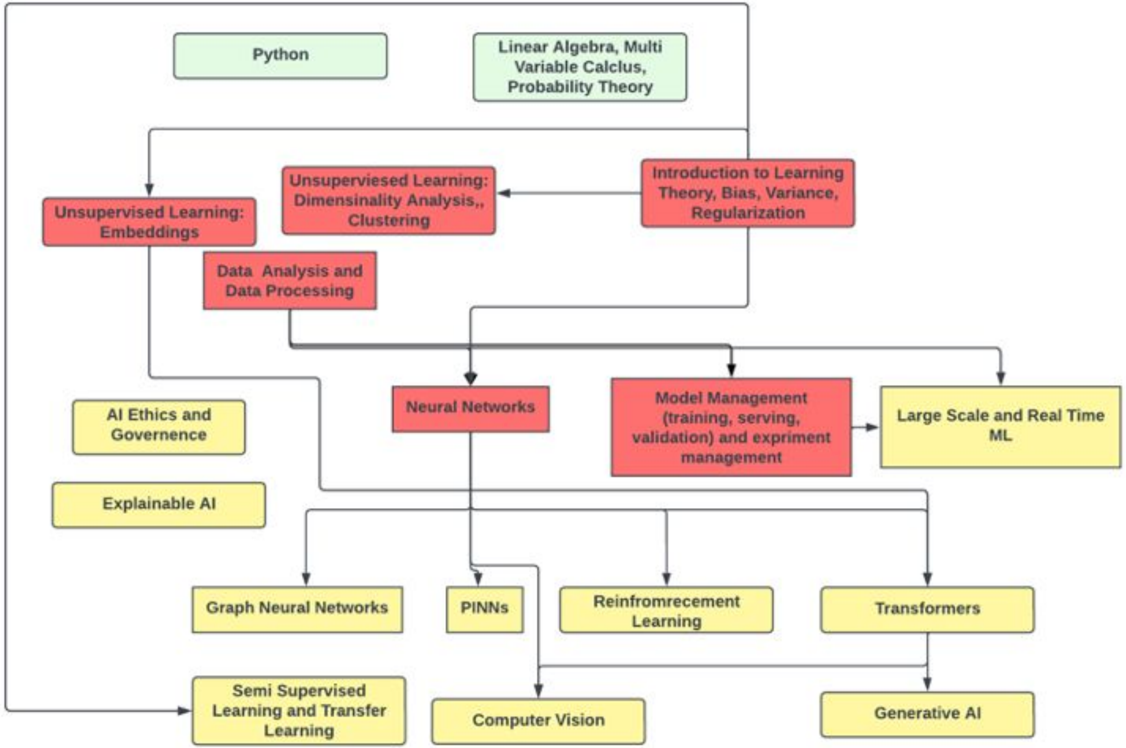
Graph Neural Network Introduction & Background Topics

Using content from :
https://web.stanford.edu/class/cs224w/slides/01-intro.pdf

AI Bootcamp program



Foundation Bootcamp Topics

Prerequisites

Specialized Bootcamp Topics

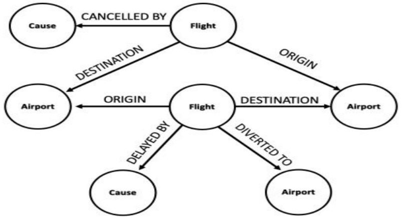
What is Covered in This Bootcamp

- Graph Representation in Machine Learning
  - Representing nodes, edges, subgraphs, and entire graphs
  - Introduction to Graph Neural Networks (GNNs)
  - What are GNNs and why are they important?
- Types of GNNs
  - GCNs (Graph Convolutional Networks): Core concepts and applications
  - GATs (Graph Attention Networks): Attention mechanisms for graphs
  - Geometric GNNs
  - Other variations: GIN (Graph Isomorphism Networks), GraphSAGE, etc
- Training and Using GNNs
  - Techniques for training and evaluating GNNs
  - Applications and practical considerations
- Extending GNNs
  - Adapting GNNs for knowledge graphs and heterogeneous graphs
- Scaling GNNs
  - Methods to scale GNNs for large and complex datasets

What is Not Covered in This Bootcamp

- Graph Generation
  - Using generative models to create graphs
- Subgraph Analysis
  - Techniques for analyzing subgraphs within larger graphs
  - Subgraph matching: Identifying and aligning subgraphs between different graphs
  - Frequent subgraph discovery: Detecting commonly occurring subgraphs
- Community Detection
  - Identifying and analyzing communities within graphs

