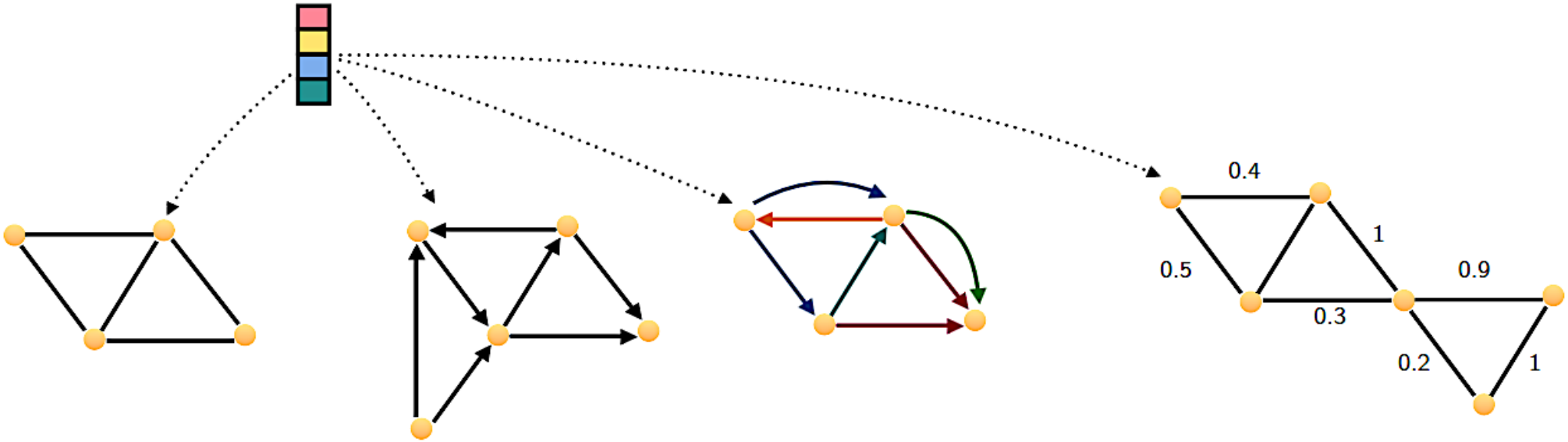
Simple, undirected, unweighted graphs attributed with node features.

Set of vertices/nodes

Set of undirected edges

Node feature matrix, which stores a feature vector for each node u.

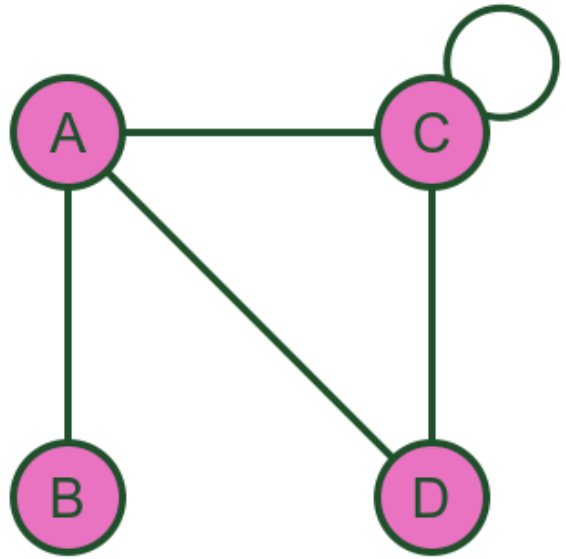
What Kind of Graphs?



Features:

- Node features can be domain-specific **attributes**, or **node degrees**, or simply **one-hot encodings**.
- We can extend the class of (attributed) graphs to include edge features .

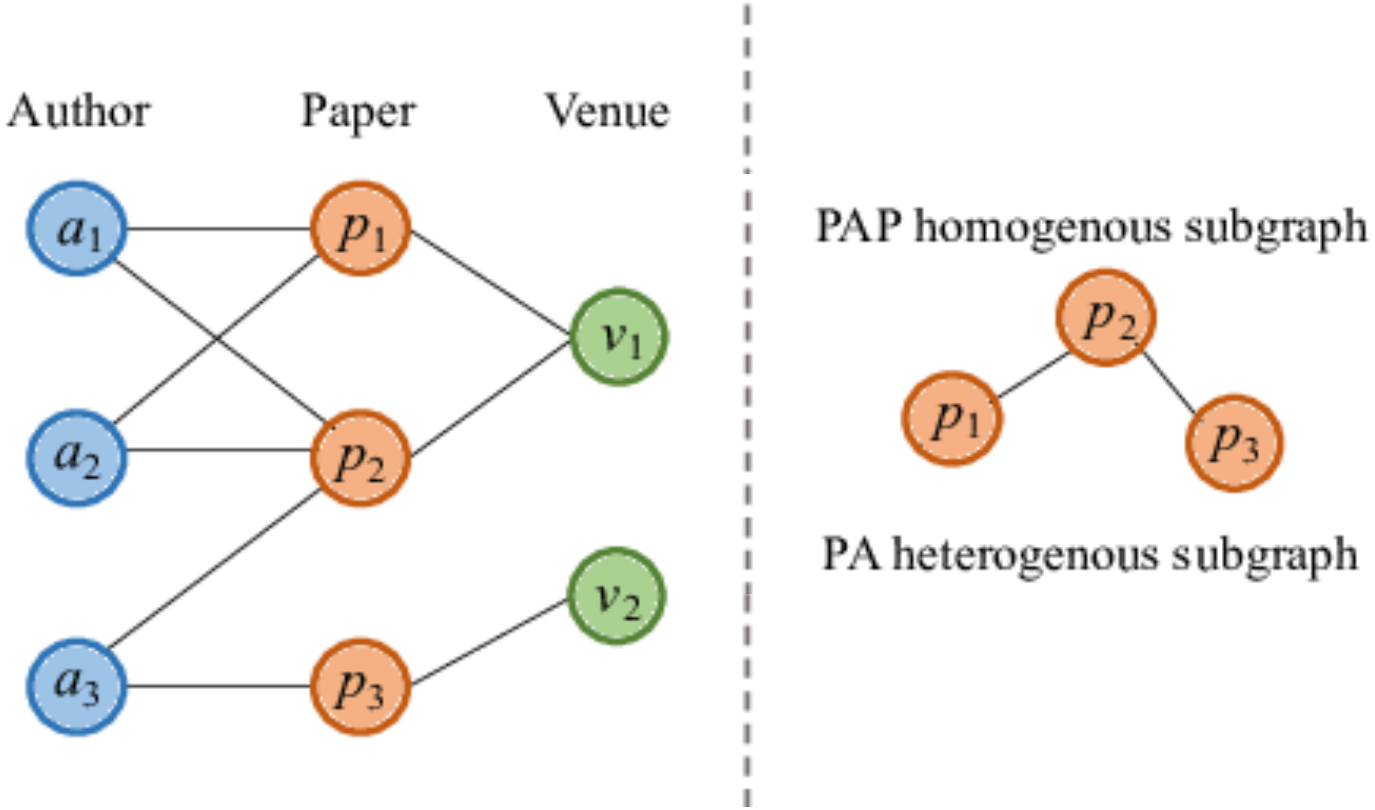
What Kind of Graphs?



