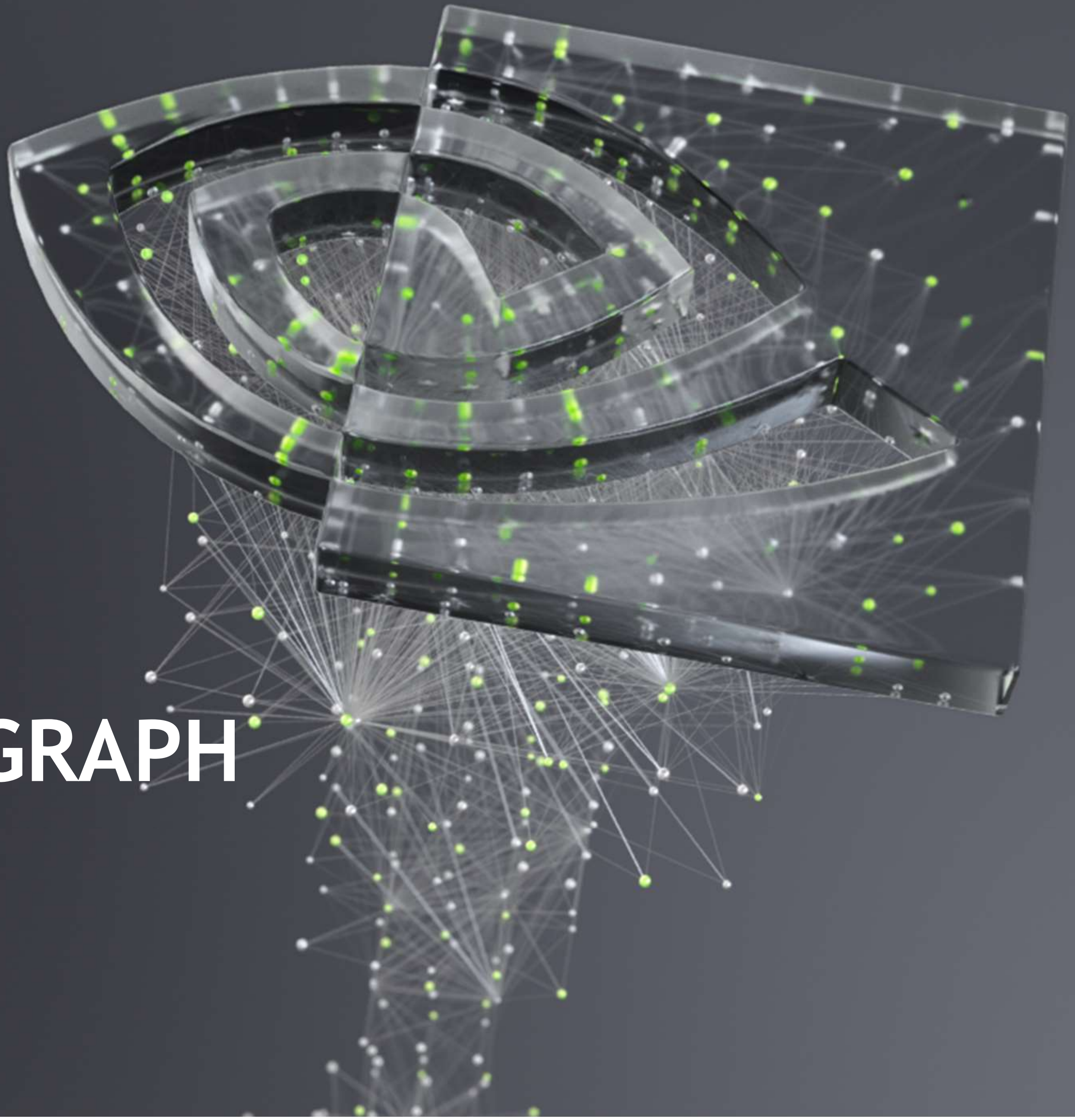


INTRODUCTION TO GRAPH NEURAL NETWORKS

NVIDIA DLI

AGENDA

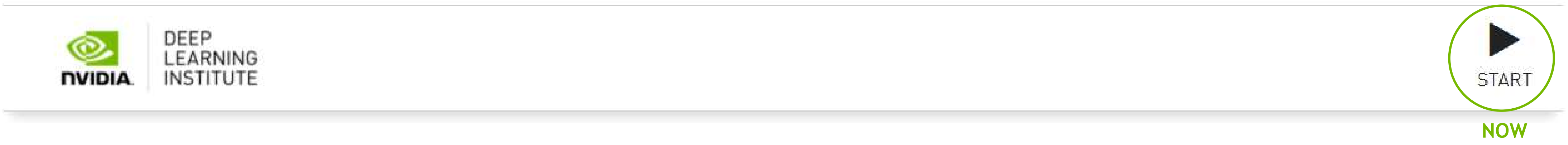
Graph 101

AI: Learning from Graph Data

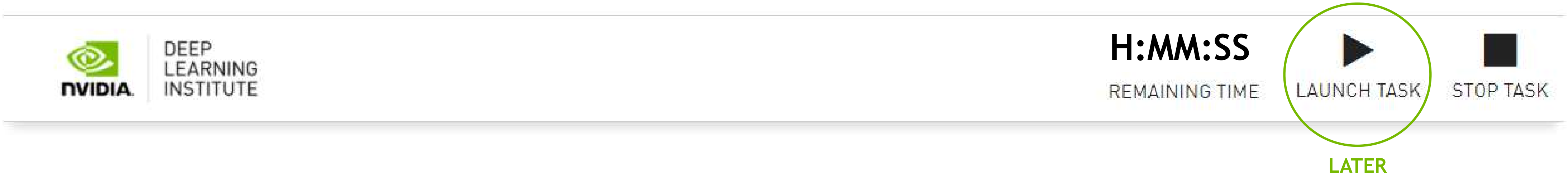
GNN Fundamentals

Hands-On Lab

WELCOME



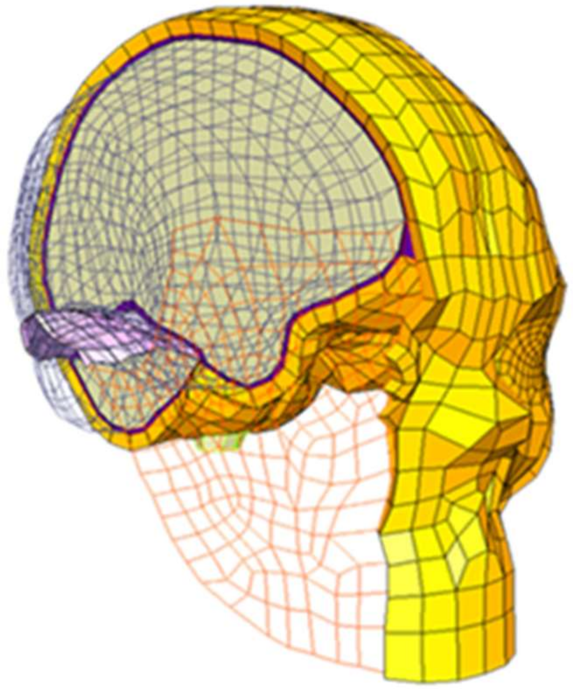
1. A GPU-accelerated server is being launched for interactive activities
2. Datasets, frameworks, and software are being loaded