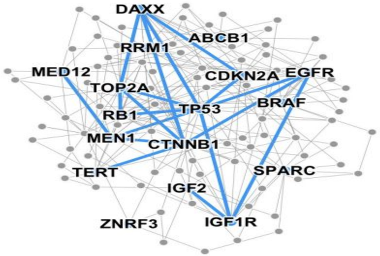
Graphs are Everywhere



Event Graphs

Image credit: SalientNetworks

Computer Networks



Disease Pathways

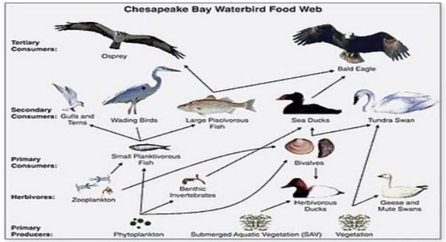


Image credit: Wikipedia

Food Webs

Image credit: Pinterest

Particle Networks

Image credit: visitlondon.com

Underground Networks

Taken from Stanford CS224W course: http://cs244w.stanford.edu

Other Examples of graphs in science and technology

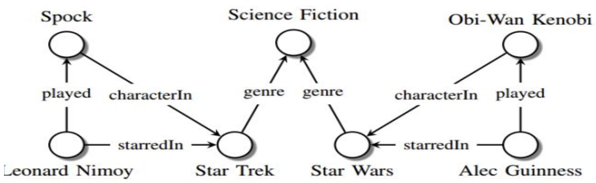


Image credit: [Maximilian Nickel et al]

Knowledge Graphs

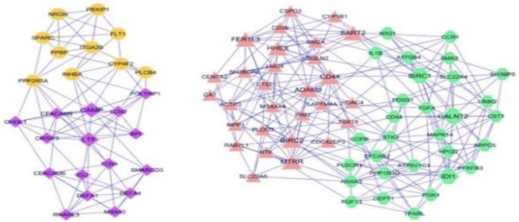


Image credit: [ese.wustl.edu]

Regulatory Networks

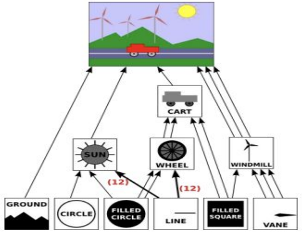


Image credit: [math.hws.edu]

Scene Graphs

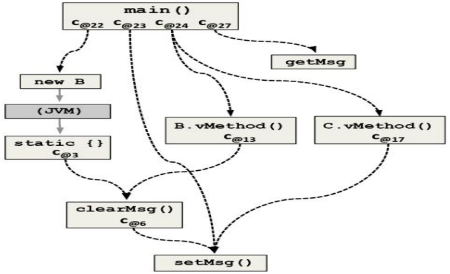


Image credit: [ResearchGate]

Code Graphs

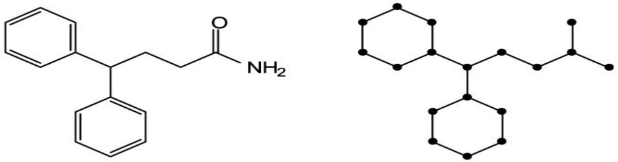


Image credit: [MDPI]

Molecules

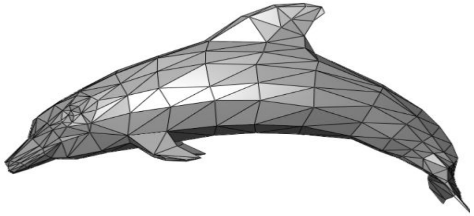


Image credit: [Wikipedia]

