Graph Neural Networks

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

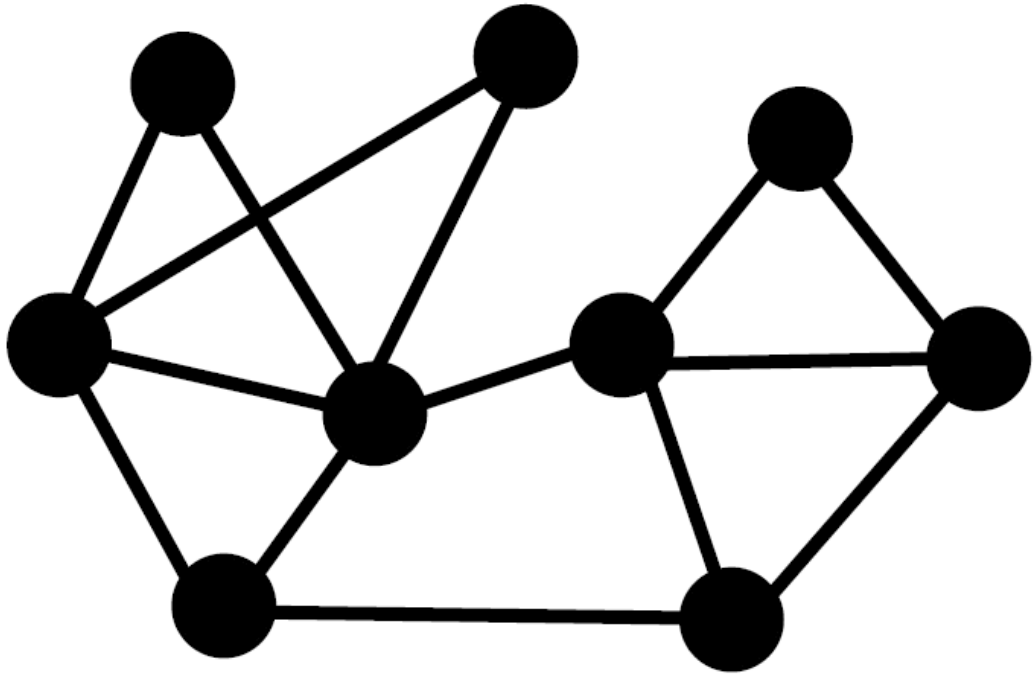
جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

KAUST Academy
King Abdullah University of Science and Technology

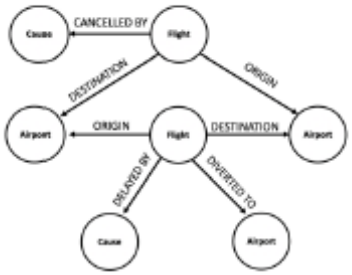
February 1, 2023

Why Graphs?

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



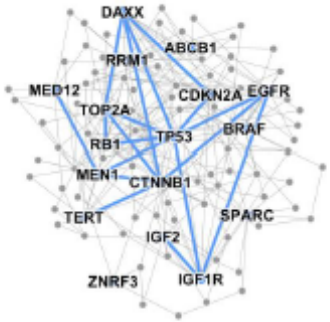
Why Graphs? (cont.)



Event Graphs

Image credit: SalientNetworks

Computer Networks



Disease Pathways

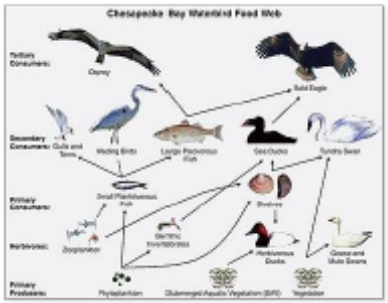


Image credit: Wikipedia

Food Webs

Image credit: Pinterest

Particle Networks

Image credit: visitlondon.com

Underground Networks

Why Graphs? (cont.)

Image credit: Medium

Social Networks

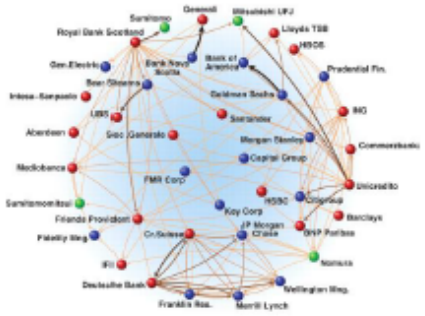


Image credit: Science

Economic Networks

Image credit: Lumen Learning

Communication Network

Citation Networks

Image credit: Missoula Current News

Internet

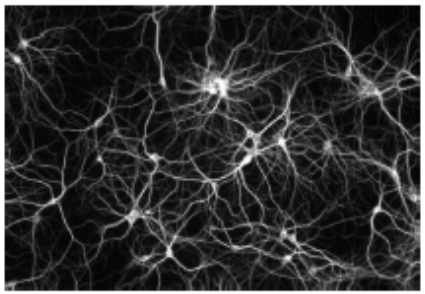


Image credit: The Conversation

Networks of Neurons

Why Graphs? (cont.)

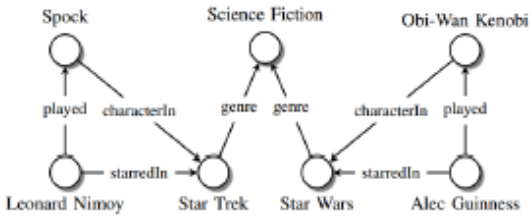


Image credit: Maximilian Nickel et al

Knowledge Graphs

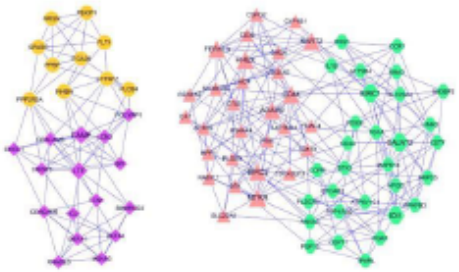


Image credit: ese.wustl.edu

Regulatory Networks

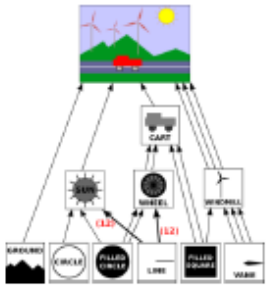


Image credit: math.hws.edu

Scene Graphs

Image credit: ResearchGate

Code Graphs

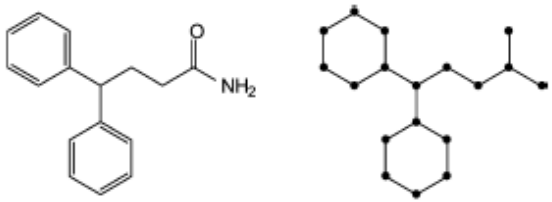


Image credit: MDPI

Molecules

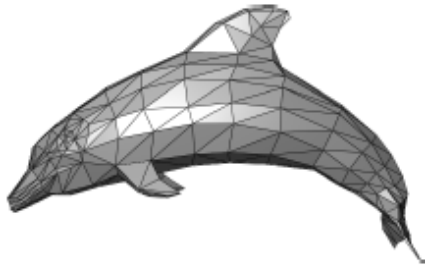


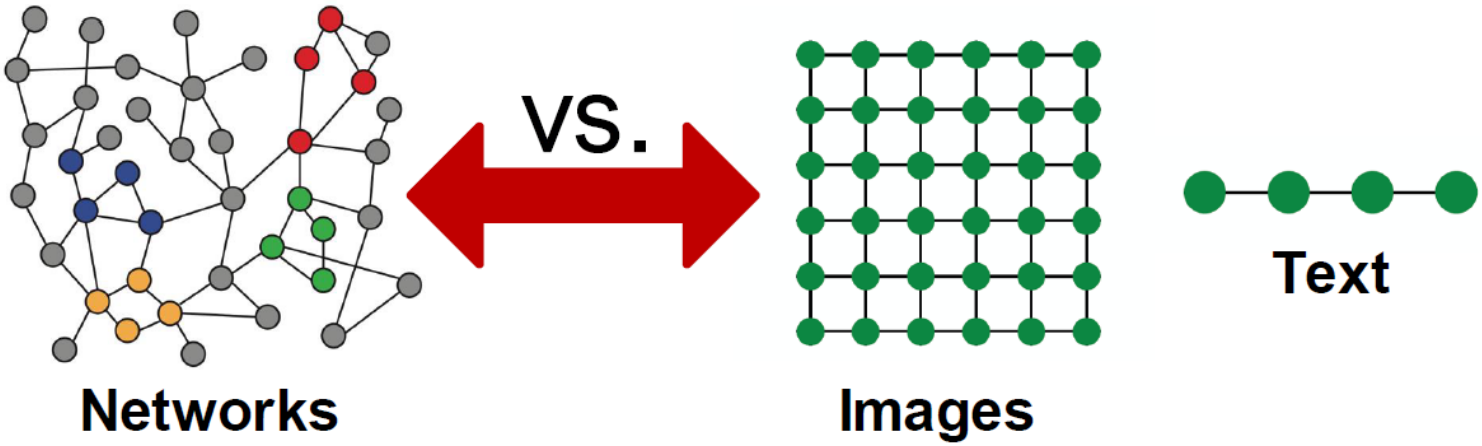
Image credit: Wikipedia

3D Shapes

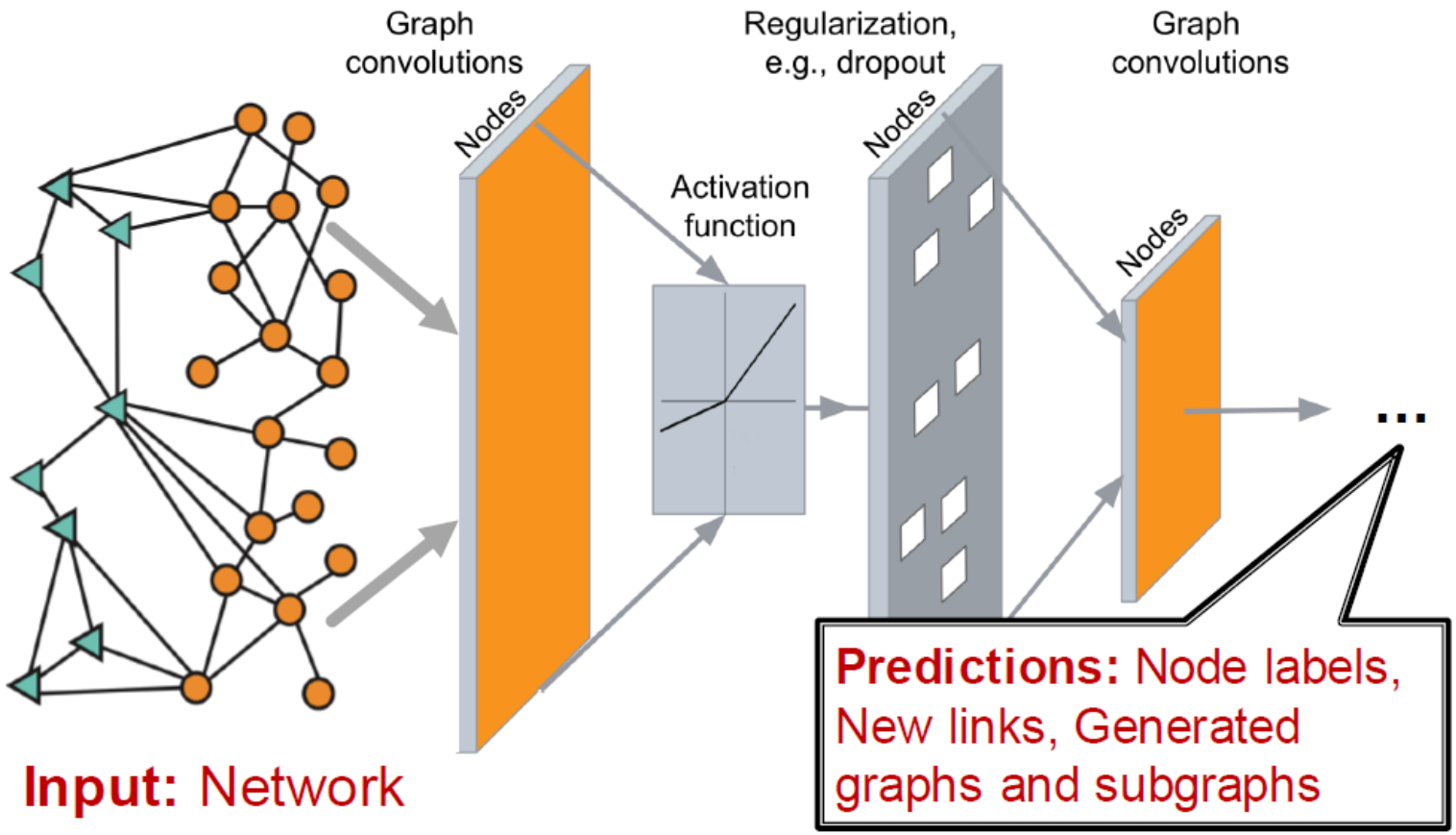
Modern Machine Learning

- Modern Machine Learning methods sepcifically Neural Networks are designed to work on sequences or grids.
- For example, test and audio signals are examples of sequences and images are an example of grids.
- But graphs are arbitrary size and have complex topological structure (i.e., no spatial locality like grids).
- Hence, we need to develop neural networks that are more broadly applicable.

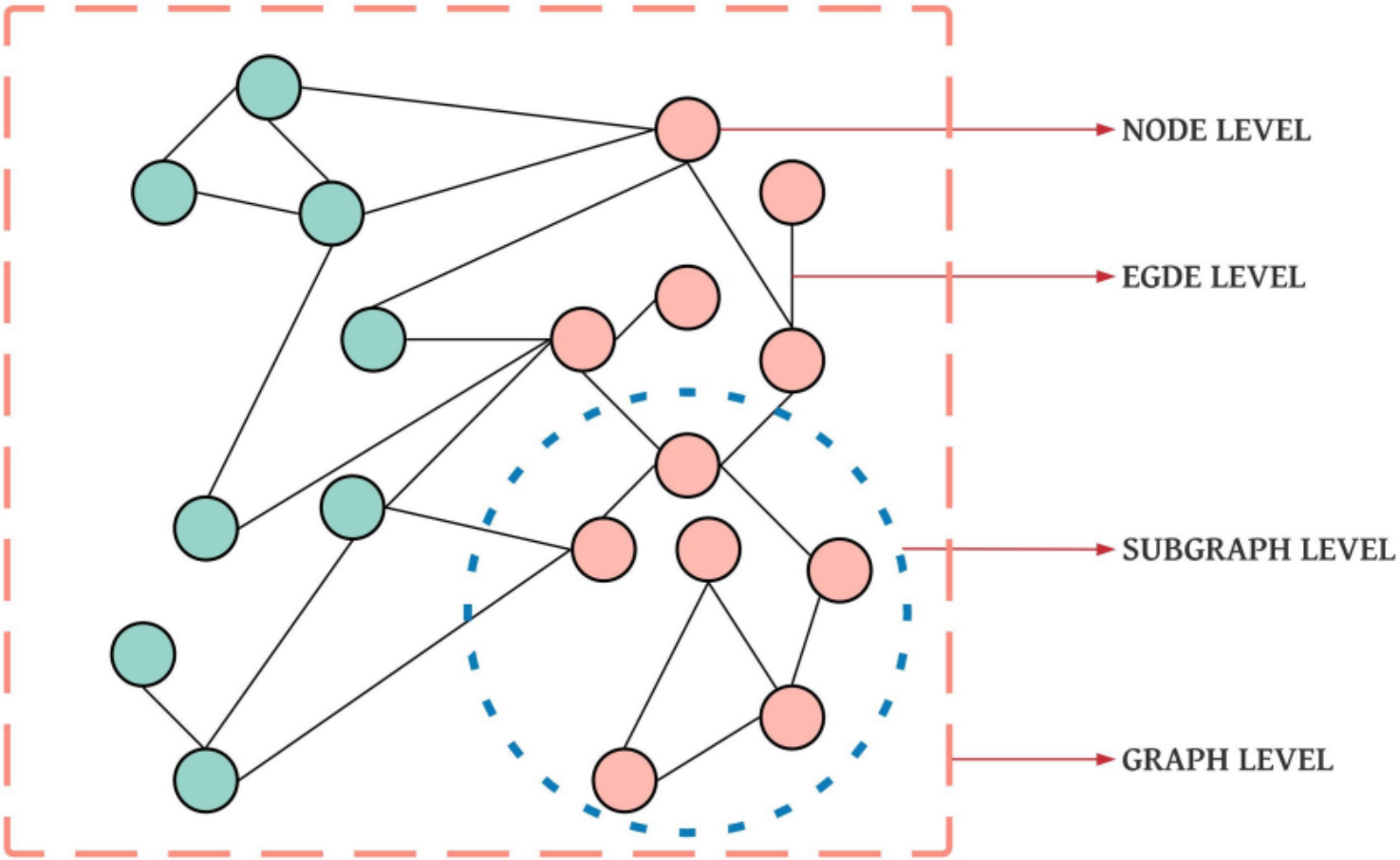
Modern Machine Learning (cont.)



