CS502: Deep Learning in Biomedicine
Graph Convolutional Neural Networks

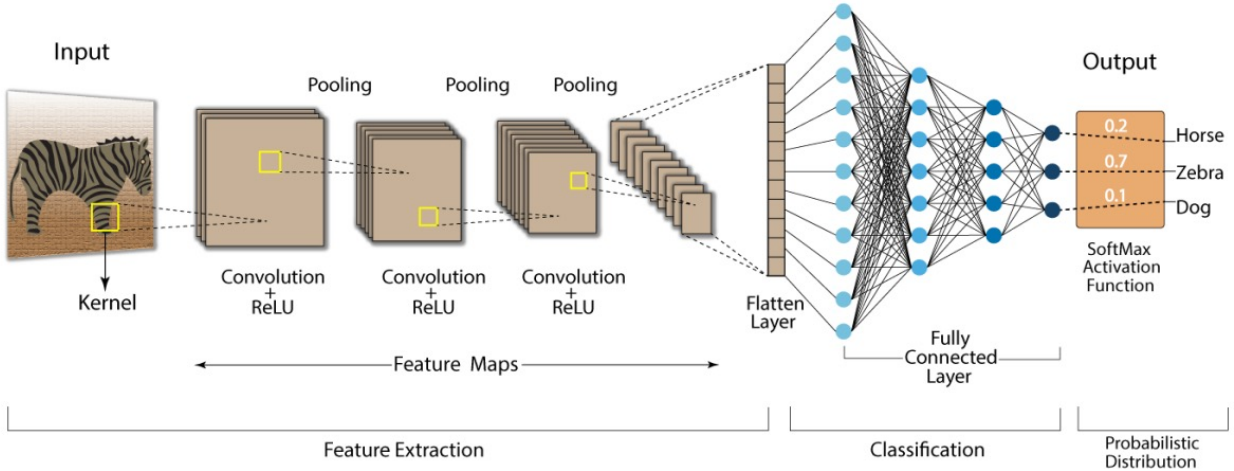
Maria Brbić

Fall 2023

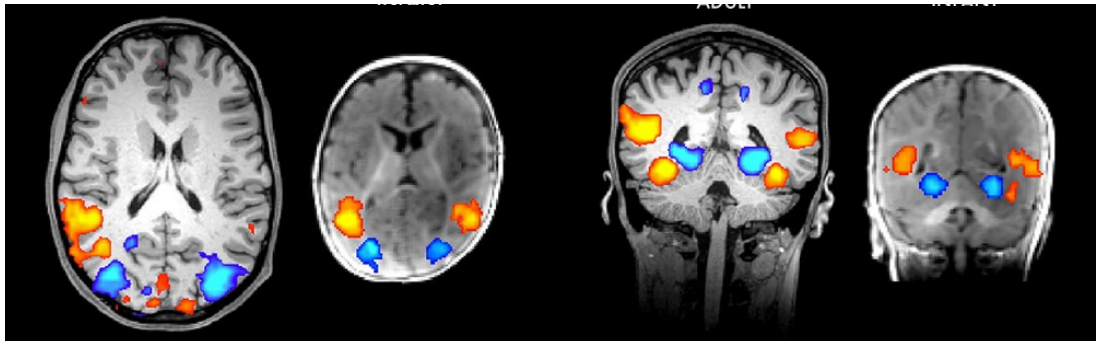
École
polytechnique
fédérale
de Lausanne

Last Week: Recap

- What we covered last time:
  - Convolutional neural networks

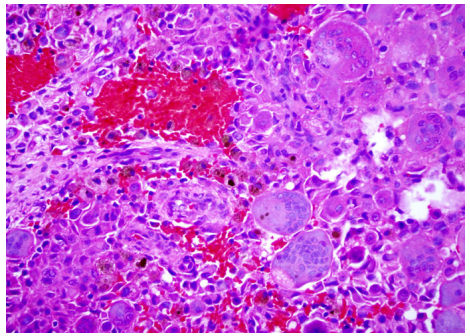


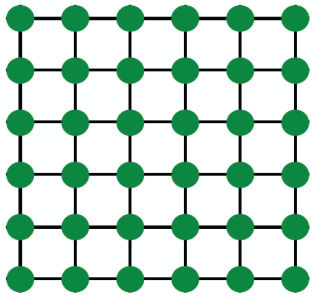
  - Applications to medical images

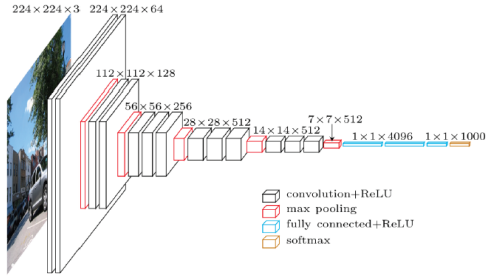


Architectures for Different Input Types

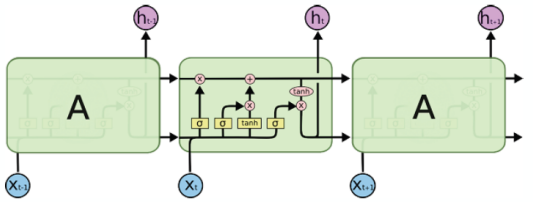
Images

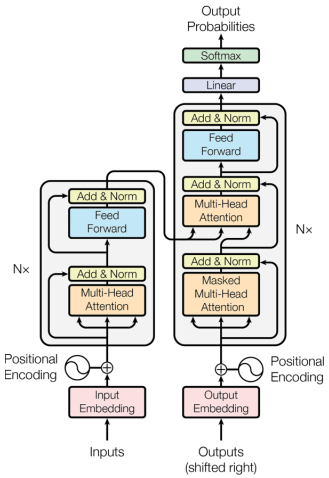






Text/Speech

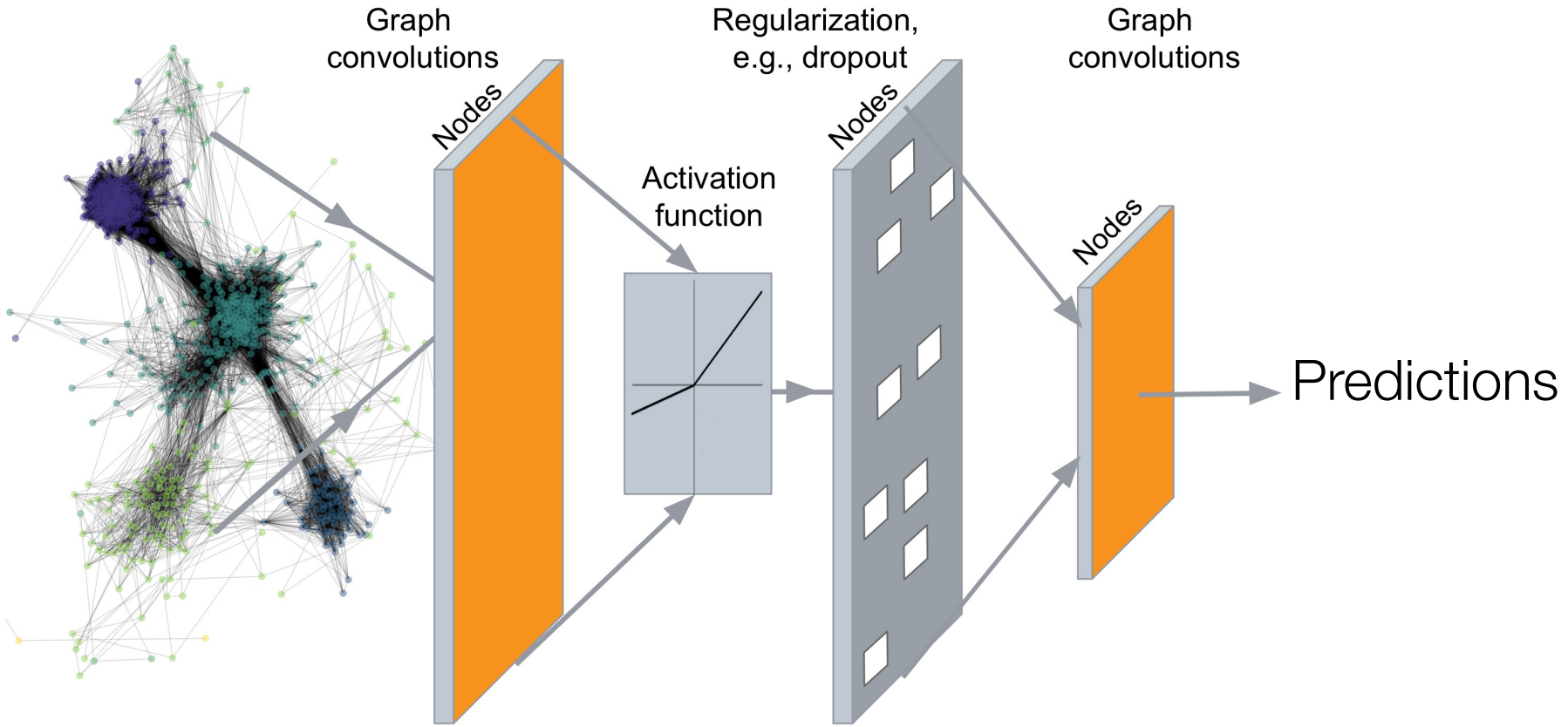




But, what if the input data is a graph?

Today: Graph Convolutional Neural Networks (GCNs)

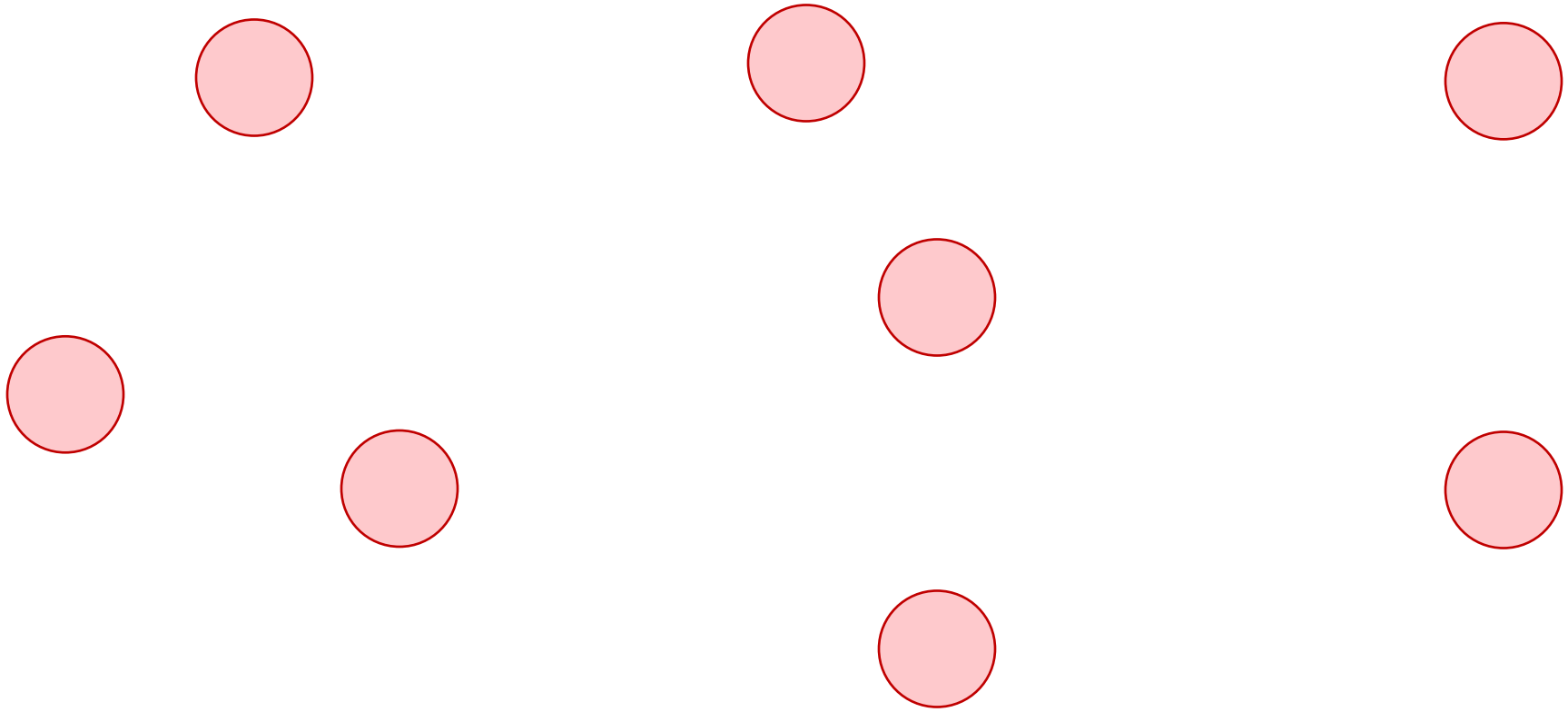
- Default architecture to handle graph structured datasets



Based on CS224W: https://web.stanford.edu/class/cs224w/

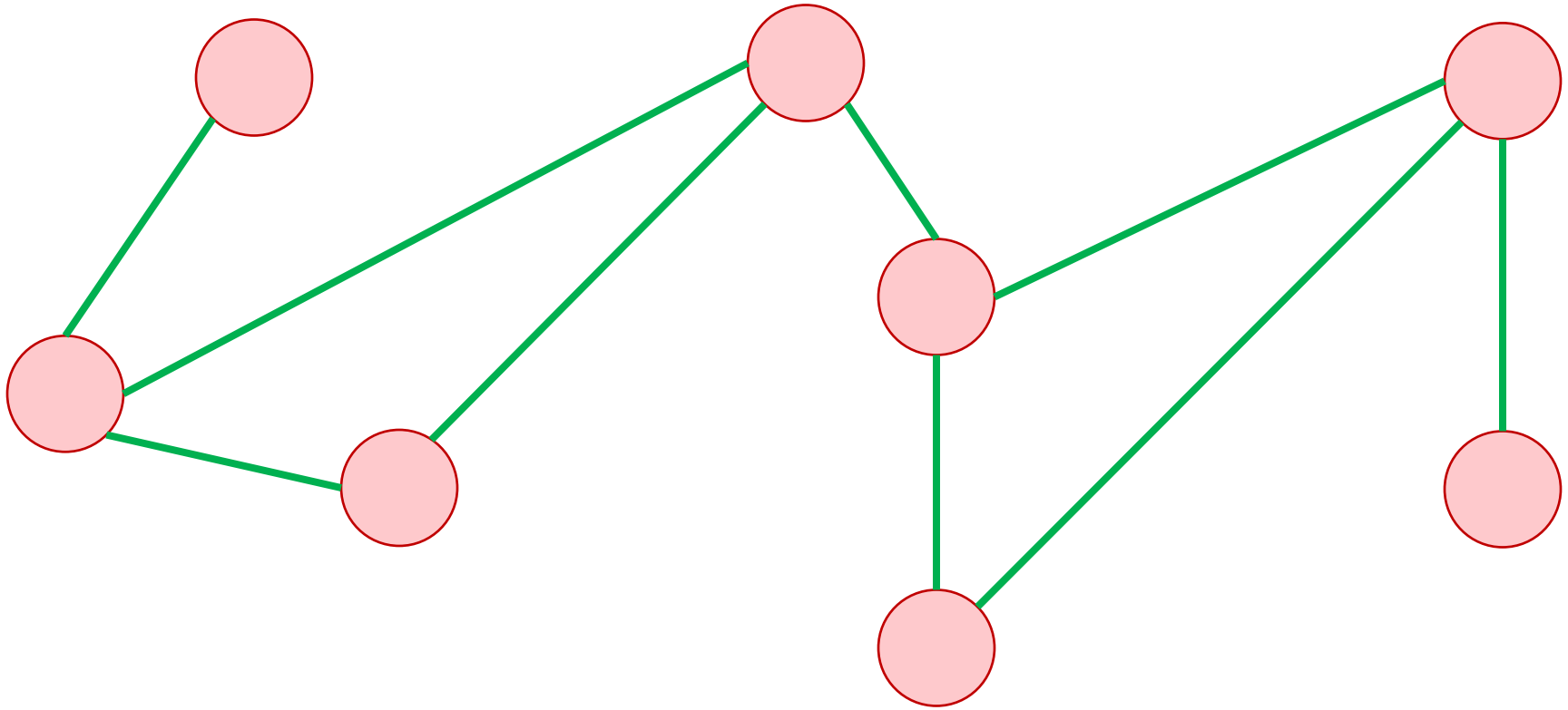
Graph Representation

Allows us to capture relationships between objects



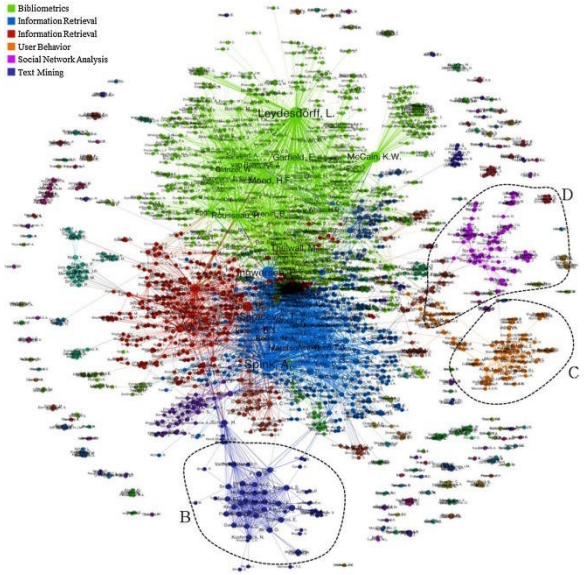
Graph Representation

Allows us to capture relationships between objects



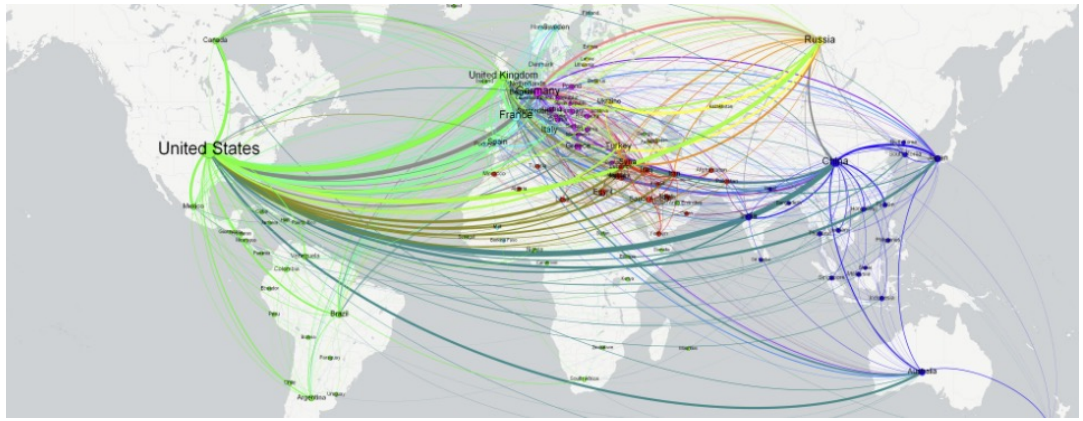
Why Graphs?

Social networks



Citation networks

Infrastucture networks

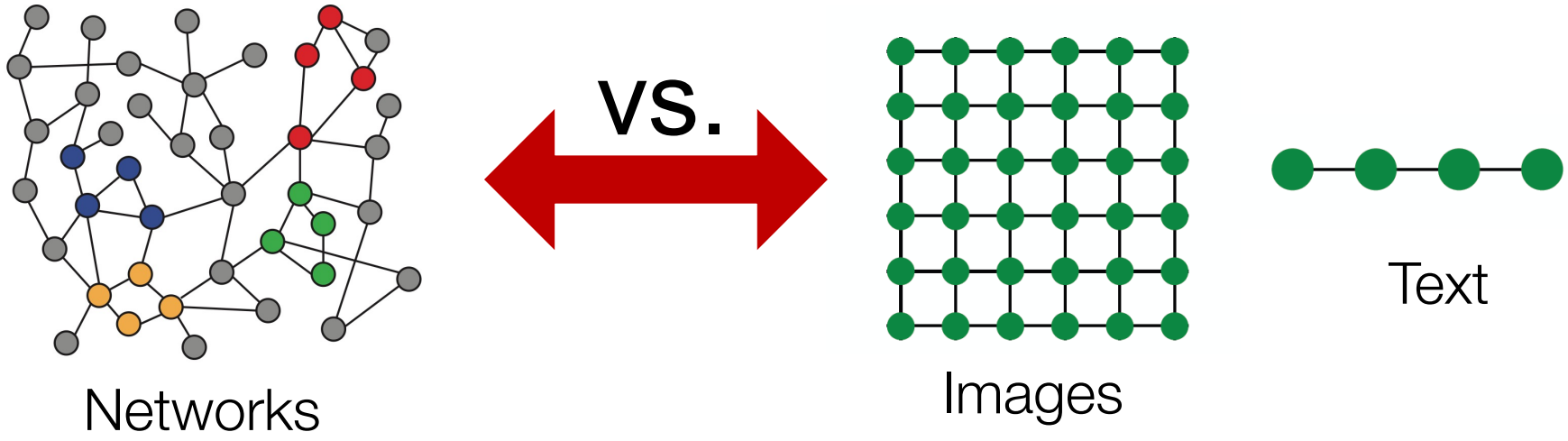


Country-level networks

How to handle graph input data with neural networks?