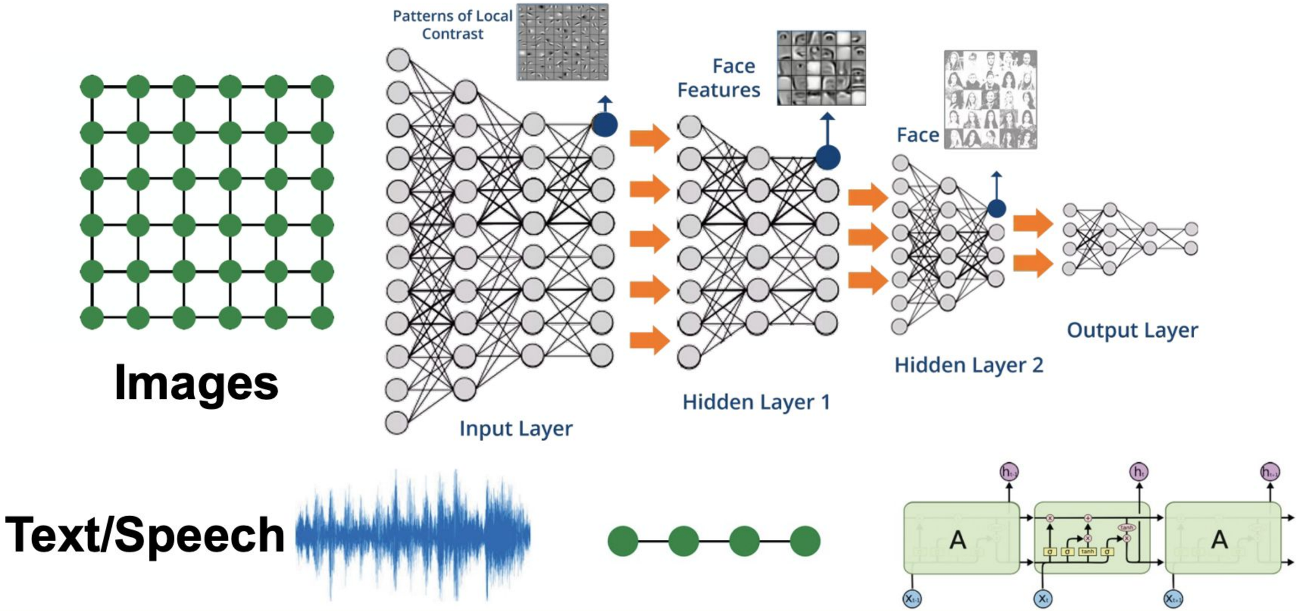
3D Shapes

Taken from Stanford CS224W course: [http://cs244w.stanford.edu]

Graphs vs. Other Mathematical Abstractions

Abstraction	Definition	Relation to Graphs
Sets	Unordered collections of objects	**Superset: Graphs are defined using sets of nodes and edges.**
Sequences	Ordered lists of elements	**Subset: Paths and traversals in graphs are sequences of nodes/edges.**
Probabilistic Models	Frameworks for randomness	**Extend graphs to model uncertainty (e.g., Bayesian networks).**
Dynamical Systems	Models of system evolution over time	**State transitions can be represented as graphs.**
Logical Abstractions	Formal reasoning via propositions	**Dependency and reasoning structures map to graphs.**
Functions	Mapping from sets to sets	**Neural Networks (Universal Approximators) are Graphs**

Deep Learning Tools Assume Structured Input (Grid/Sequence)

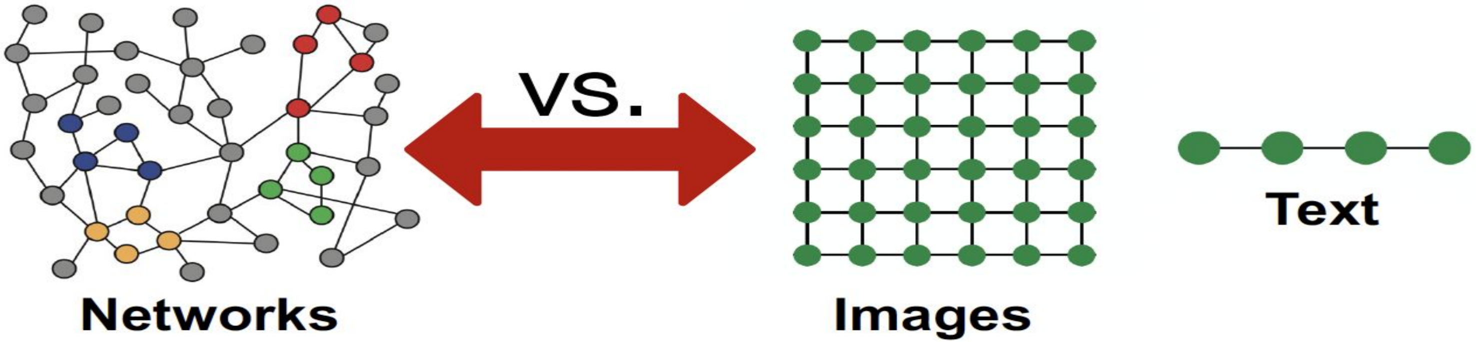


Taken from Stanford CS224W course: http://cs244w.stanford.edu

Why “Learning Graphs” is 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

Taken from Stanford CS224W course: http://cs244w.stanford.edu

Graph Representation

Heterogeneous Graphs

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

- Nodes with node types $v_i \in V$
- Edges with relation types $(v_i, r, v_j) \in E$
- Node type $T(v_i)$
- Relation type $r \in R$
- Nodes and edges have **attributes/features**

Taken from Stanford CS224W course: http://cs244w.stanford.edu

The Role of Proper Network Representation

- **Unique, Unambiguous Representation:**
  - Example: In a molecular graph, the representation is straightforward
    - Nodes = atoms and Edges = chemical bonds
    - unambiguous representation since the structure is determined by chemistry rules
- **Non-Unique Representation:** – depending on the context
  - Example: Social Networks
    - Nodes = people and Edges = friendships
    - Nodes = people and Edges = interactions (e.g., likes or comments)
    - Nodes = people and (weighted) Edges = communication frequency

Taken from Stanford CS224W course: http://cs244w.stanford.edu

The Role of Proper Network Representation

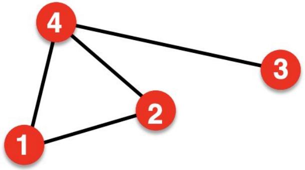
The structure of the graph influences the types of insights you can derive

