

Introduction to Graph Learning

CPSC483: Deep Learning on Graph-Structured Data

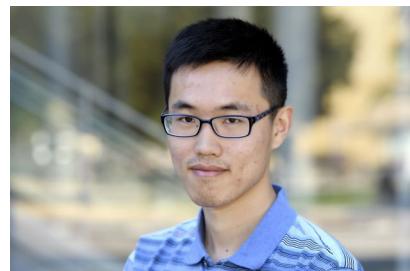
Rex Ying

CPSC483 Course Logistics

- **Welcome to CPSC 483 / 583!**
- **The class meets Monday and Wednesday 4 PM-5:15 PM**
 - Most of the lectures will be in-person
 - There will be 2-3 guest lectures, some of which might be held remotely.
(including GNN library and applications of graph learning)
- **This is a (relatively) new course, and feedbacks are especially welcomed!**
 - Lecture style, homework, projects ...
 - Many of the content comes from relatively recent research (>2017)

Logistics: Teaching Staff

Instructor



Rex Ying

Teaching Assistants

Jialin Chen (Head TA)

Agastya Rana

Yuhang Chen

Weikang Qiu

Mahdi Soleimani (tentative)

Guest Lecture

Spectral GNNs

AI for science

Course Outline

Lecture	Topic	Lecture	Topic	Lecture	Topic
1	Introduction	10	GNN Expressive Power	19	Explaining GNN Predictions
2	Graph Learning Tasks	11	PyTorch Geometric Library (virtual)	20	Learning on Dynamic Graphs
3	Deep Learning Background	12	Graph Generative Models	21	Spectral GNNs
4	Graph Neural Networks Designs	13	Graph Transformers	22	Knowledge Graph Reasoning
5	GNN Implementations, Objectives and Loss Functions	14	Self-supervised Learning with GNNs	23	Knowledge Graph Embeddings in Hyperbolic Spaces
6	Graph Attention Networks	15	Foundation models and Pre-trained GNNs	24	Applications of Graph Learning in Biology and Medicine
7	Scalable GNN Architectures	16	Distributed Node Embeddings	25	Applications in Graphics, Robotics and Scientific Simulations
8	Multi-hop GNNs and Multi-hop Atentions	17	Geometric Embeddings: Order and Box Embeddings		
9	Theory of Graph Neural Networks	18	Hyperbolic Embeddings and Hyperbolic GNNs		The syllabus, slides and reading will be available on course website

Logistics: Website

- **Course Website**

- <https://graph-and-geometric-learning.github.io/cpsc483-583-website-23fall>
- Slides will be posted before or shortly after the class

- **Readings:**

- [Graph Representation Learning Book by Will Hamilton](#)
- Research papers

- **Optional readings:**

- Papers and pointers to additional literature
- **These will be very useful for course projects**

- **PyTorch Geometric Readings**

- Python [Documentation](#)
- [GitHub](#) Repository (for the latest development)

Logistics: Communication

- **Ed Q&A website:**
 - use Ed Discussion (link integrated on Canvas) Platform
 - **Please participate and help each other!**
 - Don't post code, annotate your questions, search for answers before you ask
- We will post course announcements to the course website (make sure you check it regularly)
- Office Hours (also posted and updated on the website)
 - Instructor: Monday 3-4 pm
 - TAs: see course website and Canvas

Work for the Course & Grading

- **Final grade will be composed of:**
 - **Homework: 32%**
 - Homework 1, 2, 3, 4, each worth 8%
 - **Coding assignment: 18%**
 - 4 (short) coding assignments using Google Colab / Jupyter Notebook
 - **Exam (during the last lecture): 20%**
 - Light-weight **1 hour** exam (closed book).
 - **Course project: 30%**
 - Proposal: 10%
 - Final report: 20%
 - **Extra credit: code contribution (up to 10%)**
 - Opensource contributions: [PyG](#), new library / public benchmarks related to the course
 - Used if you are on the boundary between grades



The assignments will be short.
I'm expecting 1-2 hours each
(If pre-reqs are met)

Homework, Write-ups

- **Assignments take time. Start early!**
 - A combination of data analysis, algorithm design, and math
 - Generally due on Friday 23:59 Eastern Time
- **How to submit?**
 - Canvas
 - **Both homework (including code) and project deliverables (pdf) must be uploaded**
 - Code: ipynb file; written assignment: pdf file
 - We prefer LaTeX submissions for written assignments.
 - If you really need to use hand-writing, make sure it's **readable**.
 - TAs have no obligations to use magnifying glasses to figure out what you write
- **Total of 2 Late Periods (LP) per student (up to 5 working days):**
 - Max 1 late period per assignment (no LP for the final report)

Honor Code

- **We strictly enforce the Yale Honor Code**

- Violations of the Honor Code include:
 - Copying or allowing another to copy from one's own paper
 - Unpermitted collaboration
 - Plagiarism
 - Giving or receiving unpermitted aid on an examination
 - Representing as one's own work the work of another
 - Giving or receiving aid on an assignment under circumstances in which a reasonable person should have known that such aid was not permitted
- The sanction for even a first offense is severe

Course Projects

- **Course project:**
 - Reproduce, analyze and compare at least **2 methods** for the same task (on at least **2 large-scale datasets**)
 - Propose new algorithms / architectures and validate on at least 2 datasets
- **Performed in groups of up to 2 students:**
 - Individual project is highly recommended.
 - It's fine to have groups of 2. The team size will be taken under consideration when evaluating the scope of the project in breadth and depth.
 - In the final report, each student in a group will write their **own method and experiment sections**
 - More information will be posted on the course website in 3 weeks

Course Projects

- Project is the **important work** for the class
 - Can be very beneficial if you aim to work on research in this topic
 - **Consult with the teaching staff** if you have questions
-
- You **can** collaborate with PI / students in research lab if you are using your research project as the final project.
 - Make sure you **declare which part is your work** (only mention other collaborators' work in context/background section)

Course Schedule

Week	Assignment	Due on (11:59pm PT)
3	Homework 1	Fri, Sep 15 th
4	Colab 1	Fri, Sep 22 nd
	Homework 2	Fri, Sep 29 th
	Colab 2	Fri, Oct 06 th
	Project Proposal	TBA
	Homework 3	Fri, Nov 03 rd
	Colab 3	Fri, Nov 10 th
	Project Milestone	TBA
	Homework 4	Fri, Nov 17 th
	Colab 4	Fri, Nov 24 th
	Project Report	Fri, December 8 th (No Late Period for this)

Prerequisites

- The course has relevance to a wide range of topics. Background in graph or ML is always helpful!
- Minimum Pre-requisites
CPSC 201, 223 and one of 365 and 366
 - Nice-to-have background
(have taken a course or have done research)
 - Machine Learning (minimum: high familiarity with linear algebra and multi-variable calculus)
 - Algorithms and graph theory
 - Probability and statistics
 - Programming:
 - You should have familiarity with programming (in **Python**)
 - Knowledge about common libraries such as Numpy, Pandas etc. is a plus
 - You should have used **PyTorch** (or equivalent) before. Learning it is not too hard if you have strong background.

Graph Machine Learning Tools

- We use **PyTorch Geometric (PyG)**
- Other network analytics tools: [SNAP.PY](#), [NetworkX](#), [Graph-Tool](#), [Graphvis](#)
- [Pytorch Lightning](#) is a good training framework
- It is encouraged to read and reproduce the research code for papers on Github

Yale

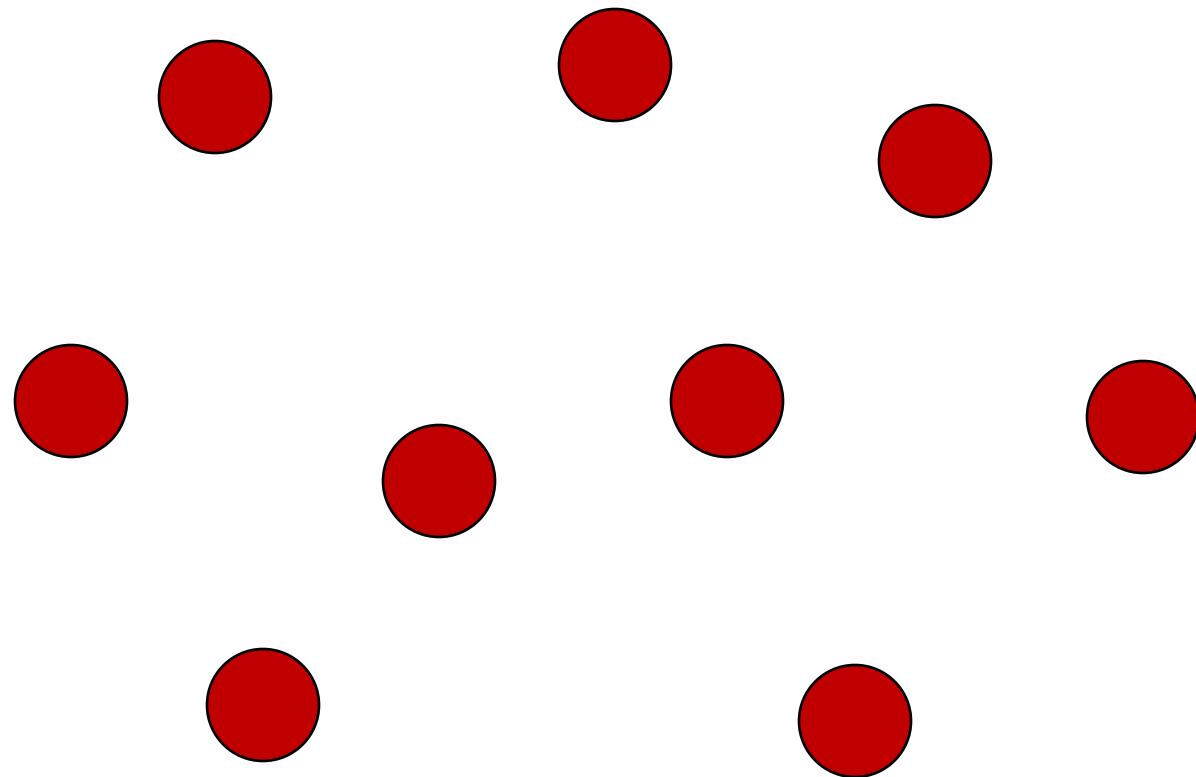
Introduction to Graph and Machine Learning on Graphs

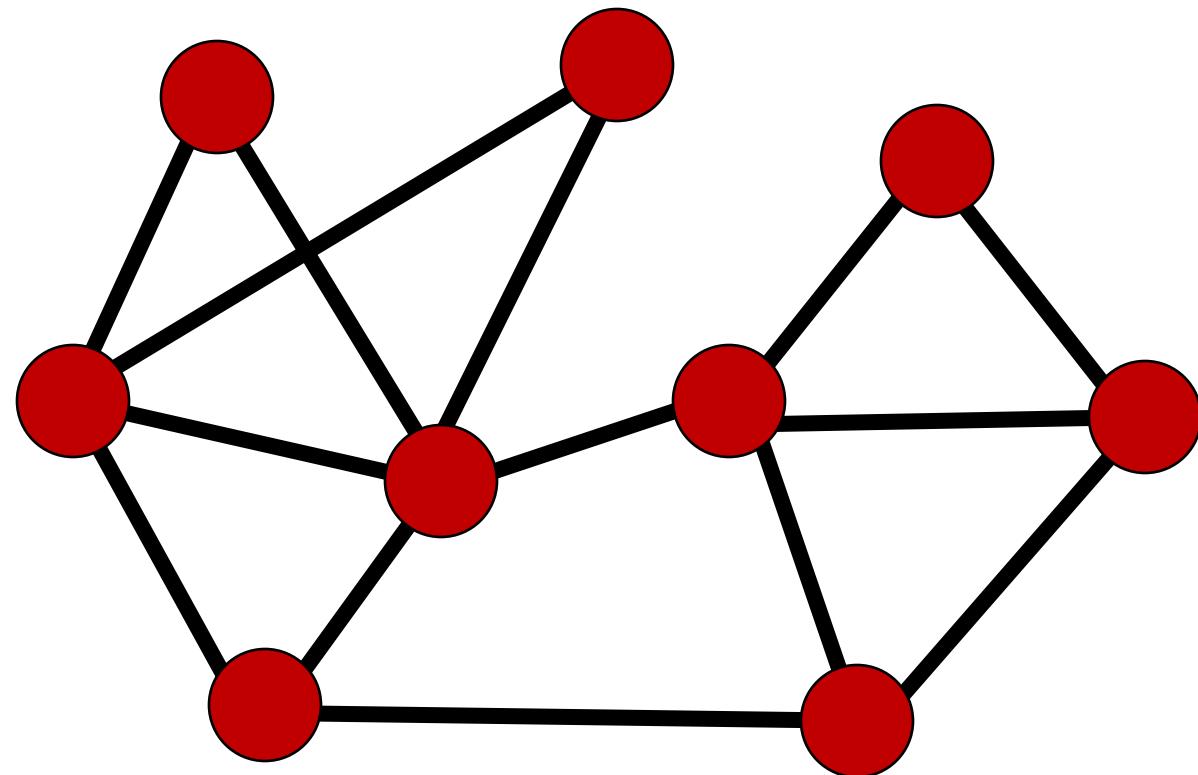
CPSC483: Deep Learning on Graph-Structured Data

Rex Ying

Why Graphs?

Graphs are a general language for describing and analyzing entities with relations/interactions





Graph

Types of Networks and Graphs (1)

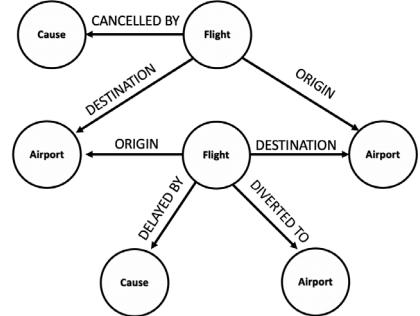
- **Networks (also known as Natural Graphs):**
 - **Social networks:**
 - **Society** is a collection of 7+ billion individuals
 - **Communication and transactions:**
 - Electronic devices, phone calls, financial transactions
 - **Biomedicine:**
 - Interactions between **genes/proteins** regulate life
 - **Brain connections:**
 - Our **thoughts** are hidden in the connections between billions of neurons

Types and Networks and Graphs (2)

- **Graphs (as a representation):**
 - **Information, knowledge** are organized and linked
 - **Software, code** can be represented as a graph
 - **Similarity networks:** Connect similar data points
 - **Relational structures:** Molecules, Scene graphs, 3D shapes, Particle-based physics simulations

The distinction between networks & graphs is often blurred.

