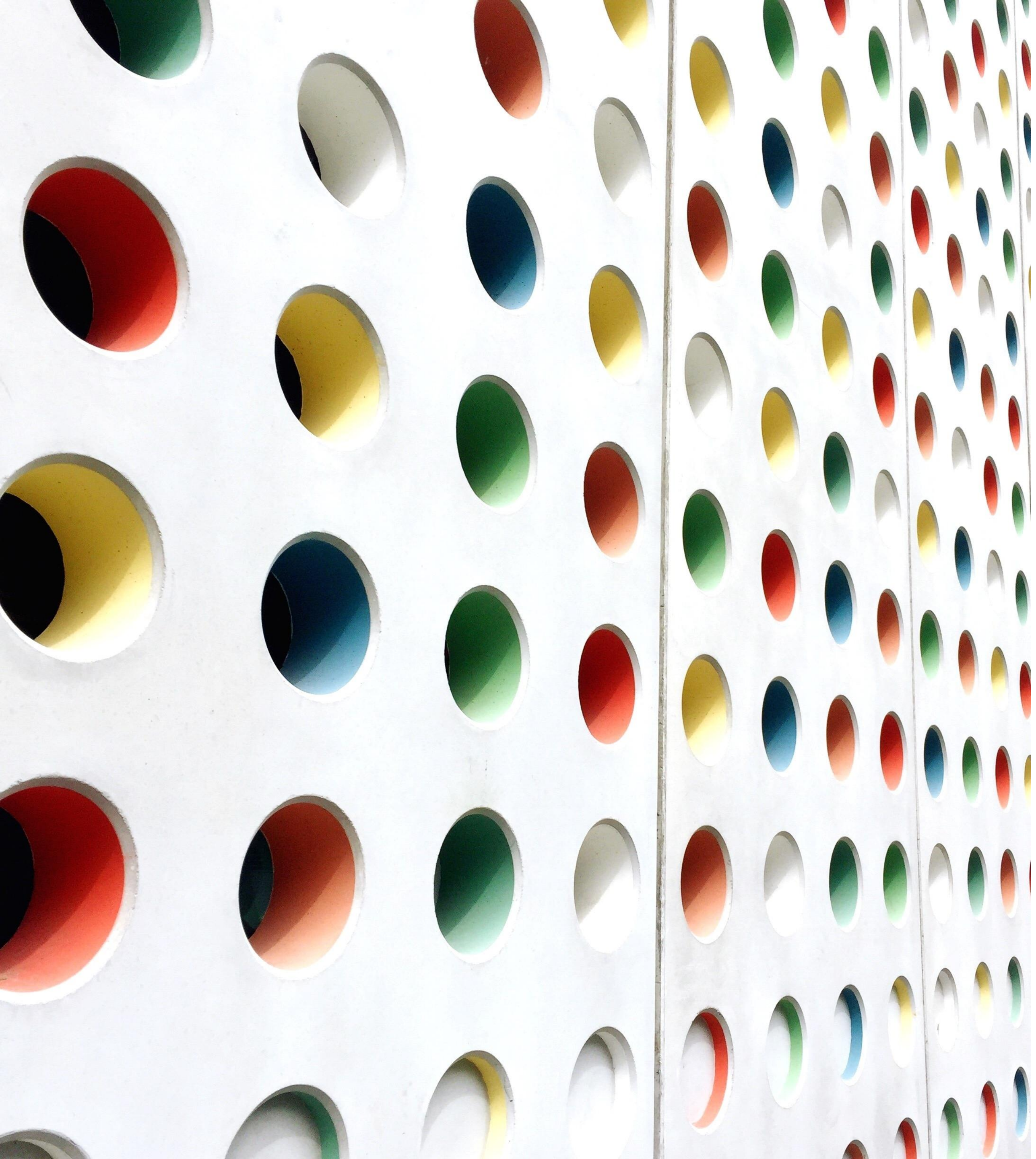


Graph Neural Networks:Part 1

MENAN VELAYUTHAN

NOTE: This was initially presented at ICAC 2022 by **Dharshana Kasthurirathna, PhD(USyd), Menan Velayuthan, Sanka Mohottala.** I have decided to use the same material for the Part 1 and then branch off to new material for Part 2 and Part 3.

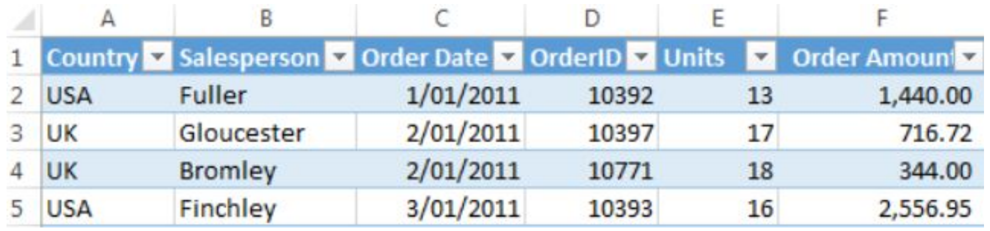
Link to the official github repo here.

Introduction

GRAPHS AND THEIR APPLICATIONS

Structured Data

Structured Data has a specific positional format to describe them.

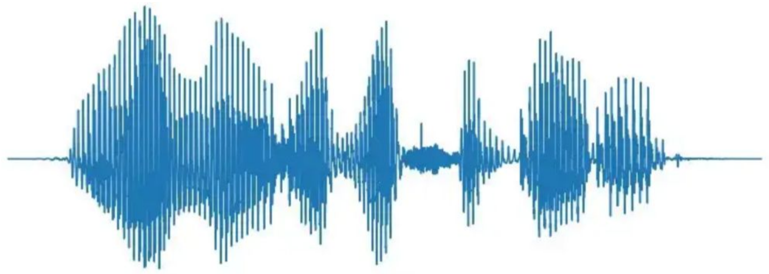


	A	B	C	D	E	F
1	Country	Salesperson	Order Date	OrderID	Units	Order Amount
2	USA	Fuller	1/01/2011	10392	13	1,440.00
3	UK	Gloucester	2/01/2011	10397	17	716.72
4	UK	Bromley	2/01/2011	10771	18	344.00
5	USA	Finchley	3/01/2011	10393	16	2,556.95

$\mathbb{N} \mapsto X$: Tabular data

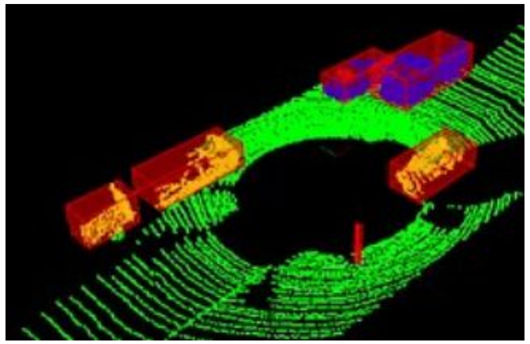
Unstructured Data

Unstructured data doesn't have a specific position format.



$\mathbb{R} \mapsto \mathbb{R}$: Signal , Speech etc.