- **undirected edges:** mutual relationships like communities or clusters
  - Example: co-authorship graph, edges represent papers co-written by authors, helping detect research communities
- **directed edges:** flow or influence
  - Example: citation network, edges point from one paper to another, helping track knowledge diffusion
- **weighted edges:** strength or importance of relationships
  - Example: In a trade network, edges represent trade volume between countries, enabling analysis of economic influence

Taken from Stanford CS224W course: http://cs244w.stanford.edu

Representing Graphs: Adjacency Matrix

Undirected



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

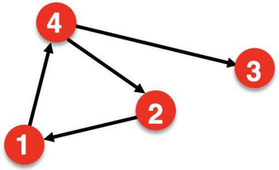
$$A_{ij} = A_{ji}$$
$$A_{ii} = 0$$

$$k_i = \sum_{j=1}^{N} A_{ij}$$

$$k_j = \sum_{i=1}^{N} A_{ij}$$

$$L = \frac{1}{2}\sum_{i=1}^{N} k_i = \frac{1}{2}\sum_{ij}^{N} A_{ij}$$

Directed



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

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

$$k_i^{out} = \sum_{j=1}^{N} A_{ij}$$

$$k_j^{in} = \sum_{i=1}^{N} A_{ij}$$

$$L = \sum_{i=1}^{N} k_i^{in} = \sum_{j=1}^{N} k_j^{out} = \sum_{i,j}^{N} A_{ij}$$

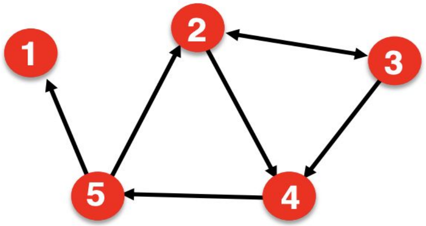
A is sparse in real world applications

Question: How can we calculate k-hop neighbors?

Taken from Stanford CS224W course: http://cs244w.stanford.edu

Representing Graphs

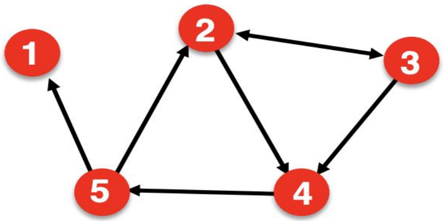
- **Represent graph as a list of edges:**
  - (2, 3)
  - (2, 4)
  - (3, 2)
  - (3, 4)
  - (4, 5)
  - (5, 2)
  - (5, 1)



Taken from Stanford CS224W course: http://cs244w.stanford.edu

Representing Graphs

- **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



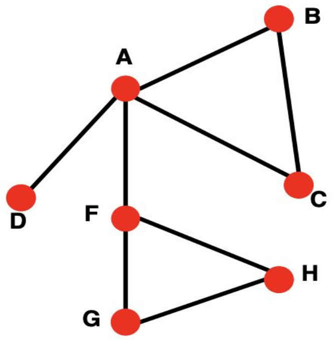
Taken from Stanford CS224W course: http://cs244w.stanford.edu

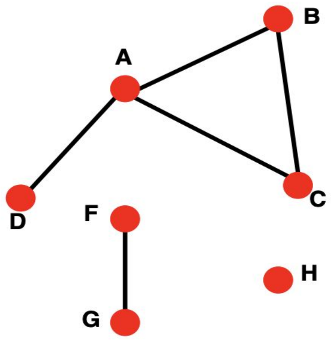
Representing Graphs

Representation	Advantages	Disadvantages	Common Use Cases
Adjacency List	Space-efficient for sparse graphs, fast neighbor iteration.	Slow edge lookups for dense graphs	Traversal algorithms (DFS, BFS).
Adjacency Matrix	Fast edge lookups, good for dense graphs and matrix-based algorithms.	High space complexity, inefficient neighbor iteration.	**Linear algebra algorithms** (e.g., PageRank).
Edge List	Very space-efficient, simple to construct and use.	Inefficient for traversal and neighbor lookups.	Storage and retrieval

Connected vs. Disconnected Graphs (Undirected)

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





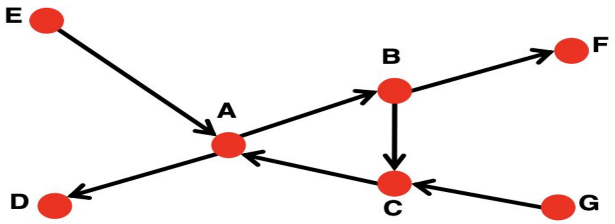
Largest Component:
Giant Component

Isolated node (node H)

Taken from Stanford CS224W course: http://cs244w.stanford.edu

Connected vs. Disconnected Graphs (Directed)

