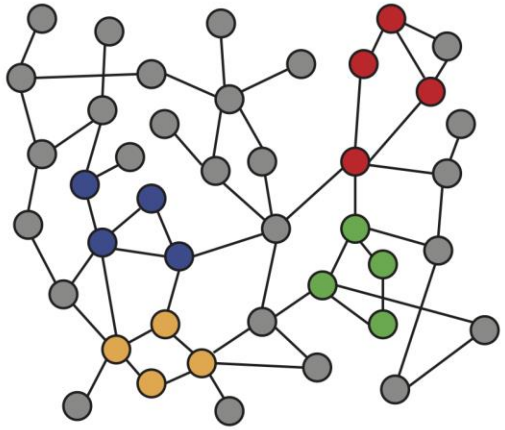
Graph Learning Basics

Jiaxuan You

Assistant Professor at UIUC CDS

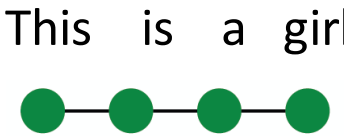
CS598: Deep Learning with Graphs, 2024 Fall

Recap: Machine Learning with Graphs is Hard

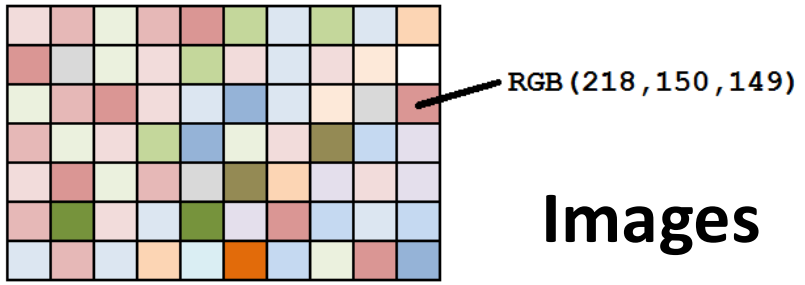


Graphs

vs.



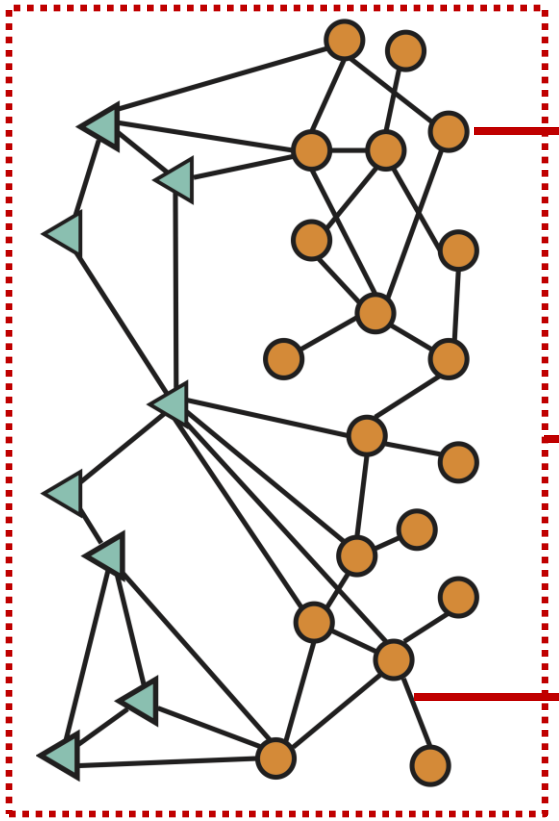
Text



Images

- **Arbitrary size** and topological **structure**
- Nodes have **no fixed ordering**

Recap: Graph Machine Learning **Tasks**



Node-level prediction
"Classify user by their type in a social network"

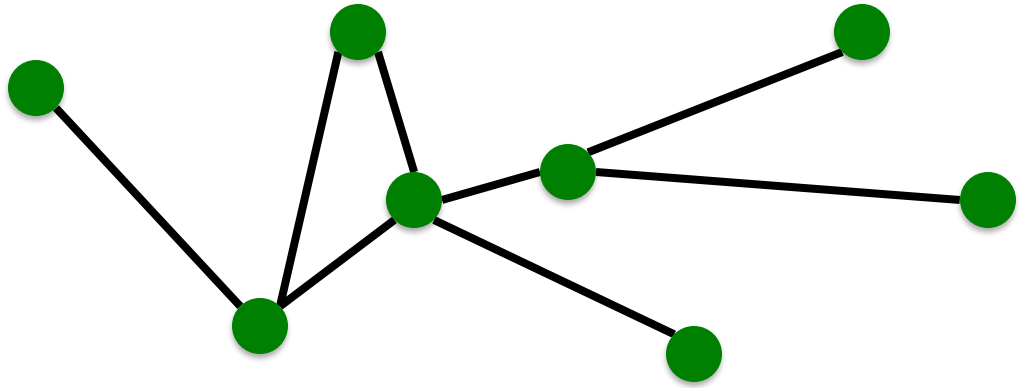
Graph-level prediction
"Predict which molecules are drug-like"

Edge-level prediction
"Recommend item nodes to user nodes"

Graph Learning Basics

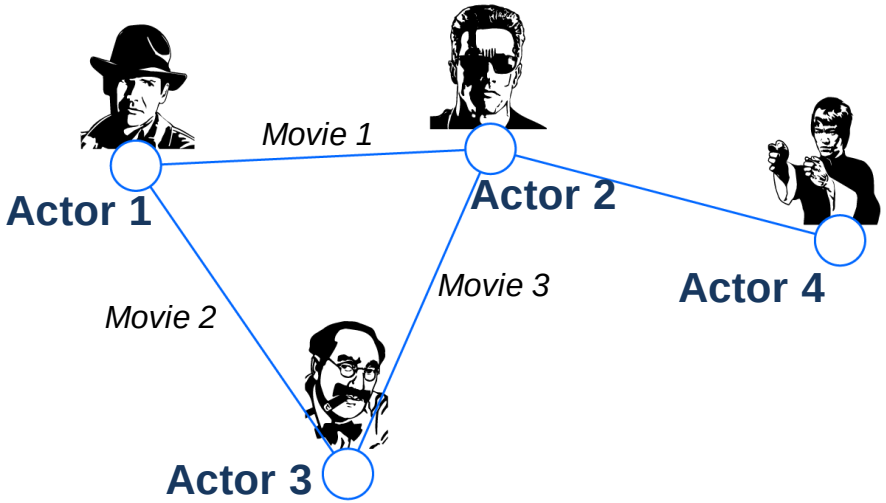
Graph Representation Basics

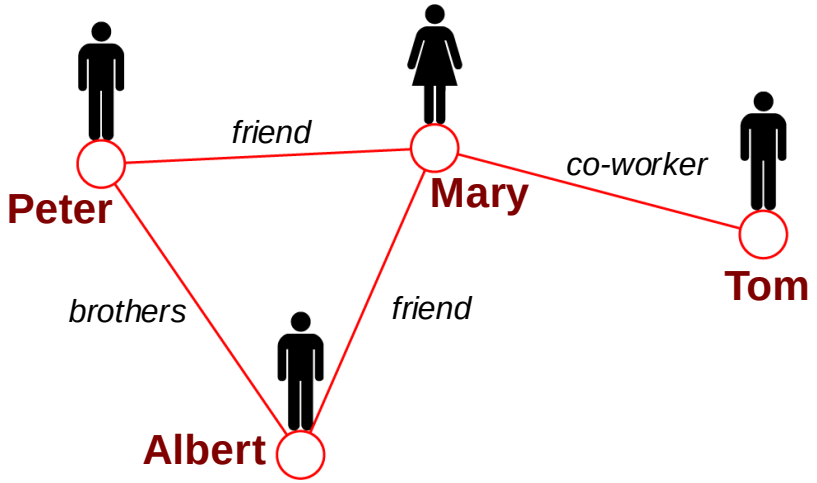
Components of a Network

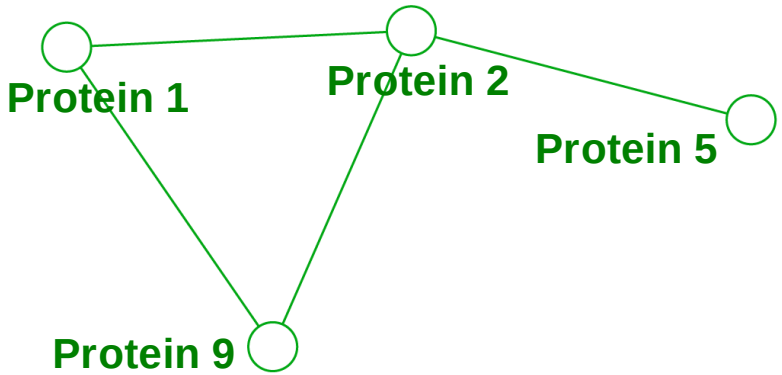


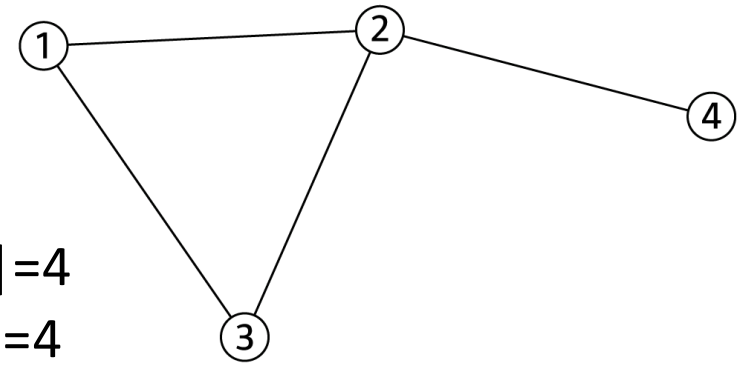
- **Objects**: nodes, vertices — ***N***
- **Interactions**: links, edges — ***E***
- **System**: network, graph — ***G(N,E)***

Graphs: A Common Language







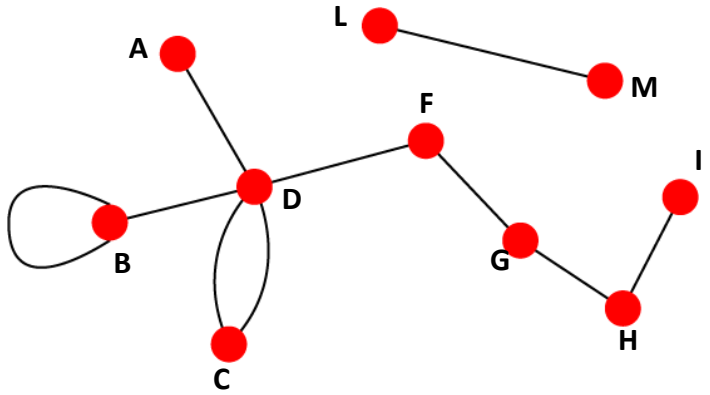


How do you define a graph?