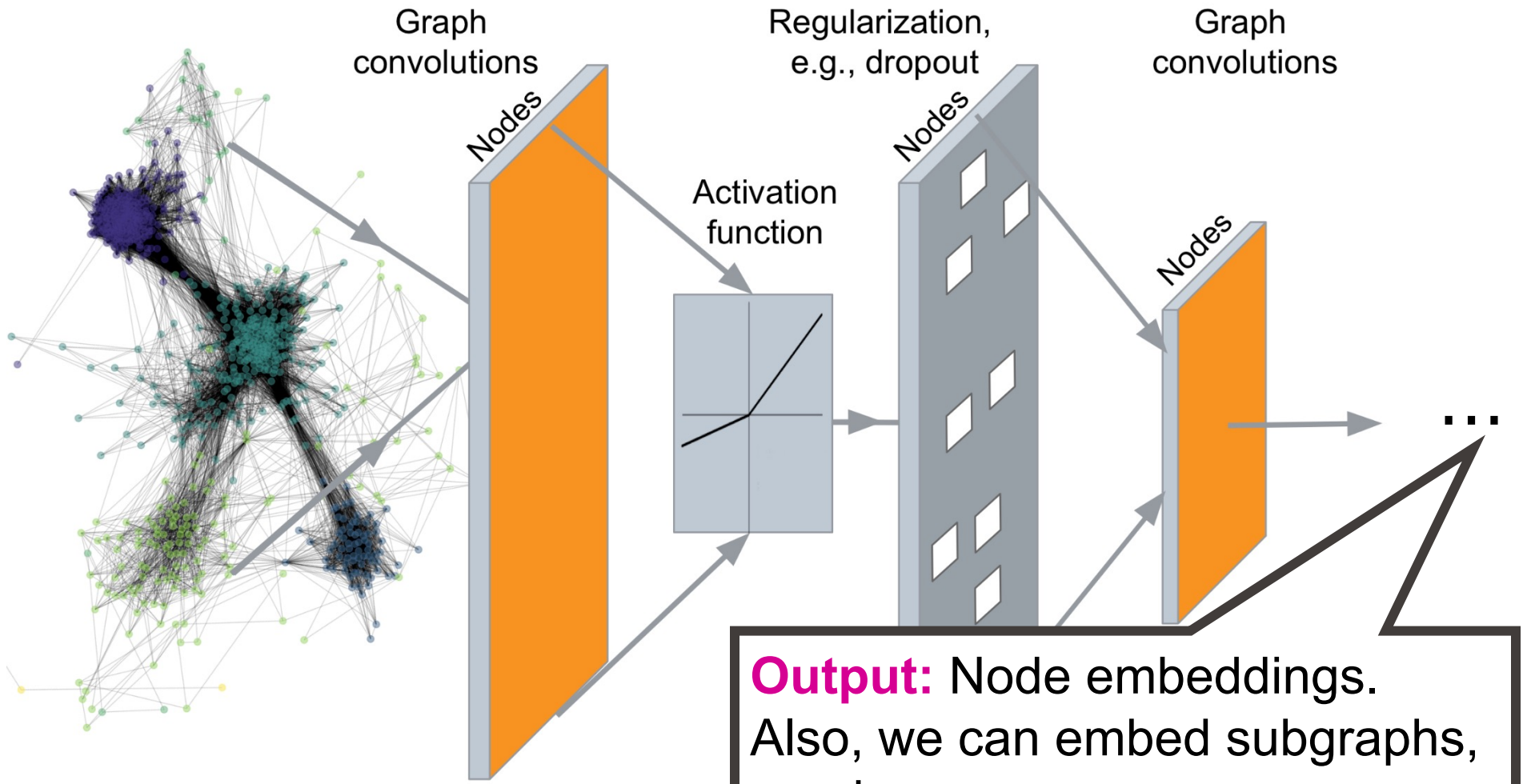
Why Is It Hard?

Networks are complex!



- Arbitrary size and complex topological structure
- No fixed node ordering or reference point
- Often dynamic and have multimodal features

Deep Learning in Graphs

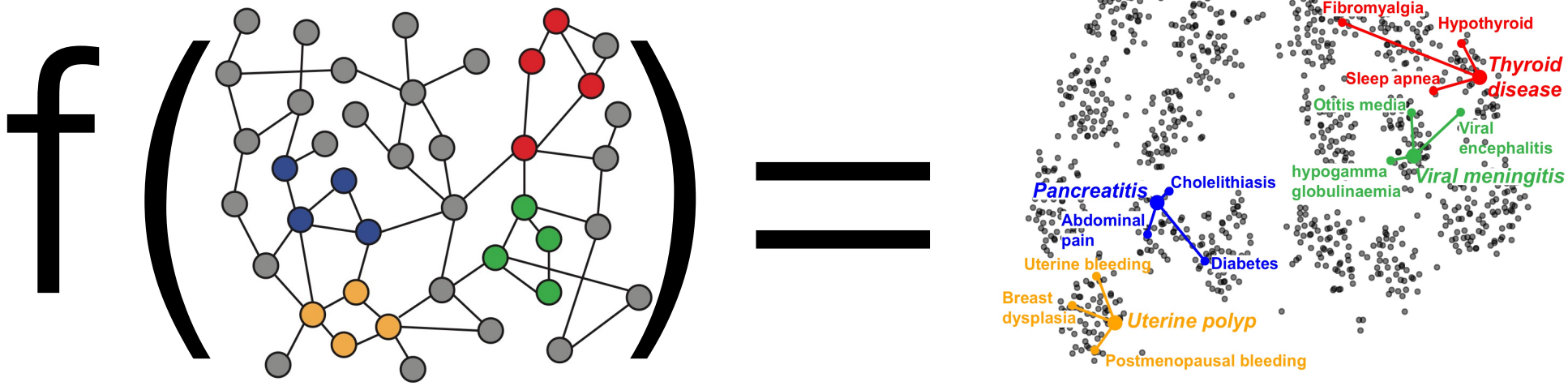


Output: Node embeddings. Also, we can embed subgraphs, graphs

Feature Learning in Graphs

Input

Output



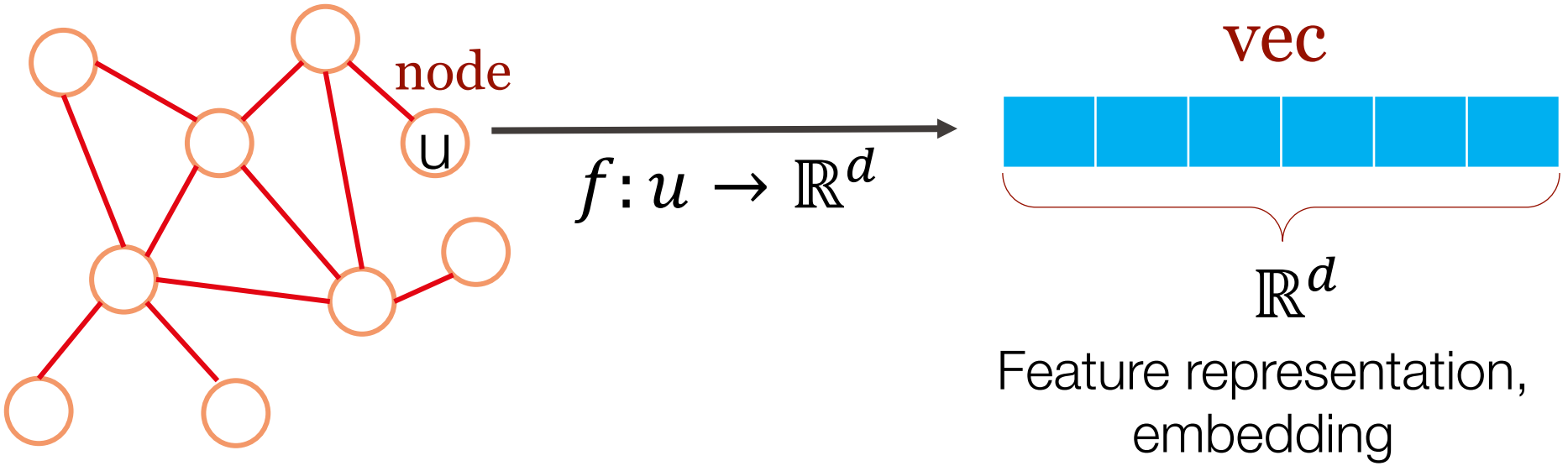
Input graph

Low-dimensional node embeddings

This lecture: How to learn mapping function f?

Feature Learning in Graphs

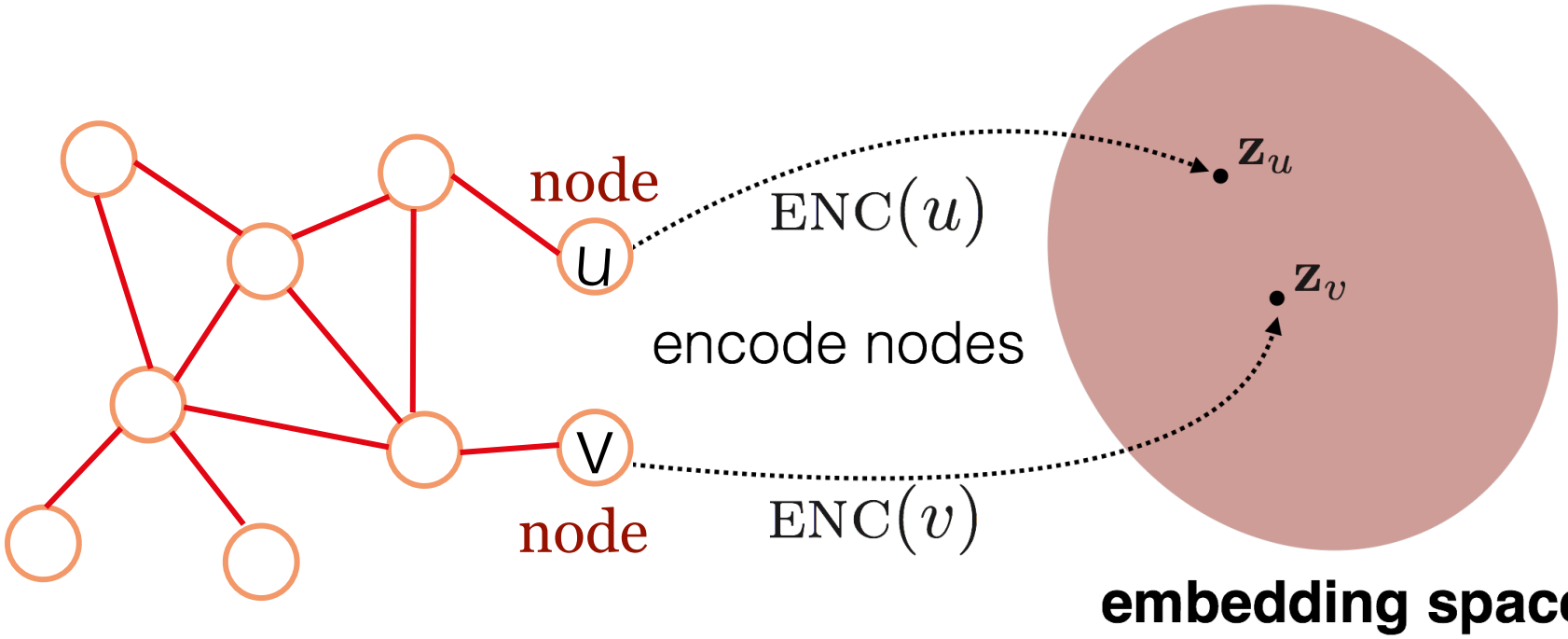
Goal: Feature learning for machine learning in networks



Feature Learning in Graphs

Map nodes so that similarity in the embedding space (e.g., dot product) approximates

similarity in the network



Ways to Analyze Networks

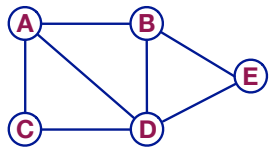
- Predict a type of a given node
  - Node classification
- Predict whether two nodes are linked
  - Link prediction
- Identify densely linked clusters of nodes
  - Community detection
- Predict a type of a given graph/subgraph
  - Graph classification

Setup

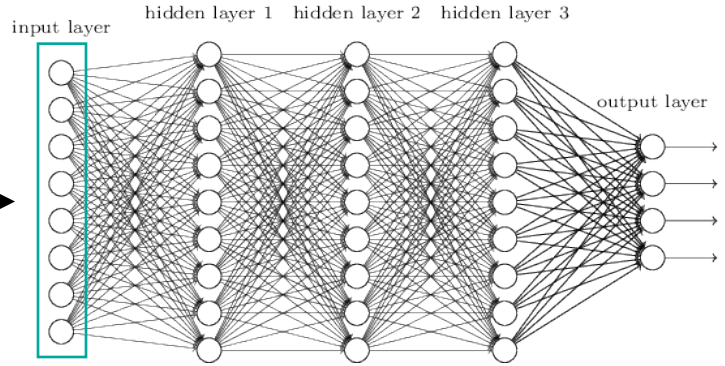
- **Assume we have a graph G:**
  - V is the **vertex set**
  - $\boldsymbol{A}$ is the **adjacency matrix**
  - $\boldsymbol{X} \in \mathbb{R}^{m \times |V|}$ is a matrix of **node features**
  - v: a node in V; $N(v)$: the set of neighbors of v.
  - **Node features:**
    - Biological networks: Gene expression profiles, gene functional information
    - When there is no node feature in the graph dataset, e.g., indicator vectors (one-hot encoding of a node) or vector of constant values

A Naïve Approach

- Join adjacency matrix and features
- Feed them into a deep neural net:



	A	B	C	D	E	Feat	
A	0	1	1	1	0	1	0
B	1	0	0	1	1	0	0
C	1	0	0	1	0	0	1
D	1	1	1	0	1	1	1
E	0	1	0	1	0	1	0

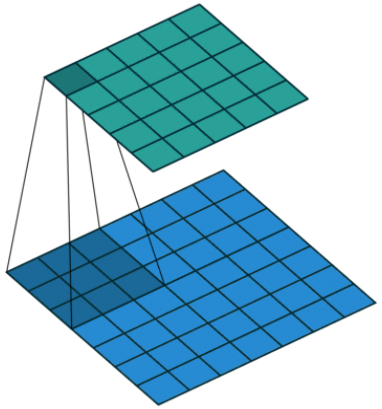


?

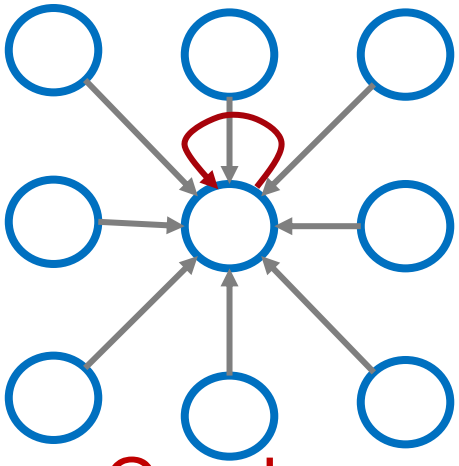
- Issues with this idea:
  - $O(|V|)$ parameters
  - Not applicable to graphs of different sizes
  - Sensitive to node ordering

From Images to Graphs

Single Convolutional neural network (CNN) layer with 3x3 filter:



Image



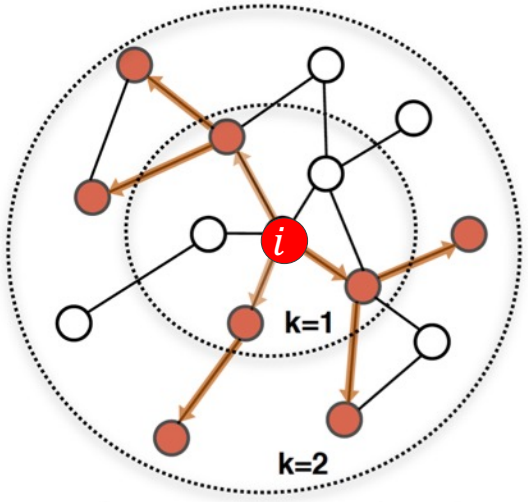
Graph

Idea: transform information at the neighbors and combine it:

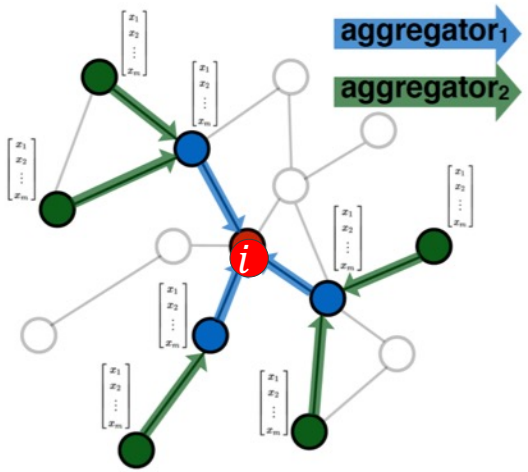
- Transform "messages" h_i from neighbors: $W_i\ h_i$
- Add them up: $\sum_i W_i\ h_i$

Graph Convolutional Networks

Idea: Node's neighborhood defines a computation graph



Determine node computation graph



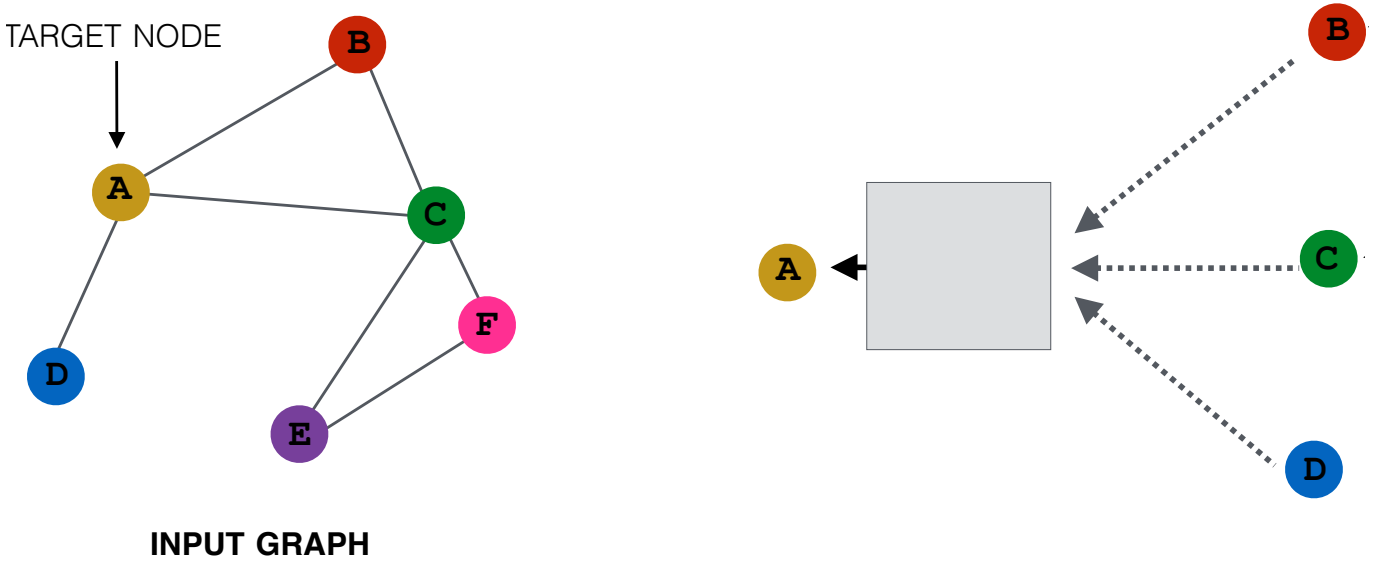
Aggregate and transform information

Learn how to propagate information across the graph to compute node features

[Kipf and Welling, ICLR 2017]