- **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

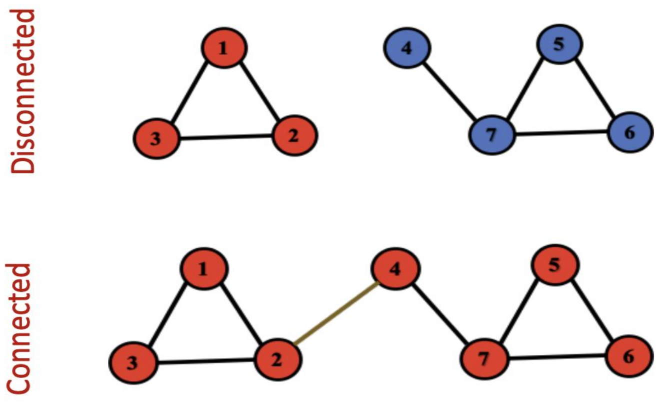


Graph on the left is connected but not strongly connected (e.g., there is no way to get from F to G by following the edge directions).

Taken from Stanford CS224W course: http://cs244w.stanford.edu

Connected vs. Disconnected Graph Representation

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:



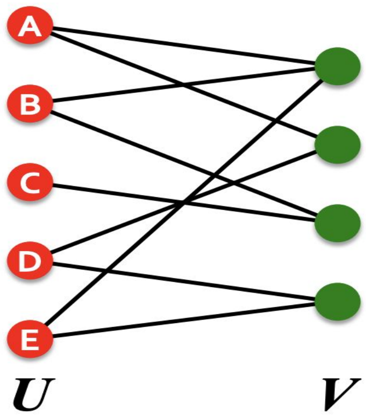
$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

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

Taken from Stanford CS224W course: http://cs244w.stanford.edu

Bipartite Graphs

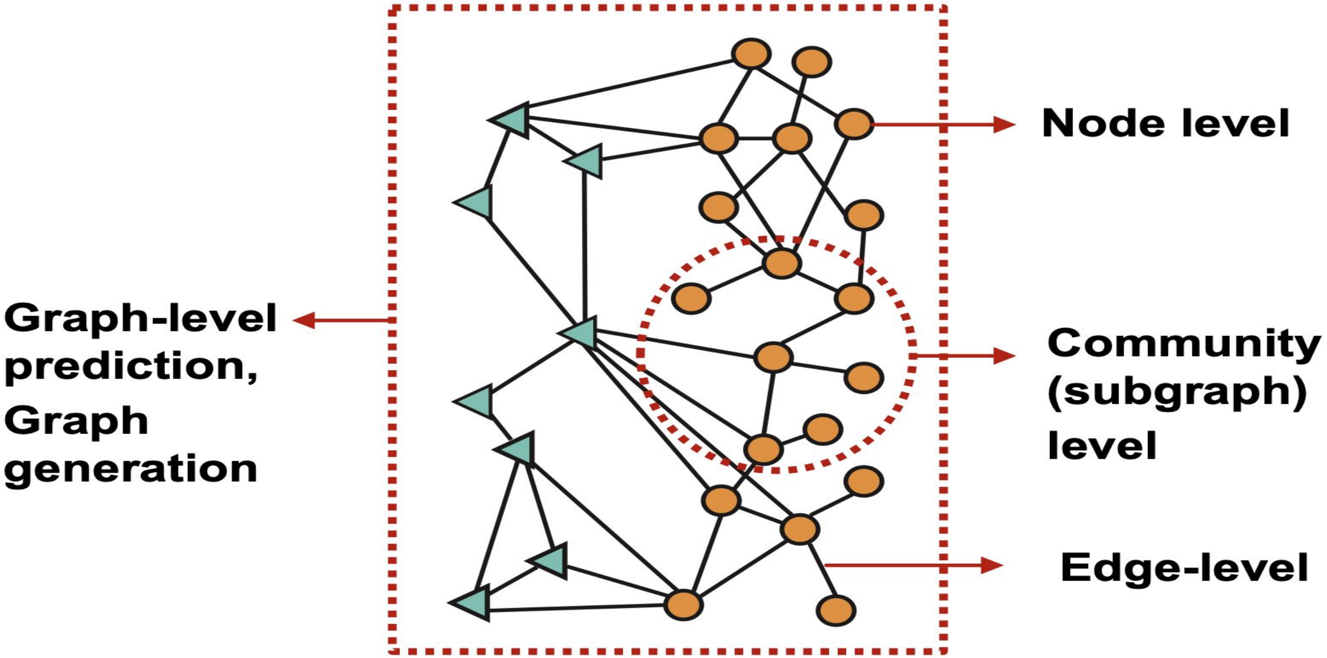
- **Bipartite graph** is a graph whose nodes can be divided into two disjoint sets ***U*** and ***V*** such that every link connects a node in ***U*** to one in ***V***; that is, ***U*** and ***V*** are **independent sets**
- **Examples:**
  - Authors-to-Papers (they authored)
  - Actors-to-Movies (they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