$\mathbb{R}^2 \mapsto \mathbb{R}$: Images

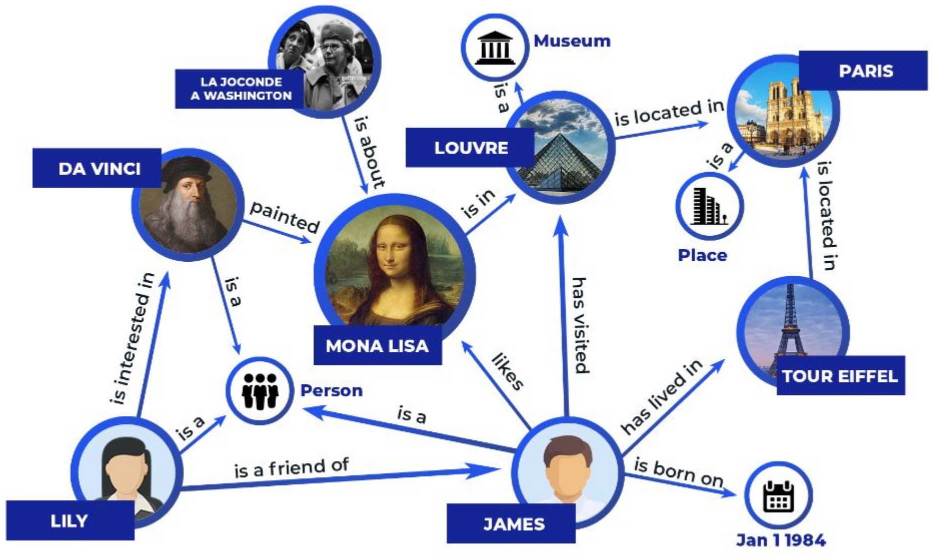


$\mathbb{R}^3 \mapsto \mathbb{R}^n$: Point clouds, Volumes

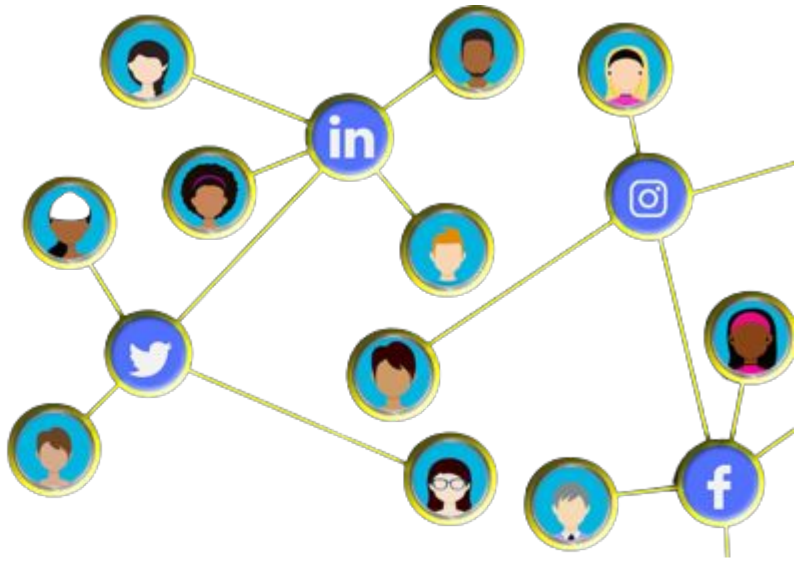
Surface Meshes

Unstructured Data

Graph data is one of the popular unstructured data types.



Knowledge Graph

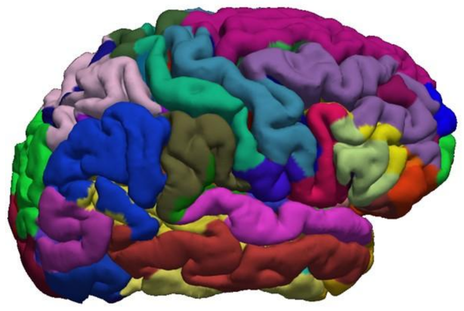


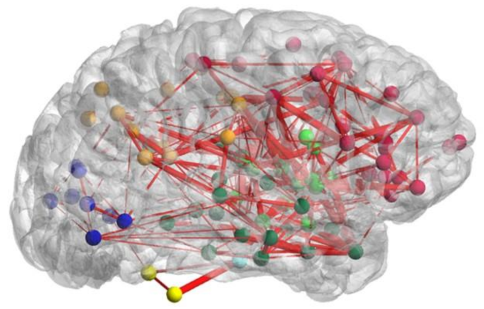
Social Networks

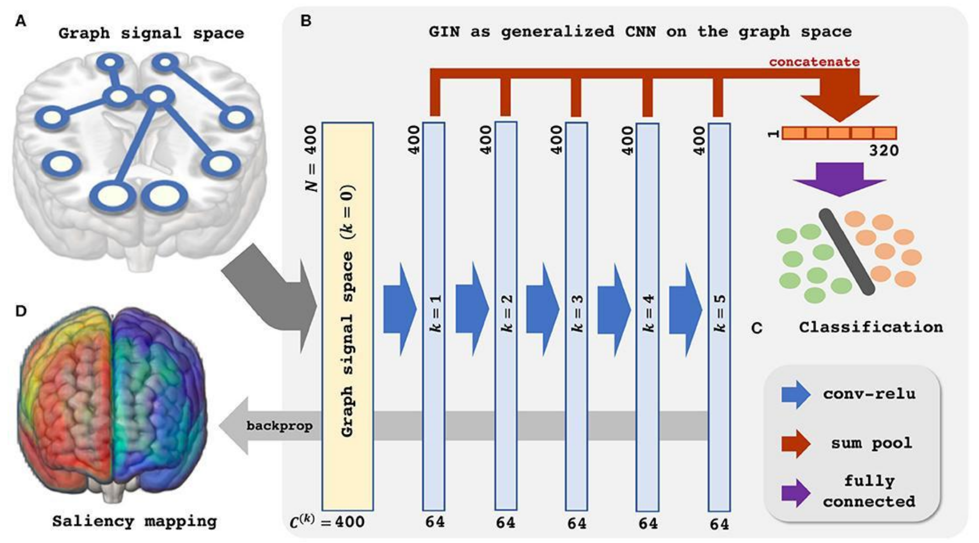
Graph neural network applications

Graphs can be used to model brain connectomes taken from fMRI/DTI data

GNN can be used for fMRI data analysis by modeling the functional connectivity of brain as a graph structure.