Many Types of Data are Graphs (1)

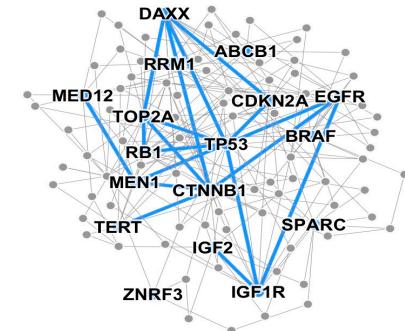


Event Graphs



Image credit: [SalientNetworks](#)

Computer Networks



Disease Pathways

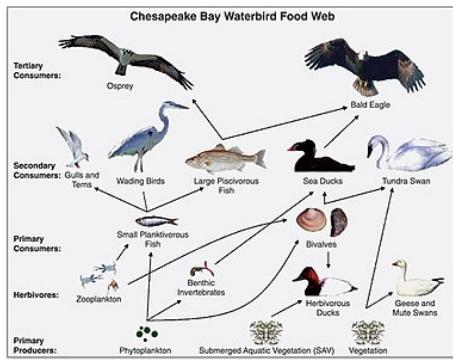


Image credit: [Wikipedia](#)

Food Webs



Image credit: [Pinterest](#)

Particle Networks



Image credit: [visitlondon.com](#)

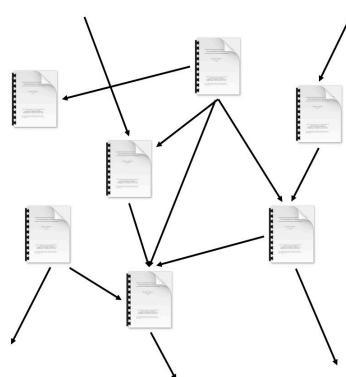
Underground Networks

Many Types of Data are Graph (2)



Image credit: [Medium](#)

Social Networks



Citation Networks

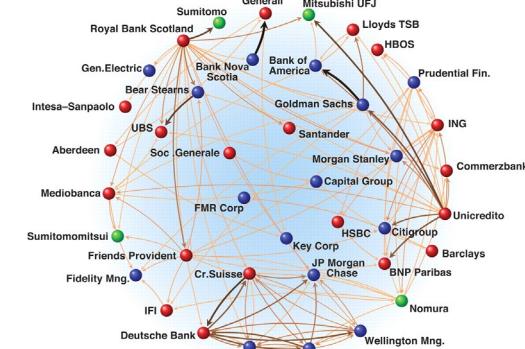


Image credit: [Science](#)

Economic Networks



Image credit: [Missoula Current News](#)

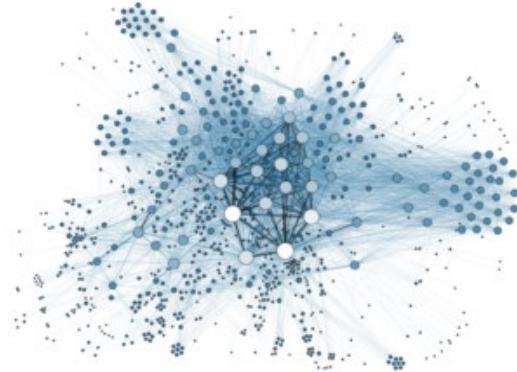


Image credit: [Lumen Learning](#)

Communication Networks

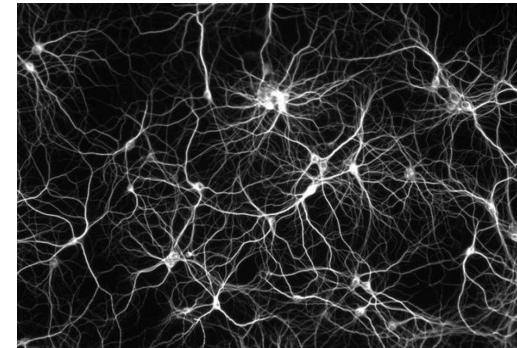


Image credit: [The Conversation](#)

Many Types of Data are Graph (3)

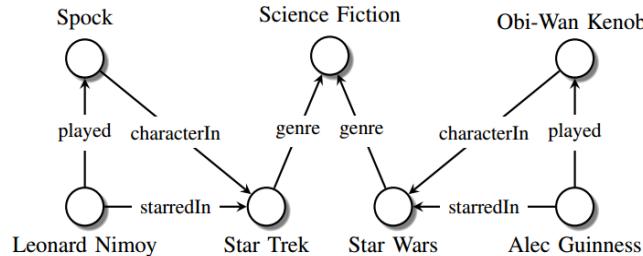


Image credit: [Maximilian Nickel et al](#)

Knowledge Graphs

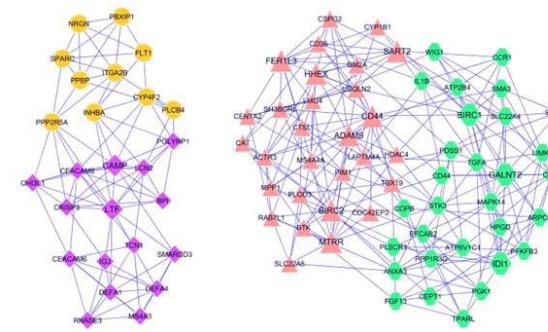


Image credit: [ese.wustl.edu](#)

Regulatory Networks

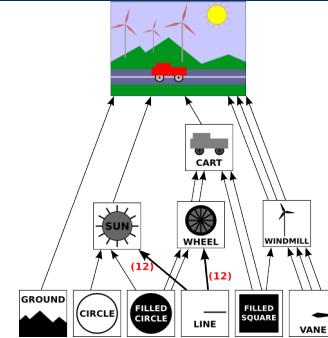


Image credit: [math.hws.edu](#)

Scene Graphs

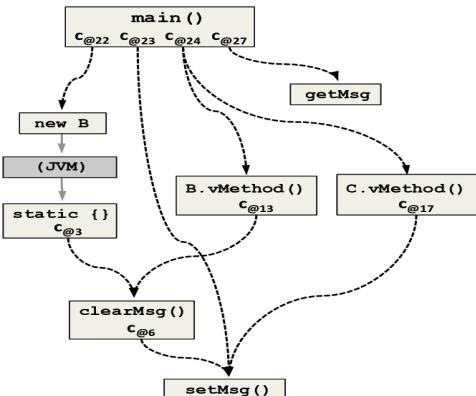


Image credit: [ResearchGate](#)

Code Graphs

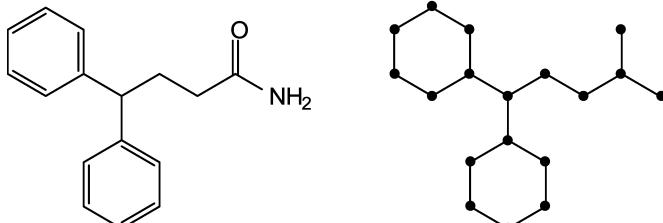


Image credit: [MDPI](#)

Molecules

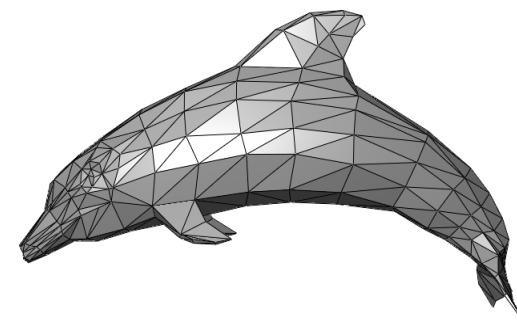


Image credit: [Wikipedia](#)

3D Shapes

Graphs and Relational Data

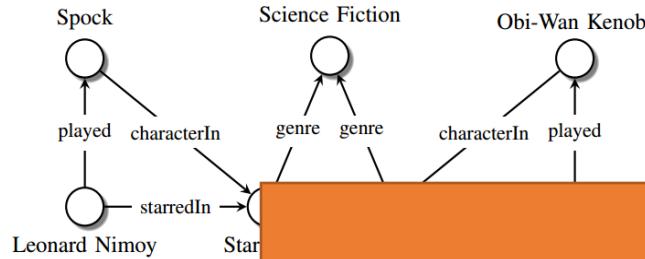


Image credit:

Know

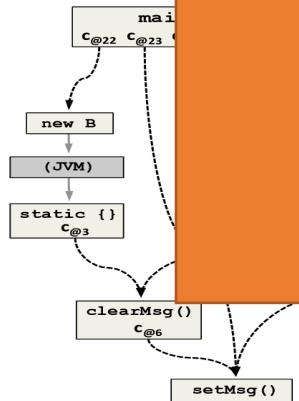


Image credit: [ResearchGate](#)

Code Graphs

Main question:

How do we take advantage of relational structure for better prediction?

Image credit: [MDPI](#)

Molecules

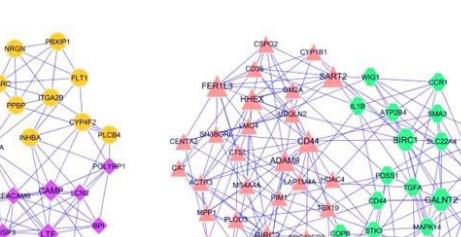


Image credit: [math.hws.edu](#)

Graphs

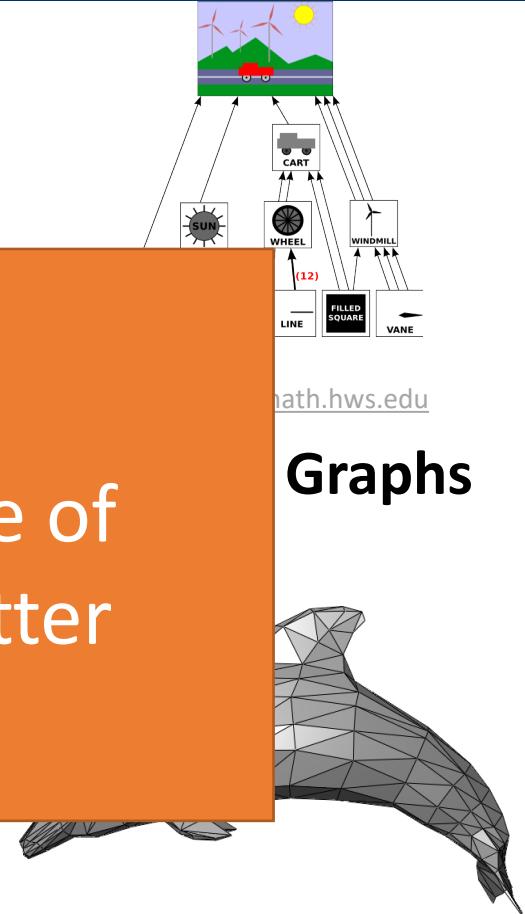


Image credit: [Wikipedia](#)

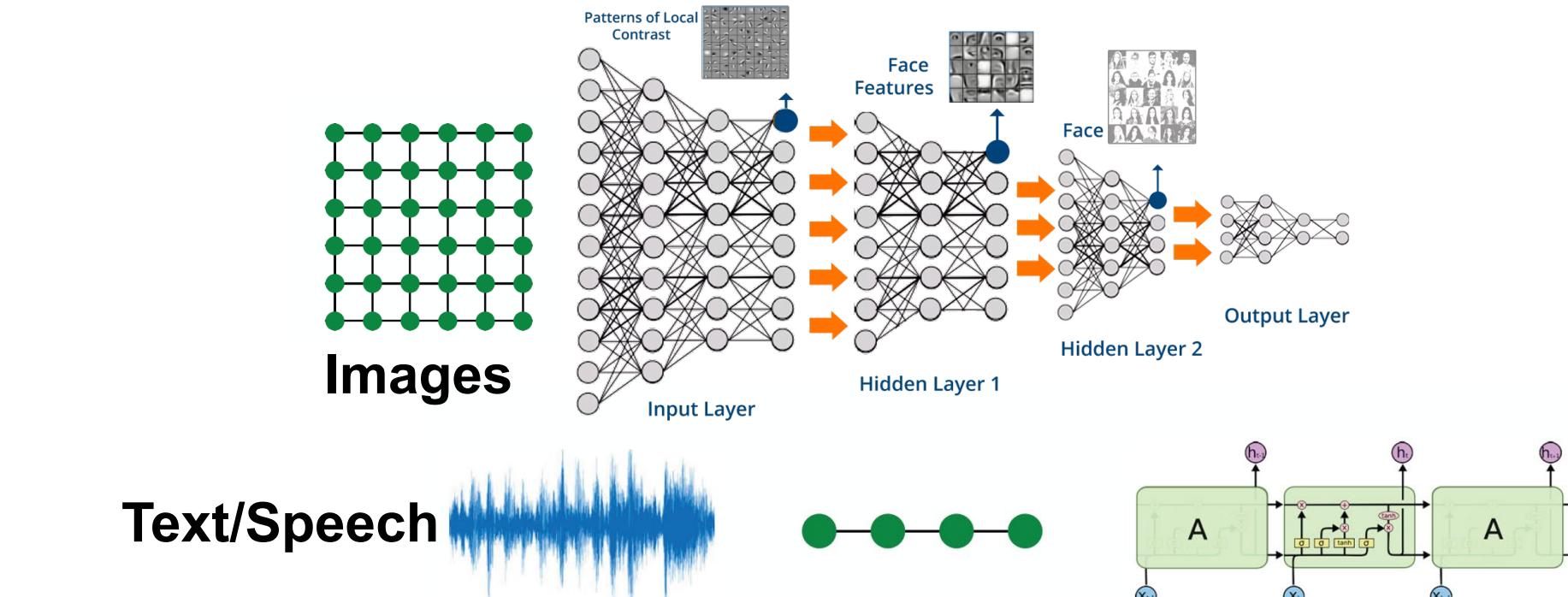
3D Shapes

Graphs: Machine Learning

Complex domains have a rich relational structure, which can be represented
as a
relational graph

By explicitly modeling relationships we achieve better performance!

Modern ML Toolbox

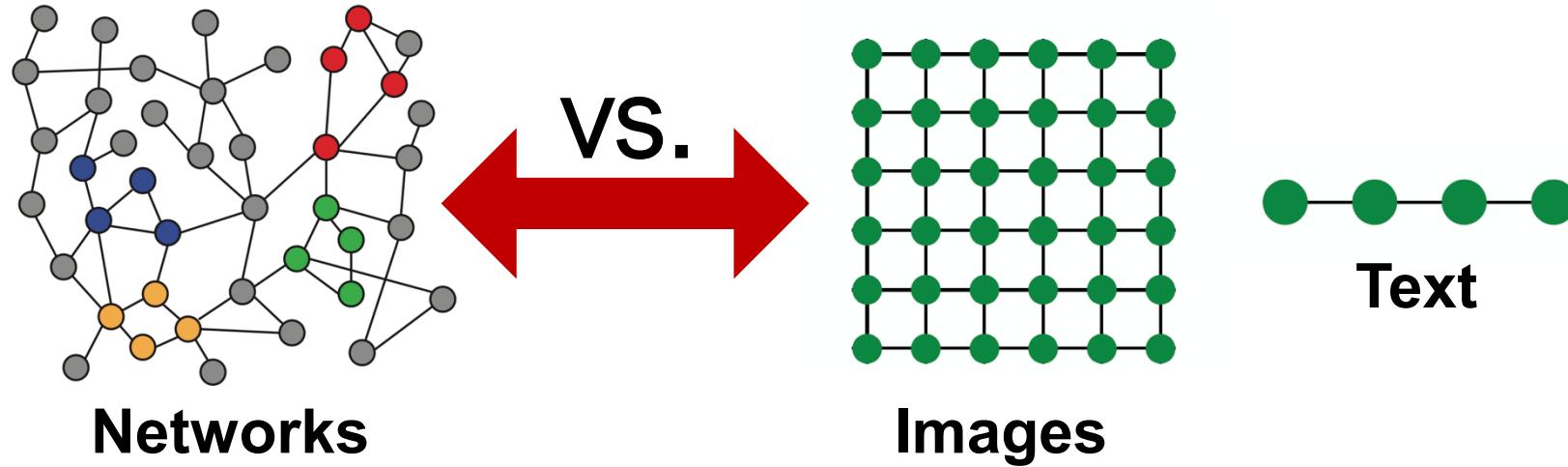


Modern deep learning toolbox is designed
for simple sequences & grids

Why is it Hard?