- **How to build a graph:**
  - **What are nodes?**
  - **What are edges?**
- **Choice of the proper network representation of a given domain/problem determines our ability to use networks successfully:**
  - In some cases, there is a unique, unambiguous representation
  - In other cases, the representation is by no means unique
  - The way you assign links will determine the nature of the question you can study

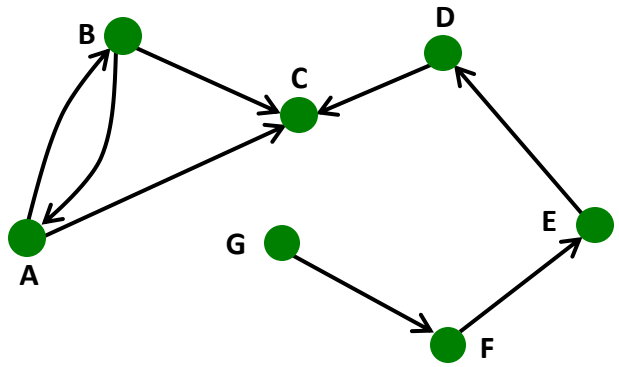
Directed vs. Undirected Graphs

- **Undirected**
  - Links: undirected (symmetrical, reciprocal)



  - Examples:
    - Collaborations
    - Friendship on Facebook
    - Pairs of positive/negative samples in contrastive learning

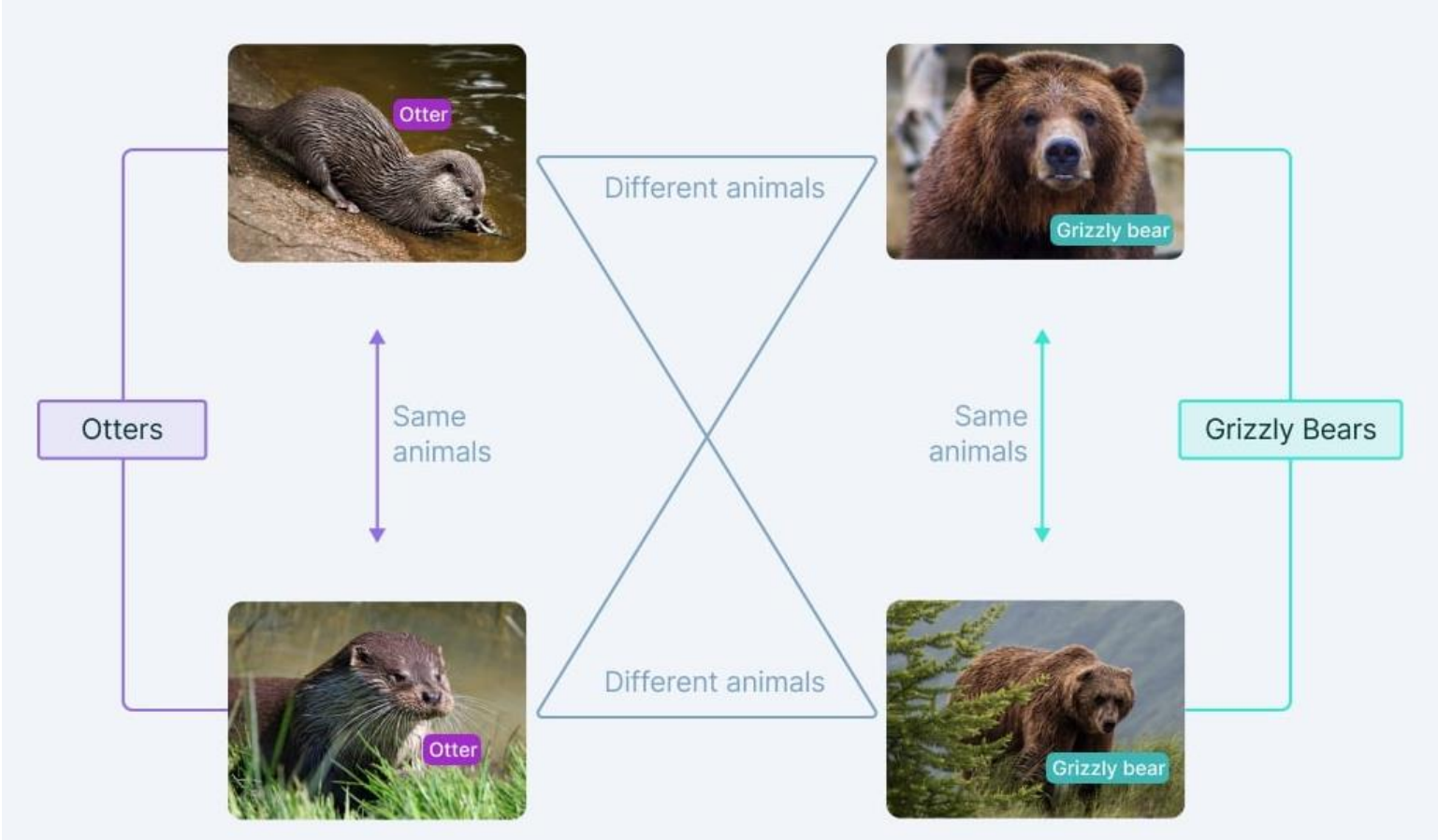
- **Directed**
  - Links: directed (arcs)



  - Examples:
    - Phone calls
    - Following on Twitter
    - Computational graphs in deep learning

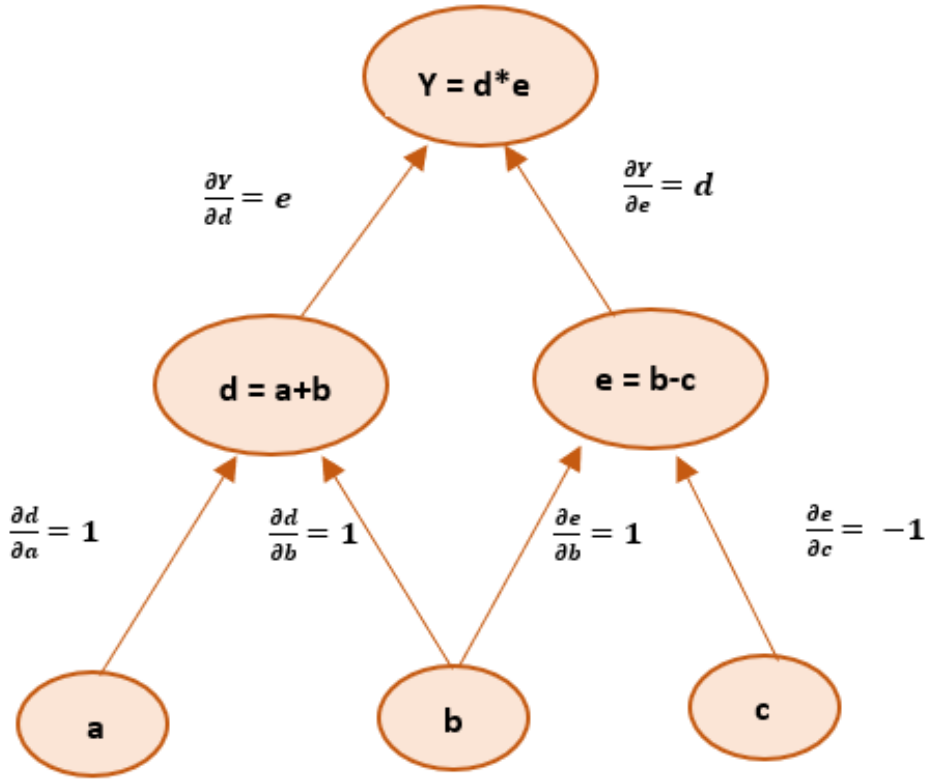
Undirected Graph Example in DL

- Contrastive learning

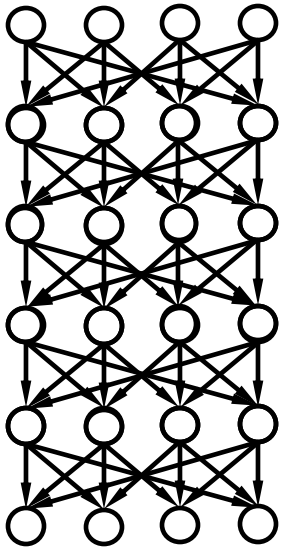


Directed Graph Examples in DL

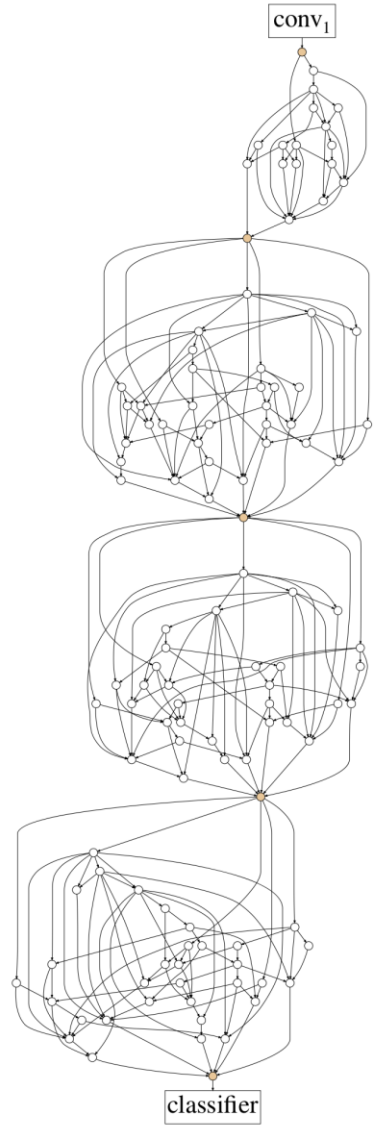
- Computational graphs



A 5-layer
Neural network



RandWire, Xie et al., 2019



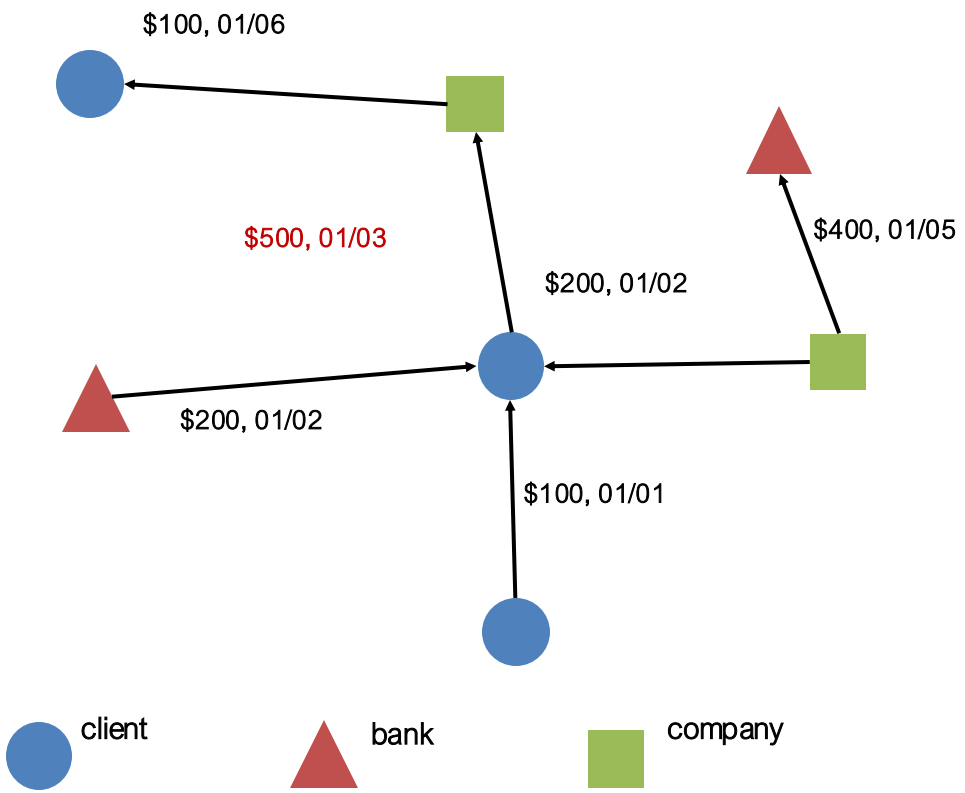
Dynamic Graphs

- **Dynamic graph representation option 1: Graph + timestamp**

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

  - Nodes $v_i \in V$
  - Edges $(v_i, v_j) \in E$
  - Timestamps $T(v_i), T(v_i, v_j)$
- **Dynamic graph representation option 2: Snapshots of graphs**
  - Each snapshot is a standard graph G_t
  - A dynamic graph is a series of graph snapshots $G = (G_1, \ldots, G_T)$