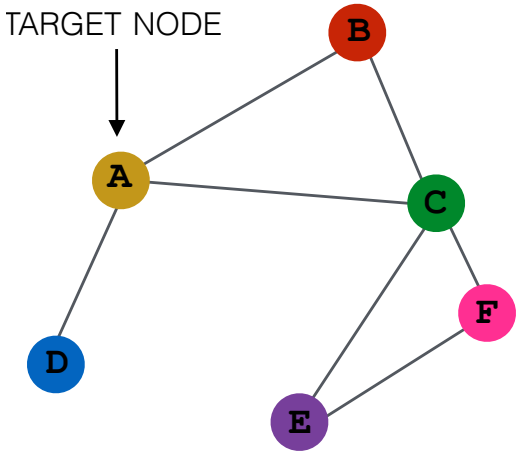
Idea: Aggregate Neighbors

- **Key idea:** Generate node embeddings based on local network neighborhoods

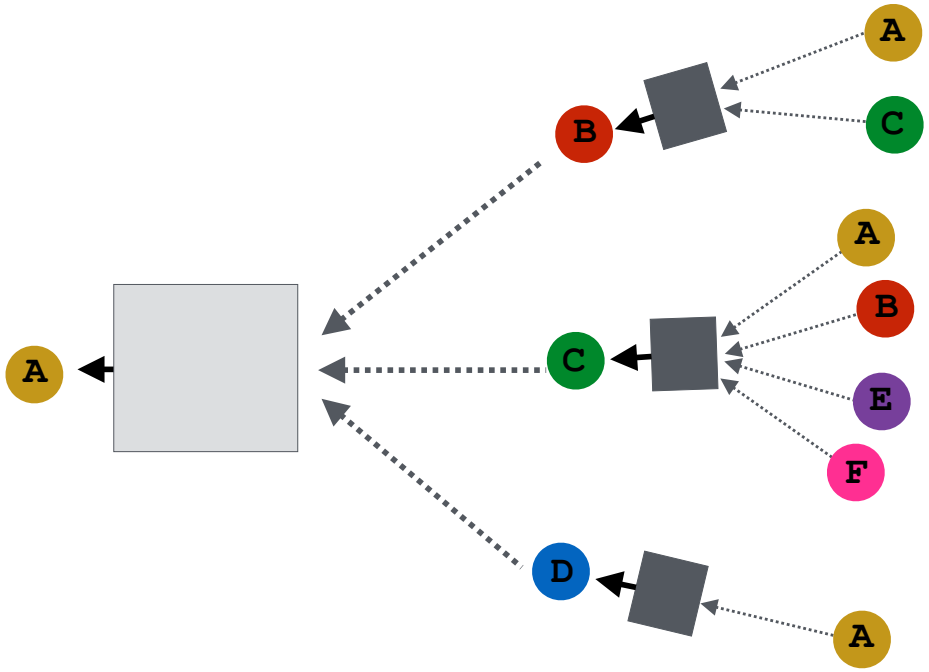


Idea: Aggregate Neighbors

- **Key idea:** Generate node embeddings based on local network neighborhoods

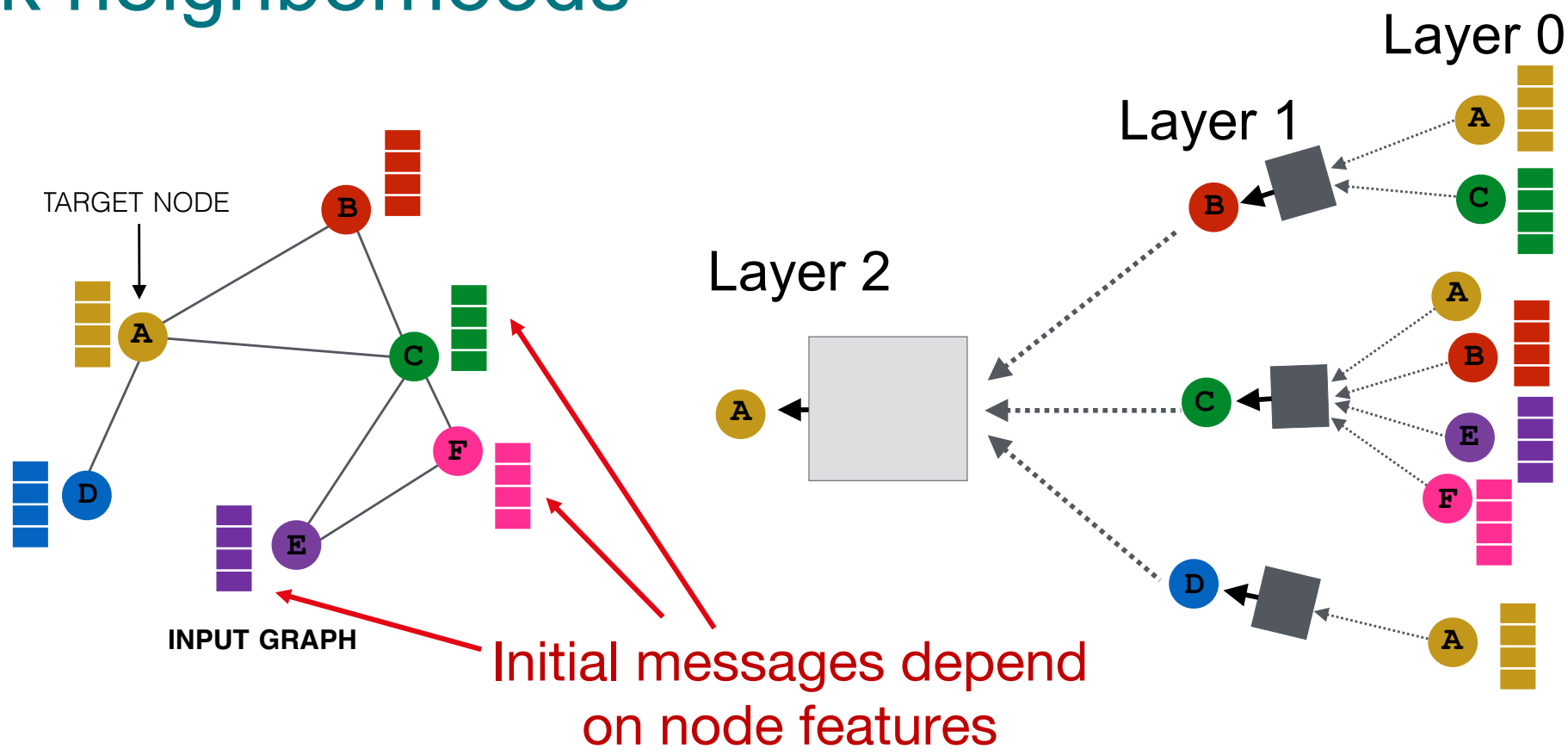


INPUT GRAPH



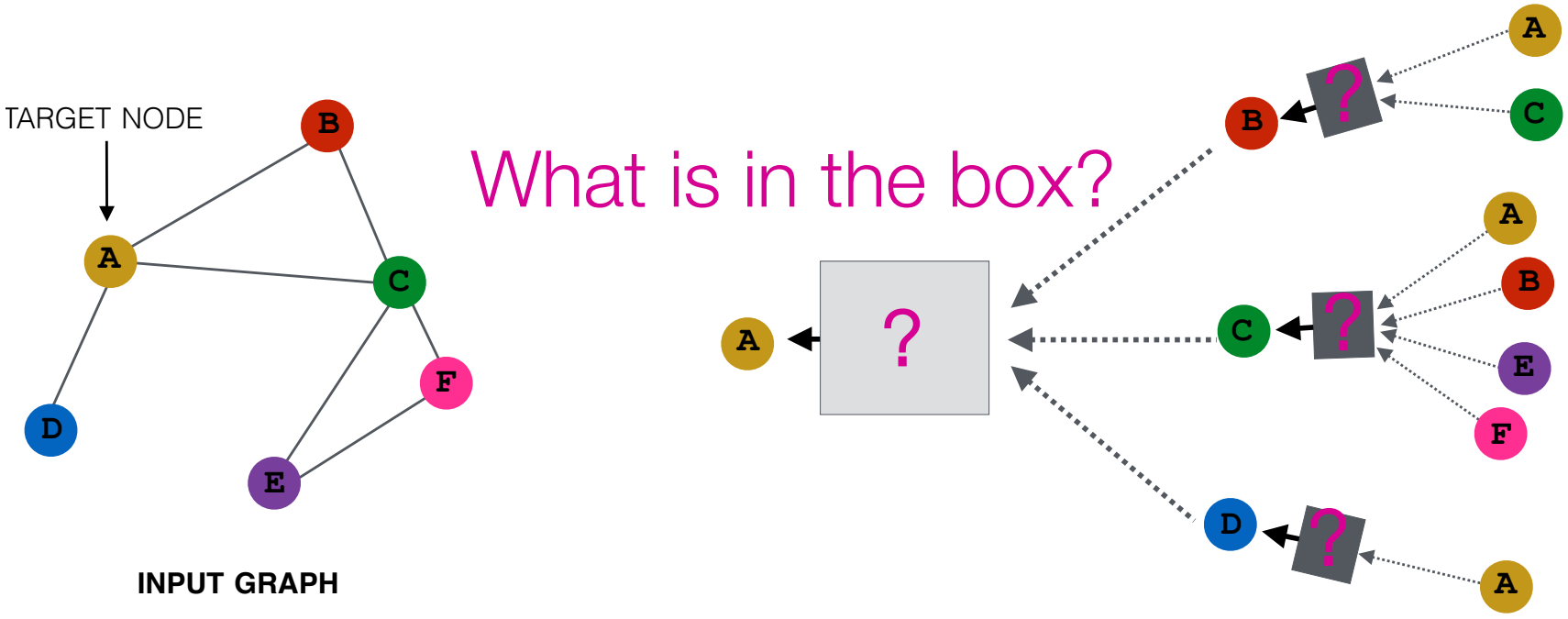
Idea: Aggregate Neighbors

- **Key idea:** Generate node embeddings based on local network neighborhoods



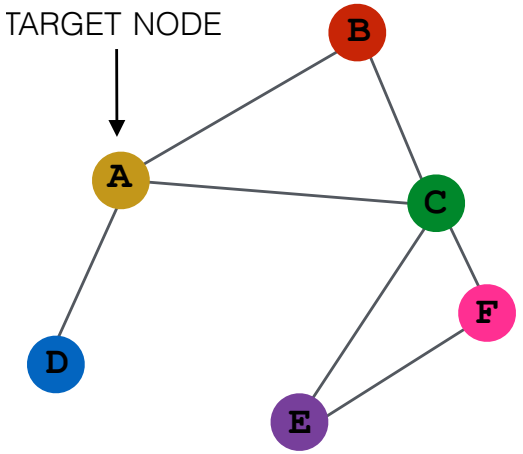
Neighborhood Aggregation

- **Key idea:** Generate node embeddings based on local network neighborhoods

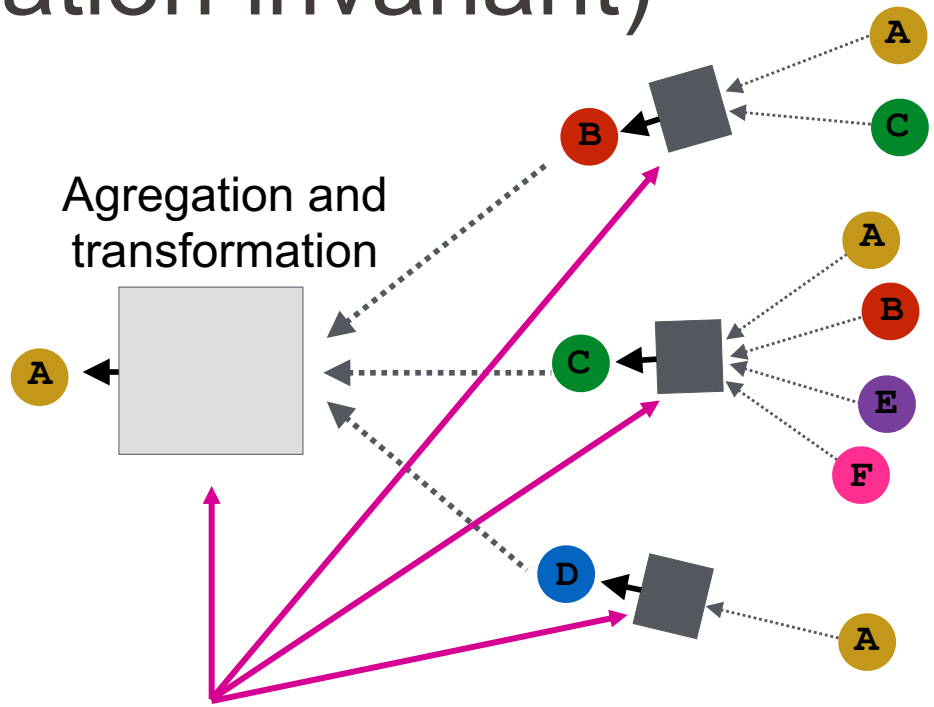


Idea: Aggregate Neighbors

- **Neighborhood aggregation:** Key distinctions are in how different approaches aggregate information across the layers (permutation invariant)



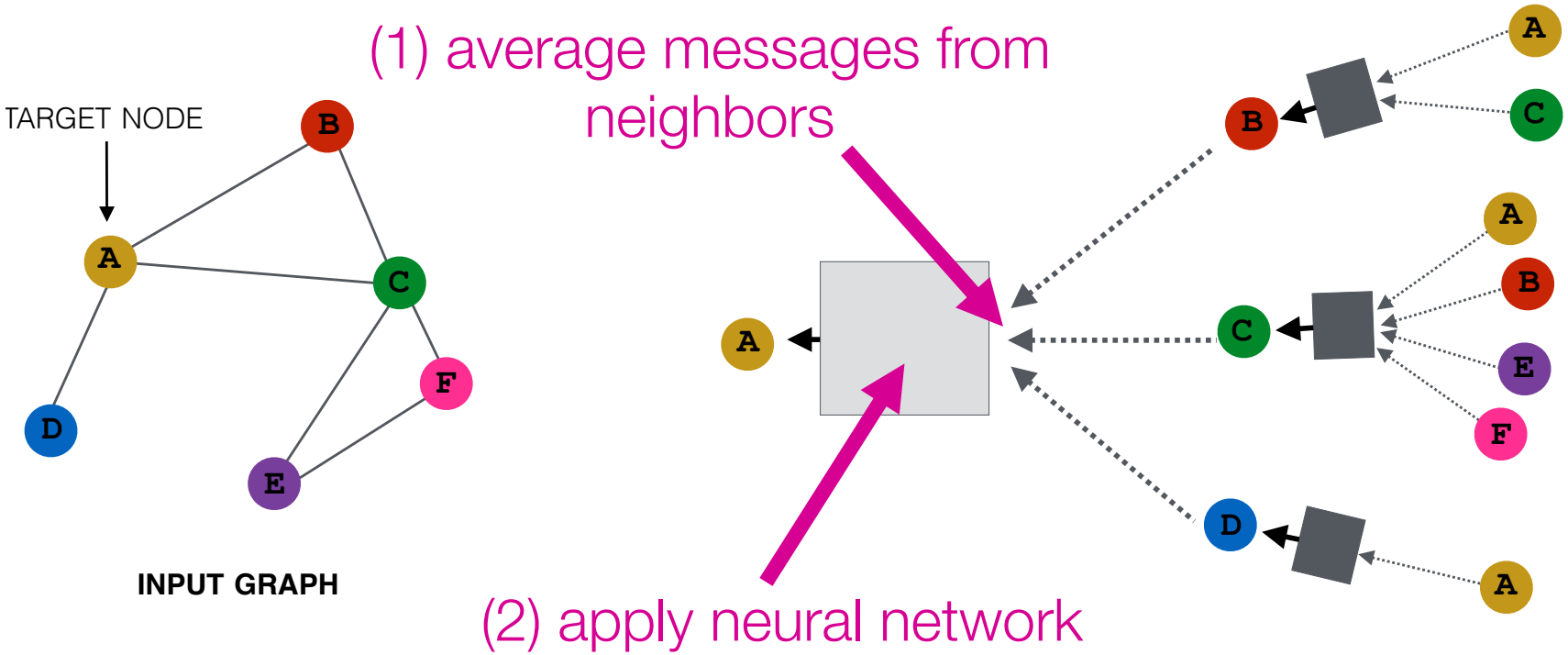
INPUT GRAPH



Neural networks

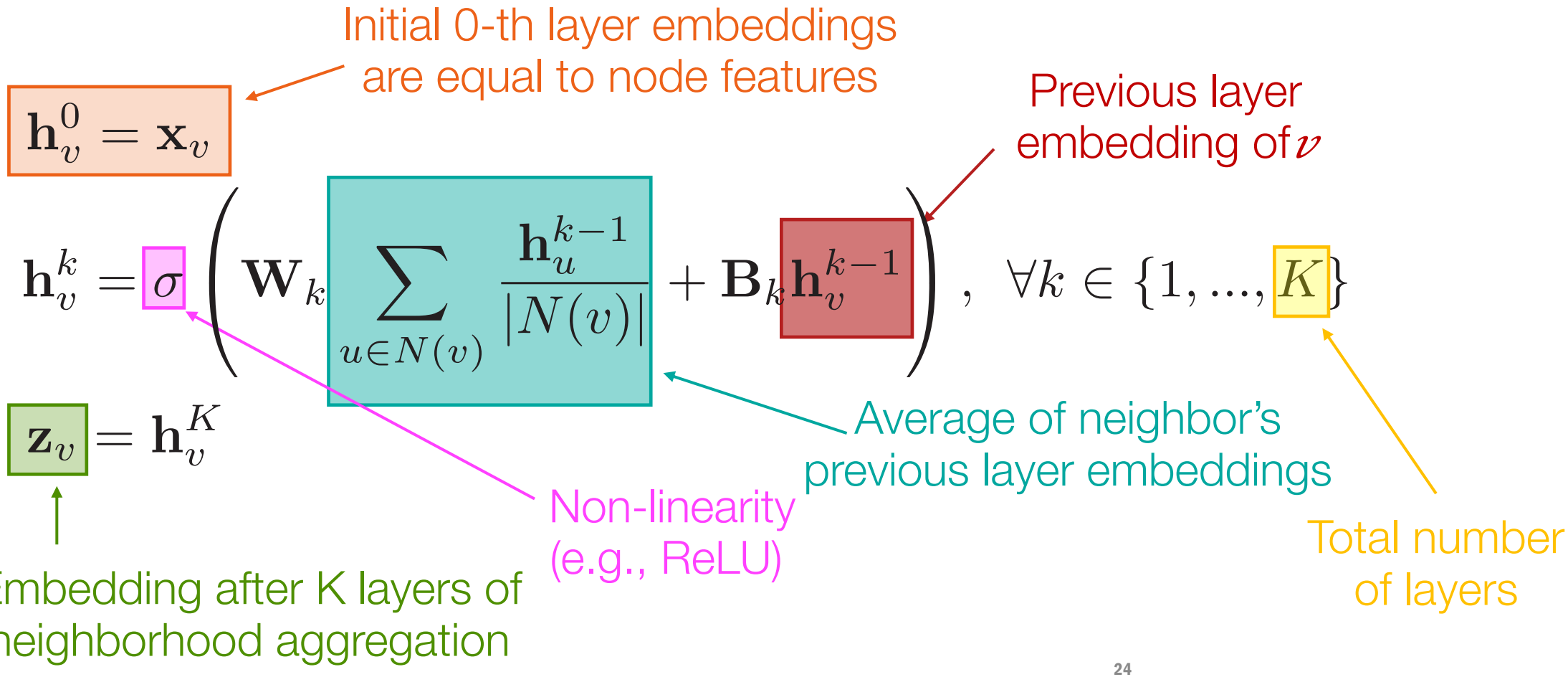
Neighborhood Aggregation

- **Basic approach:** Average information from neighbors and apply a neural network



The Math: Deep Encoder

- **Basic approach:** Average neighbor messages and apply a neural network



Model Parameters

What are our model parameters?

$$\mathbf{h}_v^0 = \mathbf{x}_v$$

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right), \quad \forall k \in \{1, ..., K\}$$

$$\mathbf{z}_v = \mathbf{h}_v^K$$

Model Parameters

trainable weight matrices
(i.e., what we learn)

$$\mathbf{h}_v^0 = \mathbf{x}_v$$

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right), \quad \forall k \in \{1, ..., K\}$$

$$\mathbf{z}_v = \mathbf{h}_v^K$$

We can feed these embeddings into any loss function and run stochastic gradient descent to train the weight parameters

Training the Network

- Unsupervised way
  - Use graph structure as supervision!
- Supervised way:
  - Node-level prediction, edge-level prediction, graph-level prediction
- Loss:
  - Depends on the task

Matrix Formulation

- Many aggregations can be performed efficiently by matrix operations
- Let $H^{(l)} = [h_1^{(l)} \dots h_{|V|}^{(l)}]^{\mathrm{T}}$
- Then: $\sum_{u \in N_v} h_u^{(l)} = \mathrm{A}_{v,:}\mathrm{H}^{(l)}$
- Let D be diagonal matrix where $D_{v,v} = \mathrm{Deg}(v) = |N(v)|$
  - The inverse of D: D^{-1} is also diagonal: $D_{v,v}^{-1} = 1/|N(v)|$
- Therefore,

$$\sum_{u \in N(v)} \frac{h_u^{(l-1)}}{|N(v)|} \longrightarrow H^{(l+1)} = D^{-1}AH^{(l)}$$