Modern Machine Learning (cont.)



Different Levels of Tasks



[0]Waikhom & Patgiri, Graph Neural Networks: Methods, Applications, and Opportunities

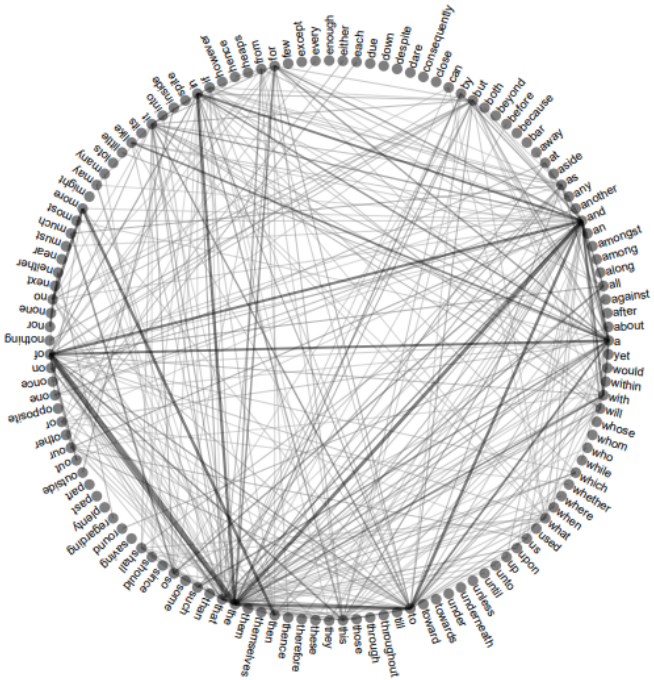
Graph ML Tasks

- Node classification: Predict a property of a node
  - Example: Categorize online users / items
- Link prediction: Predict whether there are missing links between two nodes
  - Example: Knowledge graph completion
- Graph classification: Categorize different graphs
  - Example: Molecule property prediction
- Clustering: Detect if nodes form a community
  - Example: Social circle detection
- Other tasks
  - Graph generation: Drug discovery
  - Graph evolution: Physical simulation

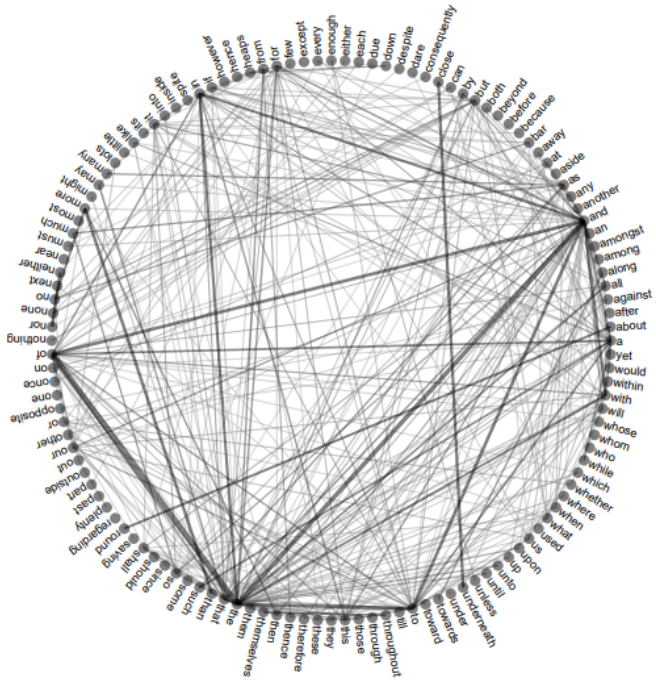
Some Examples - Authorship Attribution

- Word Adjancy Networks can be used to determine authorship.
- Nodes represent different words and edges represent how often words appear close to each other.

William Shakespeare



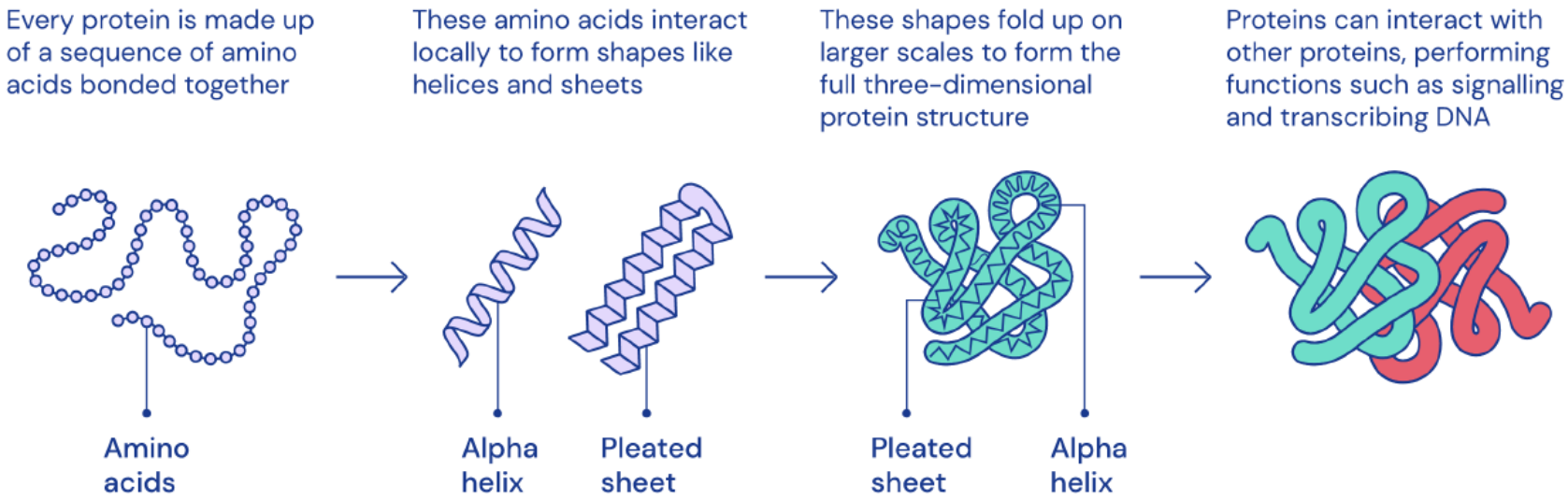
Christopher Marlowe



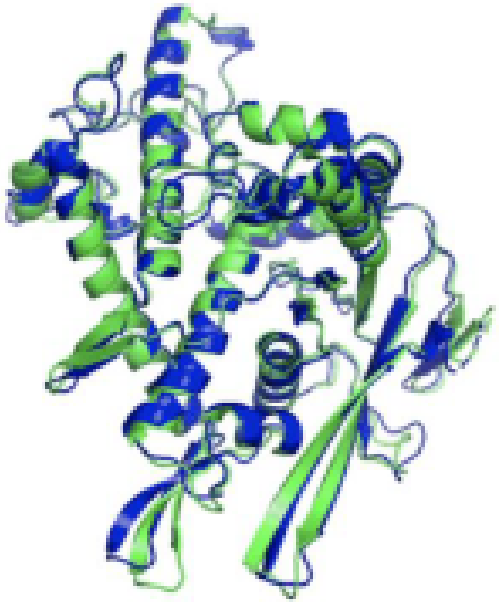
[0]Segarra-Eisen-Egan-Ribeiro, Attributing the Authorship of the Henry VI Plays by Word Adjacency

Some Examples - AlphaFold

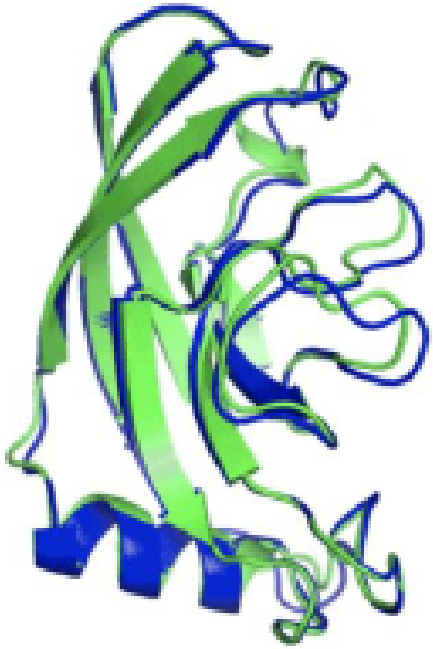
- A protein is made up amino acids and its structure is defined by interactions with each other.
- The goal is to predict the protein's structure based on the amino acid sequence.
- Amino acids act as nodes and edges represent the proximity between them.



Some Examples - AlphaFold (cont.)



T1037 / 6vr4
90.7 GDT
(RNA polymerase domain)

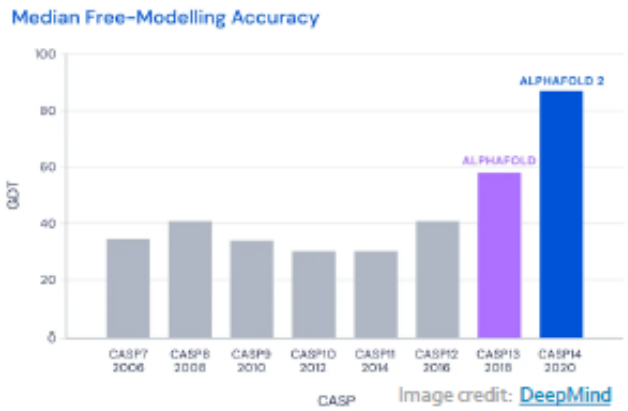


T1049 / 6y4f
93.3 GDT
(adhesin tip)

Experimental result

Computational prediction

Some Examples - AlphaFold (cont.)



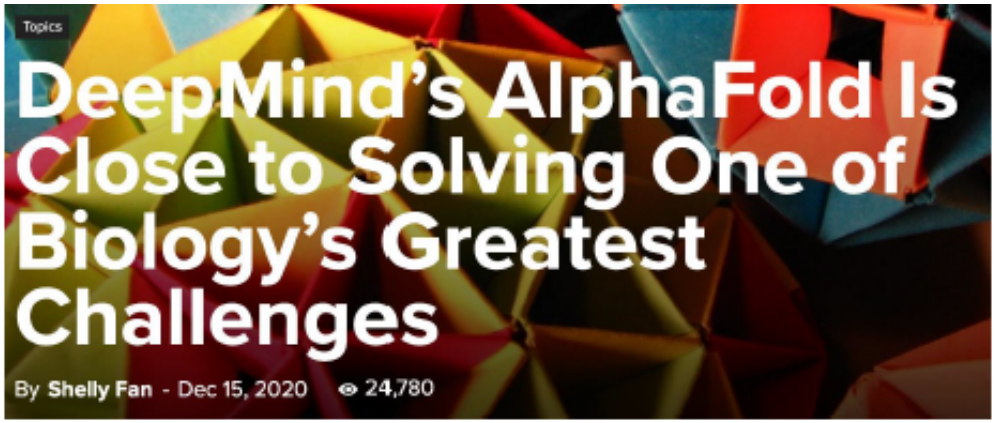


Image credit: SingularityHub

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

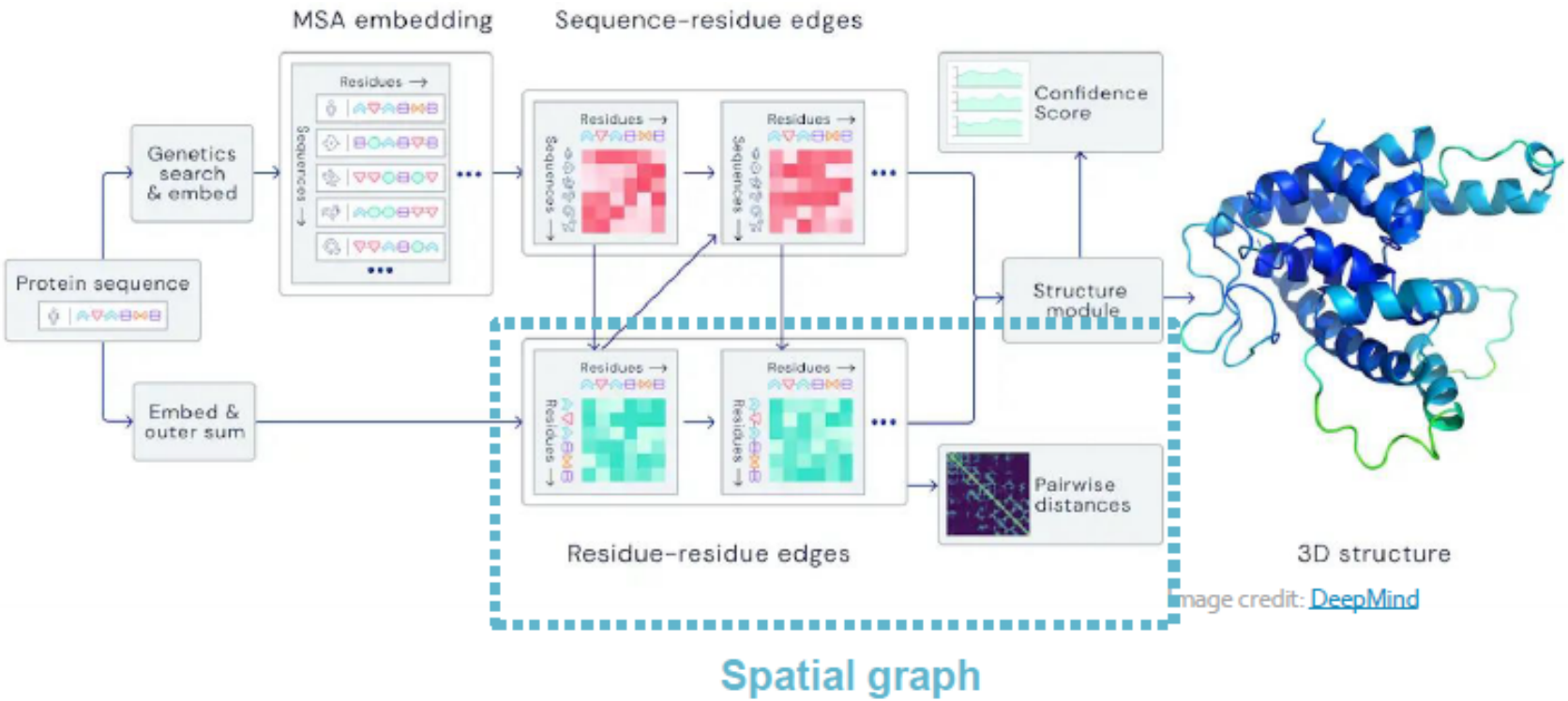
DeepMind's latest AI breakthrough can accurately predict the way proteins fold