Dynamic Graph Example: Financial Networks



- Transaction-based approach
  - "On 01/03, Client A sends Company B $500"
- **Graph-based approach**
  - Represent a transaction in a much broader context
  - A dynamic network, changing over time

Heterogeneous Graphs

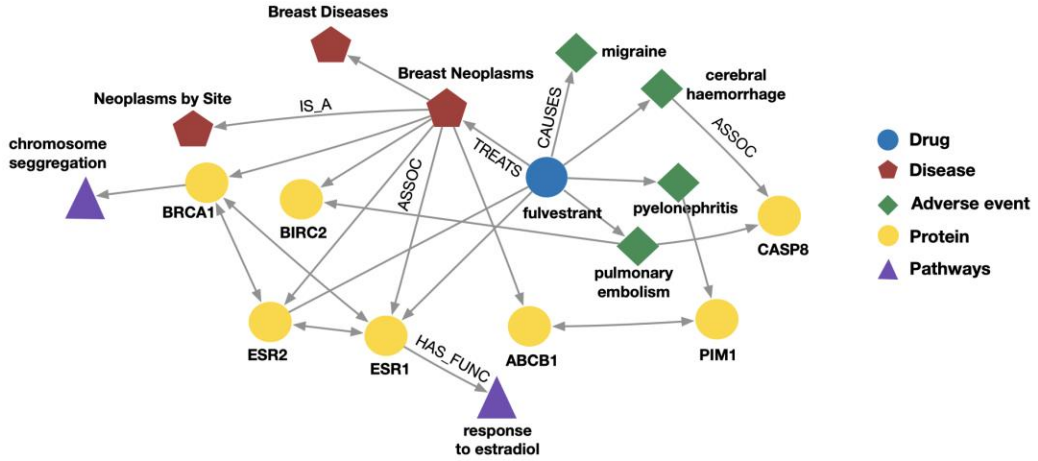
- **A heterogeneous graph is defined as**

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

  - Nodes **with node types** $v \in V$
    - **Node type** for node v: $\tau(v)$
  - Edges **with edge types** $(u, v) \in E$
    - **Edge type** for edge (u, v): $\phi(u, v)$
    - **Relation type** for edge e is a tuple: $r(u, v) = (\tau(u), \phi(u, v), \tau(v))$

An edge can be described as a pair of nodes

Many Graphs are Heterogeneous Graphs



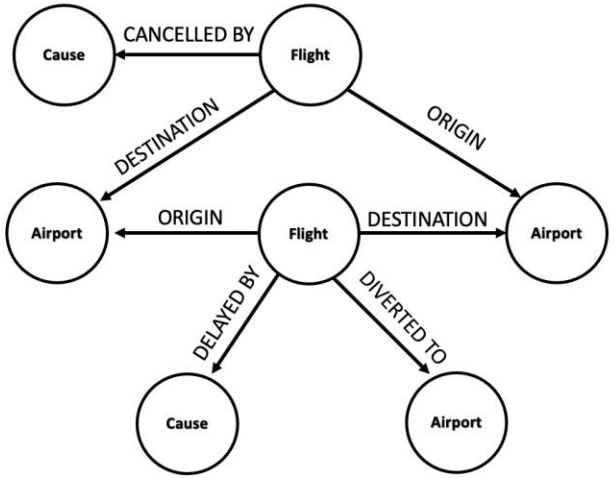
Biomedical Knowledge Graphs

Example node: Migraine

Example relation: (fulvestrant, Treats, Breast Neoplasms)

Example node type: Protein

Example edge type: Causes



Event Graphs

Example node: SFO

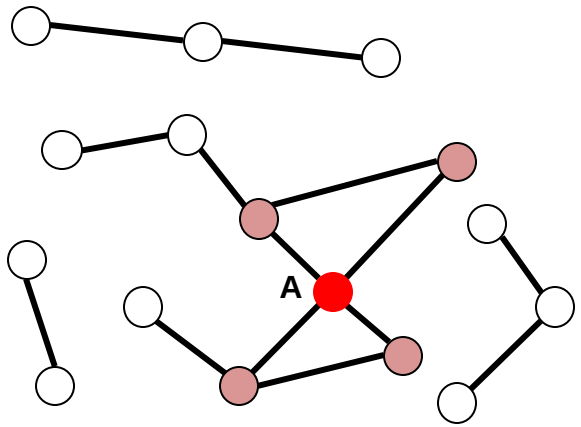
Example relation: (UA689, Origin, LAX)

Example node type: Flight

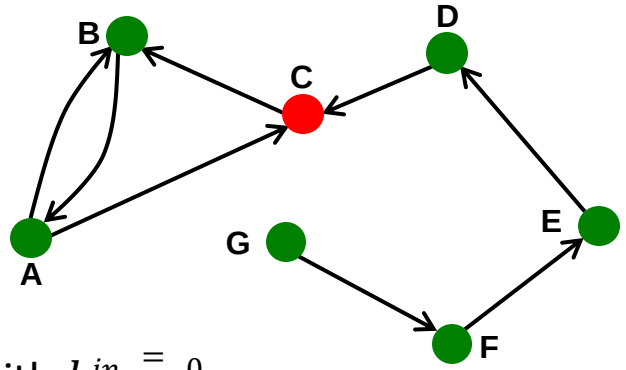
Example edge type: Destination

Node Degrees

- **Undirected**
  - **Node degree, k_i:** the number of edges adjacent to node i, $k_A = 4$
  - ***Avg. degree:*** $\bar{k} = \langle k \rangle = \frac{1}{N}\sum_{i=1}^{N} k_i = \frac{2E}{N}$



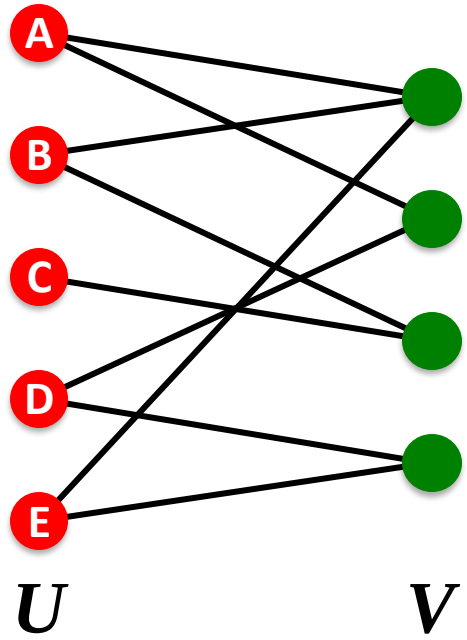
- **Directed**
  - In directed networks we define an **in-degree** and **out-degree**.
    The (total) degree of a node is the sum of in- and out-degrees.
  - $k_C^{in} = 2, k_C^{out} = 1, k_C = 3$
    $\bar{k} = \frac{E}{N}, \overline{k^{in}} = \overline{k^{out}}$



Source: Node with $k^{in} = 0$
Sink: Node with $k^{out} = 0$

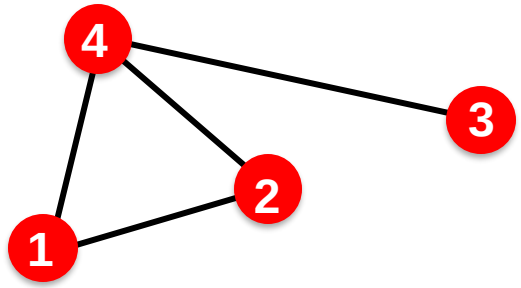
Bipartite Graph

- **Bipartite graph** is a graph whose nodes can be divided into two disjoint sets $\boldsymbol{U}$ and $\boldsymbol{V}$ such that every link connects a node in $\boldsymbol{U}$ to one in $\boldsymbol{V}$; that is, $\boldsymbol{U}$ and $\boldsymbol{V}$ are **independent sets.**
- **Examples:**
  - Authors-to-Papers (they authored)
  - Actors-to-Movies (they appeared in)
  - Users-to-Movies (they rated)
  - Recipes-to-Ingredients (they contain)
- **"Folded" networks:**
  - Author collaboration networks
  - Movie co-rating networks

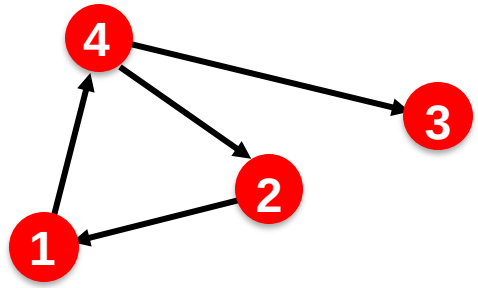


Representing Graphs: Adjacency Matrix

- $\boldsymbol{A_{ij} = 1}$ if there is a link from node $\boldsymbol{i}$ to node $\boldsymbol{j}$
- $\boldsymbol{A_{ij} = 0}$ otherwise



