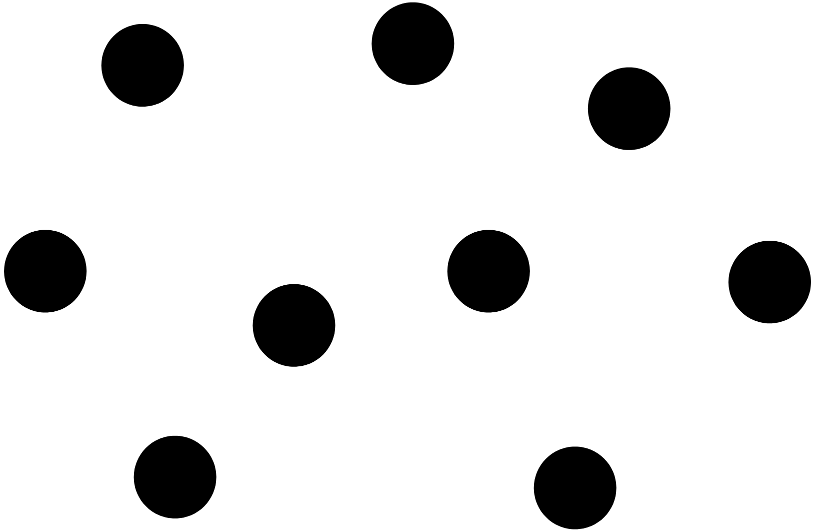
Carnegie Mellon University

Introduction to Deep Learning for Engineers

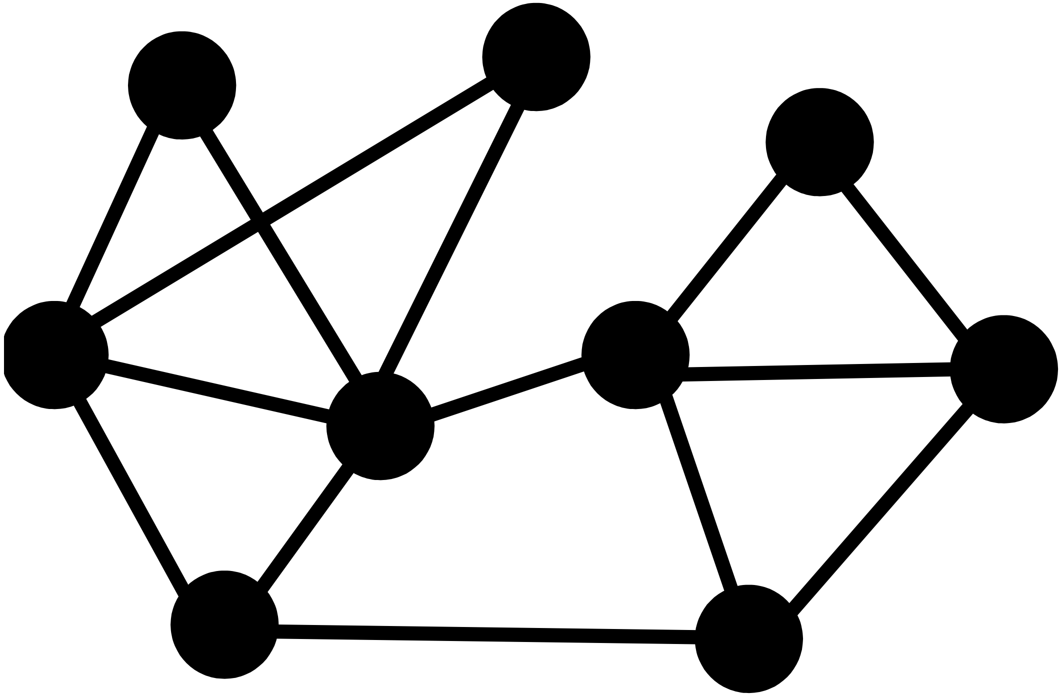
Spring 2025, Deep Learning for Engineers
Feb 13, 2025, 10th Session

Amir Barati Farimani
Associate Professor of Mechanical Engineering and Bio-Engineering
Carnegie Mellon University

Data

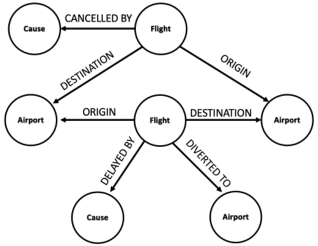


Relations between Data



Graphs

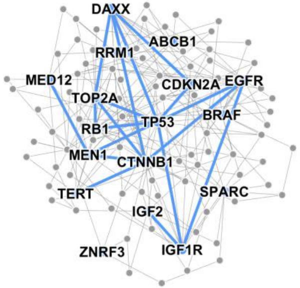
Relations between Data



Event Graphs

Image credit: SalientNetworks

Computer Networks



Disease Pathways

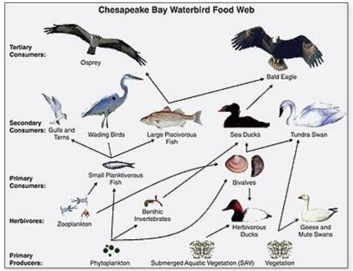


Image credit: Wikipedia

Food Webs

Image credit: Pinterest

Particle Networks

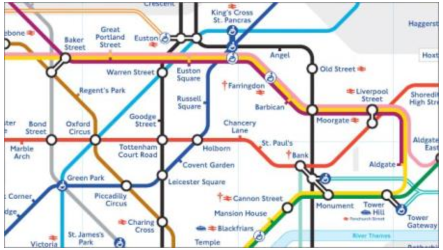


Image credit: visitlondon.com

Underground Networks

Relations between Data

Image credit: Medium

Social Networks

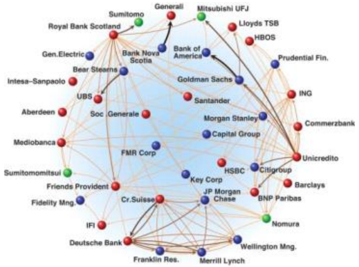


Image credit: Science

Economic Networks

Image credit: Lumen Learning

Communication Networks

Citation Networks

Image credit: Missoula Current News

Internet

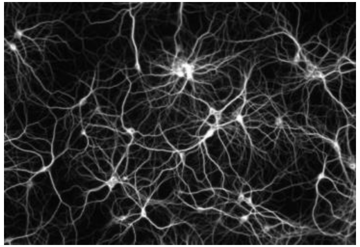


Image credit: The Conversation

Networks of Neurons

Relations between Data

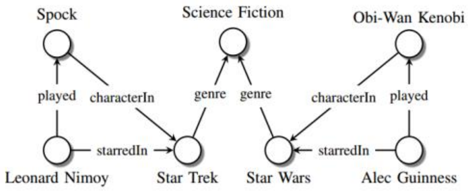


Image credit: Maximilian Nickel et al

Knowledge Graphs

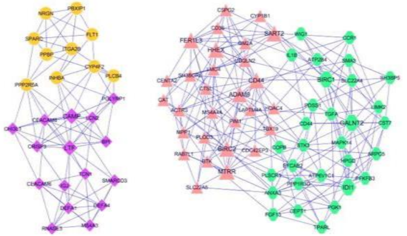


Image credit: ese.wustl.edu

Regulatory Networks

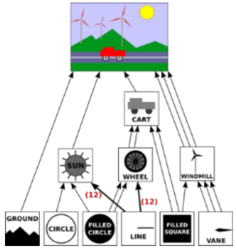


Image credit: math.hws.edu

Scene Graphs

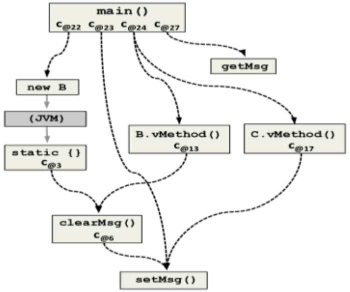


Image credit: ResearchGate

Code Graphs

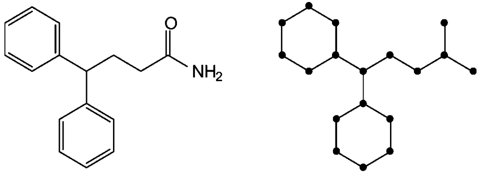


Image credit: MDPI

Molecules

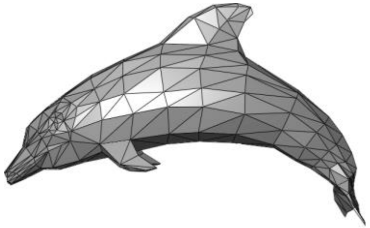
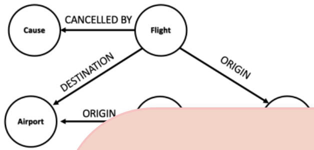
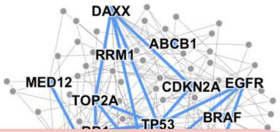


Image credit: Wikipedia

3D Shapes

Relations between Data

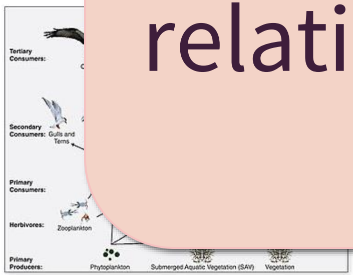




Ev

How can we learn the relation ship between data?