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

12-14-20

DeepMind's latest AI breakthrough could turbocharge drug discovery

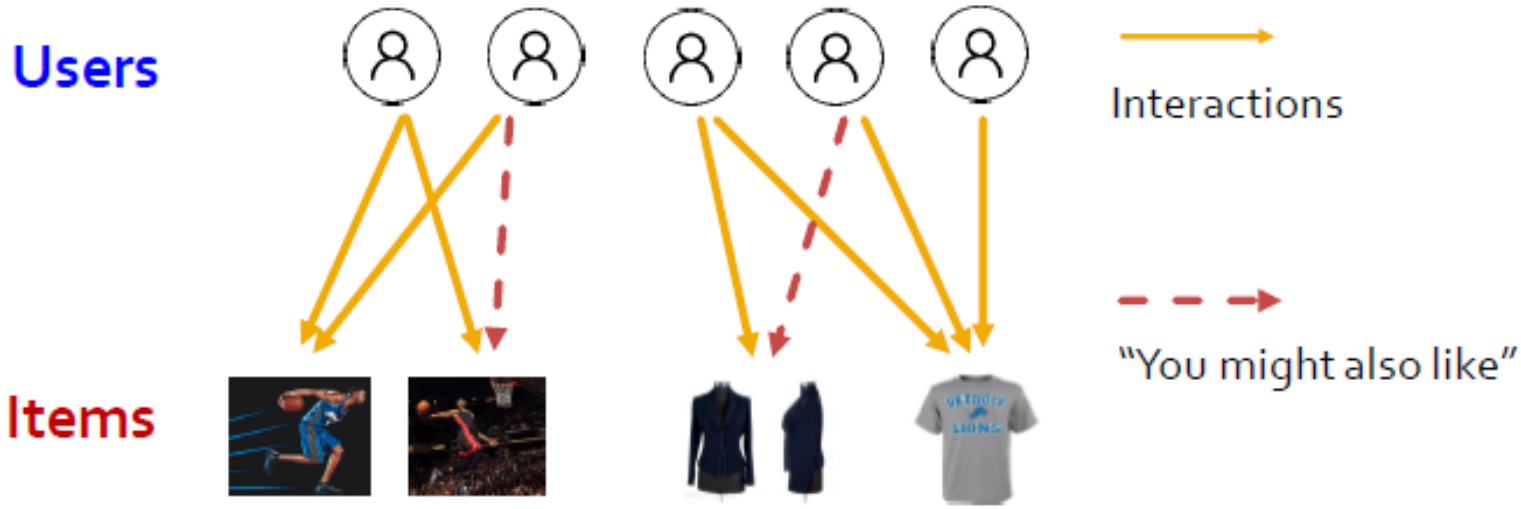
Some Examples - AlphaFold (cont.)



[0] Deepmind

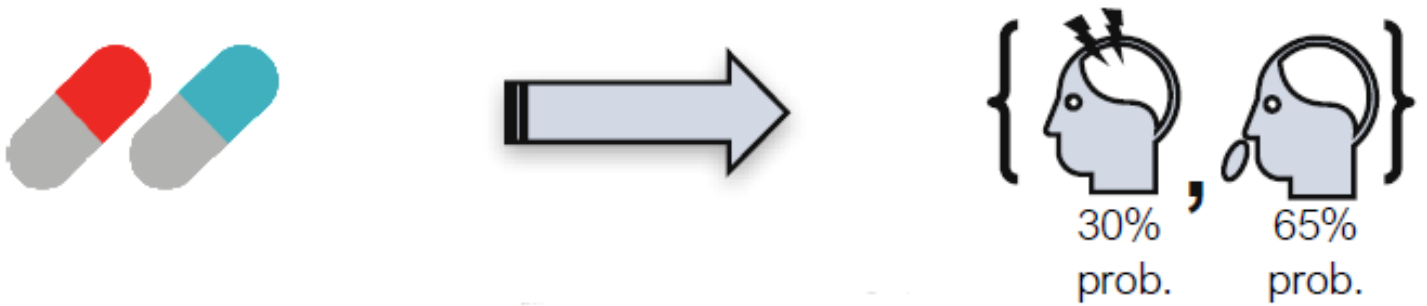
Some Examples - Recommender Systems

- Graphs can be used represent the relation between users and items they have viewed.
- Users and items are represented as nodes whereas edges represent the user-item interaction.
- The goal is to recommend new items to users.

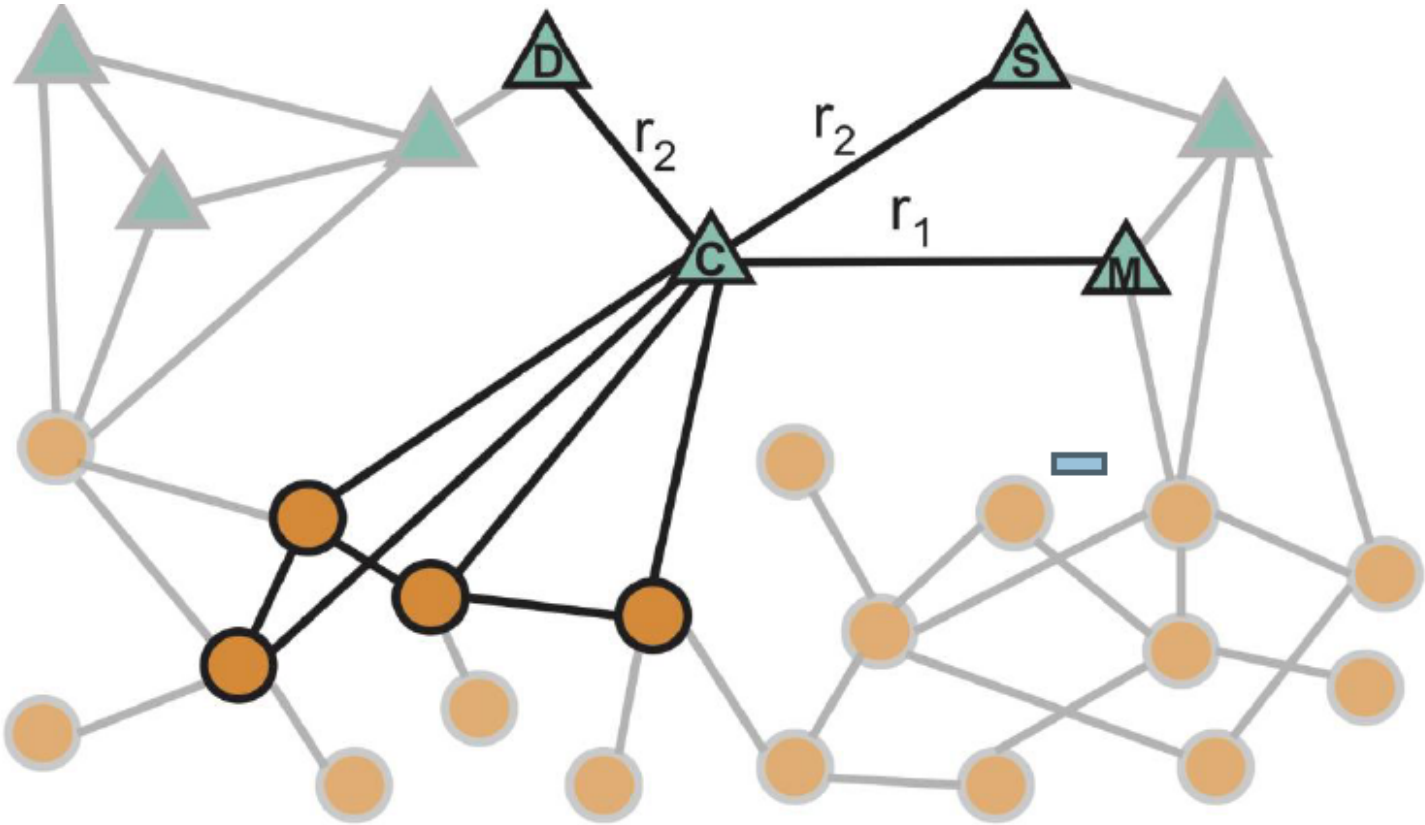


Some Examples - Drug Side Effects

- Many people take multiple drugs for their treatment.
- The goal is to determine if two drugs takes together can have an adverse side effect.
- The drugs and Proteins act as nodes whereas the edges represent the interaction between them.



Some Examples - Drug Side Effects (cont.)



△ Drug ○ Protein

r_1 Gastrointestinal bleed side effect

r_2 Bradycardia side effect

△—○ Drug-protein interaction

○—○ Protein-protein interaction

Some Examples - Drug Side Effects (cont.)

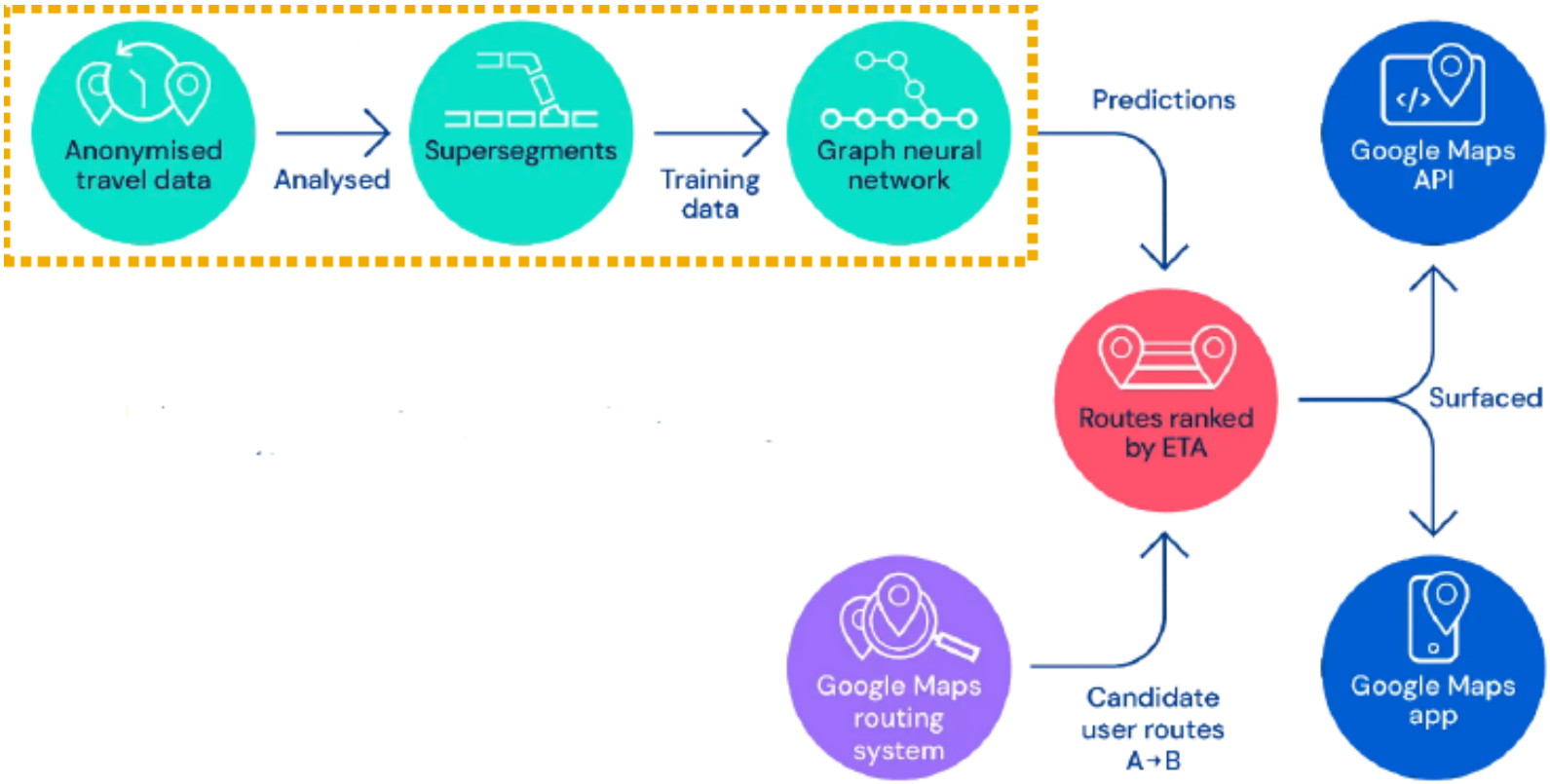
Rank	Drug c	Drug d	Side effect r	Evidence found
1	Pyrimethamine	Aliskiren	Sarcoma	Stage *et al.* 2015
2	Tigecycline	Bimatoprost	Autonomic neuropathy	
3	Omeprazole	Dacarbazine	Telangiectases	
4	Tolcapone	Pyrimethamine	Breast disorder	Bicker *et al.* 2017
5	Minoxidil	Paricalcitol	Cluster headache	
6	Omeprazole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

[0] Zitnik et al., Modeling Polypharmacy Side Effects with Graph Convolutional Networks

Some Examples - Traffic Prediction

- Intersections can be treated as nodes and roads between them as edges/
- The task could be to predict the time of arrival. For example, Google Maps uses the following.

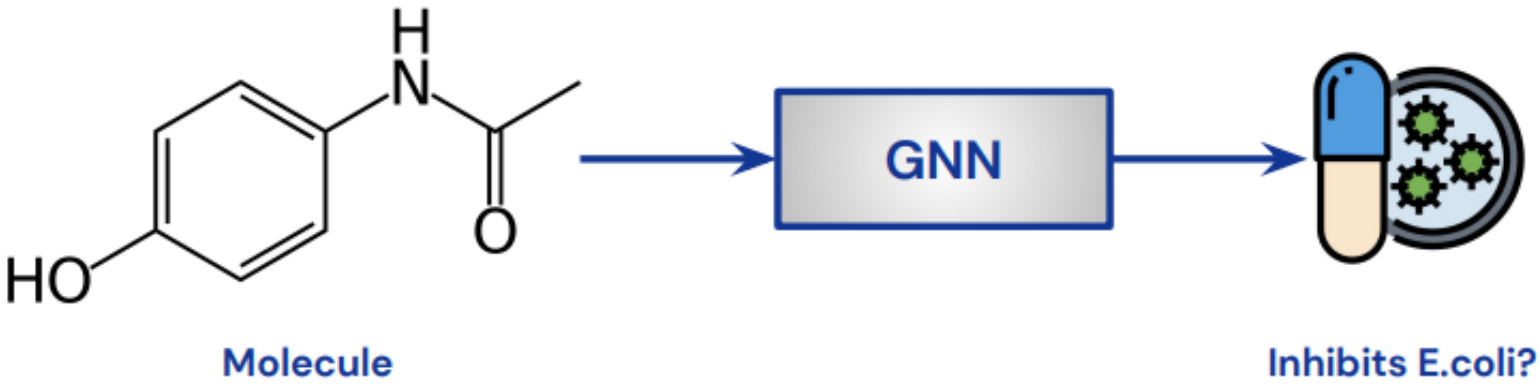
Some Examples - Traffic Prediction (cont.)