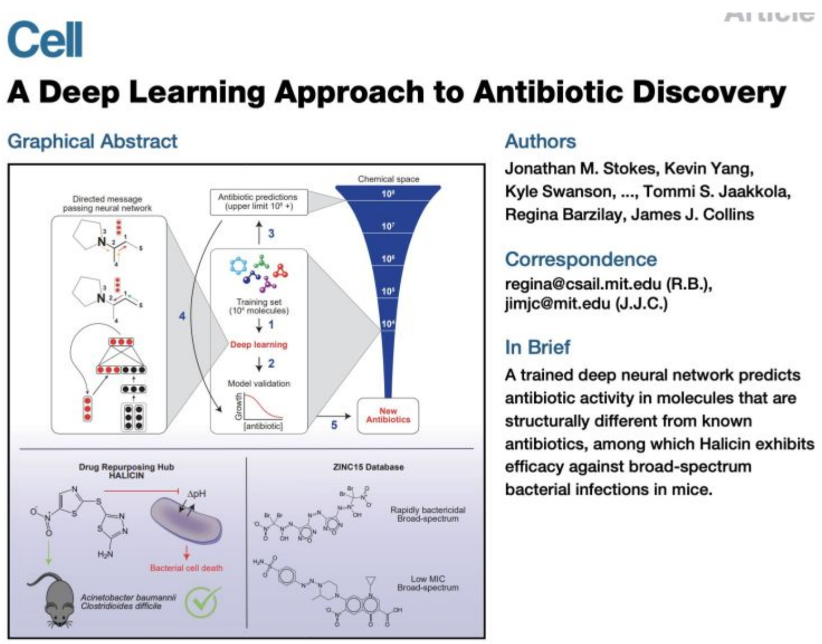




Graphs are also used in Chemistry and Pharmacology

Modeling molecules as graphs data, they can be used with graph neural networks to do

- Molecule Analysis
- Drug discovery
- Drug repurposing
- Catalysts discovery



Article

Cell

A Deep Learning Approach to Antibiotic Discovery

Graphical Abstract

Directed message passing neural network
Antibiotic predictions (upper limit 10^9 +)
Chemical space
Training set (10^4 molecules)
Deep learning
Model validation
New Antibiotics
Drug Repurposing Hub HALICIN
ZINC15 Database
Bacterial cell death
Acinetobacter baumannii Clostridioides difficile
Rapidly bactericidal Broad-spectrum
Low MIC Broad-spectrum

Authors

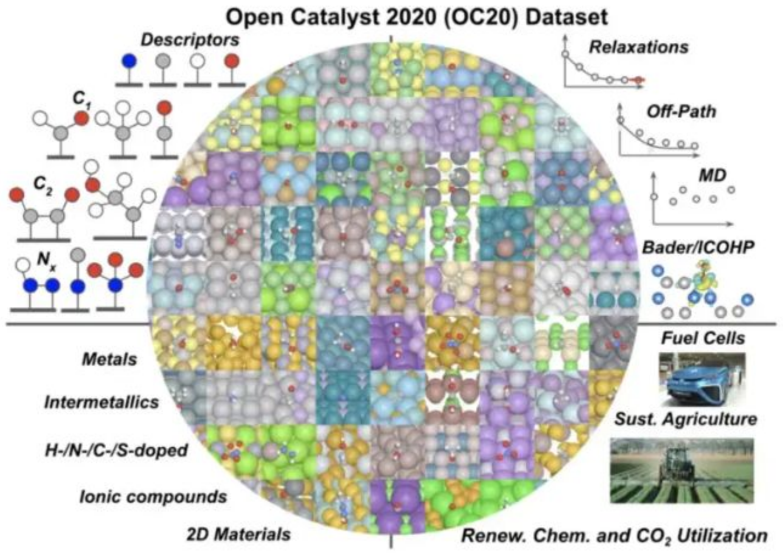
Jonathan M. Stokes, Kevin Yang, Kyle Swanson, ..., Tommi S. Jaakkola, Regina Barzilay, James J. Collins

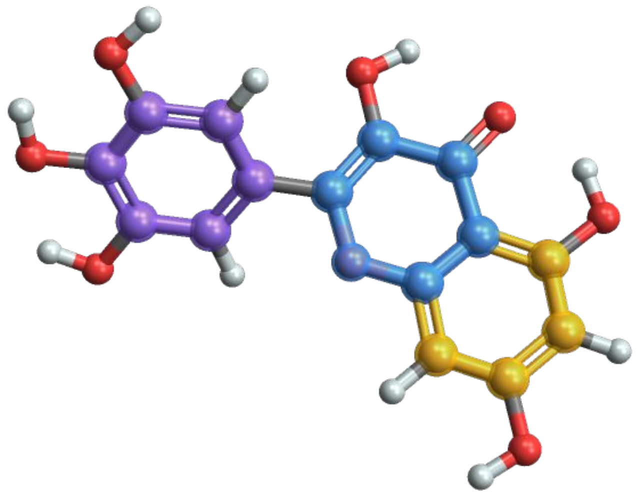
Correspondence

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

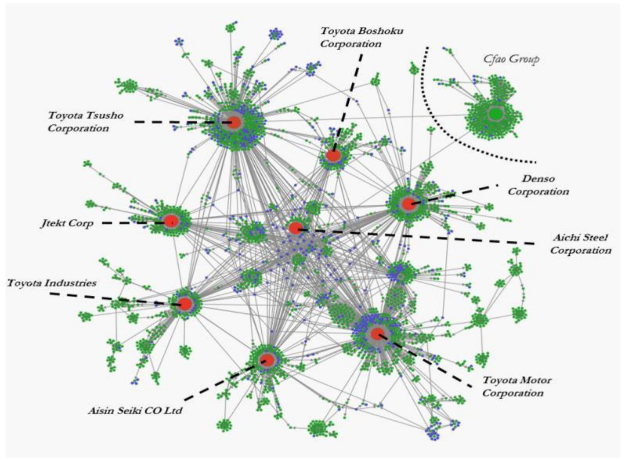
In Brief

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.

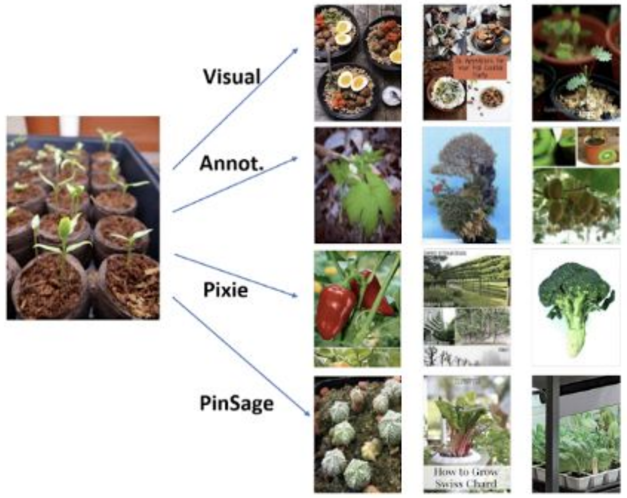


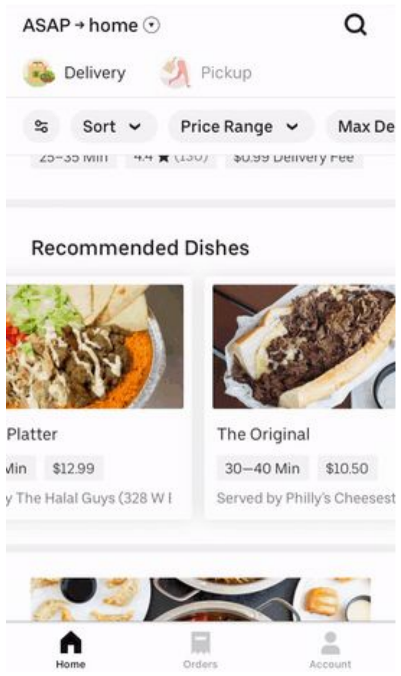


In financial sector, graphs can be used to model how the markets work as in the case of stock market prediction.