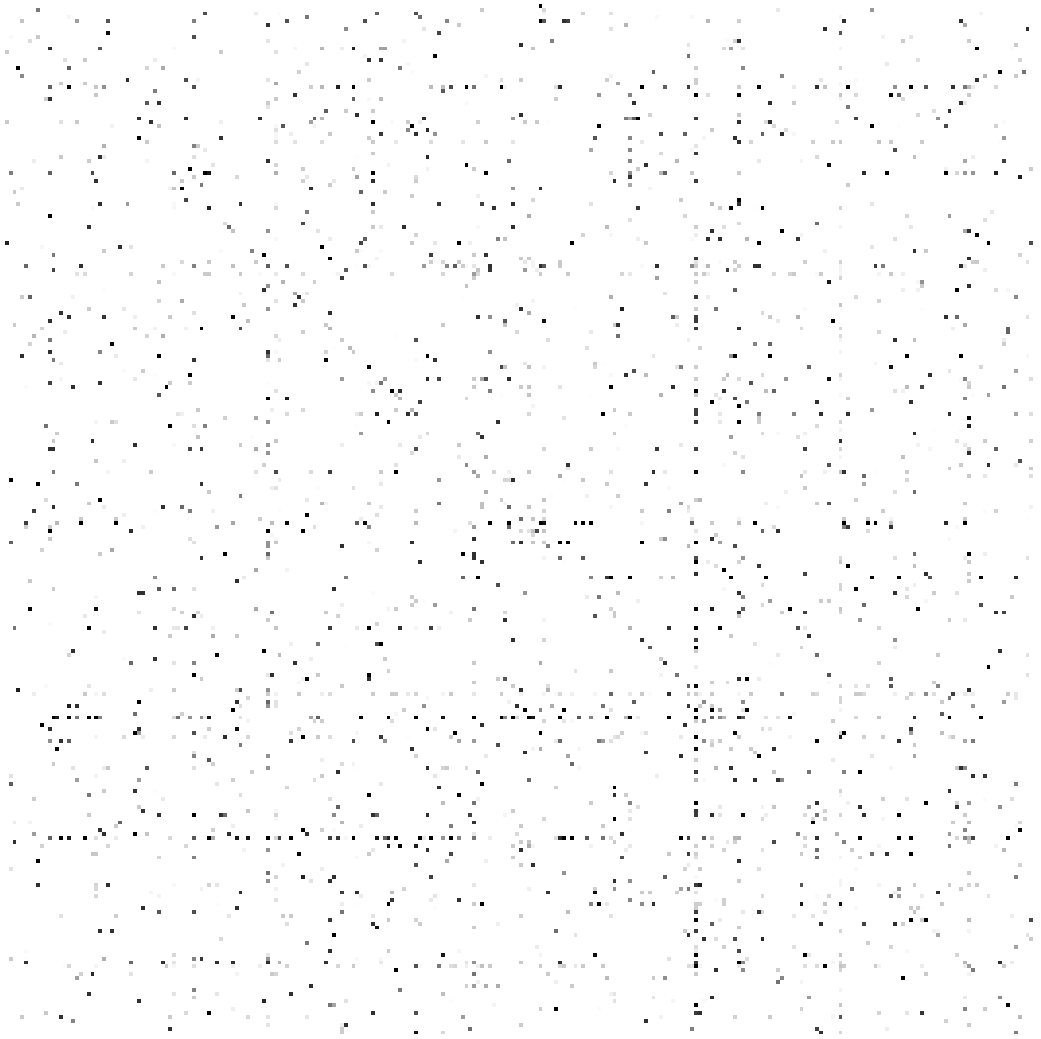
$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$



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

Note that for a directed graph (right) the matrix is not symmetric.

Adjacency Matrices are Sparse



Networks are Sparse Graphs

- **Most real-world networks are sparse**
- **$E << E_{max}$ (or $k << N-1$)**

NETWORK	NODES	LINKS	DIRECTED/ UNDIRECTED	N	L	<k>
Internet	Routers	Internet connections	Undirected	192,244	609,066	6.33
WWW	Webpages	Links	Directed	325,729	1,497,134	4.60
Power Grid	Power plants, transformers	Cables	Undirected	4,941	6,594	2.67
Phone Calls	Subscribers	Calls	Directed	36,595	91,826	2.51
Email	Email Addresses	Emails	Directed	57,194	103,731	1.81
Science Collaboration	Scientists	Co-authorship	Undirected	23,133	93,439	8.08
Actor Network	Actors	Co-acting	Undirected	702,388	29,397,908	83.71
Citation Network	Paper	Citations	Directed	449,673	4,689,479	10.43
E. Coli Metabolism	Metabolites	Chemical reactions	Directed	1,039	5,802	5.58
Protein Interactions	Proteins	Binding interactions	Undirected	2,018	2,930	2.90

- **Consequence: Adjacency matrix is filled with zeros**!
- (**Density of the matrix (E/N^2):** WWW=1.51×10^{-5}, MSN IM = 2.27×10^{-8})

Node and Edge Attributes

Possible options:

- Weight (*e.g.*, frequency of communication)
- Ranking (best friend, second best friend…)
- Type (friend, relative, co-worker)
- Sign: Friend vs. Foe, Trust vs. Distrust
- Properties depending on the structure of the rest of the graph: Number of common friends

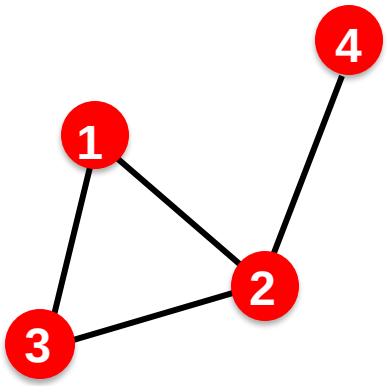
More Types of Graphs

- Unweighted (undirected)

$$A_{ij} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

$$A_{ii} = 0 \qquad A_{ij} = A_{ji}$$

$$E = \frac{1}{2}\sum_{i,j=1}^{N} A_{ij} \qquad \bar{k} = \frac{2E}{N}$$

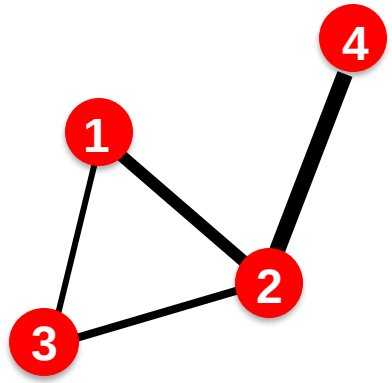


- Weighted (undirected)

$$A_{ij} = \begin{pmatrix} 0 & 2 & 0.5 & 0 \\ 2 & 0 & 1 & 4 \\ 0.5 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{pmatrix}$$

$$A_{ii} = 0 \qquad A_{ij} = A_{ji}$$

$$E = \frac{1}{2}\sum_{i,j=1}^{N} nonzero(A_{ij}) \qquad \bar{k} = \frac{2E}{N}$$



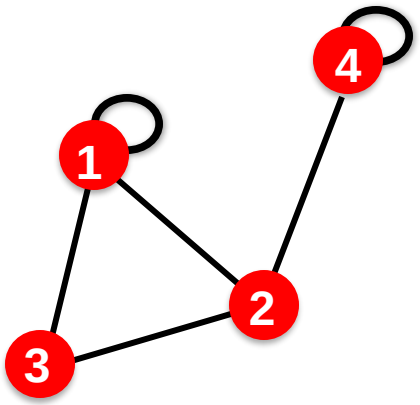
More Types of Graphs

- Self-edges (self-loops) (undirected)
  - Examples: Proteins, Hyperlinks

$$A_{ij} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

$A_{ii} \neq 0$ $\quad A_{ij} = A_{ji}$

$$E = \frac{1}{2} \sum_{i,j=1, i \neq j}^{N} A_{ij} + \sum_{i=1}^{N} A_{ii}$$

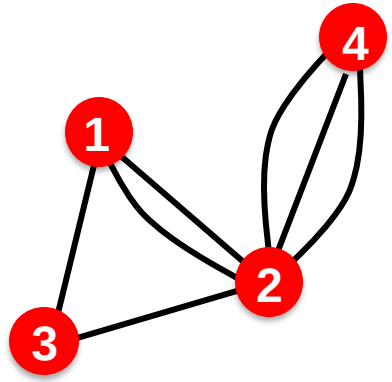


- Multigraph (undirected)
  - Examples: Communication, Collaboration

$$A_{ij} = \begin{pmatrix} 0 & 2 & 1 & 0 \\ 2 & 0 & 1 & 3 \\ 1 & 1 & 0 & 0 \\ 0 & 3 & 0 & 0 \end{pmatrix}$$

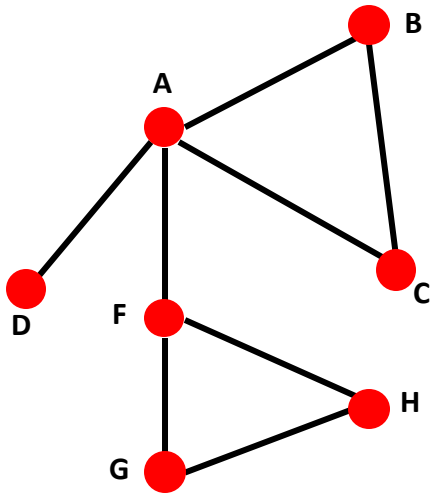
$A_{ii} = 0$ $\quad A_{ij} = A_{ji}$

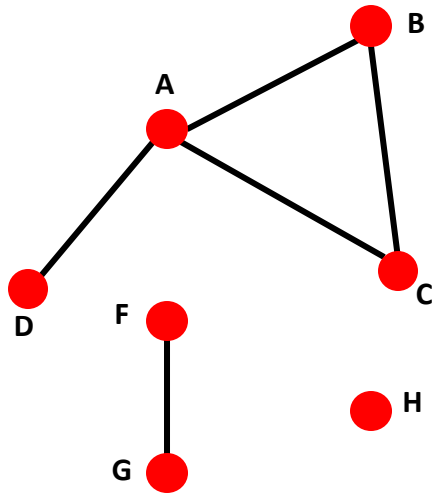
$$E = \frac{1}{2} \sum_{i,j=1}^{N} nonzero(A_{ij}) \qquad \bar{k} = \frac{2E}{N}$$



Connectivity of Undirected Graphs

- Connected (undirected) graph:
  - Any two vertices can be joined by a path
- A disconnected graph is made up by two or more connected components





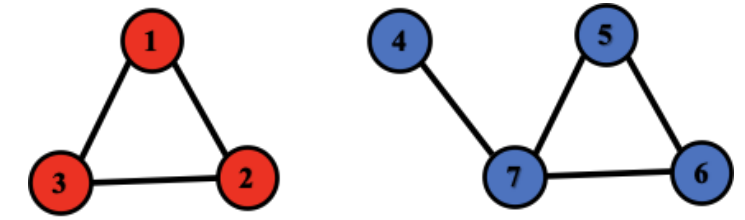
Largest Component:
Giant Component

Isolated node (node H)

Connectivity: Example

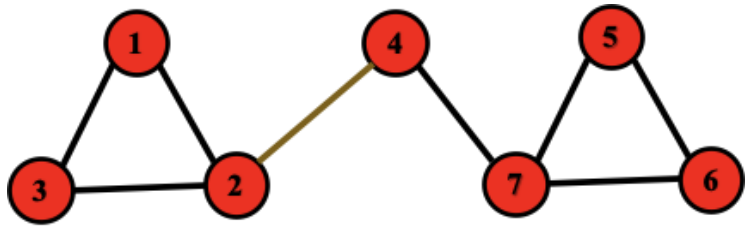
- The adjacency matrix of a network with several components can be written in a block- diagonal form, so that nonzero elements are confined to squares, with all other elements being zero:

Disconnected



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

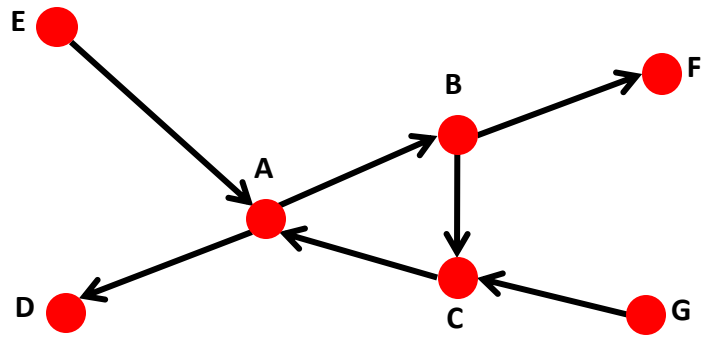
Connected



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

Connectivity of Directed Graphs

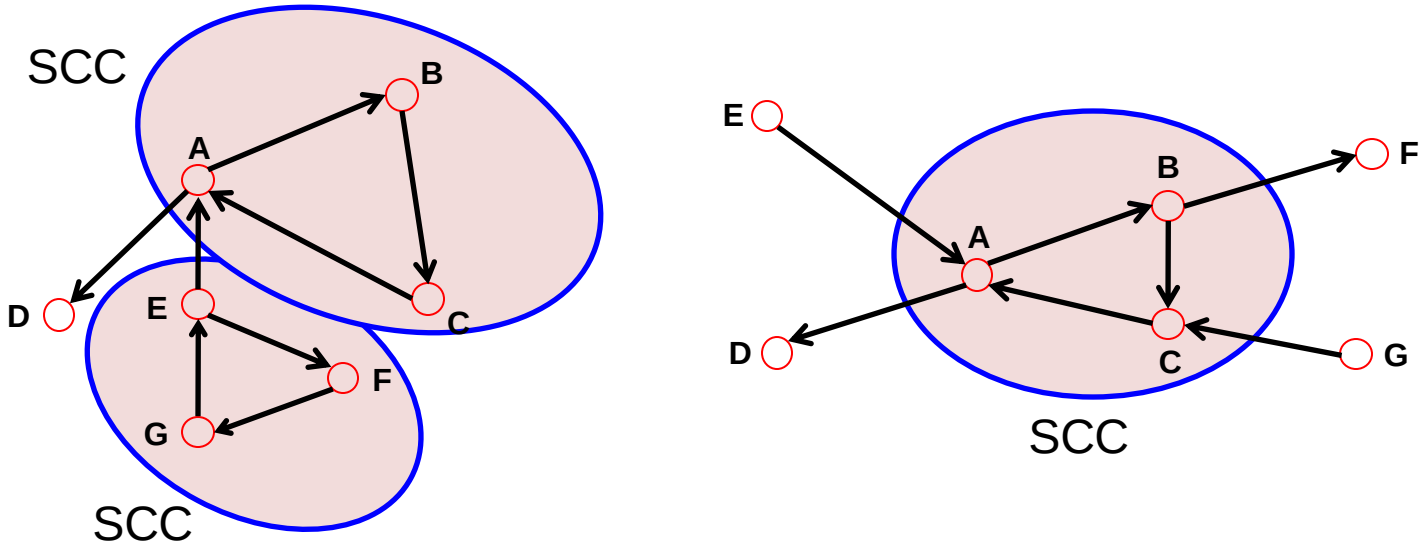
- **Strongly connected directed graph**
  - has a path from each node to every other node and vice versa (e.g., A-B path and B-A path)
- **Weakly connected directed graph**
  - is connected if we disregard the edge directions



Graph on the left is connected but not strongly connected (e.g., there is no way to get from F to G by following the edge directions).

Connectivity of Directed Graphs

- Strongly connected components (SCCs) can be identified, but not every node is part of a nontrivial strongly connected component.



In-component: nodes that can reach the SCC,

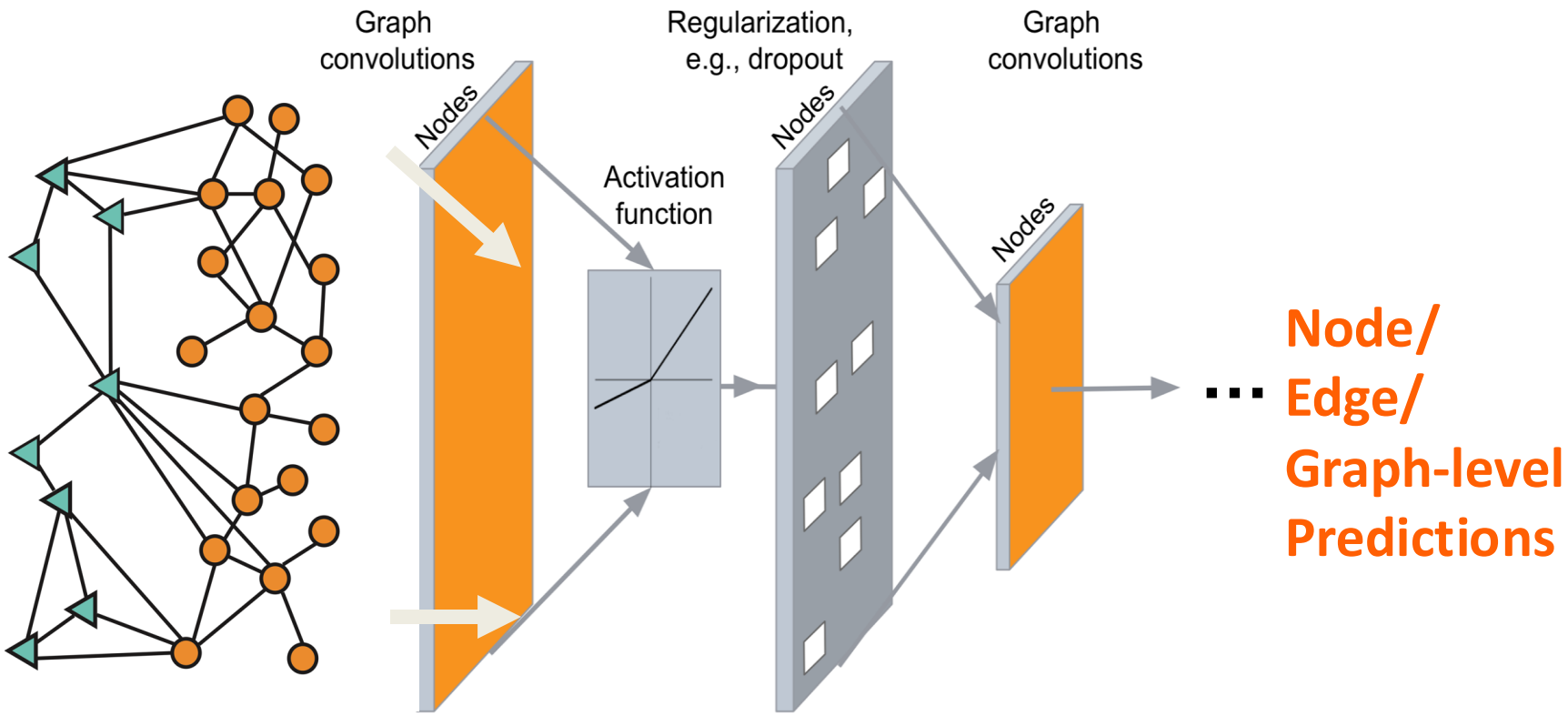
Out-component: nodes that can be reached from the SCC.