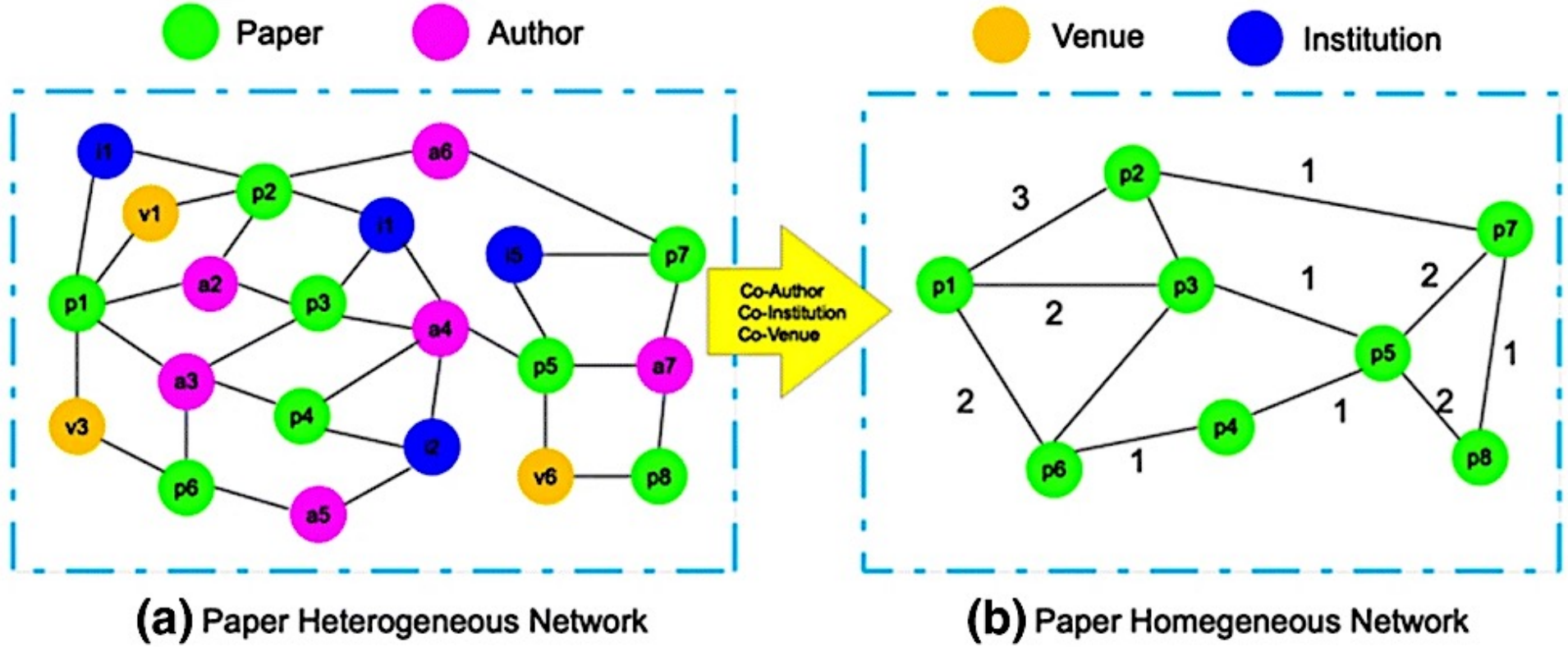
Simple graph with loop

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

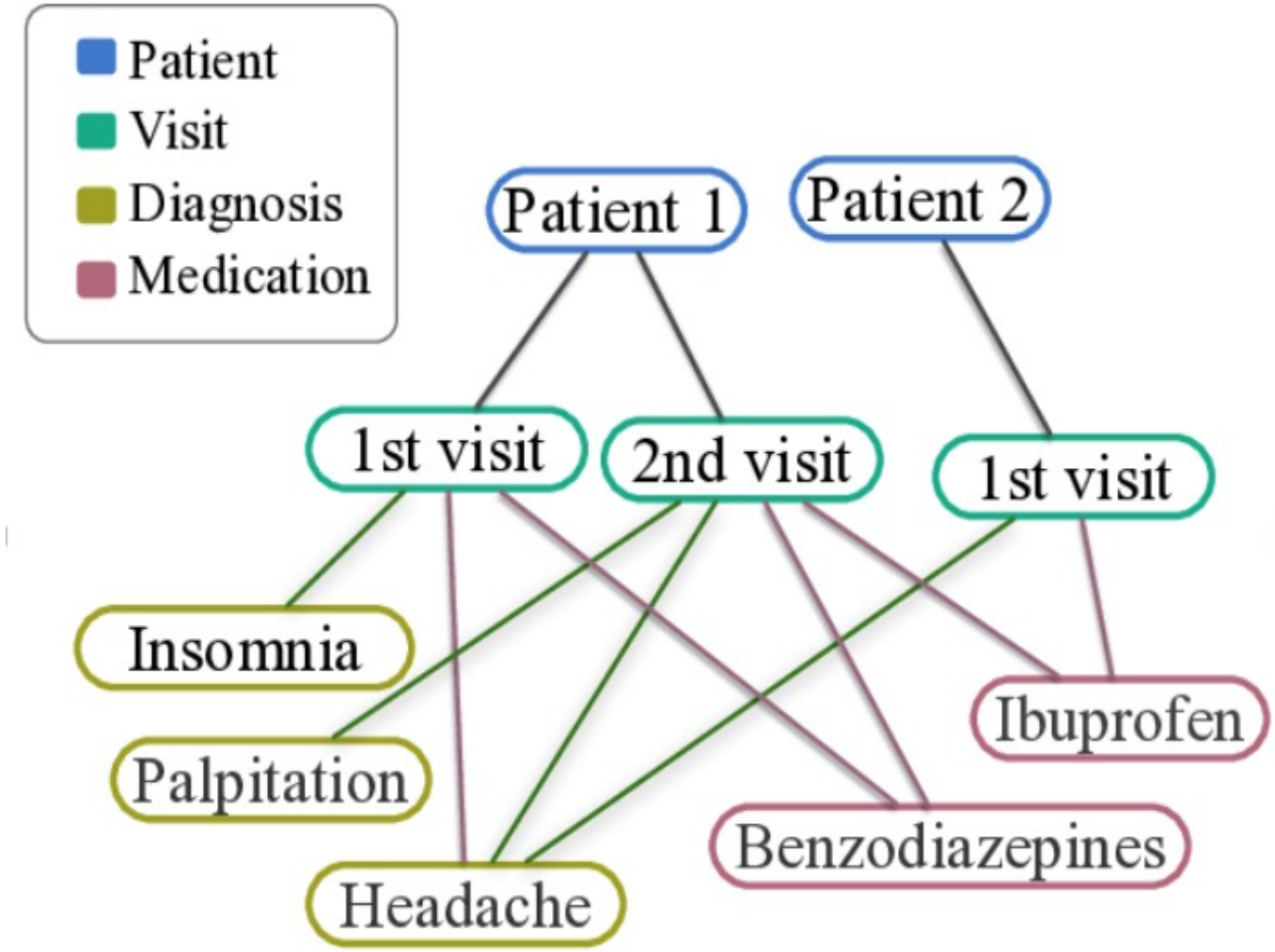
Representations of Graphs



heterogeneous vs homogeneous graphs



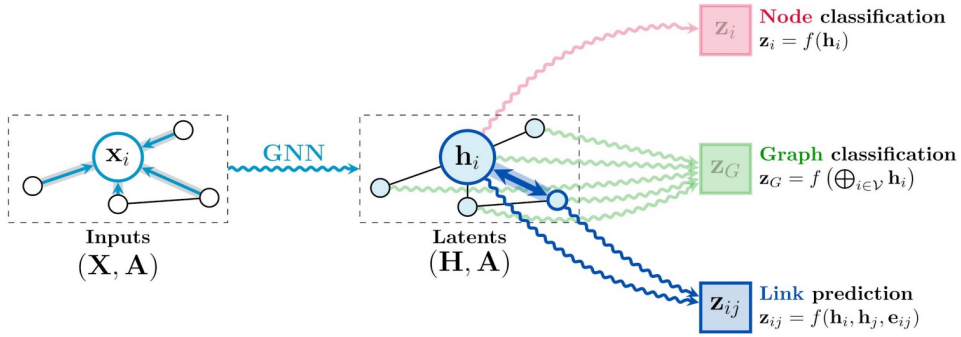
heterogeneous to homogeneous graphs



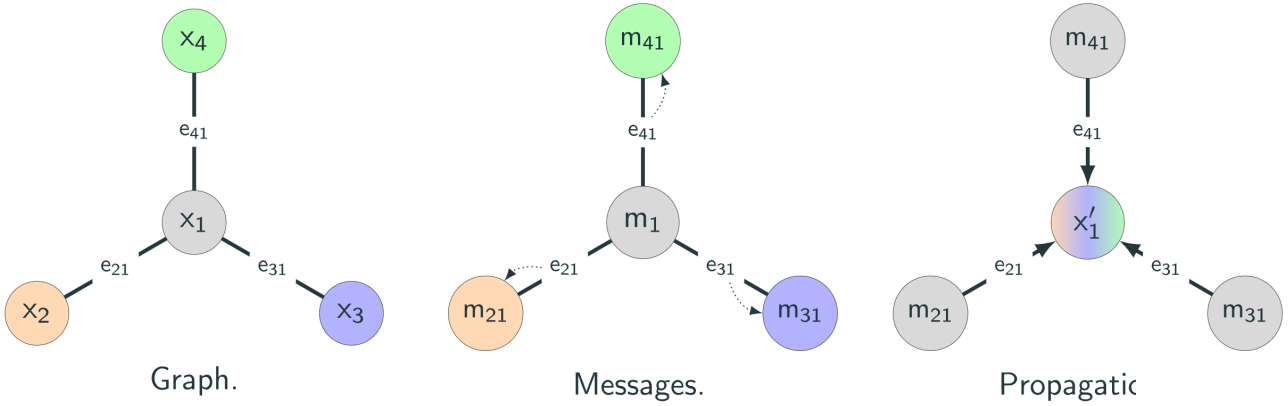
Example of a Heterogeneous graph

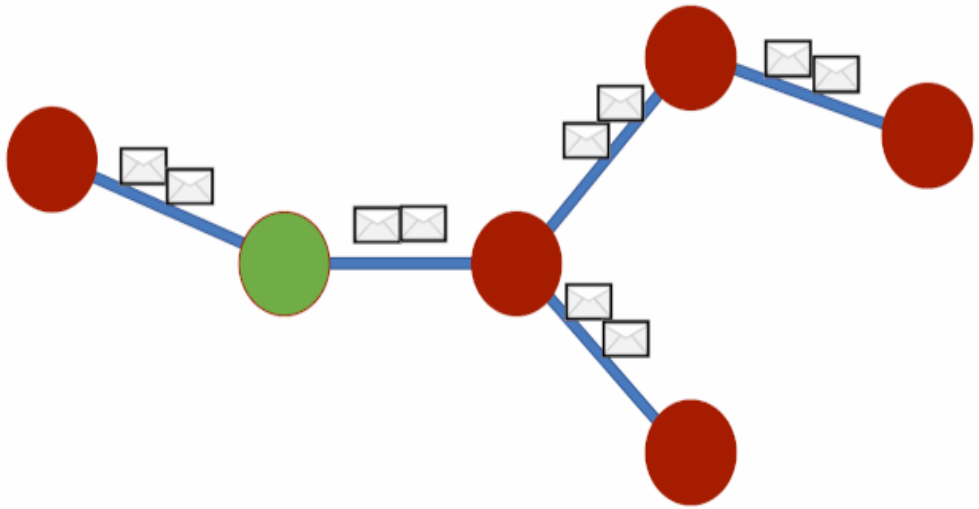
- A **graph neural network** is a neural model that we can apply directly to graphs without **prior knowledge** of every component within the graph. GNN provides a convenient way for node level, edge level and graph level prediction tasks.

- In GNNs, neighbours and connections define nodes. If we remove the neighbours and connections around a node, then the node will lose all its information. .

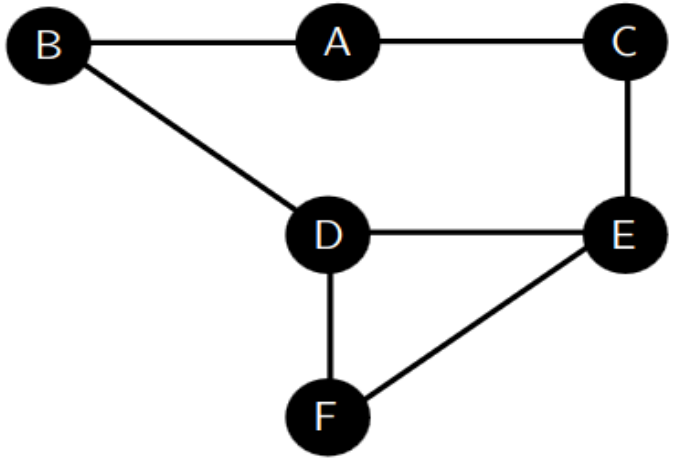


What Is a Graph Neural Network (GNN)?

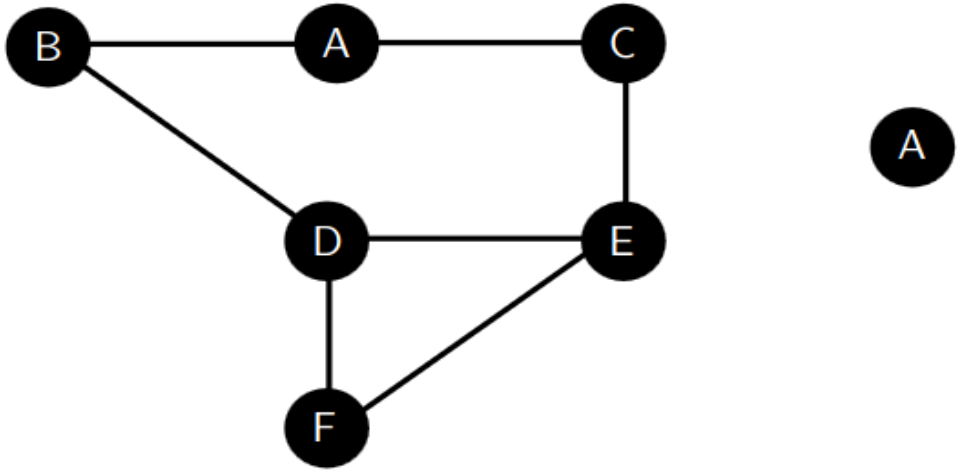




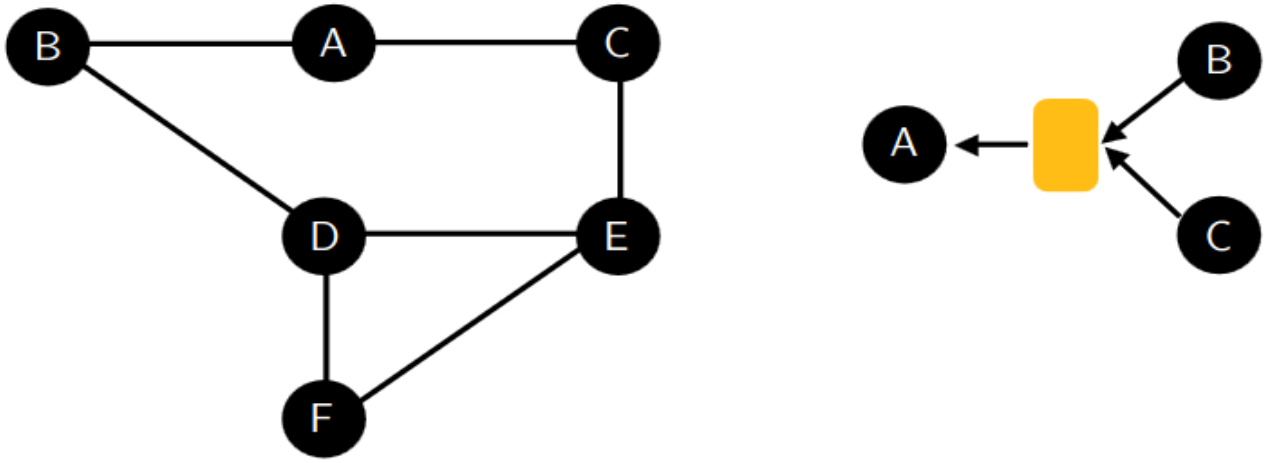
Message Passing in GNNs



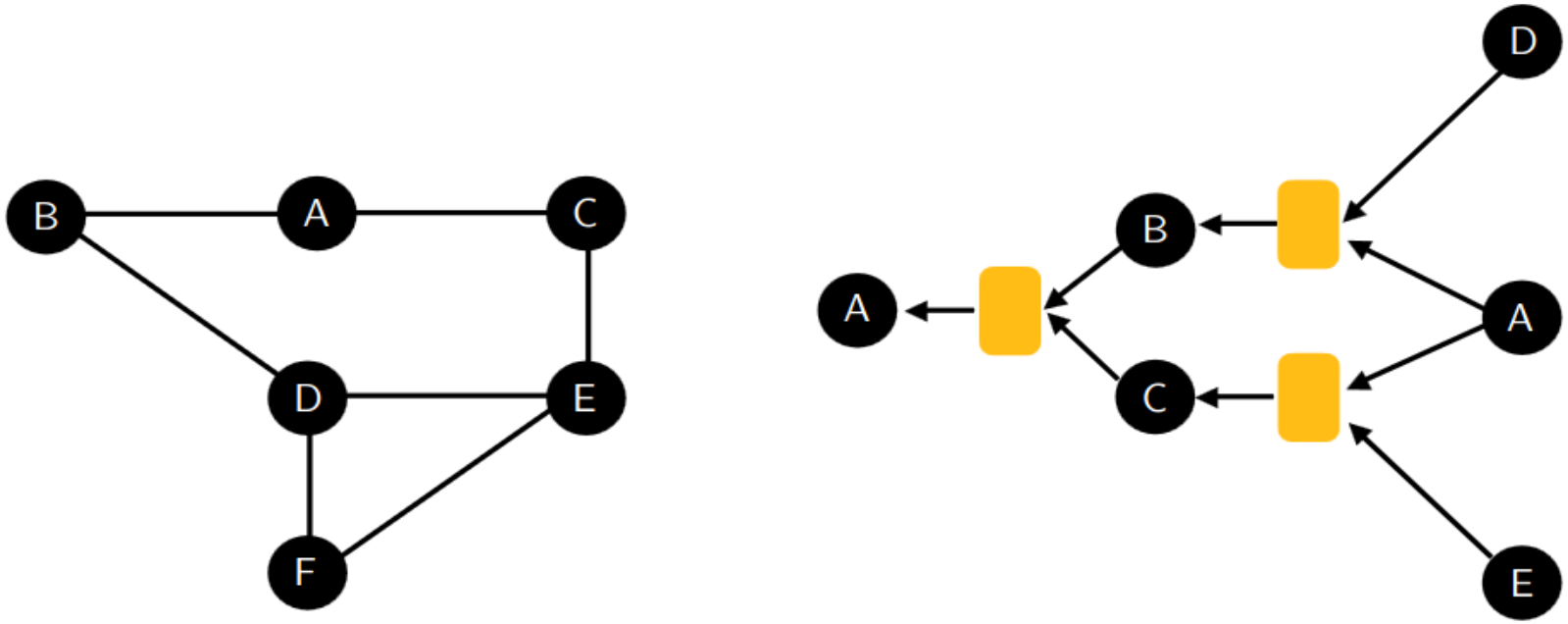
A graph (left) and an illustration of message passing on this graph with respect to the target node A for 3 iterations (right). Directed arrows depict the **messages**, and yellow boxes denote **aggregation**. At least 3 iterations are needed to get information from **all** nodes, i.e., F will not pass any messages to A with .



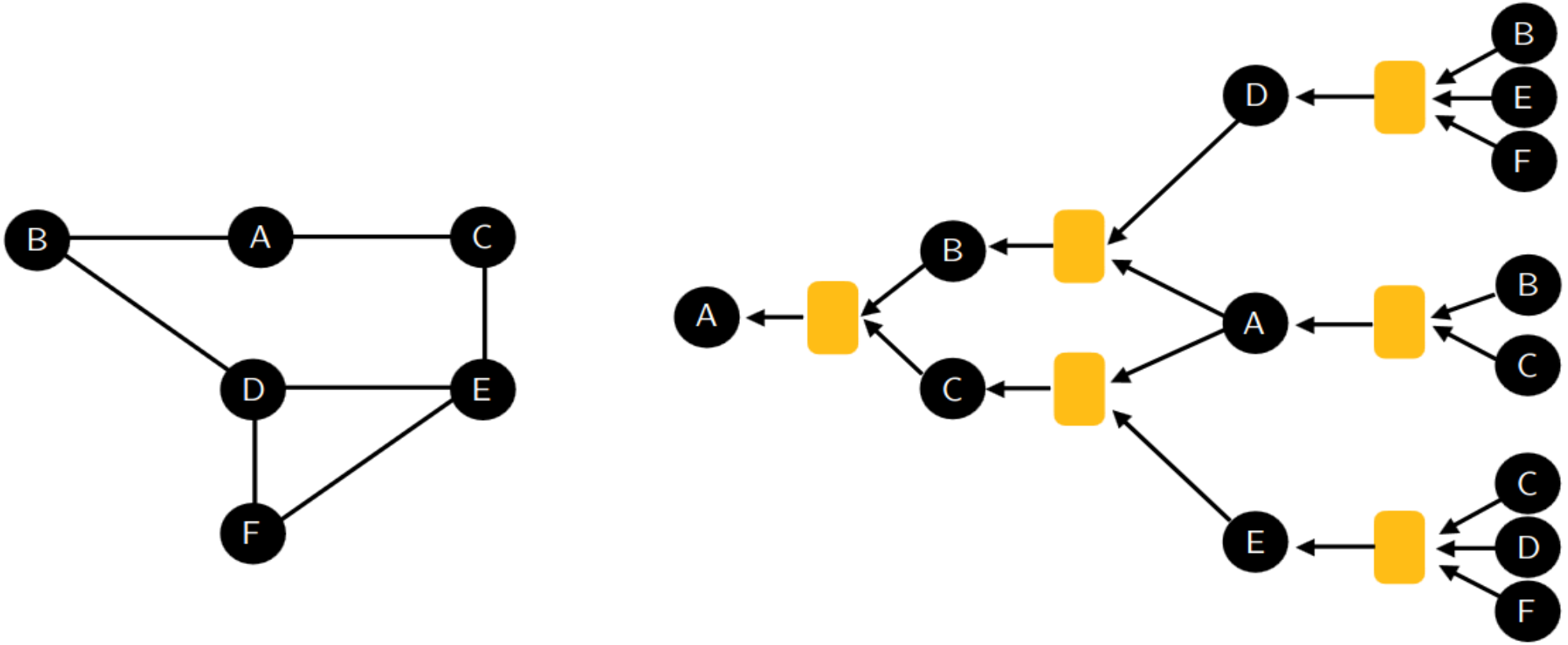
A graph (left) and an illustration of message passing on this graph with respect to the target node A for 3 iterations (right). Directed arrows depict the **messages**, and yellow boxes denote **aggregation**. At least 3 iterations are needed to get information from **all** nodes, i.e., F will not pass any messages to A with .