ys



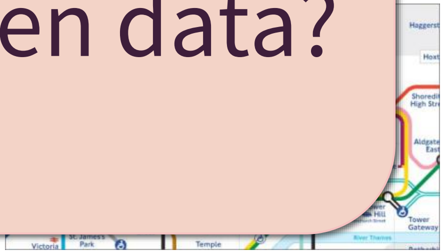


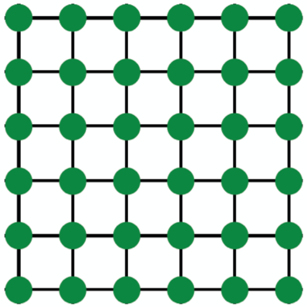
Image credit: Wikipedia

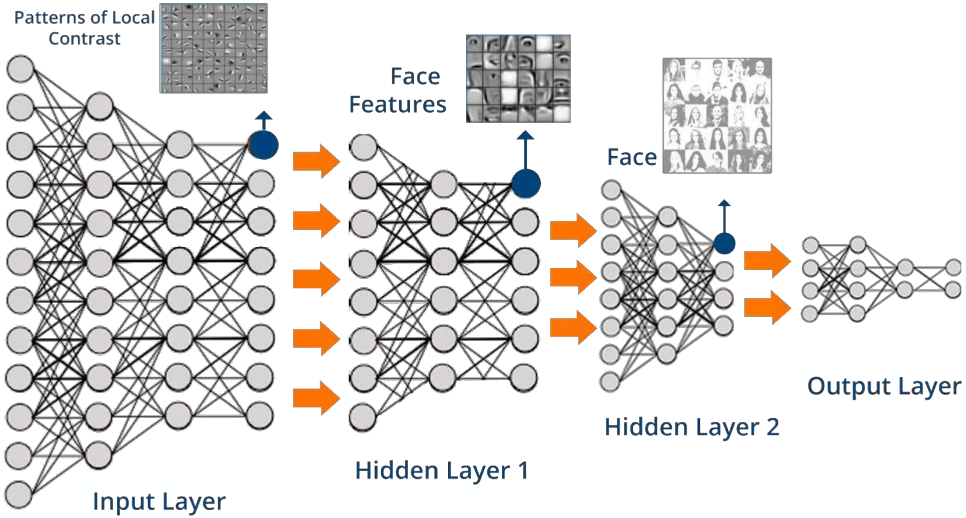
Image credit: Pinterest

Image credit: visitlondon.com

Food Webs **Particle Networks** **Underground Networks**

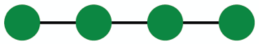
Networks are complex unlike images

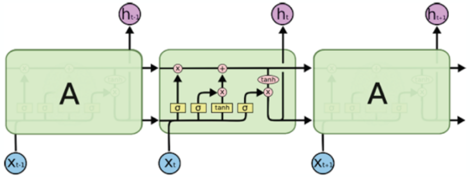




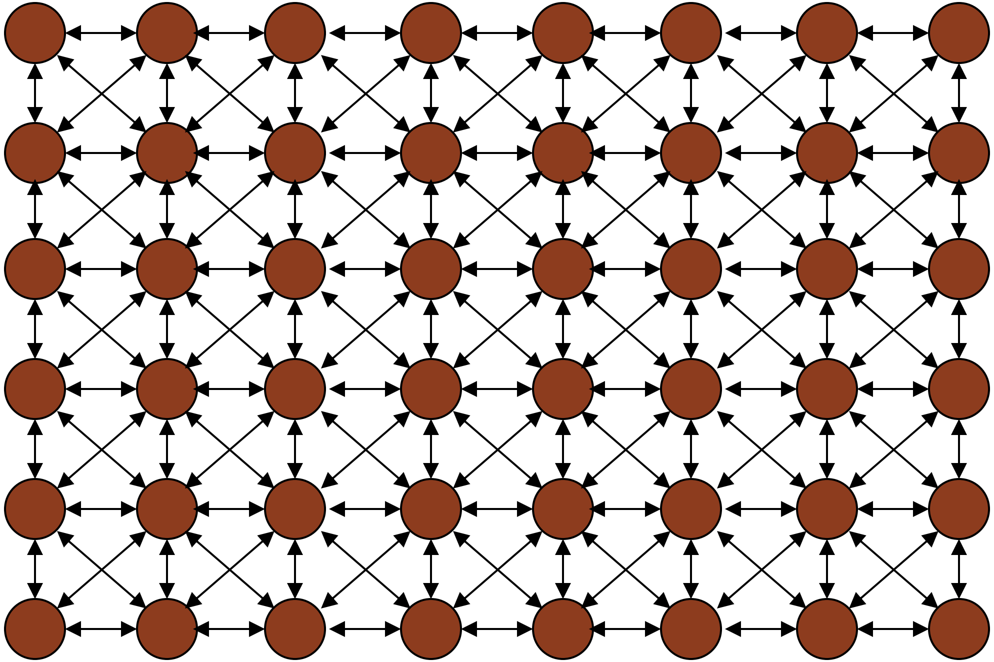
Images

Text/Speech

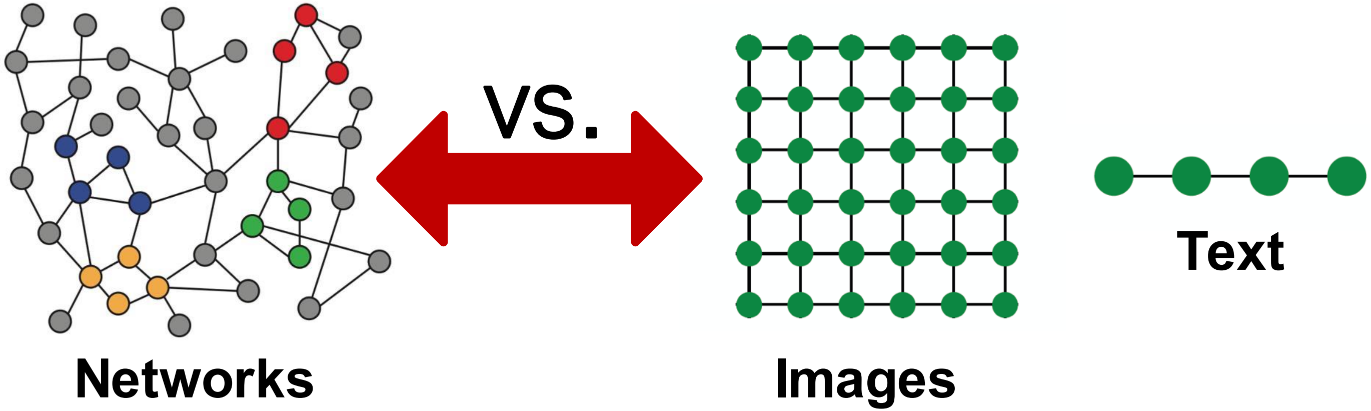




A Very Regular Graph, representing an image



Networks are complex

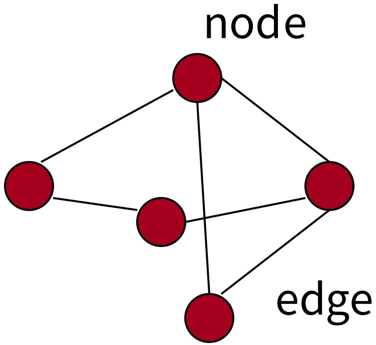


VS.

Networks

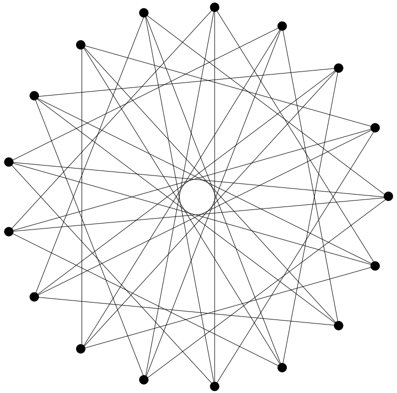
Images

Text



node

edge

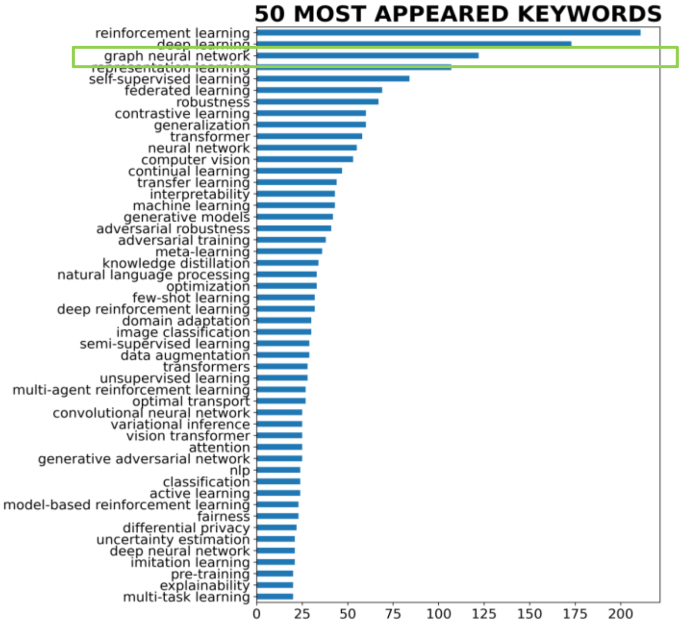


An introduction to Graph Neural Networks

Graphs are the new frontier of deep learning

Graphs connect things.

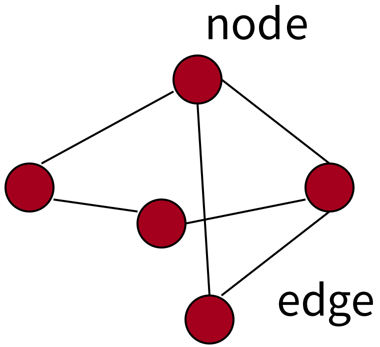
ICLR 2022 keywords

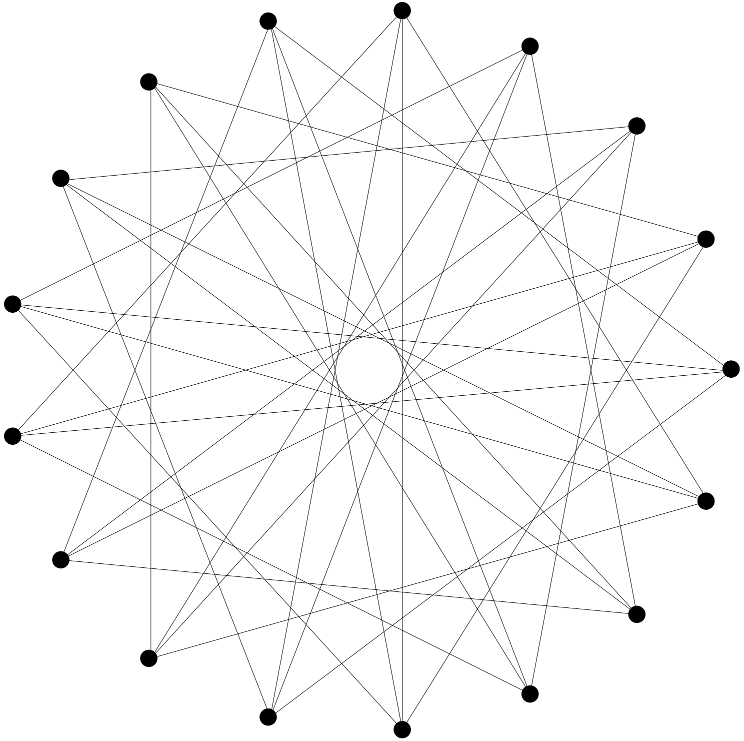


Graphs

GRAPHS ARE COMPOSED OF

- Nodes (vertices)
- Edges (arcs)
- G(V,E)





Varieties

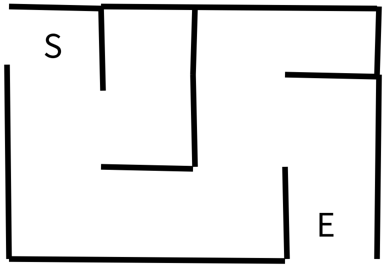
NODES

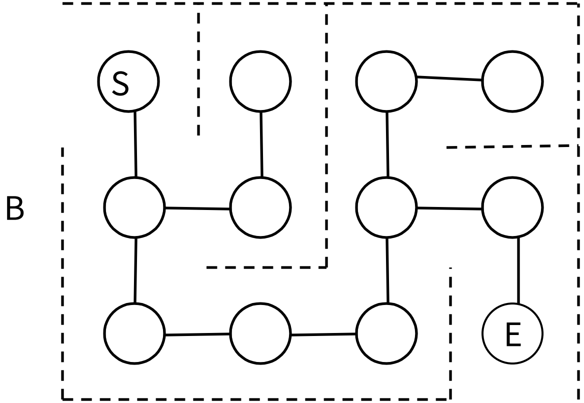
- Labeled or unlabeled

EDGES

- Directed or undirected
- Labeled or unlabeled

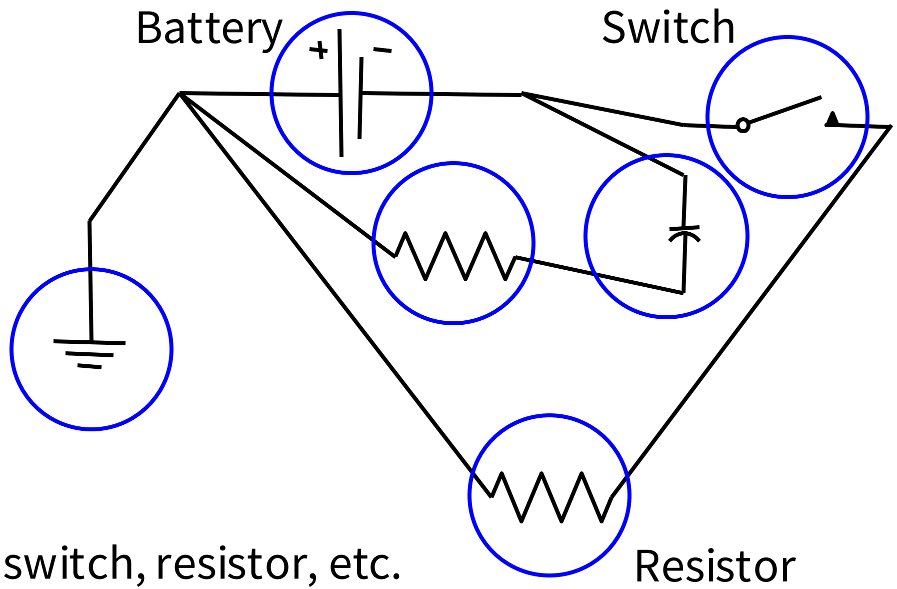
Representing a Maze





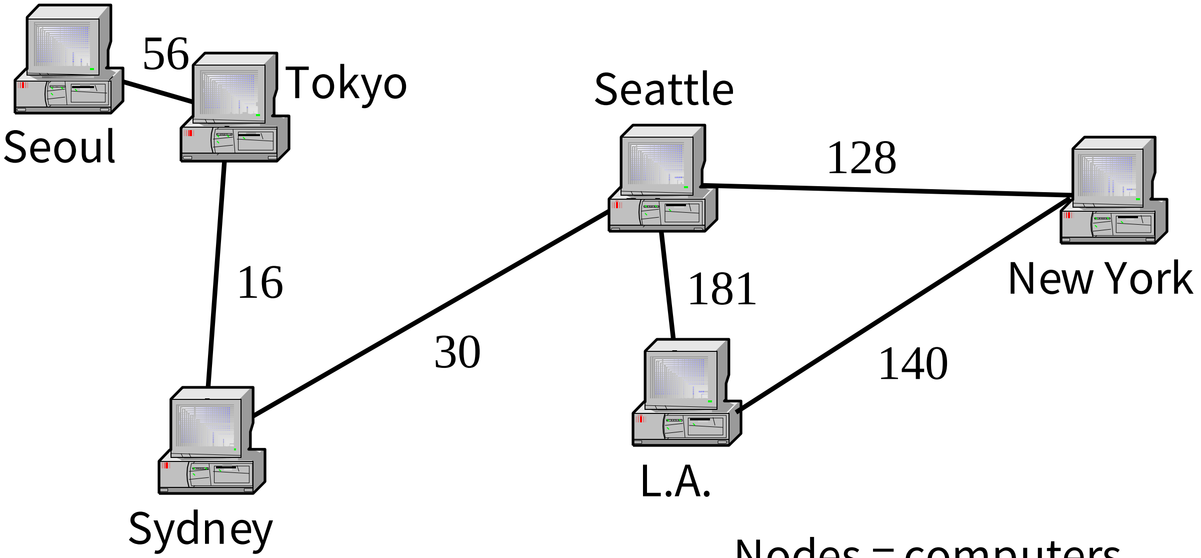
Nodes = rooms
Edge = door or passage

Representing Electrical Circuits



Nodes = battery, switch, resistor, etc.
Edges = connections

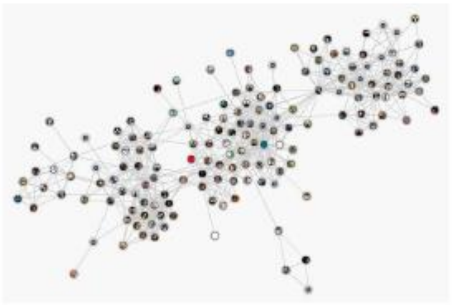
Information Transmission in a Computer Network