Graph Learning Basics

Graph Learning Prediction Tasks

Recap: Deep Learning with Graphs

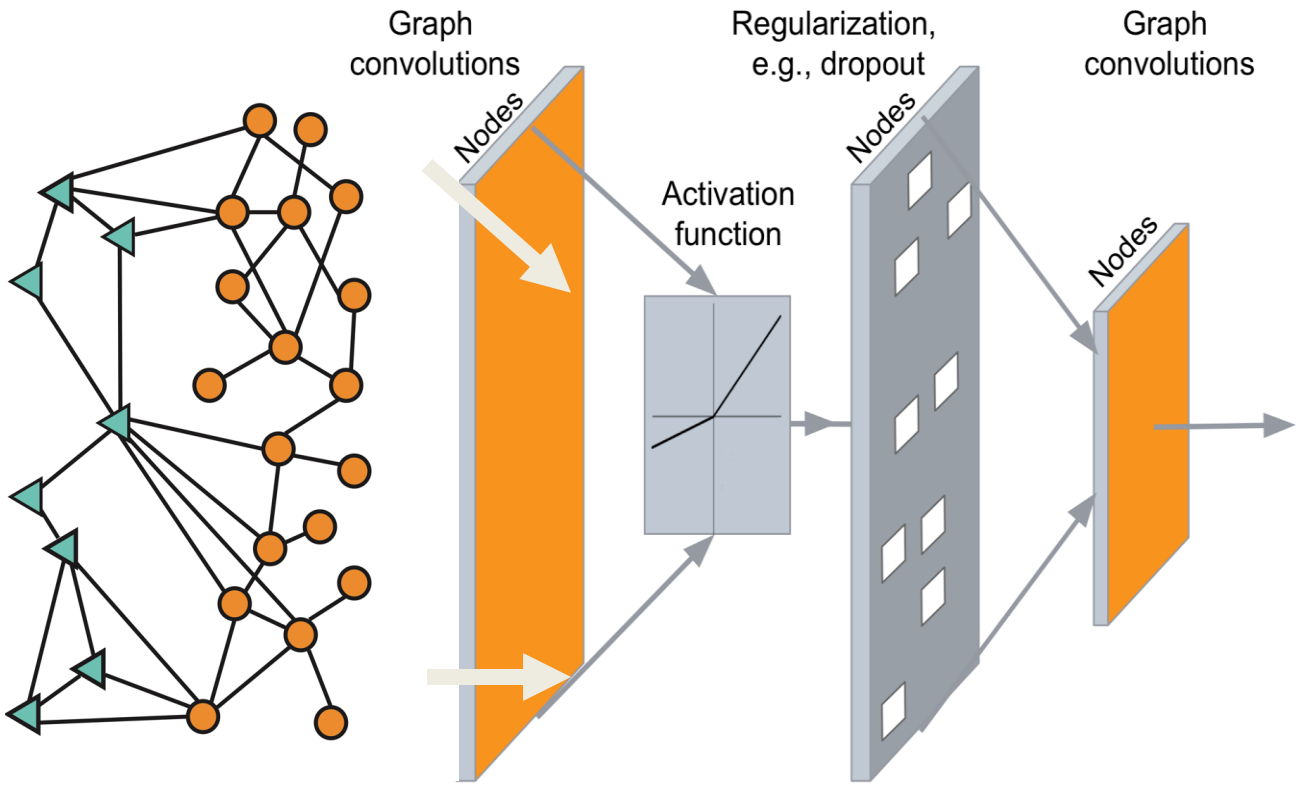
Input:
Graph



Node/ Edge/ Graph-level Predictions

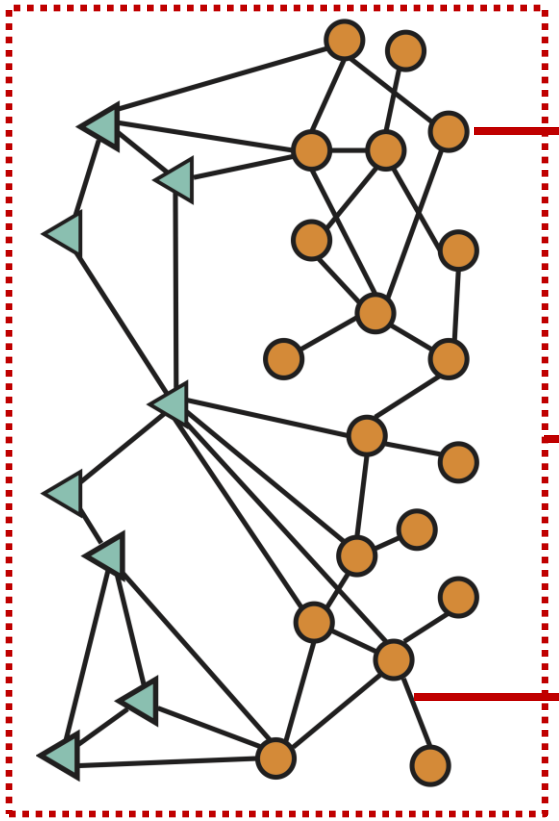
Deep Learning with Graphs

Input:
Graph



Node embedding → **Node/ Edge/ Graph-level Predictions**

Graph ML **Tasks**

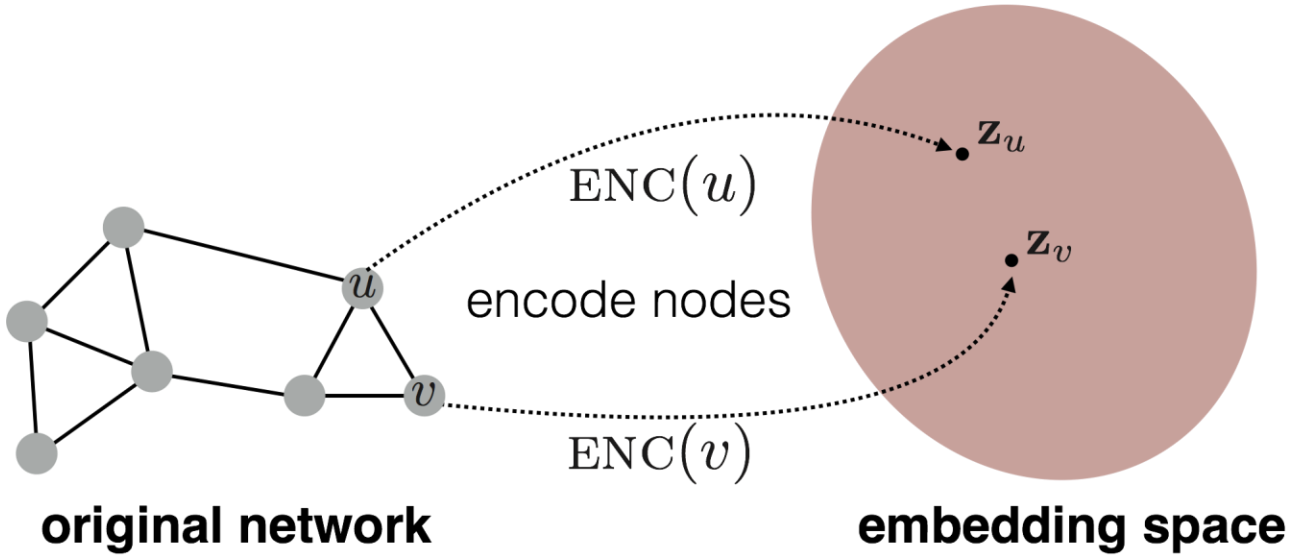


Node-level prediction

Graph-level prediction

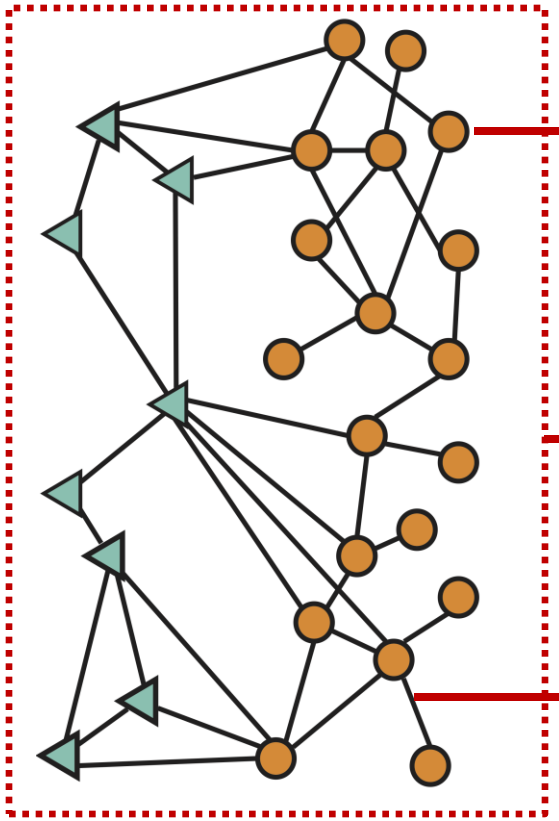
Edge-level prediction

Key Idea: **Node Embeddings**



Intuition: Map nodes to d-dimensional embeddings such that similar nodes in the graph are embedded close together

Graph ML **Tasks**

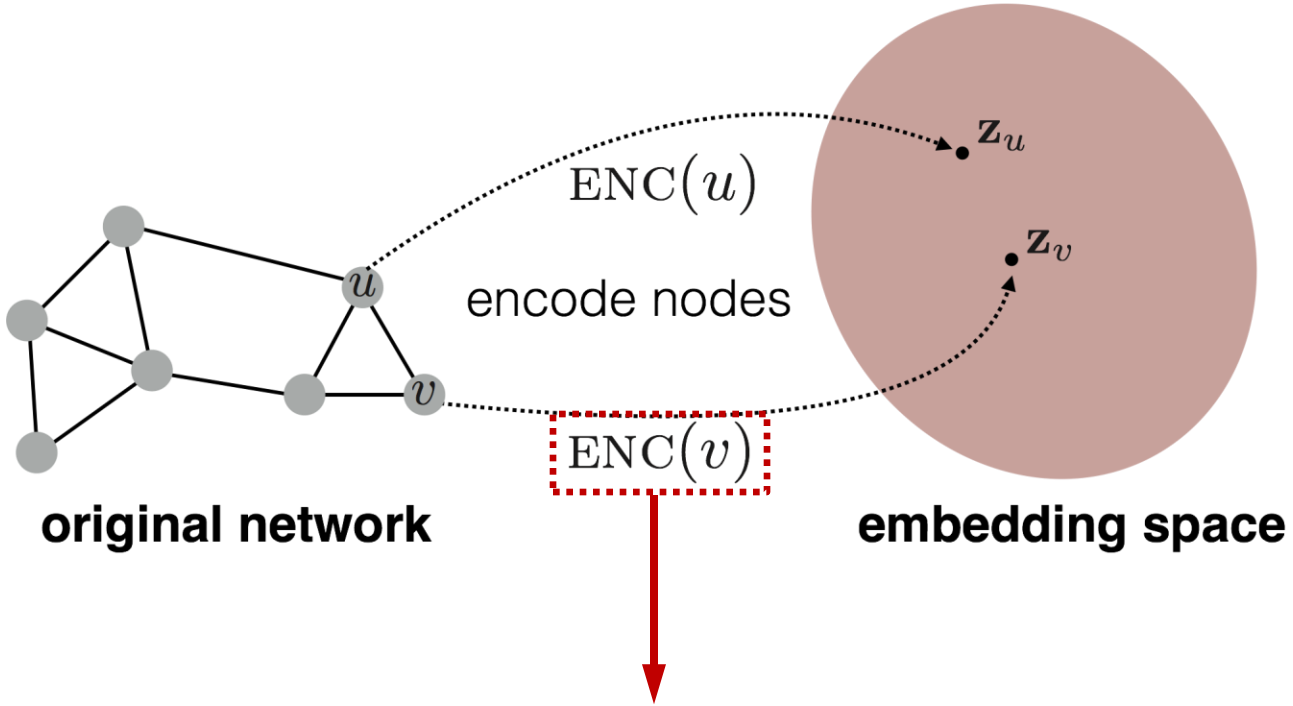


Node-level prediction

Graph-level prediction

Edge-level prediction

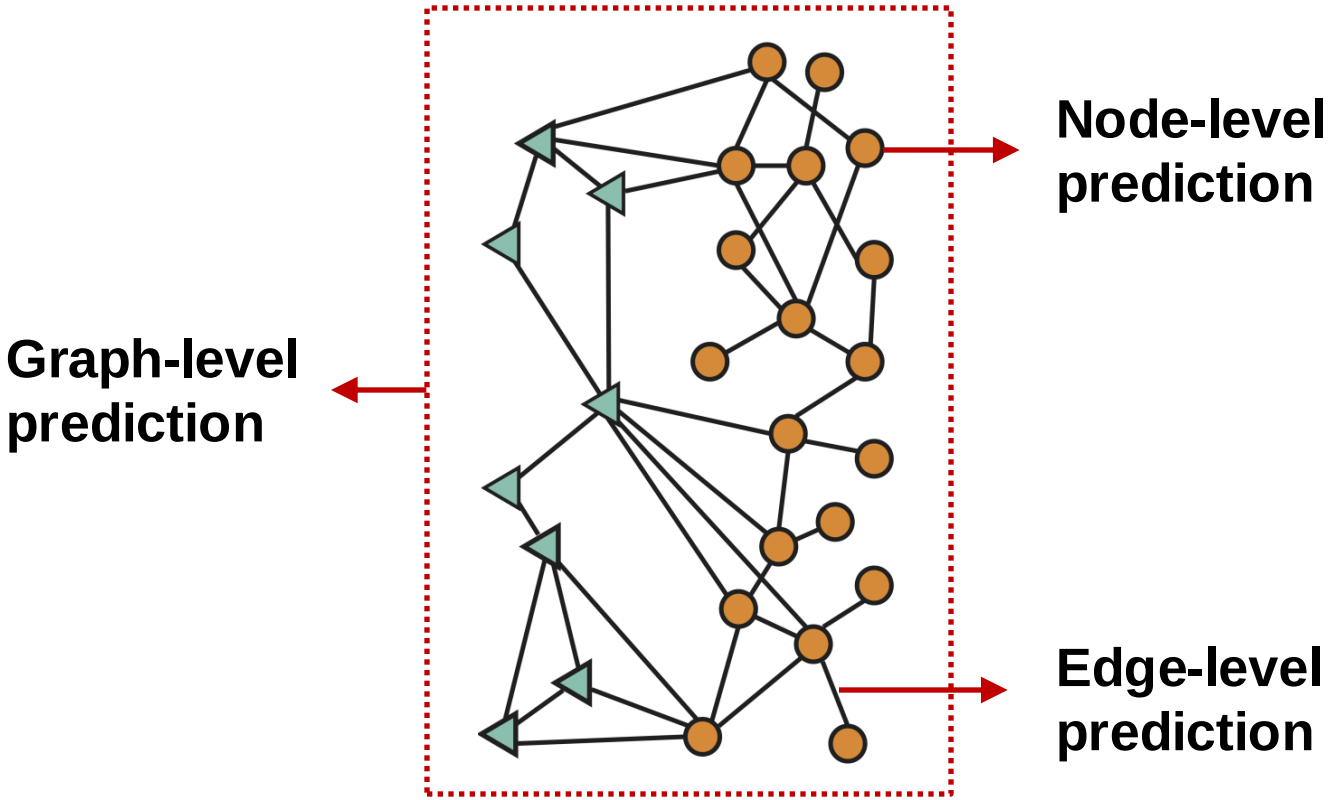
Key Idea: **Node Embeddings**



Graph Neural Networks (GNNs)