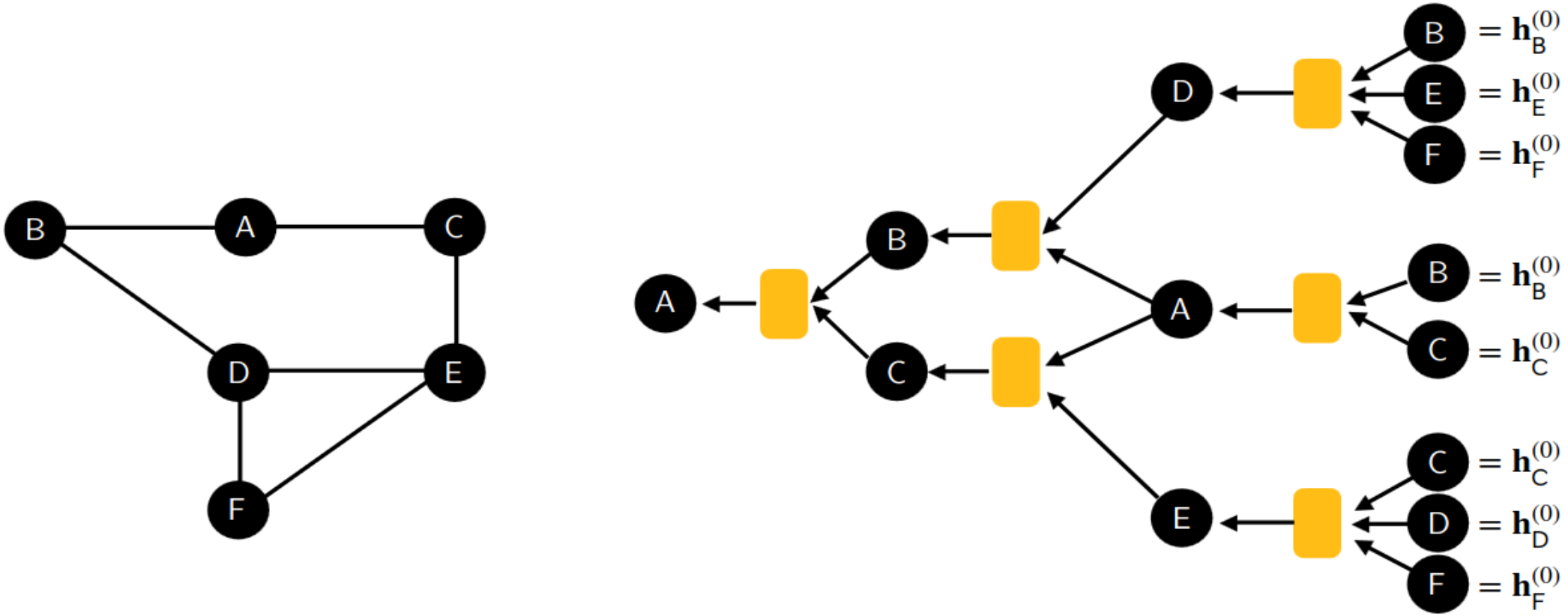
A graph (left) and an illustration of message passing on this graph with respect to the target node A for 3 iterations (right). Directed arrows depict the **messages**, and yellow boxes denote **aggregation**. At least 3 iterations are needed to get information from **all** nodes, i.e., F will not pass any messages to A with .



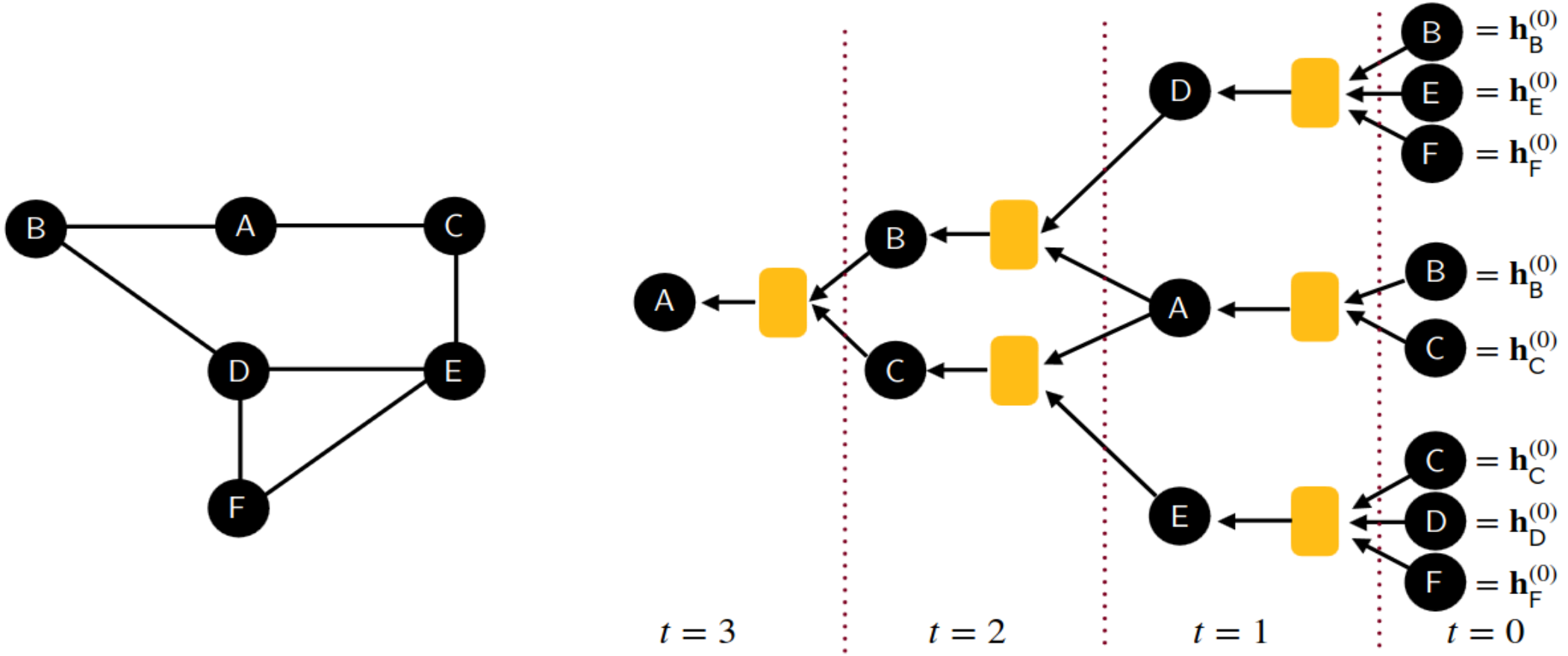
A graph (left) and an illustration of message passing on this graph with respect to the target node A for 3 iterations (right). Directed arrows depict the **messages**, and yellow boxes denote **aggregation**. At least 3 iterations are needed to get information from **all** nodes, i.e., F will not pass any messages to A with .



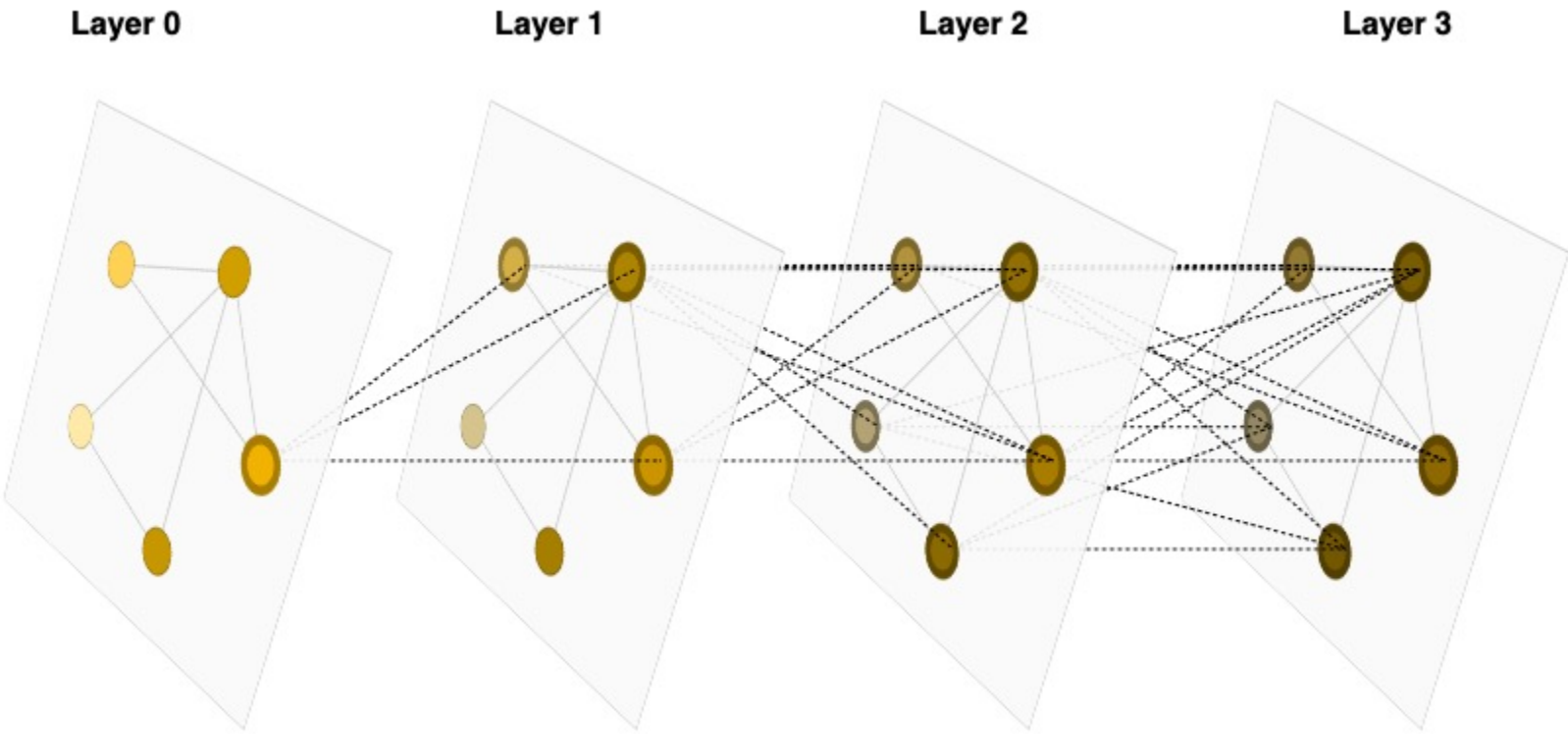
A graph (left) and an illustration of message passing on this graph with respect to the target node A for 3 iterations (right). Directed arrows depict the **messages**, and yellow boxes denote **aggregation**. At least 3 iterations are needed to get information from **all** nodes, i.e., F will not pass any messages to A with .



A graph (left) and an illustration of message passing on this graph with respect to the target node A for 3 iterations (right). Directed arrows depict the **messages**, and yellow boxes denote **aggregation**. At least 3 iterations are needed to get information from **all** nodes, i.e., F will not pass any messages to A with .



A graph (left) and an illustration of message passing on this graph with respect to the target node A for 3 iterations (right). Directed arrows depict the **messages**, and yellow boxes denote **aggregation**. At least 3 iterations are needed to get information from **all** nodes, i.e., F will not pass any messages to A with .



Message Passing Neural Networks

Given a graph $G = (V, E, \mathbf{X})$, an MPNN iteratively computes $\mathbf{h}_u^{(t)}$ for every node $u \in V$:

$$\mathbf{h}_u^{(0)} = \mathbf{x}_u, \qquad \text{initialize}$$

$$\mathbf{m}_u^{(t)} = \psi^{(t)}\left(\mathbf{h}_u^{(t-1)}, \left\{\left\{\mathbf{h}_v^{(t-1)} \mid v \in N(u)\right\}\right\}\right), \qquad \text{aggregate}$$

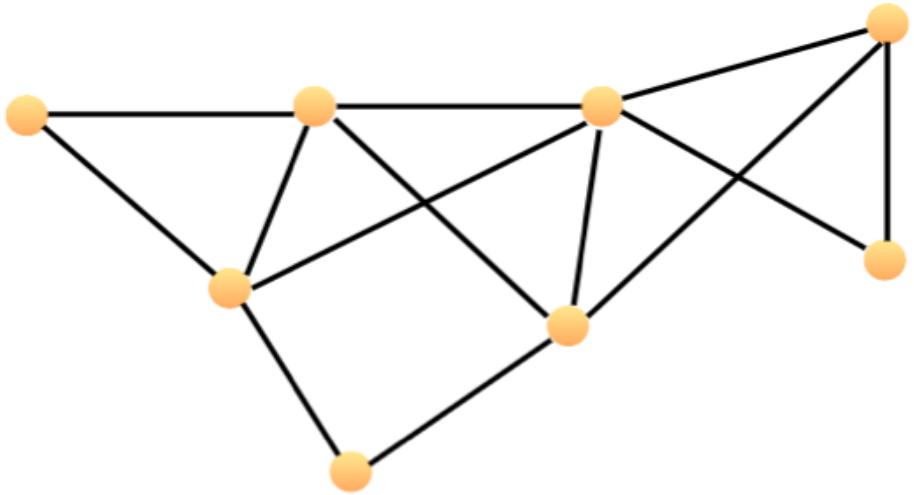
$$\mathbf{h}_u^{(t)} = \phi^{(t)}\left(\mathbf{h}_u^{(t-1)}, \mathbf{m}_u^{(t)}\right), \qquad \text{update/combine}$$

mean, sum, max, ...

message

non-linearity

$\phi^{(t)}$ and $\psi^{(t)}$ can be any differentiable function!



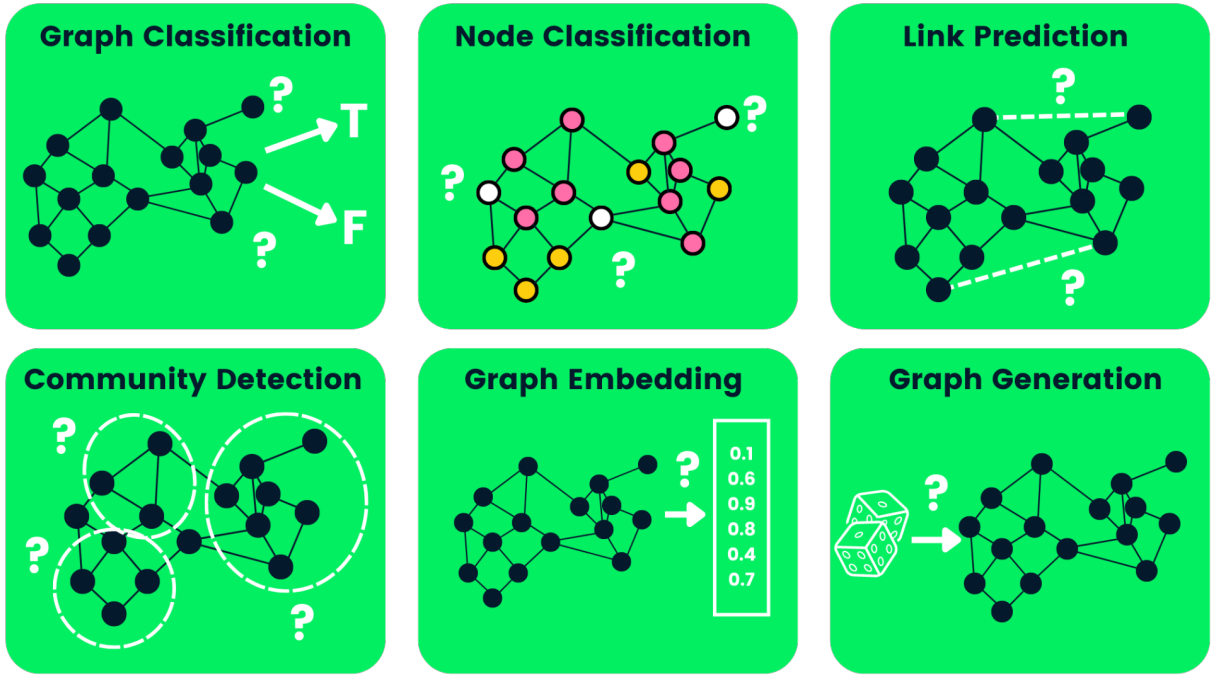
You may encounter variations, where a message computation function msg is defined w.r.t the source node:

$$\mathbf{h}_u^{(t)} = \phi^{(t)}\left(\mathbf{h}_u^{(t-1)}, \psi^{(t)}\left(\{\{\text{msg}(\mathbf{h}_u^{(t-1)}, \mathbf{h}_v^{(t-1)}) \mid v \in N(u)\}\}\right)\right),$$

Remark: The function msg typically depends on the neighborhood - hard to decouple msg from $\psi^{(t)}$. Following a common convention, we view the message computation as part of aggregation.