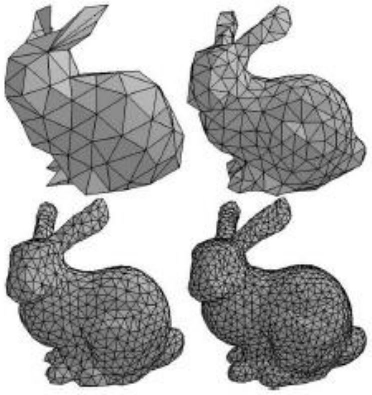
Nodes = computers
Edges = transmission rates

Non-Euclidean data structure

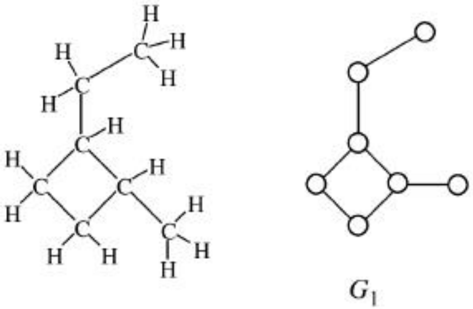
HOWEVER, there are lots of irregular data structure, …



Social Graph
(Facebook, Wikipedia)



3D Mesh

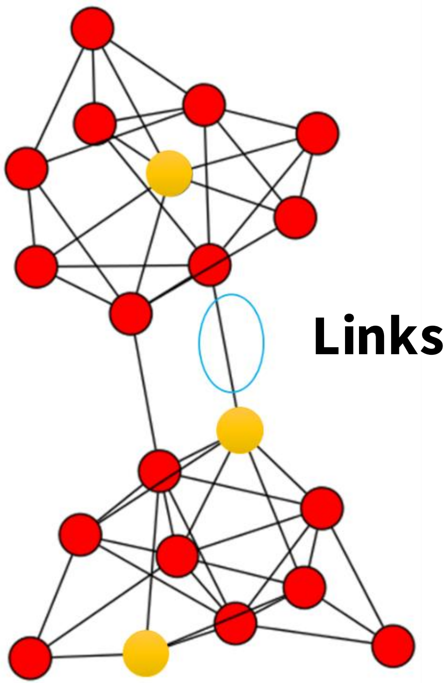


Molecular Graph

All you need is GRAPH!

Networks are not learning-friendly

G = (V, E)

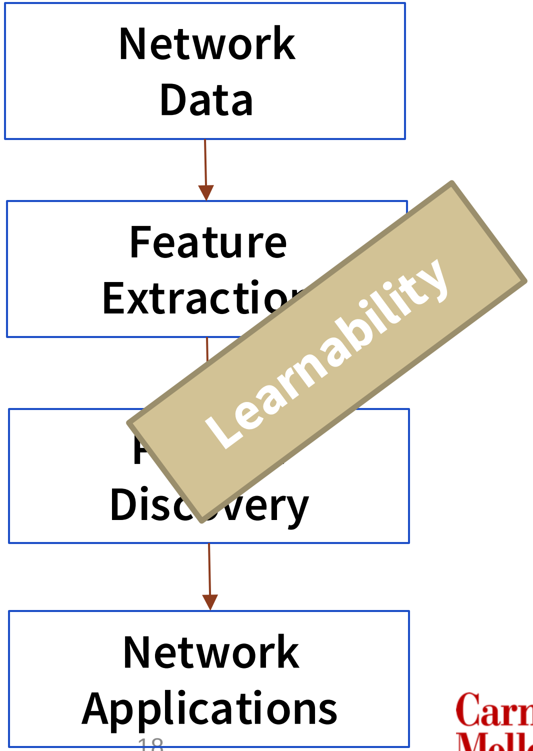


Links ⇨ Topology

⇧

Inapplicability of ML methods

Pipeline for network analysis



Network Data

Feature Extractio

F

Disc very

Network Applications

Learnability

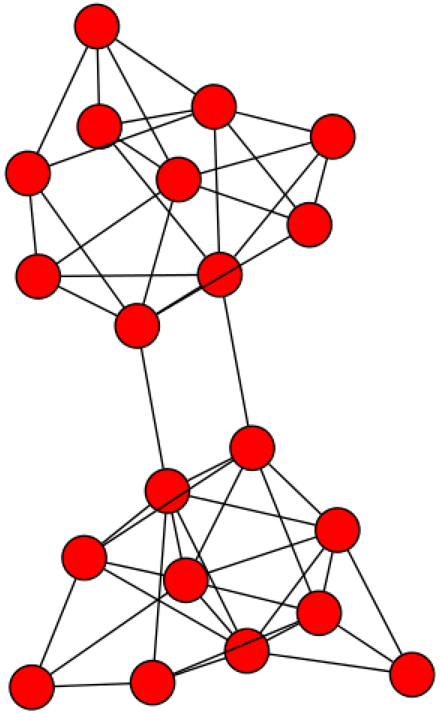
Learning from networks

Network Embedding

GNN

Network Embedding

G = (V, E)

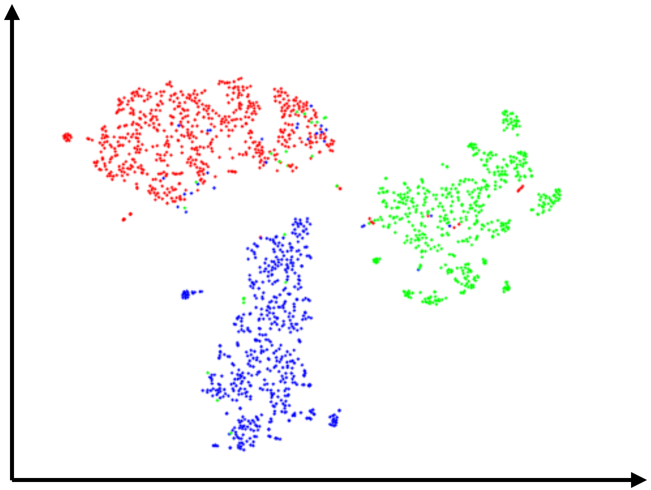


generate

embed

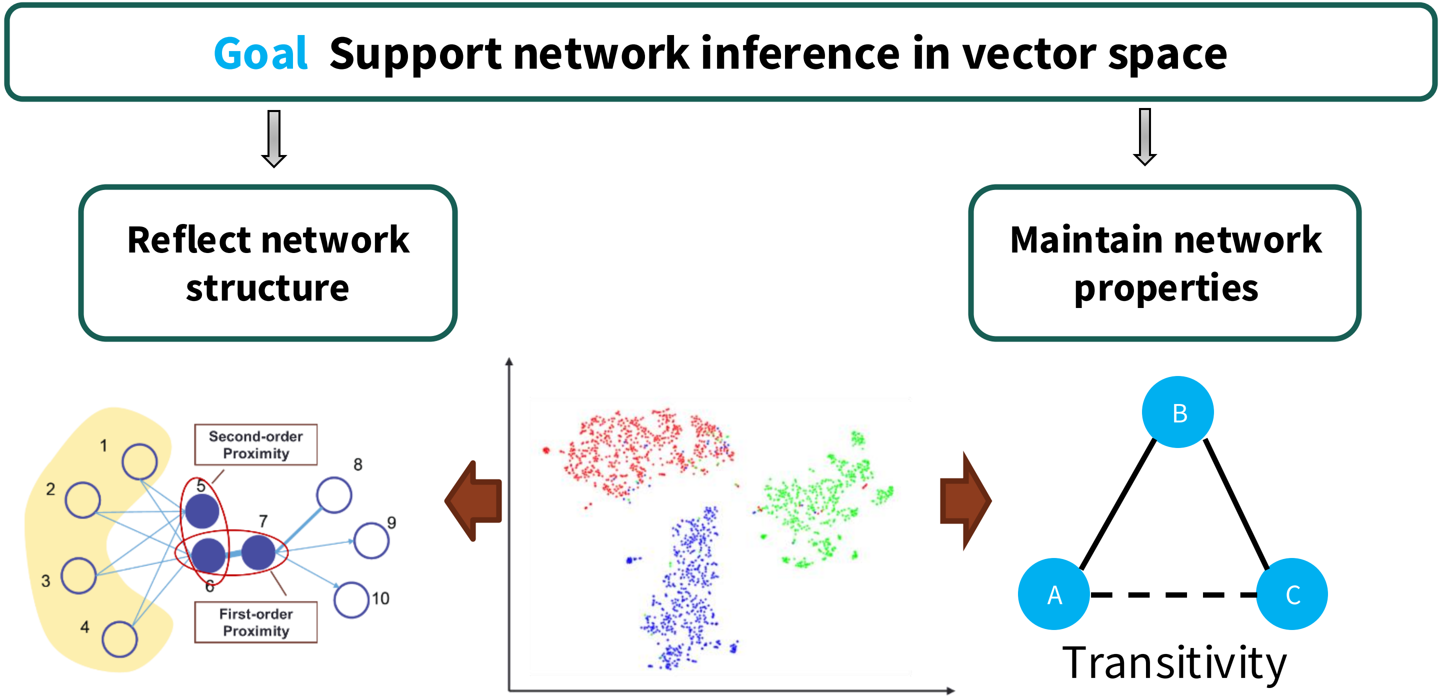
G = (V)

Vector Space



- Easy to parallel
- Can apply classical ML methods

The goal of network embedding



Goal **Support network inference in vector space**

Reflect network structure

Maintain network properties

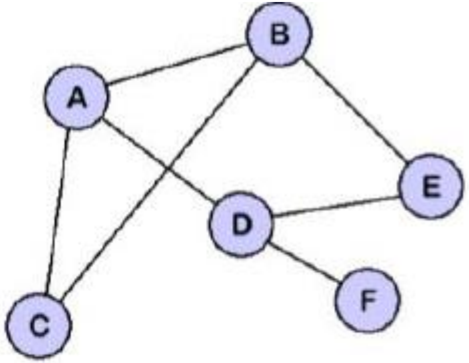
Transform network nodes into vectors that are fit for off-the-shelf machine learning models.