GRAPH DATA

Used for Representing Unstructured Data

VOLUMETRIC MESHES

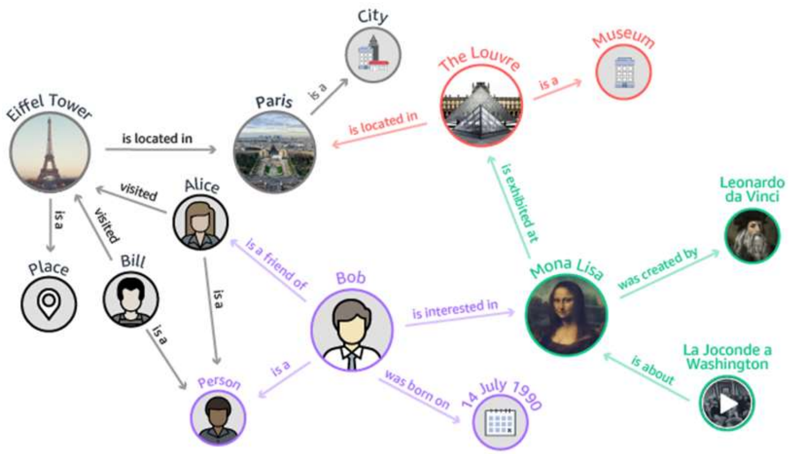


SURFACE MESHES

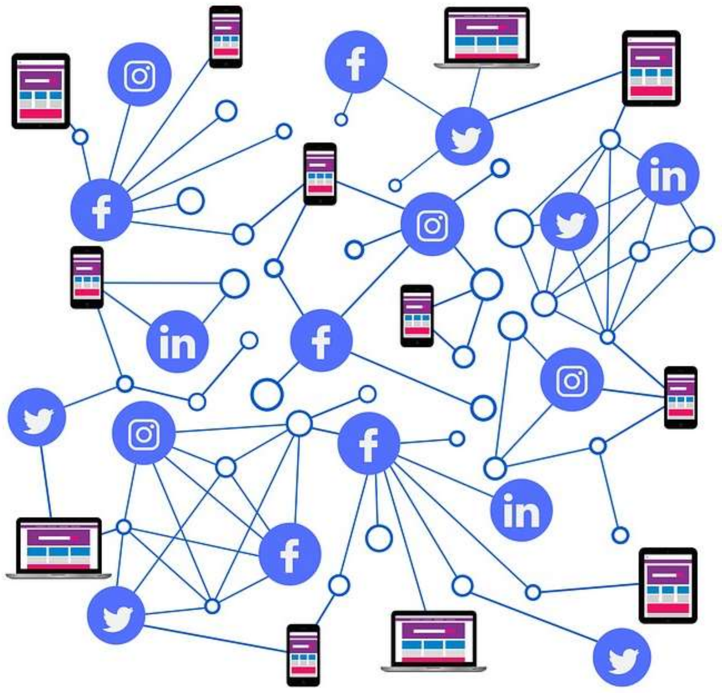
COMMON GRAPH DATA USE CASES

Used to Gather Information About Relationships Between Objects

KNOWLEDGE GRAPHS



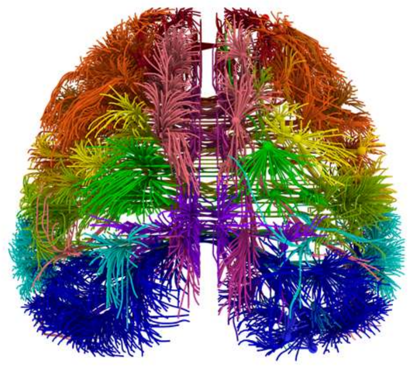
SOCIAL NETWORKS & RECOMMENDER SYSTEMS



COMMON GRAPH DATA USE CASES

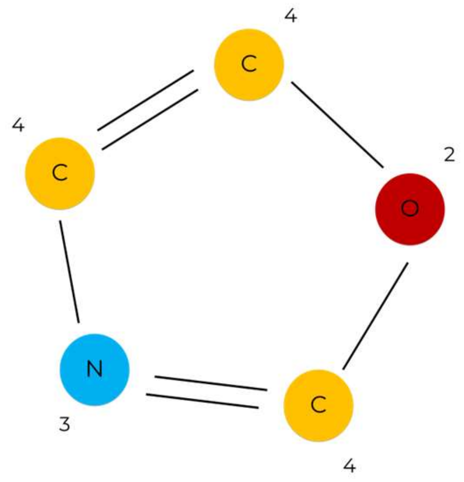
Usage Extends Across Industries

Healthcare



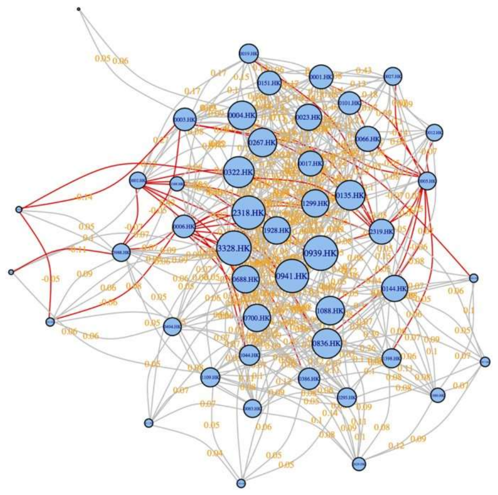
Connectomes
Brian fMRI/DTI

Chemistry & Pharmacy



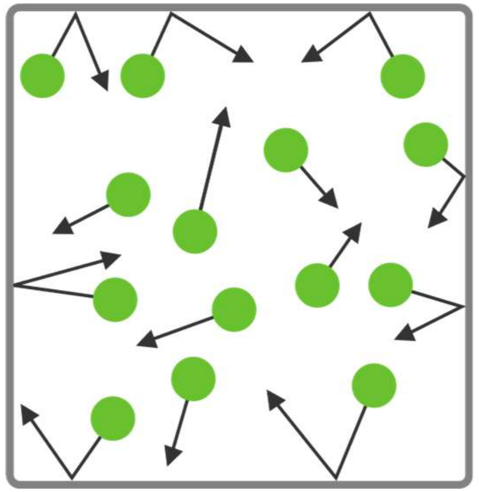
Molecular Analysis
Drug Discovery
Drug Repurposing