Networks are complex.

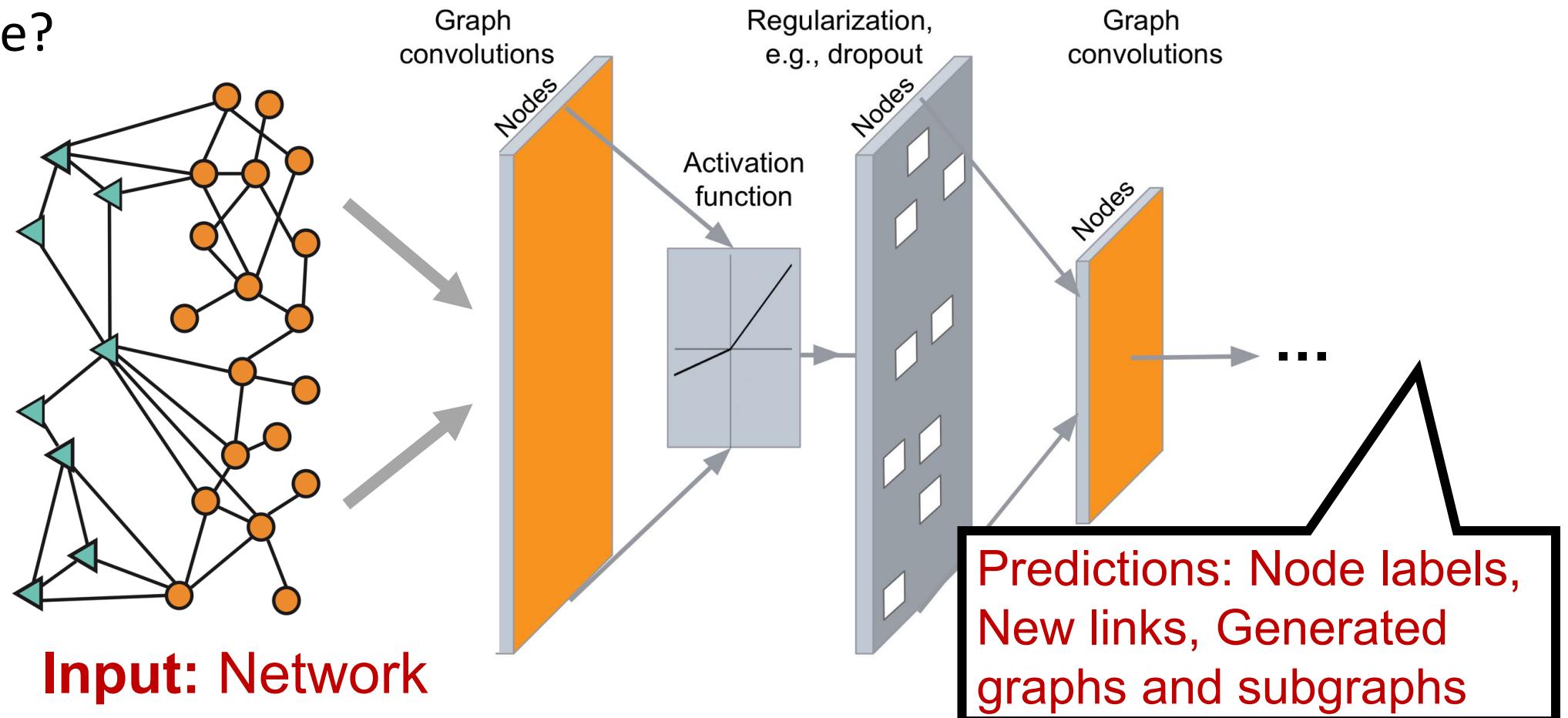
- Arbitrary size and complex topological structure (*i.e.*, no spatial locality like grids)



- No fixed node ordering or reference point
- Often dynamic and have multimodal features

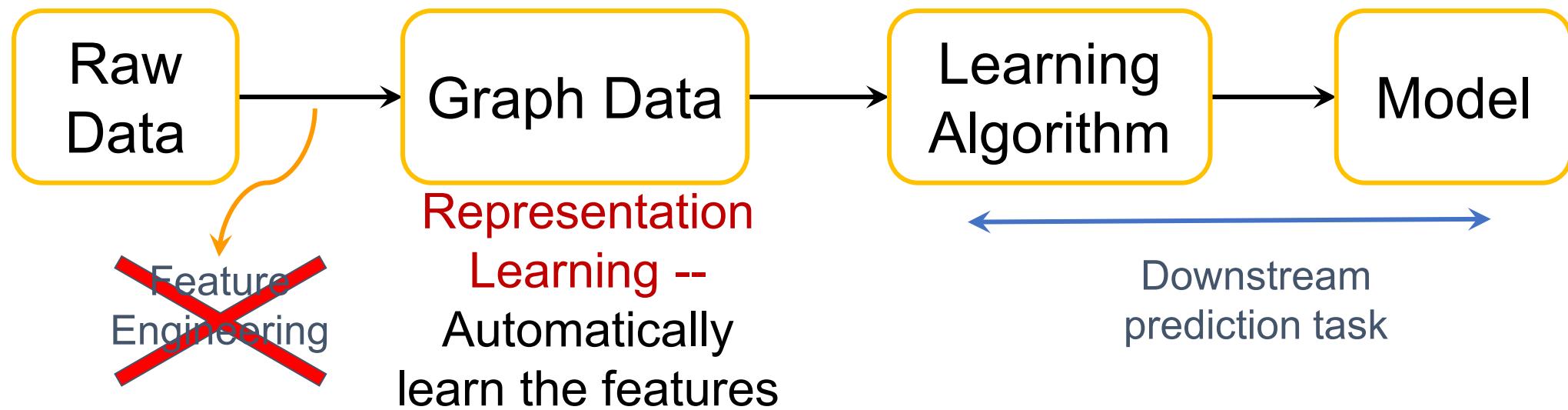
CPSC483: Deep Learning on Graph-Structured Data

- How can we develop neural networks that are much more broadly applicable?



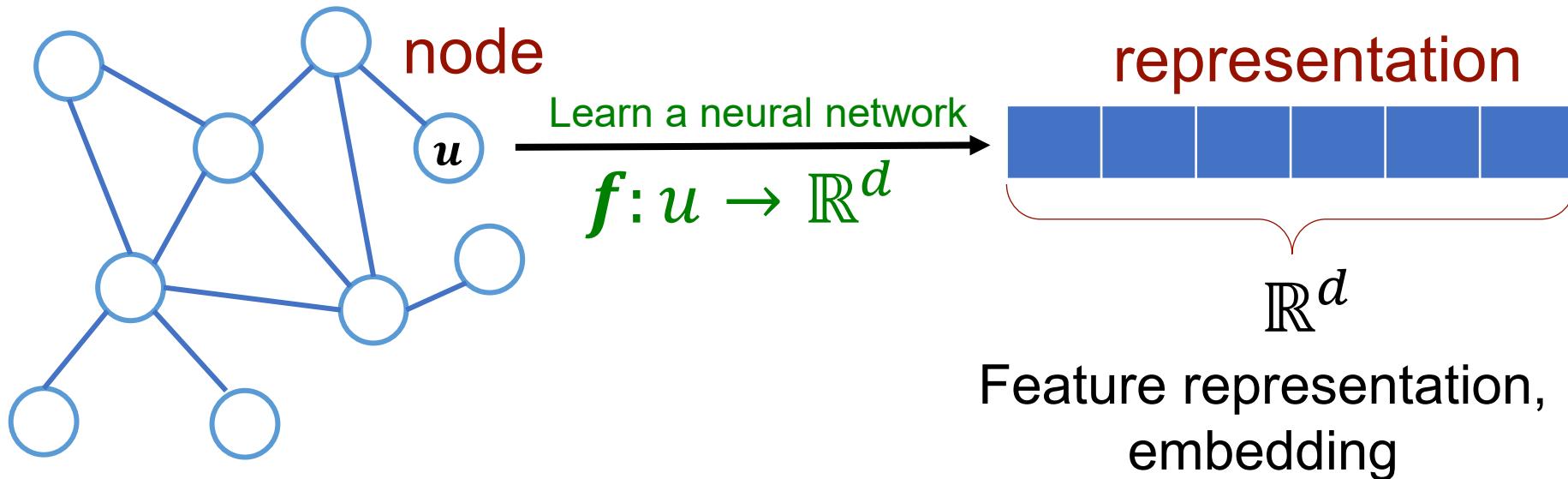
CPSC483 & Representation Learning (1)

- Conventional Supervised Machine Learning Lifecycle
 - Requires heavy feature engineering



CPSC483: Representation Learning (2)

Map nodes to d-dimensional **embeddings** such that **similar nodes** in the network are **embedded close together**



CPSC483 Course Outline

We are going to cover various topics in Machine Learning and Representation Learning for graph structured data:

- Traditional methods: Graphlets, Graph Kernels
- Graph Neural Networks: GCN, GraphSAGE, GAT, Design space of GNNs, Theory of GNNs, Graph Transformers
- Methods for node embeddings: DeepWalk, Node2Vec
- Knowledge graphs and reasoning: TransE, BetaE, ConE
- Different types of inputs: Dynamic Graphs, Heterogeneous Graphs
- Deep generative models for graphs
- Applications to Biomedicine, Science, Industry

Topics Covered in CPSC483/583 (1)

Lecture	Topic	Lecture	Topic	Lecture	Topic	
1	Background: Machine Learning and Graphs as Data Structure	10	GNN Expressive Power	19	Explaining GNN Predictions	
2	Plethora of Tasks and Features on Graphs	11	PyTorch Geometric Library (virtual)	20	Learning on Dynamic Graphs	
3	Graph Neural Networks Models	12	Graph Generative Models	21	Spectral GNNs	
4	Graph Neural Networks Designs	13	Graph Transformers	22	Knowledge Graph Reasoning	
5	GNN Implementations, Objectives and Loss Functions	14	Self-supervised Learning with GNNs	23	Knowledge Graph Embeddings in Hyperbolic Spaces	
6	Graph Attention Networks	15	Foundation models and Pre-trained GNNs	24	Applications of Graph Learning in Biology and Medicine	
7	Scalable GNN Architectures	16	Distributed Node Embeddings	25	Applications in Graphics and Scientific Simulations	
8	Multi-hop GNNs and Multi-hop Atentions	17	Geometric Embeddings: Order and Box Embeddings	The syllabus, slides and reading will be available on course website		
9	Theory of Graph Neural Networks	18	Hyperbolic Embeddings and Hyperbolic GNNs			

Topics Covered in CPSC483/583 (2)

Lecture	Topic	Lecture	Topic	Lecture	Topic
1	Background: Machine Learning and Graphs as Data Structure	GNN Fundamentals	10	GNN Expressive Power	Real-world, Large-scale Applications
2	Plethora of Tasks and Features on Graphs		11	PyTorch Geometric Library (virtual)	
3	Graph Neural Networks Models		12	Graph Generative Models	
4	Graph Neural Networks Designs		13	Graph Transformers	
5	GNN Implementations, Objectives and Optimizations		14	Semi-supervised Learning with GNNs	
6	Graph Attention Networks		15	Foundation models and Pre-trained GNNs	
7	Scalable GNN Architectures		16	Distributed Node Embeddings	
8	Multi-hop GNNs and Multi-hop Attentions		17	Geometric Embeddings: Order and Box Embeddings	
9	Theory of Graph Neural Networks		18	Hyperbolic Embeddings and Hyperbolic GNNs	

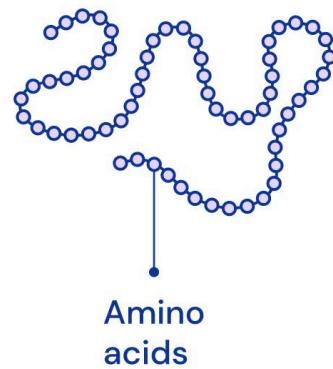
The syllabus, slides and reading will be available on course website

Example of Graph Machine Learning Tasks

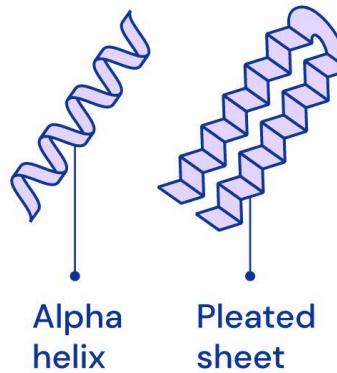
Example 1: Protein Folding

A protein chain acquires its native 3D structure

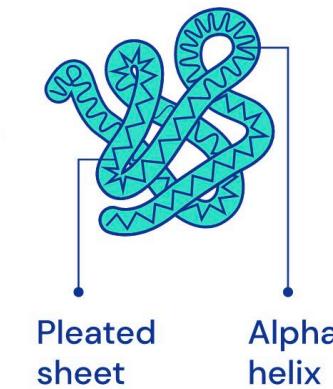
Every protein is made up of a sequence of amino acids bonded together



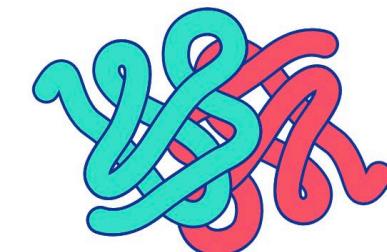
These amino acids interact locally to form shapes like helices and sheets



These shapes fold up on larger scales to form the full three-dimensional protein structure



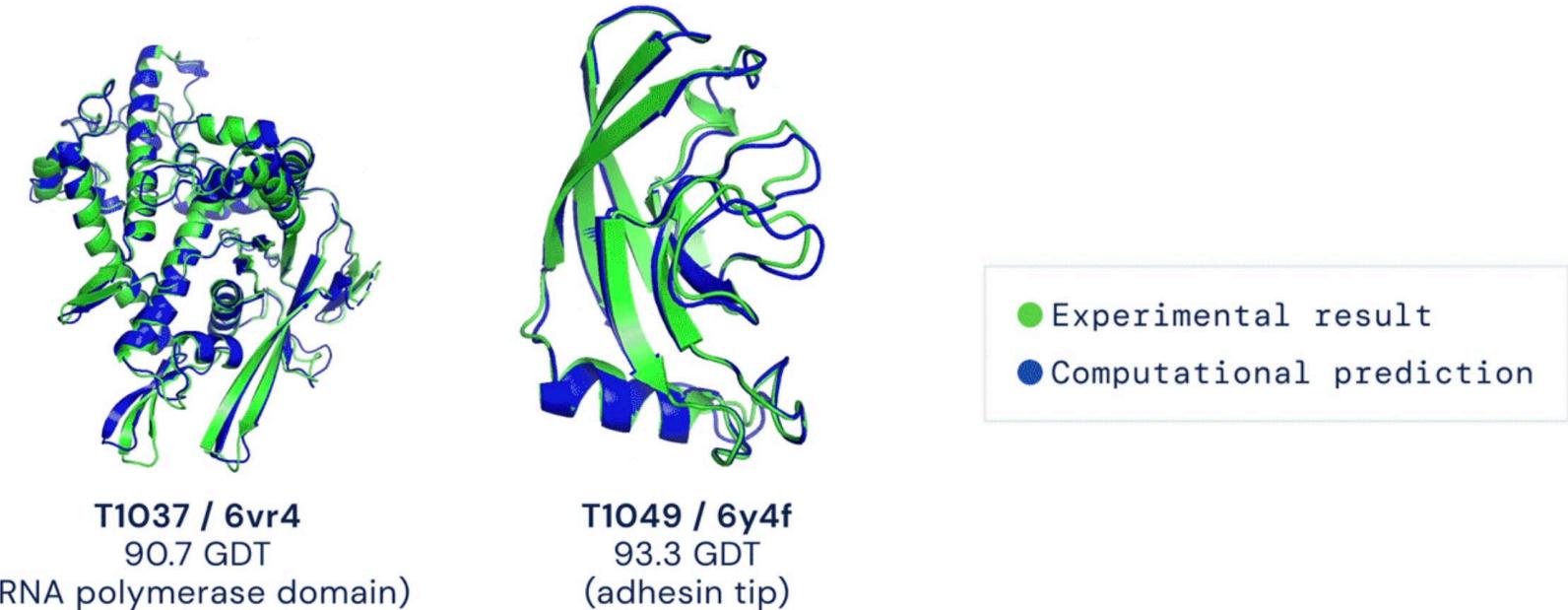
Proteins can interact with other proteins, performing functions such as signalling and transcribing DNA



AW Senior et al. Improved protein structure prediction using potentials from deep learning.