Taken from Stanford CS224W course: http://cs244w.stanford.edu

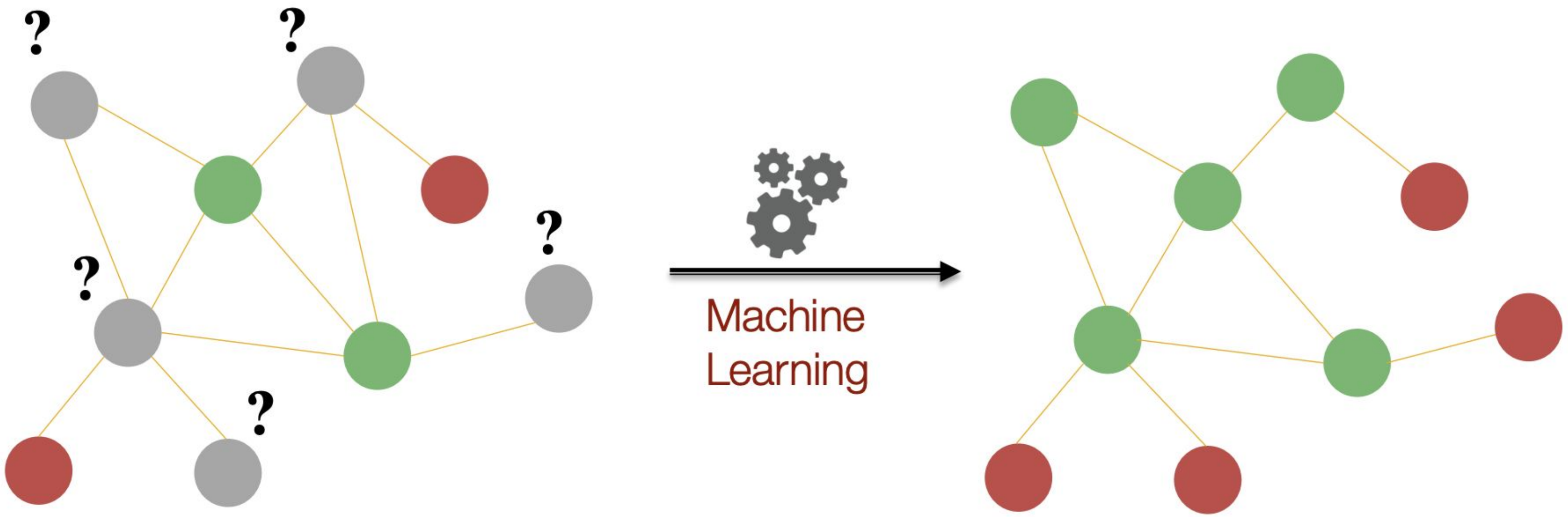
Graph ML Applicatons

Types of Graph ML Applications



Taken from Stanford CS224W course: http://cs244w.stanford.edu

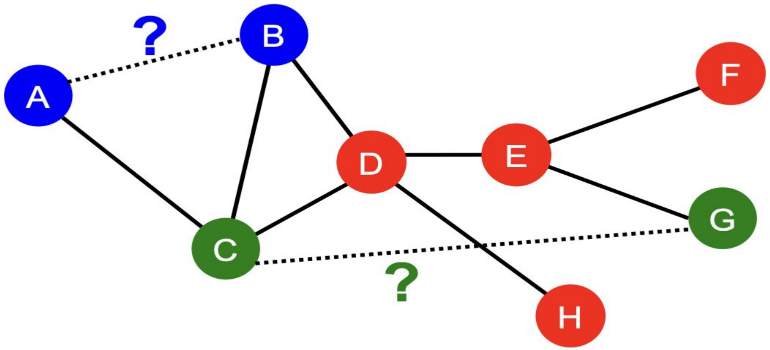
Node Classification



Taken from Stanford CS224W course: http://cs244w.stanford.edu

Link Prediction

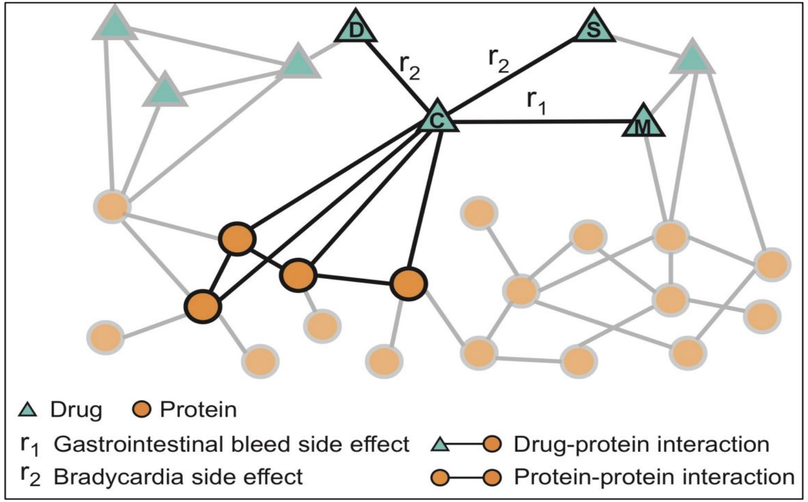
- The task is to predict **new/missing/unknown links** based on the existing links.
- At test time, node pairs (with no existing links) are ranked, and top K node pairs are predicted.
- Task: Make a prediction for **a pair of nodes**.