Matrix Formulation

- Many aggregations can be performed efficiently by matrix operations
  - Let $H^{(l)} = [h_1^{(l)} \dots h_{|V|}^{(l)}]^{\mathrm{T}}$
  - Then: $\sum_{u \in N_v} h_u^{(l)} = \mathrm{A}_{v,:}\mathrm{H}^{(l)}$
  - Let D be diagonal matrix where $D_{v,v} = \mathrm{Deg}(v) = |N(v)|$
    - The inverse of D: D^{-1} is also diagonal: $D_{v,v}^{-1} = 1/|N(v)|$
  - Therefore,

$$\sum_{u \in N(v)} \frac{h_u^{(l-1)}}{|N(v)|} \longrightarrow H^{(l+1)} = D^{-1}AH^{(l)}$$

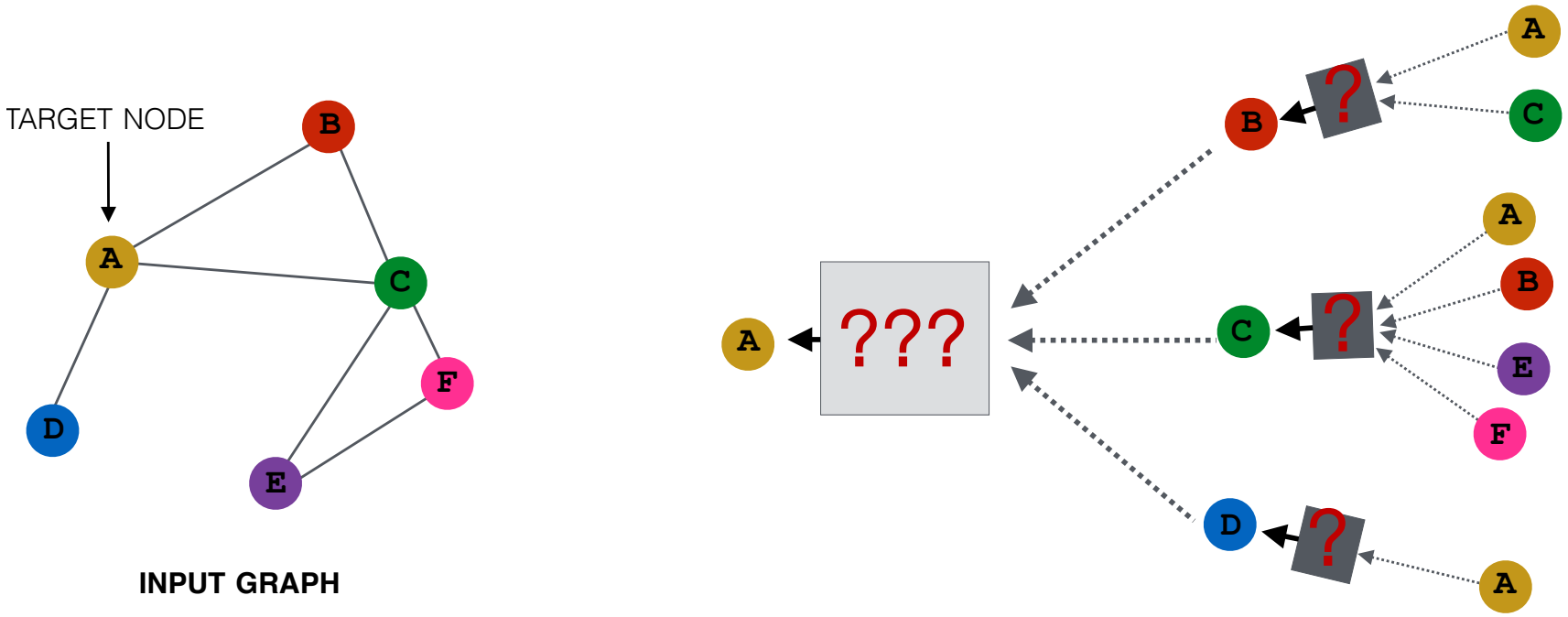
Finally:

$$H^{(l+1)} = \sigma(\tilde{A}H^{(l)}W_l^{\mathrm{T}} + H^{(l)}B_l^{\mathrm{T}})$$
where $\tilde{A} = D^{-1}A$

GraphSAGE

So far we have aggregated the neighbor messages by taking their (weighted) average

Can we do better?



[Hamilton, Ying, Leskovec. NeurIPS 2017]

GraphSAGE: Aggregation

- Generalized form of aggregation

Generalized aggregation

$$\mathrm{h}_v^{(l+1)} = \sigma([\mathrm{W}_l \cdot \mathrm{CONCAT}\left(\mathrm{h}_v^{(l)}, \mathrm{AGG}\left\{\mathrm{h}_u^{(l)}, \forall u \in N(v)\right\}\right)])$$

Transformation after concatenation with self-embedding

- GraphSAGE:
  - Aggregate message from neighbors
  - Concatenate with node itself
  - Transform and apply nonlinearity
- Apply l_2 normalization at every layer

Variants of Aggregation

Mean: Take a weighted average of neighbors

$$\text{AGG} = \sum_{u \in N(v)} \frac{\text{h}_u^{(l)}}{|N(v)|}$$

Pool: Transform neighbor vectors and apply symmetric vector function (e.g.,Mean or Max)

element-wise mean/max

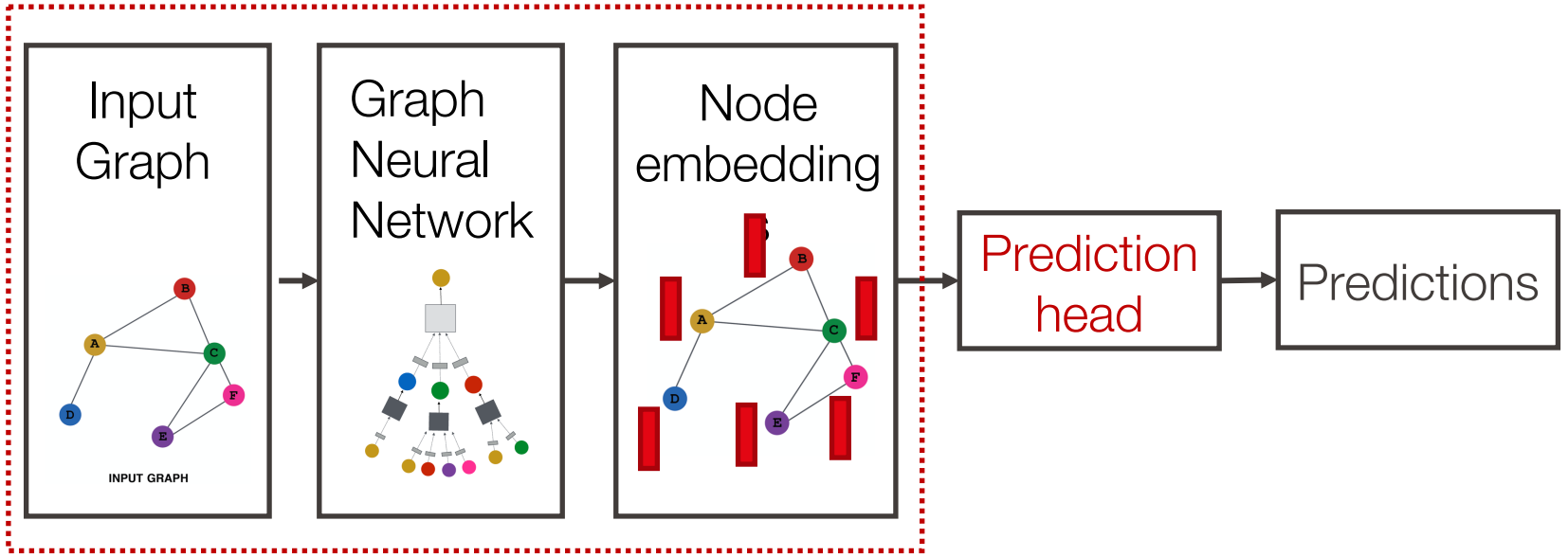
$$\text{AGG} = \gamma(\{\text{MLP}(\text{h}_u^{(l)}), \forall u \in N(v)\})$$

LSTM: Apply LSTM to reshuffled of neighbors

$$\text{AGG} = \text{LSTM}([\text{h}_u^{(l)}, \forall u \in \pi(N(v))])$$

GNN Training Pipeline

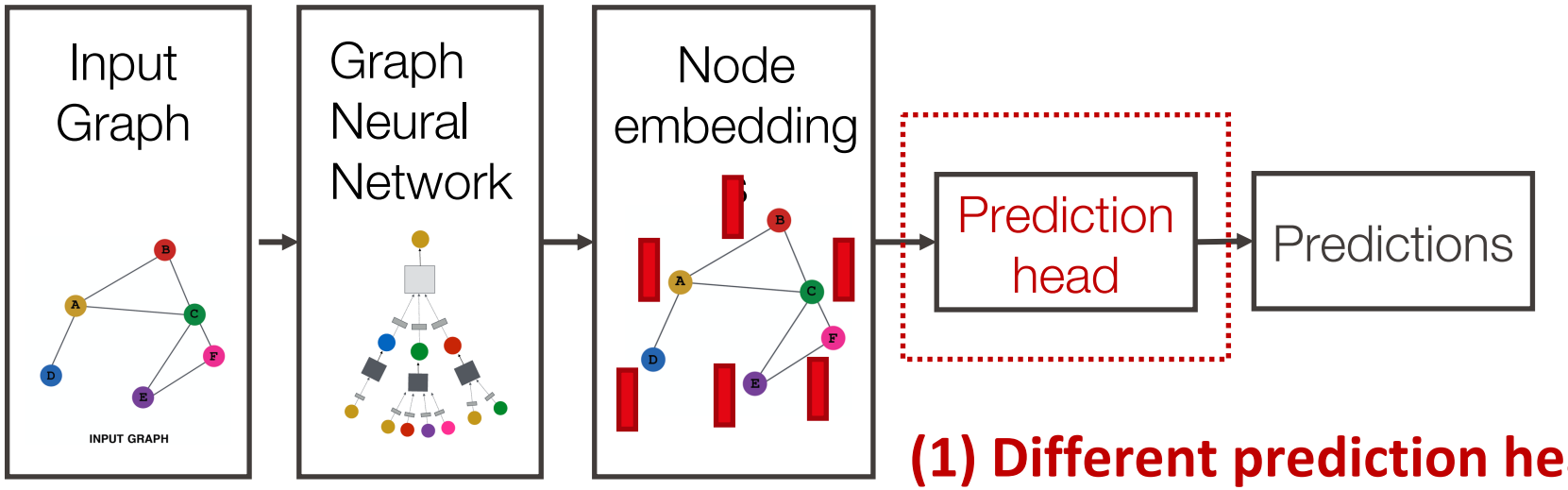
So far what we have covered



Output of a GNN: set of node embeddings

$$\{\mathbf{h}_v^{(L)}, \forall v \in G\}$$

GNN Training Pipeline (1)

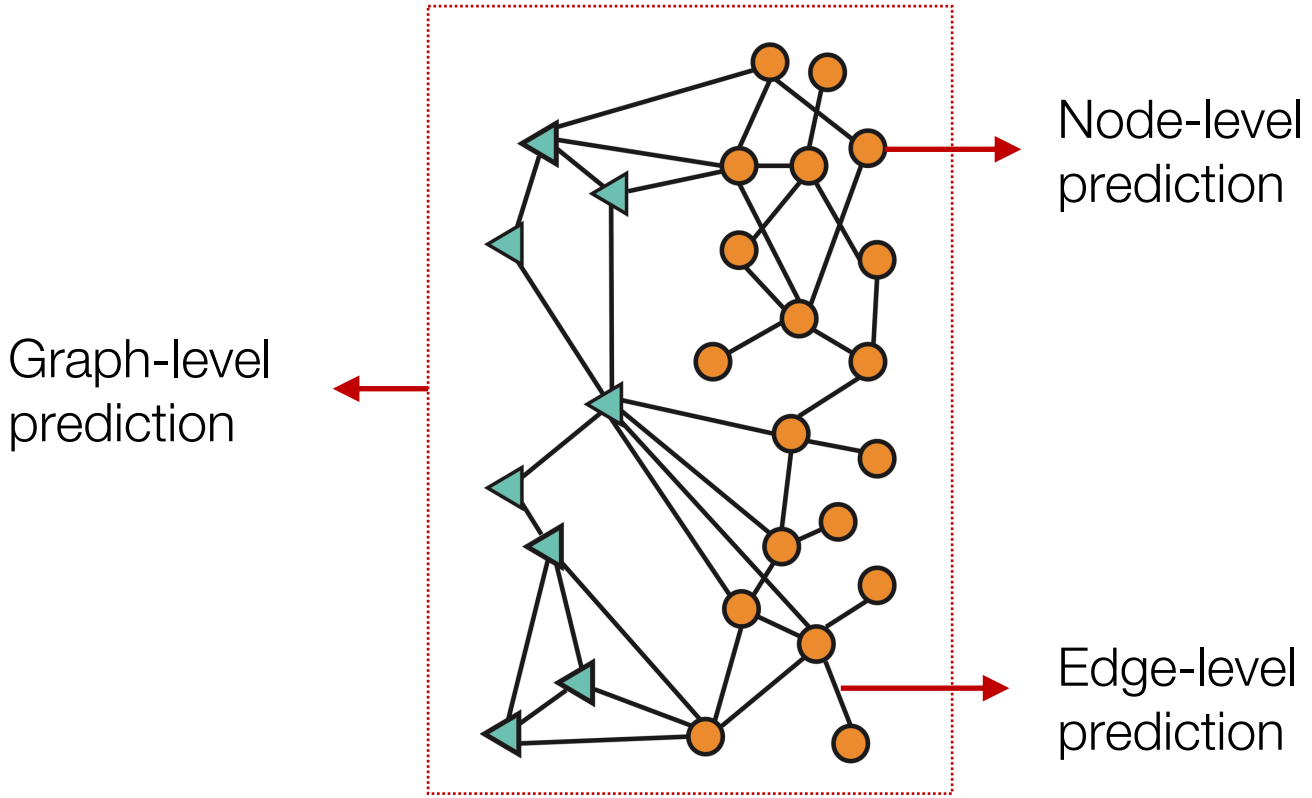


(1) Different prediction heads:

- **Node-level tasks**
- **Edge-level tasks**
- **Graph-level tasks**

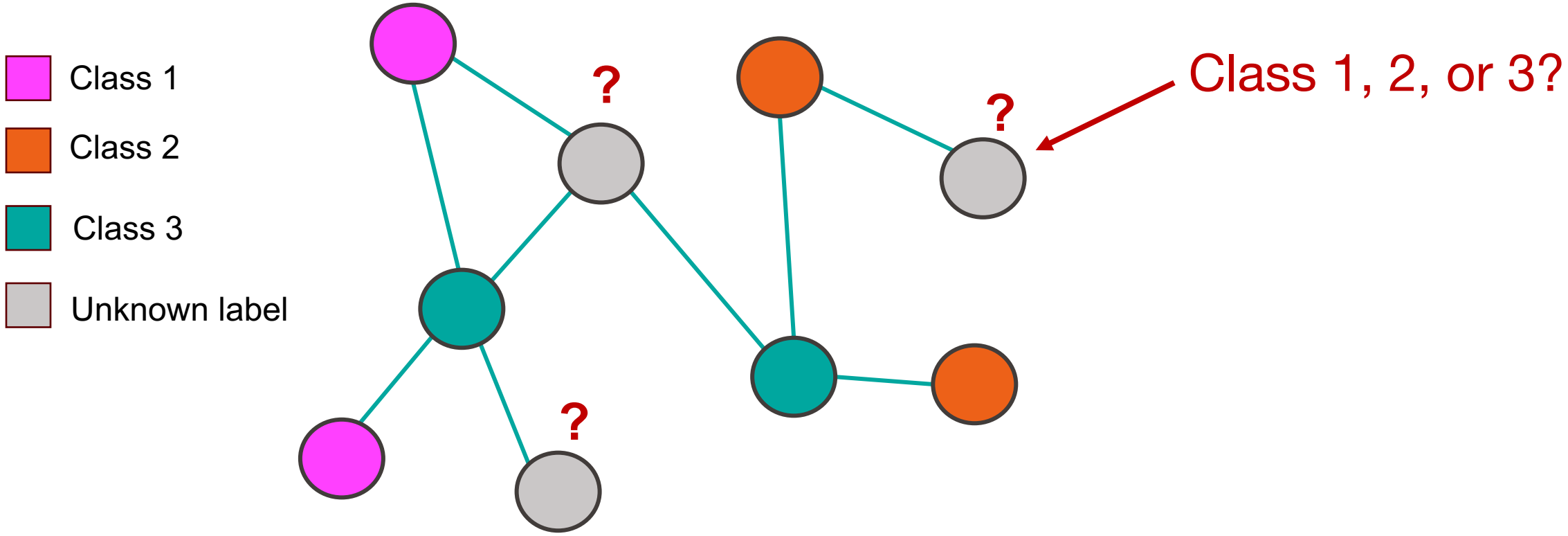
GNN Prediction Heads

- **Idea:** Different task levels require different prediction heads



Node Classification Task

- Task: Given labels of some nodes, predict labels of unlabeled nodes

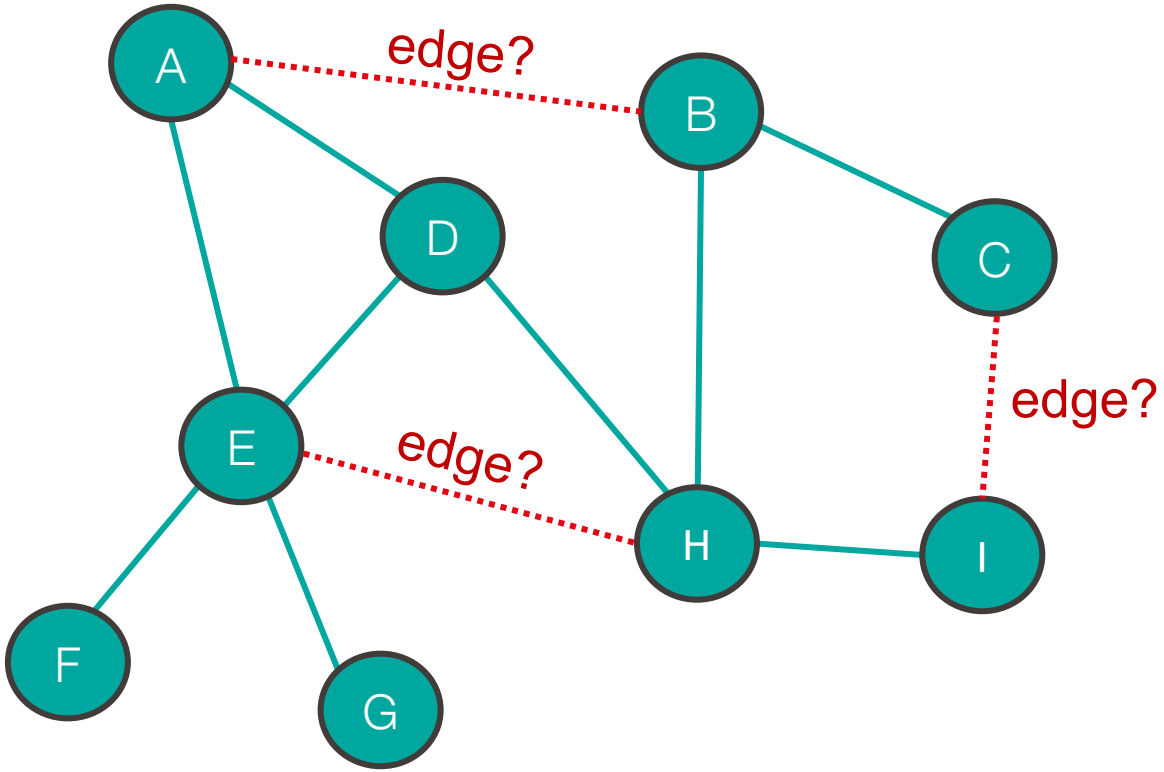


Prediction Heads: Node-level

- **Node-level prediction**: We can directly make prediction using node embeddings!
- After GNN computation, we have d-dim node embeddings: $\{\mathbf{h}_v^{(L)} \in \mathbb{R}^d, \forall v \in G\}$
- Suppose we want to make k-way prediction
  - Classification: classify among k categories
- $\hat{\boldsymbol{y}}_v = \text{Head}_{\text{node}}(\mathbf{h}_v^{(L)}) = \mathbf{W}^{(H)}\mathbf{h}_v^{(L)}$
  - $\mathbf{W}^{(H)} \in \mathbb{R}^{k*d}$: We map node embeddings from $\mathbf{h}_v^{(L)} \in \mathbb{R}^d$ to $\hat{\boldsymbol{y}}_v \in \mathbb{R}^k$ so that we can compute the loss

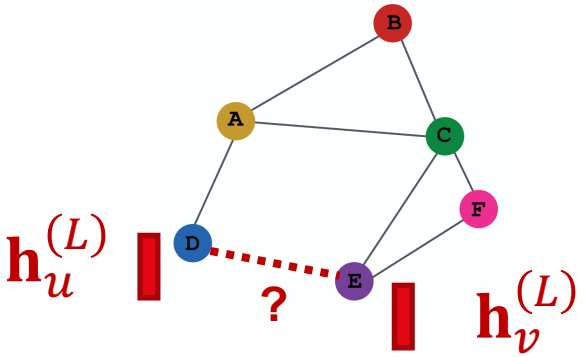
Edge Prediction Task

- Task: Given existing edges in the graph, predict new/missing edges



Prediction Heads: Edge-level

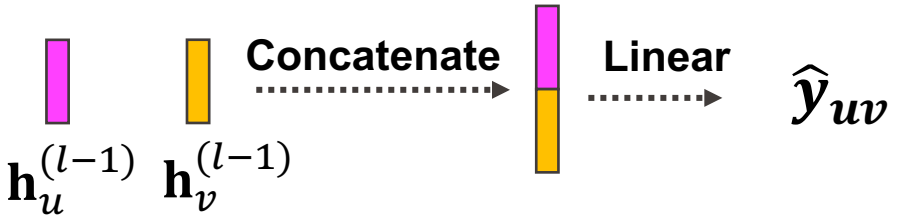
- **Edge-level prediction**: Make prediction using pairs of node embeddings
- Suppose we want to make k-way prediction
- $\hat{\boldsymbol{y}}_{uv} = \text{Head}_{\text{edge}}(\mathbf{h}_u^{(L)}, \mathbf{h}_v^{(L)})$



What are the options for $\text{Head}_{\text{edge}}(\mathbf{h}_u^{(L)}, \mathbf{h}_v^{(L)})$?