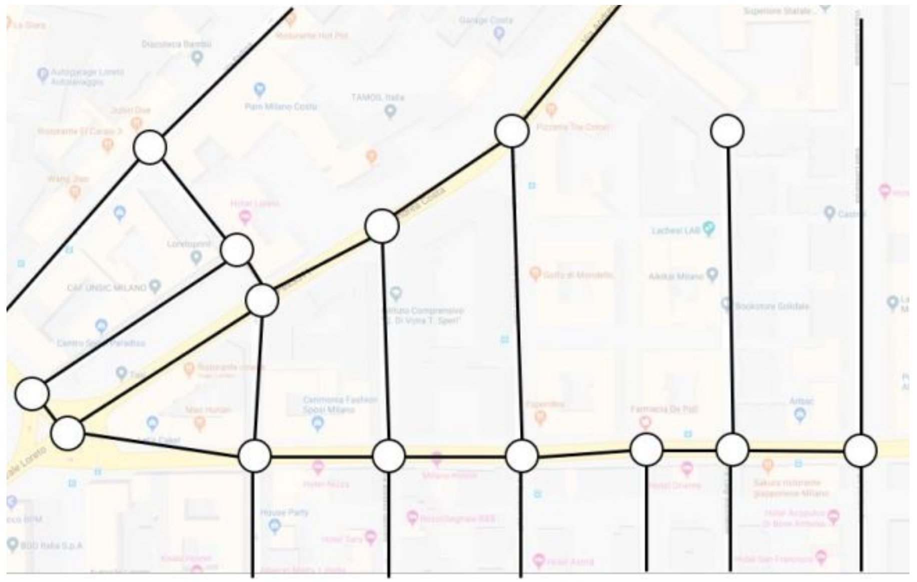
Recommender systems used in Uber Eats and Pinterest uses GNN algorithms.

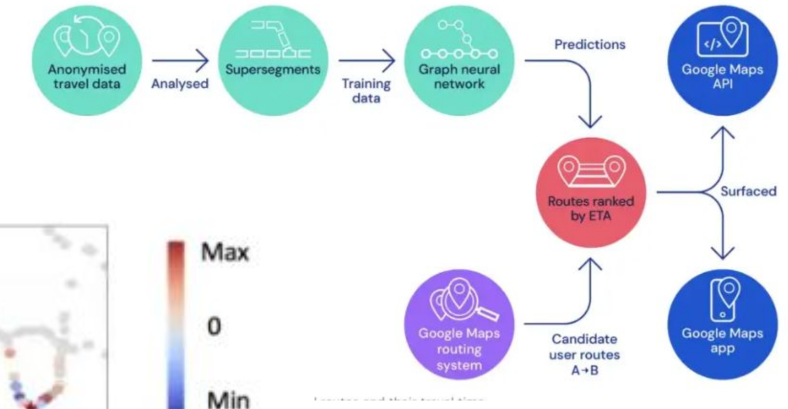


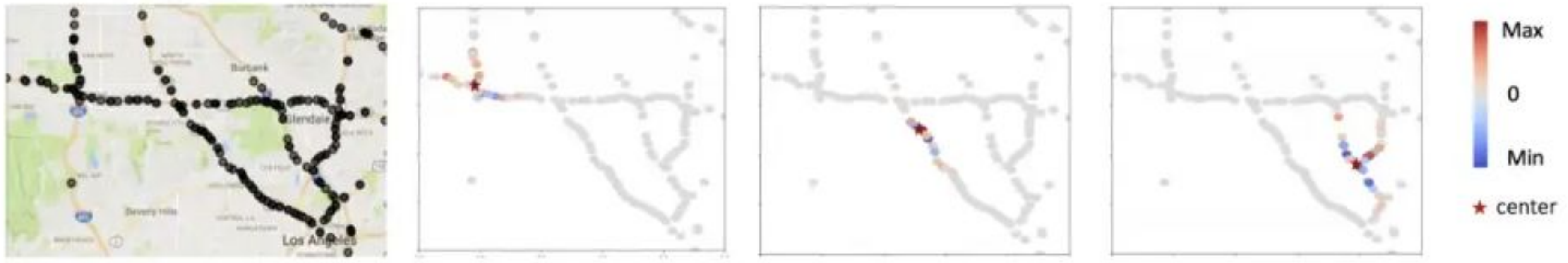


Road systems can also be modeled as graph. In such a system, intersections can be considered as nodes and roads as edges.

GNNs are being used with such data to estimate the time of arrival in Google maps.





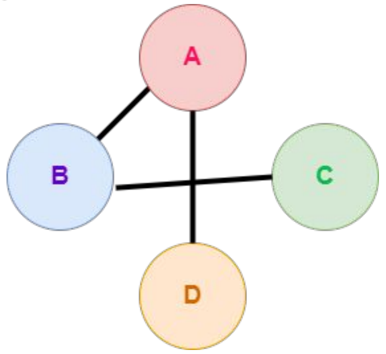


Let’s start from the Basics

GRAPHS AND BASIC DEFINITIONS

How do we define a graph?

A graph $\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{E})$ is mathematical object defined by a set of nodes ***V*** and set of edges ***E***. For example, lets take a look at the following graph.



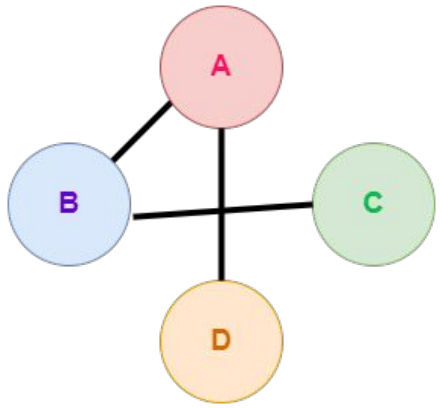
Here the graph ***G1 = (V1,E1)*** where,

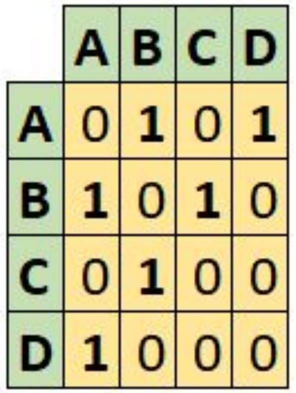
V1 = { A, B, C, D} & *E1 = {{A,B},{B,C},{D,A}}*

It important to note that this defines a special type for graph called undirected graphs, which are graphs having undirected edges, meaning edge A,B or B,A mean the same as the edges don’t have a direction.