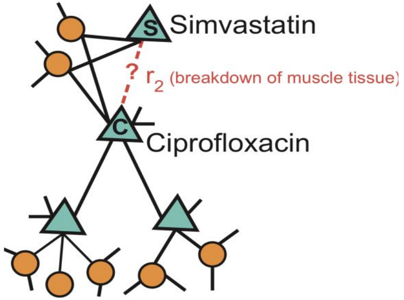
Taken from Stanford CS224W course: http://cs244w.stanford.edu

Link Prediction: Predicting Drug Interactions

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



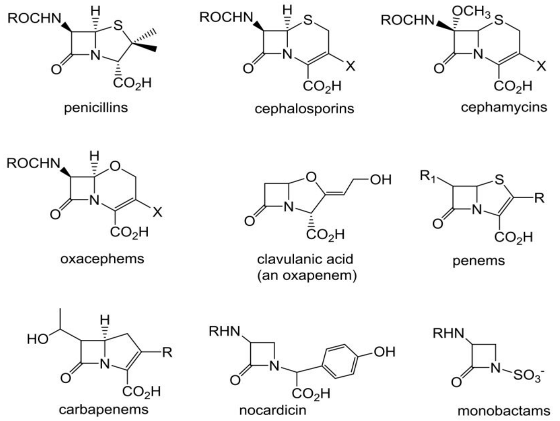
Query: How likely will Simvastatin and Ciprofloxacin, when taken together, break down muscle tissue?



Taken from Stanford CS224W course: http://cs244w.stanford.edu

Graph Level Prediction: Drug Discovery

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



Konaklieva, Monika I. "Molecular targets of β-lactam-based antimicrobials: beyond the usual suspects." Antibiotics 3.2 (2014): 128-142.

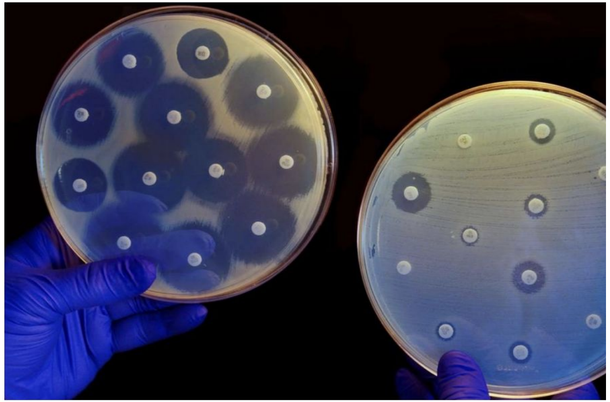
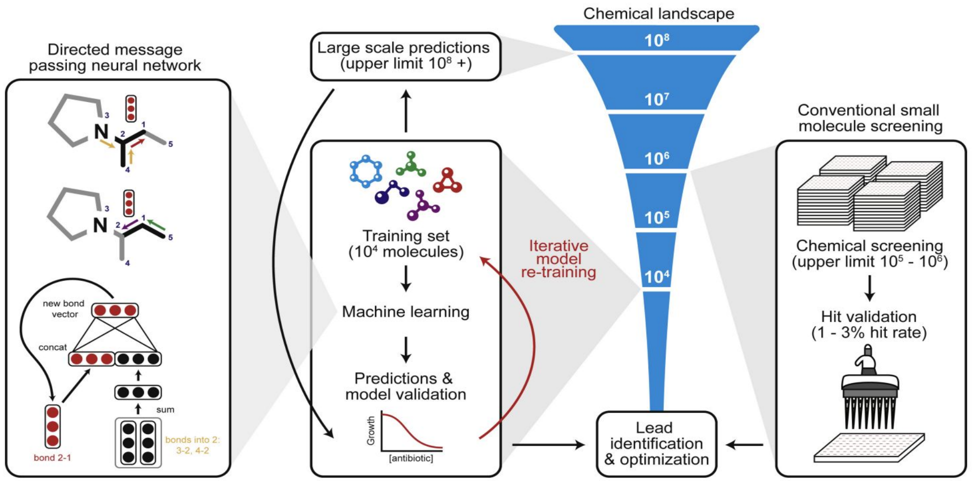