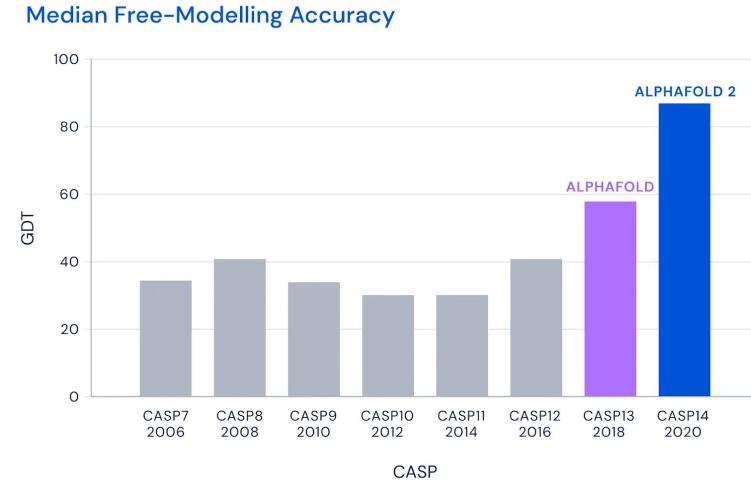
The Protein Folding Problem

Computationally predict a protein's **3D structure** based solely on its amino acid **sequence**



AW Senior *et al.* Improved protein structure prediction using potentials from deep learning.

AlphaFold



AlphaFold's AI could change the world of biological science as we know it

DeepMind's latest AI breakthrough can accurately predict the way proteins fold

Has Artificial Intelligence 'Solved' Biology's Protein-Folding Problem?

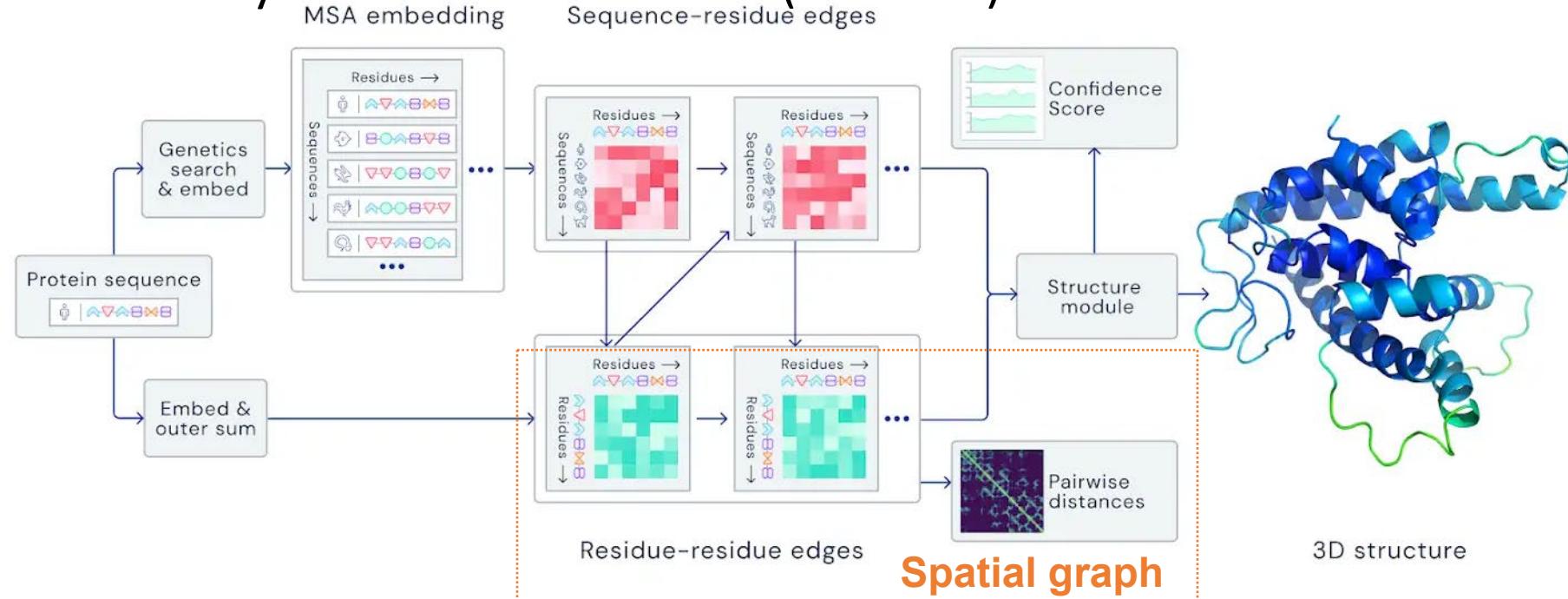
DeepMind's latest AI breakthrough could turbocharge drug discovery

AW Senior *et al.* Improved protein structure prediction using potentials from deep learning.

AlphaFold: Solving Protein Folding

- Key Idea: *Spatial Graph*

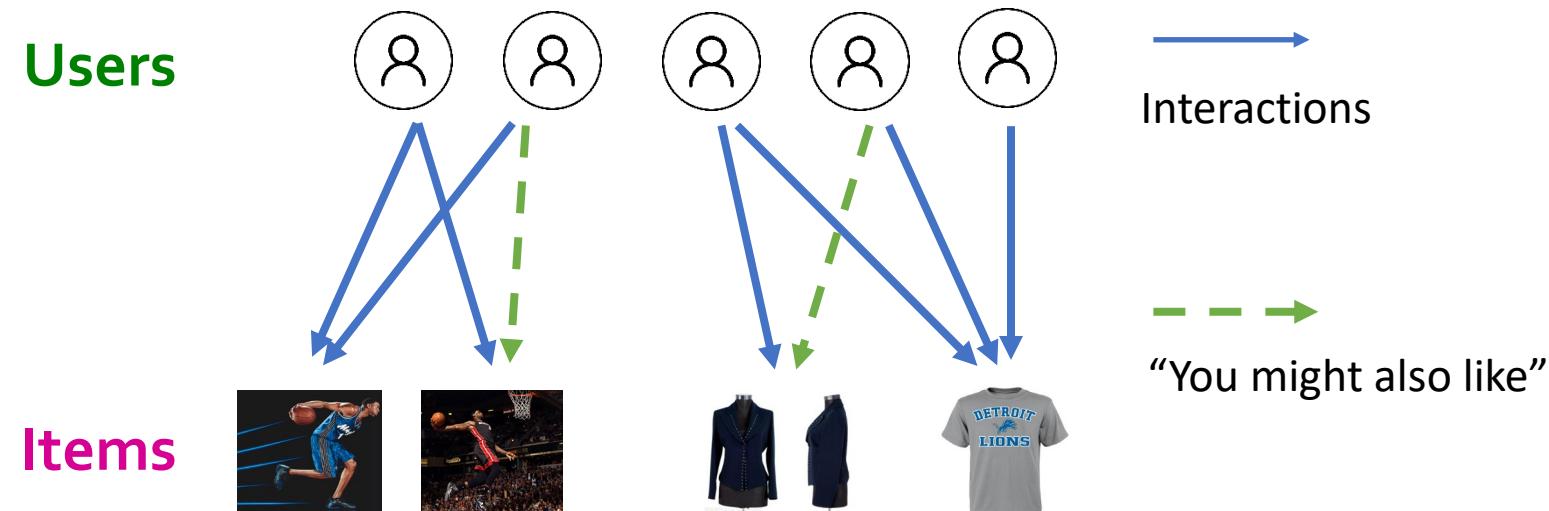
- Nodes: Amino acids in a protein sequence
- Edges: Proximity between amino acids (residues)



AW Senior et al. Improved protein structure prediction using potentials from deep learning.

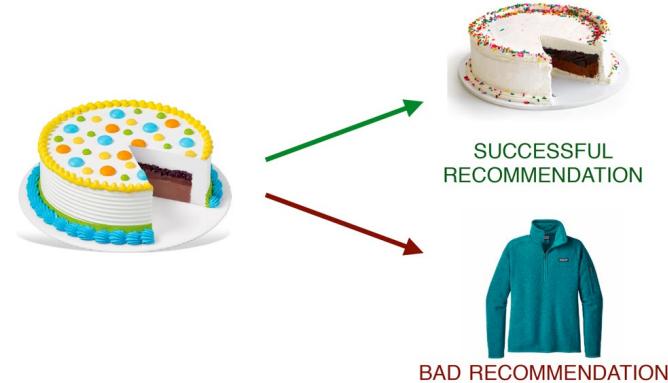
Example 2: Recommender Systems

- **Users interacts with items**
 - Watch movies, buy merchandise, listen to music
 - **Nodes:** Users and items
 - **Edges:** User-item interactions
- **Goal: Recommend items users might like**



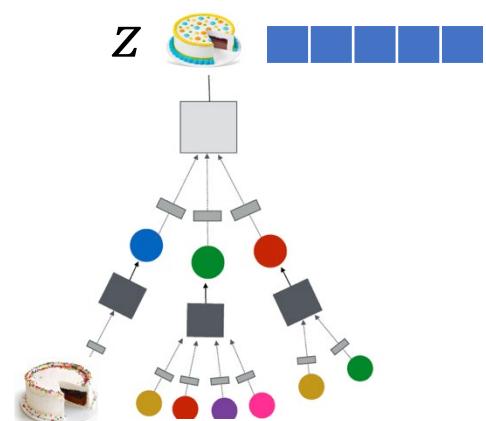
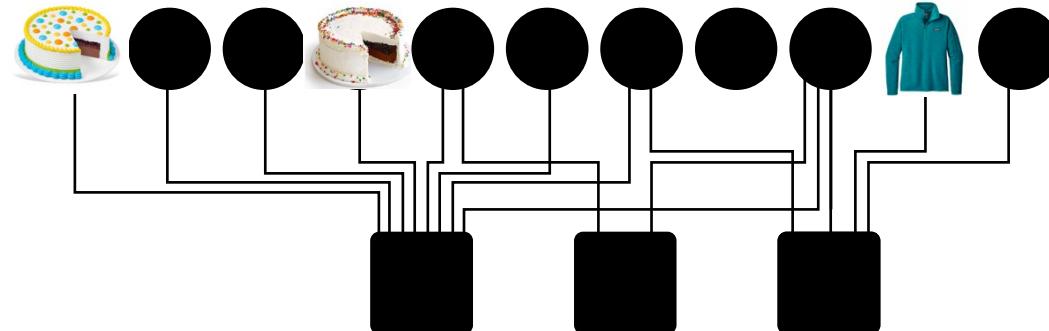
PinSage: Grpah-based Recommender

- Task: Recommend related pins to users



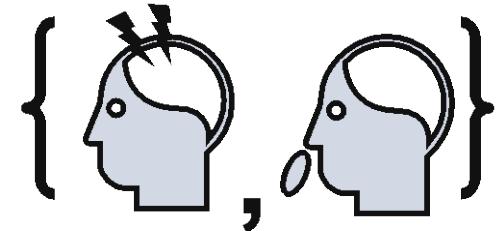
Task: Learn node embeddings z_i such that
 $d(z_{cake1}, z_{cake2}) < d(z_{cake1}, z_{sweater})$

- Predict whether two nodes in a graph are related



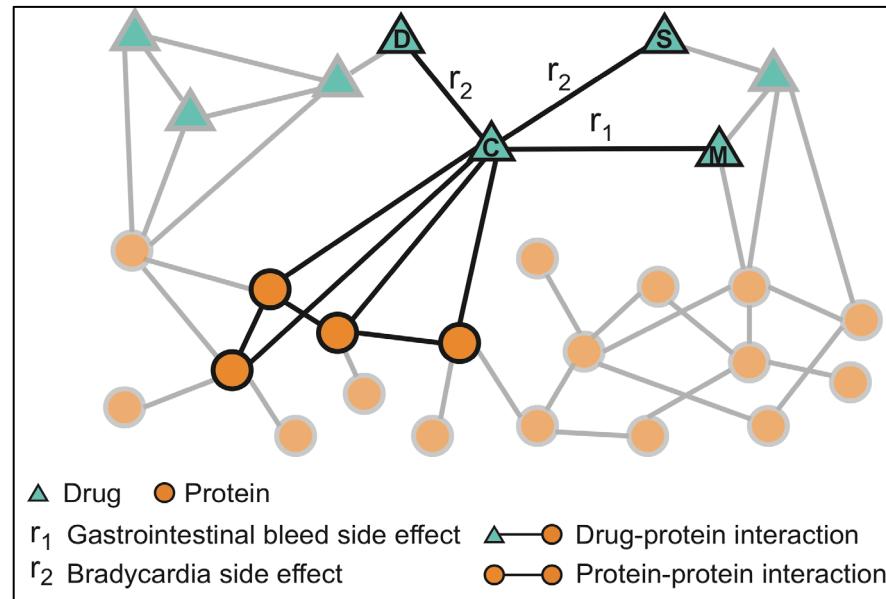
Example 3: Drug Side Effects

- Many patients **take multiple drugs** to treat **complex or co-existing diseases**:
 - 46% of people ages 70-79 take more than 5 drugs
 - Many patients take more than 20 drugs to treat heart disease, depression, insomnia, etc.
- **Task:** Given a pair of drugs predict adverse side effects

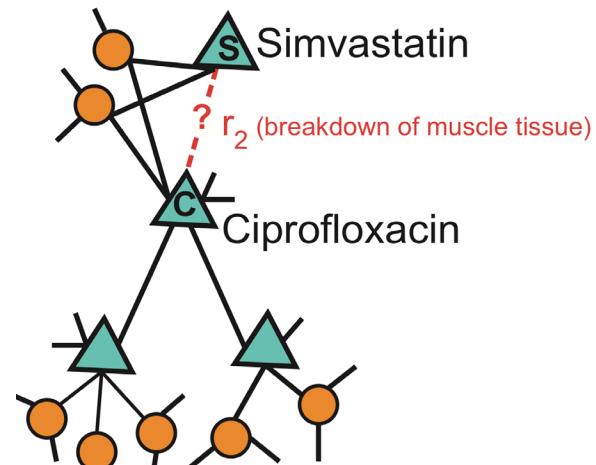


Biomedical Graph Link Prediction

- **Nodes:** Drugs & Proteins
- **Edges:** Interactions



- **Query:** How likely will Simvastatin and Ciprofloxacin, when taken together, break down muscle tissue?