[0] Deepmind

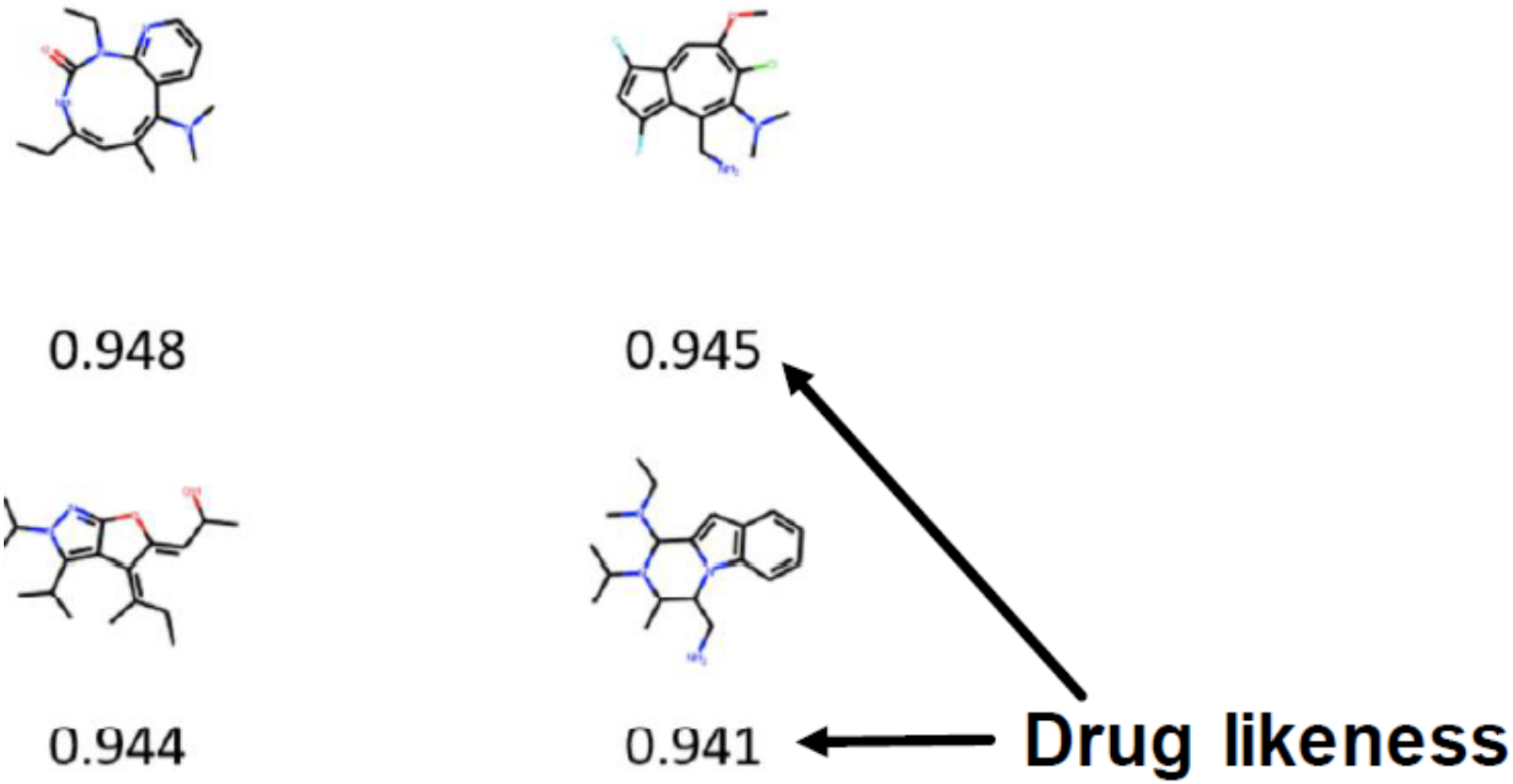
Some Examples - Drug Classification & Discovery

- A natural way to represent molecules is as graphs.
- Atoms are represented as nodes and bonds as edges.



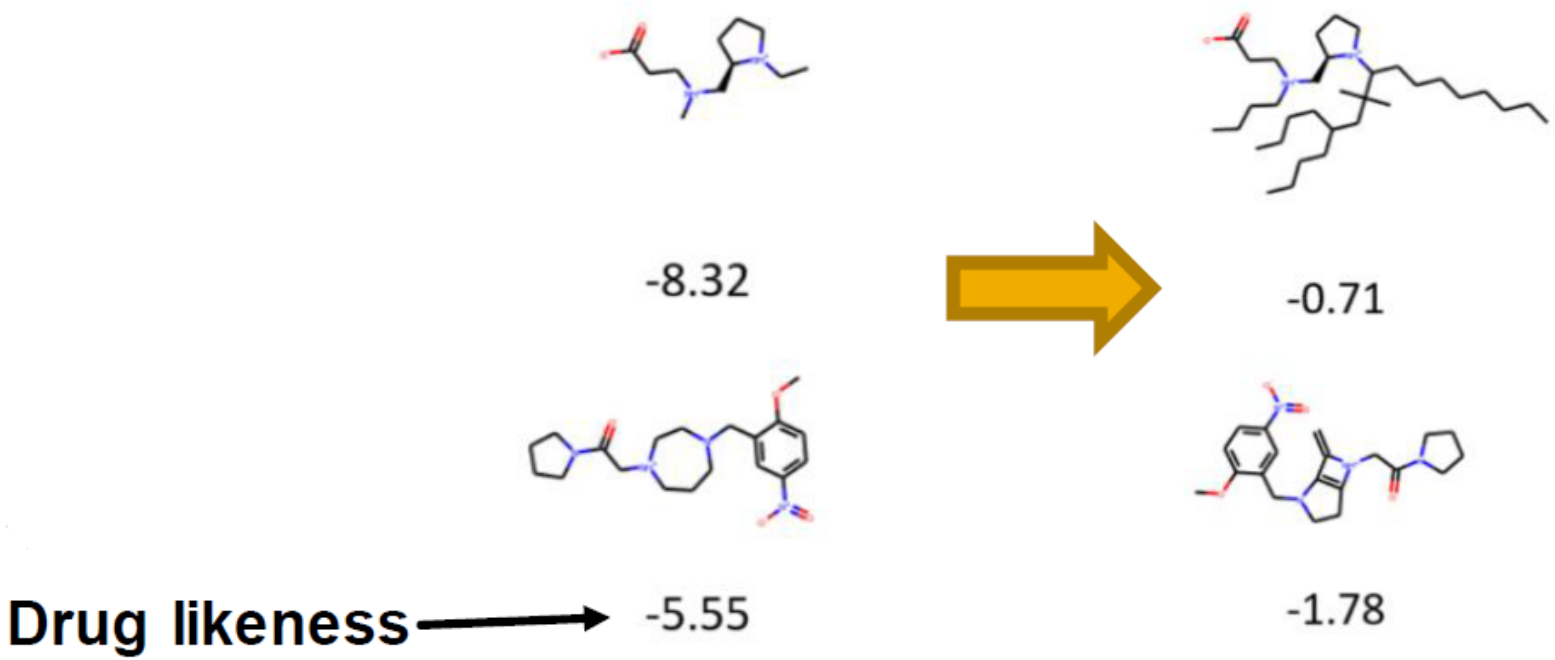
Some Examples - Drug Classification & Discovery (cont.)

- Generate novel molecules with high Drug likeness value.



Some Examples - Drug Classification & Discovery (cont.)

- Optimize existing molecules to have desirable properties.

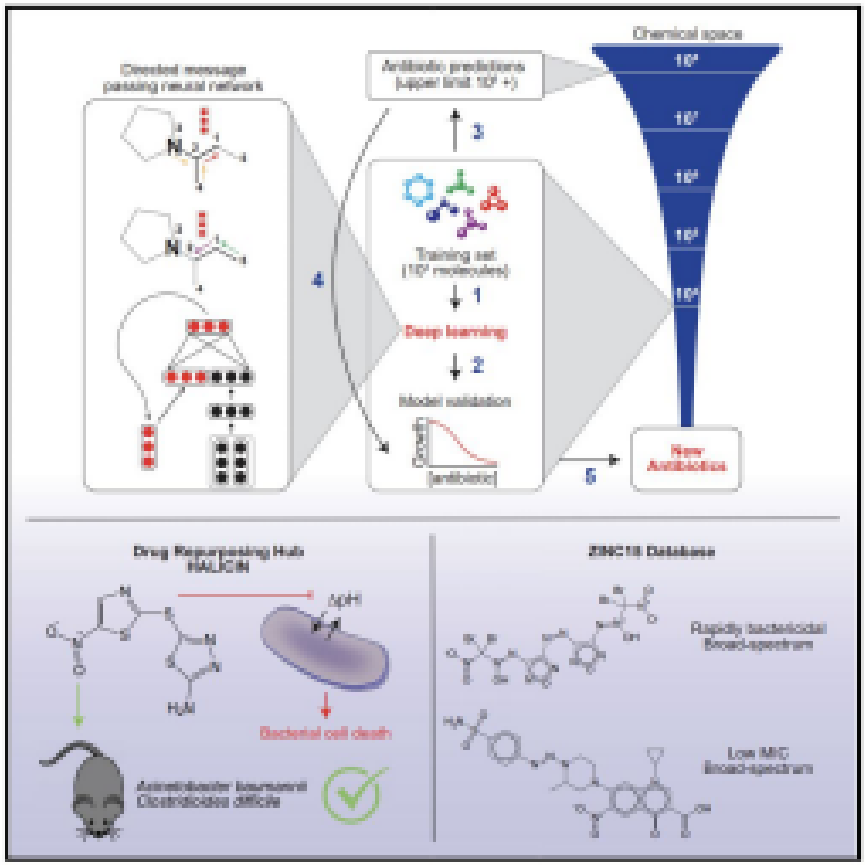


Some Examples - Drug Classification & Discovery (cont.)

Cell

A Deep Learning Approach to Antibiotic Discovery

Graphical Abstract



Authors

Jonathan M. Stokes, Kevin Yang, Kyle Swanson, ..., Tommi S. Jaakkola, Regina Barzilay, James J. Collins

Correspondence

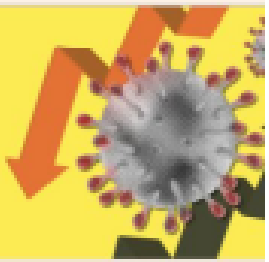
regina@csail.mit.edu (R.B.), jimjc@mit.edu (J.J.C.)

In Brief

A trained deep neural network predicts antibiotic activity in molecules that are structurally different from known antibiotics, among which Halicin exhibits efficacy against broad-spectrum bacterial infections in mice.

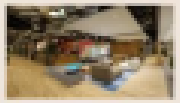
Some Examples - Drug Classification & Discovery (cont.)

FINANCIAL TIMES

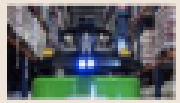


l intelligence

Robotics



'Death of the office' homeworking claims exaggerated



Anti-social robots harr increase social distanci

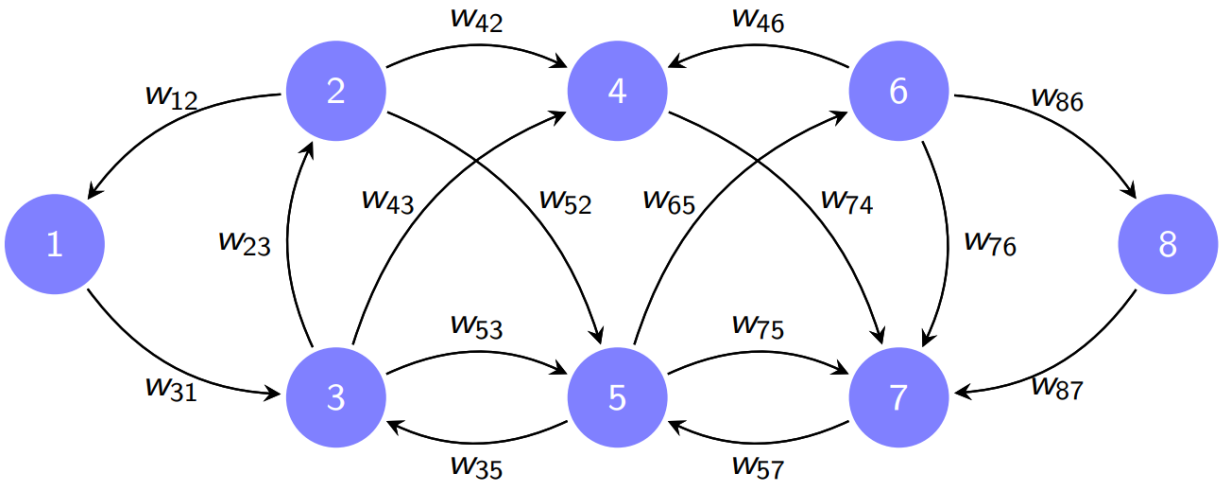
Artificial intelligence + Add to myFT

AI discovers antibiotics to treat drug-resistant diseases

Machine learning uncovers potent new drug able to kill 35 powerful bacteria

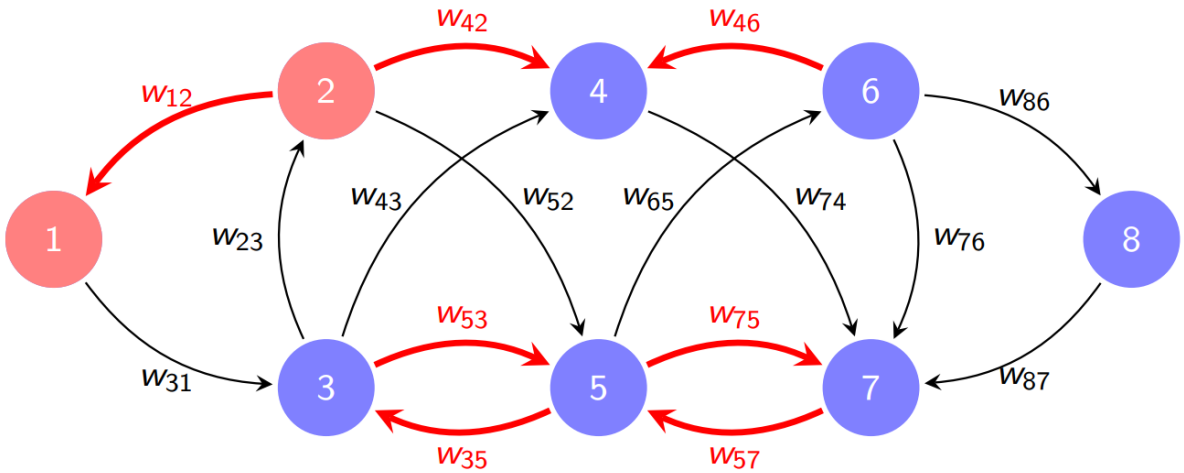
Refresher on Graphs

- A graph is a triplet $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, which includes vertices $\mathcal{V}$, edges $\mathcal{E}$, and weights $\mathcal{W}$.
  - Vertices or nodes are a set of n labels. Typical labels are $\mathcal{V} = \{1, \cdots, n\}$
  - Edges are ordered pairs of labels (i, j). We interpret $(i, j) \in \mathcal{E}$ as "i can be influenced by j."
  - Weights $w_{ij} \in \mathbb{R}$ are numbers associated to edges (i, j). "Strength of the influence of j on i."