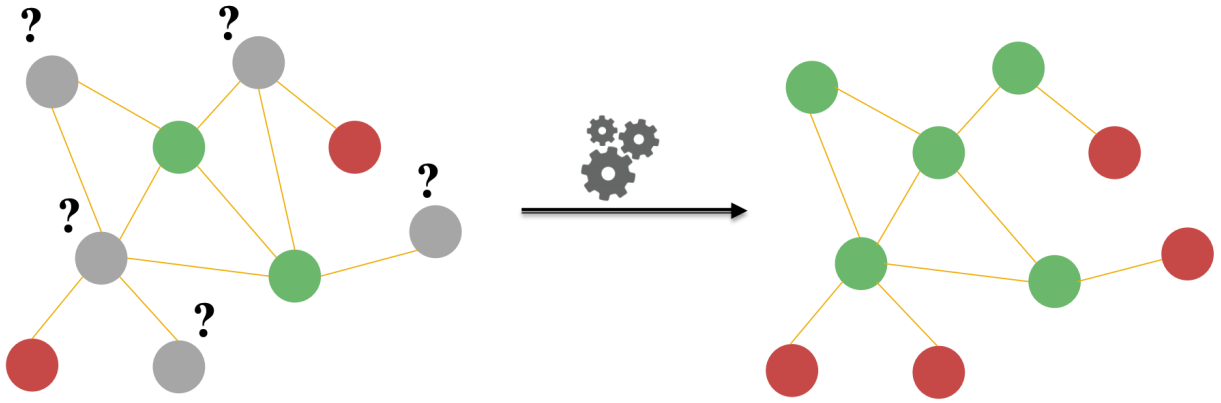
Message Passing Neural Networks

GNN Tasks


Graph Classification
?
T
F
?
Node Classification
?
?
?
Link Prediction
?
?
Community Detection
?
?
?
Graph Embedding
?
0.1
0.6
0.9
0.8
0.4
0.7
Graph Generation
?

Input: a graph where a subset of the nodes are labelled with a class.

Task: predict the labels of the remaining nodes, i.e., test nodes in the graph



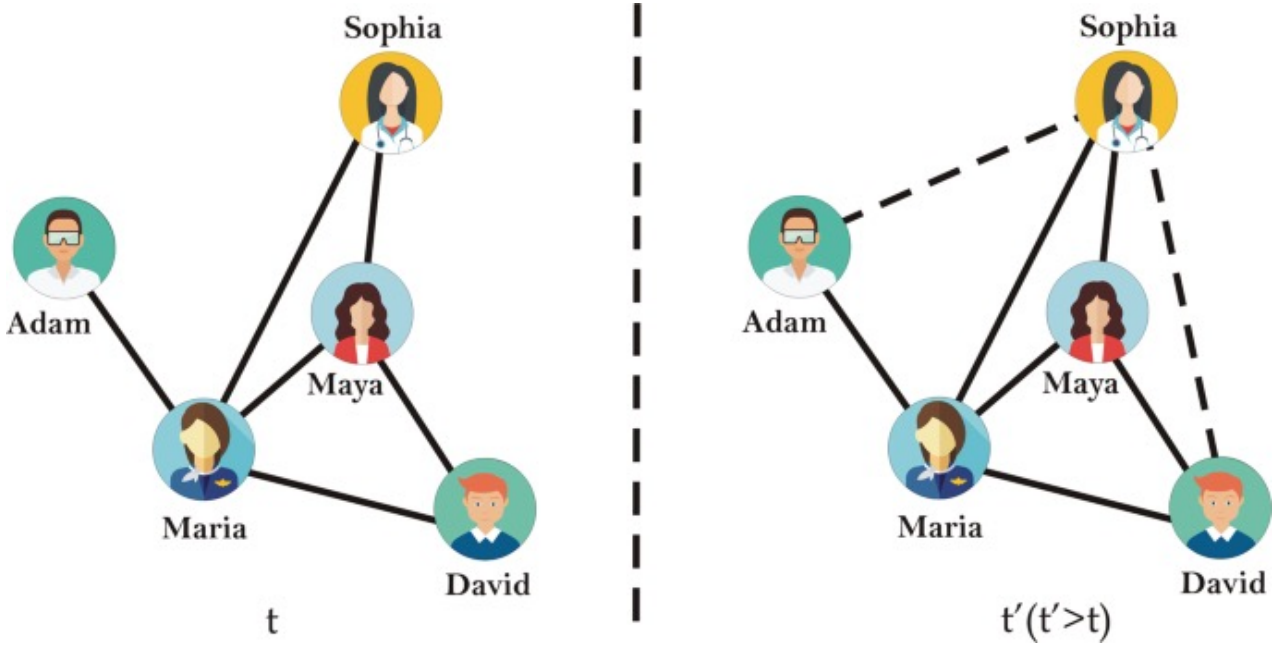
Example:

- Citeseer is a citation network, where nodes represent **papers**, and edges denote **citation links**, and a subset of the nodes are labelled with a **paper category** (e.g., AI, ML).
- Predict the category (or, categories) of the remaining papers.

Node Level Tasks- Node Classification

Input: a graph

Task: predict new links

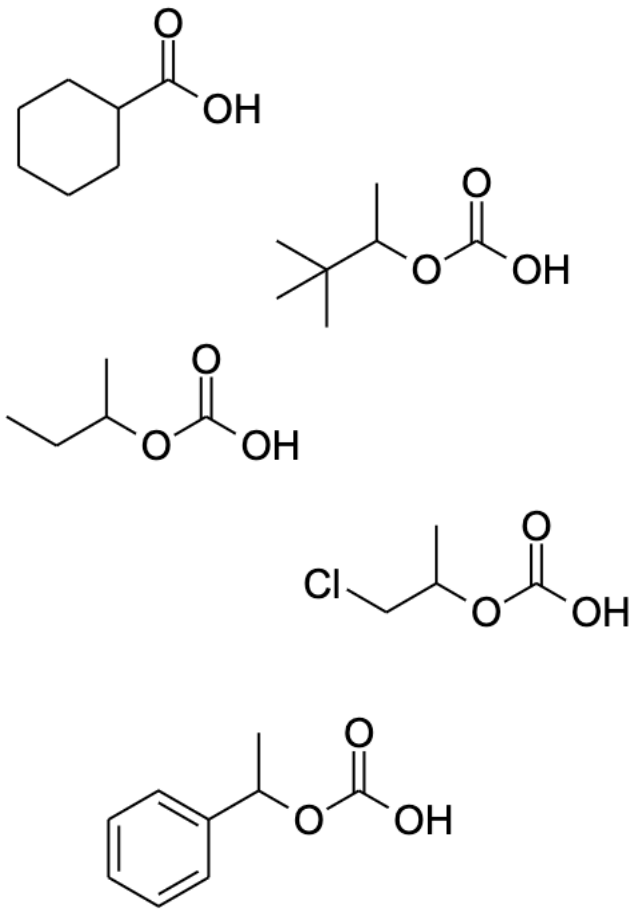


Example:

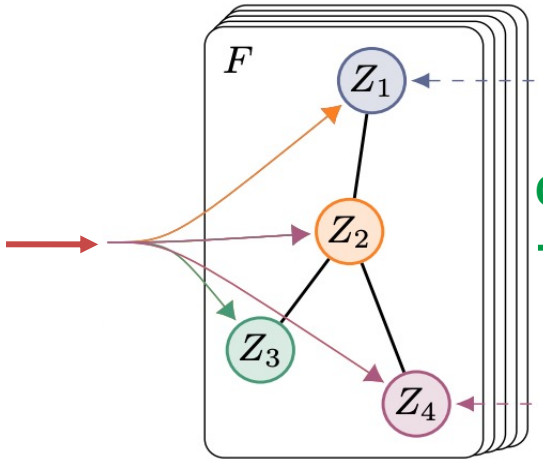
- **OGBL-DDI** is a **drug-drug interaction network**: Nodes represent drugs and edges interactions between drugs.
- Predict drug-drug interactions: rank true drug interactions higher than non-interacting drug pairs, using, e.g., a pairwise decoder.