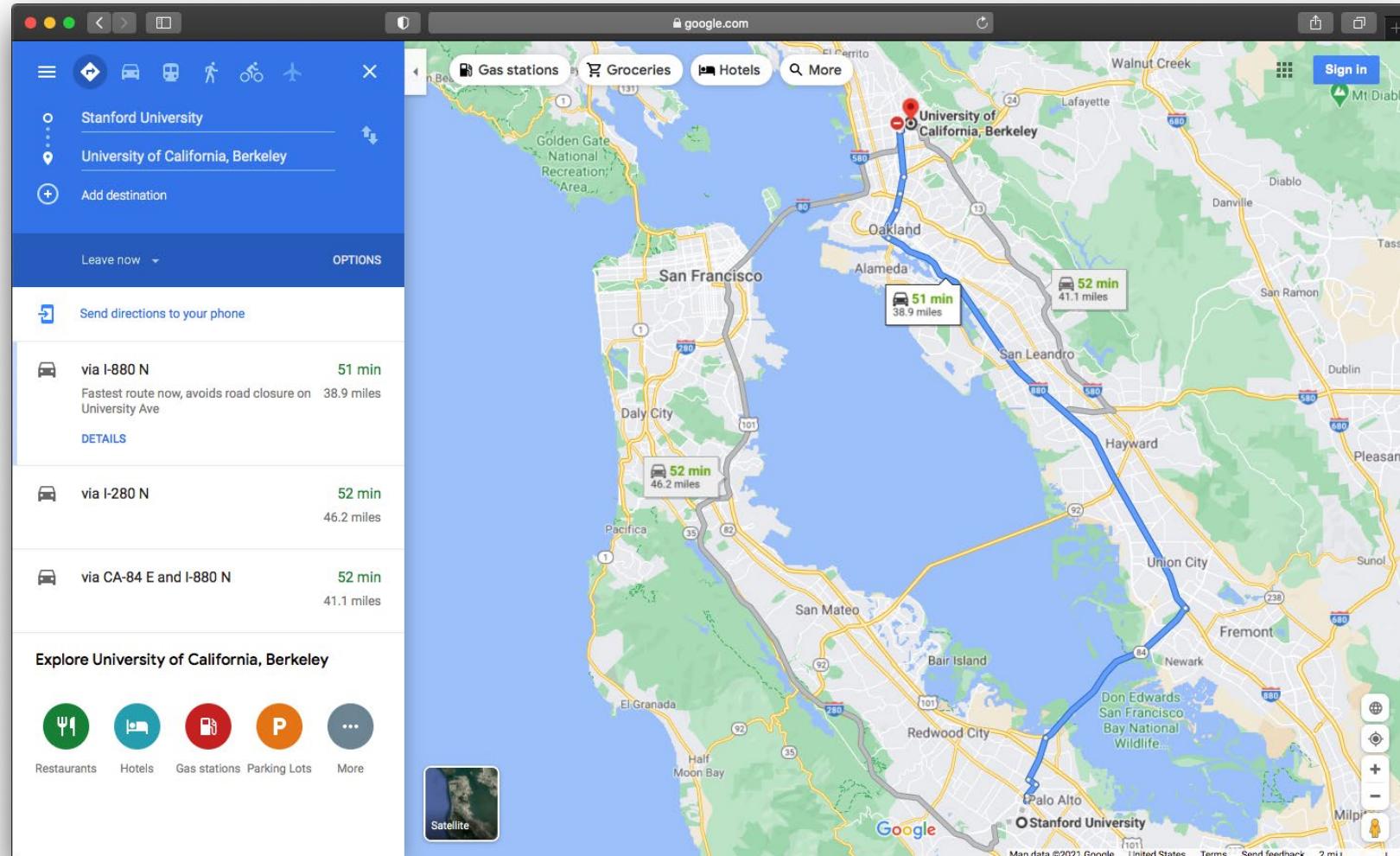
Results: De novo Predictions

Rank	Drug c	Drug d	Side effect r	Evidence found
1	Pyrimethamine	Aliskiren	Sarcoma	Stage <i>et al.</i> 2015
2	Tigecycline	Bimatoprost	Autonomic neuropathy	
3	Omeprazole	Dacarbazine	Telangiectases	
4	Tolcapone	Pyrimethamine	Breast disorder	Bicker <i>et al.</i> 2017
5	Minoxidil	Paricalcitol	Cluster headache	
6	Omeprazole	Amoxicillin	Renal tubular acidosis	Russo <i>et al.</i> 2016
7	Anagrelide	Azelaic acid	Cerebral thrombosis	
8	Atorvastatin	Amlodipine	Muscle inflammation	Banakh <i>et al.</i> 2017
9	Aliskiren	Tioconazole	Breast inflammation	Parving <i>et al.</i> 2012
10	Estradiol	Nadolol	Endometriosis	

Case Report

Severe Rhabdomyolysis due to Presumed **Drug Interactions between Atorvastatin with Amlodipine** and Ticagrelor

Example 4: Traffic Prediction



Road Network as a Graph

- **Nodes:** Road segments
- **Edges:** Connectivity between road segments

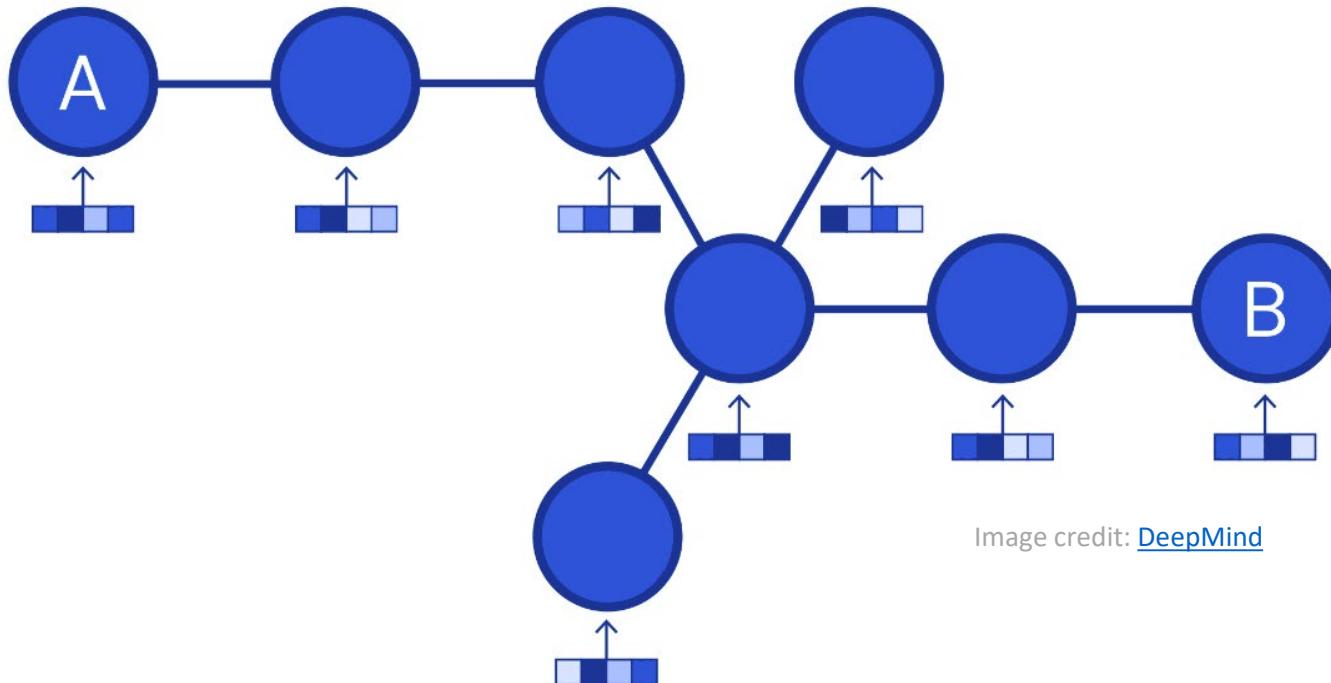
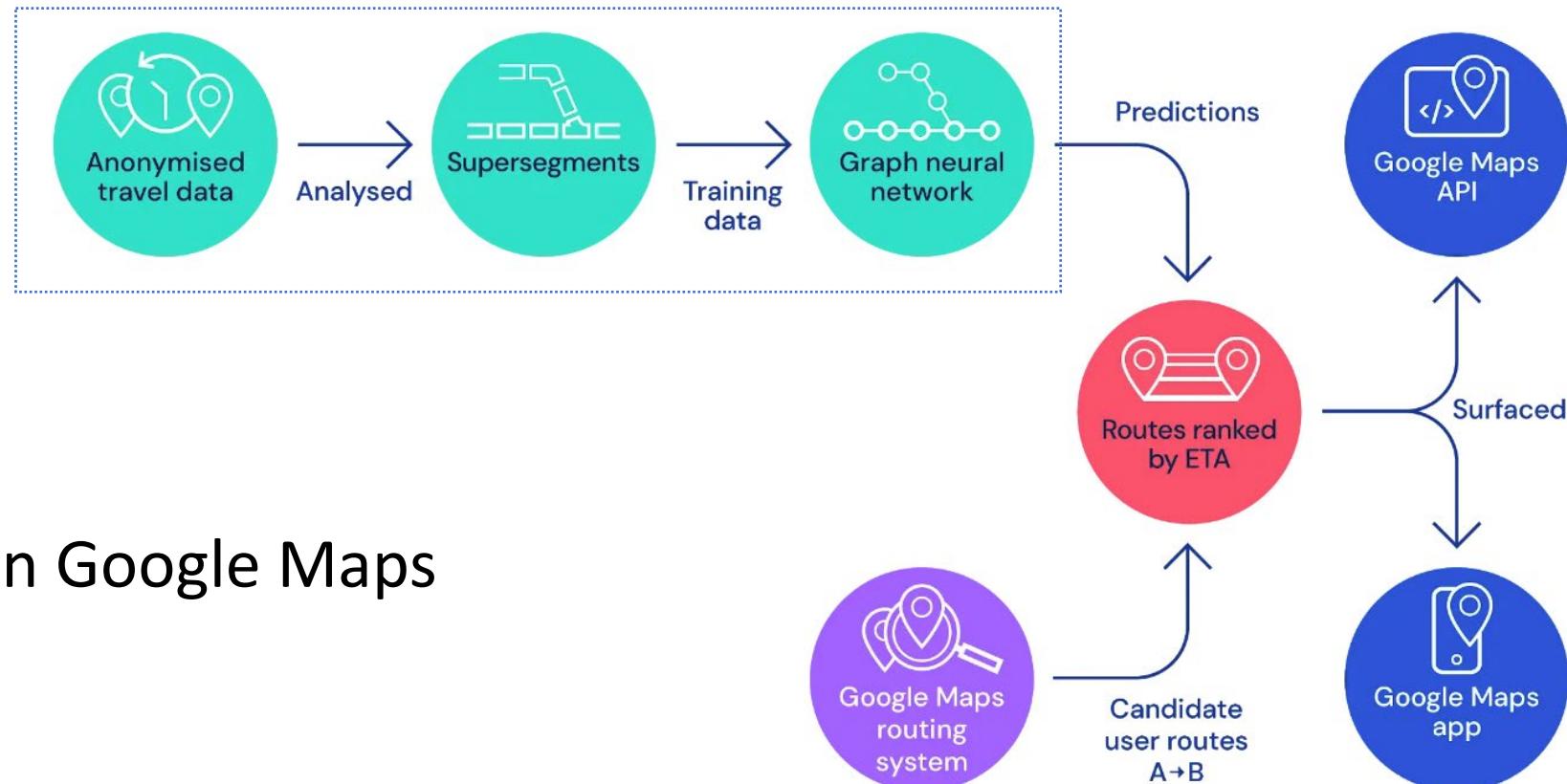


Image credit: [DeepMind](#)

Traffic Prediction via GNN

Predict via Graph Neural Networks



- Used in Google Maps

THE MODEL ARCHITECTURE FOR DETERMINING OPTIMAL ROUTES AND THEIR TRAVEL TIME.