Graph structure

***Graph*=*G*(*X*,*A*)**

A*: *Adjacency matrix

- Edges of a graph
- Connectivity, Relationship



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

Represent relationship or interaction between elements of the system

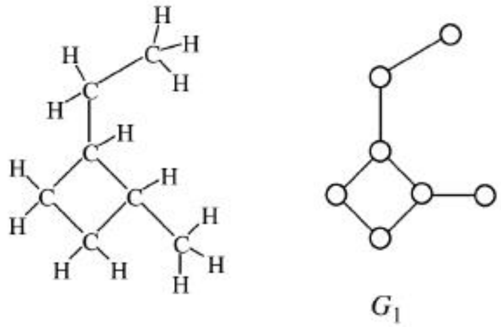
Graph structure

$\boldsymbol{Graph}=\boldsymbol{G}(\boldsymbol{X},\boldsymbol{A})$

X: Node, Vertex

- Individual person in a social network
- Atoms in a molecule

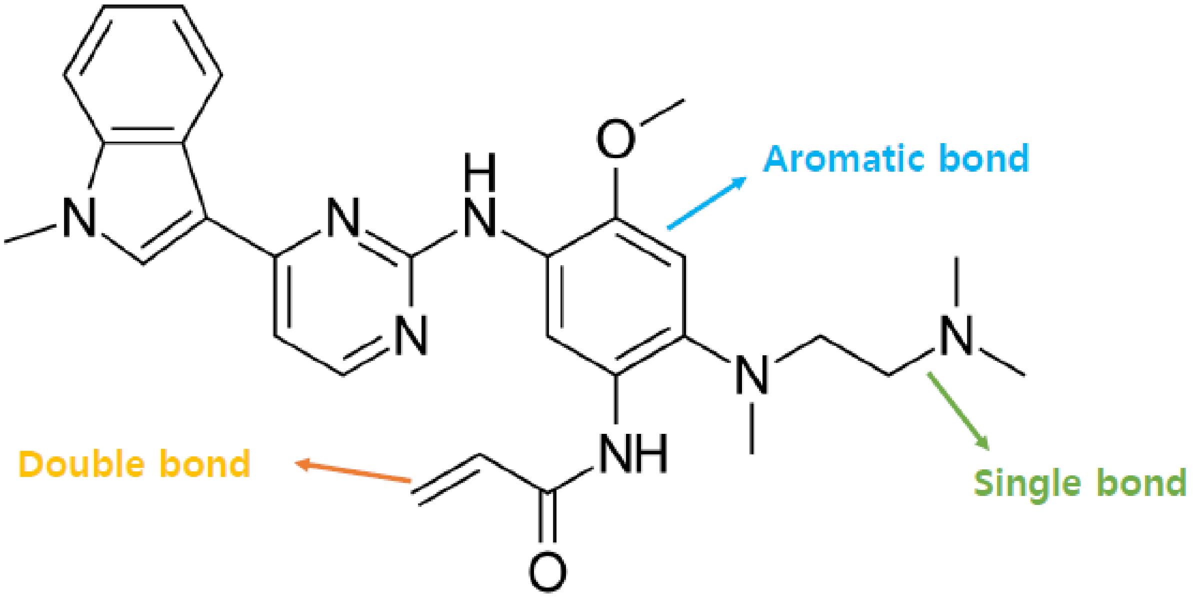
Represent elements of a system



Carnegie Mellon University

Graph structure

Edge features

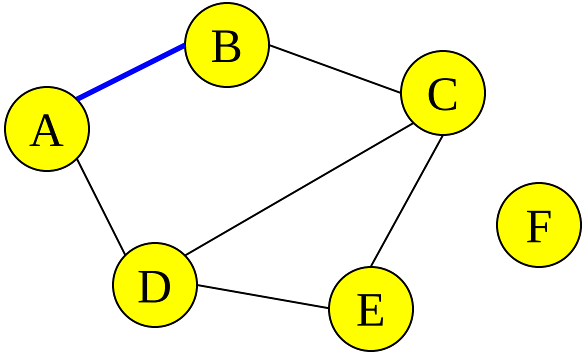


Aromatic bond

Double bond

Single bond

Adjacency Matrix



$$
\begin{array}{c|cccccc}
 & A & B & C & D & E & F \\
\hline
A & 0 & 1 & 0 & 1 & 0 & 0 \\
B & 1 & 0 & 1 & 0 & 0 & 0 \\
C & 0 & 1 & 0 & 1 & 1 & 0 \\
D & 1 & 0 & 1 & 0 & 1 & 0 \\
E & 0 & 0 & 1 & 1 & 0 & 0 \\
F & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

Graph Embedding

Graph embedding is an approach that is used to transform nodes, edges, and their features into vector space (a lower dimension) **whilst maximally preserving properties like graph structure and information.**

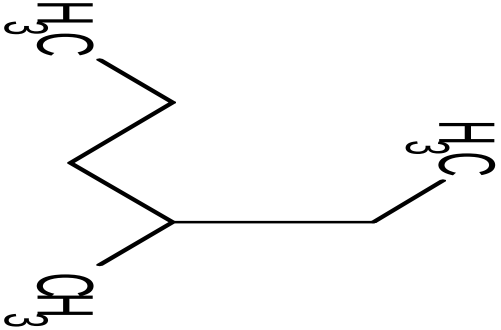
Graphs are complex because they can vary in terms of their scale, specificity, and subject

A molecule can be represented as a small, sparse, and static graph, whereas a social network could be represented by a large, dense, and dynamic graph.

An example: Molecule

A molecular structure can be interpreted as a mathematical graph where each atom is a node, and each bond is an edge.

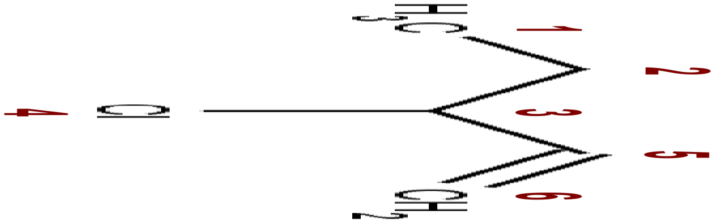
Such a representation allows for the mathematical processing of molecular structures using the graph theory



Adjacency Matrix

A molecular structure with *n* atoms may be represented by an n × n matrix (H-atoms are often omitted)

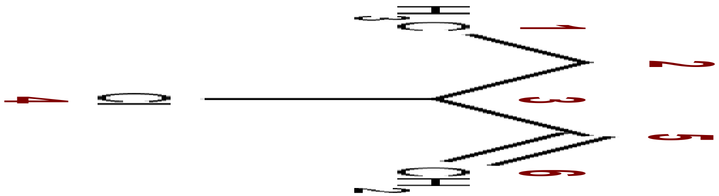
Adjacency matrix : indicates which atoms are bonded.



	1	2	3	4	5	6
1	0	1	0	0	0	0
2	1	0	1	0	0	0
3	0	1	0	1	1	0
4	0	0	1	0	0	0
5	0	0	1	0	0	1
6	0	0	0	0	1	0

Distance Matrix

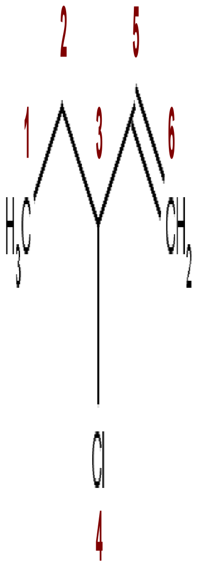
Distance matrix : encodes the distances between atoms.
The distance is defined as the number of bonds between atoms on the shortest possible path.



	1	2	3	4	5	6
1	0	1	2	3	3	4
2	1	0	1	2	2	3
3	2	1	0	1	1	2
4	3	2	1	0	2	3
5	3	2	1	2	0	1
6	4	3	2	3	1	0

Bond Matrix

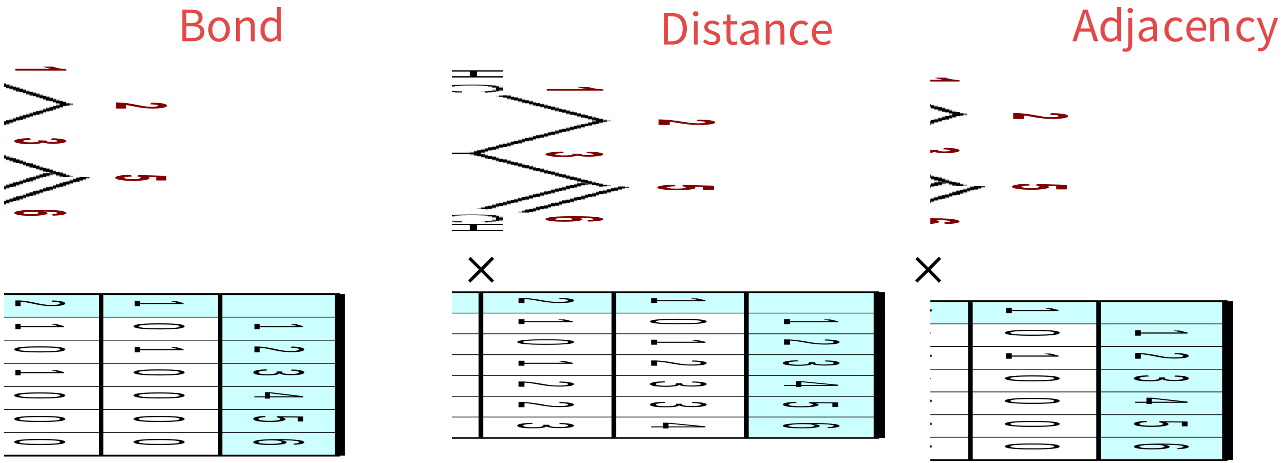
Bond matrix : indicates which atoms are bonded, and the corresponding bond orders.



	1	2	3	4	5	6
1	0	1	0	0	0	0
2	1	0	1	0	0	0
3	0	1	0	1	1	0
4	0	0	1	0	0	0
5	0	0	1	0	0	2
6	0	0	0	0	2	0

A*D*N: Topology Embedding

To create a single matrix for representation of topology, we can: A*D*N=A’

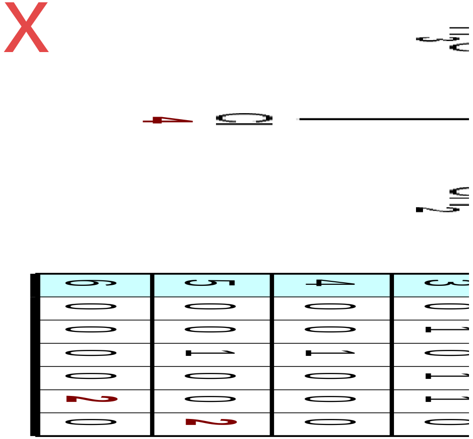


It will be an N×N matrix

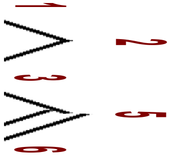
Feature Matrix

Feature Matrix for molecules:

MolWt	
ExactMolWt	
HeavyAtomCount	
HeavyAtomMolWt	
NHOHCount	
NOCount	
NumHAcceptors	
NumHDonors	
NumHeteroatoms	
NumRotatableBonds	
NumValenceElectrons	
NumAmideBonds	
Num{Aromatic,Saturated,Aliphatic}Rings	
Num{Aromatic,Saturated,Aliphatic}{Hetero,Carbo}cycles	
RingCount	
FractionCSP3	
NumSpiroAtoms	Number of spiro atoms (atoms shared between rings that share exactly one atom)
NumBridgeheadAtoms	Number of bridgehead atoms (atoms shared between rings that share at least two bonds)

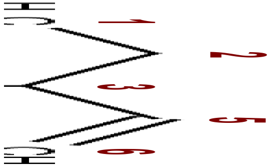


$A_{(n,n)} \times F(f\ by\ N)$



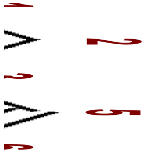
	1	2	3	4	5	6
1	0	1	0	0	0	0
2	1	0	1	0	0	0

×

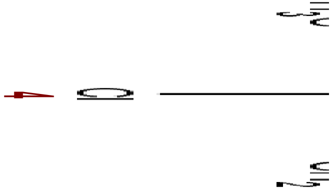


	1	2	3	4	5	6
1	0	1	2	3	3	4
2	1	0	1	2	2	3

×



	1	2	3	4	5	6
1	0	1	0	0	0	0

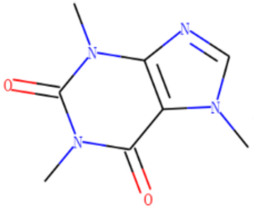


3	0	1	0	1	1	0
4	0	0	1	0	0	0
5	0	0	1	0	0	2
6	0	0	0	0	2	0

Carnegie Mellon University

Network Embedding

Caffeine Molecule



Linear Representation
SMILES String

CN1C=NC2=C1C(=O)N(C(=O)N2C)C

Molecular Fingerprint
One-Hot Encoding
Word Embedding

Graph Representation
Adjacency Matrix

0.	1.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
1.	0.	1.	0.	0.	1.	0.	0.	0.	0.	0.	0.	0.	0.
0.	1.	0.	1.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	1.	0.	1.	0.	0.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	1.	0.	1.	0.	0.	0.	0.	0.	1.	0.	0.
0.	1.	0.	0.	1.	0.	1.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	1.	0.	1.	1.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	1.	0.	0.	0.	0.	0.	0.	0.
0.	0.	0.	0.	0.	0.	1.	0.	0.	1.	0.	0.	0.	1.
0.	0.	0.	0.	0.	0.	0.	0.	1.	0.	1.	1.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.	0.	1.	0.	0.	0.	0.
0.	0.	0.	0.	1.	0.	0.	0.	0.	1.	0.	0.	1.	0.
0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	0.	1.	0.	0.
0.	0.	0.	0.	0.	0.	0.	0.	1.	0.	0.	0.	0.	0.