Adjacency Matrix

- We can store the links between nodes in a compact manner using the *Graph Adjacency Matrix A*.
- Adjacency Matrix *A1* for the graph mentioned previous is said to have $\boldsymbol{a_{ij}} = \mathbf{1}$ if nodes i and j are connected else $\boldsymbol{a_{ij}} = \mathbf{0}$.
- An adjacency matrix may have weights on its edges, also known as **weighted adjacency matrix**.
- But for the sake of simplicity, we will assume our graph to be unweighted, undirected and homogenous graph(Graph containing single type of nodes and edges)





	A	B	C	D
A	0	1	0	1
B	1	0	1	0
C	0	1	0	0
D	1	0	0	0

Graph Neural Networks

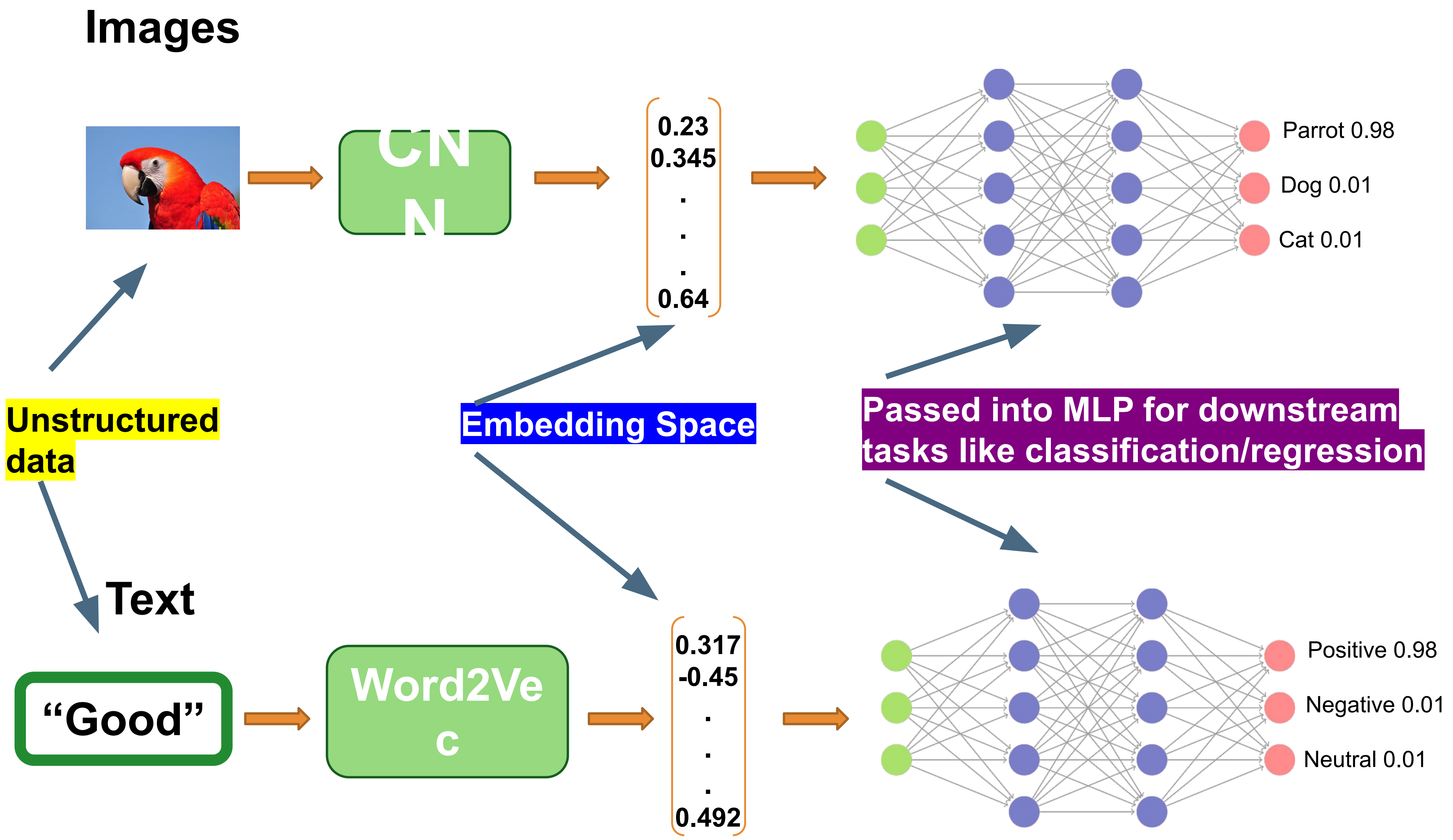
PART 1

Graph Machine Learning Task

- Based on the aspect of the graph we are looking at we can categorize the Graph ML task into 3 categories
  - **Node-level tasks :** Here we apply graph neural networks to learn to perform on node-level task such as fraud user detection, community detection etc..
  - **Edge-level tasks :** Here the model learns to perform on edge-level tasks like link prediction (whether two users can be friends in social networks) or fraudulent transaction detection.
  - **Graph-level tasks :** Here a model learns a representation on the entire graph and performs graph-level tasks like molecule property prediction, drug discovery etc..

Representational Learning

- One common denominator all the tasks in the previous slide have is finding representations for nodes or edges or the entire graphs.
- This of course isn’t new to machine learning, after all, machine learning is dominated by representational learning.
- Models such as Convolutional Neural Network(CNN), Word2Vec (Skipgram, CBOW models), transformers have been doing this under the hood.


Images
CNN
0.23
0.345
.
.
.
0.64
Parrot 0.98
Dog 0.01
Cat 0.01
Unstructured data
Embedding Space
Passed into MLP for downstream tasks like classification/regression
Text
"Good"
Word2Vec
0.317
-0.45
.
.
.
0.492
Positive 0.98
Negative 0.01
Neutral 0.01

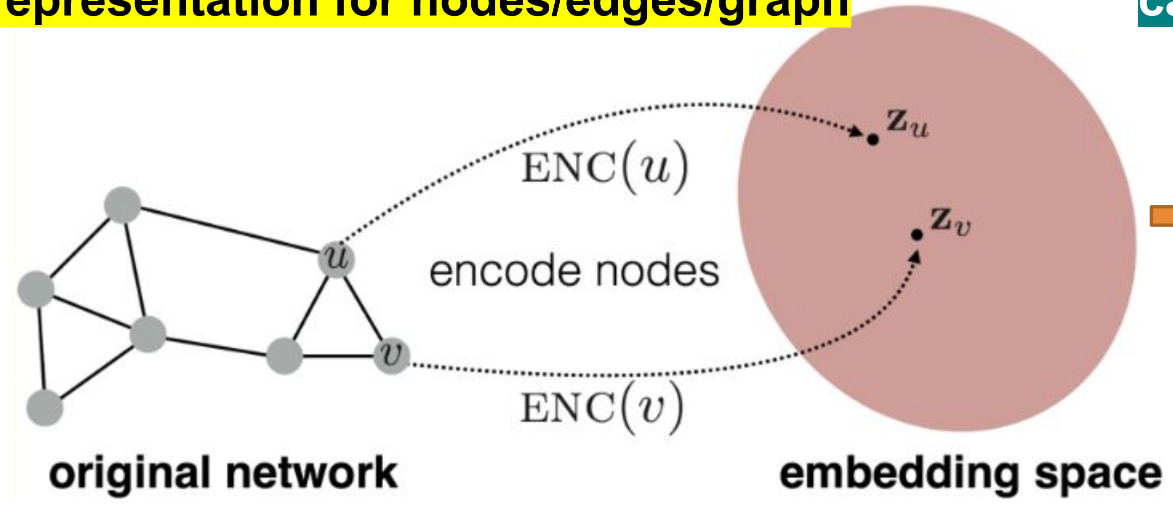
Same is required for Graph Machine Learning

- We plan to do the same for graphs as well, Graph Neural Networks(GNN) is a tool to achieve this.