Image credit: [DeepMind](#)

Example 5: Drug Discovery

- Antibiotics are small molecular graphs
 - Nodes: Atoms
 - Edges: Chemical bonds

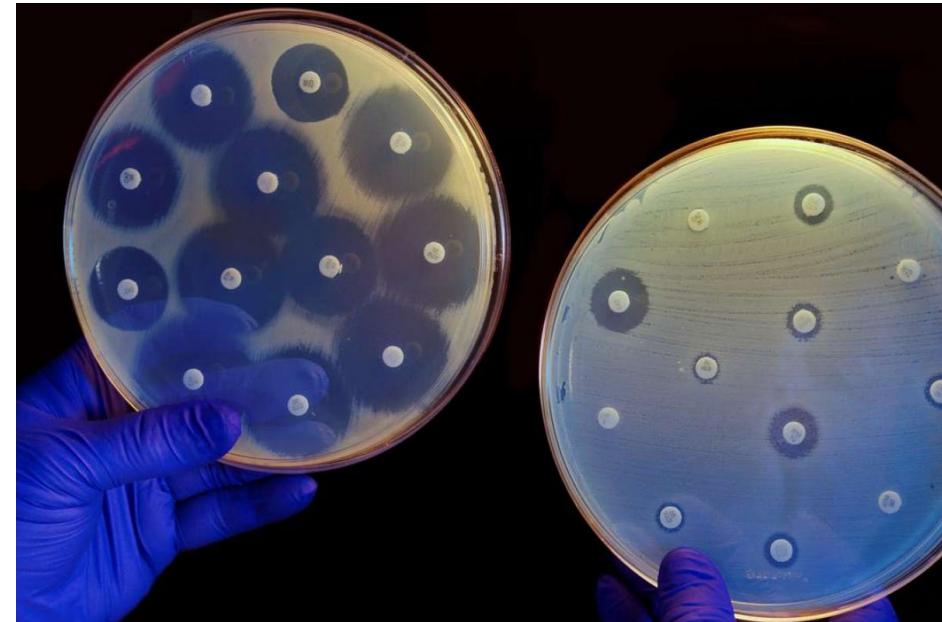
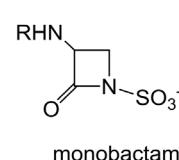
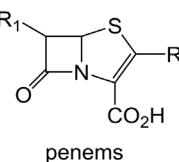
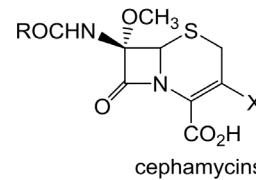
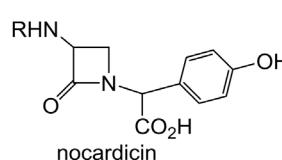
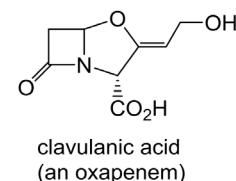
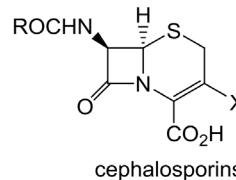
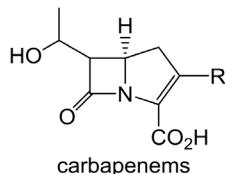
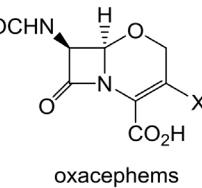
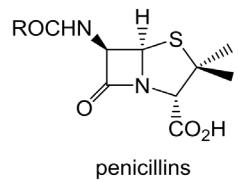
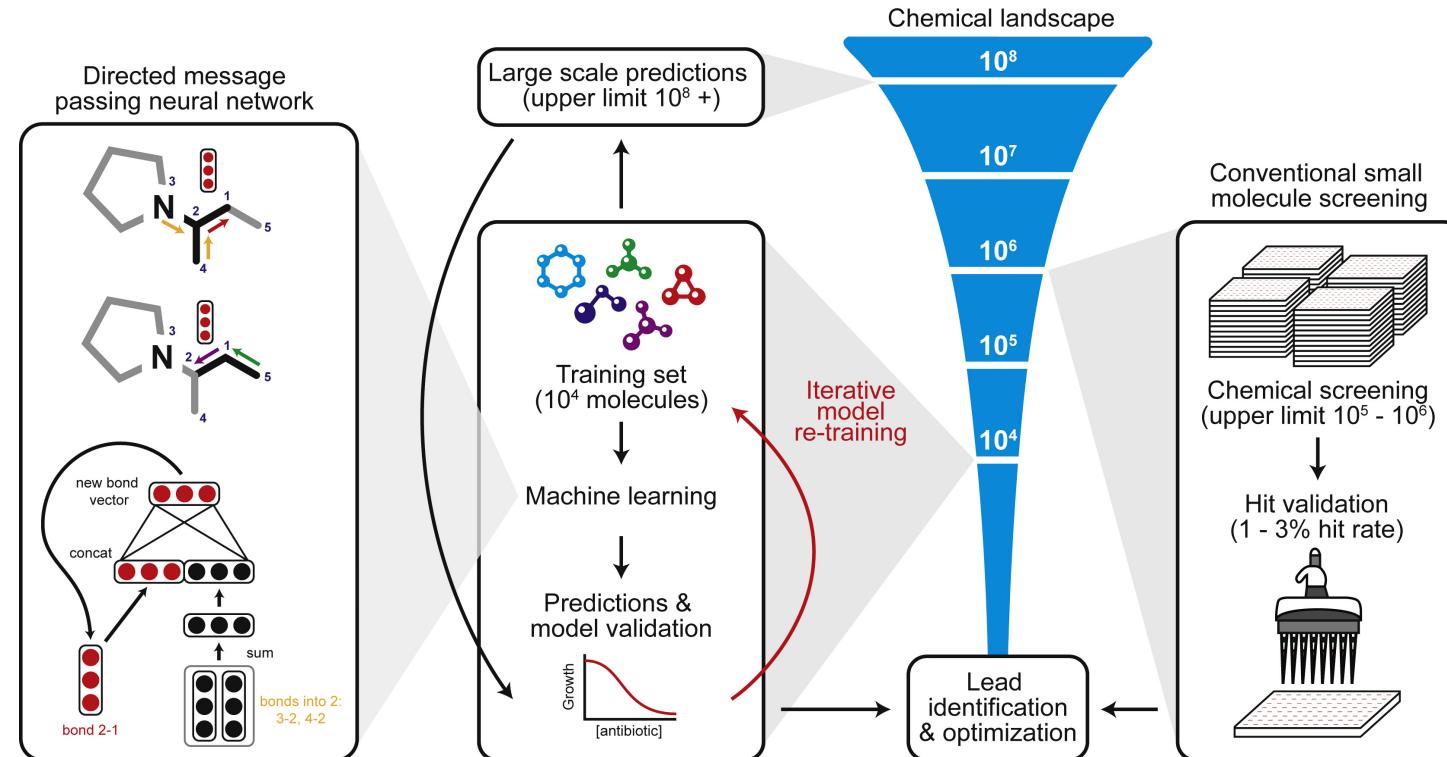


Image credit: [CNN](#)

Monika Konaklieva et al. Molecular targets of β-lactam-based antimicrobials: beyond the usual suspects.

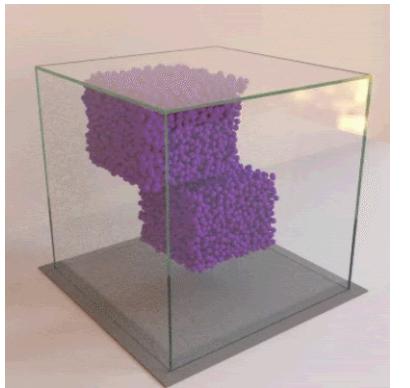
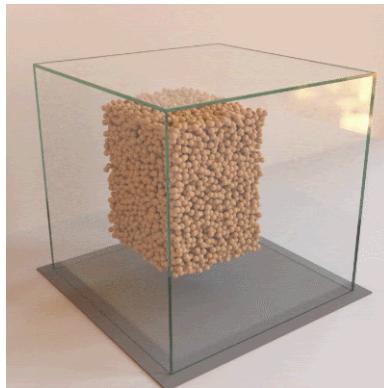
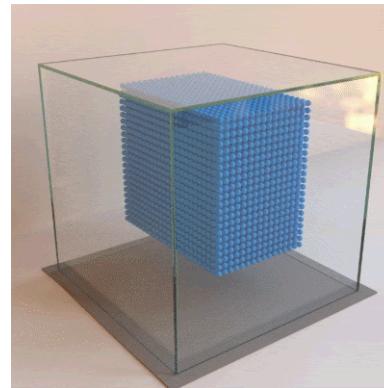
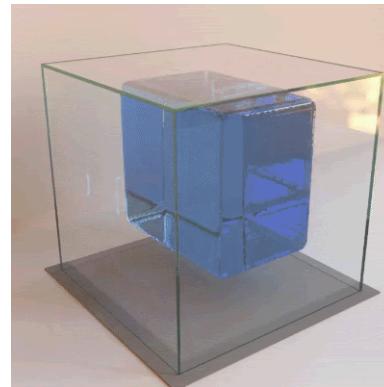
Deep Learning for Antibiotic Discovery

- A Graph Neural Network model
- Predict promising molecules from a pool of candidates



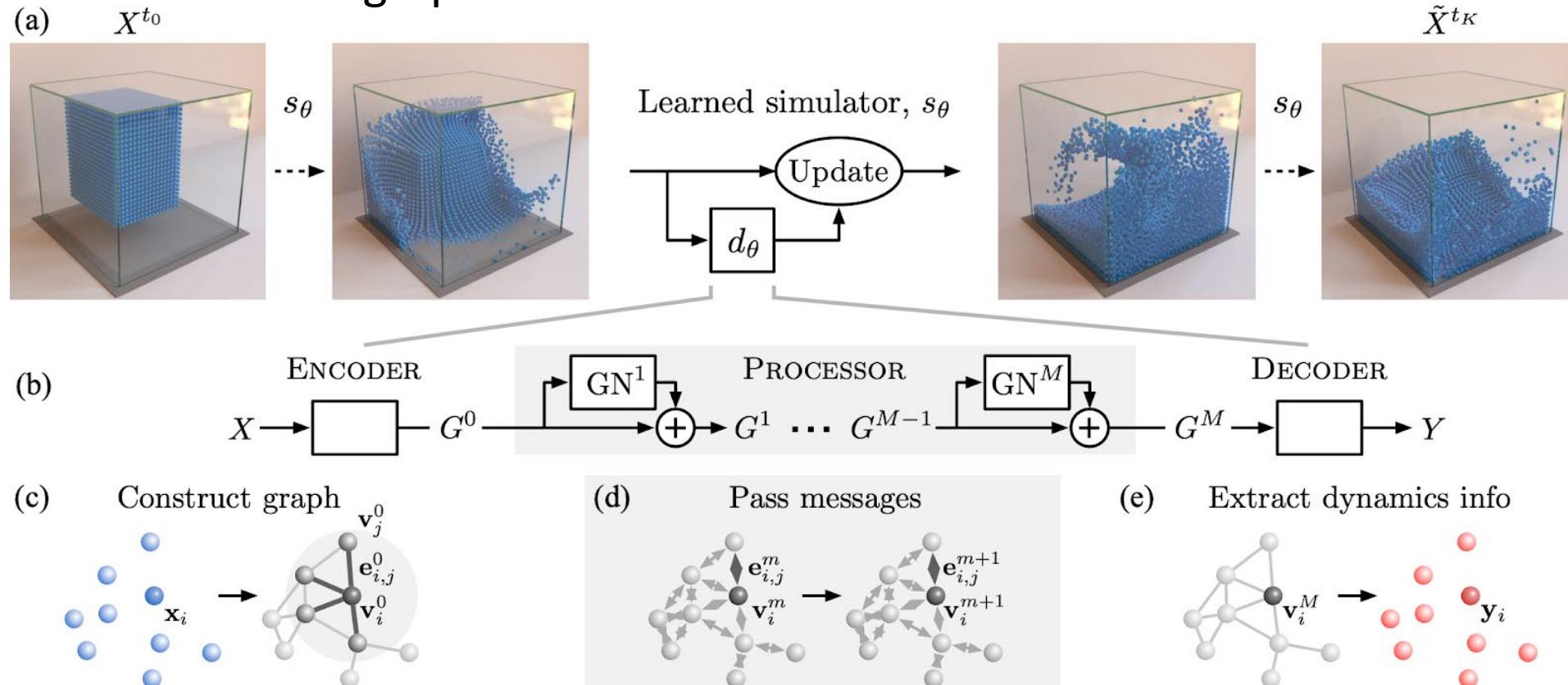
Example 6: Physics Simulation

- Physical simulation as a graph
 - **Nodes:** Particles:
 - **Edges:** Interaction between particles



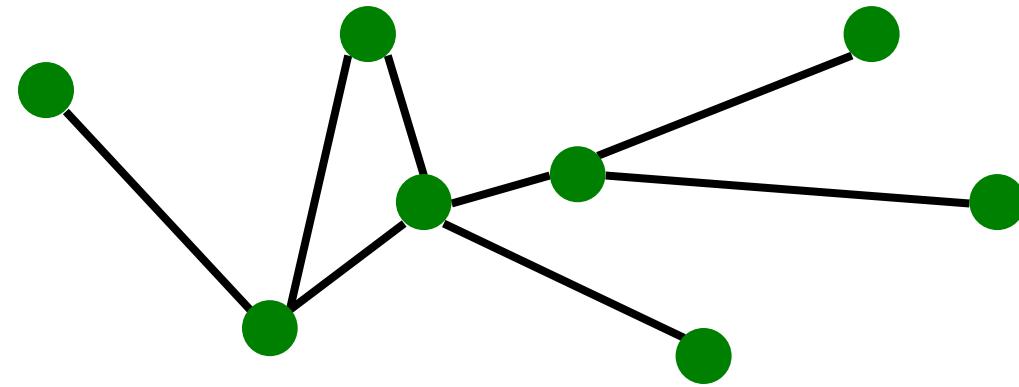
Simulation Learning Framework

- A graph evolution task
 - Goal: Predict how a graph will evolve over time



Choices of Graph Representation

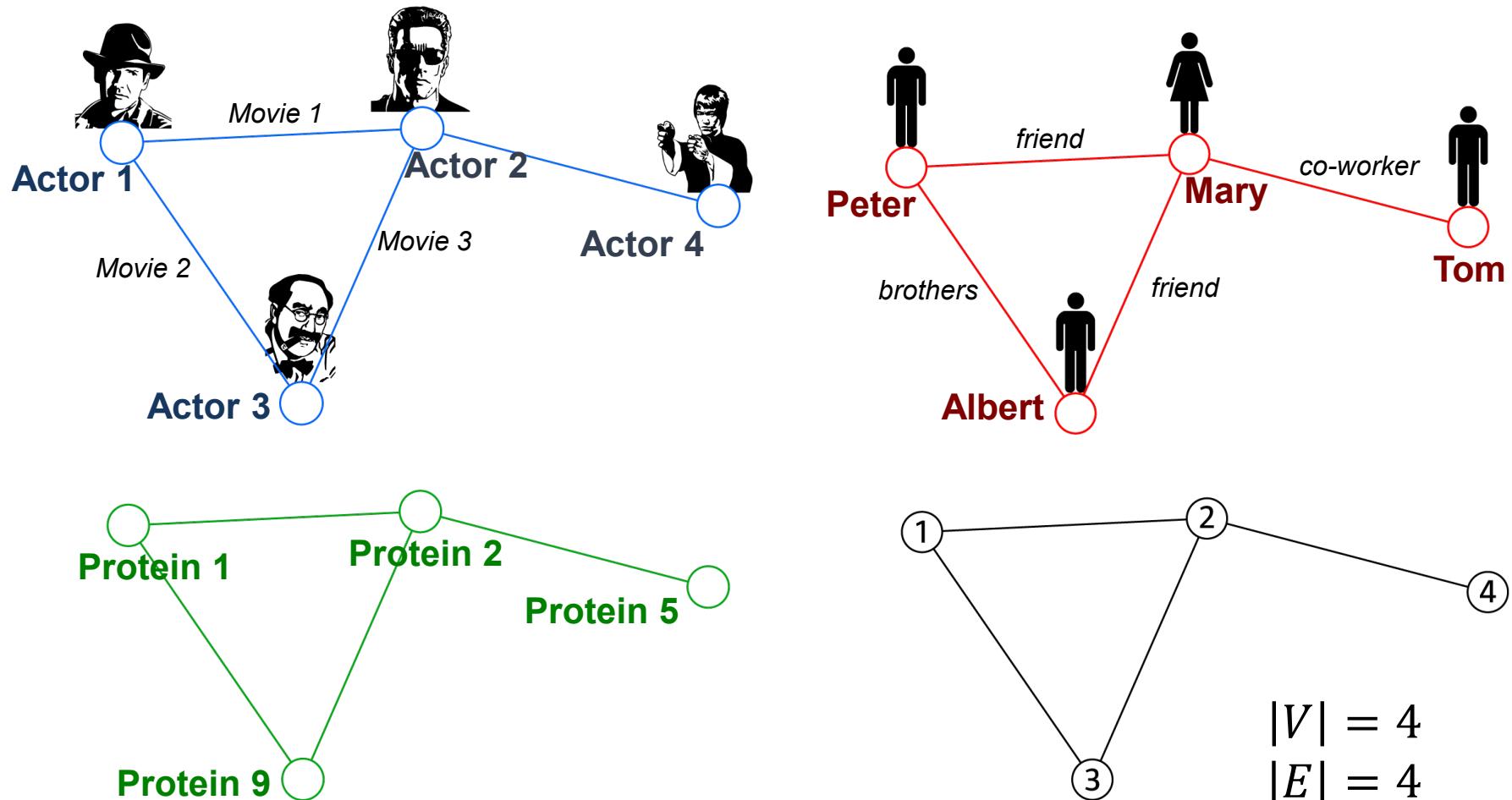
Components of a Network



- **Objects:** nodes, vertices
- **Interactions:** links, edges
- **System:** network, graph