Prediction Heads: Edge-level

- **Options for** $\text{Head}_{\text{edge}}(\mathbf{h}_u^{(L)}, \mathbf{h}_v^{(L)})$:
- **(1) Concatenation + Linear**
  - We have seen this in graph attention



  - $\hat{y}_{uv} = \text{Linear}(\text{Concat}(\mathbf{h}_u^{(L)}, \mathbf{h}_v^{(L)}))$
  - Here $\text{Linear}(\cdot)$ will map $2d$-dimensional embeddings (since we concatenated embeddings) to k-dim embeddings (k-way prediction)

What is the dimensionality of linear head?

Prediction Heads: Edge-level

- **Options for** $\text{Head}_{\text{edge}}(\mathbf{h}_u^{(L)}, \mathbf{h}_v^{(L)})$:
- **(2) Dot product**
  - $\hat{y}_{uv} = (\mathbf{h}_u^{(L)})^T \mathbf{h}_v^{(L)}$
  - **This approach only applies to 1-way prediction** (e.g., link prediction: predict the existence of an edge)
  - **Applying to k-way prediction:**
    - $\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(k)}$ trainable

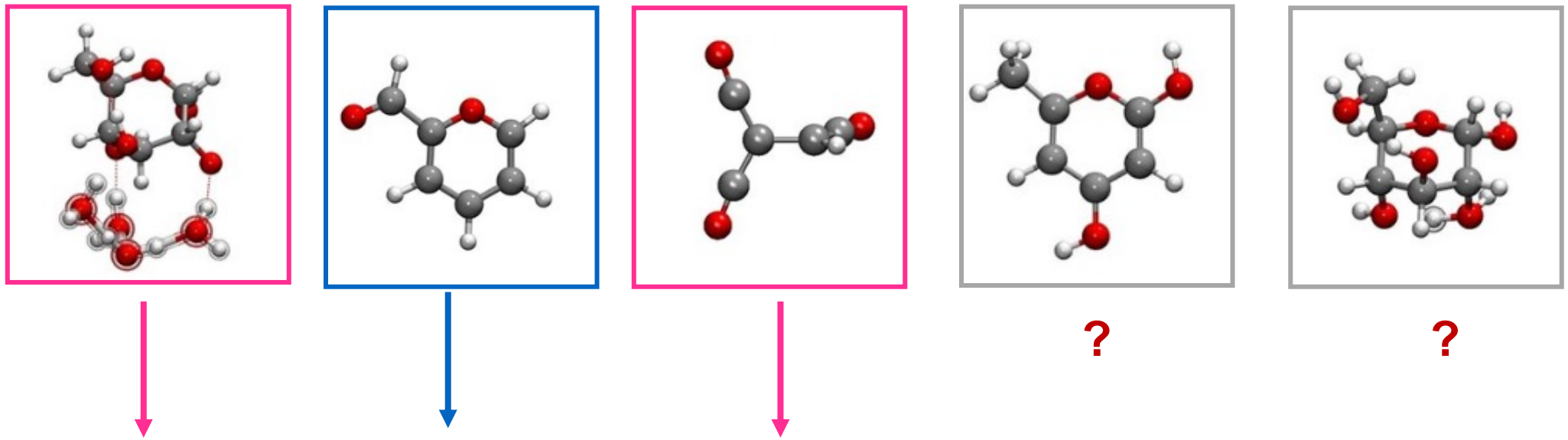
$$\hat{y}_{uv}^{(1)} = (\mathbf{h}_u^{(L)})^T \mathbf{W}^{(1)} \mathbf{h}_v^{(L)}$$

$$\ldots$$

$$\hat{y}_{uv}^{(k)} = (\mathbf{h}_u^{(L)})^T \mathbf{W}^{(k)} \mathbf{h}_v^{(L)}$$

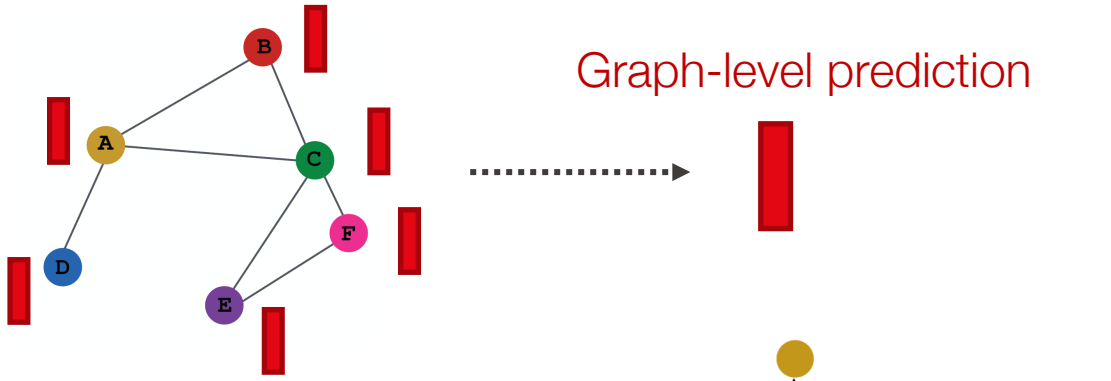
Graph Classification Task

- Task: Given labels of some graphs, predict labels of new graphs

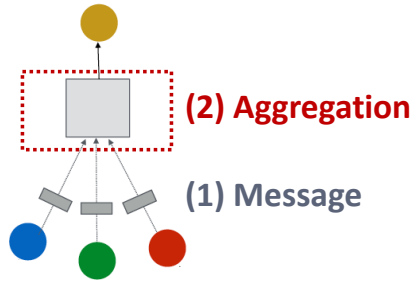


Prediction Heads: Graph-level

- **Graph-level prediction**: Make prediction using all the node embeddings in our graph
- Suppose we want to make k-way prediction
- $\hat{\boldsymbol{y}}_G = \text{Head}_{\text{graph}}(\{\mathbf{h}_v^{(L)} \in \mathbb{R}^d, \forall v \in G\})$



- $\text{Head}_{\text{graph}}(\cdot)$ is similar to $\text{AGG}(\cdot)$ in a GNN layer!



Prediction Heads: Graph-level

- Options for $\text{Head}_{\text{graph}}(\{\mathbf{h}_v^{(L)} \in \mathbb{R}^d, \forall v \in G\})$
- **(1) Global mean pooling**

$$\hat{\boldsymbol{y}}_G = \text{Mean}(\{\mathbf{h}_v^{(L)} \in \mathbb{R}^d, \forall v \in G\})$$

- **(2) Global max pooling**

$$\hat{\boldsymbol{y}}_G = \text{Max}(\{\mathbf{h}_v^{(L)} \in \mathbb{R}^d, \forall v \in G\})$$

- **(3) Global sum pooling**

$$\hat{\boldsymbol{y}}_G = \text{Sum}(\{\mathbf{h}_v^{(L)} \in \mathbb{R}^d, \forall v \in G\})$$

- These options work great for small graphs,but global pooling over large graph looses information
- **Solution:** Hierarchical Pooling [Ying et al. NeurIPS 2018]

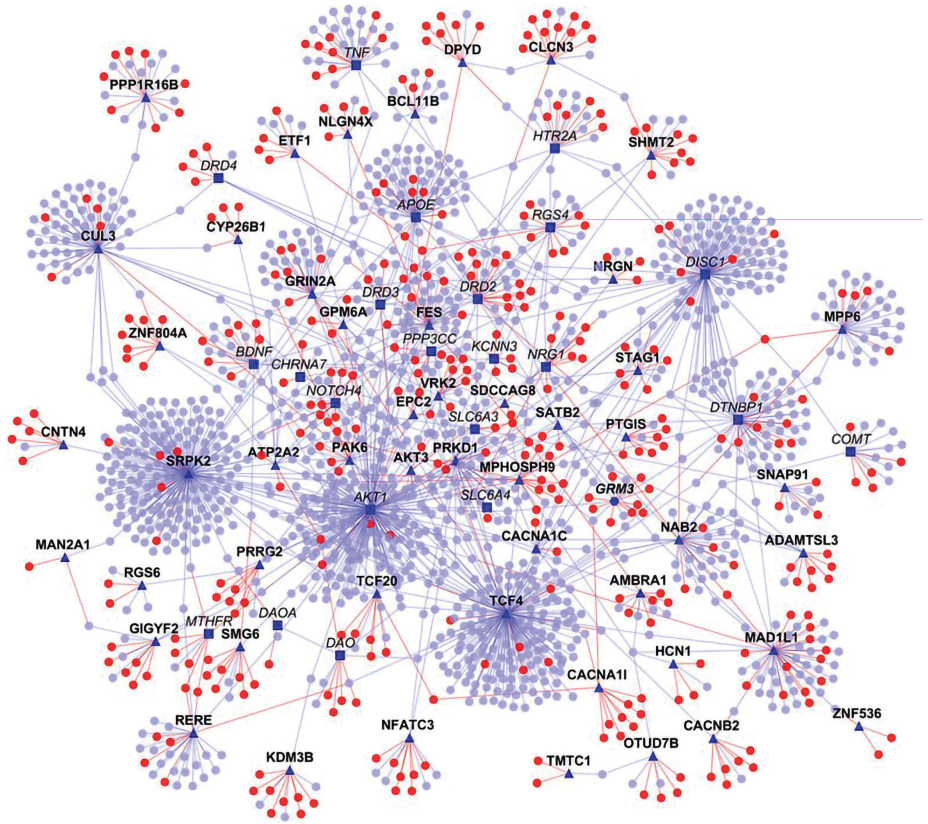
Biomedical Applications

What are examples of graphs in biomedicine?

Graphs are omnipresent in biomedical applications!

Graphs Are Omnipresent in Biomedical Applications

Example: Protein-protein interaction network



- Protein-protein interactions (PPIs) are the physical contacts between two or more proteins
- Proteins rarely act alone as their functions tend to be regulated
- Proteins involved in a certain cellular pathway or biological process are often found to interact with each other

Graphs Are Omnipresent in Biomedical Applications

Example: Disease network

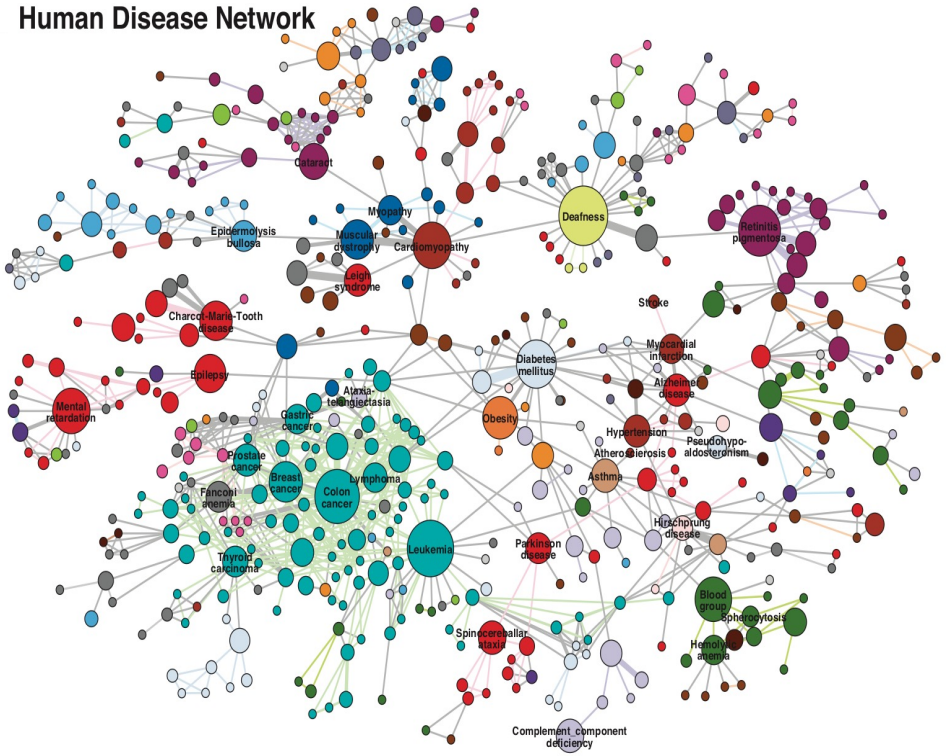


Figure from Goh et al. PNAS '07

- Genes associated with similar diseases are likely to interact and have similar expression profiles
- Two diseases are connected to each other if they share at least one gene in which mutations are associated with both disorders

Graphs Are Omnipresent in Biomedical Applications

Example: Disease-gene network

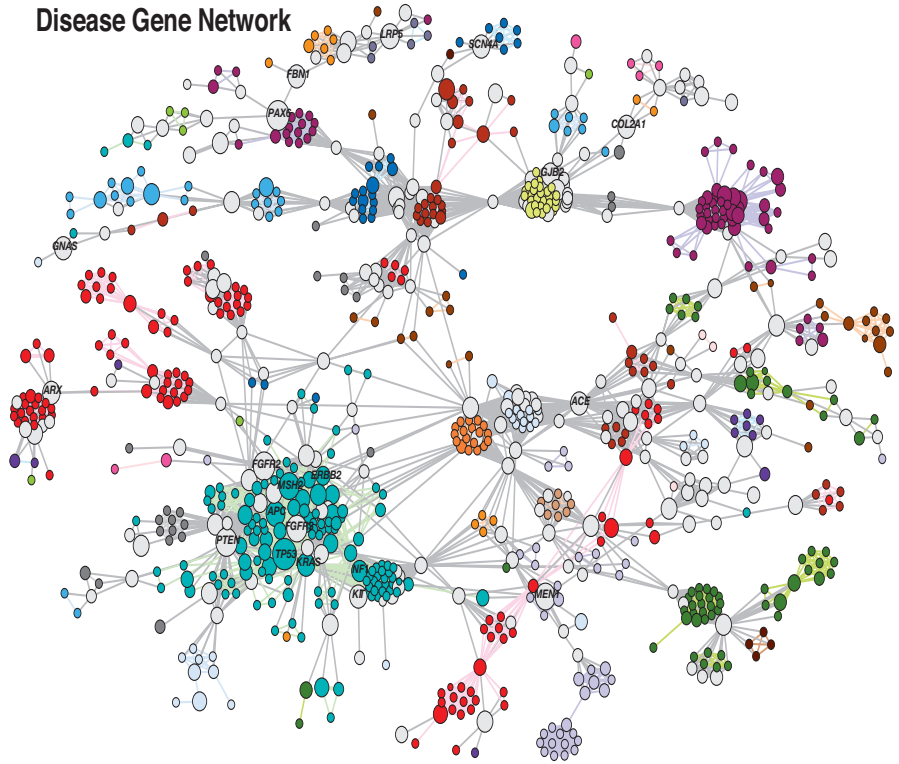


Figure from Goh et al. PNAS '07

- Two diseases are connected to each other if they share at least one gene in which mutations are associated with both disorders
- Two different node types:
  - Disease nodes
  - Gene nodes
- Edge:
  - Gene is associated with a particular disease

Graphs Are Omnipresent in Biomedical Applications

Example: Ecological networks

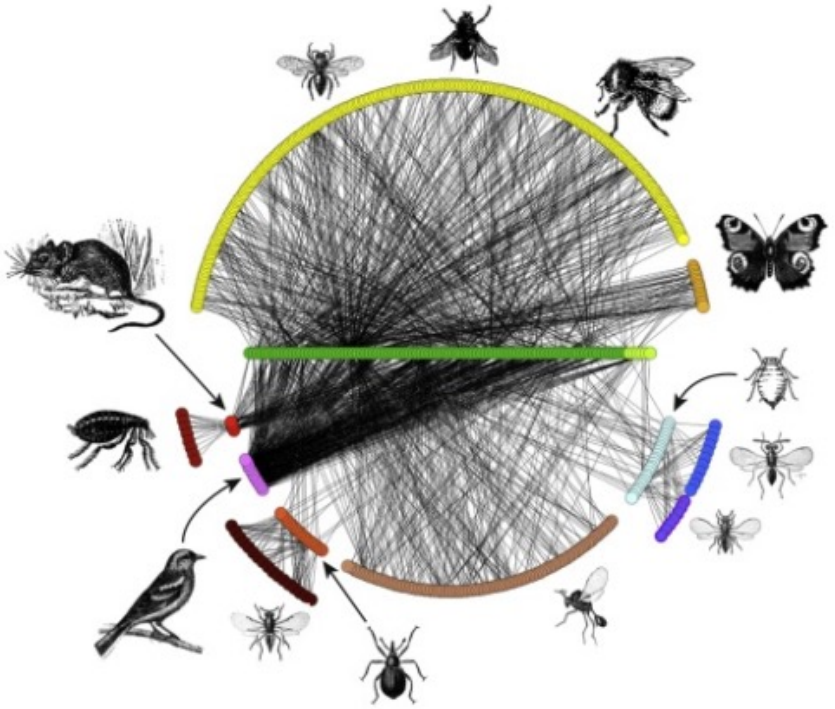
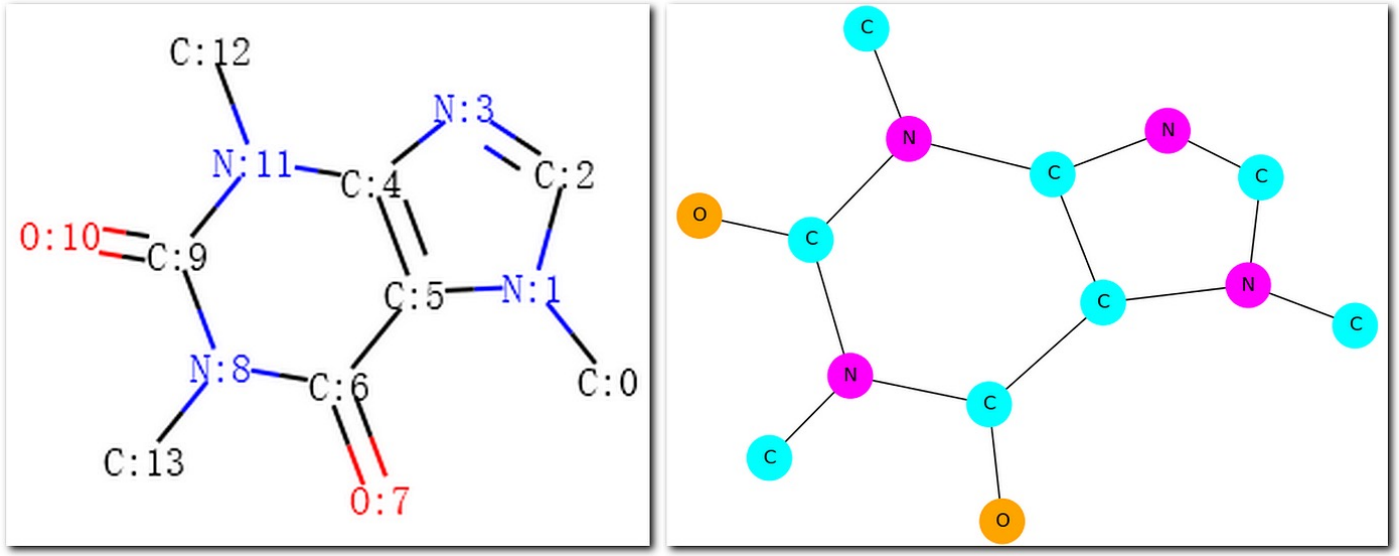