Image credit: CNN

Taken from Stanford CS224W course: http://cs244w.stanford.edu

Graph Level Prediction: Drug Discovery

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

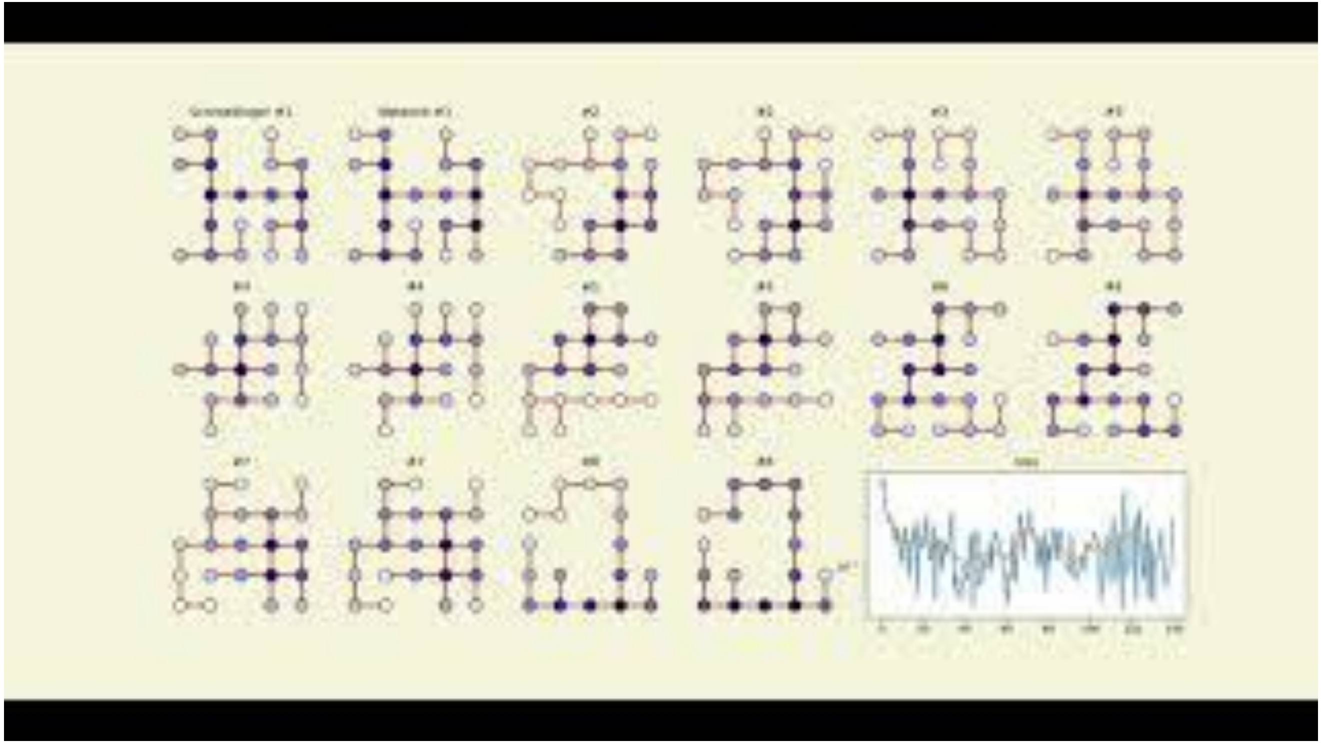


Stokes, Jonathan M., et al. "A deep learning approach to antibiotic discovery." Cell 180.4 (2020): 688-702.

Taken from Stanford CS224W course: http://cs244w.stanford.edu

Example from Physics

Graph Neural Network predicting Quantum Ground States



From https://www.youtube.com/watch?v=nyKuAm-HWn4&t=0s

Example from Math

SAT Solver Selection

Experimental Results

- Main results on the LEC and SC benchmarks. We report the average and standard deviation over 5 train-test folds.

	Avg Runtime (s) ↓	Solved (%) ↑	ACC ↑
	LEC		
Oracle	[illegible]	[illegible]	1.000±0.000
Best base solver	[illegible]	[illegible]	[illegible]
SATzilla07	[illegible]	[illegible]	[illegible]
SATzilla12	[illegible]	[illegible]	[illegible]
ArgoSmArT	[illegible]	[illegible]	[illegible]
CNN*	[illegible]	[illegible]	[illegible]
GraSS (ours)	[illegible]	[illegible]	[illegible]
	SC		
Oracle	[illegible]	[illegible]	1.000±0.000
Best base solver	[illegible]	[illegible]	[illegible]
SATzilla07	[illegible]	[illegible]	[illegible]
SATzilla12	[illegible]	[illegible]	[illegible]
ArgoSmArT	[illegible]	[illegible]	[illegible]
CNN	[illegible]	[illegible]	[illegible]
GraSS (ours)	[illegible]	[illegible]	[illegible]

From https://www.youtube.com/watch?v=aM_o70JfJLc