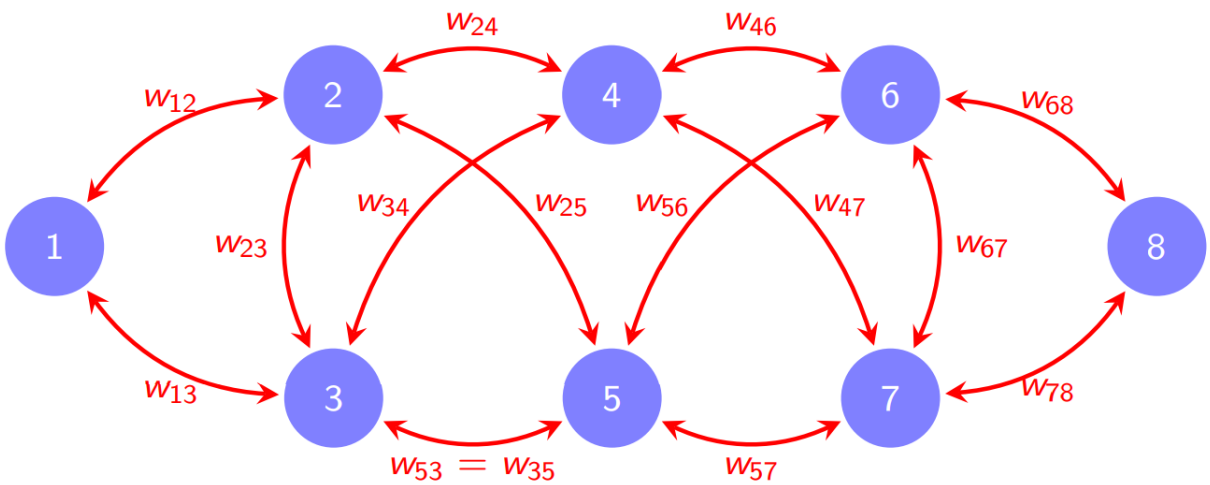
Directed Graphs

- Edge (i, j) is represented by an arrow pointing from j into i. Influence of node j on node i.
- Edge (i, j) is different from edge $(j, i) \Longrightarrow$ It is possible to have $(i, j) \in \mathcal{E}$ and $(j, i) \in \mathcal{E}$
- If both edges are in the edge set, the weights can be different $\Longrightarrow$ It is possible to have $w_{ij} \neq w_{ji}$



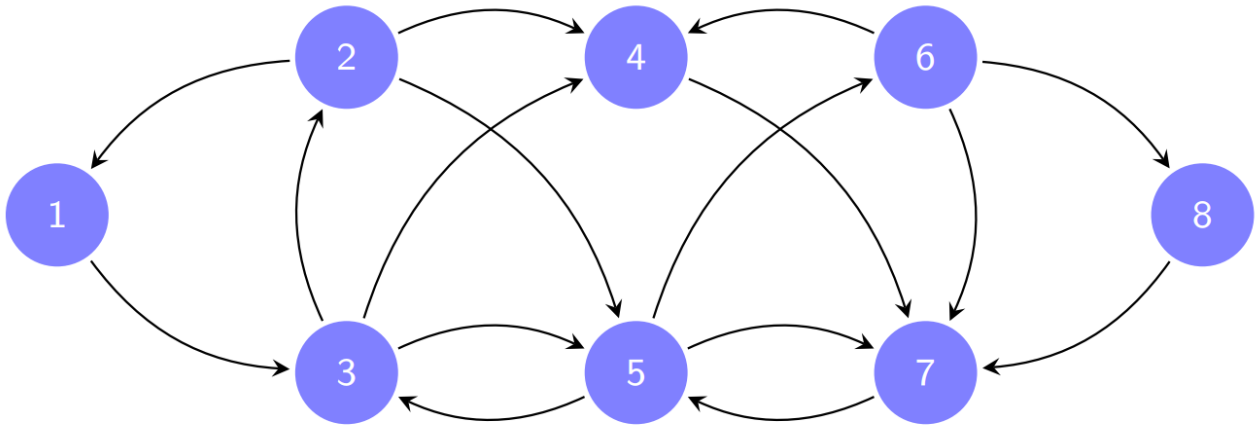
Undirected/Symmetric Graphs

- A graph is symmetric or undirected if both, the edge set and the weight are symmetric
  - Edges come in pairs $\Longrightarrow$ We have $(i, j) \in \mathcal{E}$ if and only if $(j, i) \in \mathcal{E}$
  - Weights are symmetric $\Longrightarrow$ We must have $w_{ij} = w_{ji}$ for all $(i, j) \in \mathcal{E}$



Unweighted Graphs

- A graph is unweighted if it doesn't have weights. Equivalently, all weights are unit i.e., $w_{ij} = 1 \, \forall \, (i, j) \in \mathcal{E}$.
- Unweighted graphs could be directed or symmetric.

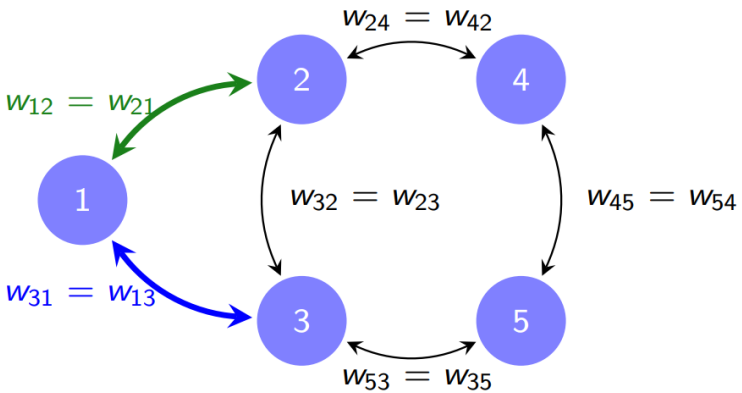


Adjacency Matrix

- The adjacency matrix of graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$ is the sparse matrix $\mathbf{A}$ with nonzero entries

$$A_{ij} = w_{ij}, \ \forall \, (i,j) \in \mathcal{E}$$

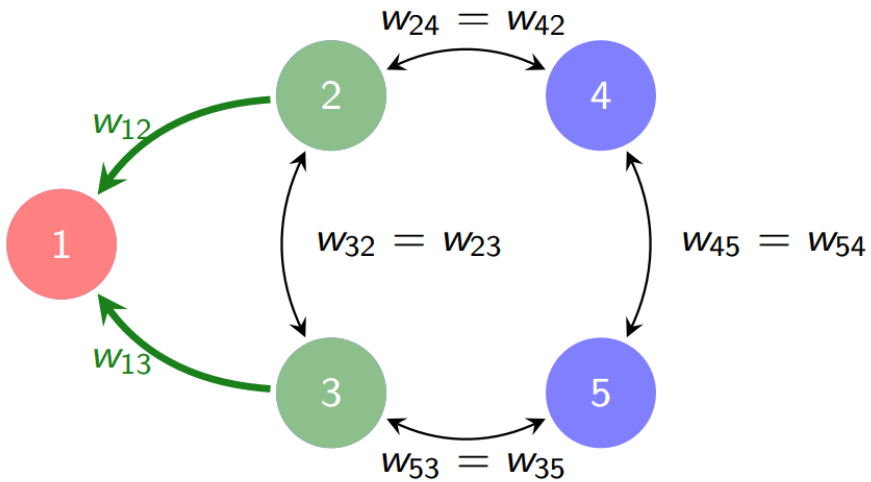
- If the graph is symmetric, the adjacency matrix is symmetric $\Longrightarrow \mathbf{A} = \mathbf{A}^T$



$$\begin{bmatrix} 0 & w_{12} & w_{13} & 0 & 0 \\ w_{21} & 0 & w_{23} & w_{24} & 0 \\ w_{31} & w_{32} & 0 & 0 & w_{35} \\ 0 & w_{42} & 0 & 0 & w_{45} \\ 0 & 0 & w_{53} & w_{54} & 0 \end{bmatrix}$$

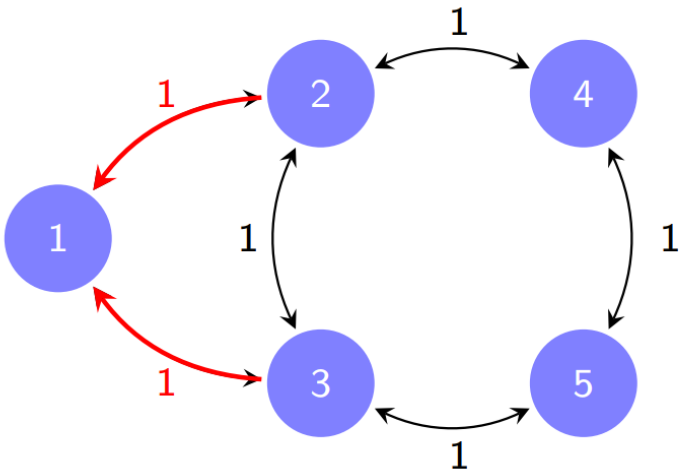
Neighborhoods and Degrees

- The neighborhood of node i is the set of nodes that influence $i \Longrightarrow n(i) := \{j : (i,j) \in \mathcal{E}\}$
  - Node 1 neighborhood $\Rightarrow n(1) = \{2, 3\}$.
- Degree d_i of node i is the sum of the weights of its incident edges $\Longrightarrow d_i = \sum_{j \in n(i)} w_{ij} = \sum_{j:(i,j)\in\mathcal{E}} w_{ij}$
  - Node 1 degree $\Rightarrow n(1) = w_{12} + w_{13}$



Degree Matrix

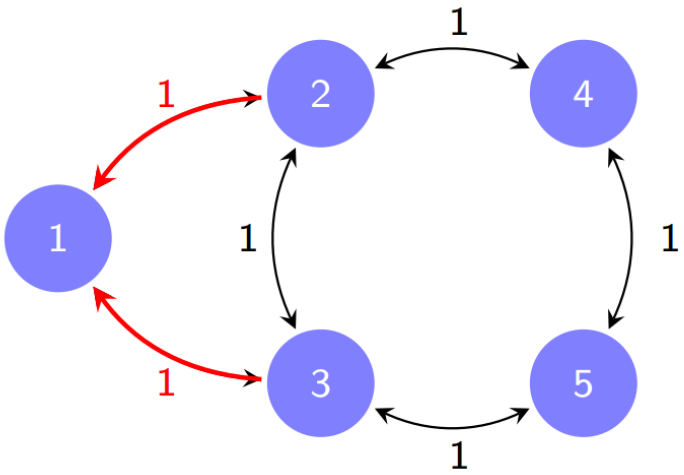
- The degree matrix is a diagonal matrix **D** with degrees as diagonal entries $\Rightarrow D_{ii} = d_i$



$$\begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

Laplacian Matrix

- The Laplacian matrix of a graph with adjacency matrix $\mathbf{A}$ is
  $\Rightarrow \mathbf{L} = \mathbf{D} - \mathbf{A} = \text{diag}(\mathbf{A1}) - \mathbf{A}$



$$\begin{bmatrix} 2 & -1 & -1 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ -1 & -1 & 3 & 0 & -1 \\ 0 & -1 & 0 & 2 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{bmatrix}$$

Normalized Matrix Representations

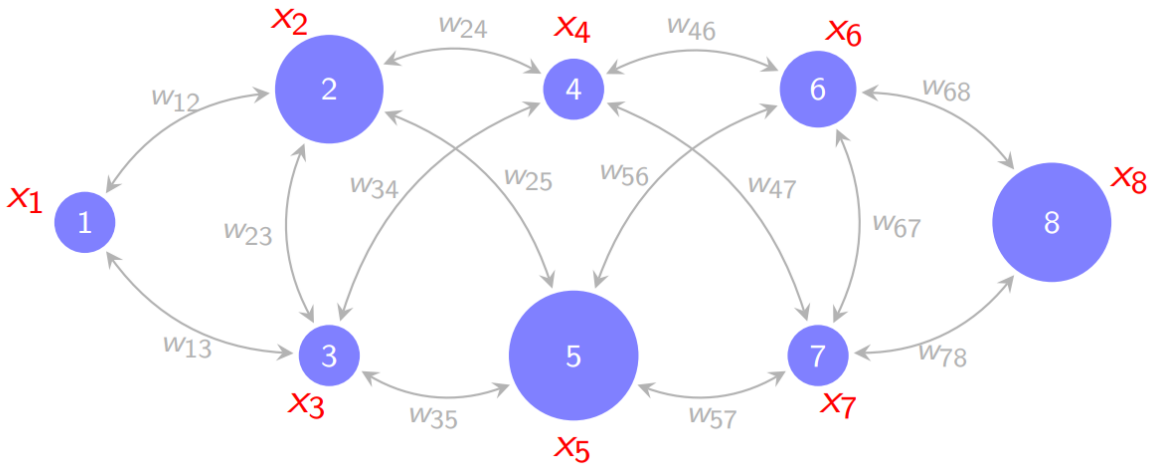
- Normalized Adjacency and Laplacian matrices express weights relative to the nodes' degrees.
- Normalized adjacency matrix $\Rightarrow \bar{\mathbf{A}} := \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \Rightarrow$ Results in entries $(\bar{\mathbf{A}})_{ij} = \frac{w_{ij}}{\sqrt{d_i d_j}}$
- Normalized Laplacian matrix $\Rightarrow \bar{\mathbf{L}} := \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{A})\mathbf{D}^{-1/2} = \mathbf{I} - \bar{\mathbf{A}}$
- Normalized operators are more homogeneous.

Graph Shift Operator

- The Graph Shift Operator $\mathbf{S}$ is a stand in for any of the matrix representations of the graph.
  - Adjacency Matrix: $\mathbf{S} = \mathbf{A}$
  - Laplacian Matrix: $\mathbf{S} = \mathbf{L}$
  - Normalized Adjacency Matrix: $\mathbf{S} = \bar{\mathbf{A}}$
  - Normalized Laplacian Matrix: $\mathbf{S} = \bar{\mathbf{L}}$
- The specific choice matters in practice but most of results and analysis hold for any choice of $\mathbf{S}$.