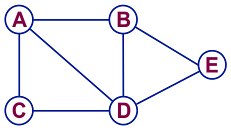
Node Attribute Matrix

Edge Attribute Matrix

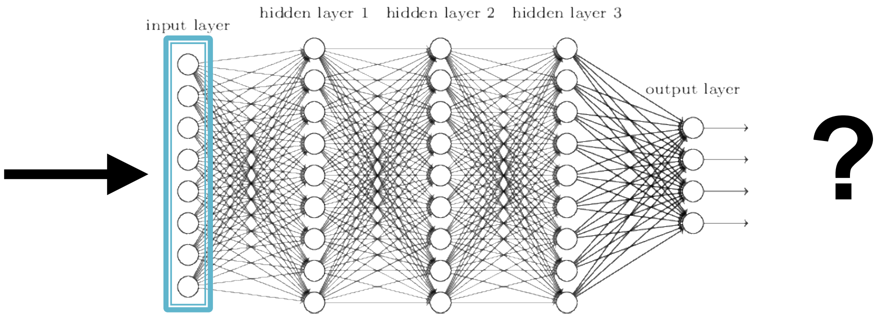
Graph Embedding

Learning for Networks vs. Learning via Graphs

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

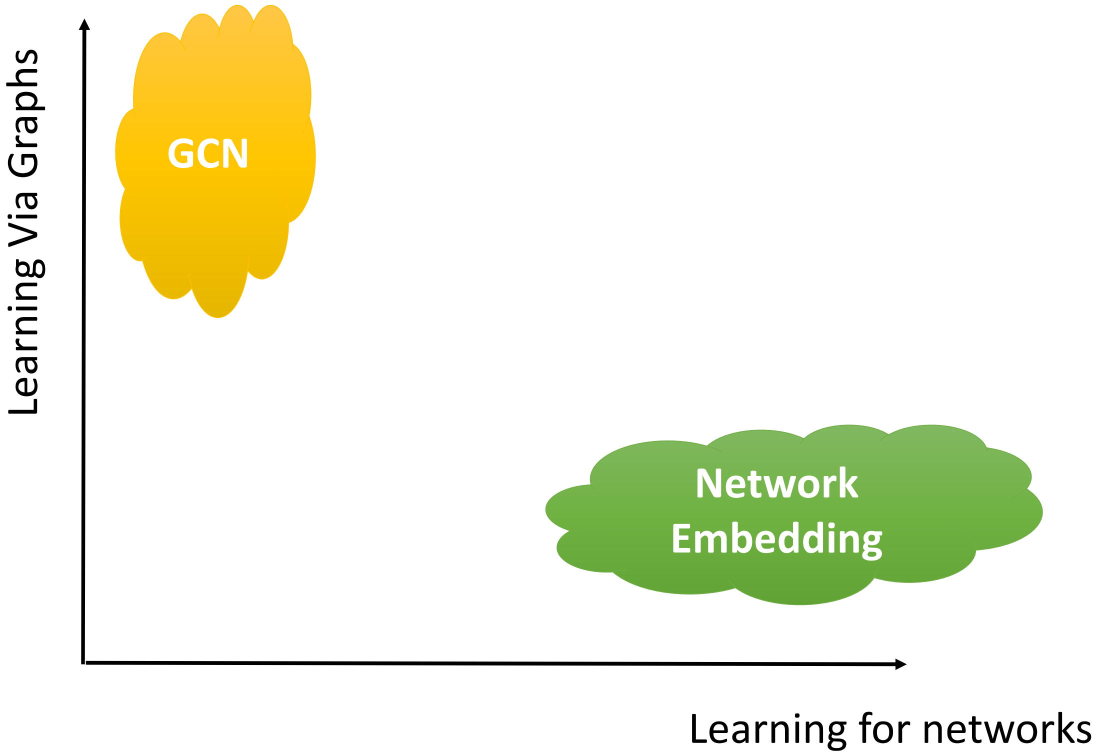


$$\begin{array}{c|ccccc|cc} & A & B & C & D & E & \multicolumn{2}{c}{Feat} \\ A & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ B & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ C & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ D & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ E & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{array}$$



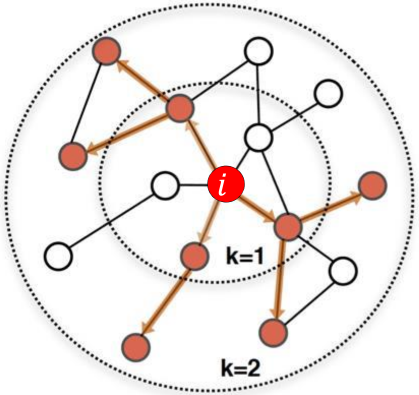
- Issues with this idea:
  - $O(|V|)$ parameters
  - Not applicable to graphs of different sizes
  - Sensitive to node ordering

Learning for Networks vs. Learning via Graphs

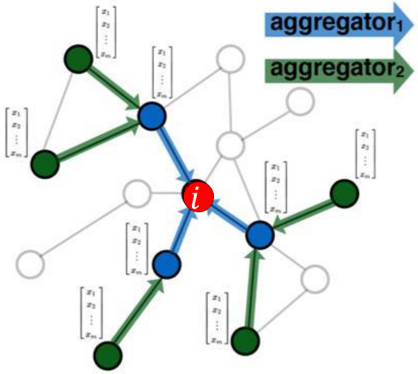


How can we learn via graph neural nets?

Idea: Node’s neighborhood defines a computation graph



Determine node computation graph

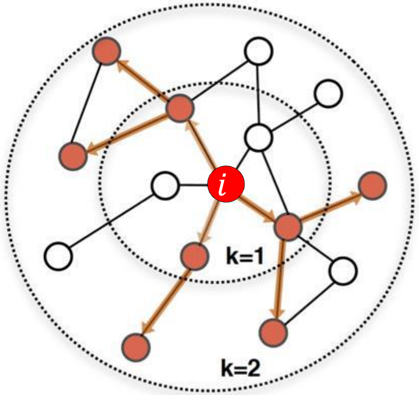


Propagate and transform information

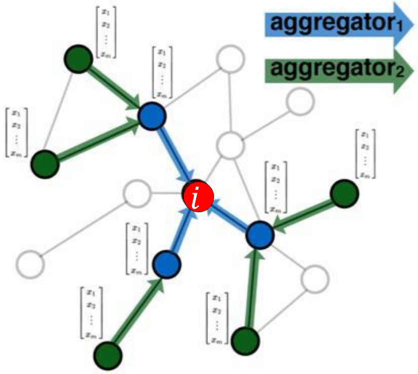
Learn how to propagate information across the graph to compute node features

How can we learn via graph neural nets?

Idea: Node’s neighborhood defines a computation graph



Determine node computation graph

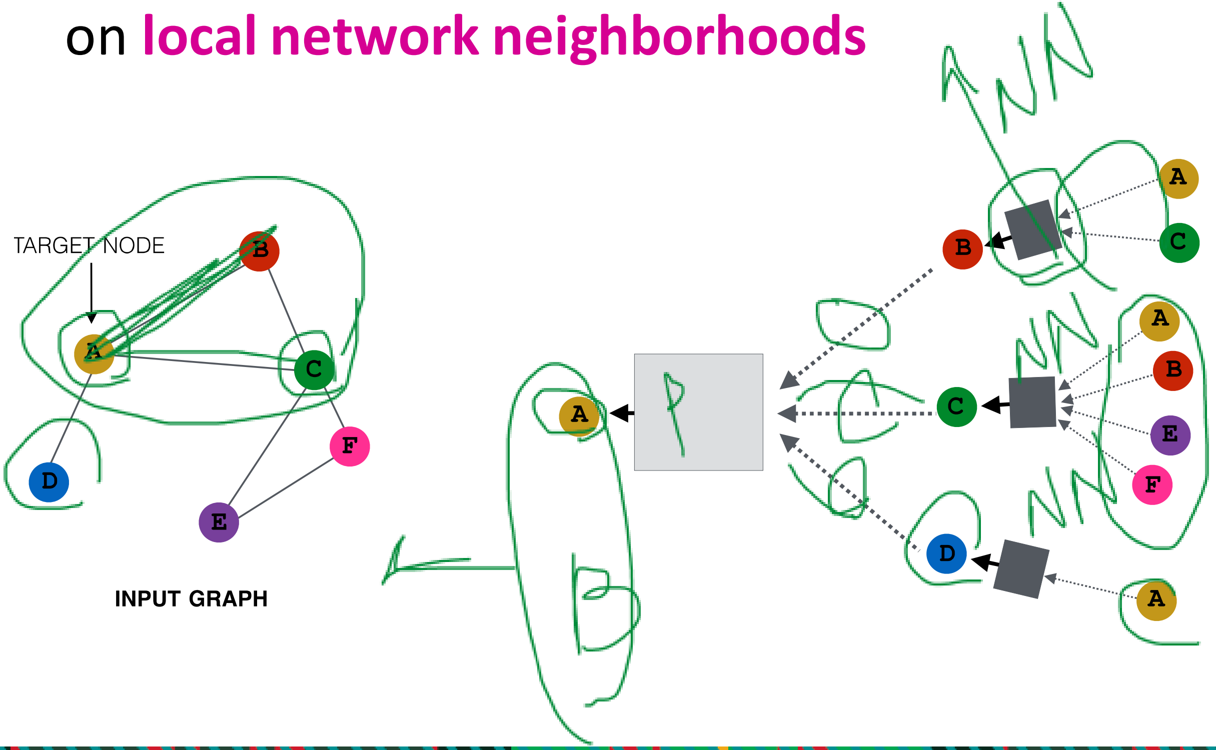


Propagate and transform information

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

How can we learn via graph neural nets?

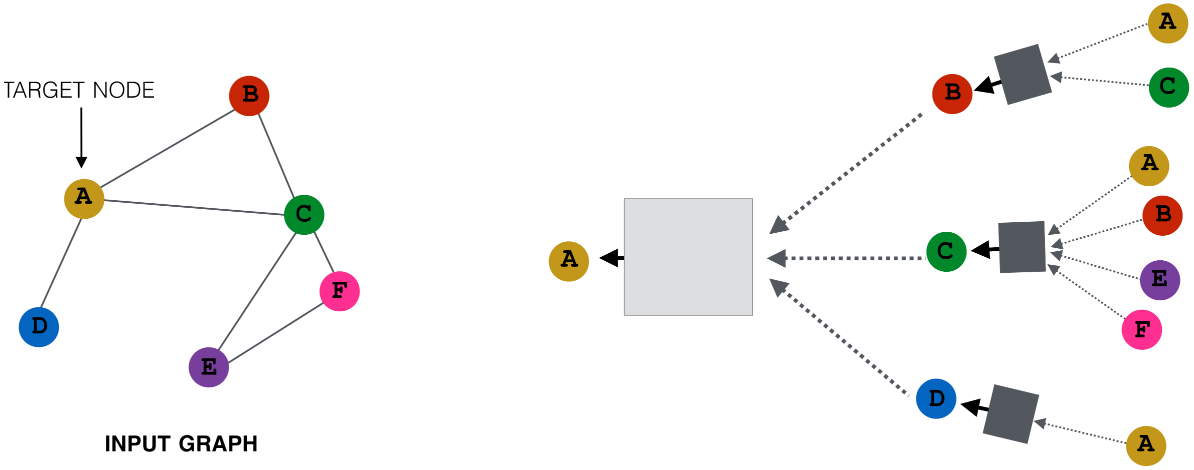
- **Key idea:** Generate node embeddings based on **local network neighborhoods**



Carnegie Mellon University

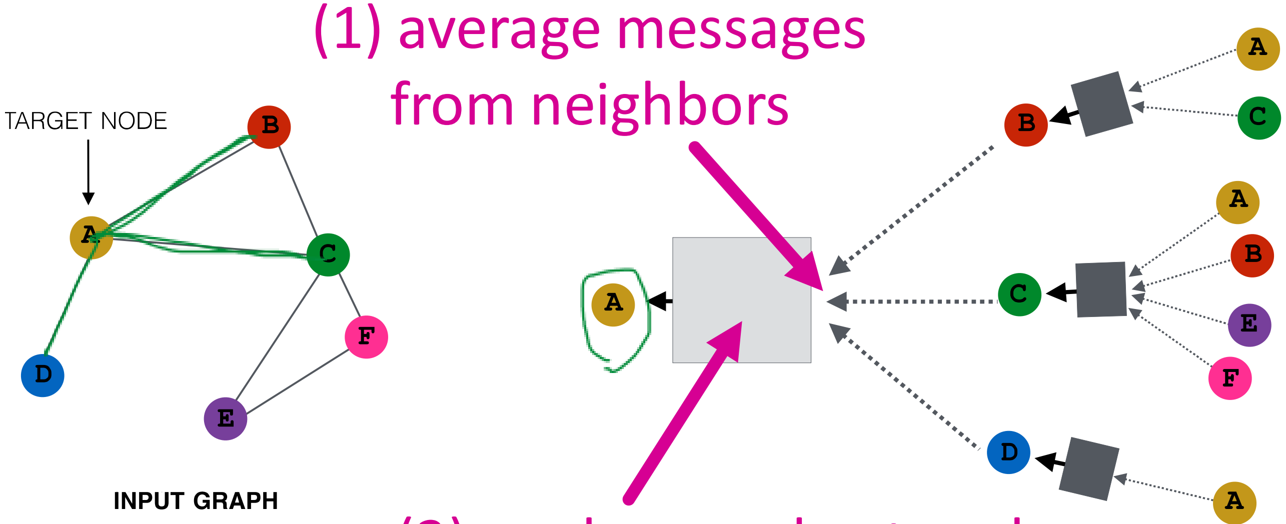
How can we learn via graph neural nets?