Financial Soundness Indicators



Stock Market Prediction

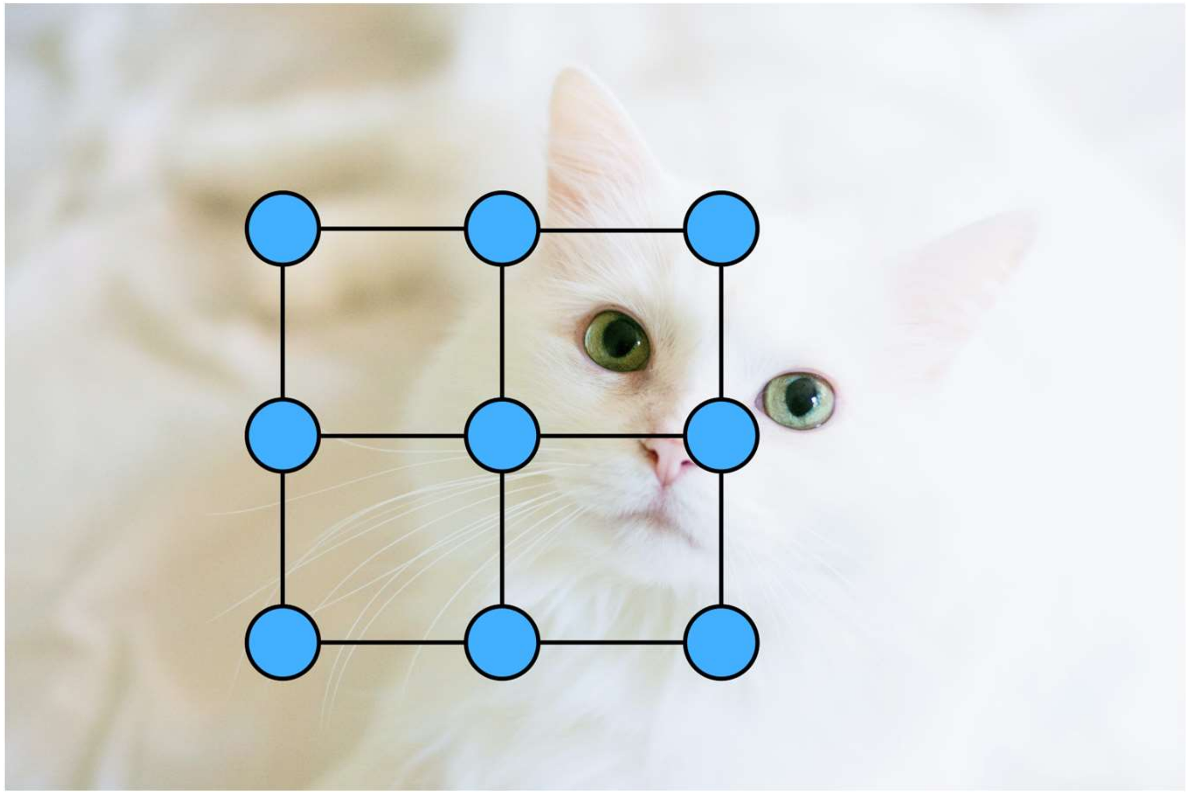
Physics



Particle Systems
Thermodynamics

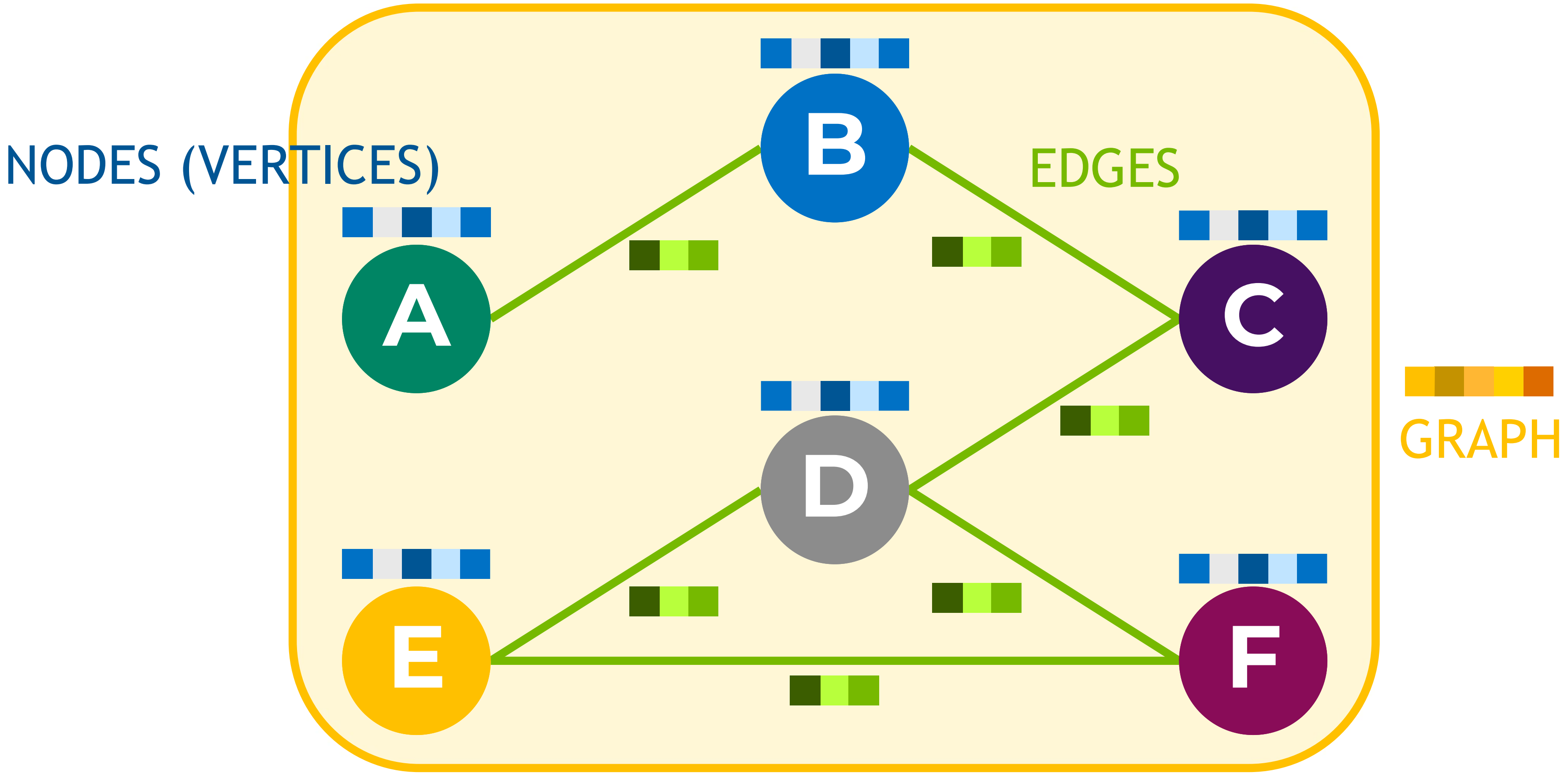
IMAGES - GRID GRAPHS

Treating Images as Graphs Could Present Some Challenges



GRAPH 101

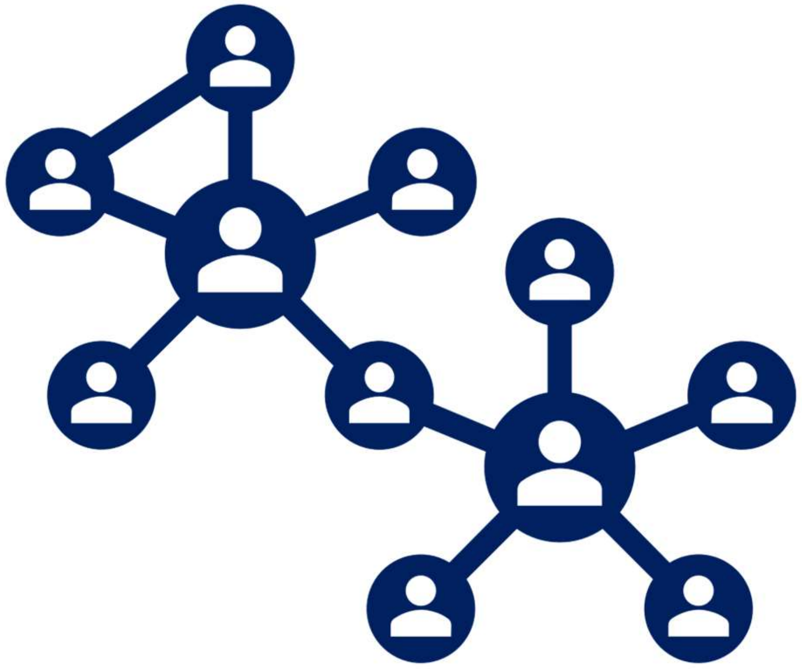
ANATOMY OF A GRAPH



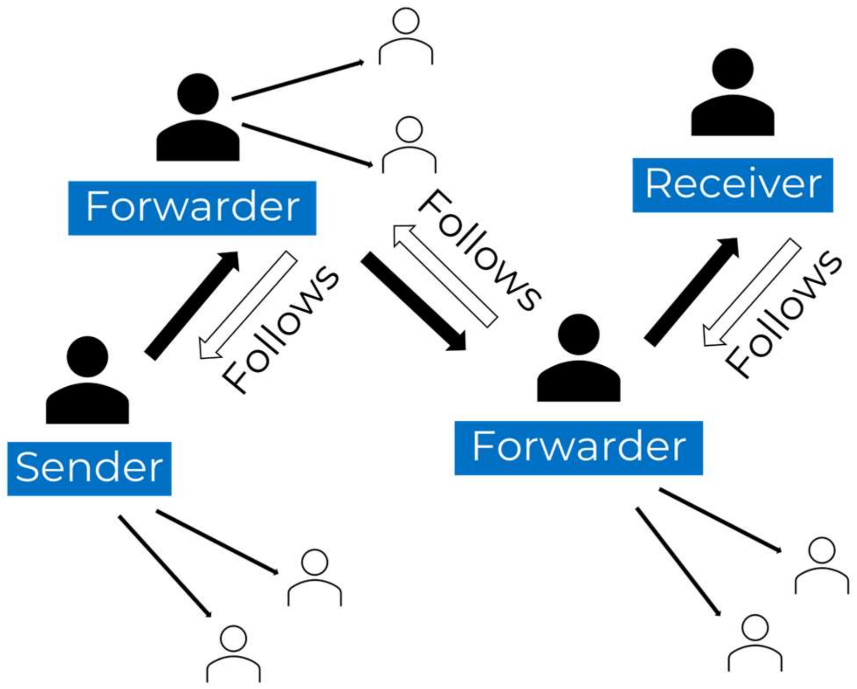
EDGES

Directed, Undirected, Weighted, and Unweighted

FACEBOOK

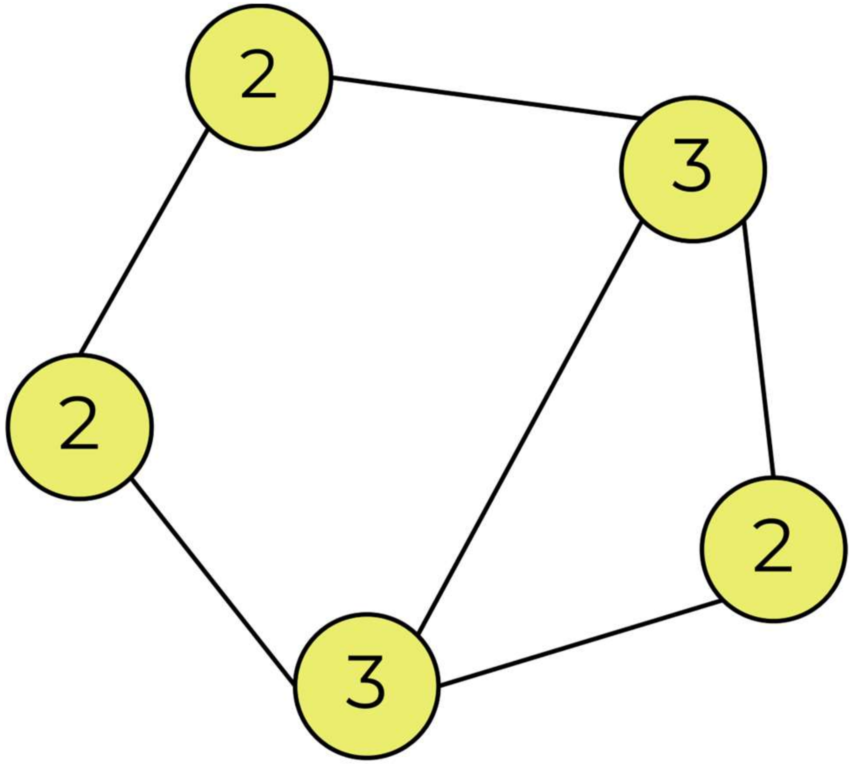


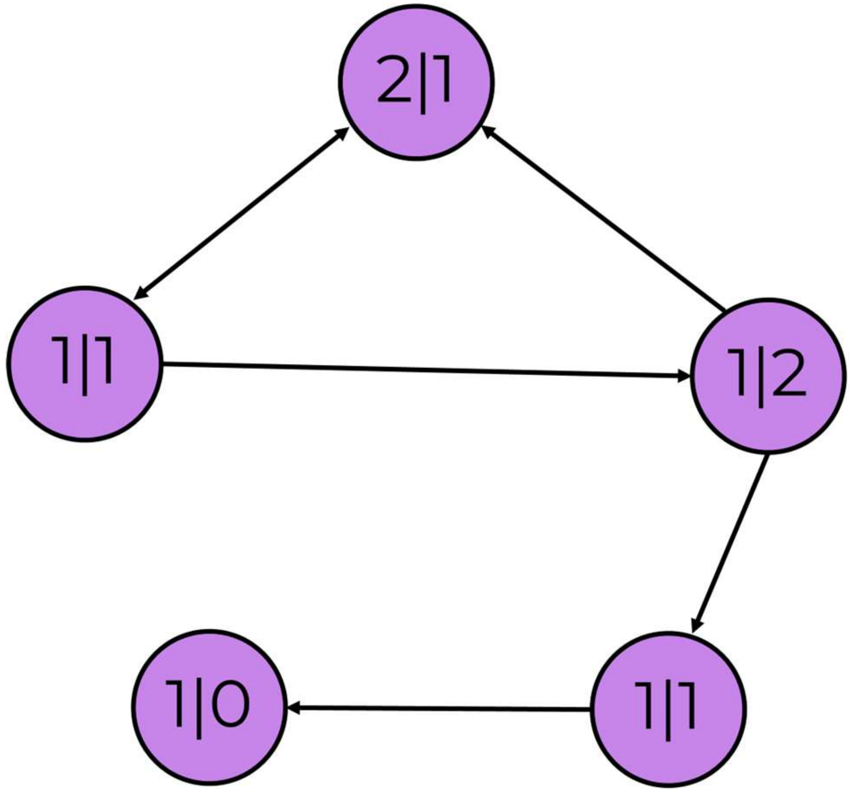
TWITTER



EDGES

Degree, In-Degree, Out-Degree

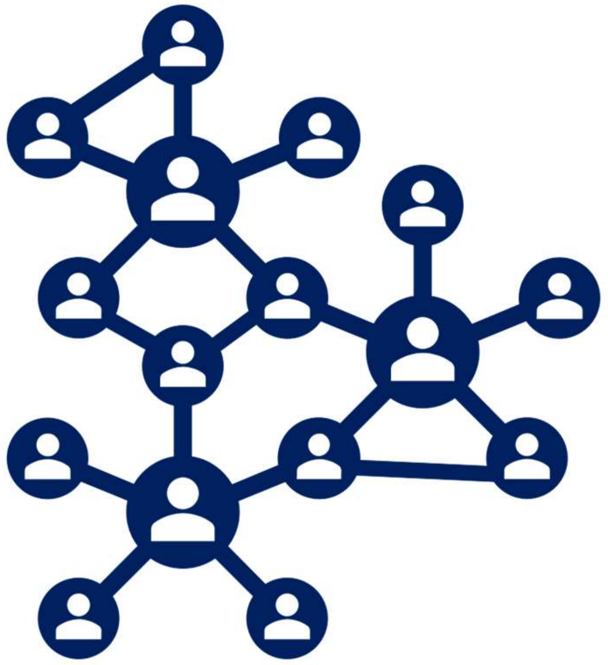




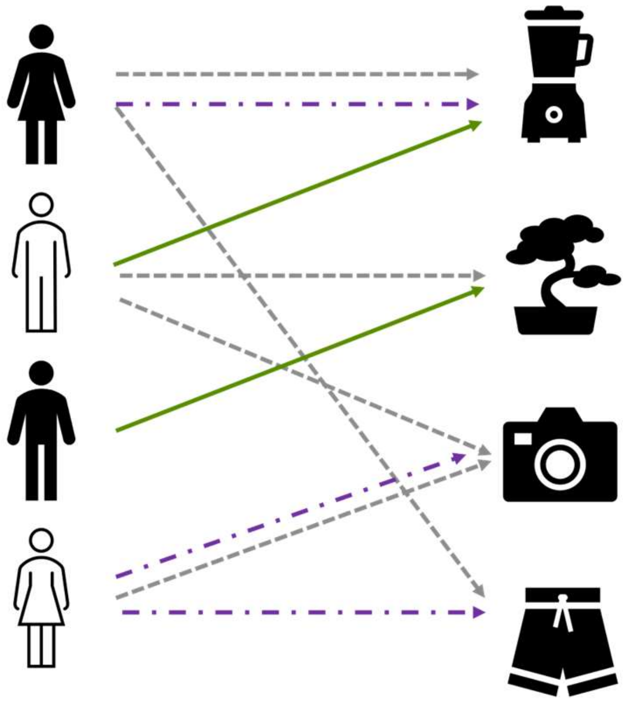
EDGES

Homogeneous vs. Heterogeneous

SOCIAL NETWORK

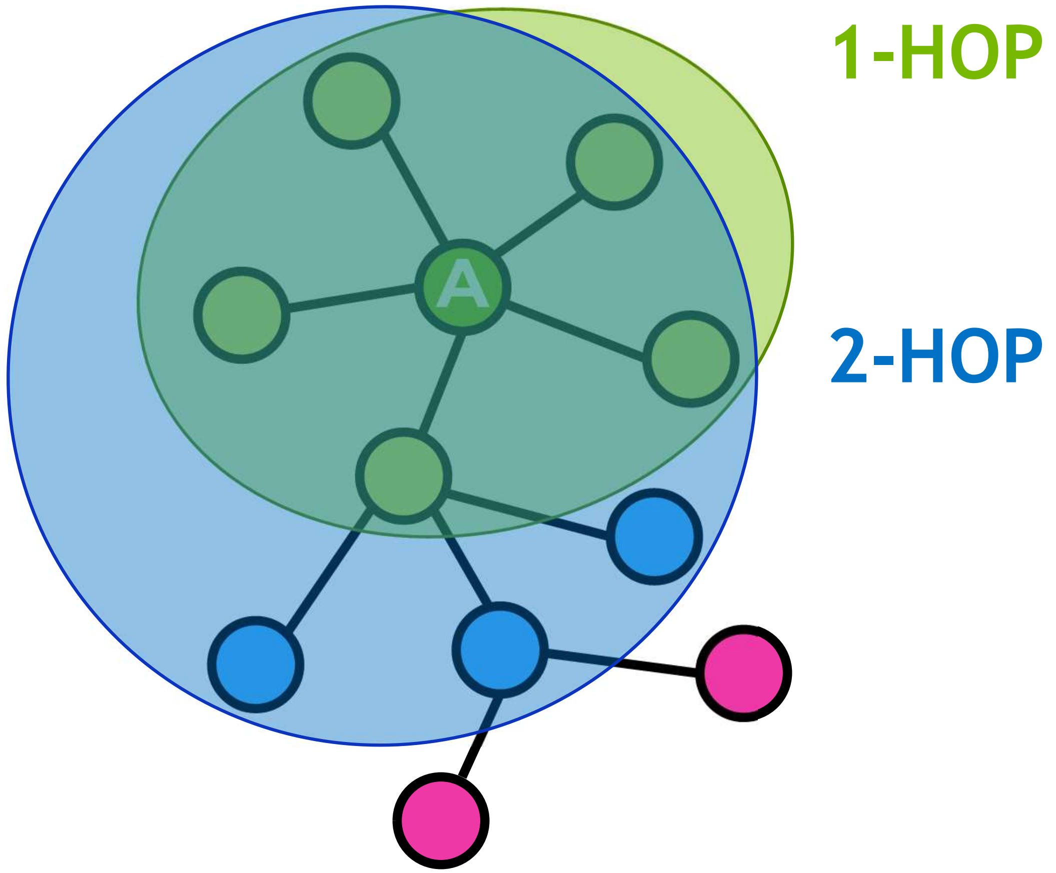


E-COMMERCE



EDGES

Neighborhood

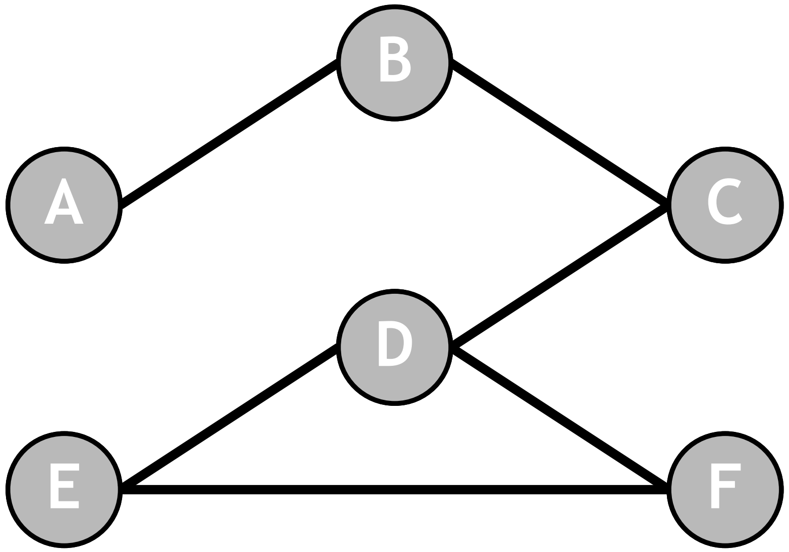


REPRESENTING GRAPHS

Adjacency Matrix Shows the Neighborhood of Each Node

	A	B	C	D	E	F
A	0	1	0	0	0	0
B	1	0	1	0	0	0
C	0	1	0	1	0	0
D	0	0	1	0	1	1
E	0	0	0	1	0	1
F	0	0	0	1	1	0

Note: The adjacency matrix is symmetric for undirected graphs

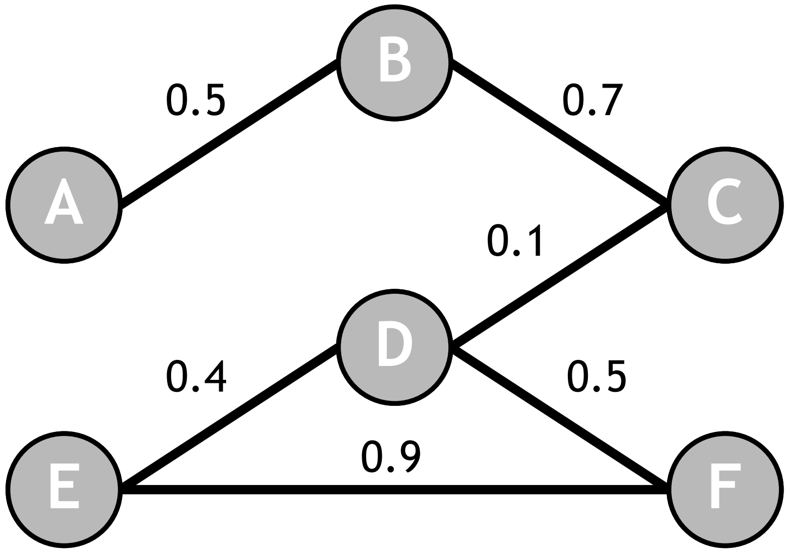