We have a graph and we find a representation for nodes/edges/graph

Take the embedding, in this case a node embedding

Perform down stream task



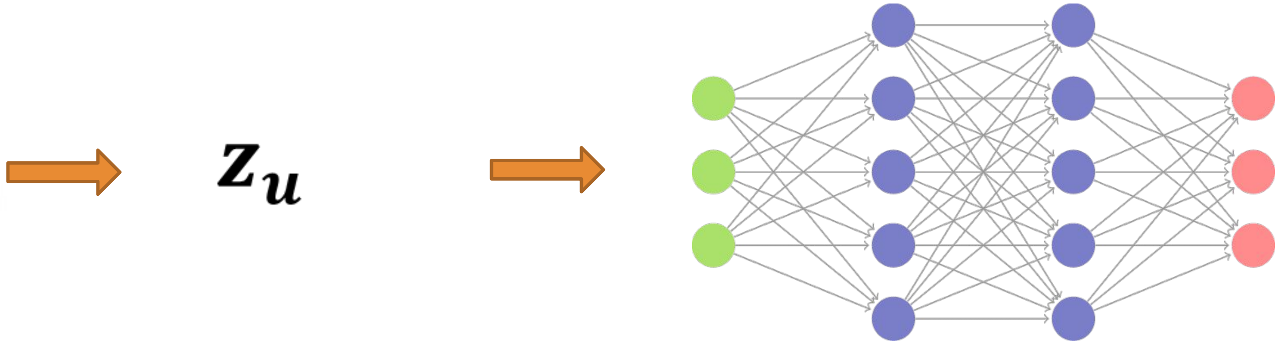


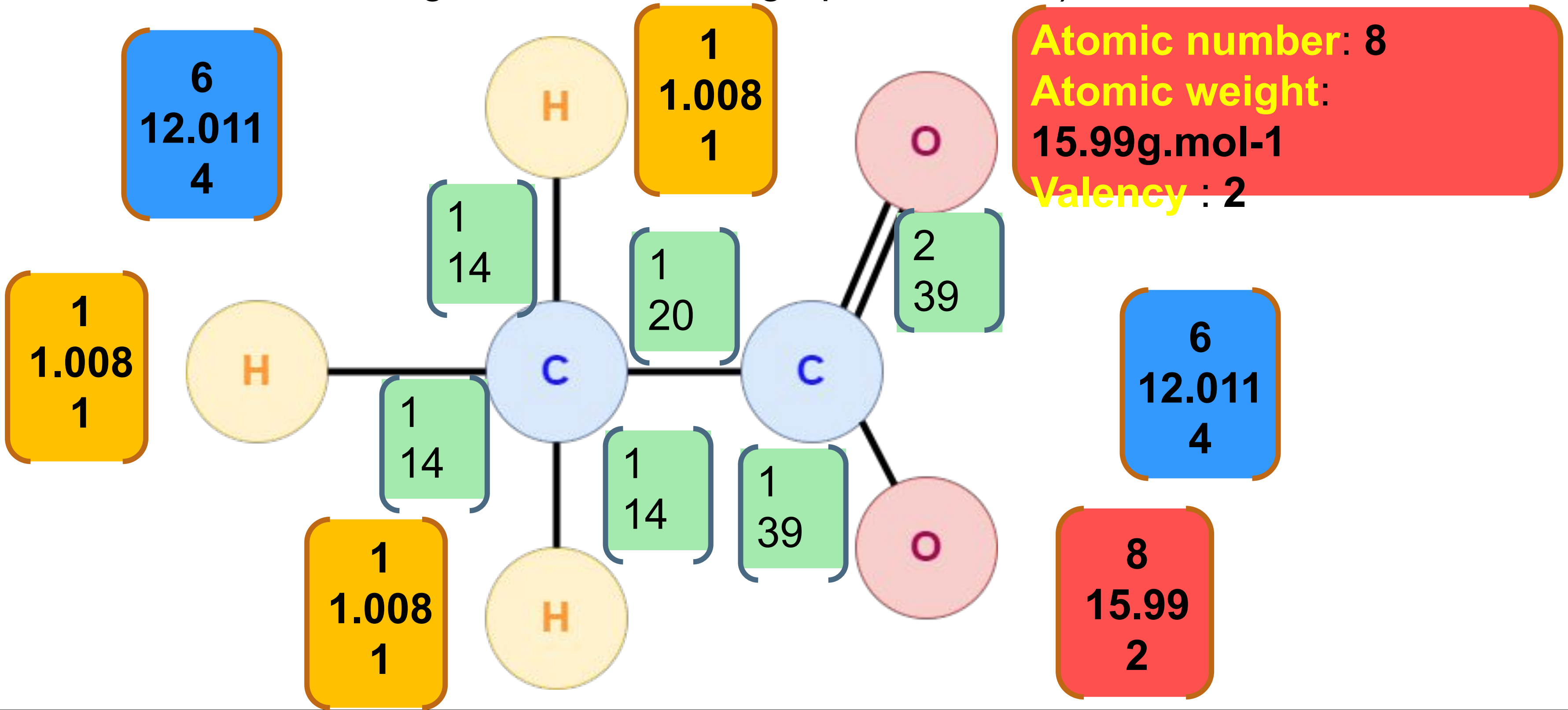
Image source :
https://snap-stanford.github.io/cs224w-notes/machine-learning-with-networks/node-representation-learning

Graphs may have features as well

- The graphs we come across in real-life have features present in nodes/edges/graphs or in all combination (ie : present in nodes and edges or nodes and graph level etc..).

Graph Level Properties
Organic: Yes
Flammable: Yes
Corrosive: Yes
Molecular Weight: 60.05

$$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 60.05 \end{pmatrix}$$



Graph Neural Networks

PART 2

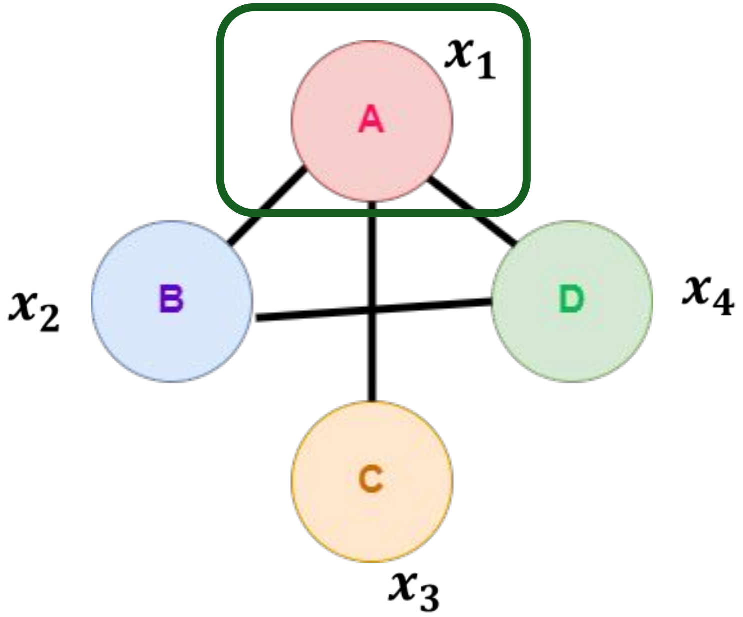
Graph Neural Networks

- Our method should utilize the bias of graph structures as well as the features present the graph to learn representations.
- Majority of the GNN models present in the industry follow the 3 building blocks given below,
  - **Message Passing**
  - **Aggregation**
  - **Update**
- Many variants of GNN have been introduced over time, they mainly differ in varying the approach towards one of the 3 mentioned above. Few examples of the variants are Graph Convolution Network (GCN), Graph Sage, Graph Attention Network etc..
- We will address the above-mentioned methodology one by one.
- Here we will take undirected graph with node features only for simplicity. This same method can be extended to tackle edge features or graph level features.