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



How can we learn via graph neural nets?

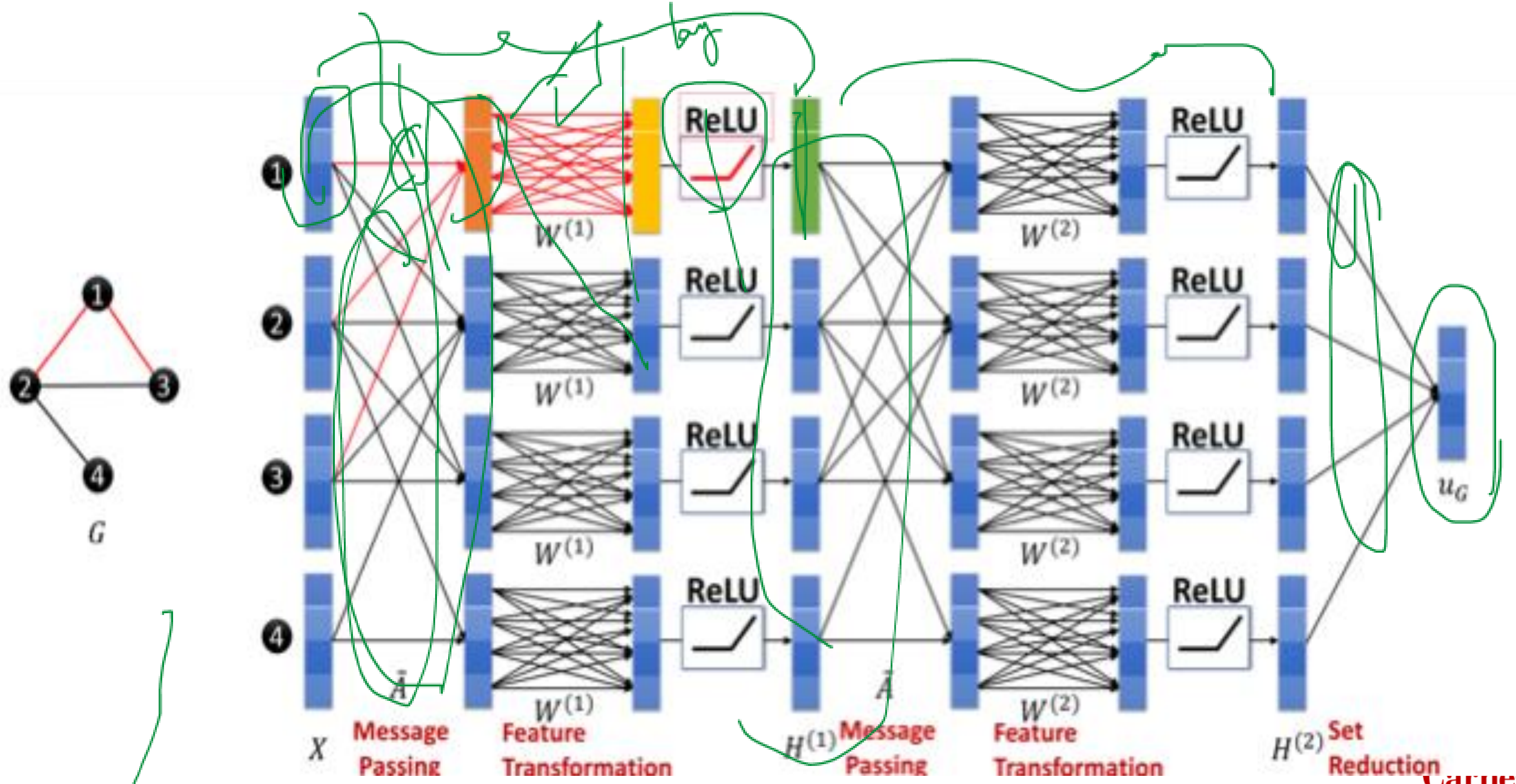
- **Basic approach:** Average information from neighbors and apply a neural network



(1) average messages from neighbors

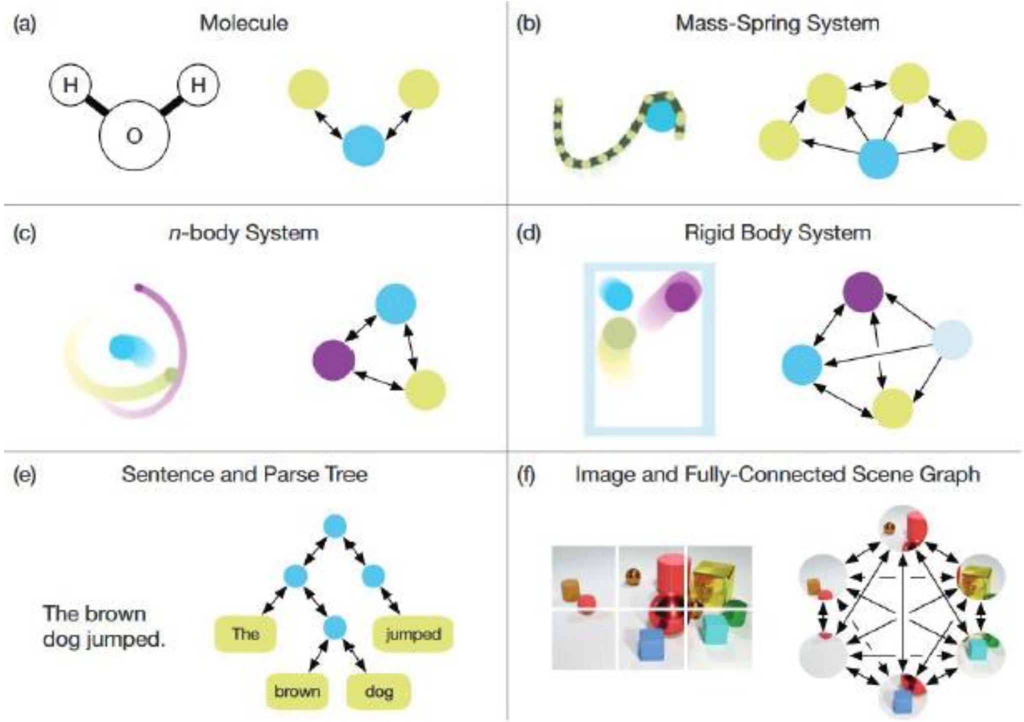
(2) apply neural network

How can we learn via graph neural nets?



Learning relation and interaction

What can we do with graph neural networks?



Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." arXivpreprint arXiv:1806.01261(2018).

Learning relation and interaction

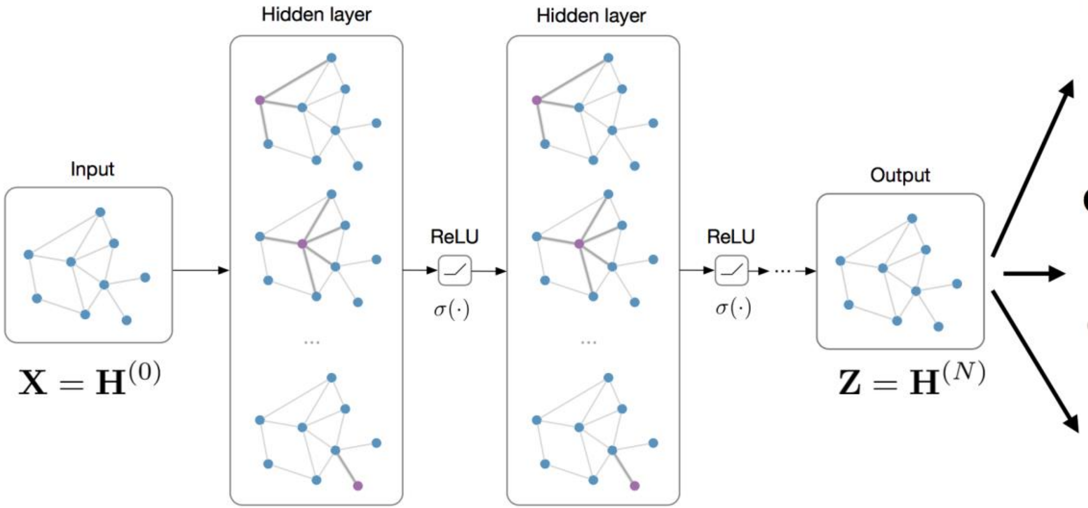
What can we do with graph neural networks?

- Node classification
- Link prediction
- Node2Vec, Subgraph2Vec, Graph2Vec: Embedding node/substructure/graph structure to a vector
- Learning physics law from data
- And you can do more amazing things with GNN!

Graph neural networks

Graph neural networks

Input: Feature matrix $\mathbf{X} \in \mathbb{R}^{N \times E}$, preprocessed adjacency matrix $\hat{\mathbf{A}}$



$\mathbf{X} = \mathbf{H}^{(0)}$

$\mathbf{Z} = \mathbf{H}^{(N)}$

$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

Node classification:

$$\mathrm{softmax}(\mathbf{z_n})$$

e.g. Kipf & Welling (ICLR 2017)

Graph classification:

$$\mathrm{softmax}(\textstyle\sum_n \mathbf{z_n})$$

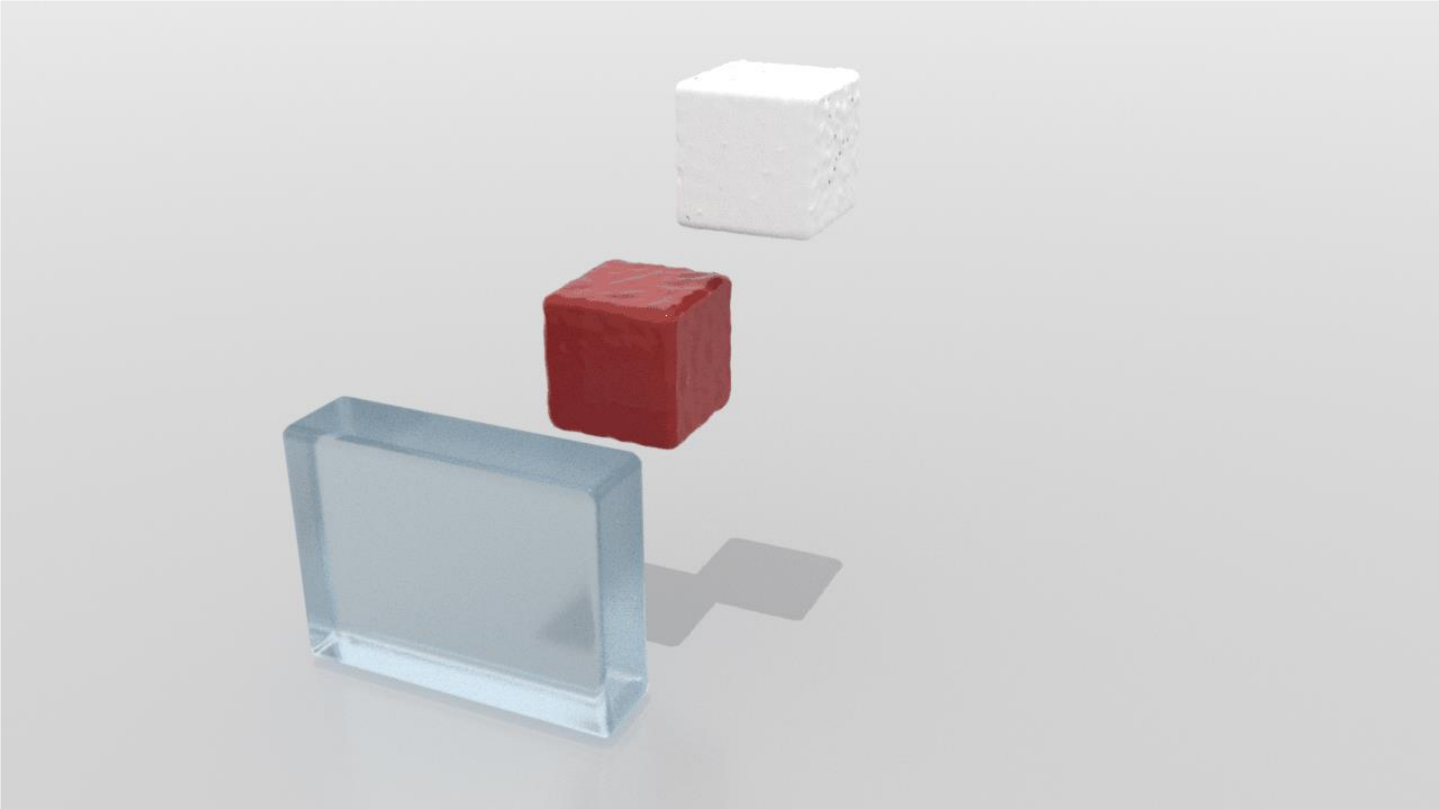
e.g. Duvenaud et al. (NIPS 2015)