Edge Level Tasks-Link Prediction

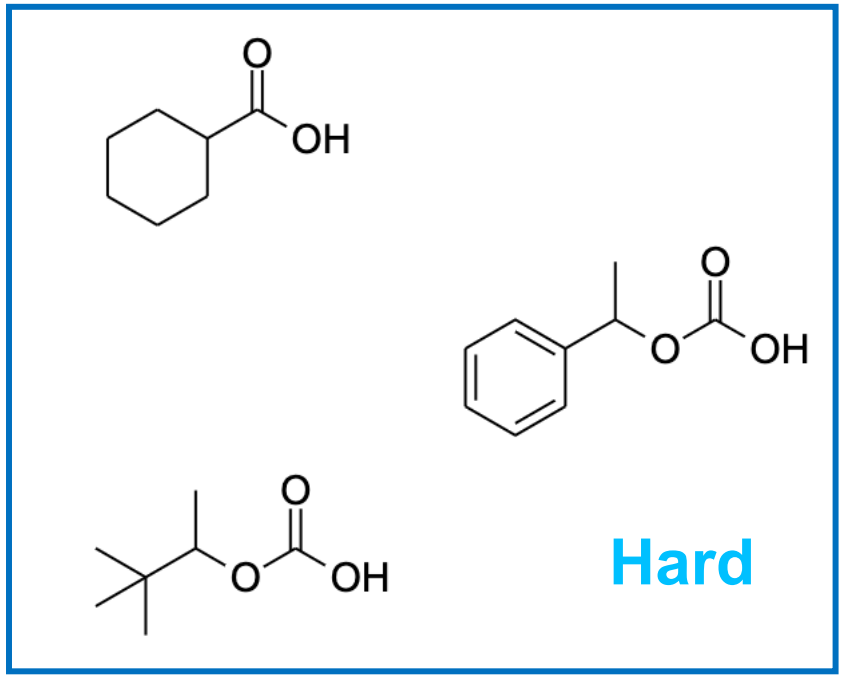
homopolymers

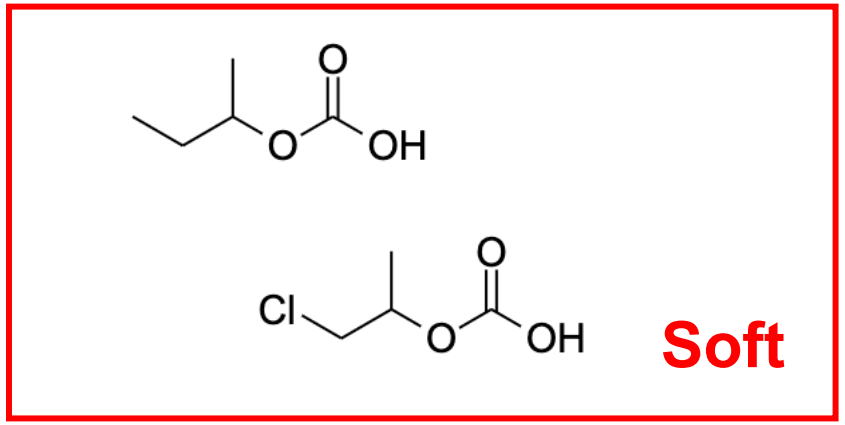


GNN



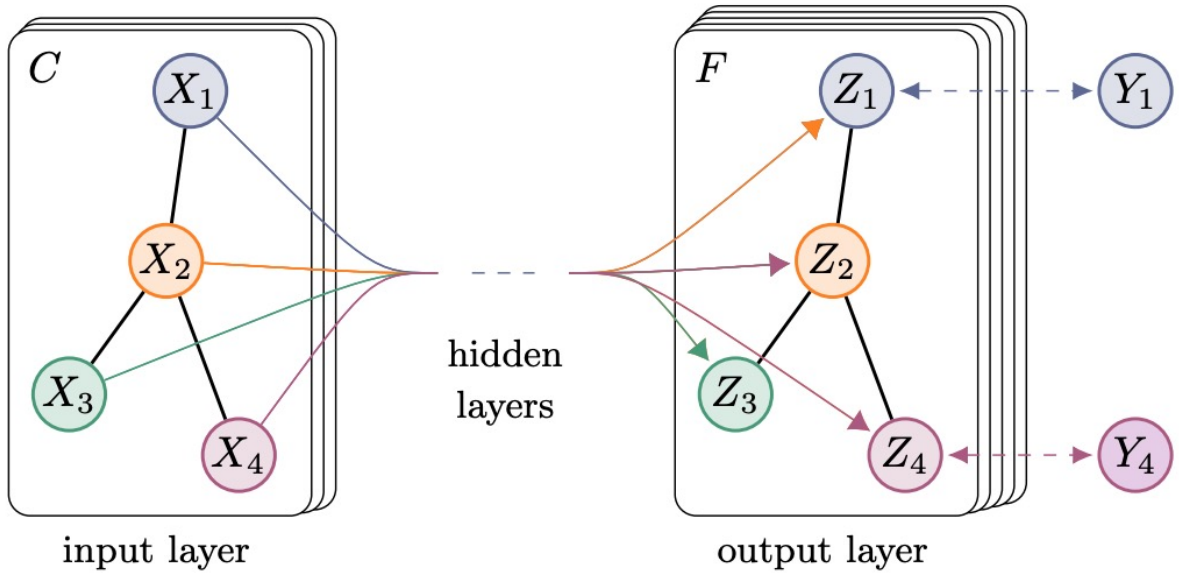
Classification

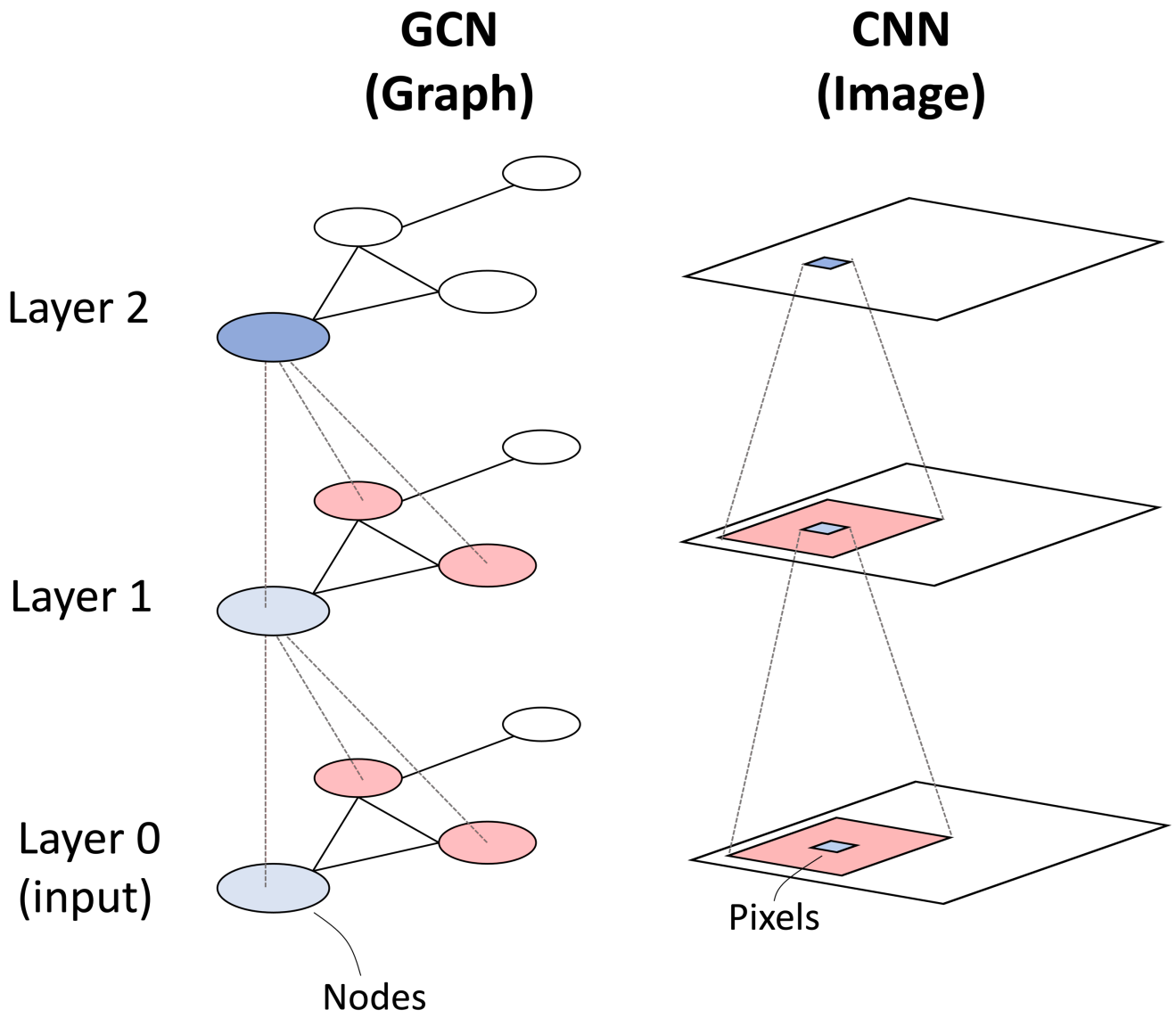




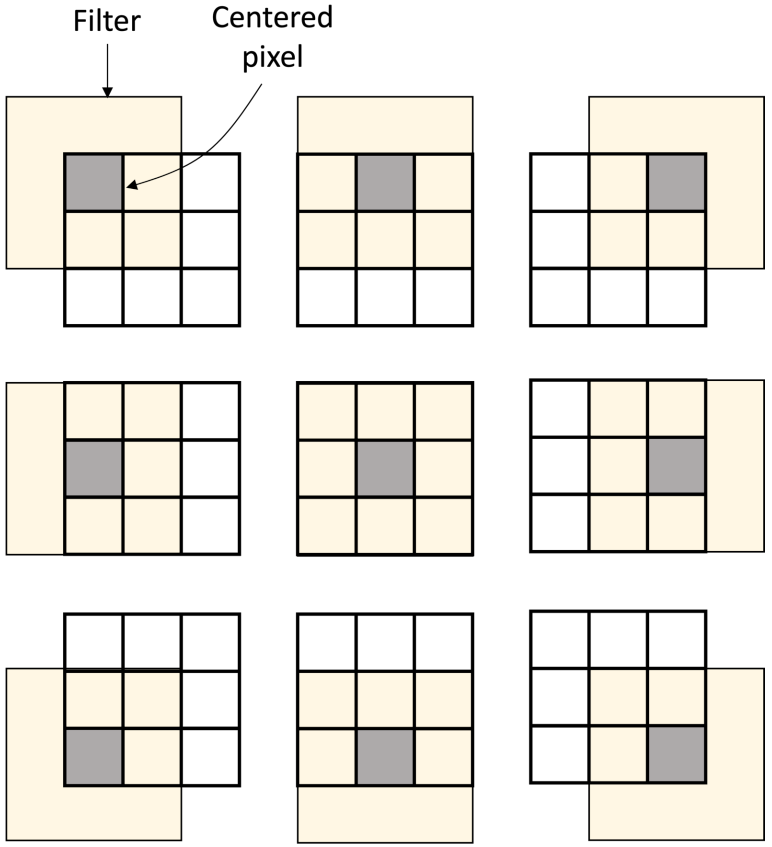
Graph Level Tasks- Graph Classification

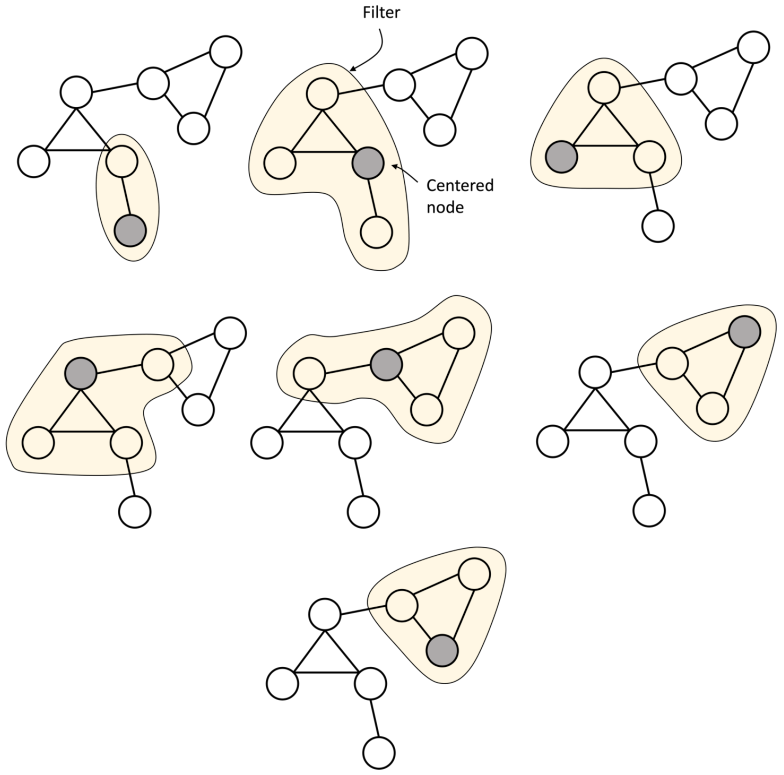
Graph Convolutional Network



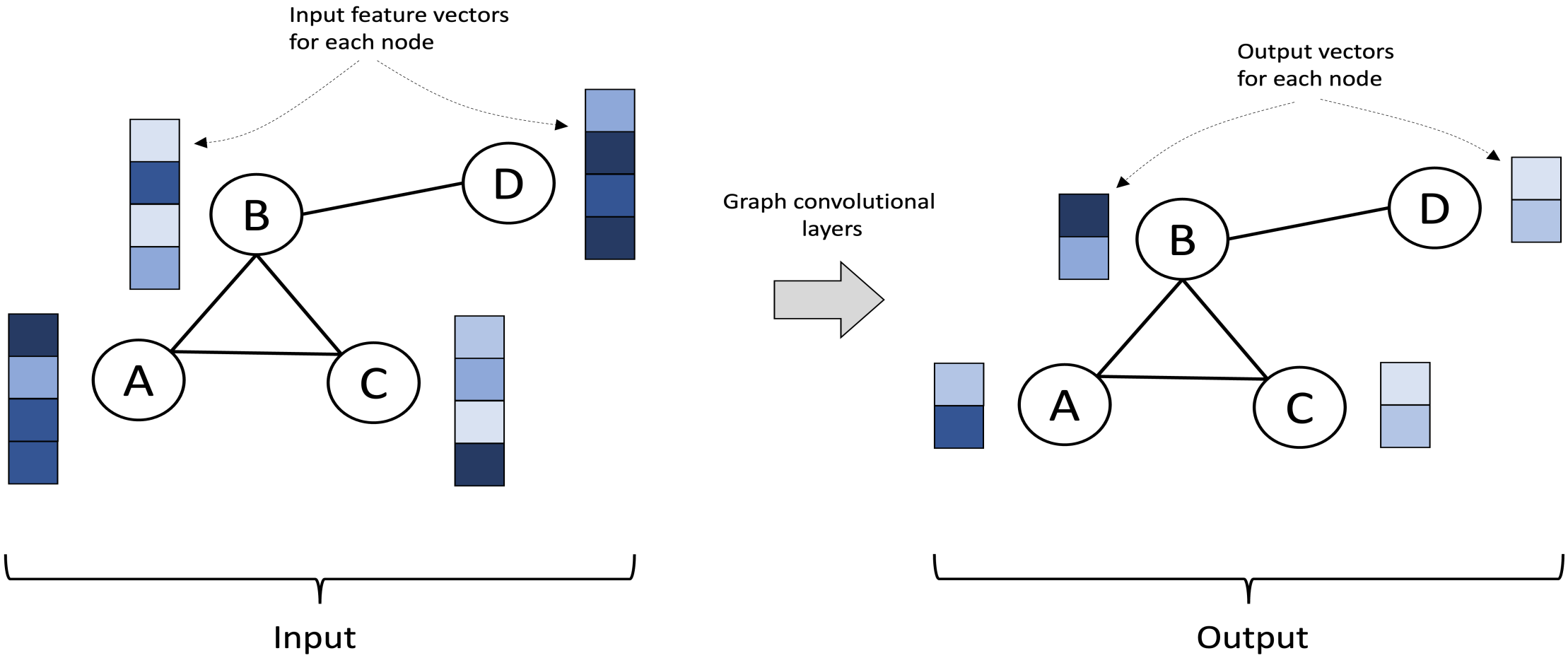


The graph convolutional layer

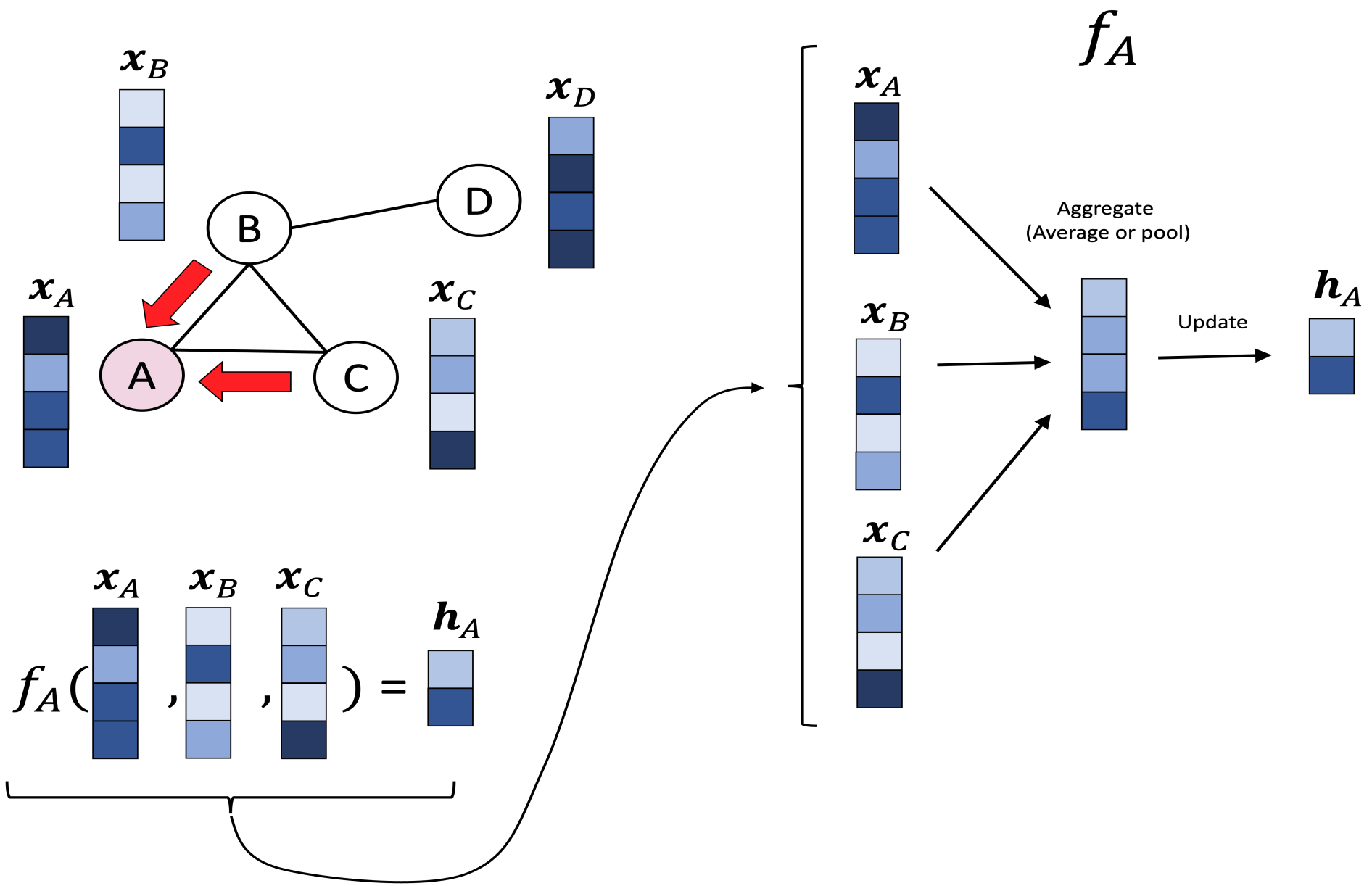




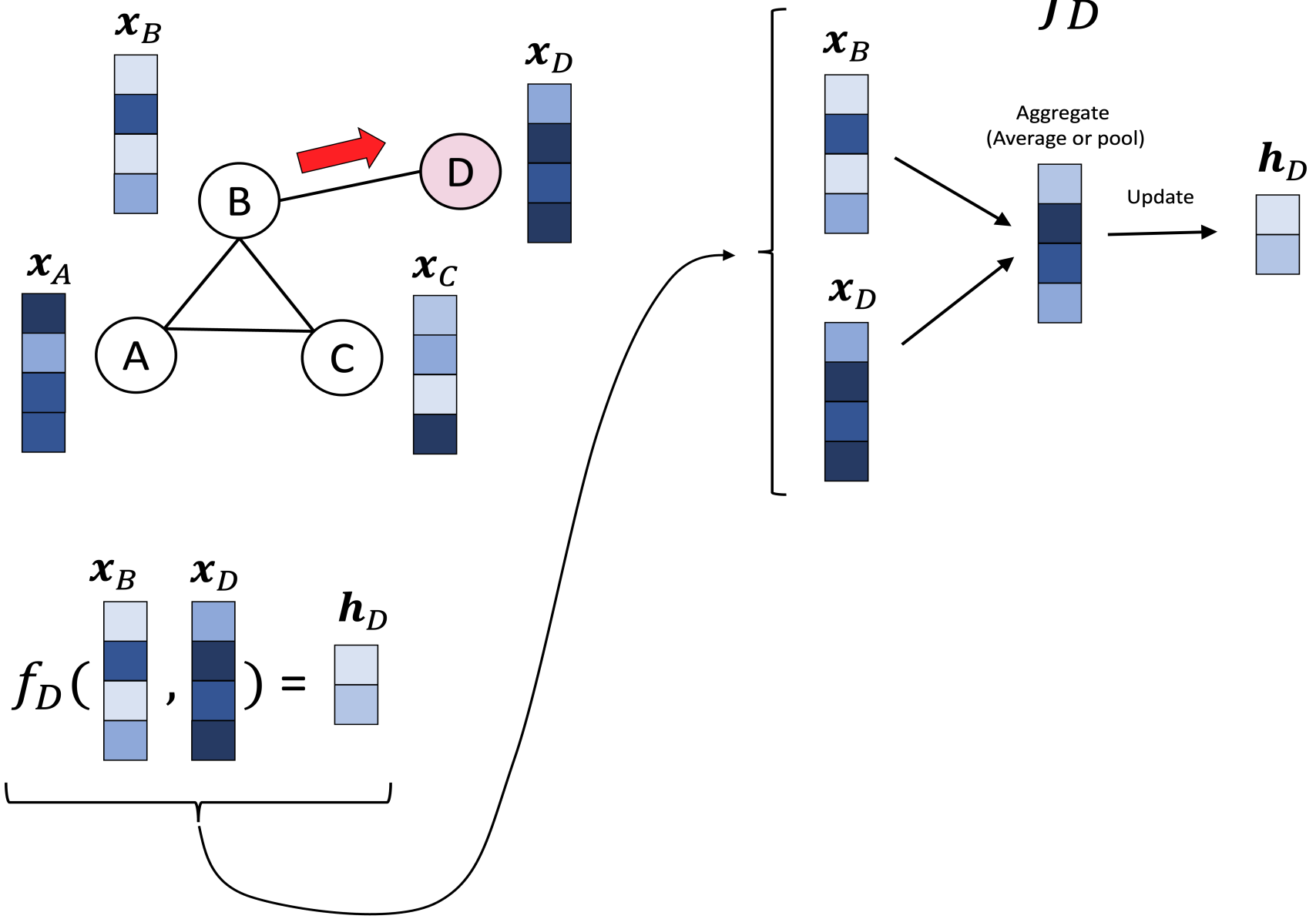
Comparing GCNs and CNNs



The graph convolutional Output



The graph convolutional Process



The graph convolutional Process

$$f(\boldsymbol{X}, \boldsymbol{A}) \coloneqq \sigma\left(\boldsymbol{D}^{-1/2}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-1/2}\boldsymbol{X}\boldsymbol{W}\right)$$