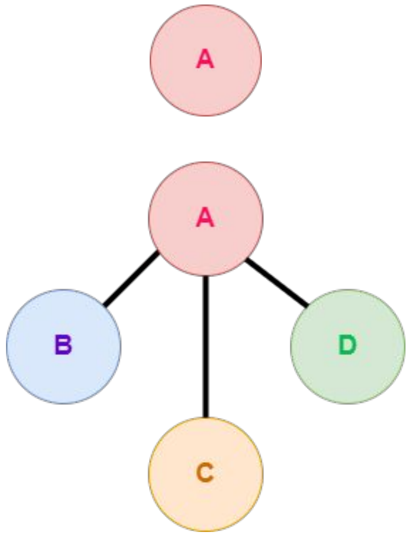
Message Passing

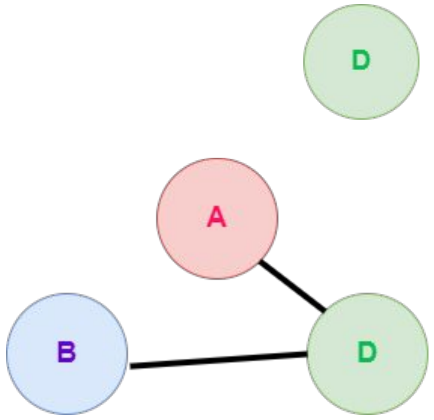
- Message Passing has been the key workforce of Graph Neural Networks, although there are work ongoing beyond Message Passing, it has proven to be effective.
- In English there is a famous saying, ***"A person is known by the company he keeps"***, message passing works similar to this. In a more technical term this called **Homophily,** which means the nodes connected to each other tend to have similar features and/or belong to similar classes.
- For our example of node classification task, our Message Passing layer will take a source node, identify its neighbours, transform the neighbours features and finally pass it to the source node. This is process is parallelly done to all the nodes in the graph. Hence at the end of this step all the nodes are updated.
- Now lets look at what is discussed using an example.

Message Passing Example

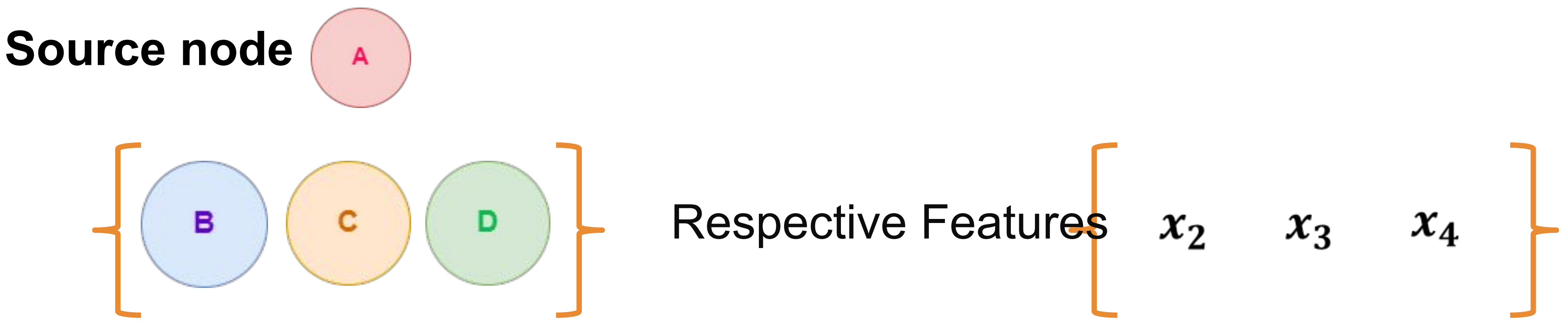


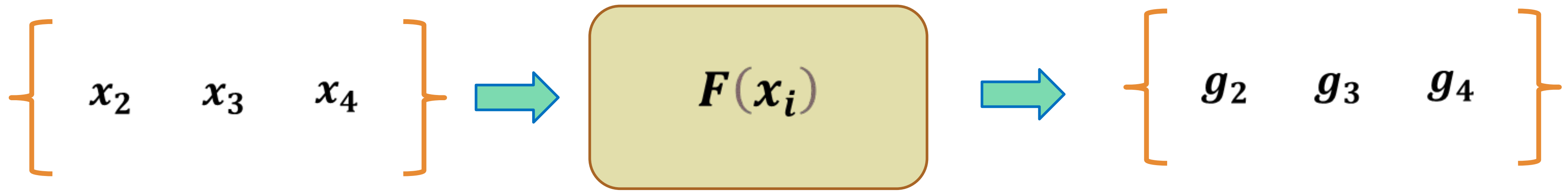
For example, lets identify the **1-hop neighbours** of the graph





Now let's take Node A as the source node and perform message passing

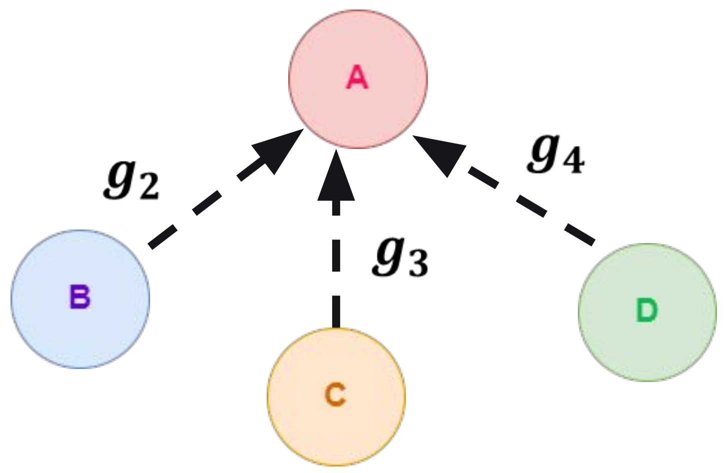




Here ***F*** can be MLP or RNN or simple affine function meaning simple linear tranformation will also work. Hence,