Figure from The QUINTESSENCE Consortium

- Species interaction networks
- Nodes:
  - Species
- Edges
  - Who-eats-who food webs
  - Mutually beneficial interactions

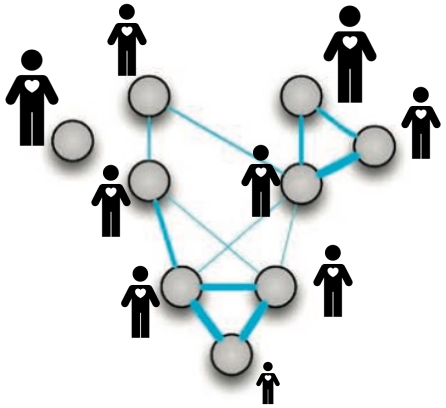
Graphs Are Omnipresent in Biomedical Applications

Example: Molecules

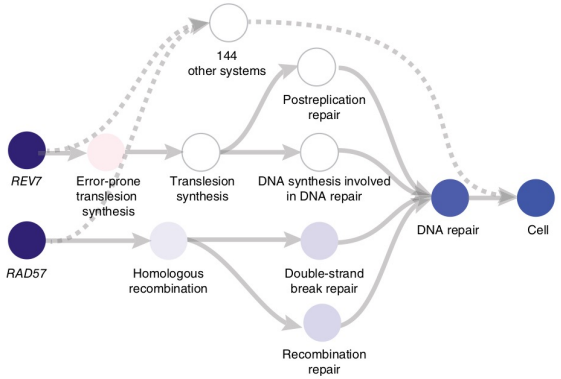


- Molecular graphs
- Nodes:
  - Atoms
- Edges
  - Bonds between atoms

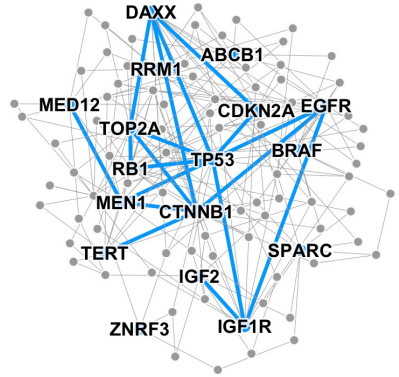
Graphs Are Omnipresent in Biomedical Applications



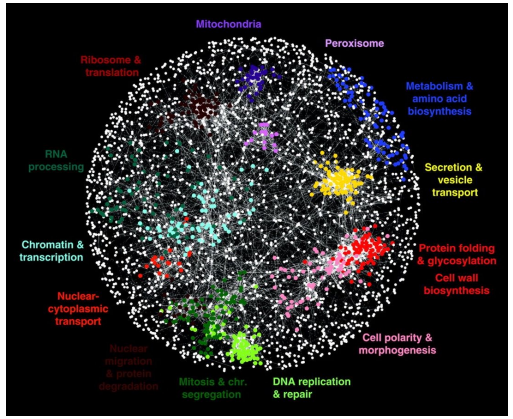
Patient networks



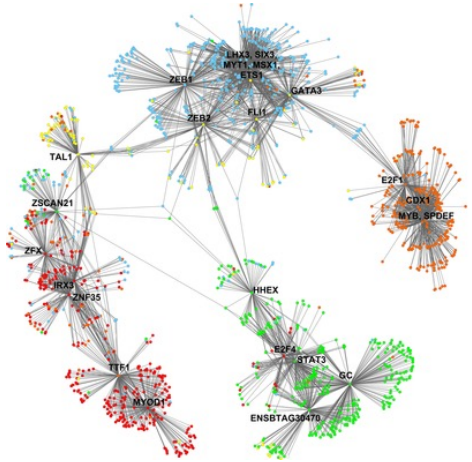
Hierarchies of cell systems



Disease pathways



Genetic interaction networks



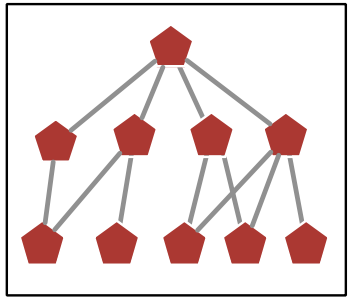
Gene co-expression networks

Cell-cell similarity networks

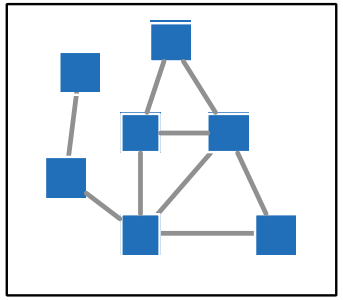
Heterogenous Graphs: Example

- Different node and edge types

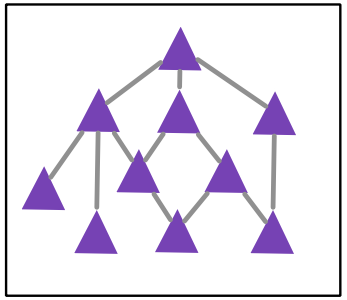
DISEASES



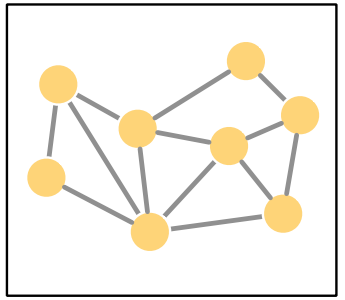
DRUGS



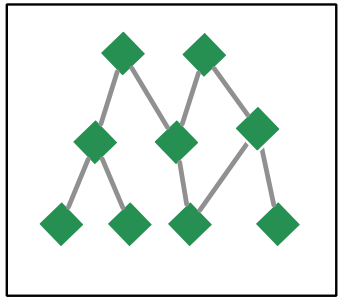
PATHWAYS



PROTEINS

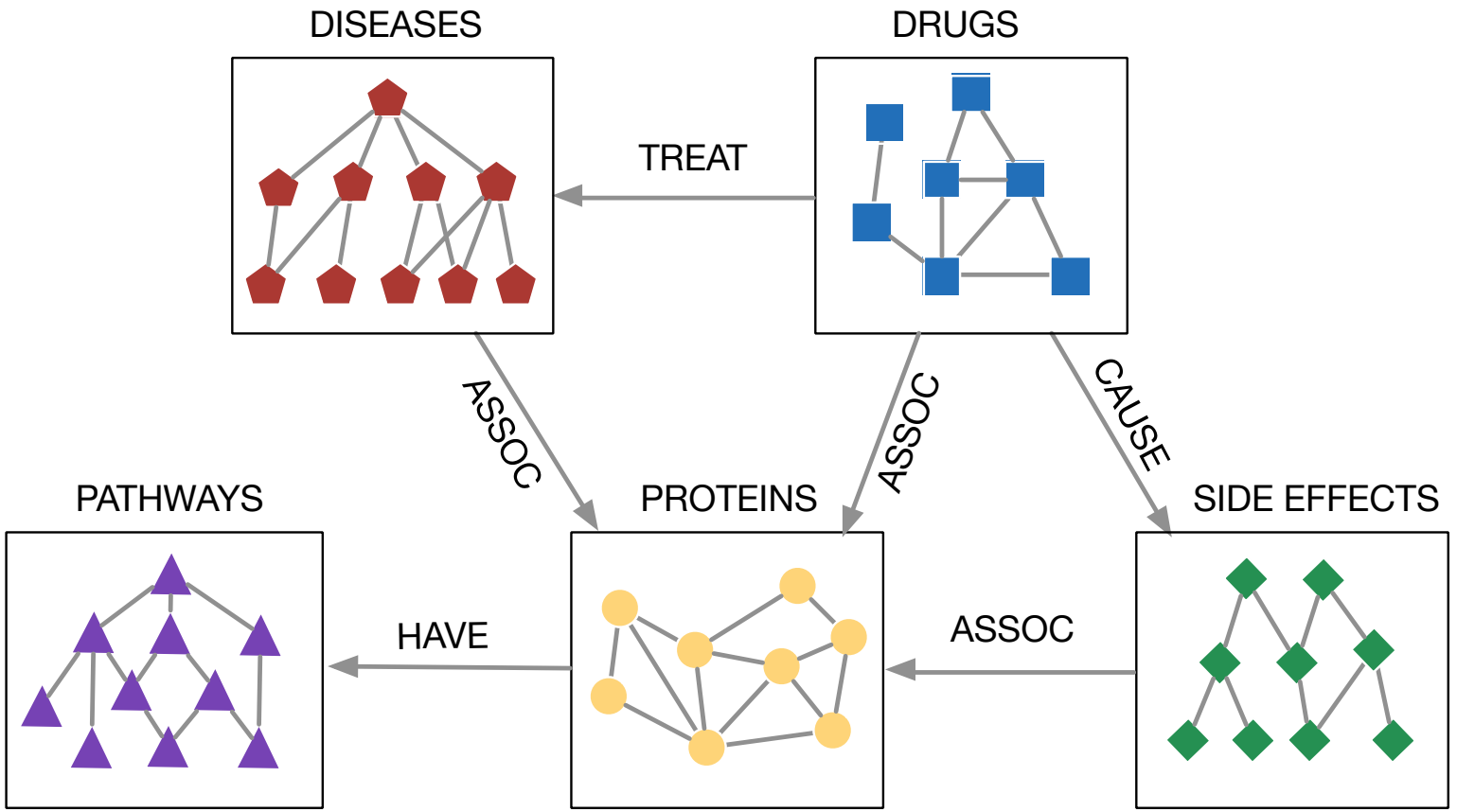


SIDE EFFECTS



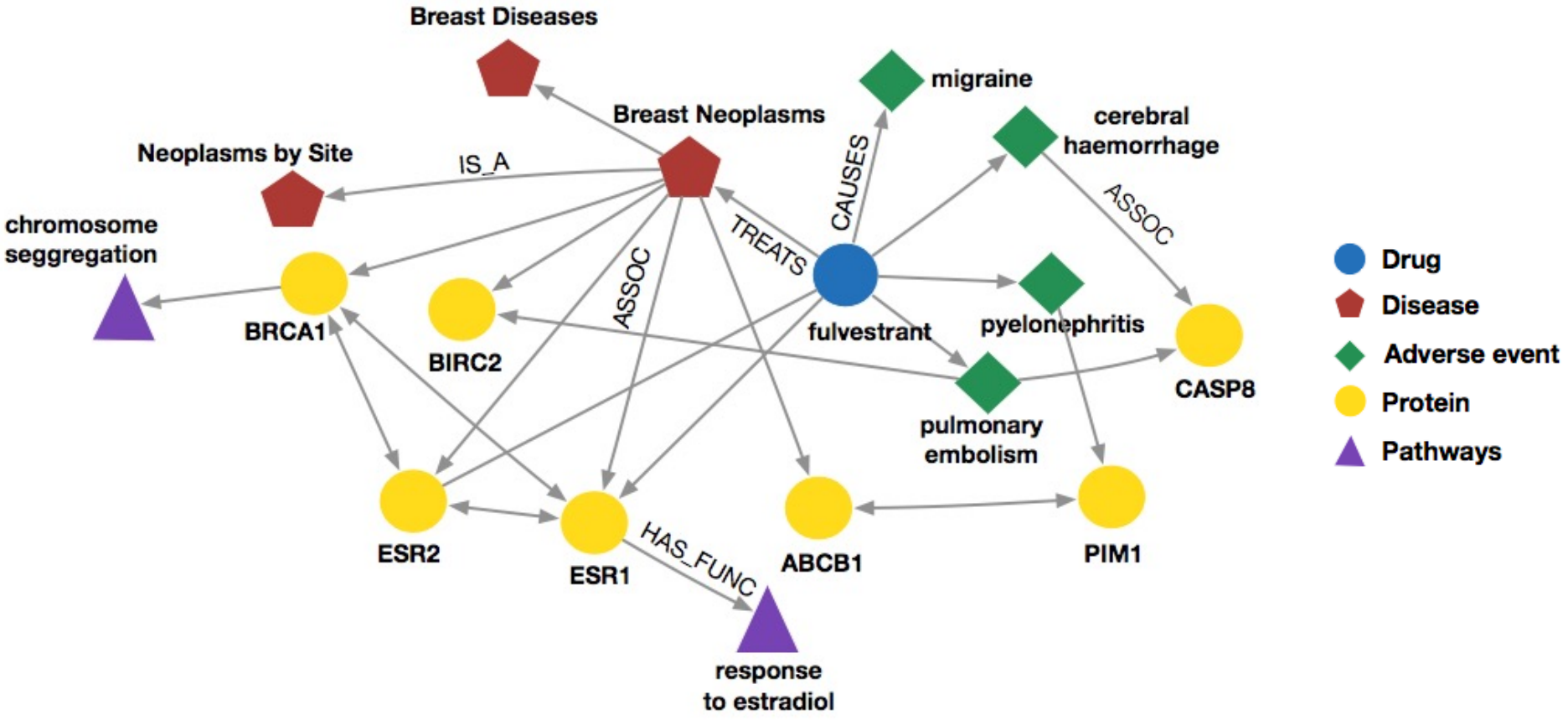
Heterogenous Graphs: Example

- Different node and edge types

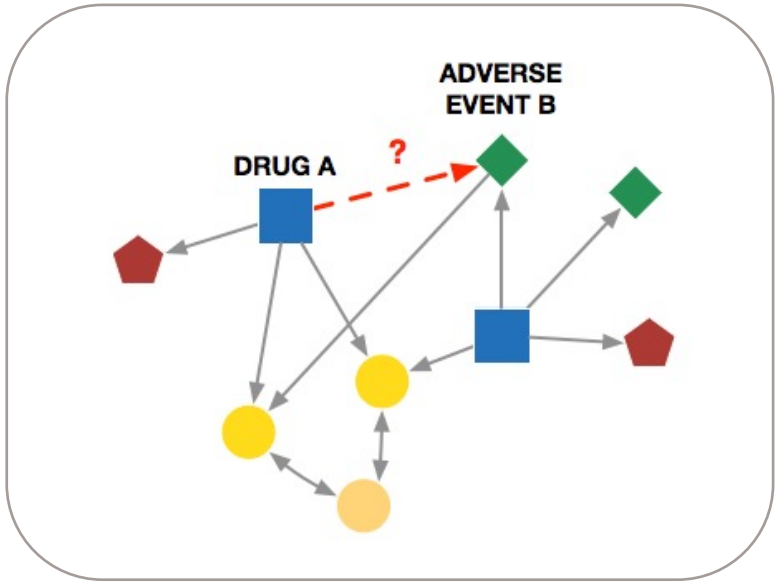


Heterogenous Graphs: Example

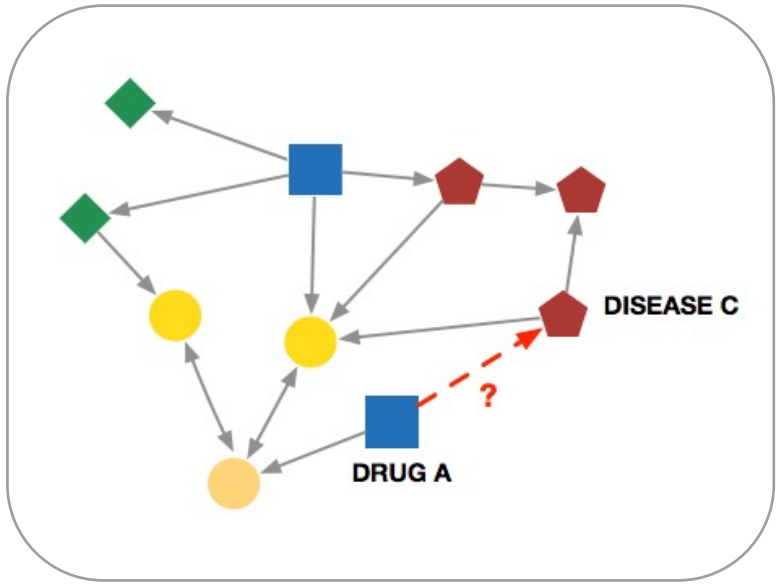
- Different node and edge types



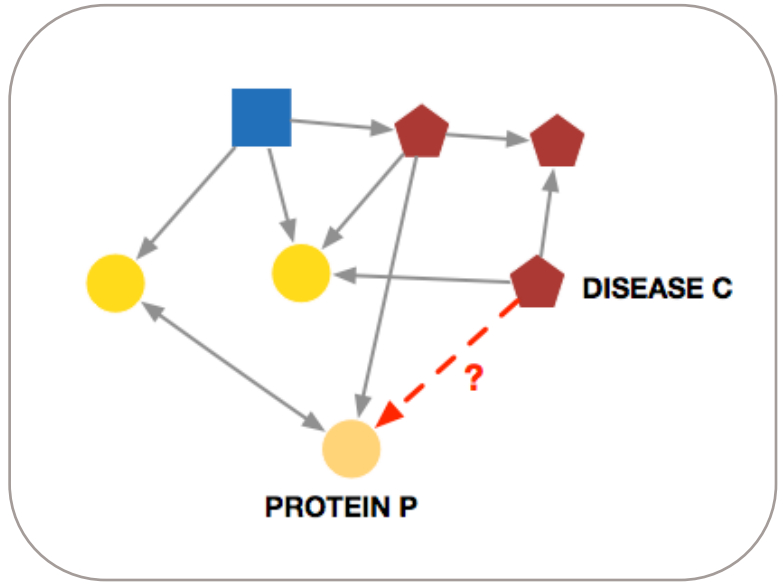
Heterogenous Graphs: Example



Can drug A cause adverse event B?

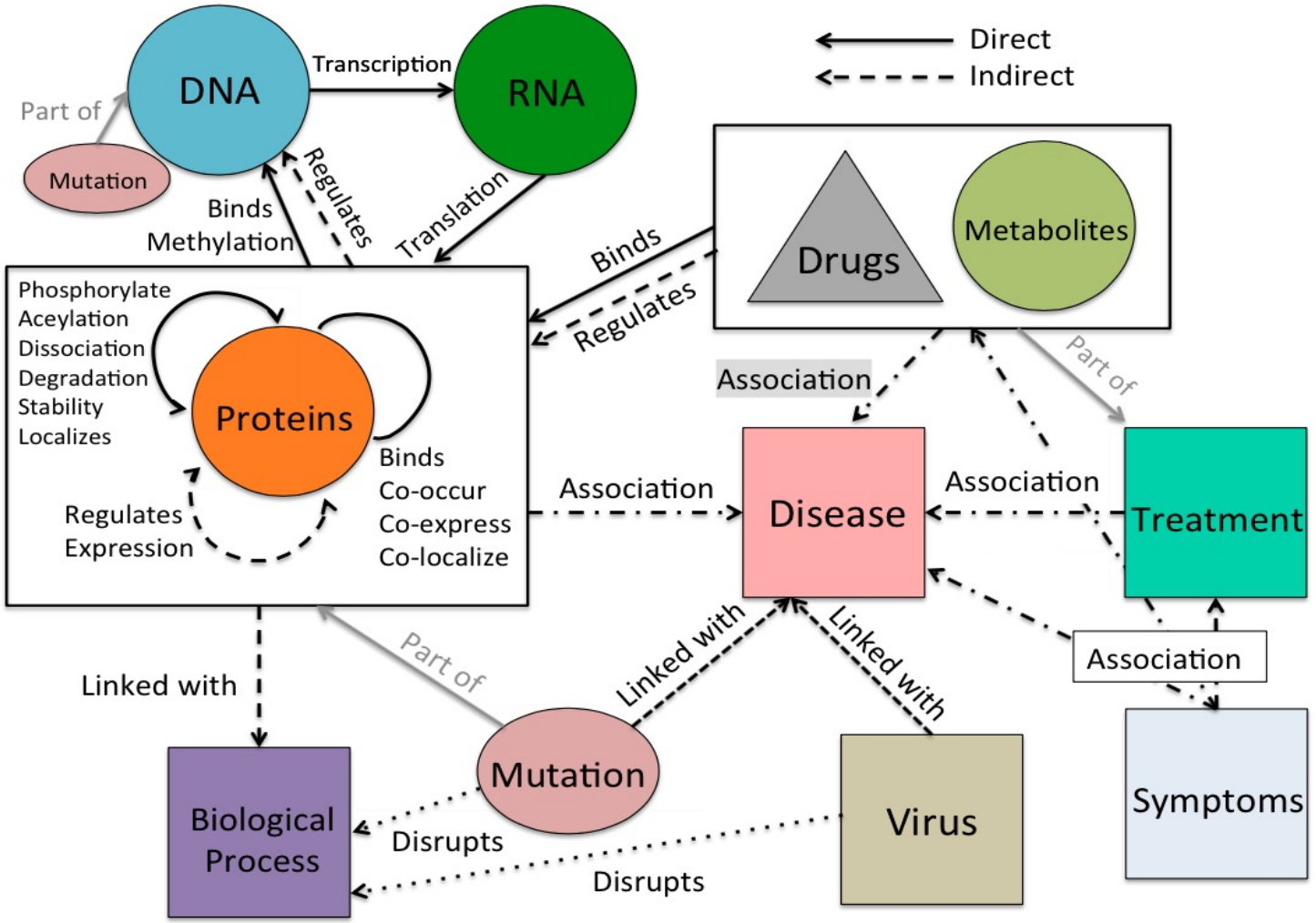


Can drug A treat disease C?