Add self-loops

Normalize adjacency matrix

Aggregate

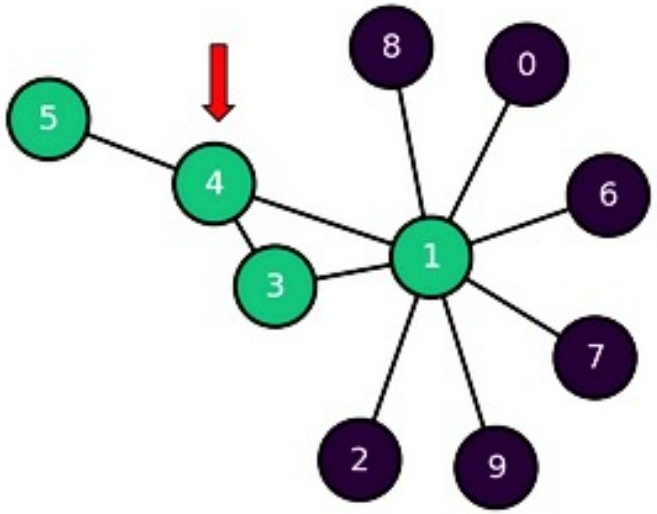
Update

$\boldsymbol{A} \in \mathbb{R}^{n \times n}$:= The adjacency matrix
$\boldsymbol{I} \in \mathbb{R}^{n \times n}$:= The identity matrix
$\boldsymbol{D} \in \mathbb{R}^{n \times n}$:= The degree matrix of $\boldsymbol{A} + \boldsymbol{I}$
$\boldsymbol{X} \in \mathbb{R}^{n \times d}$:= The input data (i.e., the per-node feature vectors)
$\boldsymbol{W} \in \mathbb{R}^{d \times w}$:= The layer's weights
$\sigma(.)$:= The activation function (e.g., ReLU)

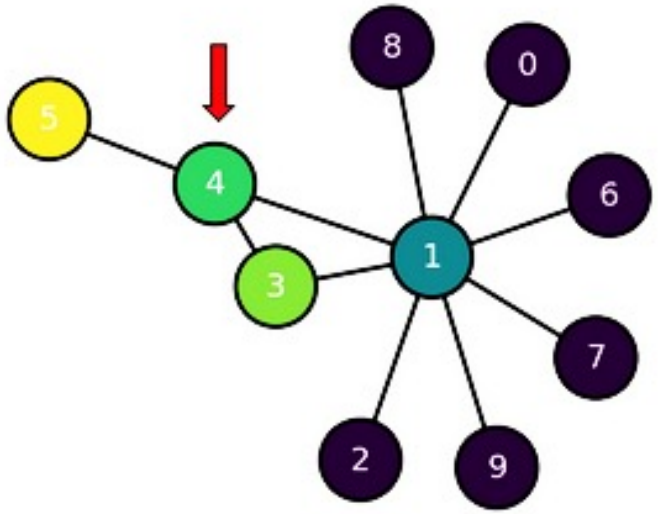
$$\boldsymbol{D} \coloneqq \begin{bmatrix} d_{1,1} & 0 & 0 & \dots & 0 \\ 0 & d_{2,2} & 0 & \dots & 0 \\ 0 & 0 & d_{3,3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & d_{n,n} \end{bmatrix}$$

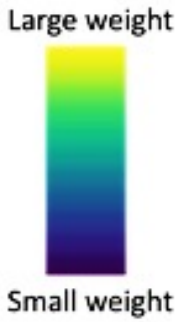
Graph convolutional Equation

$D^{-1}\hat{A}$



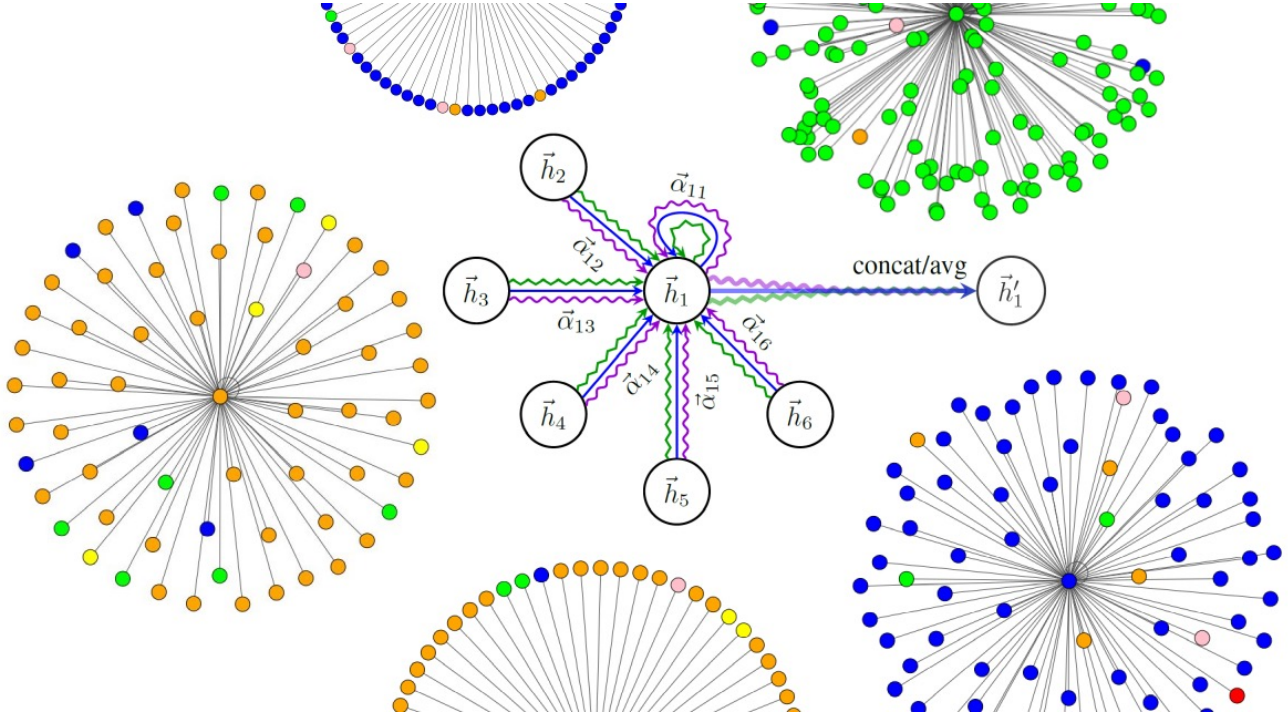
$D^{-1/2}\hat{A}D^{-1/2}$

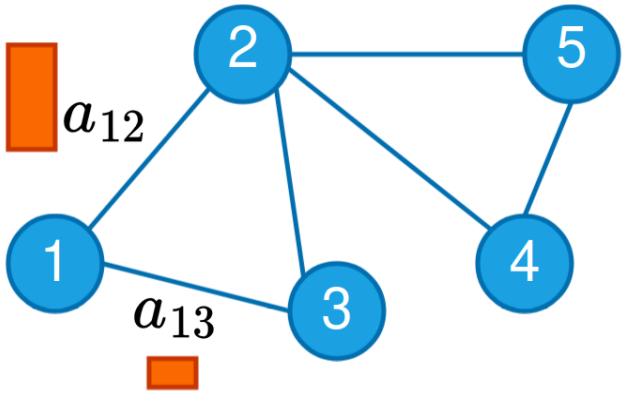




Normalization

Graph Attention Network

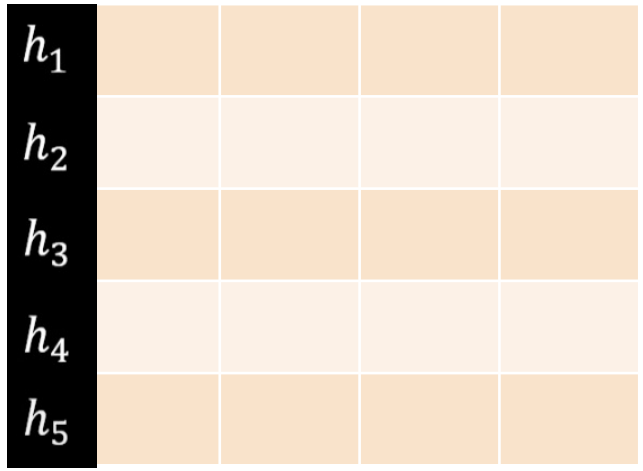




$$h_i' = \sigma\left(\sum_{j \in \mathcal{N}(i)} W h_j\right)$$

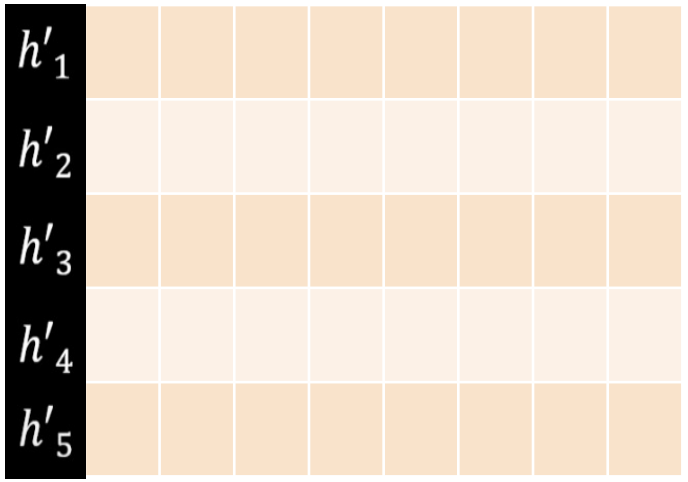
1	1	1	0	0
1	1	1	1	1
1	1	1	0	0
0	1	0	1	1
0	1	0	1	1

Adj Matrix



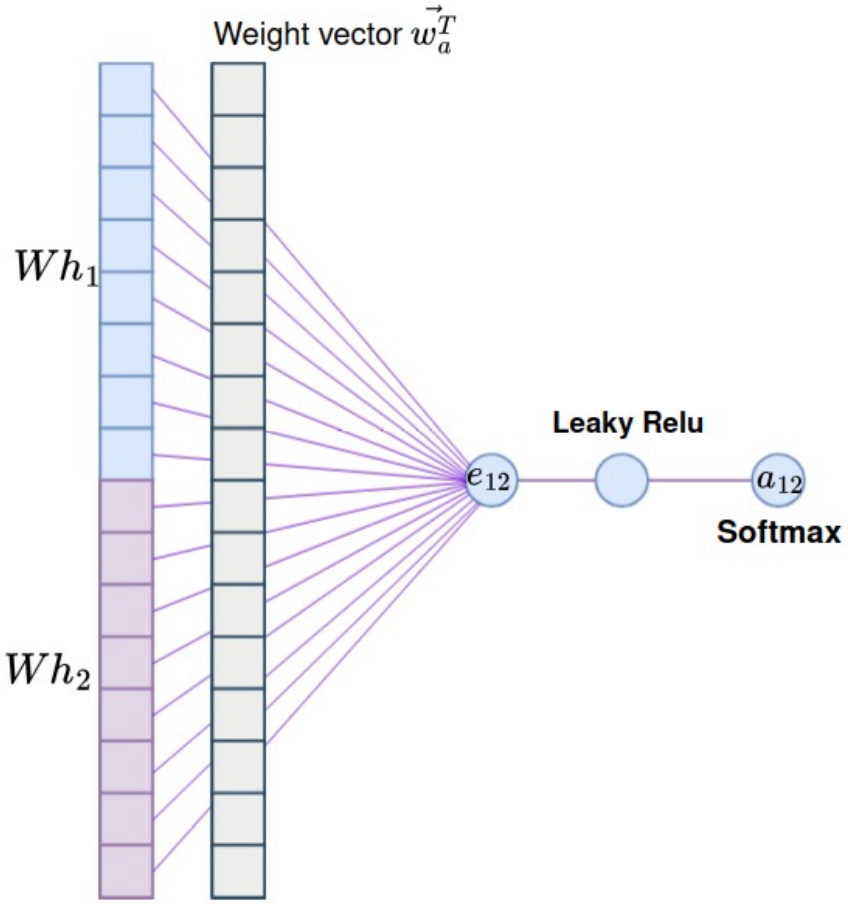
Features per node

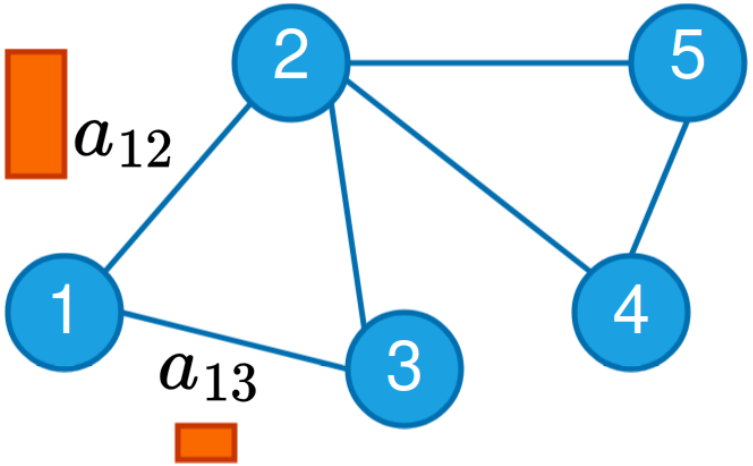
Learnable weight Matrix



Embeddings per node

Graph Attention Network



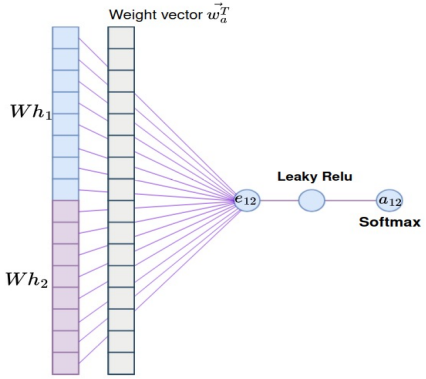


$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{w}_a^T \left[Wh_i \parallel Wh_j\right]\right)\right)}{\sum_{k \in \mathcal{N}(i)} \exp\left(\text{LeakyReLU}\left(\vec{w}_a^T \left[Wh_i \parallel Wh_k\right]\right)\right)}$$