REPRESENTING GRAPHS

Adjacency Matrix Can Also Be Used for a Weighted Graph

	A	B	C	D	E	F
A	0	0.5	0	0	0	0
B	0.5	0	0.7	0	0	0
C	0	0.7	0	0.1	0	0
D	0	0	0.1	0	0.4	0.5
E	0	0	0	0.4	0	0.9
F	0	0	0	0.5	0.9	0

Note: The adjacency matrix is symmetric for undirected graphs

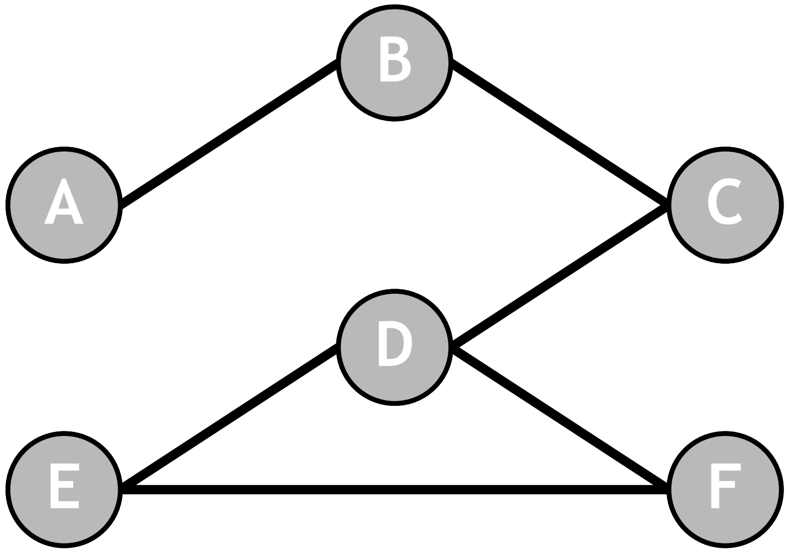


REPRESENTING GRAPHS

Degree Matrix Counts Number of Connections to Each Node

	A	B	C	D	E	F
A	2	0	0	0	0	0
B	0	2	0	0	0	0
C	0	0	2	0	0	0
D	0	0	0	3	0	0
E	0	0	0	0	2	0
F	0	0	0	0	0	2

Note: Illustration assumes unweighted and undirected graph



REPRESENTING GRAPHS

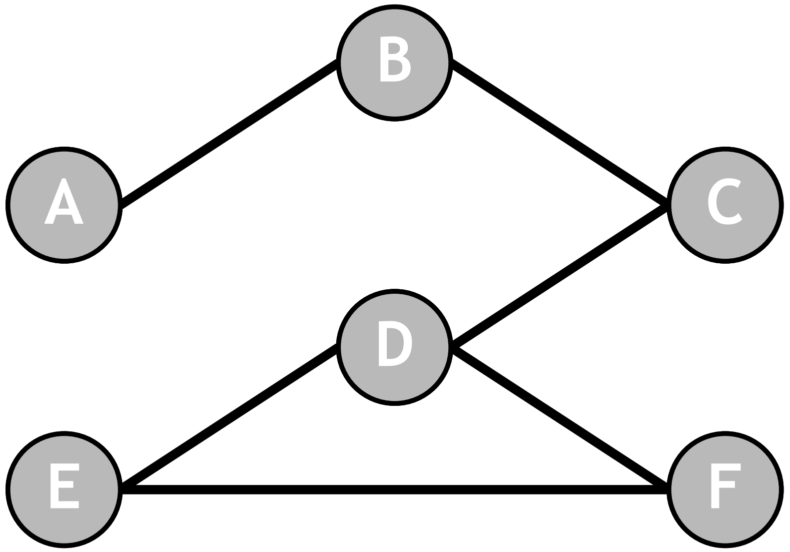
Laplacian Matrix Shows the Smoothness of the Graph

	A	B	C	D	E	F
A	2	-1	0	0	0	0
B	-1	2	-1	0	0	0
C	0	-1	2	-1	0	0
D	0	0	-1	3	-1	-1
E	0	0	0	-1	2	-1
F	0	0	0	-1	-1	2

Represented as L, where

L = { D - A }

D = Degree Matrix and
A = Adjacency Matrix

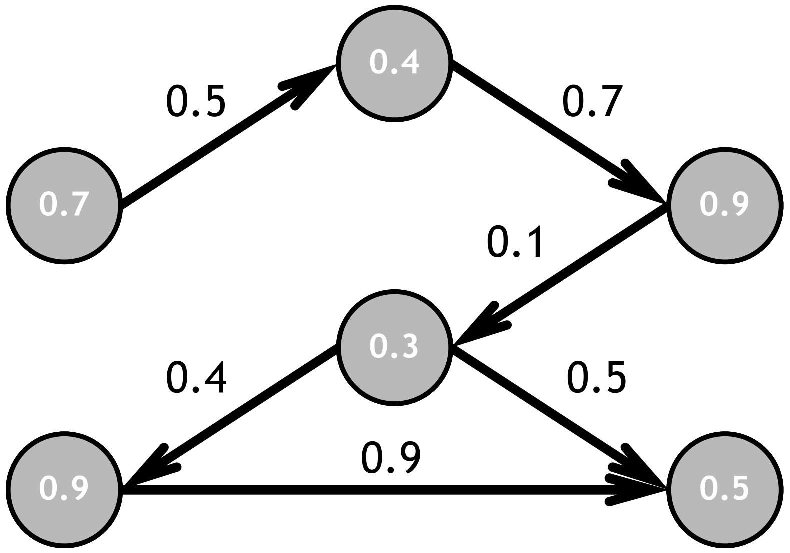


REPRESENTING GRAPHS

Graphs Can Also Be Represented by an Adjacency List

Adjacency	Edges	Nodes
[[1, 2], [2, 3], [3, 4], [4, 5], [4, 6], [5, 6]]	[0.5, 0.7, 0.1, 0.4, 0.5, 0.9]	[0.7, 0.4, 0.9, 0.3, 0.9, 0.5]