Is disease C associated with protein P?

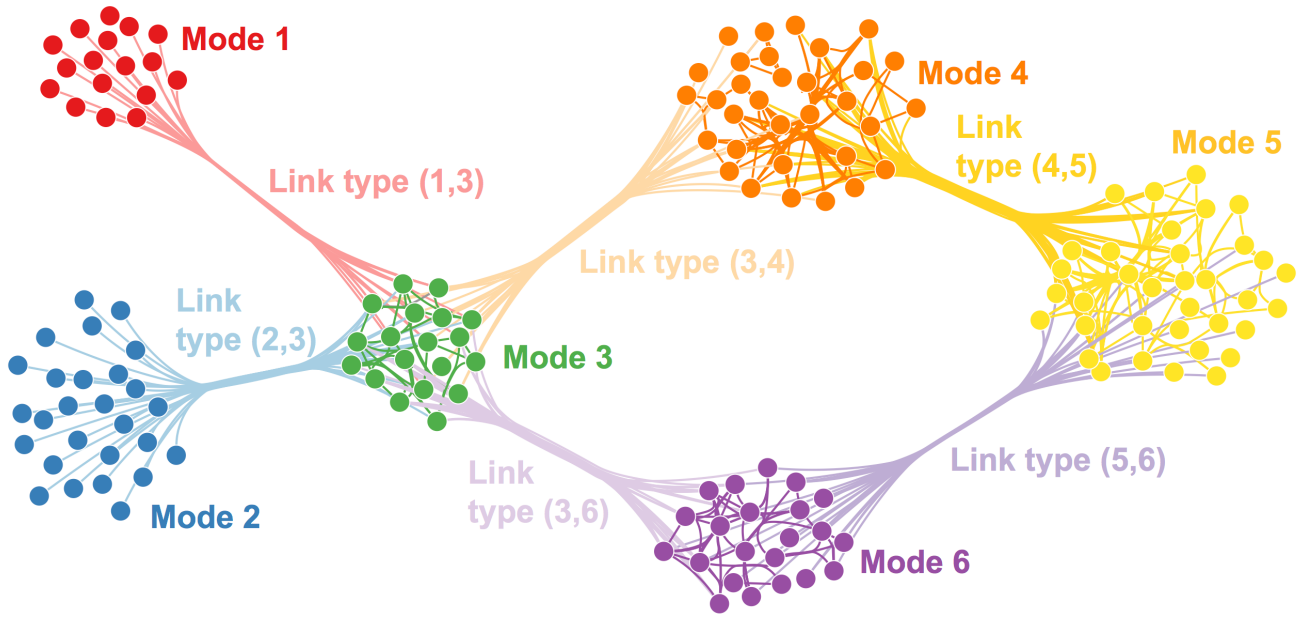
Many Het Nets in Biology



Homogeneous Nets

So far we have embedded homogeneous networks

Can we embed heterogeneous networks?



Modeling polypharmacy side effects

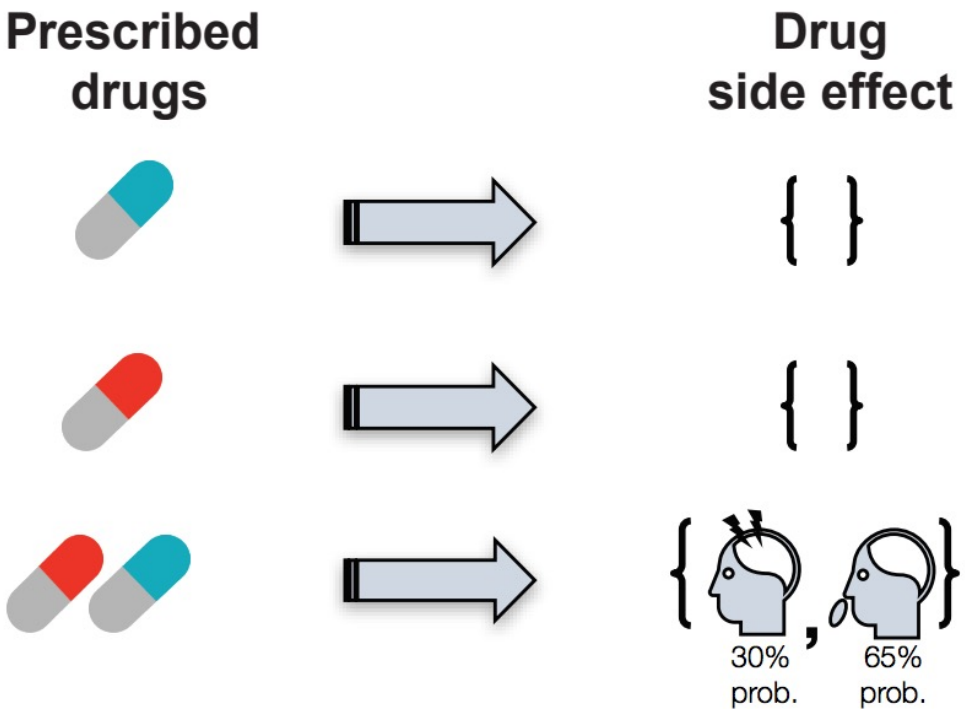
Based on:

- Zitnik et al. Modeling polypharmacy side effects with graph convolutional network. *Bioinformatics* ’18
- ISMB tutorial: Deep Learning for Network Biology

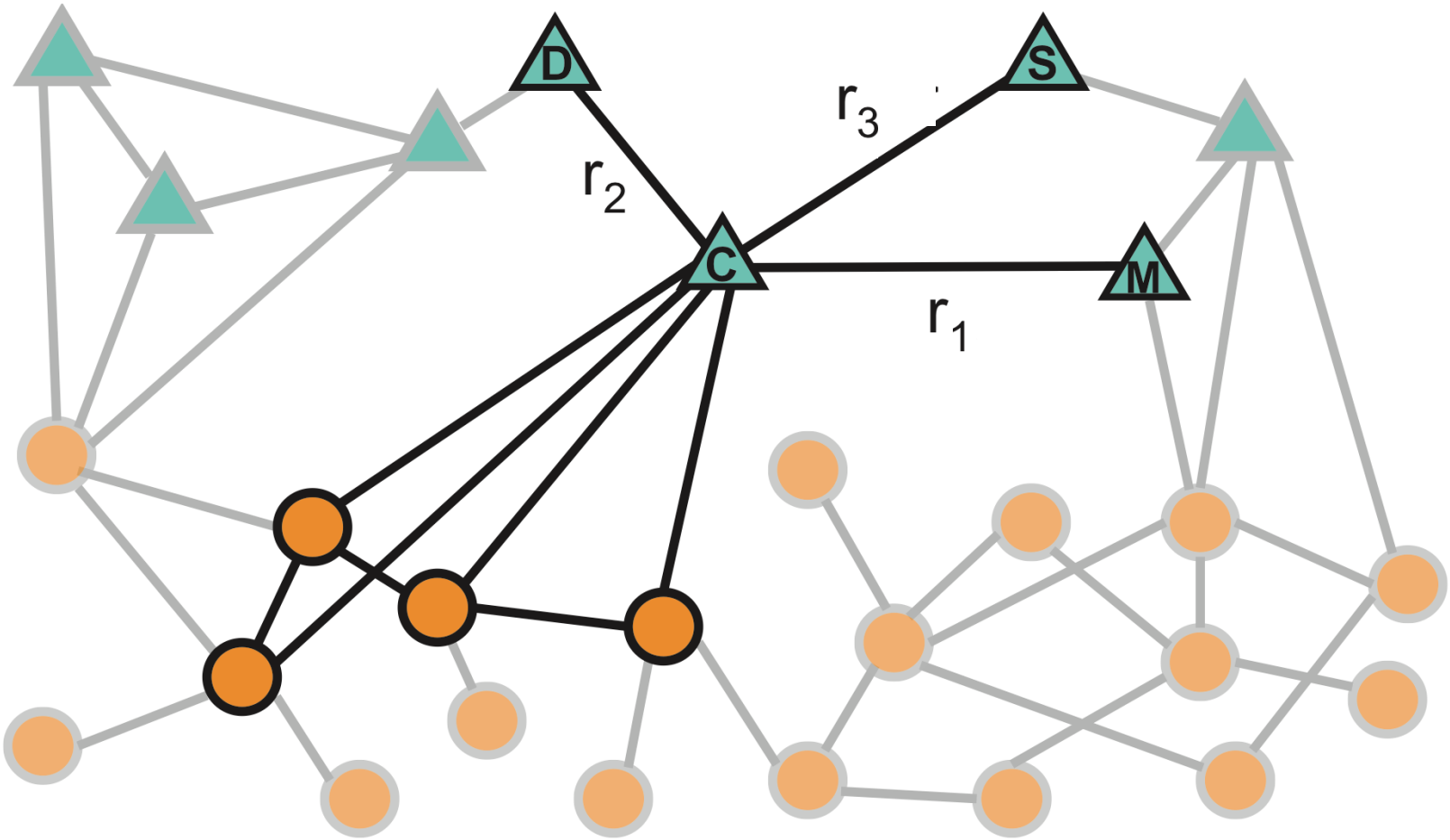
Applications: Biology

Modeling polypharmacy side effects

Side effects that result from combinations of drugs



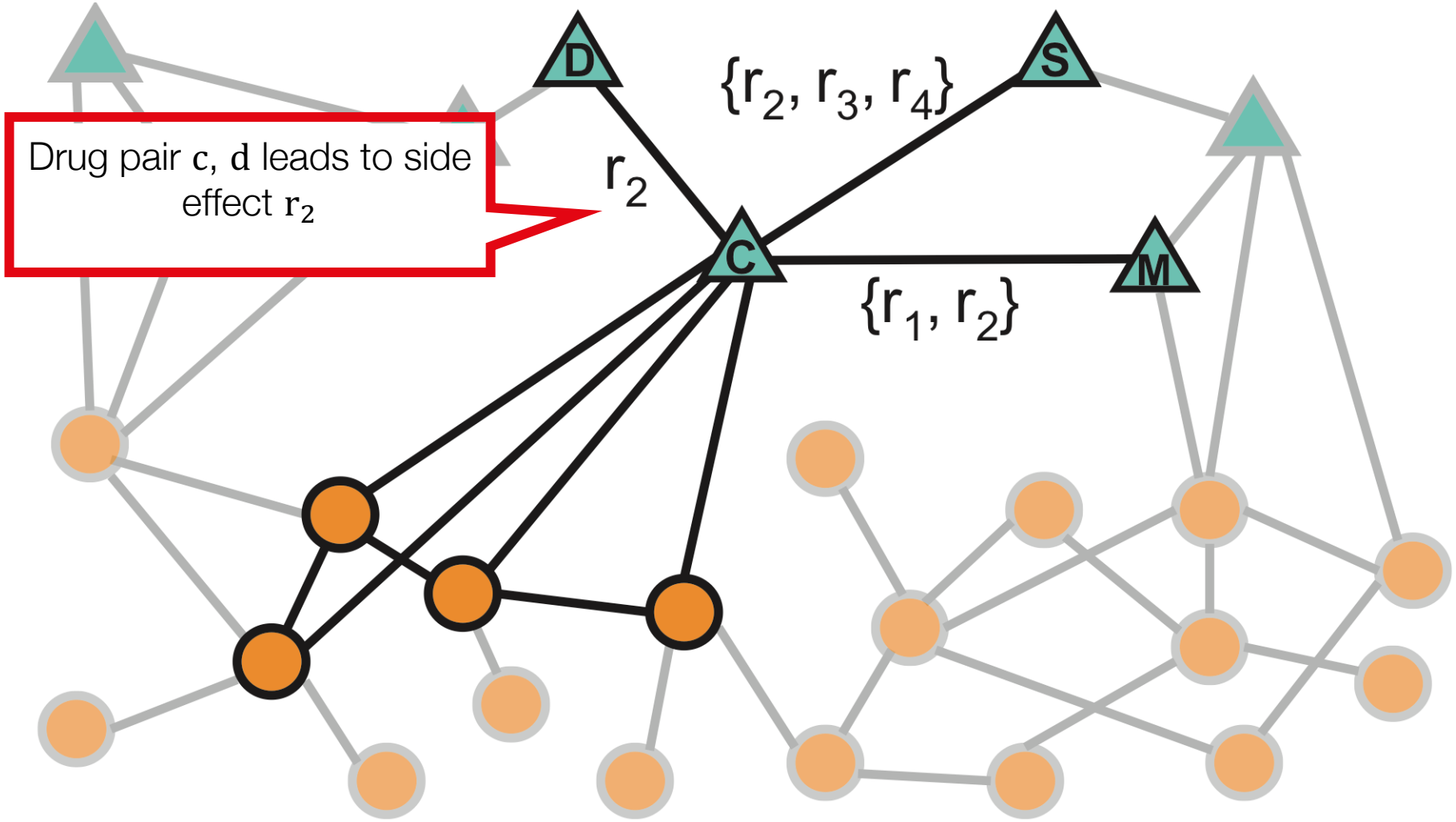
Example: Het Net



r_1 Gastrointestinal bleed side effect
r_2 Bradycardia side effect
r_3 Nausea side effect
r_4 Mumps side effect

Drug-protein interaction
Protein-protein interaction

Running Het Net Example



r_1 Gastrointestinal bleed side effect
r_2 Bradycardia side effect
r_3 Nausea side effect
r_4 Mumps side effect
Drug-protein interaction
Protein-protein interaction

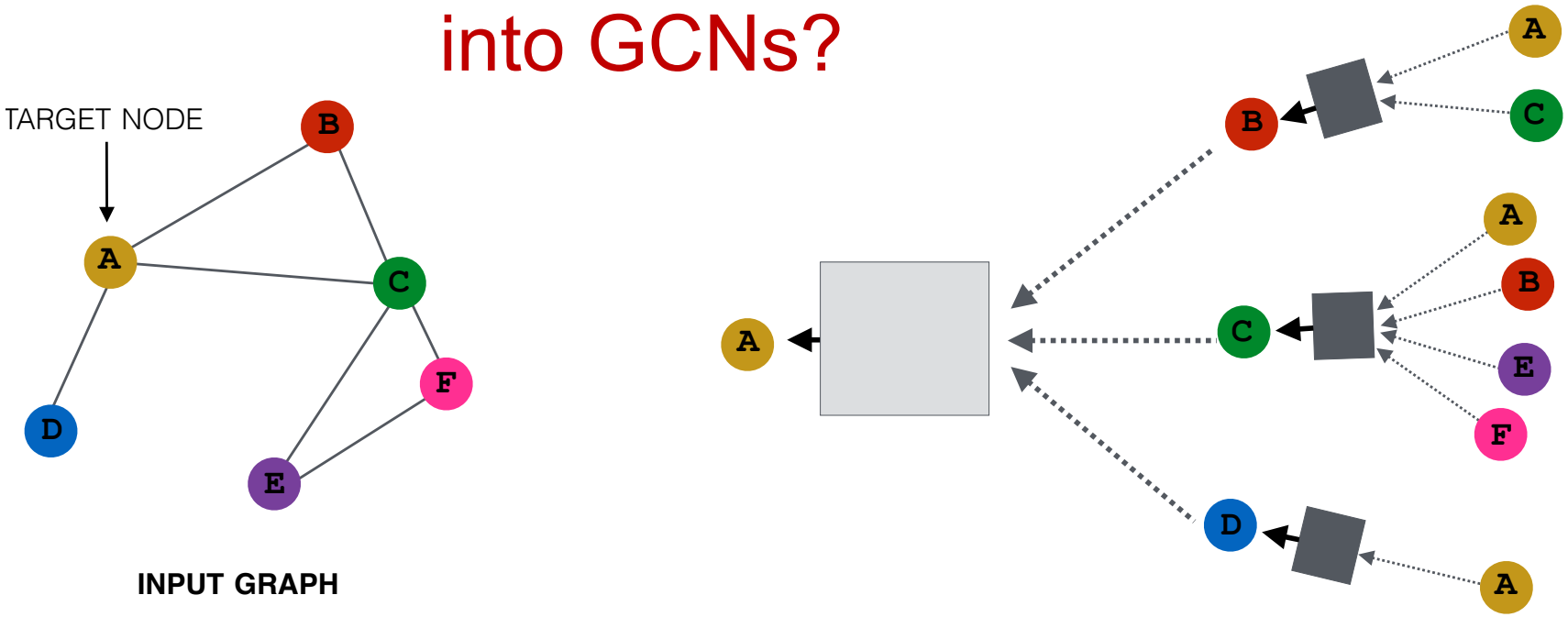
Setup

- Assume we have a graph G:
  - V_t is the vertex set for node type t
  - $\boldsymbol{A}_r$ is the adjacency matrix for edge type r
  - $\mathbf{X}_t \in \mathbb{R}^{m_t \times |V_t|}$ is a matrix of features for nodes of type t
    - Biologically meaningful node features:
      - E.g., immunological signatures, gene expression profiles, gene functional information
    - No features:
      - Indicator vectors (one-hot encoding of a node)

Idea: Aggregate Neighbors

Key idea: Generate node embeddings based on network neighborhoods separated by edge type

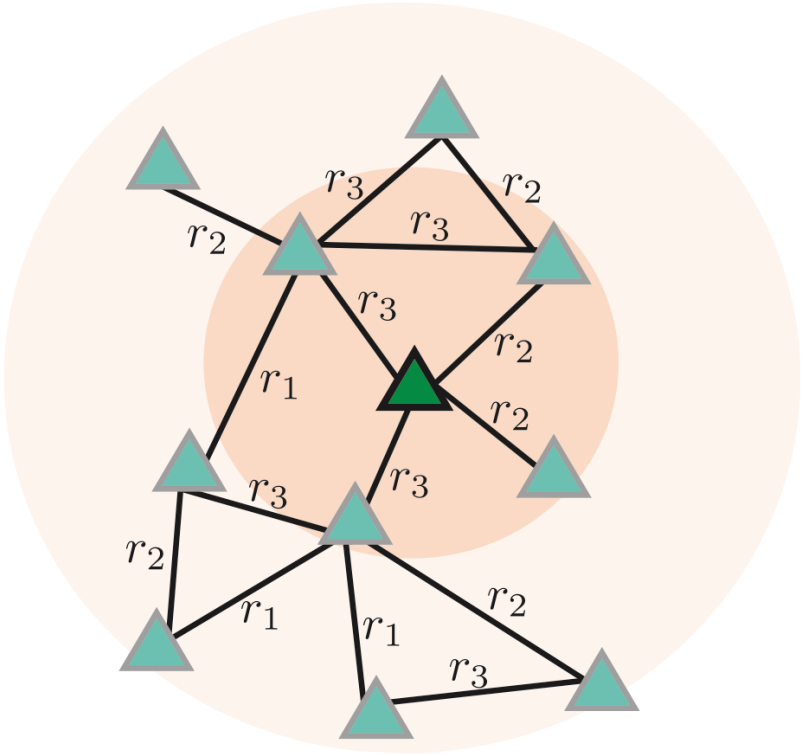
How do we incorporate edge type information into GCNs?