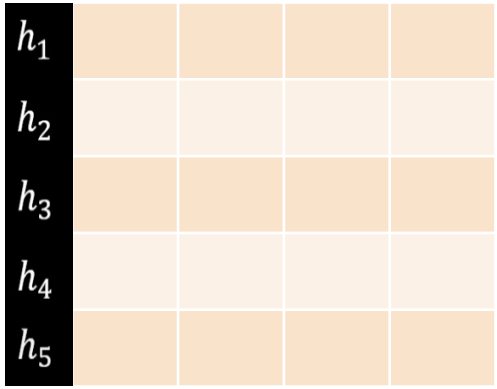
$$h_i' = \sigma\left(\sum_{j \in N(i)} a_{ij} W h_j\right)$$

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

Adj Matrix

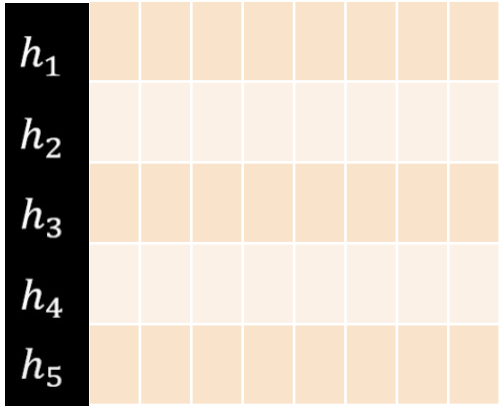


Attention



Features per node

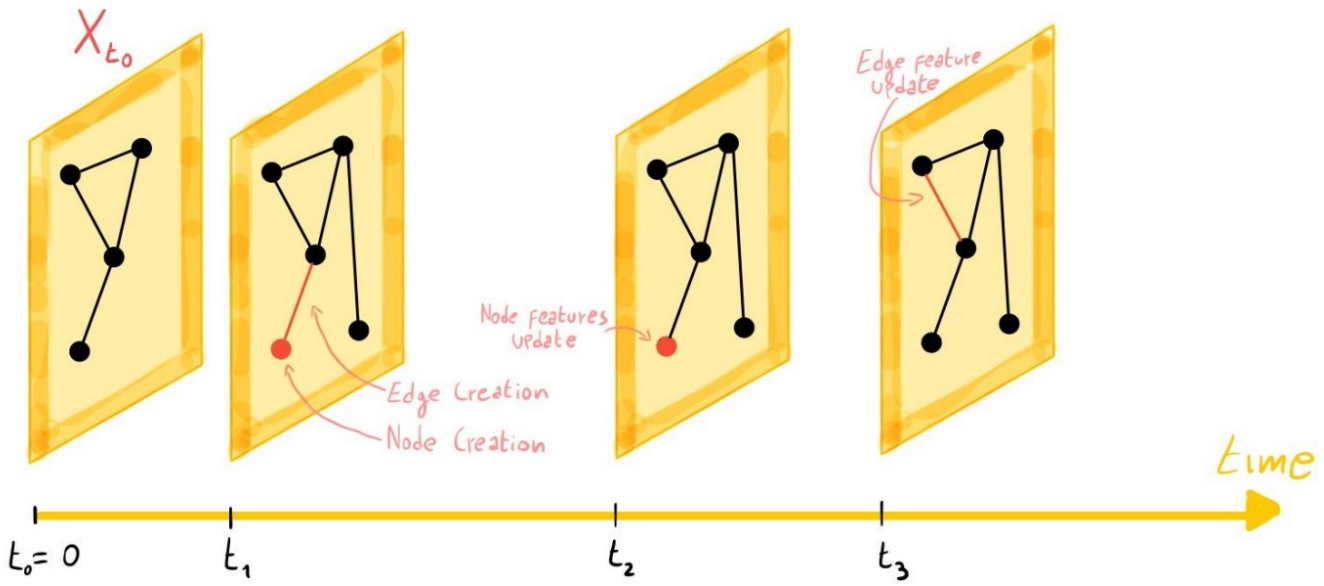
Learnable weight Matrix

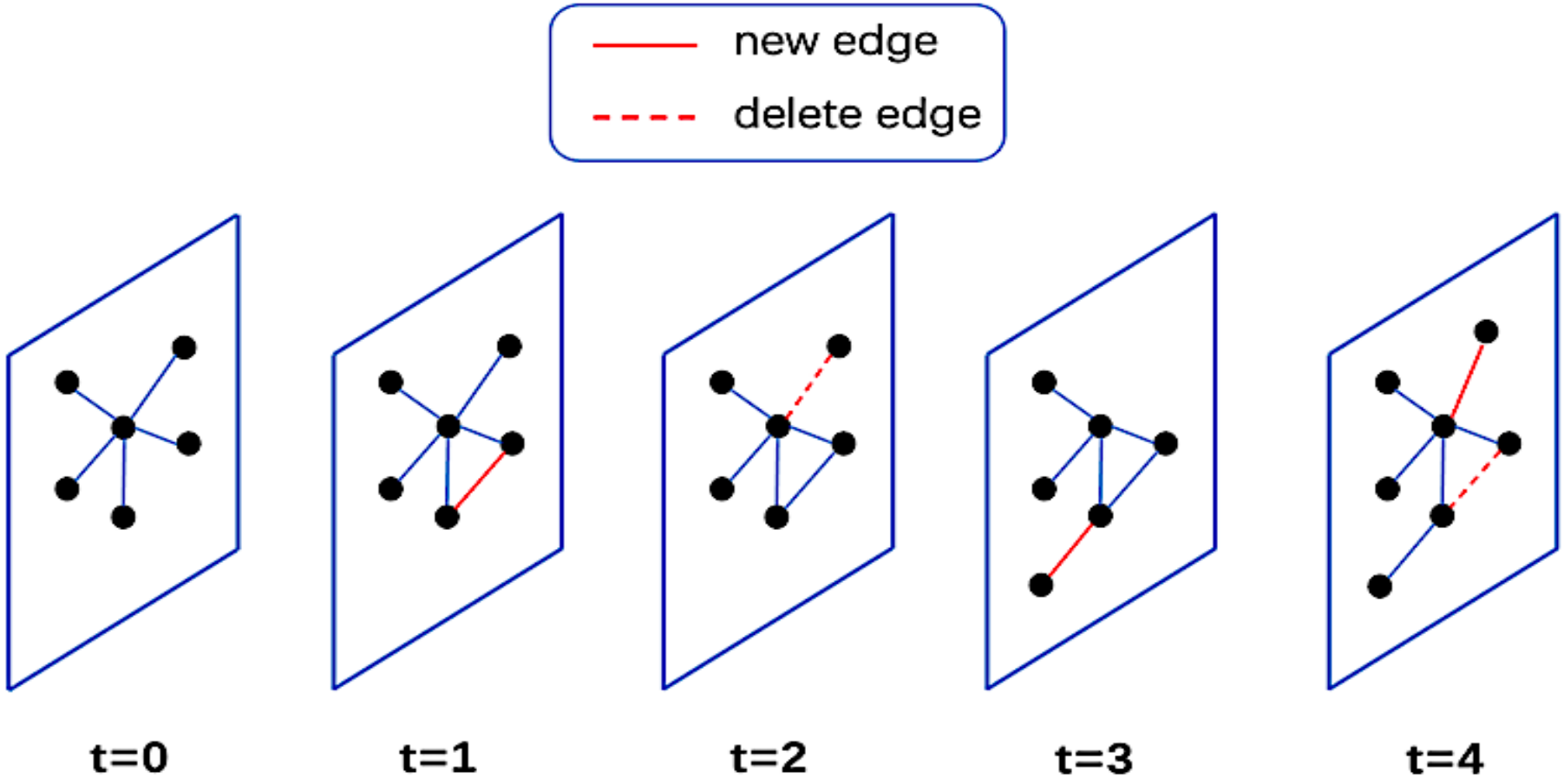


Embeddings per node

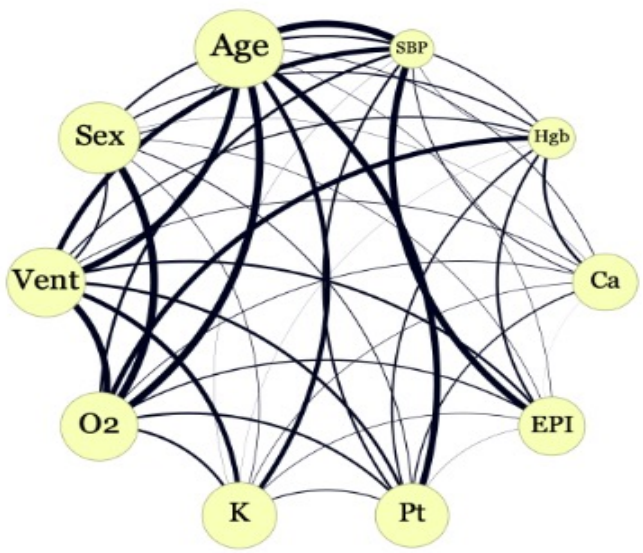
Graph Attention Network

Dynamic Graph Neural Network

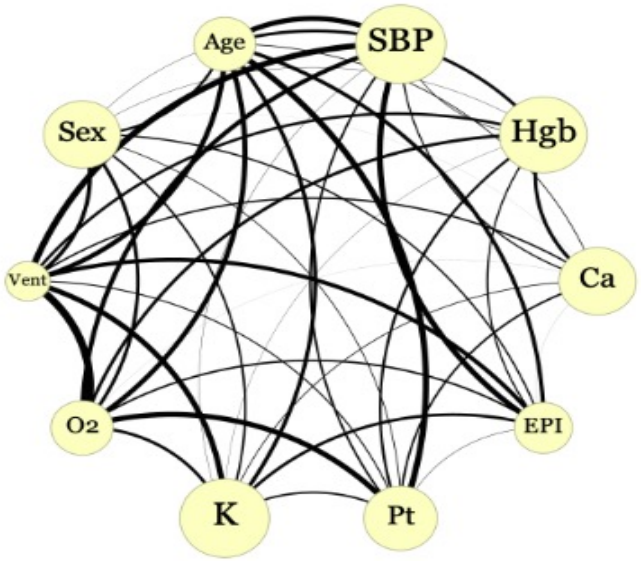




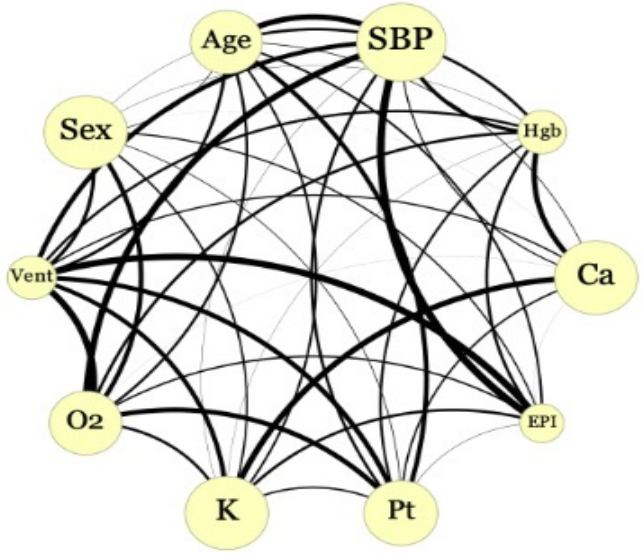
Dynamic Graphs



(a) t=1

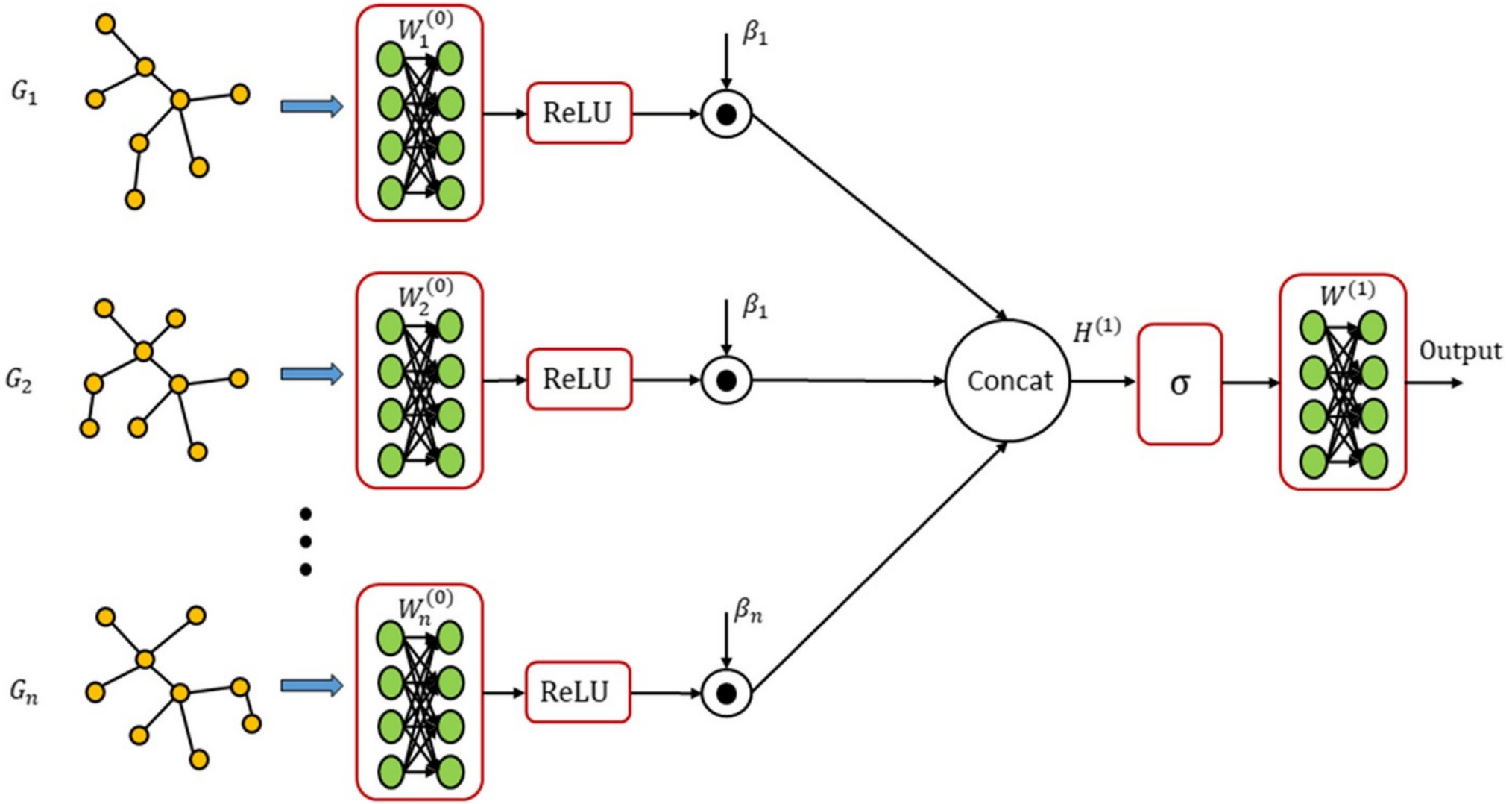


(b) t=4

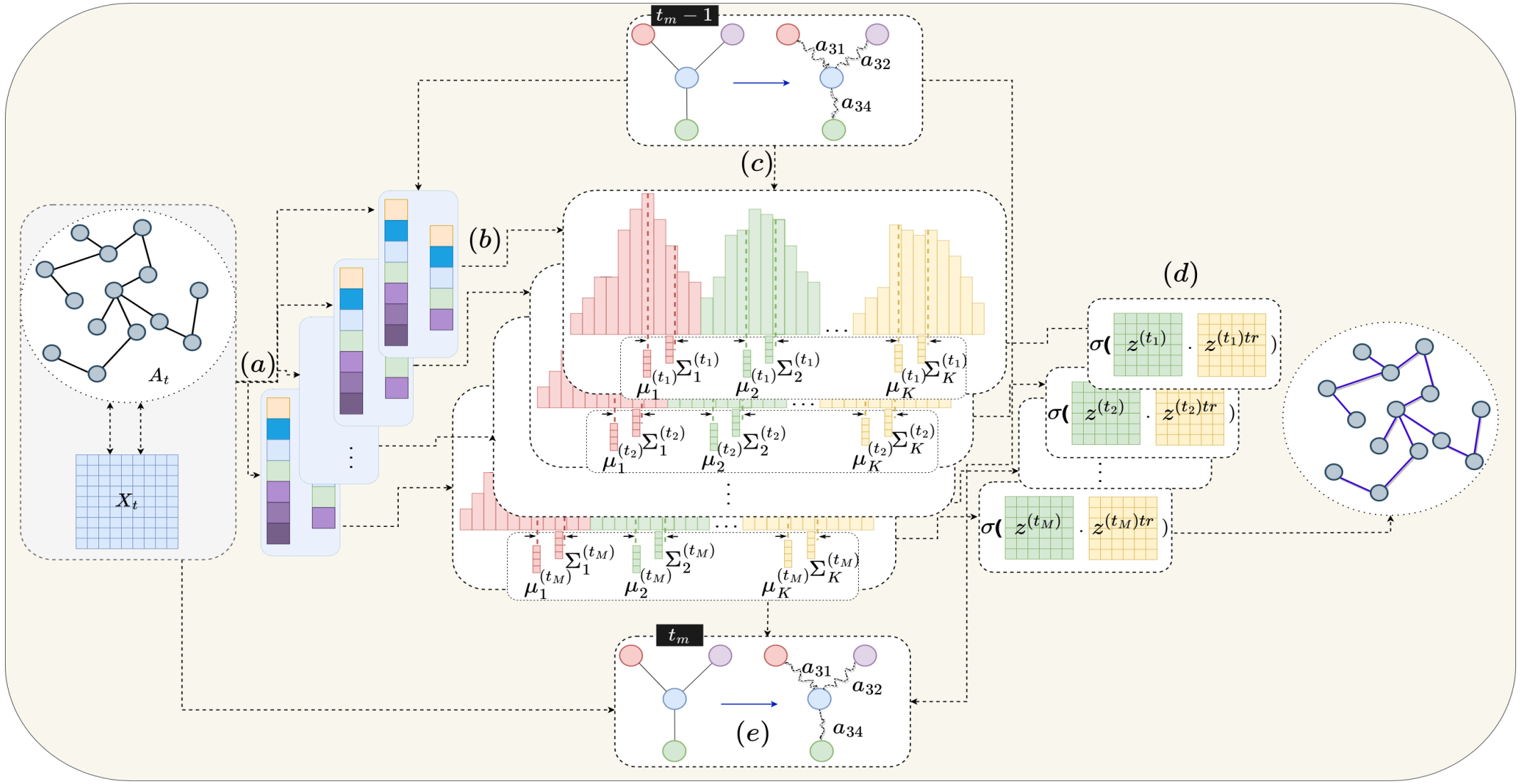


(c) t=6

Dynamic GNN Network



Dynamic GNN Network



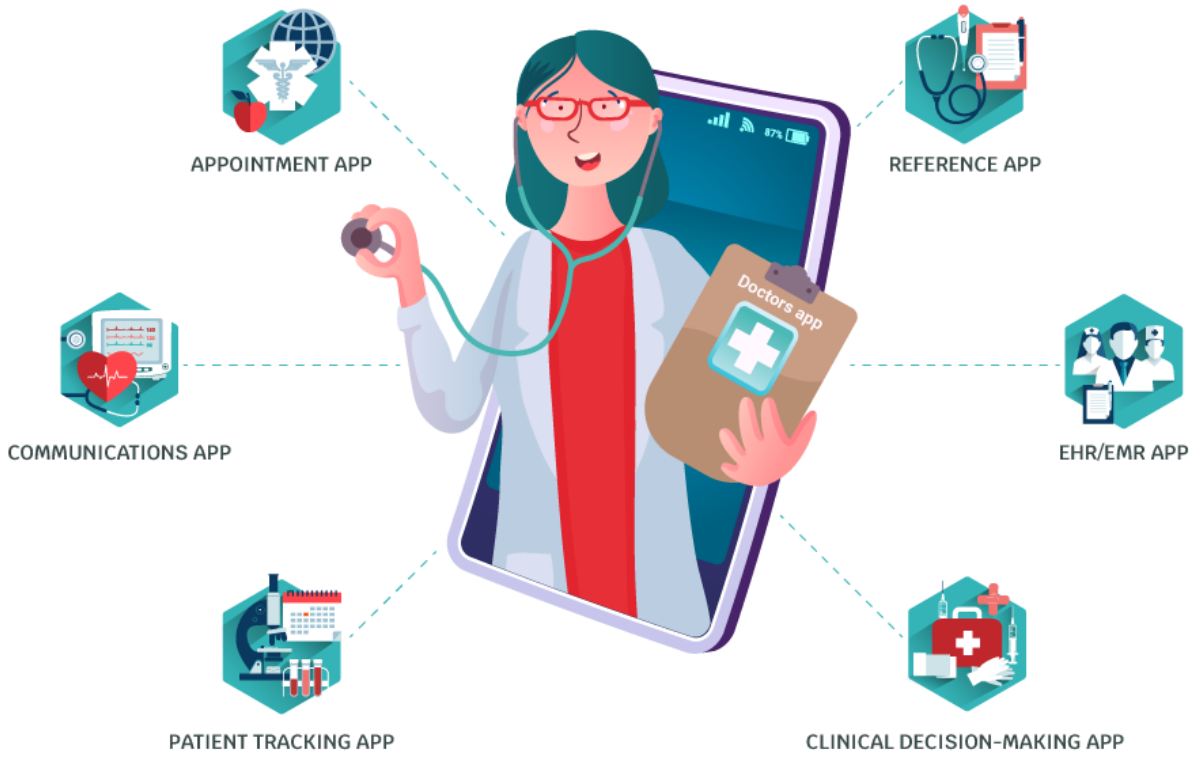
Computational Health Informatics

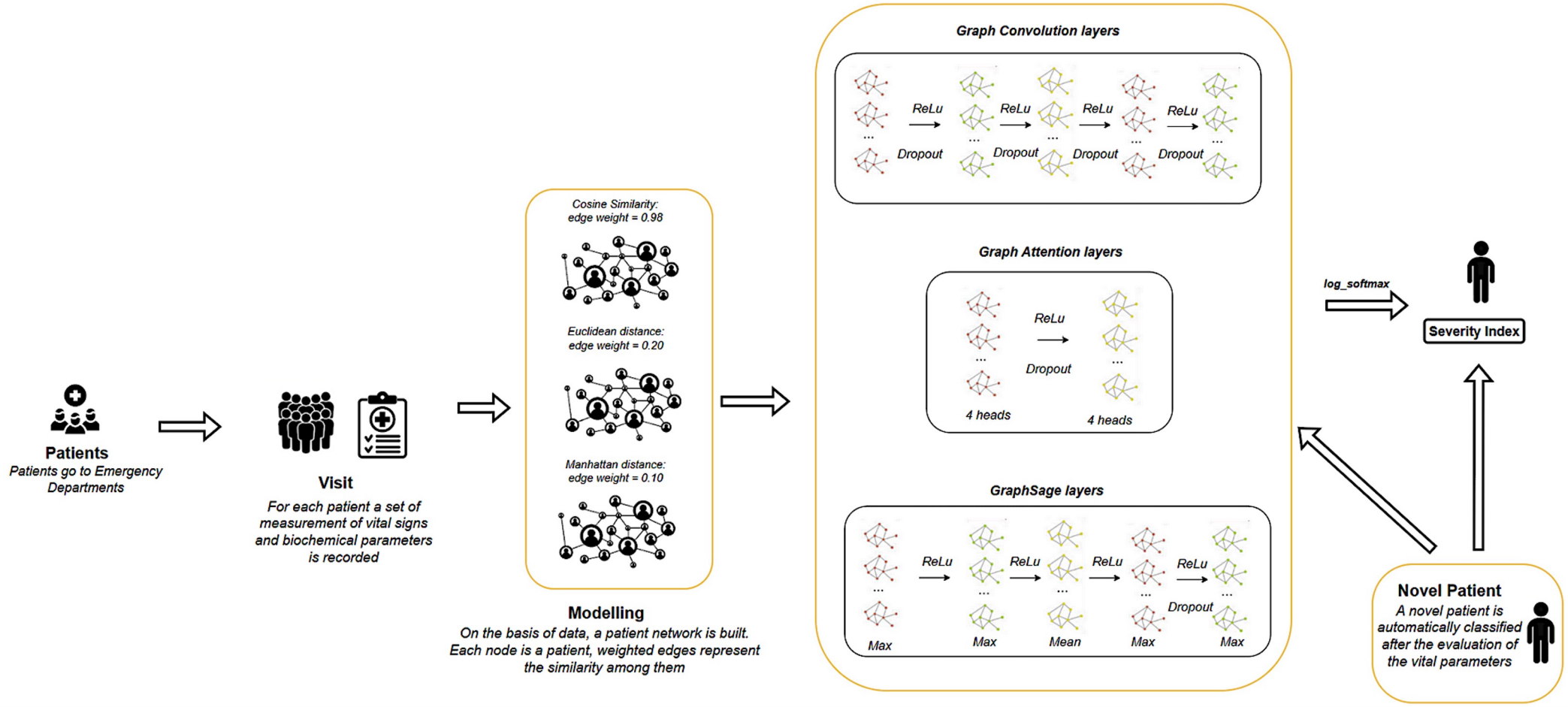
Dynamic Variational Model

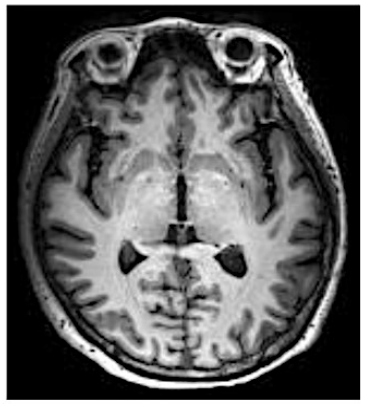
- Handling **Small Datasets**
- Robustness to **Missing Information**