Link prediction:

$$p(A_{ij}) = \sigma(\mathbf{z_i^T z_j})$$

Kipf & Welling (NIPS BDL 2016)
"Graph Auto-Encoders"

* slide from Thomas Kipf, **University of Amsterdam**



- Overall architecture of graph neural networks
- Updating node states

  -Graph Convolutional Network (GCN)

  -Graph Attention Network (GAT)

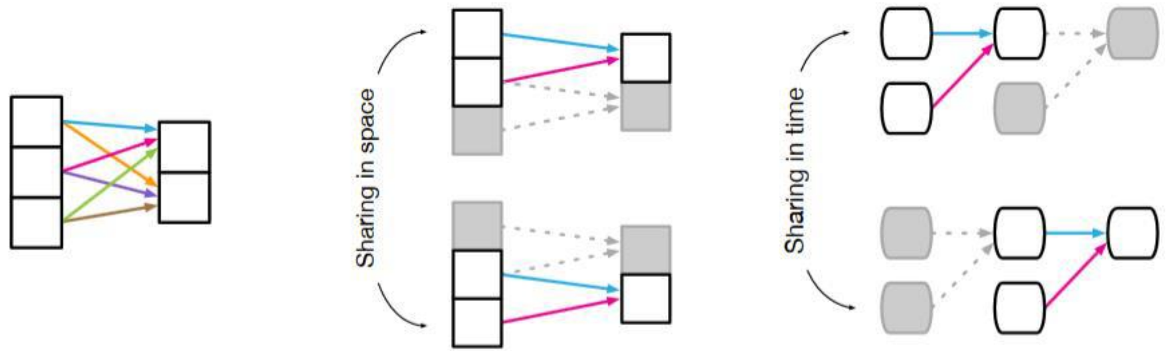
  -Gated Graph Neural Network (GGNN)

- Readout : permutation invariance on changing node orders
- Graph Auto-Encoders
- Practical issues

  -Skip connection-Inception-Dropout

Principles of graph neural network

- Weights using in updating hidden states of fully-connected Net, CNN and RNN



(a) Fully connected

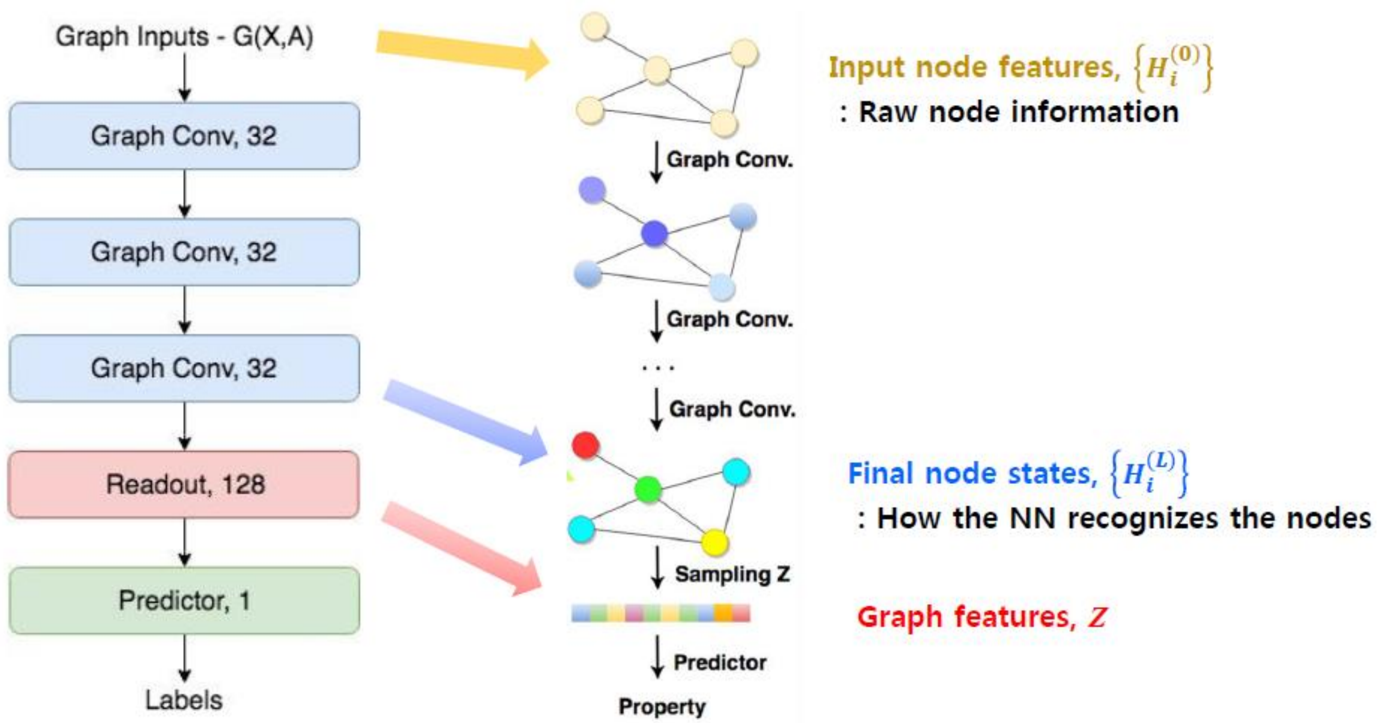
(b) Convolutional

(c) Recurrent

Figure 1: Reuse and sharing in common deep learning building blocks. (a) Fully connected layer, in which all weights are independent, and there is no sharing. (b) Convolutional layer, in which a local kernel function is reused multiple times across the input. Shared weights are indicated by arrows with the same color. (c) Recurrent layer, in which the same function is reused across different processing steps.

Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." arXivpreprint arXiv:1806.01261(2018).

Overall neural network structure–case of supervised learning



Graph Inputs - G(X,A)

Graph Conv, 32

Graph Conv, 32

Graph Conv, 32

Readout, 128

Predictor, 1

Labels

Graph Conv.

Graph Conv.

. . .

Graph Conv.

Sampling Z

Predictor

Property

Input node features, $\left\{H_i^{(0)}\right\}$

: Raw node information

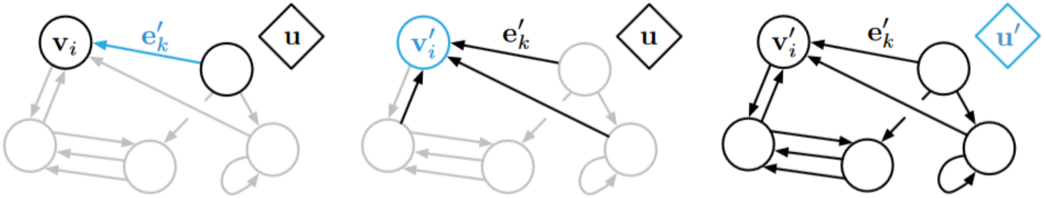
Final node states, $\left\{H_i^{(L)}\right\}$

: How the NN recognizes the nodes

Graph features, Z

Principles of graph neural network

Updates in a graph neural network



(a) Edge update (b) Node update (c) Global update

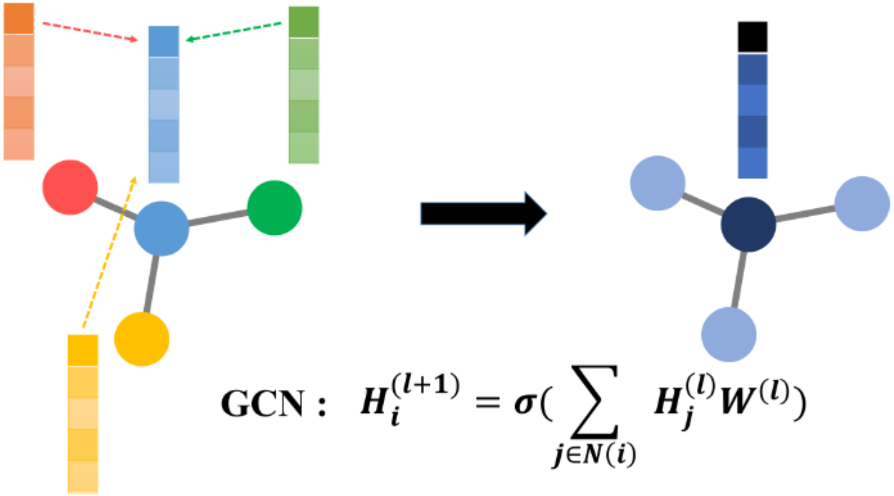
Figure 3: Updates in a GN block. Blue indicates the element that is being updated, and black indicates other elements which are involved in the update (note that the pre-update value of the blue element is also used in the update). See Equation 1 for details on the notation.

- **Edge update** : relationship or interactions, sometimes called as 'message passing'ex) the forces of spring
- **Node update** : aggregates the edge updates and used in the node updateex) the forces acting on the ball
- **Global update** : an update for the global attributeex) the net forces and total energy of the physical system

Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." arXivpreprint arXiv:1806.01261(2018).

Principles of graph neural network

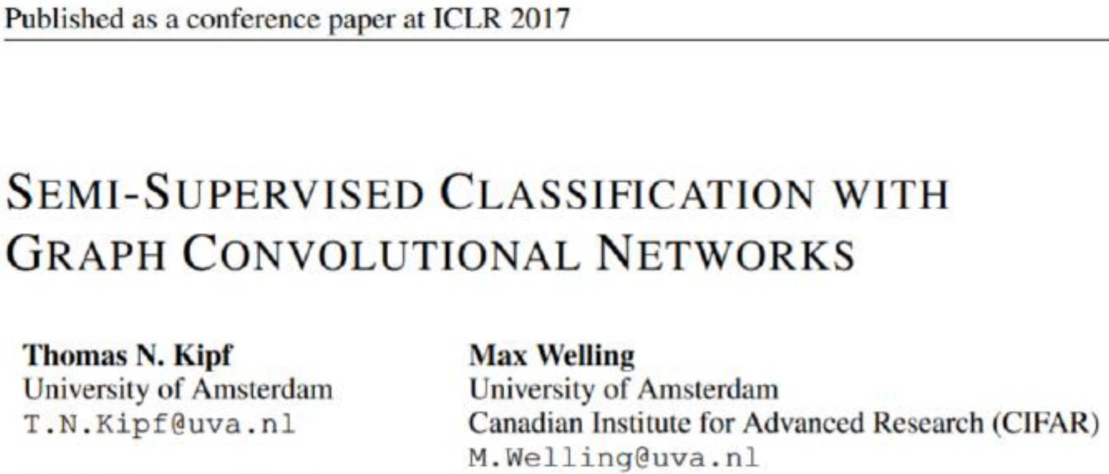
Weights using in updating hidden states of GNN



$$\mathbf{GCN:}\quad \boldsymbol{H}_i^{(l+1)} = \boldsymbol{\sigma}\Big(\sum_{j\in N(i)} \boldsymbol{H}_j^{(l)} \boldsymbol{W}^{(l)}\Big)$$

Sharing weights for all nodes in graph, but nodes are differently updated by reflecting individual node features $H_j^{(l)}$

GCN : Graph Convolutional Network



Published as a conference paper at ICLR 2017

SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

Thomas N. Kipf
University of Amsterdam
T.N.Kipf@uva.nl

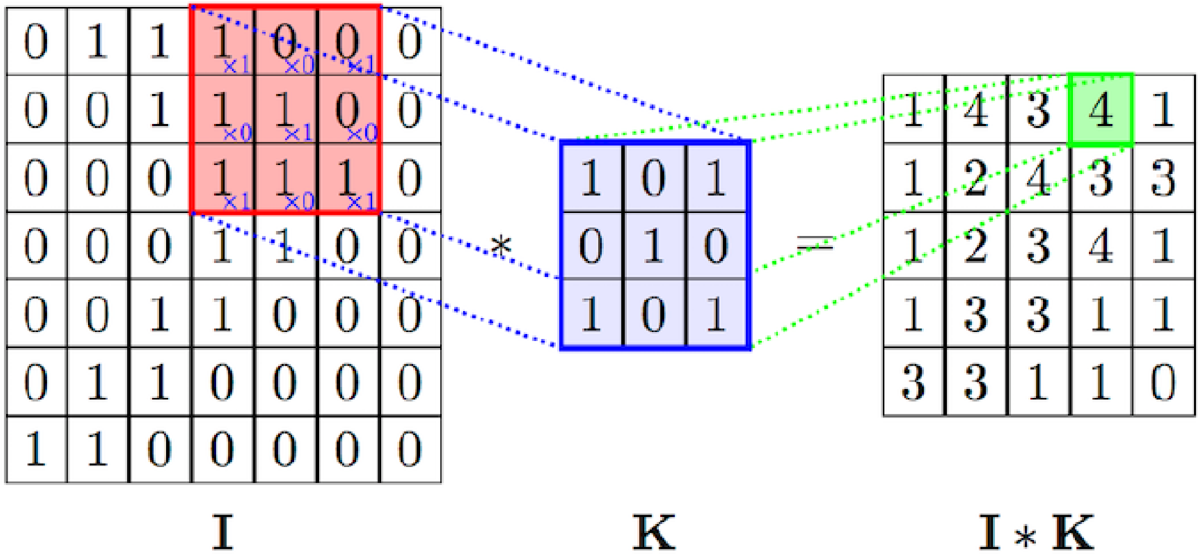
Max Welling
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
M.Welling@uva.nl

http://tkipf.github.io/

Famous for variational autoencoder (VAE)

Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." arXivpreprint arXiv:1609.02907 (2016)

GCN : Graph Convolutional Network

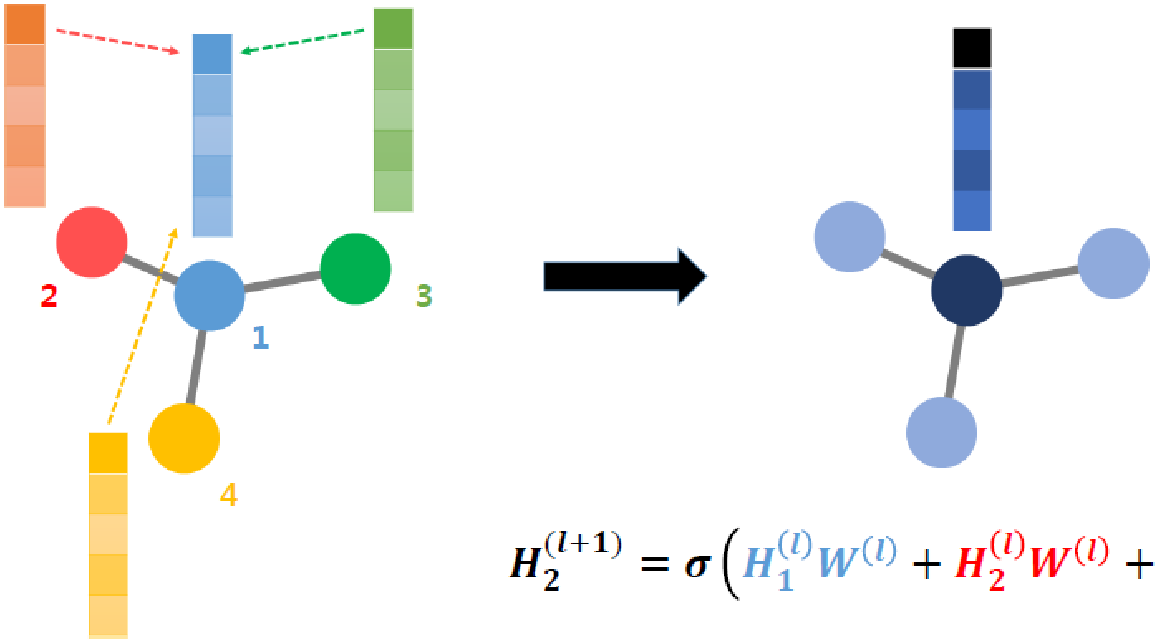


$$
\begin{bmatrix} 0&1&1&1&0&0&0\\ 0&0&1&1&1&0&0\\ 0&0&0&1&1&1&0\\ 0&0&0&1&1&0&0\\ 0&0&1&1&0&0&0\\ 0&1&1&0&0&0&0\\ 1&1&0&0&0&0&0 \end{bmatrix} * \begin{bmatrix} 1&0&1\\ 0&1&0\\ 1&0&1 \end{bmatrix} = \begin{bmatrix} 1&4&3&4&1\\ 1&2&4&3&3\\ 1&2&3&4&1\\ 1&3&3&1&1\\ 3&3&1&1&0 \end{bmatrix}
$$

What NN learns

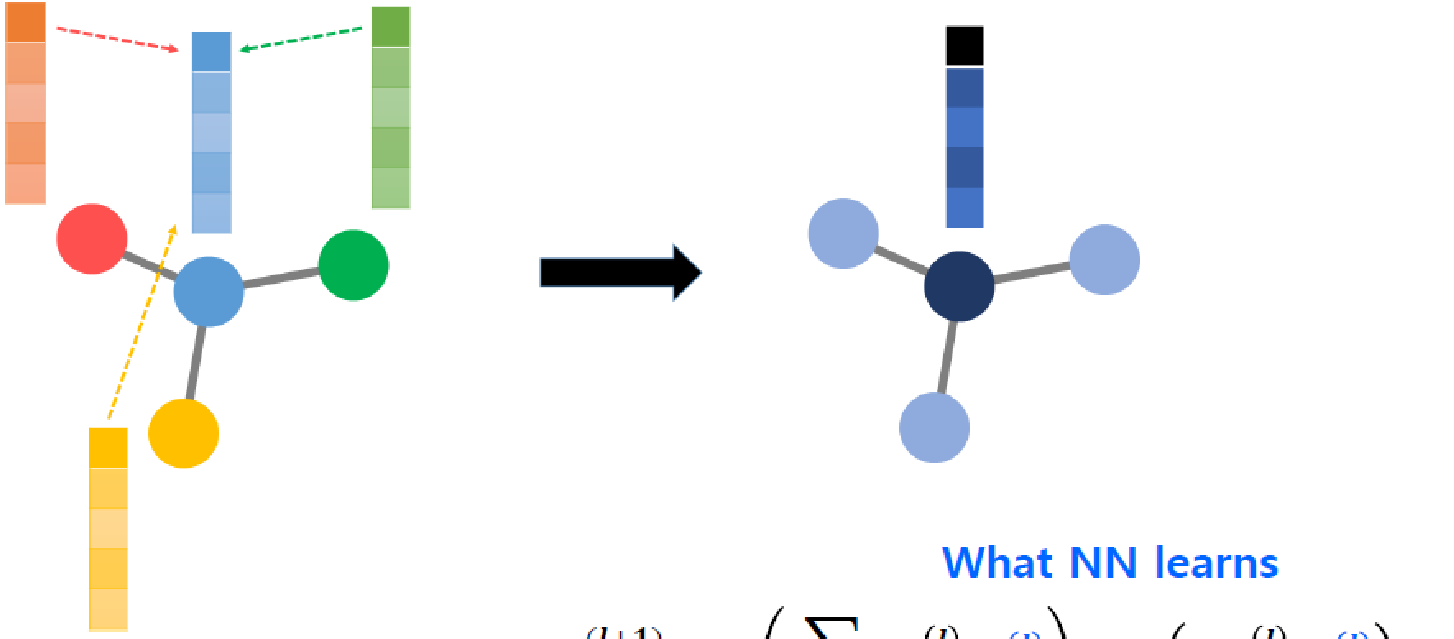
$$X_i^{(l+1)} = \sigma(\sum_{j\in[i-k,i+k]} W_j^{(l)} X_j^{(l)} + b_j^{(l)})$$

GCN : Graph Convolutional Network



$$H_2^{(l+1)} = \sigma\left(H_1^{(l)}W^{(l)} + H_2^{(l)}W^{(l)} + H_3^{(l)}W^{(l)} + H_4^{(l)}W^{(l)}\right)$$

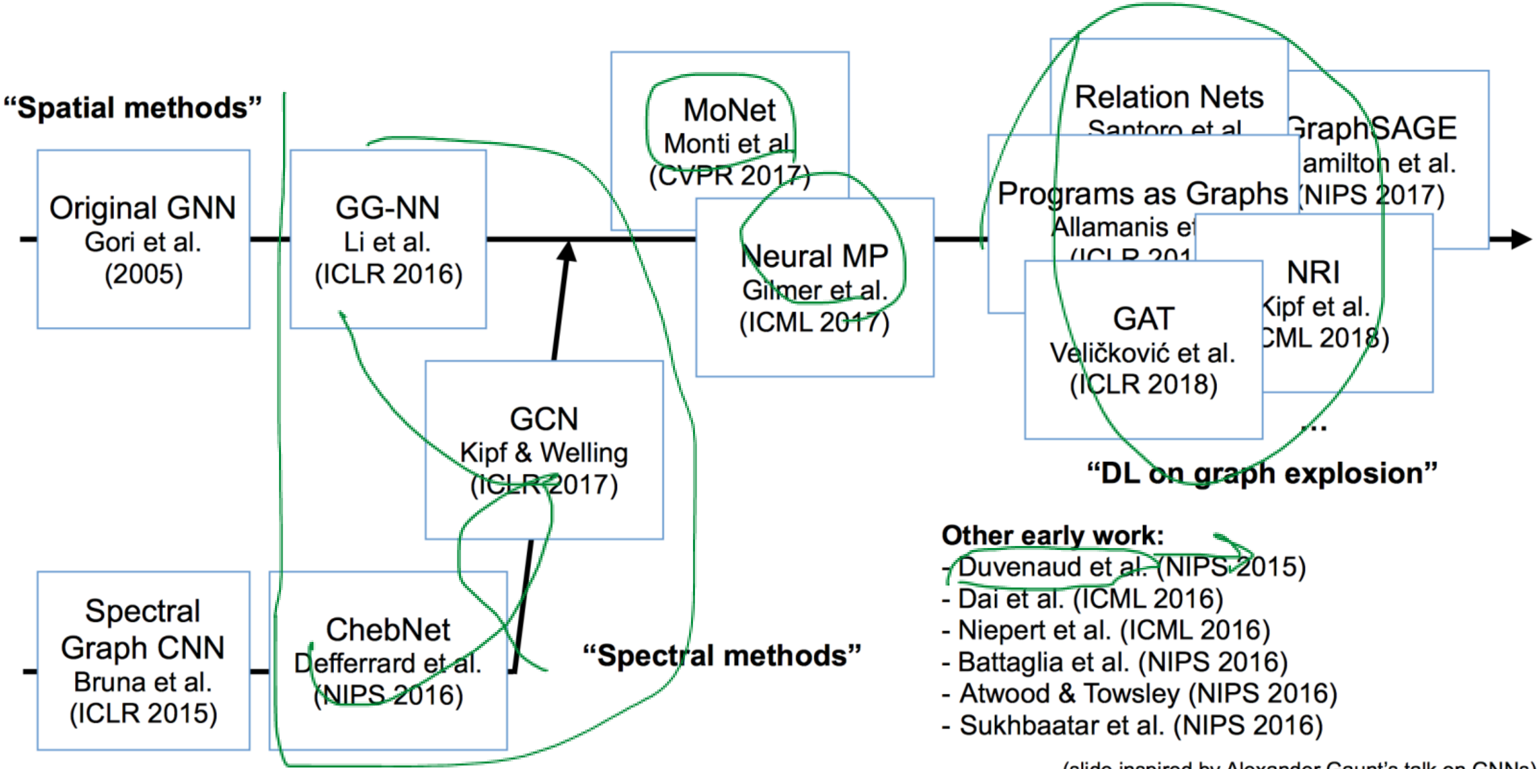
GCN : Graph Convolutional Network



What NN learns

$$H_i^{(l+1)} = \sigma\left(\sum_{j\in N(i)} H_j^{(l)} W^{(l)}\right) = \sigma\left(AH_j^{(l)} W^{(l)}\right)$$

A **Brief History** of Graph Neural Nets

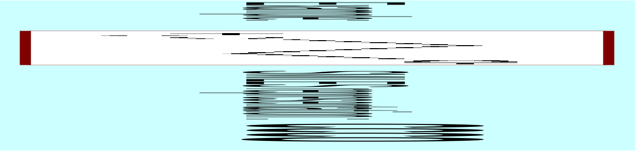


Other early work:

- Duvenaud et al. (NIPS 2015)
- Dai et al. (ICML 2016)
- Niepert et al. (ICML 2016)
- Battaglia et al. (NIPS 2016)
- Atwood & Towsley (NIPS 2016)
- Sukhbaatar et al. (NIPS 2016)

(slide inspired by Alexander Gaunt's talk on GNNs)

SMILES



1.Atoms are represented by their atomic symbols.
2.Hydrogen atoms are omitted (are implicit).
3.Neighboring atoms are represented next to each other.
4.Double bonds are represented by ‘=‘, triple bonds by ‘#’.
5.Branches are represented by parentheses.
6.Rings are represented by allocating digits to the two connecting ring atoms.