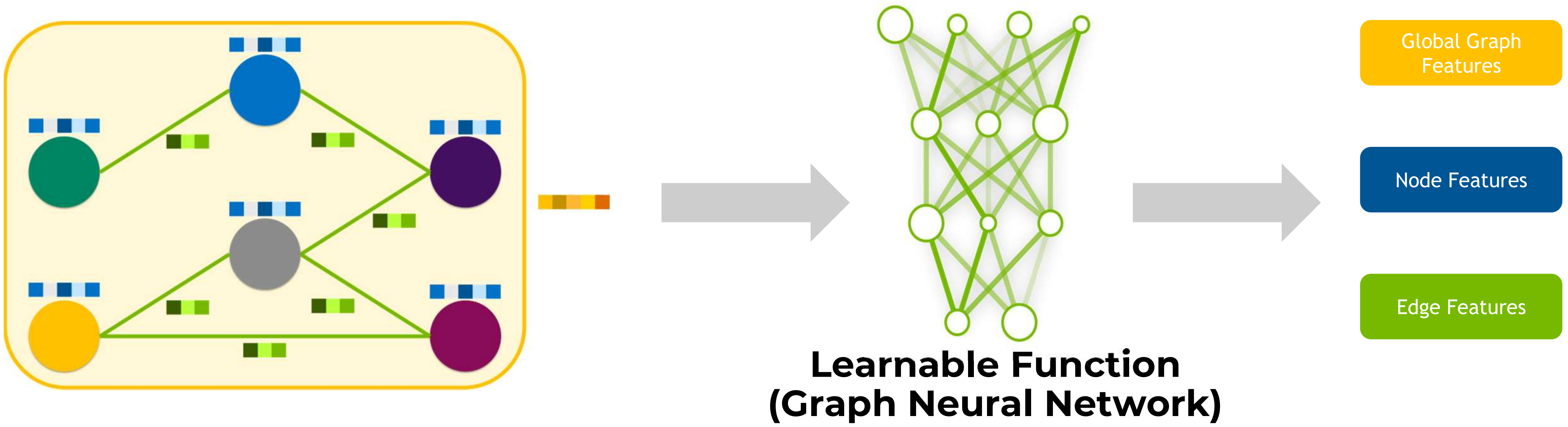
Note: Representing graph structure as adjacency lists can be more efficient



AI: LEARNING FROM GRAPH DATA

GRAPH NEURAL NETWORKS

Learn the Function(s) to Predict Graph-Level, Node-Level, or Edge-Level Features



GRAPH NEURAL NETWORKS

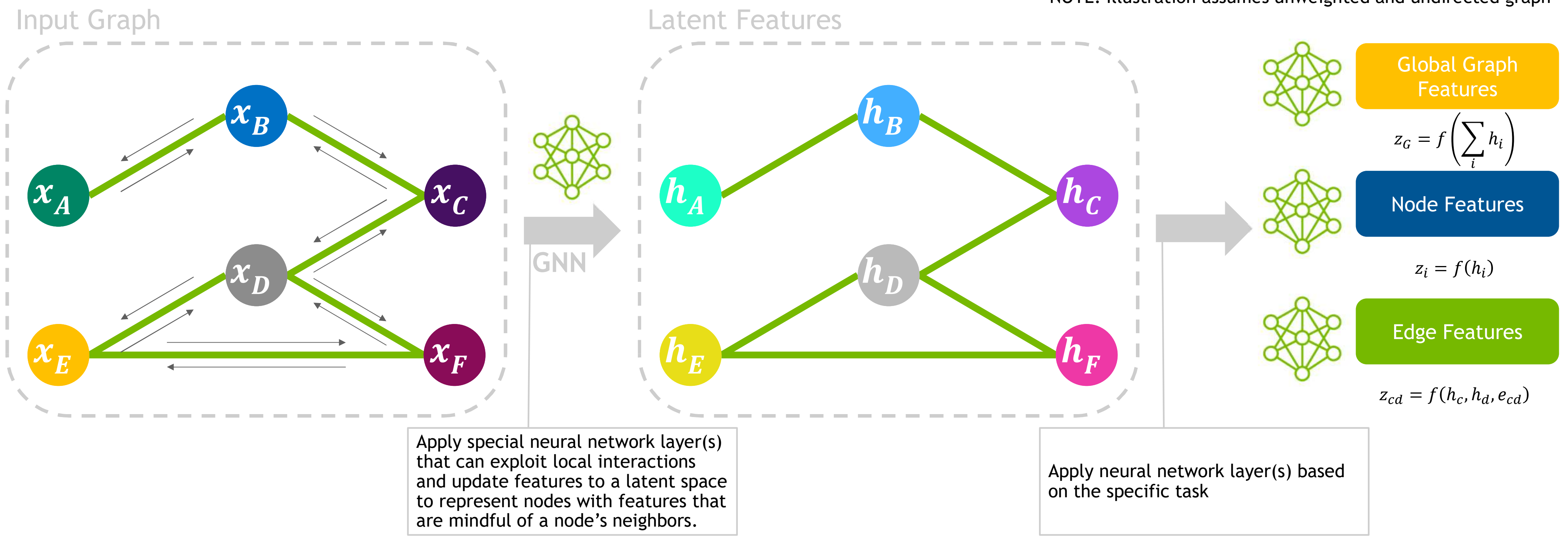
Neural Networks That Operate on Graphs

- Graph data is a challenge as standard deep learning methods focus primarily on structured data, such as fixed-size pixel grids (images) and sequences (text).
- Graph neural networks, or GNN, refers to a variety of different approaches for applying deep learning on graphs that:
    - Take full advantage of the graph structure
    - Considers scalability and efficiency based on the size of the graph and its features
- Graph neural networks that use various techniques, which typically involve learning the latent / embedding space(s) to represent node features that are mindful of a node’s direct neighborhood in the graph, have become the most popular approaches:
    - Message Passing Neural Networks (MPNN)
    - Graph Convolution Networks (GCN)
    - Graph Attention networks (GAT)

GNN TASKS

Learn the Function(s) to Predict Graph-Level, Node-Level, or Edge-Level Features

NOTE: Illustration assumes unweighted and undirected graph

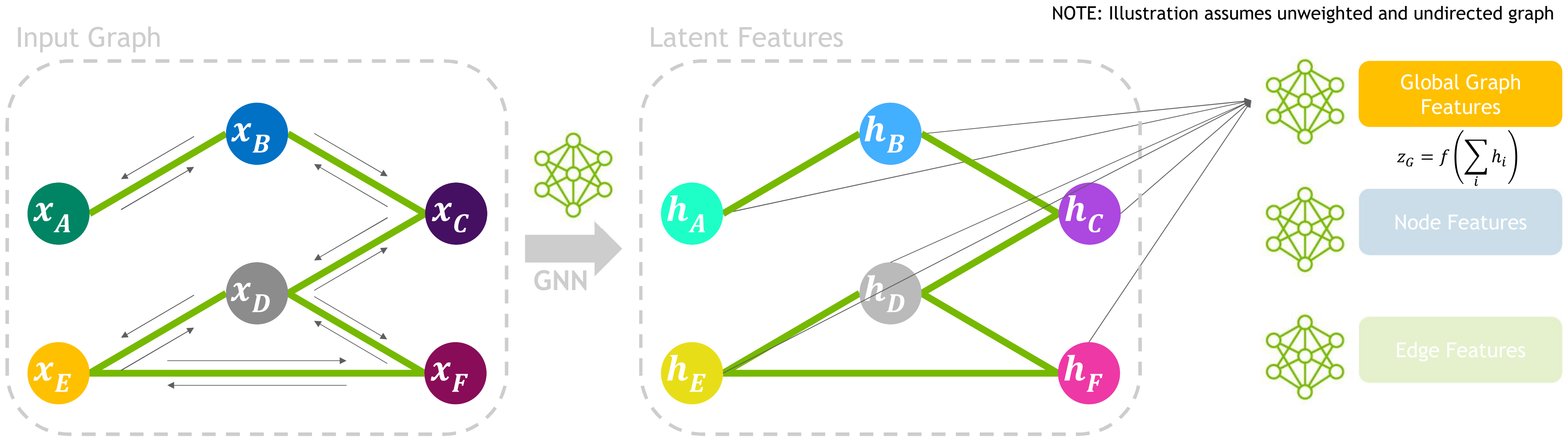


Apply special neural network layer(s) that can exploit local interactions and update features to a latent space to represent nodes with features that are mindful of a node's neighbors.

Apply neural network layer(s) based on the specific task

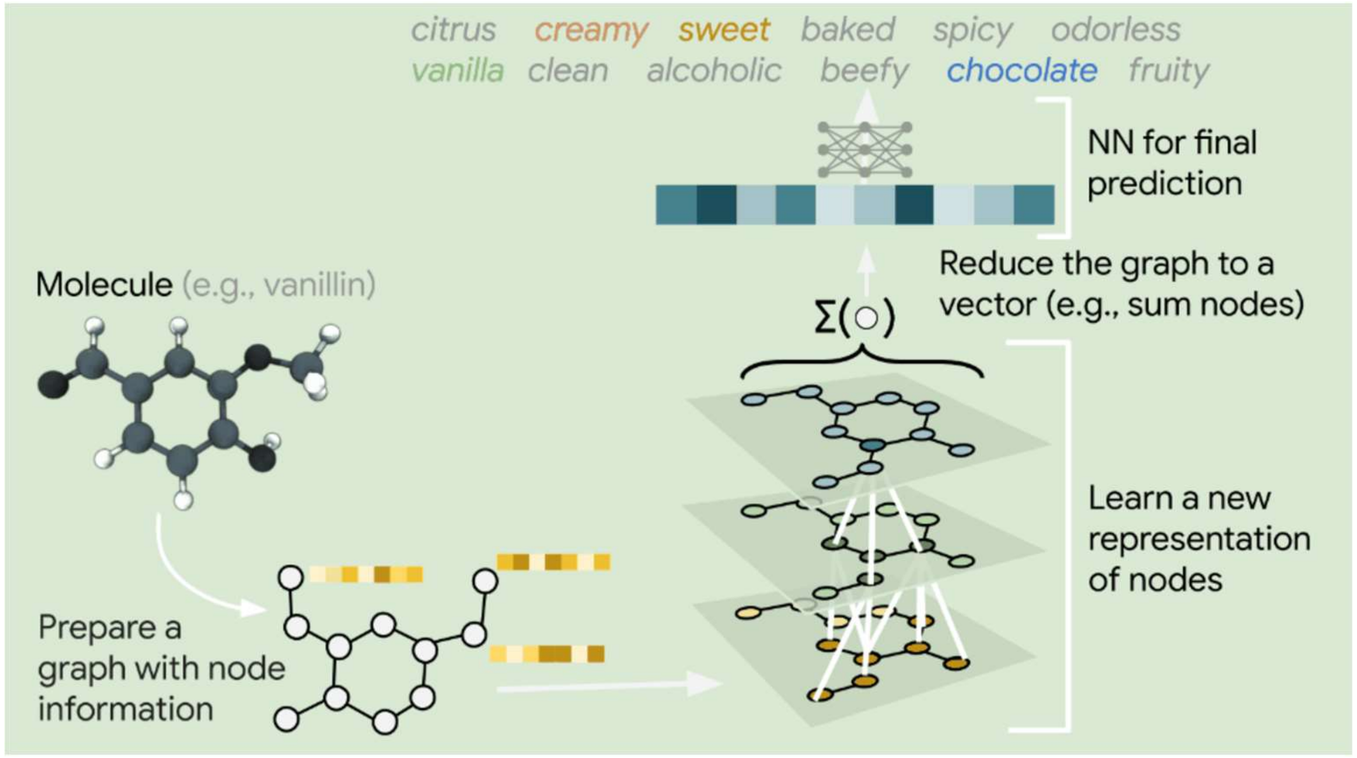
GRAPH-LEVEL TASK

Use (Latent) Node Features and Edge Features to Predict a Global Graph Property



EXAMPLE GRAPH-LEVEL TASK

Google Experimented with Using GNN to Perform Classification at the Graph Level



citrus
creamy
sweet
baked
spicy
odorless
vanilla
clean
alcoholic
beefy
chocolate
fruity
NN for final prediction
Reduce the graph to a vector (e.g., sum nodes)
$\Sigma(\circ)$
Molecule (e.g., vanillin)
Learn a new representation of nodes
Prepare a graph with node information