- Effective with **Sparse** Data
- Adaptability to **Non-Euclidean** Data
- Capturing **Complex Interactions**
- Suitability for **Dynamic** and **Heterogeneous** Data

Practical Scenarios

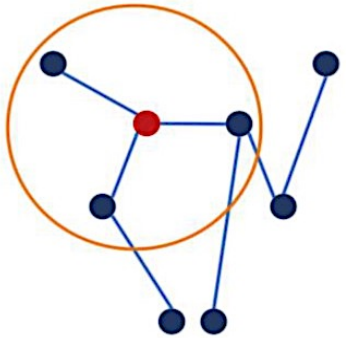
Applications in Healthcare



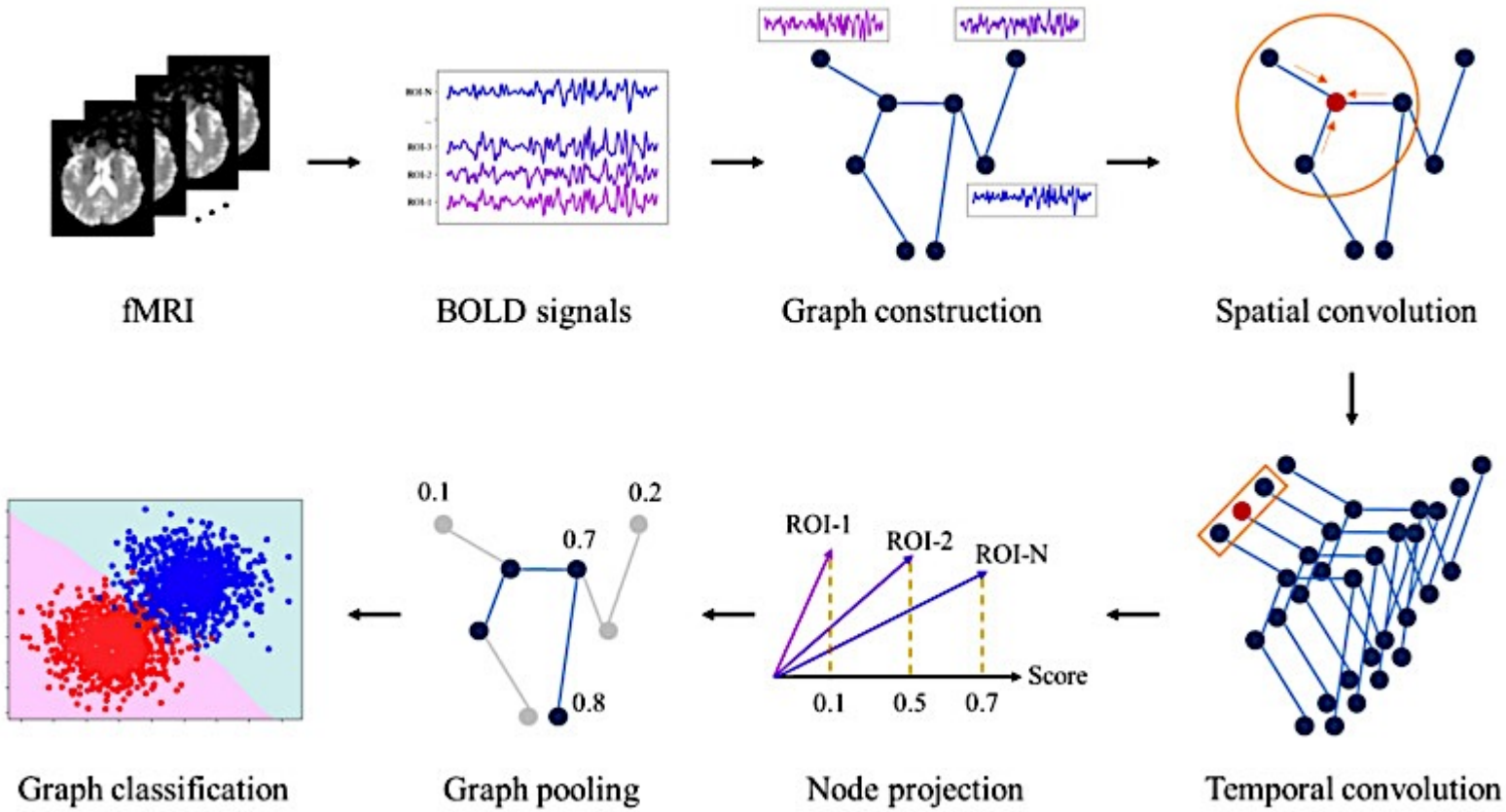

Patients
Patients go to Emergency Departments
Visit
For each patient a set of measurement of vital signs and biochemical parameters is recorded
Cosine Similarity: edge weight = 0.98
Euclidean distance: edge weight = 0.20
Manhattan distance: edge weight = 0.10
Modelling
On the basis of data, a patient network is built. Each node is a patient, weighted edges represent the similarity among them
Graph Convolution layers
ReLu
Dropout
Graph Attention layers
4 heads
GraphSage layers
Max
Mean
log_softmax
Severity Index
Novel Patient
A novel patient is automatically classified after the evaluation of the vital parameters



(a) Brain MRI



(b) GNN modeling of the brain



Brain Graph

Thank You