V
 E
 $G(V, E)$

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

Choosing a Proper Representation

- If you connect individuals that work with each other, you will explore a **professional network**
- If you connect scientific papers that cite each other, you will be studying the **citation network**
- **Meaningful graph construction and graph structure discovery are important areas of research**



Image credit: [Euro Scientists](#)

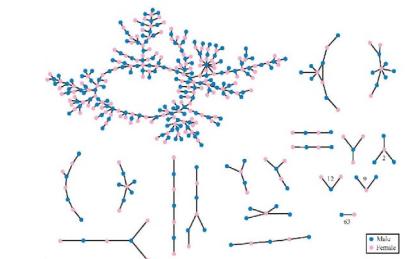
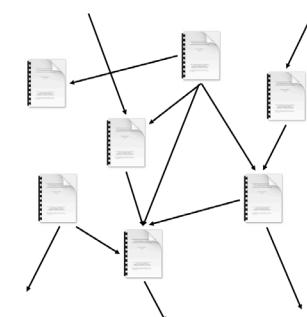


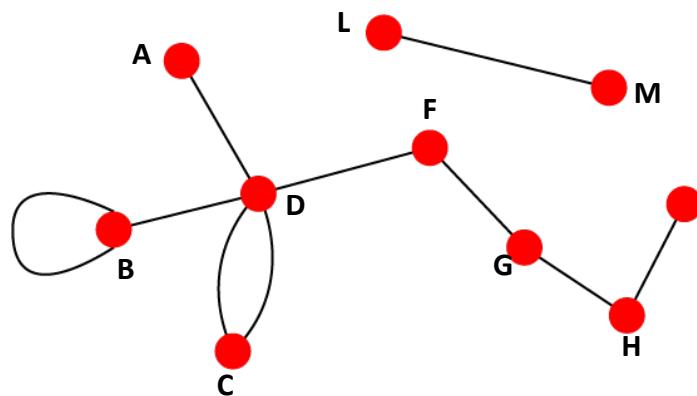
Image credit: [ResearchGate](#)



Directed vs. Undirected Graphs

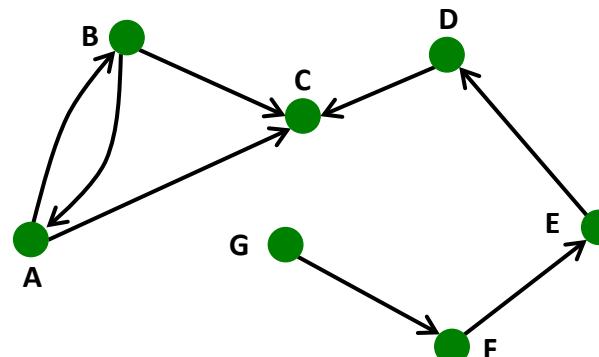
Undirected

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



Directed

- **Links:** directed (arcs)



- **Examples:**

- Collaborations
- Friendship on Facebook

- **Examples:**

- Phone calls
- Following on Twitter

Directed vs. Undirected Graphs: Node Degrees

- Node degree d_i : the number of edges adjacent to node i

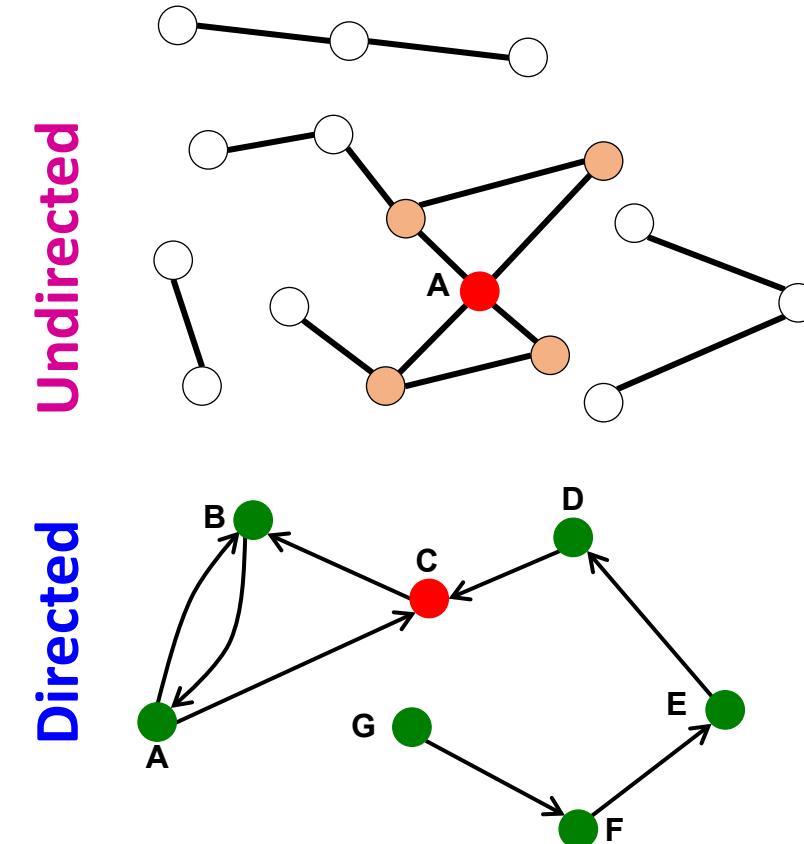
$$d_A = 4$$

- Avg. degree: $\bar{d} = \langle d \rangle = \frac{1}{|V|} \sum_{i=1}^{|V|} d_i = \frac{2|E|}{|V|}$

- In **directed networks** we define an **in-degree** and **out-degree**. The (total) degree of a nodes is the sum of in- and out-degrees.

$$d_C^{in} = 2, d_C^{out} = 1, d_c = 3$$

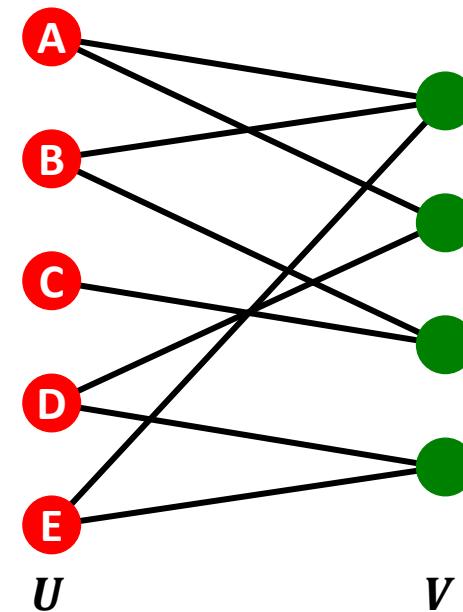
$$\bar{d} = \frac{|E|}{|V|}, \bar{d}^{in} = \bar{d}^{out}$$



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

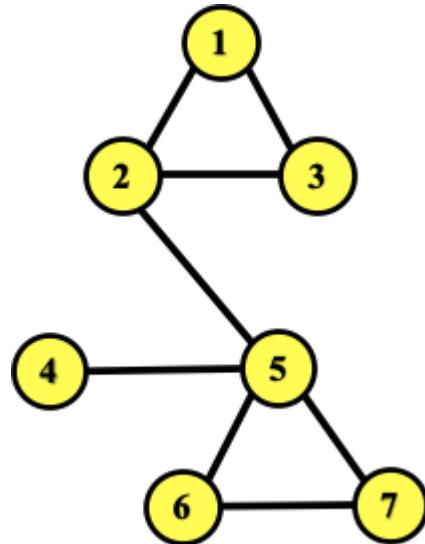
Bipartite Graph

- **Bipartite graph** is a graph whose nodes can be divided into two disjoint sets U and V such that every link connects a node in U to one in V ; that is, U and 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



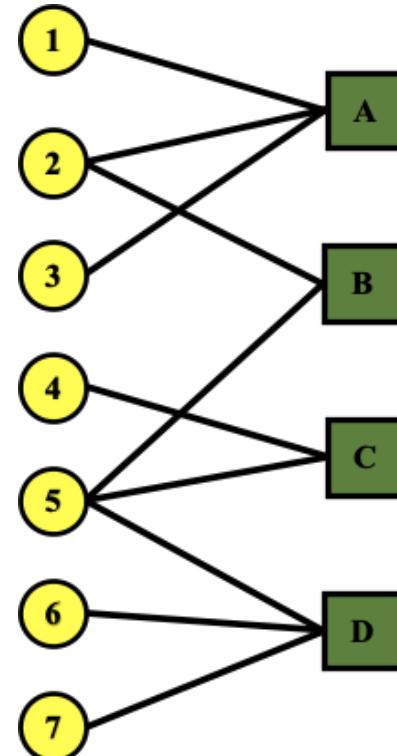
Folded / Projected Bipartite Graphs

Projection U

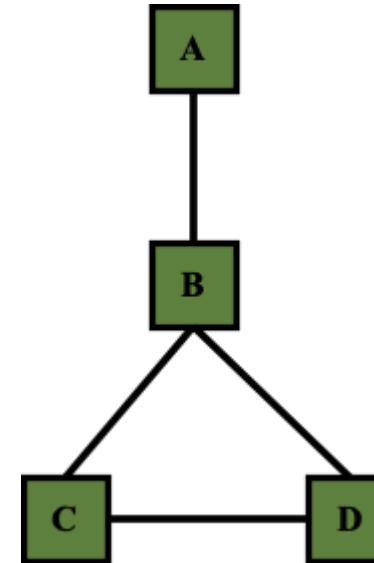


**Two nodes are connected
if they share the same
green neighbor**

U **V**

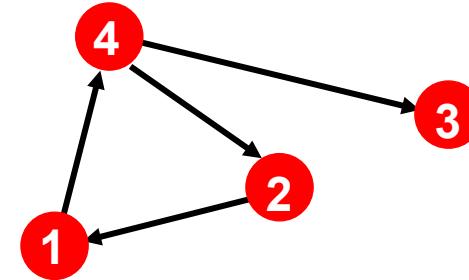
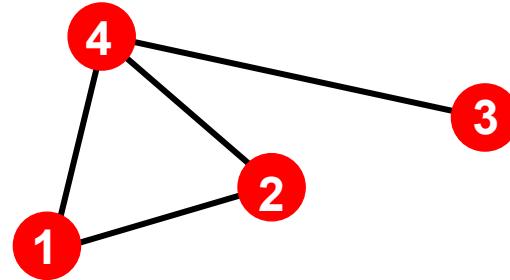


Projection V



**Two nodes are connected
if they share the same
yellow neighbor**

Representing Graphs: Adjacency Matrix



$A_{ij} = 1$ if there is a link from node i to node j

$A_{ij} = 0$ otherwise

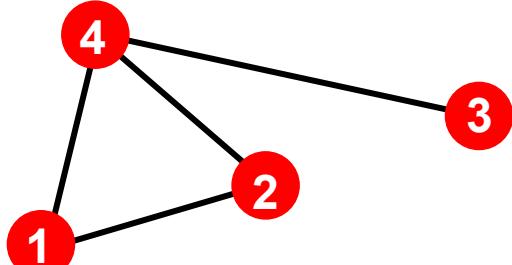
$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

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

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

Adjacency Matrix

Undirected



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

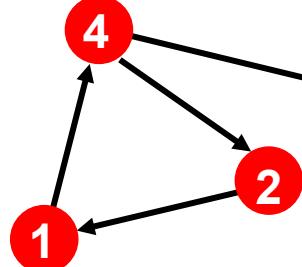
$A_{ij} = A_{ji}$
 $A_{ii} = 0$

$$d_i = \sum_{j=1}^{|V|} A_{ij}$$

$$d_j = \sum_{i=1}^{|V|} A_{ij}$$

$$|E| = \frac{1}{2} \sum_{i=1}^{|V|} d_i = \frac{1}{2} \sum_{i,j} A_{ij}$$

Directed



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

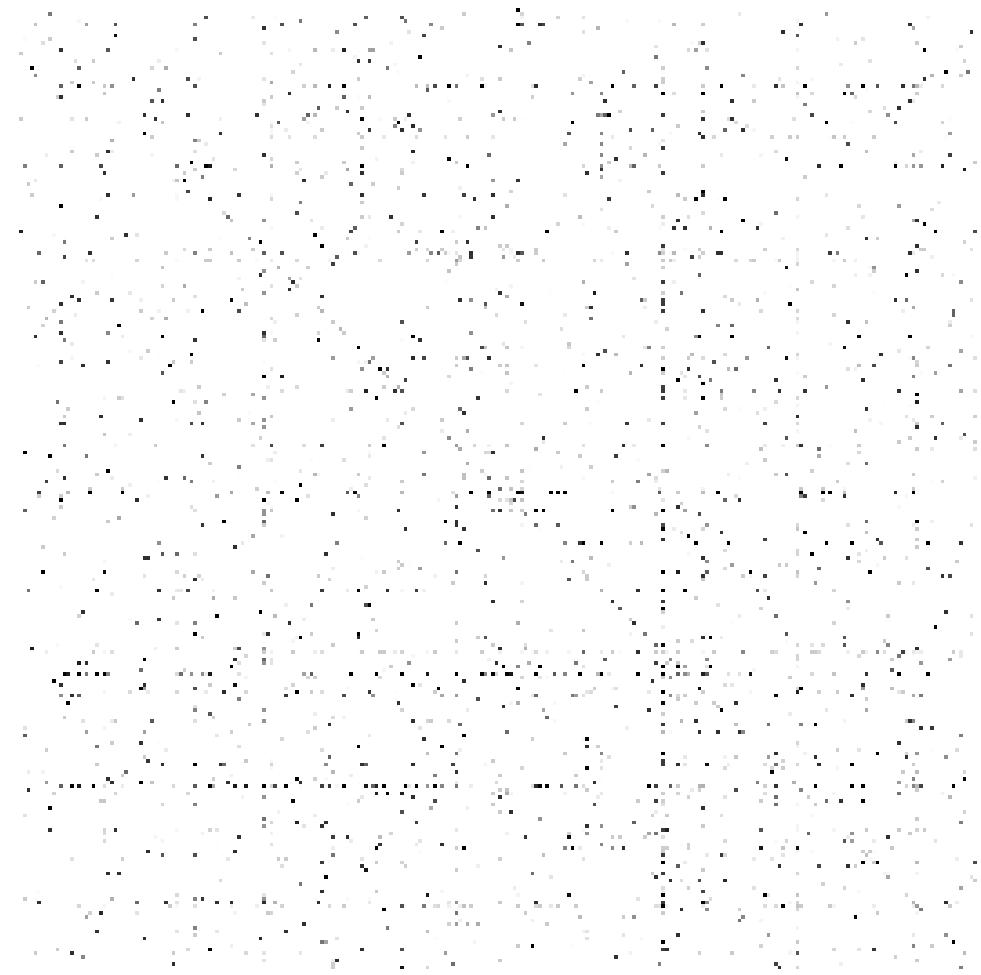
$A_{ij} = A_{ji}$
 $A_{ij} \neq A_{ji}$
 $A_{ii} = 0$

$$d_i^{out} = \sum_{j=1}^{|V|} A_{ij}$$

$$d_j^{in} = \sum_{i=1}^{|V|} A_{ij}$$

$$|E| = \sum_{i=1}^{|V|} d_i^{in} = \sum_{i=1}^{|V|} d_i^{in} = \sum_{i,j} A_{ij}$$