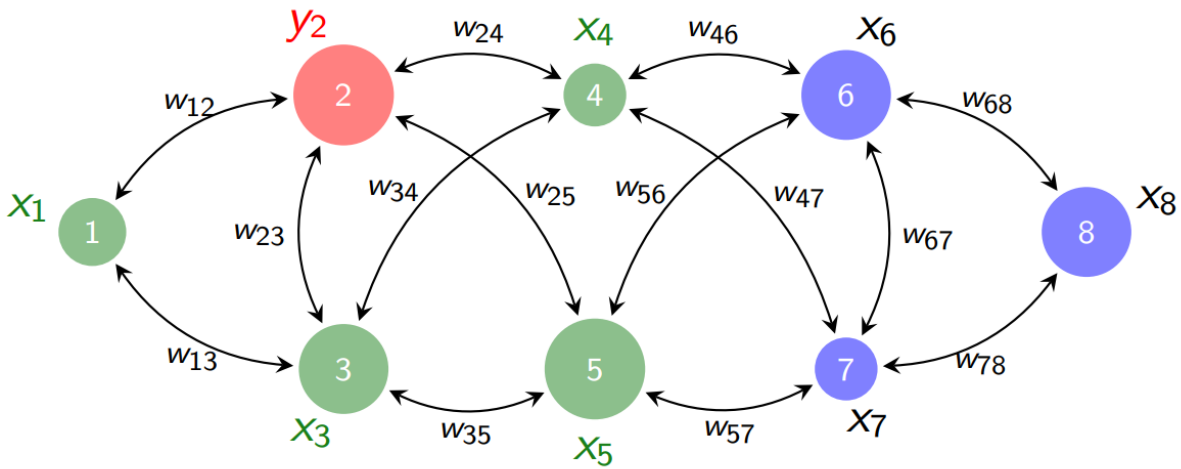
Graph Signals

- Consider a given graph $\mathcal{G}$ with n nodes and shift operator $\mathbf{S}$.
- A graph signal is a vector $\mathbf{x} \in \mathbb{R}^n$ in which component x_i is associated with node i.
- To emphasize that the graph is intrinsic to the signal we may write the signal as a pair $(\mathbf{S}, \mathbf{x})$
- The graph is an expectation of proximity or similarity between components of the signal $\mathbf{x}$.



Graph Signal Diffusion

- Multiplication by the graph shift operator implements diffusion of the signal over the graph.
- Define diffused signal $\mathbf{y} = \mathbf{S}\mathbf{x} \Rightarrow$ Components are $y_i = \sum_{j \in n(i)} w_{ij} x_j$.
  - Stronger weights contribute more to the diffusion output.
  - Codifies a local operation where components are mixed with components of neighboring nodes.



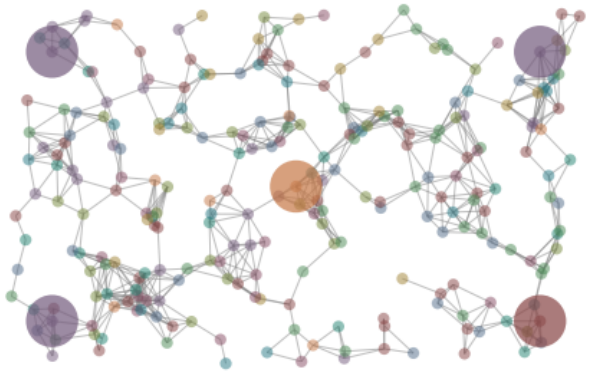
Graph Signal Diffusion (cont.)

- This diffusion operator can be composed to create a diffusion sequence.

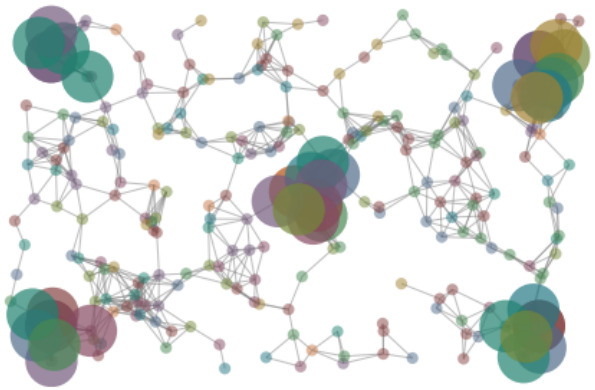
$$\mathbf{x}^{(k+1)} = \mathbf{S}\mathbf{x}^{(k)}, \text{ with } \mathbf{x}^{(0)} = \mathbf{x}$$

- This can also be unrolled and written as power sequence i.e., $\mathbf{x}^{(k)} = \mathbf{S}^k\mathbf{x}$
- The k^{th} element of the diffusion sequence $\mathbf{x}^{(k)}$ diffuses information to k-hop neighborhoods.

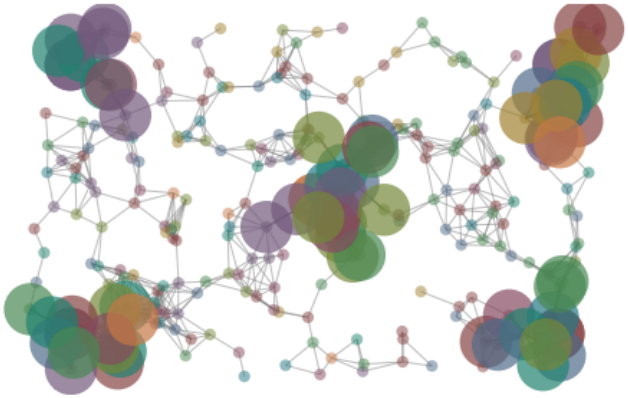
Graph Signal Diffusion (cont.)



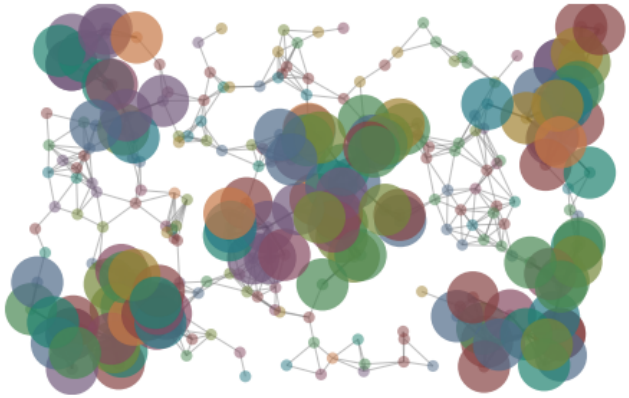
$$\mathbf{x}^{(0)} = \mathbf{x} = \mathbf{S}^0\mathbf{x}$$



$$\mathbf{x}^{(1)} = \mathbf{S}\mathbf{x}^{(0)} = \mathbf{S}^1\mathbf{x}$$



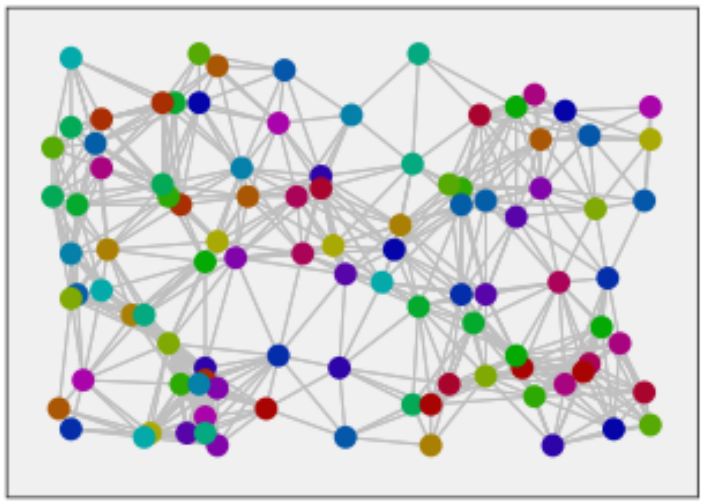
$$\mathbf{x}^{(2)} = \mathbf{S}\mathbf{x}^{(1)} = \mathbf{S}^2\mathbf{x}$$



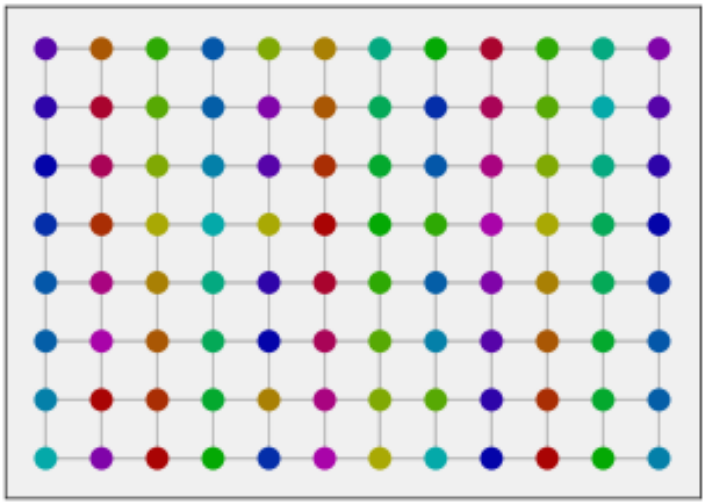
$$\mathbf{x}^{(3)} = \mathbf{S}\mathbf{x}^{(2)} = \mathbf{S}^3\mathbf{x}$$

- Generic NNs do not scale to large dimensions but Convolutional Neural Networks (CNNs) do scale.

Challenge is we want to run a NN over this



But we are good at running NNs over this

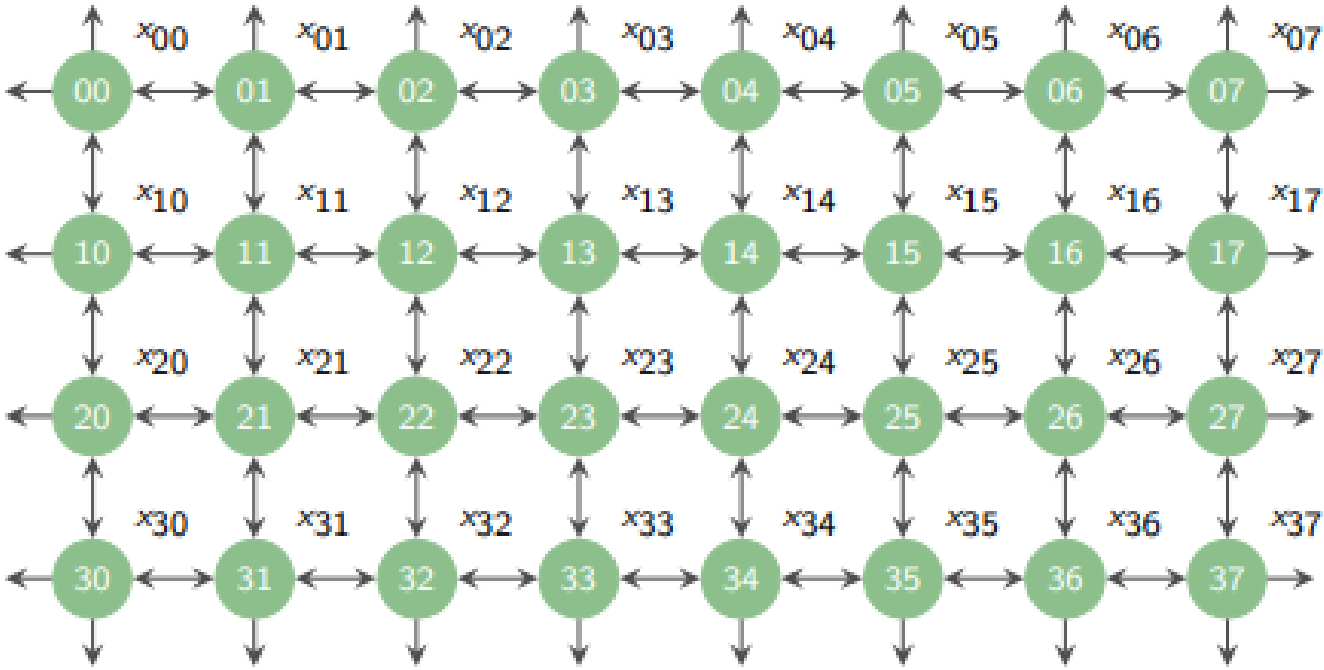


Graph Convolutional Neural Networks (cont.)

- CNNs are made up of layers composing convolutional filter banks with pointwise nonlinearities.
- Generalize convolutions to graphs i.e., Compose graph filter banks with pointwise nonlinearities.
- Just as we process images with convolutional NNs, now process graphs with graph convolutional NNs.
- How do we generalize convolutions in time and space to operate on graphs?
- Even though we do not often think of them as such, convolutions are operations on graphs.

Graph Convolutional Neural Networks (cont.)

- We can describe discrete time and space using graphs that support time or space signals.
- For example, we can describe images (space) with a grid graph.



- Given graph shift operator $\mathbf{S}$ and coefficients h_k , a graph filter is a polynomial (series) on $\mathbf{S}$.

$$\mathbf{H}(\mathbf{S}) = \sum_{k=0}^{\infty} h_k \mathbf{S}^k$$

- The result of applying the filter H(S) to the signal x is the signal

$$\mathbf{y} = \mathbf{H}(\mathbf{S})\mathbf{x} = \sum_{k=0}^{\infty} h_k \mathbf{S}^k \mathbf{x}$$

- We say that $\mathbf{y} = \mathbf{h} \star \mathbf{S}\mathbf{x}$ is the graph convolution of the filter $\mathbf{h} = \{h_k\}_{k=0}^{\infty}$ with the signal $\mathbf{x}$.

Graph Filters (cont.)