Image credit: ai.googleblog.com

Objective:

To predict the smell of molecules by framing molecules as graphs

EXAMPLE GRAPH-LEVEL TASKS

MIT Successfully Discovered *Halicin*’s Effectiveness Against Multi-Resistant Bacteria

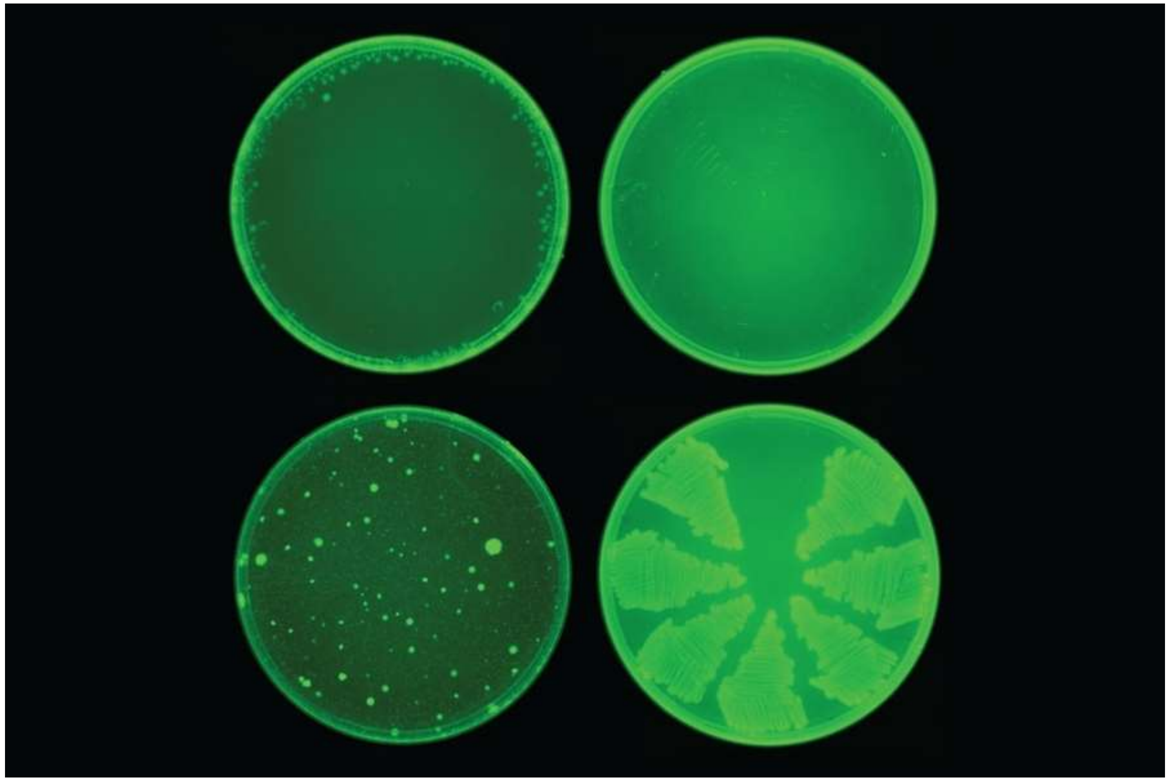


Image credit: news.mit.edu

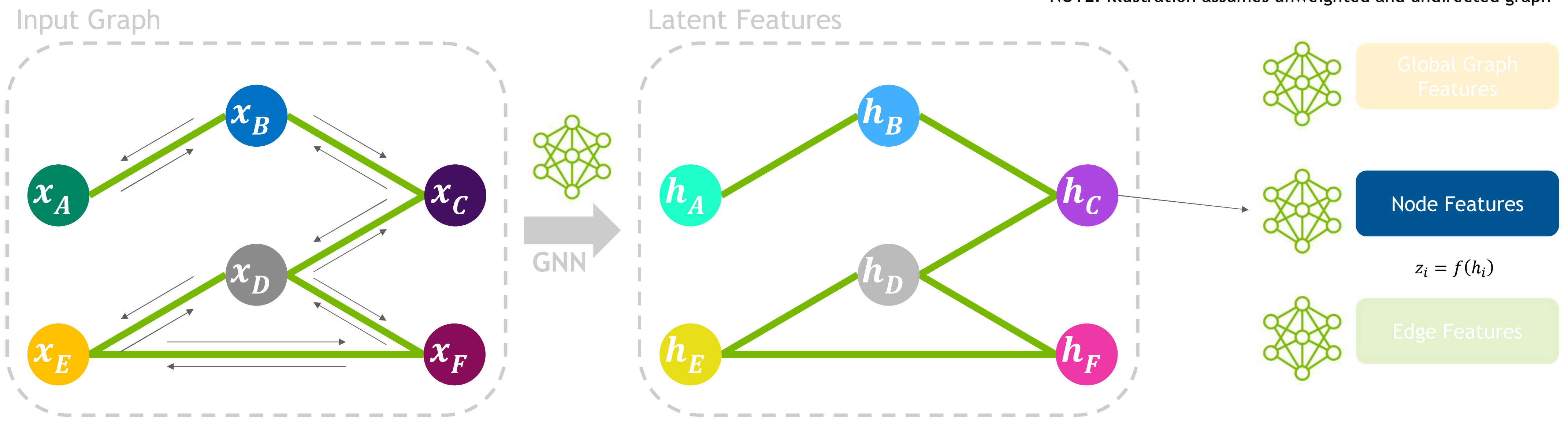
Objective:

To predict molecule toxicity rating

NODE PREDICTION

Take (Latent) Node Features and Edge Features to Predict a Node Property

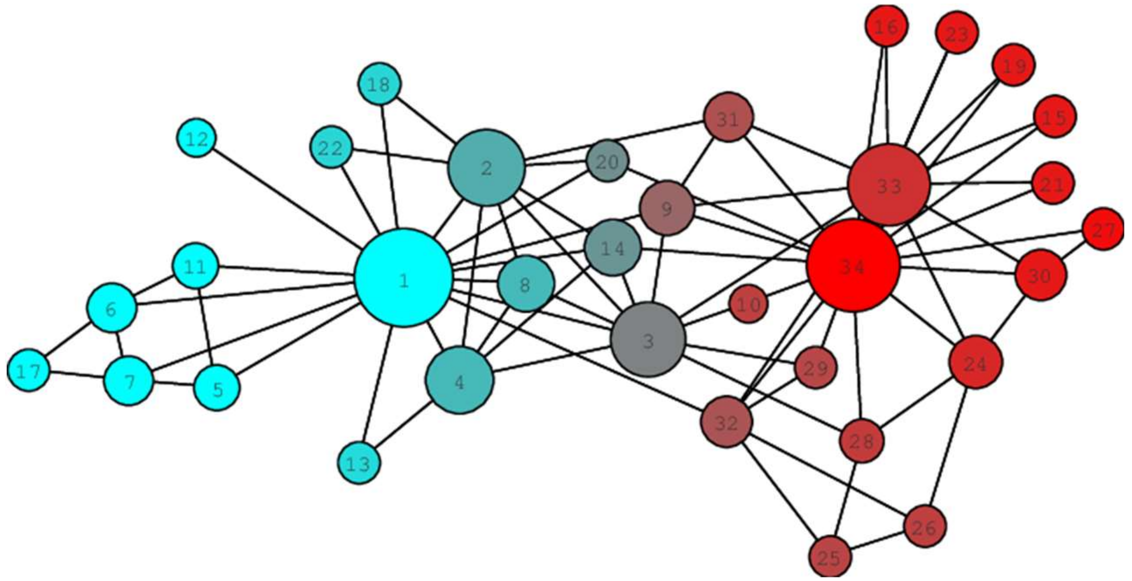
NOTE: Illustration assumes unweighted and undirected graph



NODE-LEVEL TASKS

Classification

Zachary Karate Club – the 'Hello World' of GNNs



NODE-LEVEL TASKS

Regression

Predict values in each node: continuous outcome variables

Research at Cornell: utilizing geospatial and temporal information for crop yield prediction

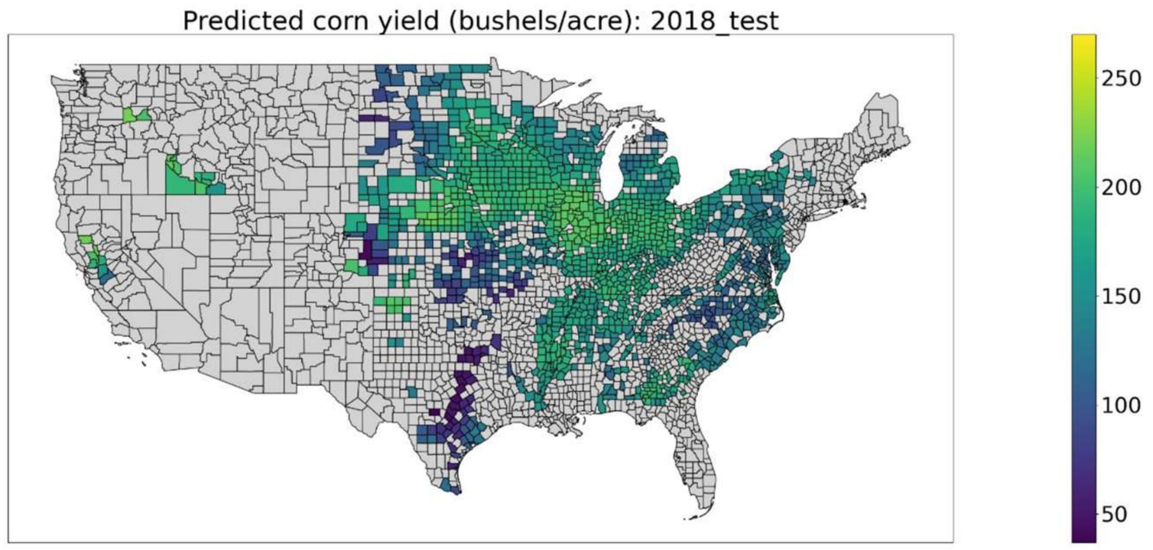
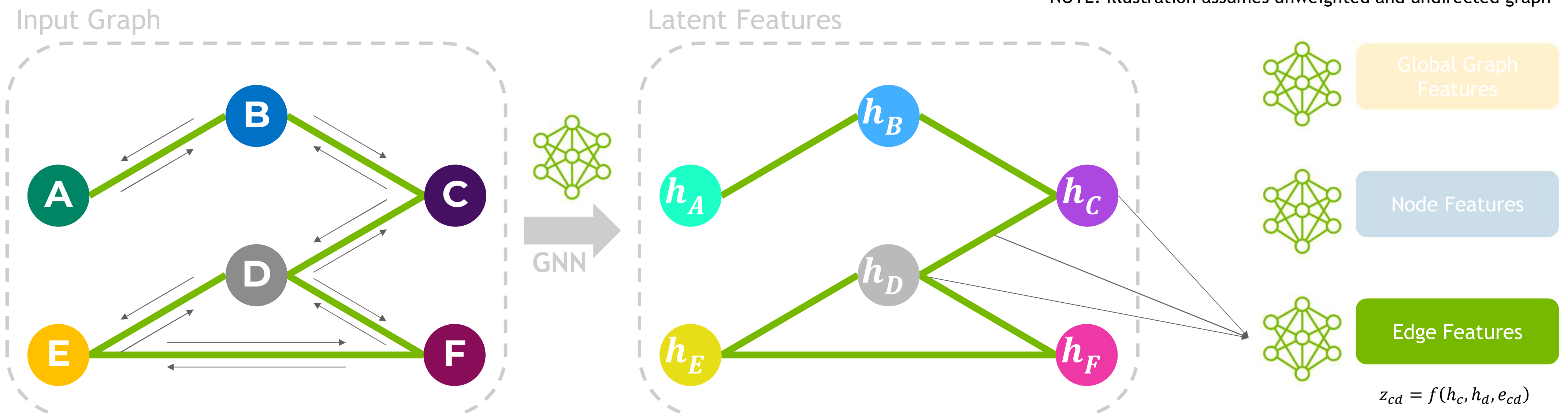