GNN Prediction Heads

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

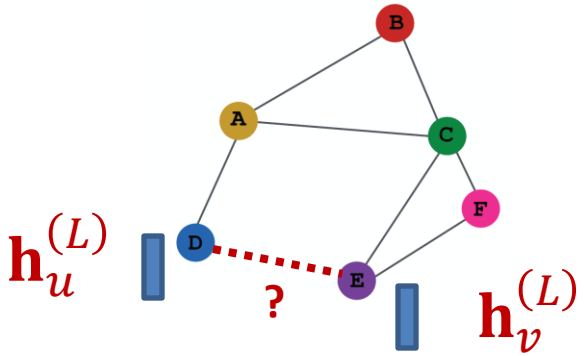


Prediction Heads: Node-level

- **Node-level prediction**: We can directly make prediction using node embeddings!
- Assuming 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
  - Regression: regress on k targets
- $\hat{\boldsymbol{y}}_v = \text{Head}_{\text{node}}(\mathbf{h}_v^{(L)}) = \mathbf{W}^{(H)}\mathbf{h}_v^{(L)}$
  - $\mathbf{W}^{(H)} \in \mathbb{R}^{k*d}$: 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

Prediction Heads: Edge-level

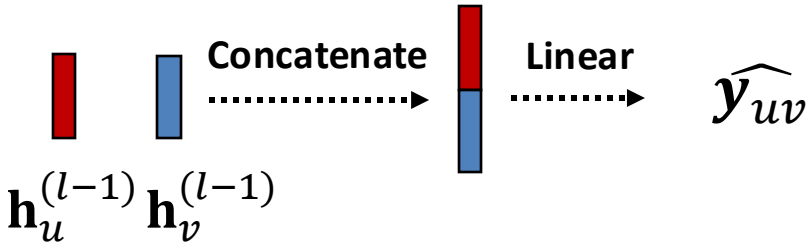
- **Edge-level prediction**: Make prediction using pairs of node embeddings
- Suppose we want to make k-way prediction
- $\hat{\boldsymbol{y}}_{\boldsymbol{uv}} = \text{Head}_{\text{edge}}(\mathbf{h}_u^{(L)}, \mathbf{h}_v^{(L)})$



- 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



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

Prediction Heads: Edge-level

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

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

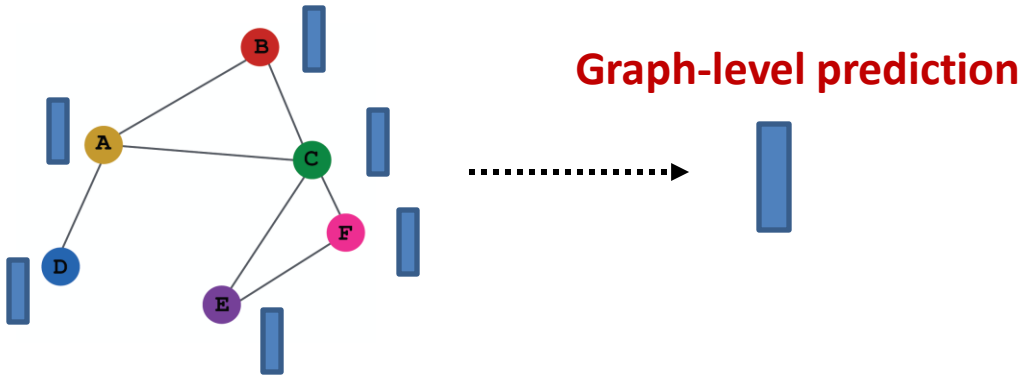
$$\ldots$$

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

$$\hat{\boldsymbol{y}}_{uv} = \text{Concat}(\hat{\boldsymbol{y}}_{uv}^{(1)}, \ldots, \hat{\boldsymbol{y}}_{uv}^{(k)}) \in \mathbb{R}^k$$

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



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**

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

Reading papers

Suggestions for Research

Sources of AI/ML papers

Recent major AI/ML conferences:

- NeurIPS 2023:

https://openreview.net/group?id=NeurIPS.cc/2023/Conference#tab-accept-oral

- ICML 2024:

https://openreview.net/group?id=ICML.cc/2024/Conference#tab-accept-oral

- ICLR 2024:

https://openreview.net/group?id=ICLR.cc/2024/Conference#tab-accept-oral

- LOG 2023 (Learning on graphs):

https://openreview.net/group?id=logconference.io/LOG/2023/Conference#tab-accept-oral

Sources of AI/ML papers

Latest Arxiv papers:

https://arxiv.org/list/cs.LG/pastweek?skip=0&show=25

arXiv > cs.LG

Machine Learning

Authors and titles for recent submissions

- Fri, 30 Aug 2024
- Thu, 29 Aug 2024
- Wed, 28 Aug 2024
- Tue, 27 Aug 2024
- Mon, 26 Aug 2024

See today's new changes

Total of 600 entries : 1-25 26-50 51-75 76-100 ... 576-600
Showing up to 25 entries per page: fewer | more | all

Fri, 30 Aug 2024 (showing first 25 of 117 entries)

[1] arXiv:2408.16765 [pdf, html, other]

A Score-Based Density Formula, with Applications in Diffusion Generative Models
Gen Li, Yuling Yan
Subjects: **Machine Learning (cs.LG)**; Artificial Intelligence (cs.AI); Probability (math.PR); Statistics Theory (math.ST); Machine Learning (stat.ML)

[2] arXiv:2408.16717 [pdf, html, other]

A GREAT Architecture for Edge-Based Graph Problems Like TSP
Attila Lischka, Jiaming Wu, Morteza Haghir Chehreghani, Balázs Kulcsár
Comments: 15 pages, 7 figures
Subjects: **Machine Learning (cs.LG)**; Artificial Intelligence (cs.AI)

Arxiv Copilot

Still feeling it's too much work to track the recent papers?

- Try **Arxiv Copilot**, a **personalized paper reading tool by U Lab**!
- https://huggingface.co/spaces/ulab-ai/ArxivCopilot

ulab-ai / **ArxivCopilot** like 27 Running Logs App Files Community Settings

Arxiv Copilot

➡️ **Goals**: Arxiv Copilot aims to provide personalized academic service!

✨ **Guidance**:

Step (1) Enter researcher name and generate research profile in "Set your profile!"👨‍💼

Step (2) Select time range and get relevant trending research topic and ideas in "Get trending topics and ideas!"; Here you also can sign up with email to get weekly research news 💡

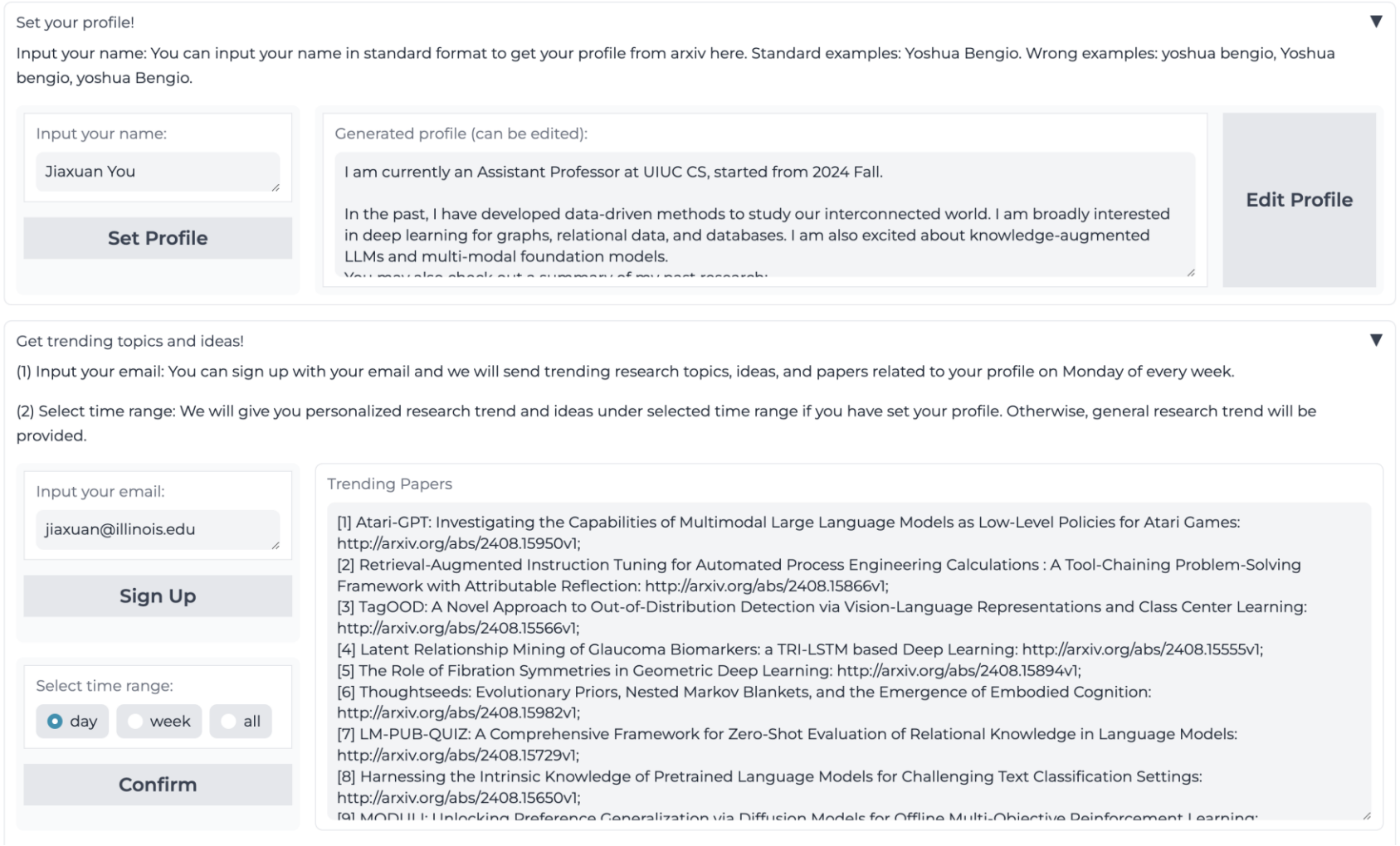
Step (3) Chat with Arxiv Copilot and choose the better response from two answers in "Chat with Arxiv Copilot!"; Here we appreciate any further feedback 🎉

⚠️ **Limitations**: We mainly provide research service related to machine learning field now, other fields will be added in the future.

🗄️ **Disclaimer**: User behavior data will be collected for the pure research purpose. If you use this demo, you may implicitly agree to these terms.