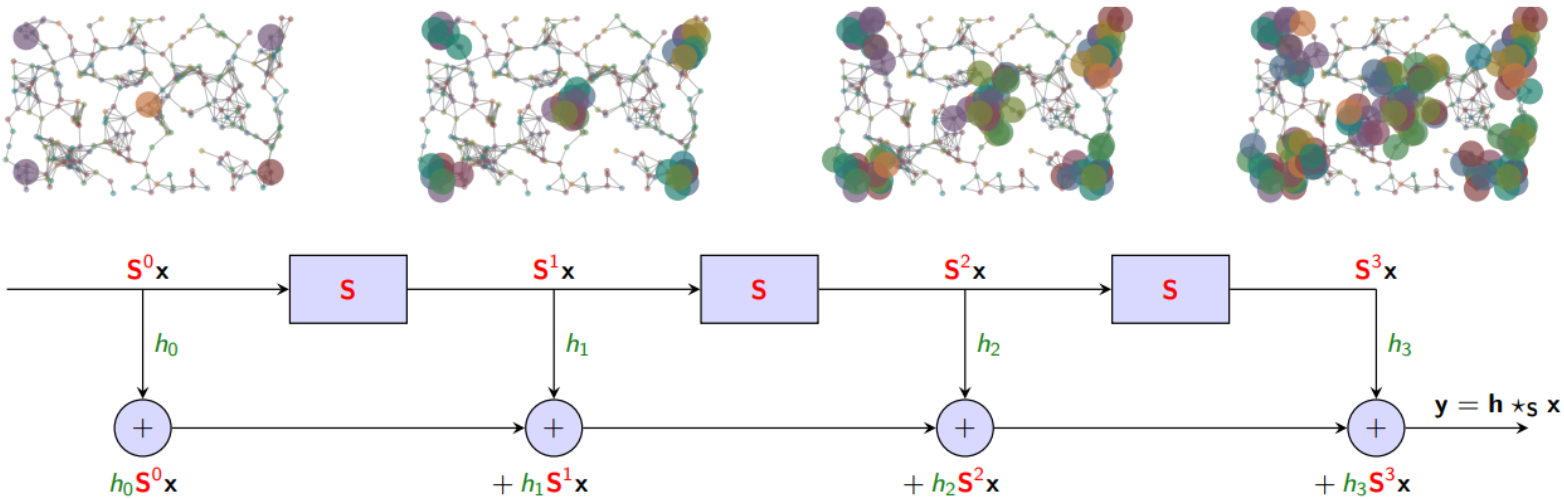
- Graph convolutions aggregate information growing from local to global neighborhoods.
- Consider a signal $\mathbf{x}$ supported on a graph with shift operator $\mathbf{S}$. Along with filter $\mathbf{h} = \{h_k\}_{k=0}^{K-1}$.
- Graph convolution output is then given by

$$\mathbf{y} = \mathbf{h} \star \mathbf{S}\mathbf{x} = h_0\mathbf{S}^0\mathbf{x} + h_1\mathbf{S}^1\mathbf{x} + h_2\mathbf{S}^2\mathbf{x} + h_3\mathbf{S}^3\mathbf{x} + \cdots = \sum_{k=0}^{K-1} h_k\mathbf{S}^k\mathbf{x}$$

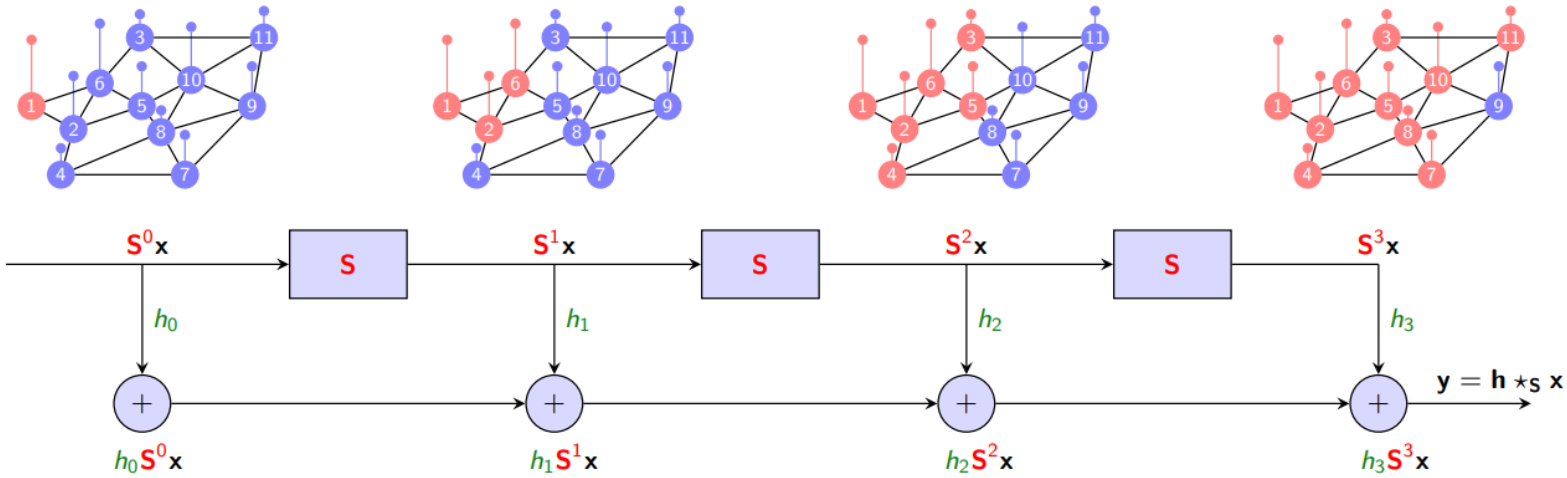
- The same filter $\mathbf{h} = \{h_k\}_{k=0}^{K-1}$ can be executed in multiple graphs $\Rightarrow$ We can transfer the filter.
- Output depends on the filter coefficients $\mathbf{h}$, the graph shift operator $\mathbf{S}$ and the signal $\mathbf{x}$.

Graph Filters (cont.)

- A graph convolution is a weighted linear combination of the elements of the diffusion sequence.
- Can represent graph convolutions with a shift register $\Rightarrow$ Convolution $\equiv$ Shift. Scale. Sum.

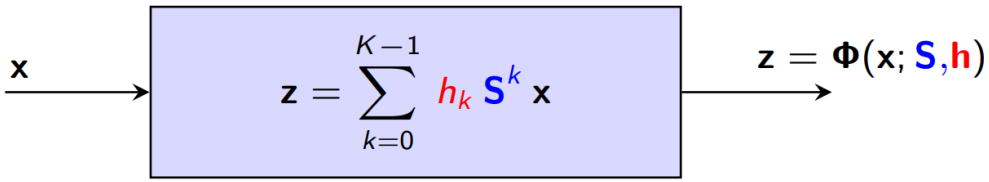


Graph Filters (cont.)



Learning with Graph Filters

- Function class $\Rightarrow$ graph filters of order K supported on $\mathbf{S}$ $\Rightarrow \Phi(\mathbf{x}) = \sum_{k=0}^{\infty} h_k \mathbf{S}^k \mathbf{x} = \Phi(\mathbf{x}; \mathbf{S}, \mathbf{h})$

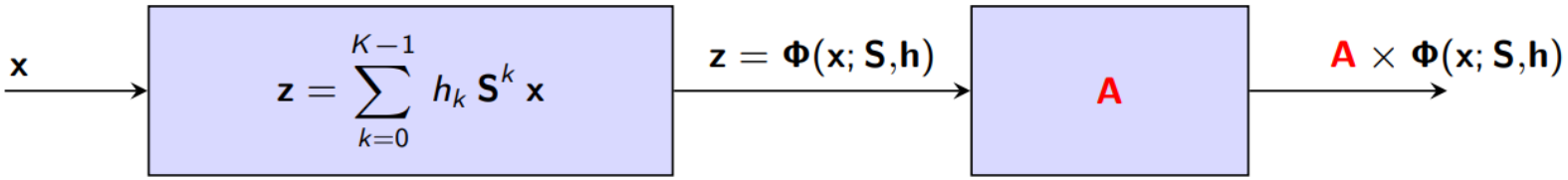


- Optimization is over filter coefficients h with the graph shift operator S given

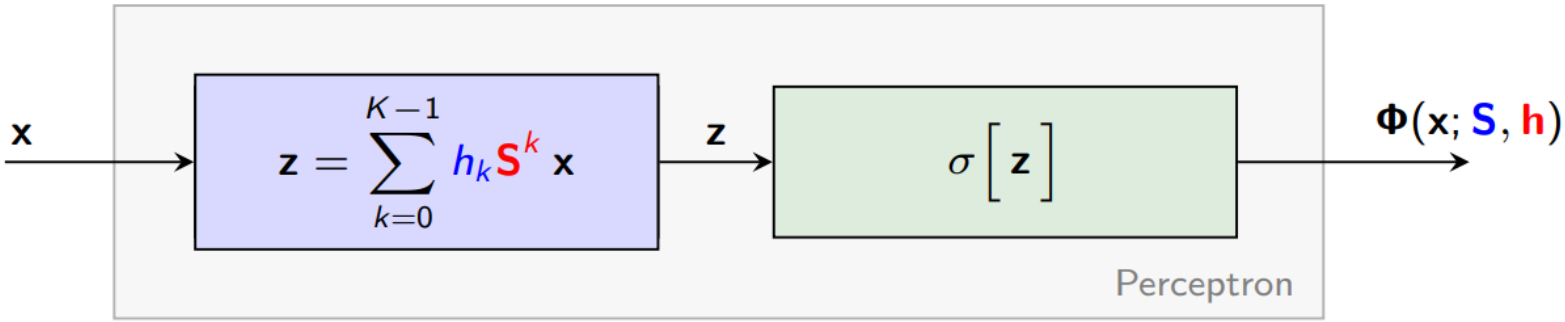
$$h^* = \underset{h}{\operatorname{argmin}} \sum_{(x,y)\in\mathcal{D}} \mathcal{L}\left(\mathbf{y}, \Phi(\mathbf{x}; \mathbf{S}, \mathbf{h})\right)$$

Learning with Graph Filters (cont.)

- If outputs are not signals, we can add a readout layer to match dimensions.
- Readout matrix $\mathbf{A} \in \mathbb{R}^{mn}$ yields parametrization.



- Graph filters have limited expressive power because they can only learn linear maps.
- We can also appl activation functions over the output to introduce non-linear mapping.
- This is called a graph perceptron.
- Perceptron allows learning of nonlinear maps which makes them more expressive. Larger Representable Class.



Graph Neural Networks (cont.)

- To define a GNN we compose several graph perceptrons i.e., We layer graph perceptrons.
- Layer 1 processes input signal $\mathbf{x}$ with the perceptron $h1 = [h_{10}, \cdots, h_{1,K-1}]$ to produce output $\mathbf{x_1}$.

$$\mathbf{x_1} = \sigma\left[\mathbf{z}_1\right] = \sigma\left[\sum_{k=0}^{K-1} h_{1k}\mathbf{S}^k\mathbf{x}\right]$$

- The Output of Layer 1 $\mathbf{x_1}$ becomes an input to Layer 2. Still $\mathbf{x}$ but with different interpretation.
- Repeat analogous operations for L times (the GNNs depth) $\Rightarrow$ Yields the GNN predicted output $\mathbf{x_L}$.

Graph Neural Networks (cont.)

- A generic layer of the GNN, Layer l, takes as input the output $xl-1$ of the previous layer $(l-1)$.
- Layer l processes input signal $\mathbf{x_{l-1}}$ with the perceptron $h_l = [h_{l0}, \cdots, h_{l,K-1}]$ to produce output $\mathbf{x_l}$.

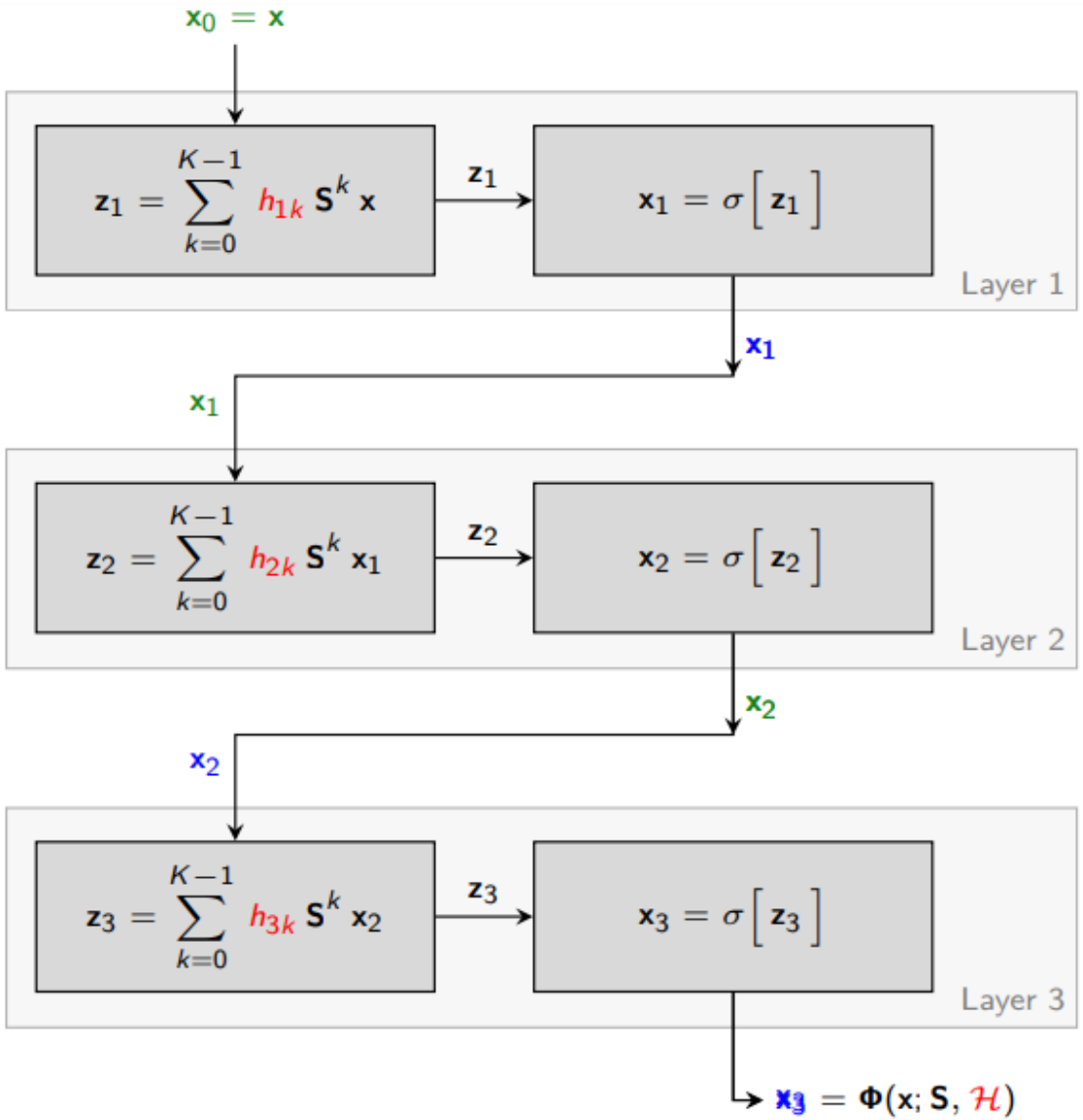
$$\mathbf{x_l} = \sigma\left[\mathbf{z}_l\right] = \sigma\left[\sum_{k=0}^{K-1} h_{lk}\mathbf{S}^k\mathbf{x_{l-1}}\right]$$

- With the convention that the Layer 1 input is $\mathbf{x}_0 = \mathbf{x}$, this provides a recursive definition of a GNN.
- If it has L layers, then the GNN output

$$x_L = \Phi(\mathbf{x}; \mathbf{S}, \mathbf{h}_1, \cdots, \mathbf{h}_L) = \Phi(\mathbf{x}; \mathbf{S}, \mathcal{H})$$

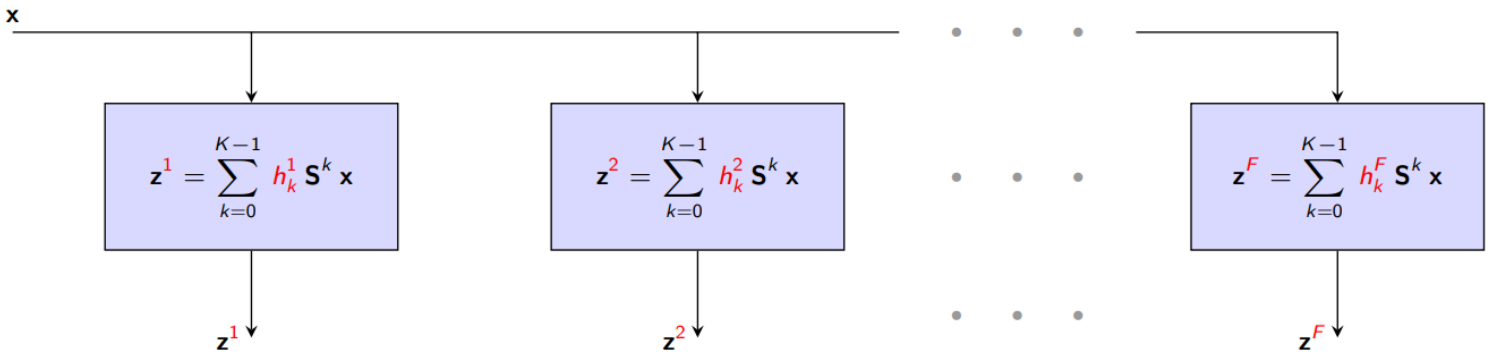
- The filter tensor $\mathcal{H} = [h_1, \cdots, h_L]$ is the trainable parameter. The graph shift operator $\mathbf{S}$ is prior information.