Image credit: proceeding NeurIPS 2021

LINK/EDGE PREDICTION

Predict the Weight and Class of Graph Edges

NOTE: Illustration assumes unweighted and undirected graph



EDGE-LEVEL TASKS

Link prediction

E-commerce / digital media: recommend product / content A to a customer / user B

Social network: recommend friends

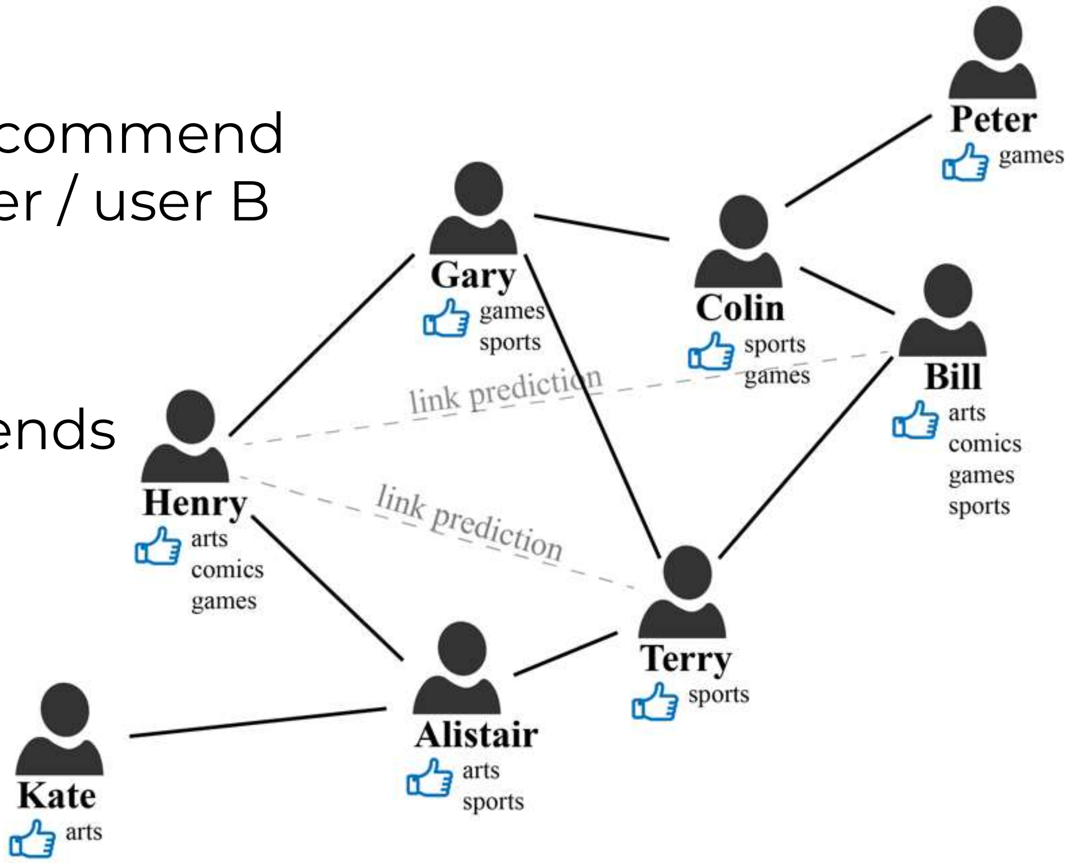


Image credit: aws.amazon.com/blogs

EDGE-LEVEL TASKS

Classification

Predict fradulent transactions:

APATE: A novel approach for automated credit card transaction fraud detection using network-based extensions

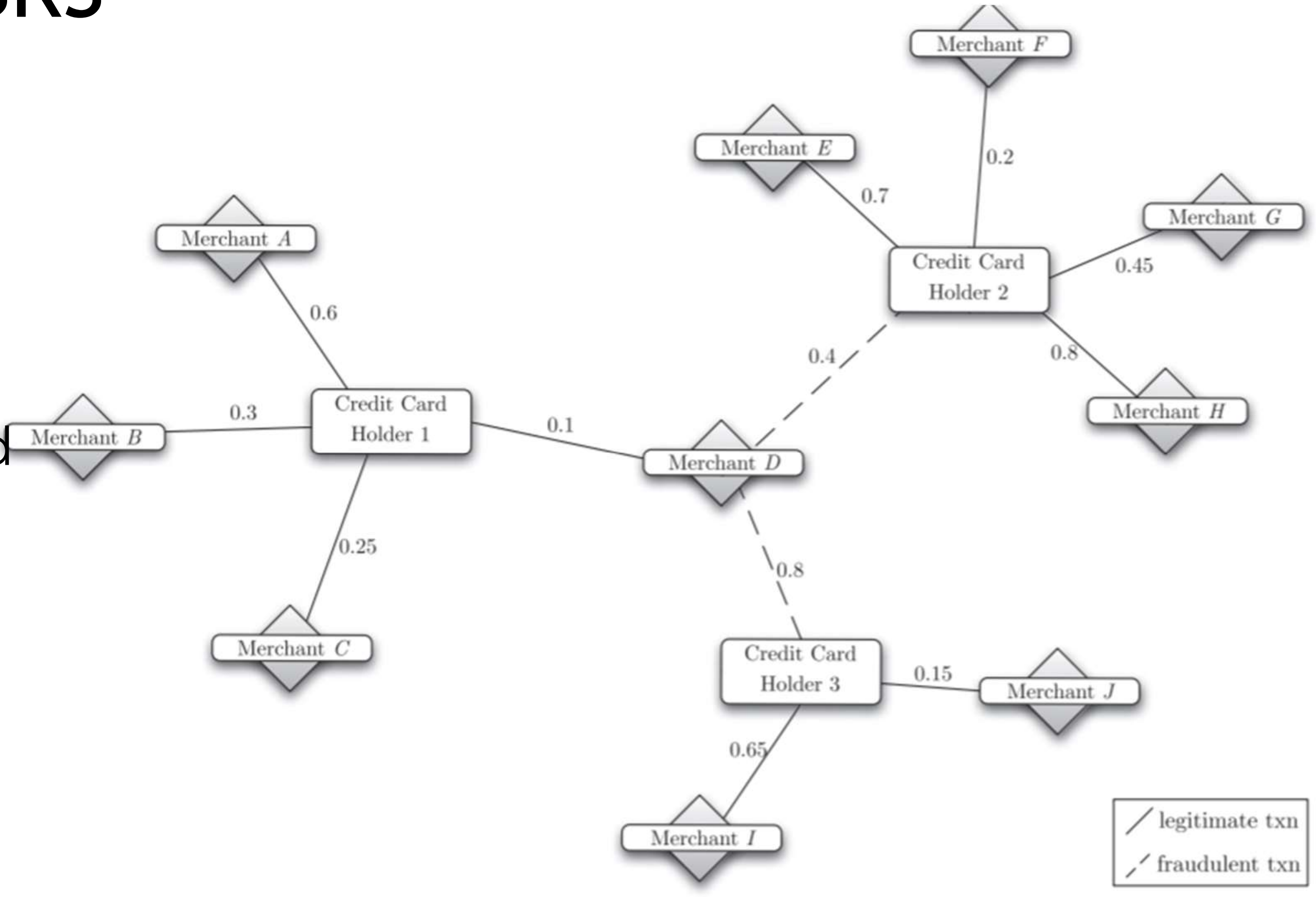
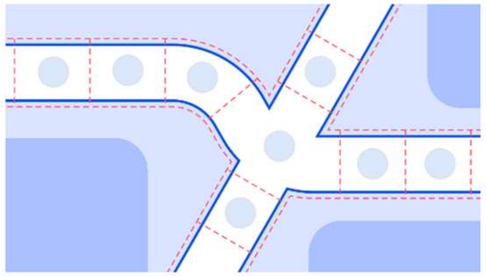