Arxiv Copilot

Arxiv Copilot summarize latest Arxiv papers based on your profile



Arxiv Copilot is a research prototype demo. Feedback is welcomed!

Research behind Arxiv Copilot

Arxiv Copilot: A Self-Evolving and Efficient LLM System for Personalized Academic Assistance

Guanyu Lin[1 2*], Tao Feng[1*], Pengrui Han[1 3*], Ge Liu[1], Jiaxuan You[1]
[1]University of Illinois at Urbana-Champaign, [2]Carnegie Mellon University, [3]Carleton College
[*]Equal Contribution

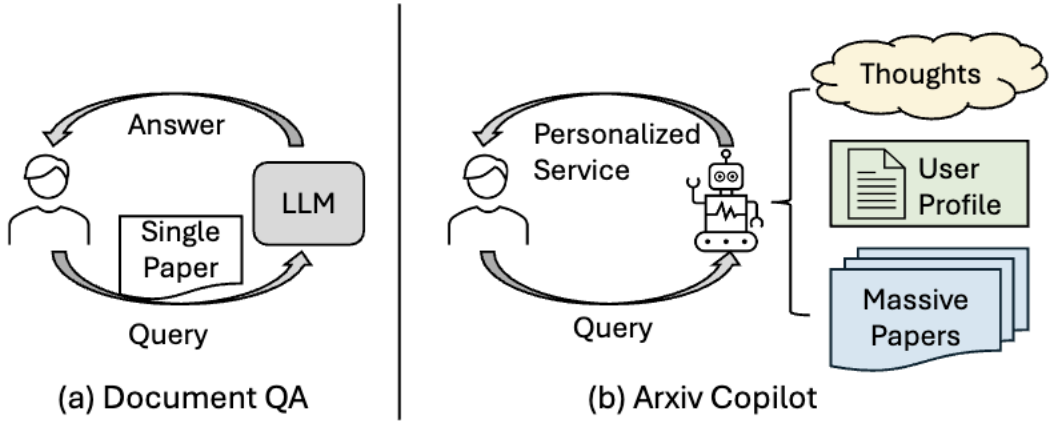


Figure 1: **Comparison of (a) document Question Answering (QA) with our (b) Arxiv Copilot.** Conven-

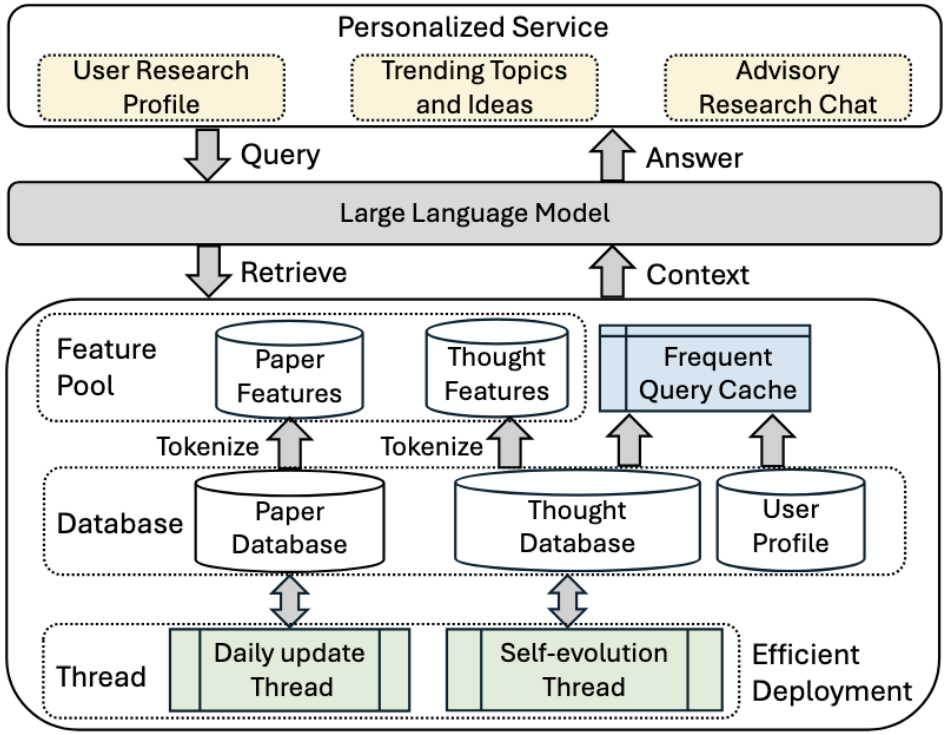


Figure 2: **Architecture of Arxiv Copilot from bottom-to-up perspective.** (a) In personalized service, Arxiv

Tips for Writing Papers

Highly recommend this webpage:

https://cs.stanford.edu/people/widom/paper-writing.html

Jennifer Widom

Frederick Emmons Terman Dean of the School of Engineering
Fletcher Jones Professor in Computer Science and Electrical Engineering

Stanford University

The 5 Questions for A Good Paper

(Originally for writing paper introduction)

- What is the problem?
- Why is it interesting and important?
- Why is it hard? (E.g., why do naive approaches fail?)
- Why hasn't it been solved before? (Or, what's wrong with previous proposed solutions? How does mine differ?)
- What are the key components of my approach and results? Also include any specific limitations.

An Example - GraphRNN

☐ GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models 1035 2018

J You, R Ying, X Ren, WL Hamilton, J Leskovec

International Conference on Machine Learning (ICML 2018)

My first paper during my PhD :p

Abstract: Modeling and generating graphs is fundamental for studying networks in biology, engineering, and social sciences. However, modeling complex distributions over graphs and then efficiently sampling from these distributions is challenging due to the non-unique, high-dimensional nature of graphs and the complex, non-local dependencies that exist between edges in a given graph. Here we propose GraphRNN, a deep autoregressive model that addresses the above challenges and approximates any distribution of graphs with minimal assumptions about their structure. GraphRNN learns to generate graphs by training on a representative set of graphs and decomposes the graph generation process into a sequence of node and edge formations, conditioned on the graph structure generated so far. In order to quantitatively evaluate the performance of GraphRNN, we introduce a benchmark suite of datasets, baselines and novel evaluation metrics based on Maximum Mean Discrepancy, which measure distances between sets of graphs. Our experiments show that GraphRNN significantly outperforms all baselines, learning to generate diverse graphs that match the structural characteristics of a target set, while also scaling to graphs 50 times larger than previous deep models.

An Example - GraphRNN

What is the research problem?

- By Jiaxuan in Aug 2024:

Modeling arbitrary distributions of graphs and sampling from it

- By GPT-4o in Aug 2024 (included full abstract in the prompt):

How can we model and generate complex graph distributions with minimal assumptions while efficiently sampling diverse and large-scale graphs?

An Example - GraphRNN

Why is it interesting and important?

- By Jiaxuan in Aug 2024:

Fundamental in studying graphs across science disciplines

- By GPT-4o in Aug 2024 (included full abstract in the prompt):

It's important because it advances the ability to model and generate complex graphs, which is crucial for understanding networks in various fields, while also offering scalability and accuracy beyond existing methods.

An Example - GraphRNN

Why is it hard? (E.g., why do naive approaches fail?)

- By Jiaxuan in Aug 2024:

Graphs are unordered, high-dimensional, complex, and present non-local dependencies between edges

- By GPT-4o in Aug 2024 (included full abstract in the prompt):

It's hard because graphs are high-dimensional, non-unique, and have complex, non-local dependencies between edges, making it challenging to model and sample from their distributions efficiently.

An Example - GraphRNN

Why hasn't it been solved before? (Or, what's wrong with previous proposed solutions? How does mine differ?)

- By Jiaxuan in Aug 2024:

Classic graph generative models are hand-engineered to model a particular family of graphs, while deep models are either limited to learning from a single graph or generating tiny graphs.

- By GPT-4o in Aug 2024 (included full abstract + part of intro in the prompt):

Previous solutions to generative graph modeling have been limited by their reliance on hand-engineered assumptions, restricting them to specific graph families and failing to capture the full complexity of real-world graphs. Deep generative models have advanced the field but are constrained by their ability to either learn from a single graph or generate only small graphs, typically with fewer than 40 nodes

An Example - GraphRNN

- **What are the key components of my approach and results? Also include any specific limitations.**
- By Jiaxuan in Aug 2024:

The first autoregressive graph generator, modeling graph generation as a sequence of additions of new nodes and edges, and a comprehensive suite of benchmark tasks and baselines for the graph generation problem

- By GPT-4o in Aug 2024 (included full abstract + part of intro in the prompt):

GraphRNN, is a deep autoregressive model designed to generate graphs by sequentially adding nodes and edges, capturing complex joint probabilities. Key components include:

Hierarchical RNNs: A graph-level RNN generates nodes, and an edge-level RNN generates edges.

BFS Node-Ordering: Improves scalability and reduces computational complexity.

MMD-Based Evaluation: Provides rigorous quantitative evaluation of generated graphs.

Results: GraphRNN outperforms baselines, generating high-quality graphs up to 50× larger than previous models, with an 80%-90% reduction in MMD.

Project Writing Tasks

Week	Date	Knowledge learning	Research training	Events	Deadlines
1	Aug 28 Wed	Introduction	Paper reading & analysis		
	Aug 30 Fri	Graph learning tasks	Paper reading & analysis	Writing task, out	
2	Sept 04 Wed	"Shallow" graph learning	Paper reading & analysis		
	Sept 06 Fri	Graph neural networks: perspective	Paper reading & analysis		
3	Sept 11 Wed	Graph neural networks: model I	Paper reading & analysis		
	Sept 13 Fri	Graph neural networks: model II	Paper reading & analysis		Writing task due
4	Sept 18 Wed	Paper reading discussions	Ideate & discussion		
	Sept 20 Fri	Graph neural networks: objective	Ideate & discussion	Proposal task, out	
5	Sept 25 Wed	Graph neural networks: pipeline	Ideate & discussion		
	Sept 27 Fri	Graph neural networks: theory	Ideate & discussion		
6	Oct 02 Wed	Graph neural networks: add-ons	Ideate & discussion		
	Oct 04 Fri	GNN implementation: PyG & GraphGym	Ideate & discussion		Proposal due
7	Oct 09 Wed	Project idea discussions	Prototype implementation		
	Oct 11 Fri	Beyond simple graphs: heterogeneous graphs	Prototype implementation	Submission task, out	

Project Task – Paper Reading and Analysis

15% of final grade:

- **Pick 5 recent research papers** related to graphs in the suggested paper sources (NeurIPS 2023, ICML 2024, ICLR 2024, LOG 2023, Arxiv 2024), pick as diverse as possible
- **Read the paper.** Search / Ask LLM tool for relevant concepts to help you understand the paper, if needed.
- **Answer the 5 questions for each paper**
- Choose your favorite LLM tool, **ask them to answer the question**
- Short comment on **how LLM benefits you** and **you can help LLM improve**
- **Suggested DDL: Sept 15**
- We will announce the submission link next week.

Summary

- Choice of a graph representation:
  - Directed, undirected, bipartite, weighted, adjacency matrix
- Different types of tasks:
  - Node level
  - Edge level
  - Graph level
- Tips for reading research papers