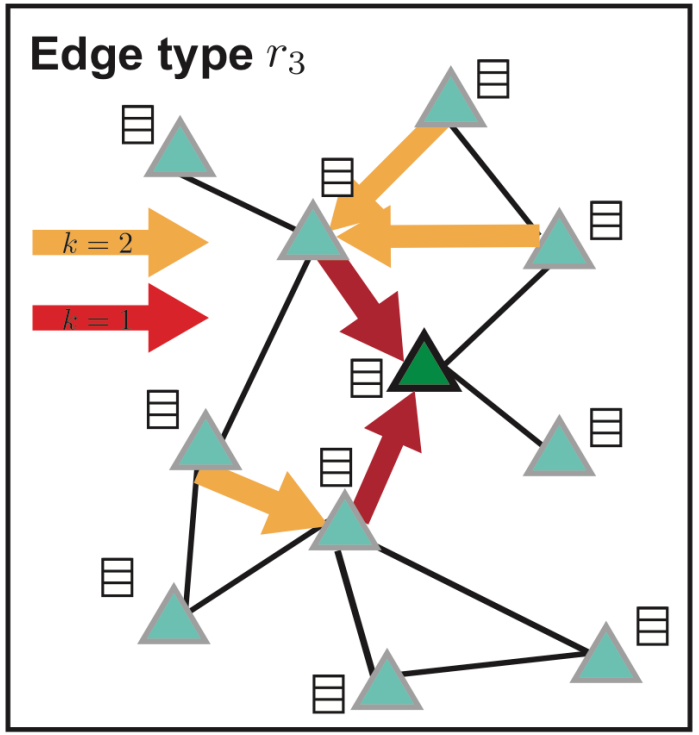
Idea: Aggregate Neighbors

Each edge type is **modeled separately**

A node's neighborhood defines a **computation graph**

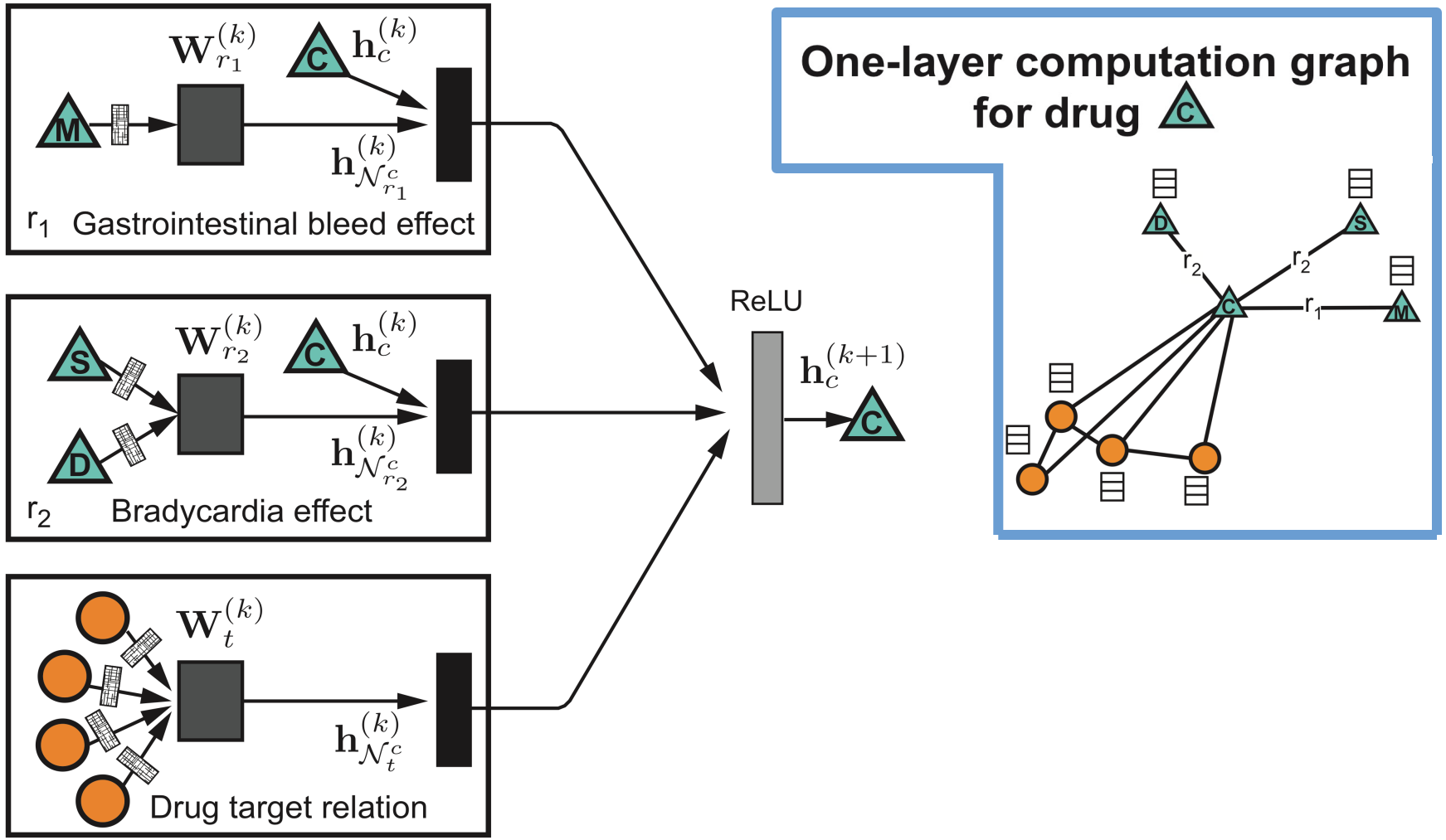


Determine a node's computation graph



Learn how to transform and propagate information across the graph

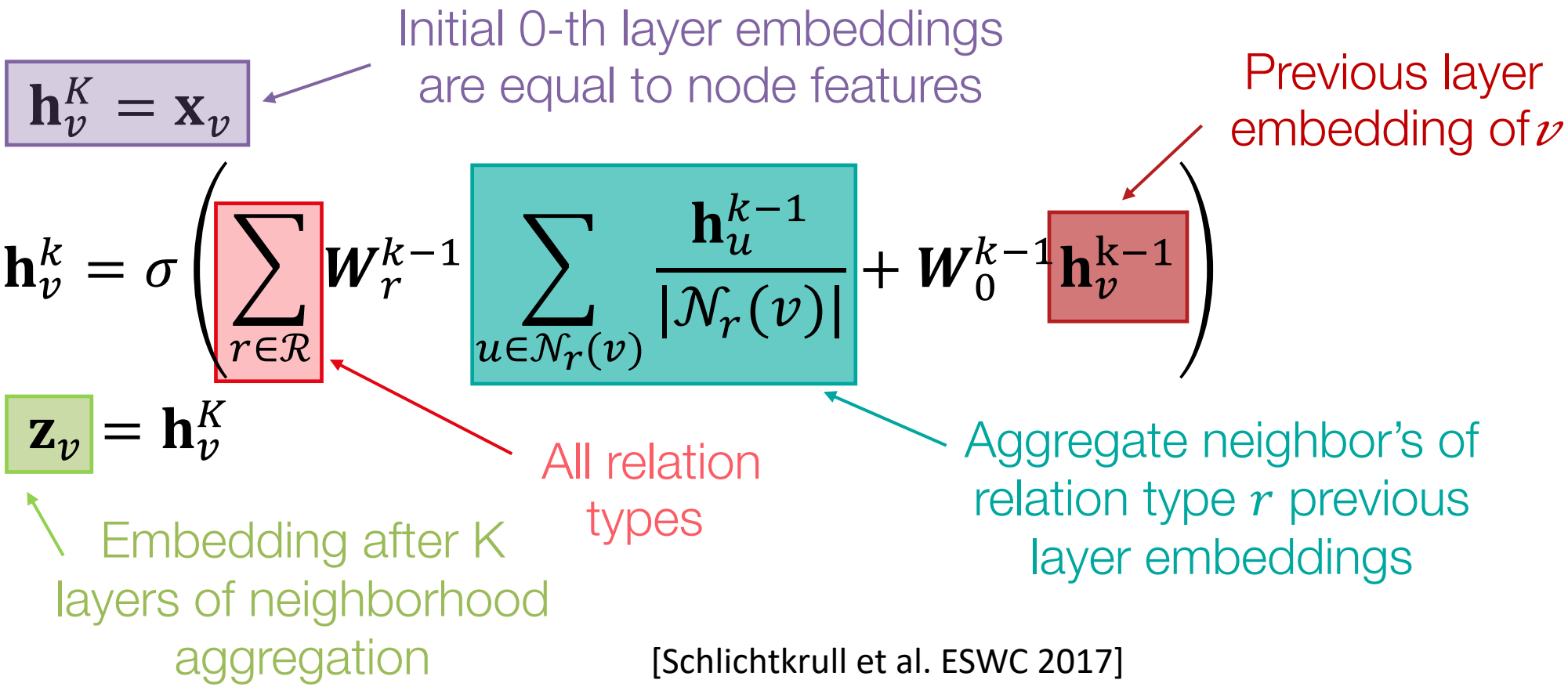
Example: Aggregation



$\mathbf{W}_{r_i}^{(k)}$ Neural network weight matrices

The Math: Deep Encoder

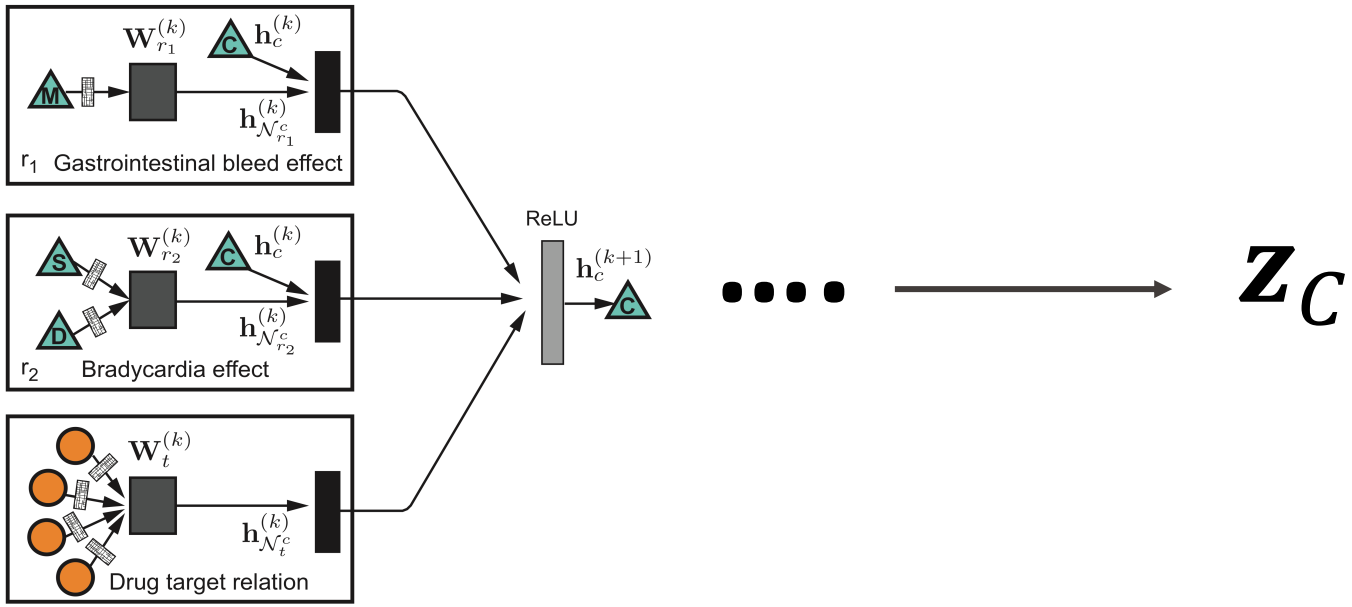
- **Approach:** Average neighbor messages for each edge type and apply a neural network



[Schlichtkrull et al. ESWC 2017]

Training the Model

How do we train the model to generate embeddings?

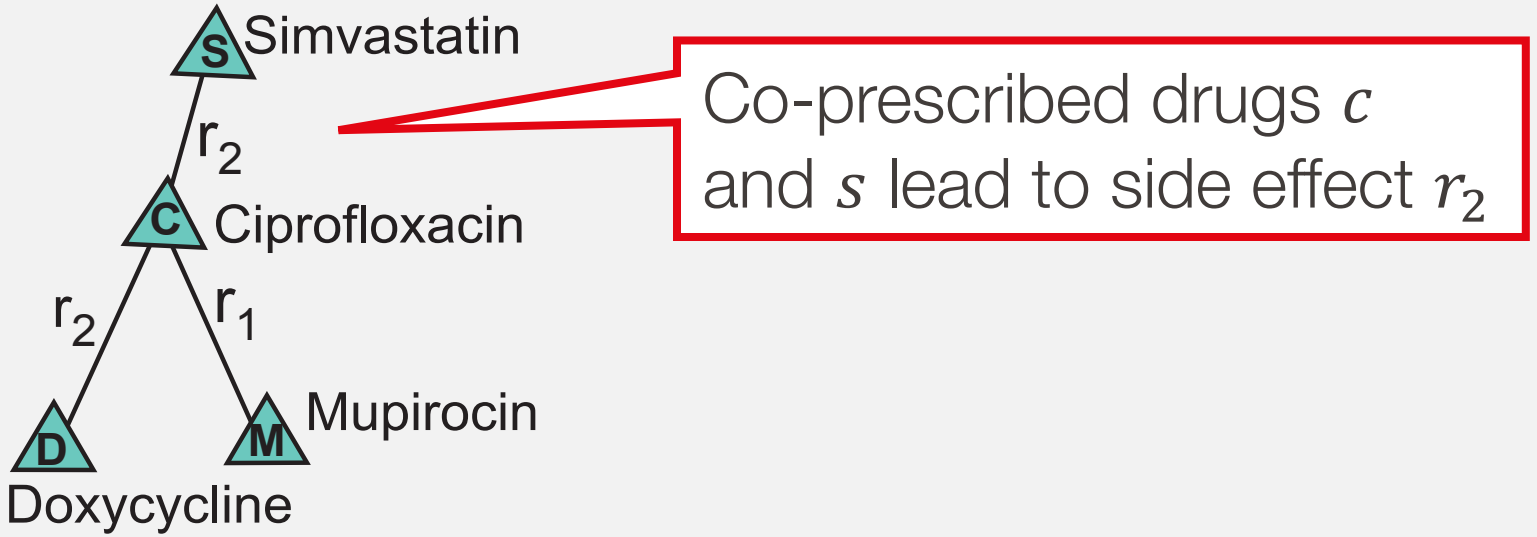


Need to define a loss function on the embeddings!

Example: Drug Side Effects

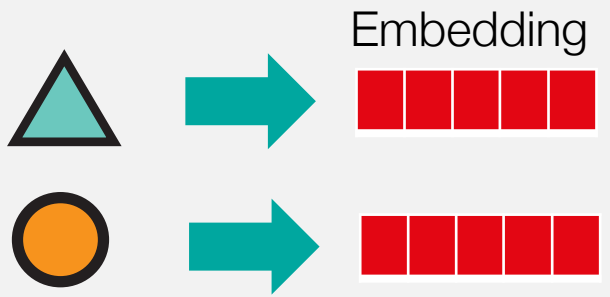
Goal: Predict labeled edges between drug nodes

Query: Given a drug pair c, s, how likely does an edge (c, r_2, s) exist?

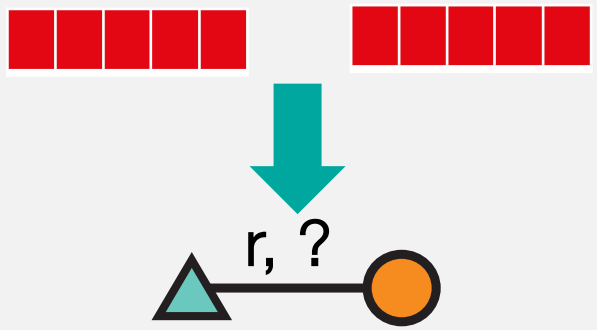


Example: Drug Side Effects

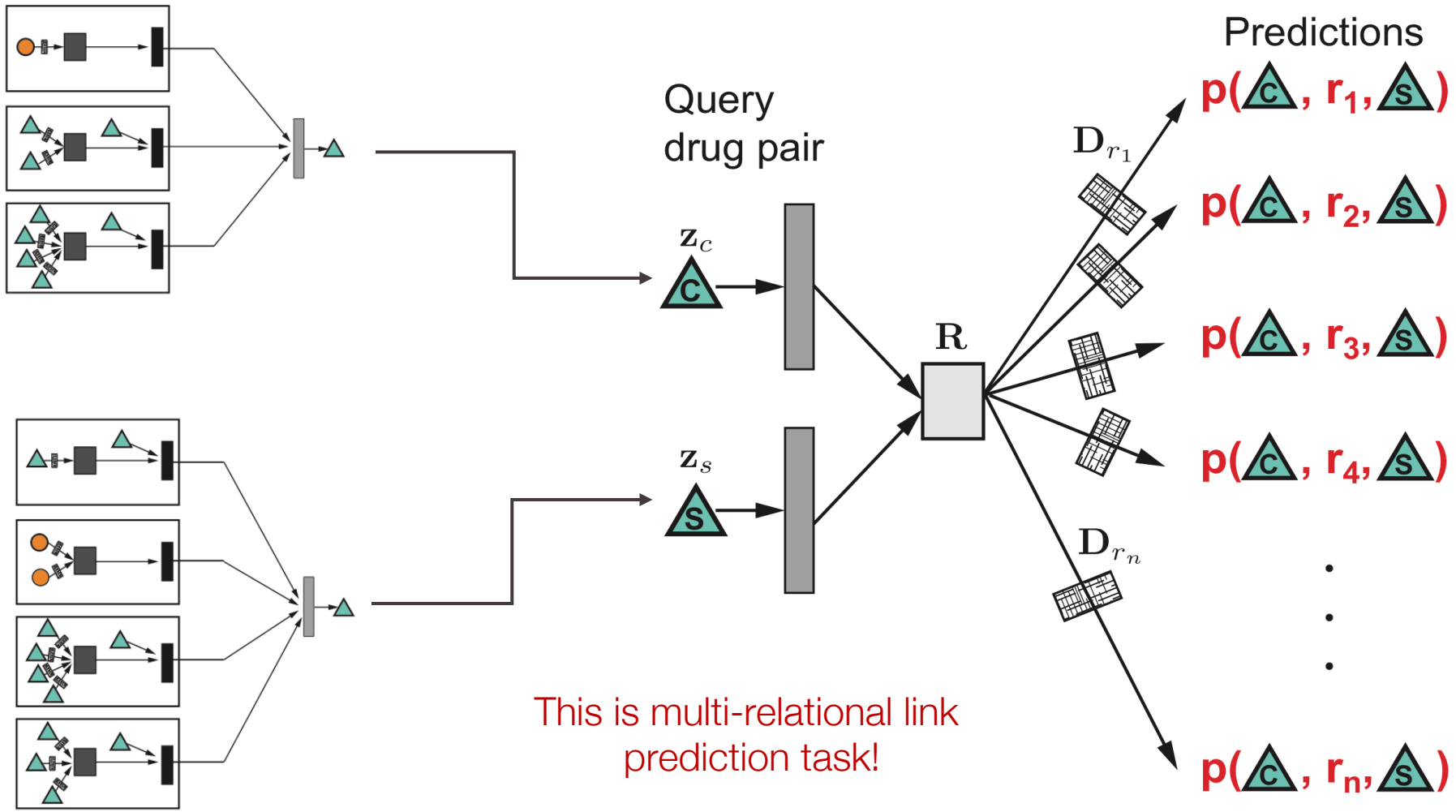
1) Take the graph and learn a d-dimensional vector (*embedding*) for every node



2) Use the learned embeddings to predict side effects of drug pairs



Example: Drug Side Effects



$\mathbf{R}, \mathbf{D}_{r_i}$ Neural network weight matrices

Recap

- Graph convolution:
  - Define computation graph based on node's neighborhood
  - Learn how to propagate information across the graph to compute node features
- Different prediction tasks:
  - Node-level, edge-level, graph-level
- Heterogenous GCNs:
  - Different relation types and different node types
- Biomedical applications:
  - Many graphs in biomedicine!
  - Example: polypharmacy side effects prediction

Additional Readings

- Hamilton, Ying, Leskovec. Representation Learning on Graphs: Methods and Applications. *IEEE Data Engineering Bulletin* (2017)
- Kipf, Welling. Semi-supervised Classification with Graph Convolutional Networks. *ICLR* (2017)
- Hamilton, Ying, Leskovec. Inductive Representation Learning on Large Graphs. *NeurIPS* (2017)
- Ying et al. Hierarchical Graph Representation Learning with Differentiable Pooling. *NeurIPS* 2018
- Schlichtkrull, Kipf, Bloem, Berg, Titov, Welling. Modeling Relational Data with Graph Convolutional Networks. *ESWC* (2017)
- Zitnik, Agrawal, Leskovec. Modeling polypharmacy side effects with graph convolutional network. *Bioinformatics* '18
- CS224W: https://web.stanford.edu/class/cs224w/

Next Lecture: Guest Lecture Vikas Garg (Aalto University)

https://epfl.zoom.us/j/61607134687?pwd=aG5nMWI0dnhPR2w2QlpyY2s4N0xnZz09

Meeting ID: 616 0713 4687 Passcode: 437112 --- Join by SIP 61607134687@fr.zmeu.us

Homework 2 will be released tomorrow!

Any Feedback?

Give us feedback on the lecture:

- https://go.epfl.ch/cs502-lecture-3-feedback