Image credit: Decision Support Systems 2015

EDGE-LEVEL TASKS

Regression

Google Maps: estimate time to travel from A to B



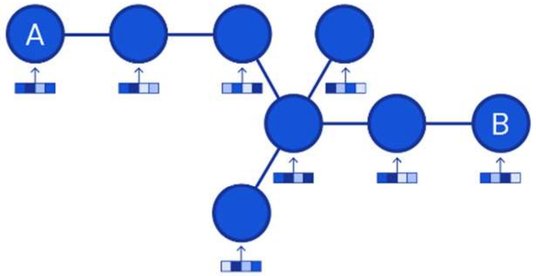
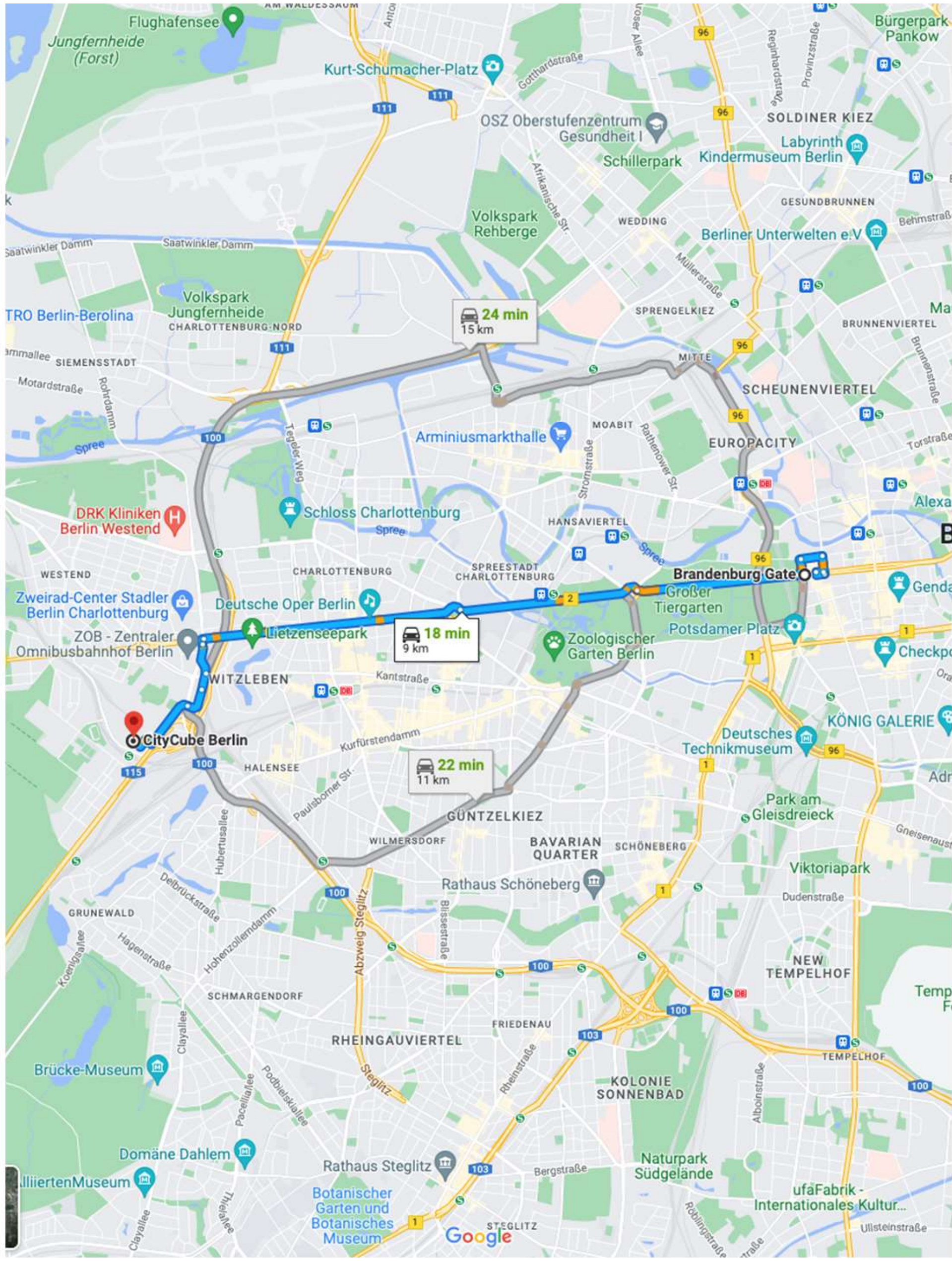


Image credit: DeepMind



GNN FUNDAMENTALS

MESSAGE PASSING AND GRAPH CONVOLUTION

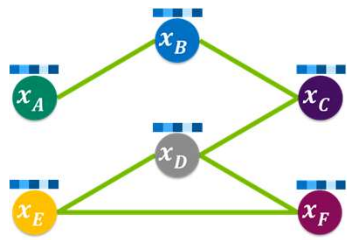
Principal Ideas Behind Most GNN Architectures

- **Message passing** is when each node in a graph sends information about itself to its neighbors and receives messages back from them to update its status and understand its environment
    - Enables nodes to aggregate neighboring features and update its own
- Convolution is popularly used to analyze images, which is generalized to be applicable on graphs
- **Graph convolution** combines information from neighborhoods and encodes updated node features to embeddings
    - Various aggregation methods to handle different use cases
    - The aggregation method should be **permutation invariant**, i.e. independent from the ordering of the nodes
    - Can be thought of as a simple message passing algorithm that involves a linear combination of neighborhood features where weights used for the aggregation depend only on the structure of the graph

GNN: SUM-POOLING

Sum-Pooling Update Rule

- A simple sum aggregation of the neighborhoods can be done efficiently by multiplying the feature vectors and the adjacency matrix together



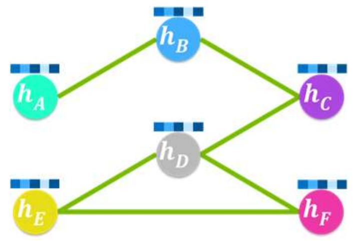
Note: Assuming graph is *unweighted* and *undirected*, i.e. adjacency matrix is *binary* and *symmetrical*

$$\begin{array}{c|cccccc} & A & B & C & D & E & F \\ \hline A & 1 & 1 & 0 & 0 & 0 & 0 \\ B & 1 & 1 & 1 & 0 & 0 & 0 \\ C & 0 & 1 & 1 & 1 & 0 & 0 \\ D & 0 & 0 & 1 & 1 & 1 & 1 \\ E & 0 & 0 & 0 & 1 & 1 & 1 \\ F & 0 & 0 & 0 & 1 & 1 & 1 \end{array} \times \begin{bmatrix} x_A \\ x_B \\ x_C \\ x_D \\ x_E \\ x_F \end{bmatrix} = \begin{bmatrix} x_A + x_B \\ x_A + x_B + x_C \\ x_B + x_C + x_D \\ x_C + x_D + x_E + x_F \\ x_D + x_E + x_F \\ x_D + x_E + x_F \end{bmatrix}$$

$$\tilde{A} \qquad H \qquad H'$$

Where $\tilde{A}$ is the adjacency matrix after adding the identity matrix to enforce that a node is always connected to itself and keep the context

The adjacency matrix masks out all the values except for the one(s) that the node has connection with. This operations sum the values of the nodes connections and ignored all the others



- Leverage the power of deep learning by giving some layered processing of the features, therefore also multiplying by some learnable node-wise shared linear transformation, W, as well as applying a non-linear function, σ, to make the feature representation more complex

- $H' = \sigma(\tilde{A}HW)$
    - This effectively recombines the information in the neighborhood into just one vector and the output of this dense layer is the new vector representation of the nodes

GNN: MEAN-POOLING

Mean-Pooling Update Rule

- Graph neural networks that use a mean-pooling update rule normalizes the vector to prevent features from *exploding* since the scale of the output features can increase
    - Becomes a more obvious problem when stacking layers
- Multiplying by the inverse of the degree matrix, which measures the number of connections for each node (i.e. neighborhood size), to arrive at the mean
- $\acute{H} = \sigma(\widetilde{D}^{-1}\tilde{A}HW)$
    - This effectively takes the average of all neighborhood features

GNN: SYMMETRIC NORMALIZATION

Popular GCN Proposed by Kipf and Welling

- The commonly cited **graph convolutional network (GCN)** uses **symmetric normalization** in the update rule
    - Multiplying by the inverse square root of the degree matrix from both sides, which is calculated by dividing by the square root of the product of neighborhood sizes of a node and neighborhood sizes of the neighbor.
- Currently the most popular graph convolution layer
    - Simple to implement
    - Scalable
    - Powerful
- $H' = \sigma(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}HW)$

MESSAGE PASSING NEURAL NETWORKS

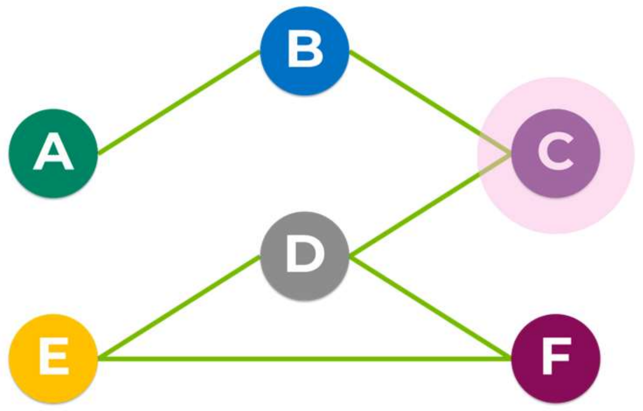
Nodes Can Send and Receive Messages Along Graph Edges

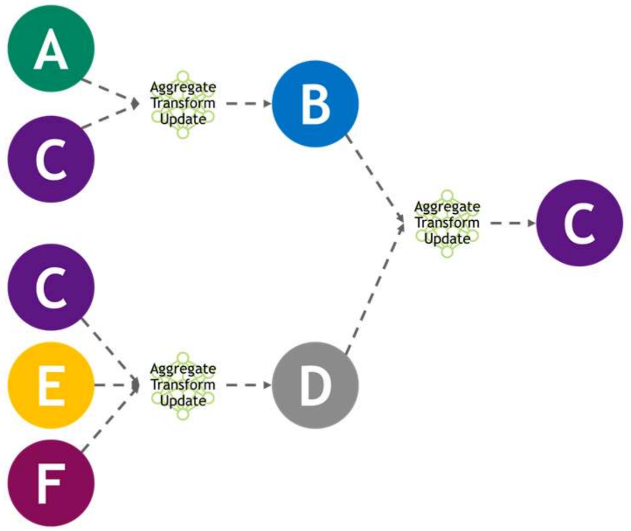
- Previous methods described (i.e. graph convolutional networks) only indirectly supports edge features
- A more generalized **message passing neural network (MPNN)** can focus on edge-wise mechanisms, i.e. for a single node in the graph:
  - Let message sent across edge $i \rightarrow j$ be defined as $m_{ij} = f_e(h_i, h_j, e_{ij})$
  - Aggregate all messages sent to node i using a **permutation-invariant** function such that $h_i' = f_v(h_i, \sum_{j \in N_i} m_{ij})$
  - f_e and f_v can be neural networks
- Powerful GNN approach but requires storage and manipulation of edge messages, which can be costly

STACKING LAYERS

Getting Signals From More Distant Neighborhoods in the Graph

- By repeating the step N times for all nodes in the graph, the feature representations are updated with information of nodes up to N hops away
    - Often treated as a hyperparameter for model tuning

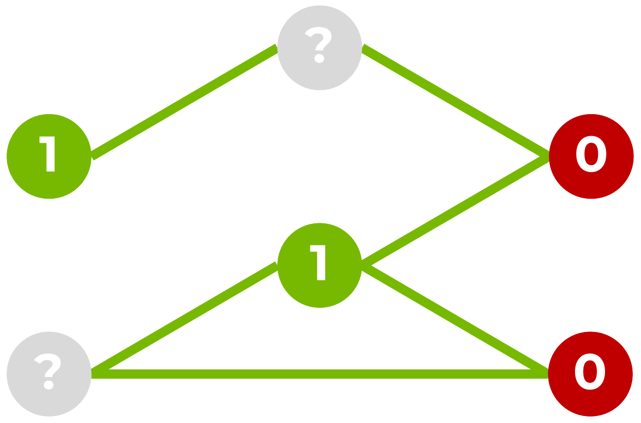




GNN TRAINING

Training Is Done in a Similar Way as Other Neural Networks

- Learning is done by gradient-descent to minimize the loss function, typically implemented as backpropagation
- GNN methods commonly use a semi-supervised learning approach
- Depending on the use case, **transductive learning** may be used
    - Train the neural network model and label unlabeled points which we have already encountered
    - Compute embeddings using the entire graph
    - Can only perform inference on nodes that the model has encountered before



HANDS-ON LAB