Adjacency Matrices are Sparse



Networks are Sparse Graphs

- Most real-world networks are sparse
- $|E| \ll |E|_{\max}$ or $k \ll |V| - 1$

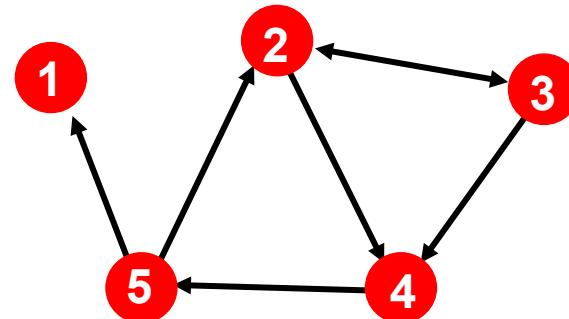
NETWORK	NODES	LINKS	DIRECTED/ UNDIRECTED	V	E	$\langle k \rangle$
Internet	Routers	Internet connections	Undirected	192,244	609,066	6.33
WWW	Webpages	Links	Directed	325,729	1,497,134	4.60
Power Grid	Power plants, transformers	Cables	Undirected	4,941	6,594	2.67
Phone Calls	Subscribers	Calls	Directed	36,595	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 ($|E|/|V|^2$):** WWW=1.51x10⁻⁵, MSN IM = 2.27x10⁻⁸)

Representing Graphs: Edge List

- Represent graph as a **list of edges**:

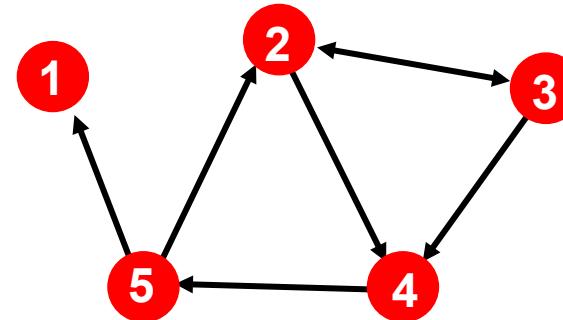
- (2, 3)
- (2, 4)
- (3, 2)
- (3, 4)
- (4, 5)
- (5, 2)
- (5, 1)



Representing Graphs: Adjacency List

- **Adjacency list:**

- Easier to work with if network is
 - Large
 - Sparse
- Allows us to quickly retrieve all **neighbors** of a given node
 - 1:
 - 2: 3, 4
 - 3: 2, 4
 - 4: 5
 - 5: 1, 2



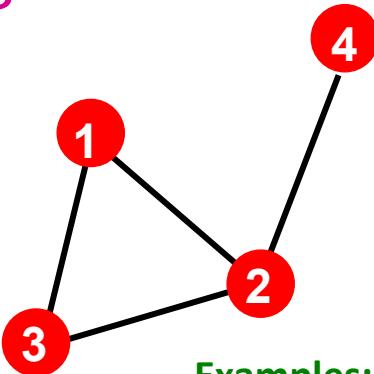
Nodes and Edge Attributes

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 (1)

- Unweighted

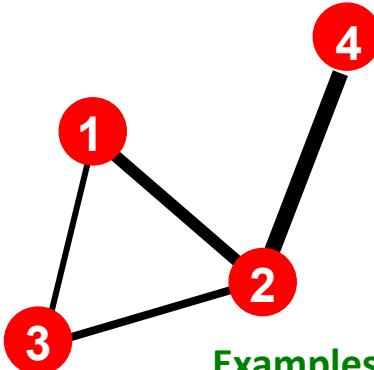


Examples: Friendship, Hyperlink

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

$$A_{ii} = 0, A_{ij} = A_{ji}$$
$$|E| = \frac{1}{2} \sum_{i,j=1}^{|V|} A_{ij}, \bar{d} = \frac{2|E|}{|V|}$$

- Weighted



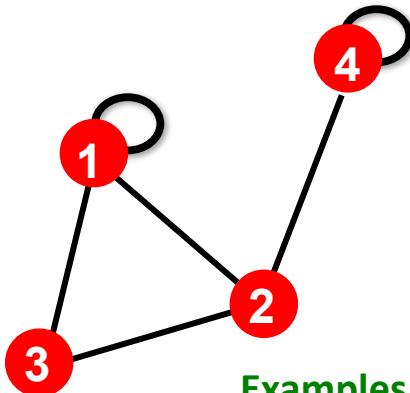
Examples: Collaboration, Internet, Roads

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

$$A_{ii} = 0, A_{ij} = A_{ji}$$
$$|E| = \frac{1}{2} \sum_{i,j=1}^{|V|} \text{nonzero}(A_{ij}), \bar{d} = \frac{2|E|}{|V|}$$

More Types of Graphs (2)

- Self-edges (self-loops) (undirected)



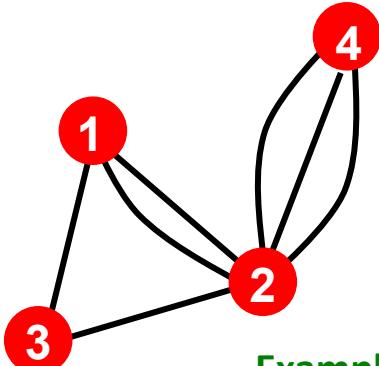
Examples: Proteins, Hyperlinks

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

$$|E| = \frac{1}{2} \sum_{i,j=1, i \neq j}^{|V|} A_{ij} + \sum_i^{|V|} A_{ii}, \bar{d} = \frac{2|E|}{|V|}$$

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

- Multigraph (undirected)



Examples: Communication, Collaboration

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

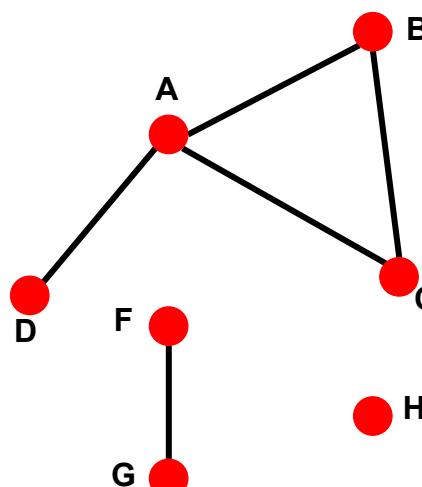
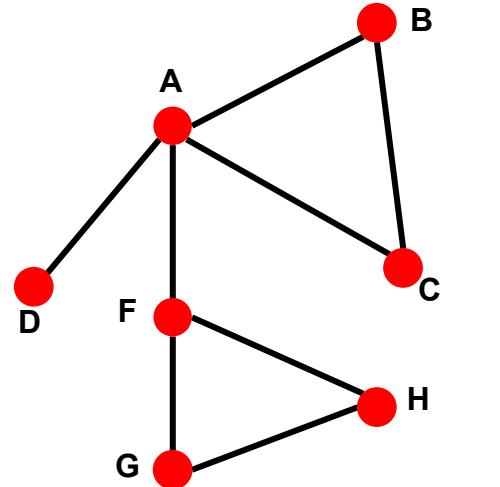
$$|E| = \frac{1}{2} \sum_{i,j=1}^{|V|} \text{nonzero}(A_{ij}), \bar{d} = \frac{2|E|}{|V|}$$

$A_{ii} = 0, A_{ij} = A_{ji}$

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



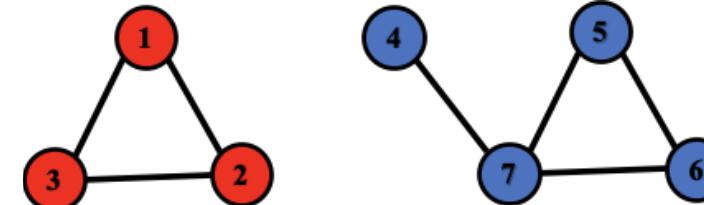
Largest Component:
Giant Component

Isolated node (node H)

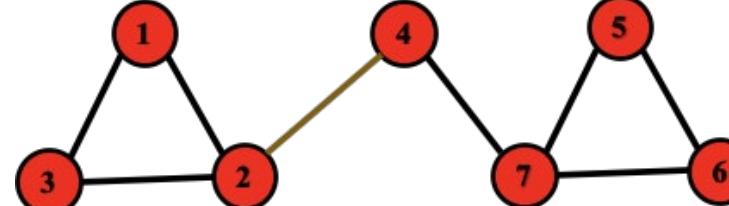
Connectivity: Example

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



Connected



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

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

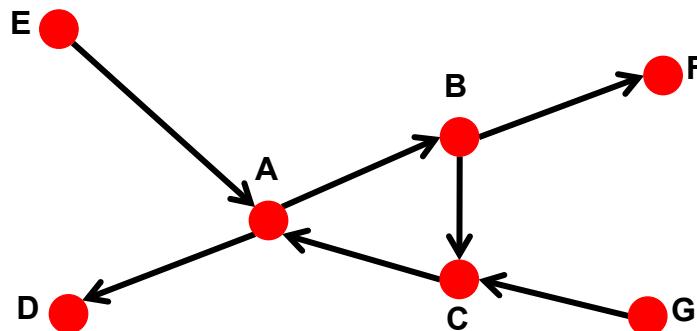
Connectivity of Directed Graphs (1)

- **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**

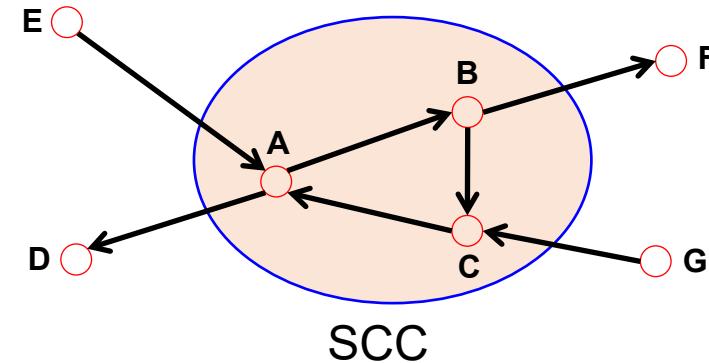
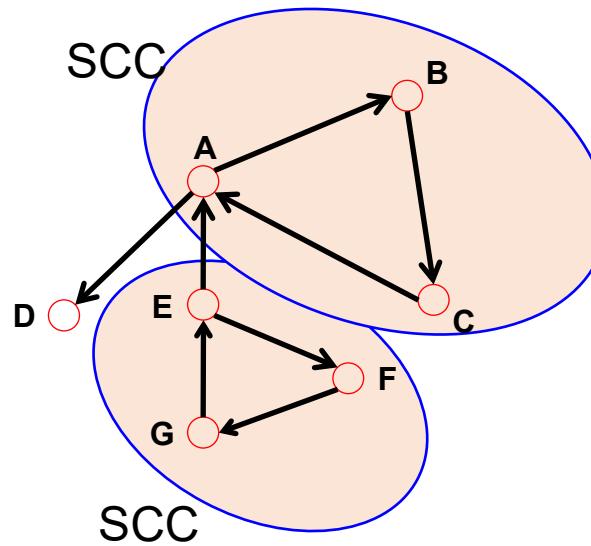
- is connected if we disregard the edge directions



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 (2)

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

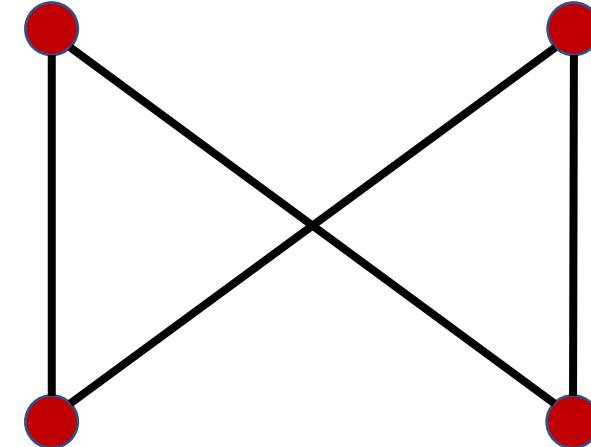
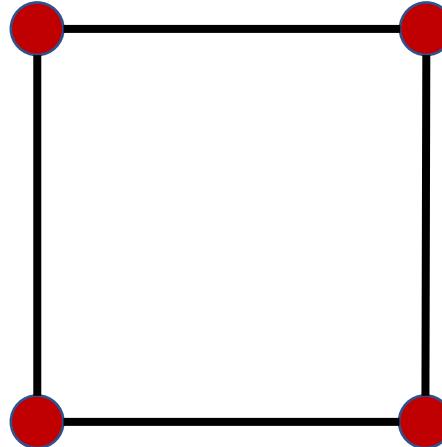


In-component: nodes that can reach the SCC,

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

Isomorphism of Graphs

- Graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are **isomorphic** if there exists a one-to-one and onto map $f: V_1 \rightarrow V_2$, with the property that $(a, b) \in E_1$ iff $(f(a), f(b)) \in E_2$, $\forall a, b \in V_1$.
- Example:



Summary

- **Machine learning with Graphs**
 - Applications and use cases
- **Different types of tasks:**
 - Biological networks, proteins
 - Recommender systems, social networks
 - Dynamic networks, simulations
- **Choice of a graph representation:**
 - Directed, undirected, bipartite, weighted, adjacency matrix

Readings

- P. Erdos, A. Renyi. [*On Random Graphs I*](#). Publ. Math. Debrecen, 1959.
- P. Erdos, A. Renyi. [*On the evolution of random graphs*](#). Magyar Tud. Akad. Mat. Kutato Int. Koezl., 1960.
- B. Bollobas. [*Random Graphs*](#). Cambridge University Press.
- M.E.J. Newman, S. H. Strogatz and D.J. Watts. [*Random graphs with arbitrary degree distributions and their applications*](#). Phys. Rev. E 64, 026118, 2001.
- R. Milo, N. Kashtan, S. Itzkovitz, M.E.J. Newman, U. Alon. [*On the uniform generation of random graphs with prescribed degree sequences*](#). Arxiv, 2004.
- D. Ellis. [*The expansion of random regular graphs*](#). Lecture notes from Algebraic methods in combinatorics, Cambridge University, 2011.
- S. Arora, S. Rao and U. Vazirani. [*Expander Flows, Geometric Embeddings and Graph Partitioning*](#). In proc. STOC '04, 2004.