Graph Neural Networks (cont.)



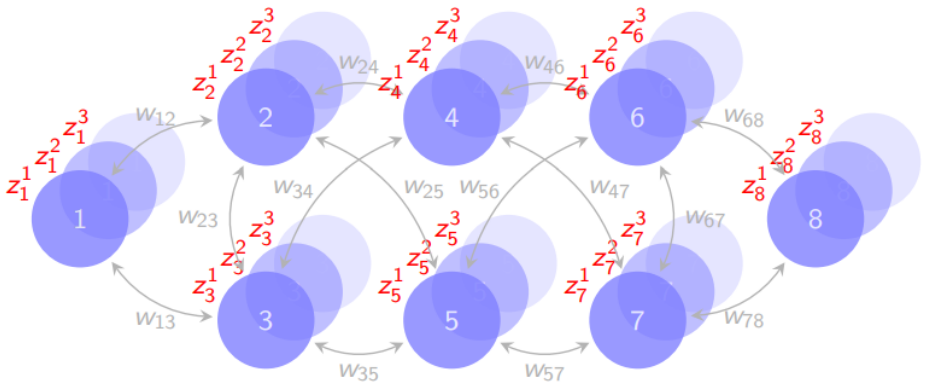
Filter Banks

- Filters isolate features. When we are interested in multiple features, we use Banks of filters.
- A graph filter bank is a collection of filters. Use F to denote total number of filters in the bank.
- Filter f in the bank uses coefficients $\mathbf{h}^f = [h_1^f, \cdots, h_{K-1}^f]$ Output z^f is a graph signal.
- Filter bank output is a collection of F graph signals. Matrix graph signal $\mathbf{Z} = [\mathbf{z}^1, \cdots, \mathbf{z}^F]$



Filter Banks (cont.)

- The input of a filter bank is a single graph signal $\mathbf{x}$. Rows of $\mathbf{x}$ are signals components x_i.
- Output matrix $\mathbf{Z}$ is a collection of signals $\mathbf{z}^f$. Rows of which are components z_i^f.
- Vector $\mathbf{z}_i$ supported at each node. Columns of $\mathbf{Z}$ are graph signals $\mathbf{z}^f$.Rows of Z are node features $\mathbf{z}_i$.

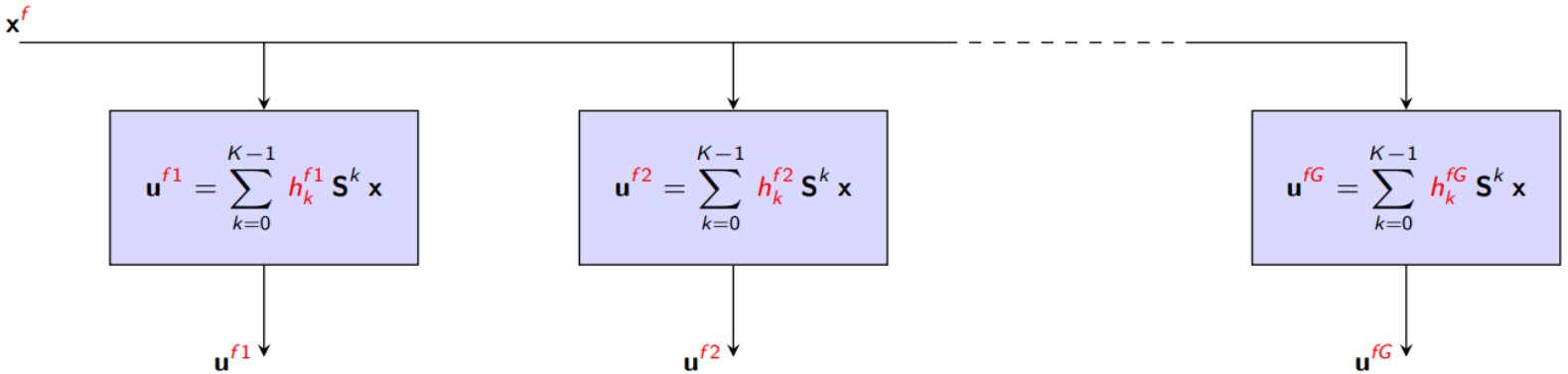


$$\mathbf{Z} = \begin{bmatrix} z_1^1 & \cdots & z_1^f & \cdots & z_1^F \\ \vdots & & \vdots & & \vdots \\ z_i^1 & \cdots & z_i^f & \cdots & z_i^F \\ \vdots & & \vdots & & \vdots \\ z_n^1 & \cdots & z_n^f & \cdots & z_n^F \end{bmatrix} = \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_i \\ \vdots \\ \mathbf{z}_n \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{z}^1 & \cdots & \mathbf{z}^f & \cdots & \mathbf{z}^F \end{bmatrix}$$

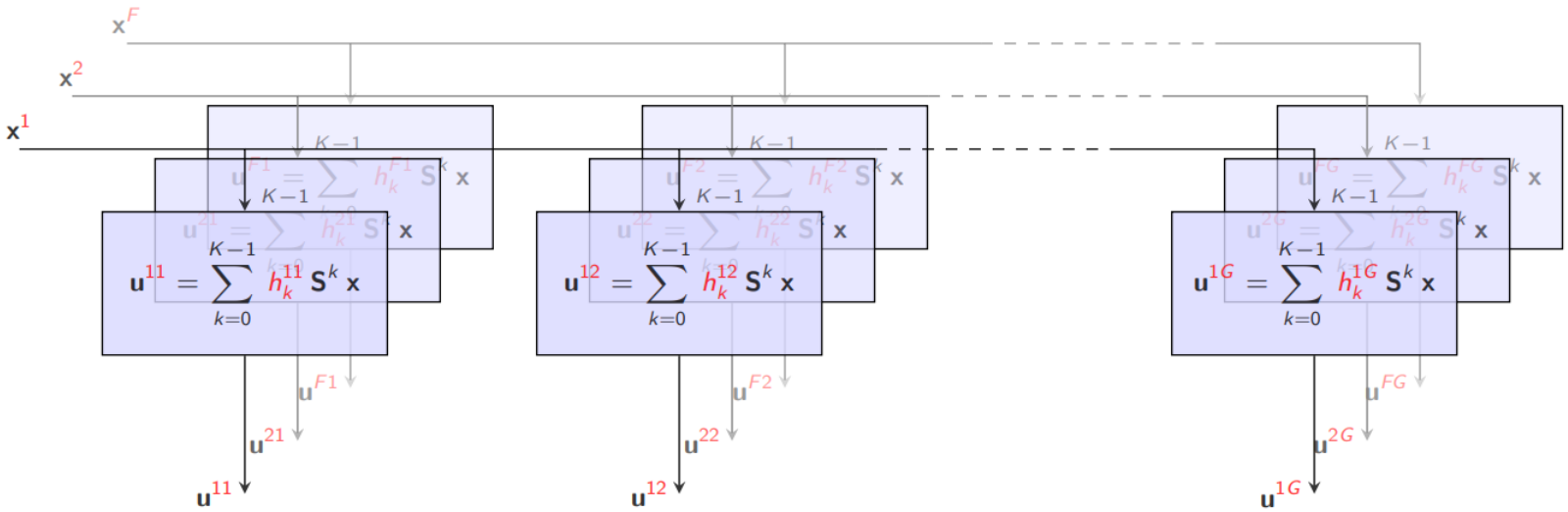
Multiple Feature GCNNs

- We leverage filter banks to create GCNNs that process multiple features per layer.
- Each of the F features $\mathbf{x}^f$ is processed with G filters with coefficients $h_k^{fg} \Longrightarrow \mathbf{u}^{fg} = \sum_{k=0}^{K-1} h_k^{fg} \mathbf{S}^k \mathbf{x}^{\mathbf{f}}$



Multiple Feature GCNNs (cont.)

- This Multiple-Input-Multiple-Output (MIMO) Graph Filter generates an output with $F \times G$ features.
- Reduce to G outputs with sum over input features for given $g \Rightarrow z^g = \sum_{f=1}^{f} \mathbf{u}^{fg} = \sum_{f=1}^{f} \sum_{k=0}^{K-1} h_k^{fg} \mathbf{S}^k \mathbf{x}^{\mathbf{f}}$



$\mathbf{z}^1 = \mathbf{u}^{11} + \mathbf{u}^{21} + \ldots + \mathbf{u}^{F1}$ $\quad$ $\mathbf{z}^2 = \mathbf{u}^{12} + \mathbf{u}^{22} + \ldots + \mathbf{u}^{F2}$ $\quad$ $\mathbf{z}^2 = \mathbf{u}^{1G} + \mathbf{u}^{2G} + \ldots + \mathbf{u}^{FG}$

Multiple Feature GCNNs (cont.)

- It is easier to MIMO graph filters with matrices i.e., $G \times F$ coefficient matrix $\mathbf{H}_k$ with entries $(\mathbf{H}_k)_{fg} = h_k^{fg}$.

$$\mathbf{Z} = \sum_{k=0}^{K-1} \mathbf{S}^k \times \mathbf{X} \times \mathbf{H}_k$$

- This is a more compact format of the MIMO filter.

Multiple Feature GCNNs (cont.)

- MIMO GNN stacks MIMO perceptrons. Composed of MIMO filters with activation functions.
- Layer l processes input signal $\mathbf{X_{l-1}}$ with the perceptron $\mathbf{H}_l = [\mathbf{H}_{l0}, \cdots, \mathbf{H}_{l,K-1}]$ to produce output $\mathbf{X_l}$.

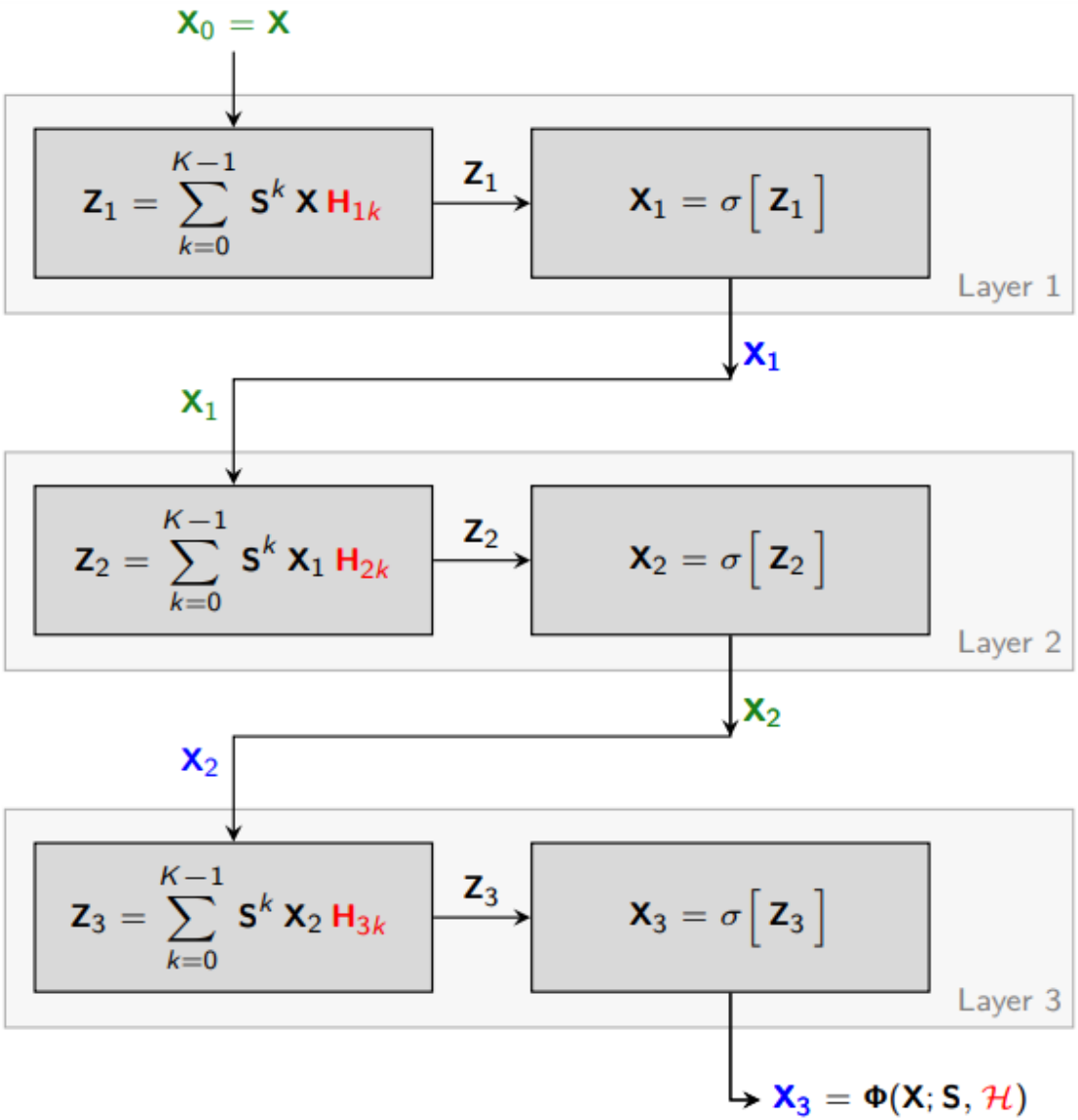
$$\mathbf{X_l} = \sigma\left[\mathbf{Z}_l\right] = \sigma\left[\sum_{k=0}^{K-1} \mathbf{S}^k \mathbf{X_{l-1}} \mathbf{H}_{lk}\right]$$

- With the convention that the Layer 1 input is $\mathbf{X}_0 = \mathbf{X}$, this provides a recursive definition of a MIMO GNN.
- If it has L layers, then the GNN output

$$X_L = \Phi(\mathbf{x}; \mathbf{S}, \mathbf{H}_1, \cdots, \mathbf{H}_L) = \Phi(\mathbf{x}; \mathbf{S}, \mathcal{H})$$

- The filter tensor $\mathcal{H} = [H_1, \cdots, H_L]$ is the trainable parameter. The graph shift operator $\mathbf{S}$ is prior information.

Multiple Feature GCNNs (cont.)



Summary

- Images can be represented as grids in space. Generalize convolutional filters to graphs.
- Graphs can be represented in form of matrices. Giving us graph Shift operators and making it easier to operate over them.
- Graph signal is a vector in which each component x_i is associated with node i.
- Graph signal, graph shift operator and graph filter make up the ingredients for the graph neural network.
- Graph signal is the input. Graph Shift operator is a parameter. We can also treat it as input if we want to consider different graphs. Graph filters are trainable parameters.
- A GCNN is composed of multiple graph perceptrons (graph filter + activation function).
- Individuals graph filters can be replaced with filter banks to learn multiple features.

References

These slides have been adapted from

- Alejandro Ribeiro, UPenn EE5140: Graph Neural Networks
- Jure Leskovec, Stanford CS224W: Machine Learning with Graphs
- Petar Veličković, Deepmind: Theoretical Foundations of Graph Neural Networks