$$F(x_i) = Wx_i + b \; (here\ we\ perform\ linear\ transformation\ on\ x_i\ )$$

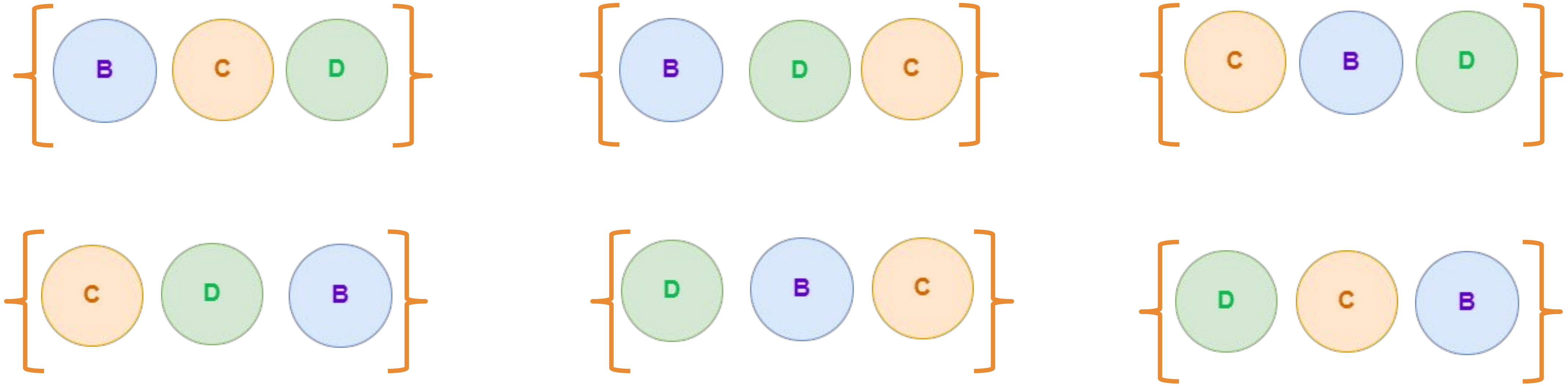
Finally, the transformed messages $(\boldsymbol{g_i})$ gets passed to the source node



Aggregate

- Now that the messages from the 1-hop neighbour nodes have been passed into the source node.
- We have to now find a way to combine these messages together (AGGREGATE).
- But here is the catch, our aggregate function should be "**Permutation Invariant**"**.** This a key bias we will introduce to the GNNs. The justification is, that the nieghbour nodes should be taken in any order and shouldn't affect the AGGREGATION.

Meaning, independent of the order we take the neighbours of the source node, the AGGREGATE function should produce the same results (Permutation Invariant)



Nothing tells the first order of nodes is better the other combination of node order, this is the reason for Permutation Invariant Aggregation function selection.

Some Famous Permutation Invariant Aggregation Functions

1. Summation ->1+2+3=3+2+1=2+1+3=6
2. Mean -> (1+2+3)/3=(3+2+1)/3=(2+1+3)=6/3=2
3. Min -> min(1,2,3)=min(3,2,1)=min(2,1,3) = 1
4. Max -> max(1,2,3)=max(3,2,1)=max(2,1,3) = 3

Hence let's define permutation invariant function AGG

$$m_i = AGG(g_j; j \in N_i)$$

m_i - Combined message of neighbours of I

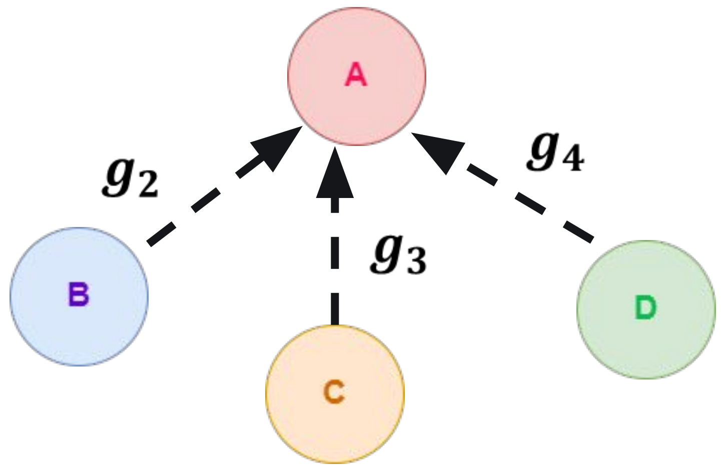
AGG – Can be any permutation invariant function (sum,mean,min ,max etc)

g_j - Transformed features of the neighbour nodes

N_i - Neigbourhood nodes of node i

To Recap

We perform message passing by transforming the features first.



Then we aggregate the messages by using a permutation invariant function

$$m_i = AGG(g_j; j \in N_i)$$

Update

- Finally, we need update the feature of the source node.
- Here we need to utilize the incoming aggregated message as well for the update.

Step 1

- We will first transform the feature of the source node using MLP or an affine function.

$$\boldsymbol{G(x_i)}$$

Step 2

- Now we need to combine the transformed feature with $\boldsymbol{m_i}$.
- To combine $\boldsymbol{m_i}$ and $\boldsymbol{G(x_i)}$ normally in literature we use,
    1. Summation or Averaging
    2. Concatenation.

To elaborate

We can combine $\boldsymbol{m_i}$ and $\boldsymbol{G(x_i)}$ in the following way

$$(G(x_i) + m_i)\ OR\ (G(x_i)\ ||\ m_i),$$

Here "||" stand for concatenation

NOTE:

- Few items to note, if summation is your choice of combining $\boldsymbol{m_i}$ and $\boldsymbol{G(x_i)}$ then dimensions of $\boldsymbol{m_i}$ and $\boldsymbol{G(x_i)}$ should agree.
- Normally we perform concatenation as the choice of aggregation in the hidden layers and summation in the final layer.

Step 3

- Now that we have combined $\boldsymbol{m_i}$ and $\boldsymbol{G(x_i)}$ we will add the usual Neural Networks.
- By that, we will pass the combination of $\boldsymbol{m_i}$ and $\boldsymbol{G(x_i)}$ into a MLP and then introduce a non-linearity to it.

$$\boldsymbol{h_i = \sigma(H(G(x_i), m_i))}$$

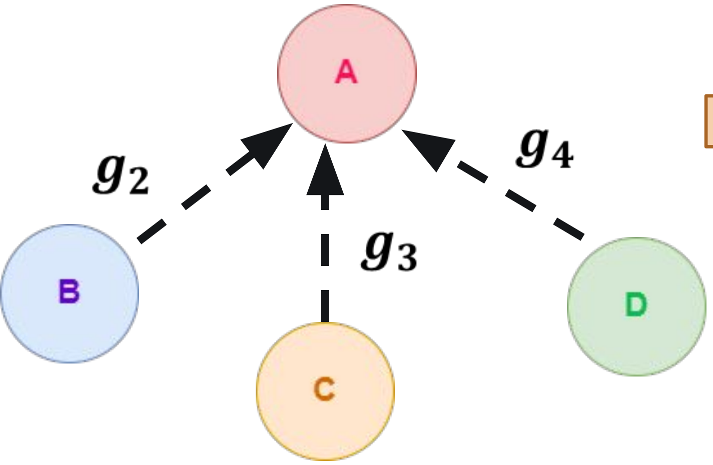
Where,

H = MLP

$\boldsymbol{\sigma}$ = Non-linearity (eg:sigmoid, ReLu, Tanh etc..)

To sum up the operations in a GNN layer

Message Passing | **Aggregate** | **Update**

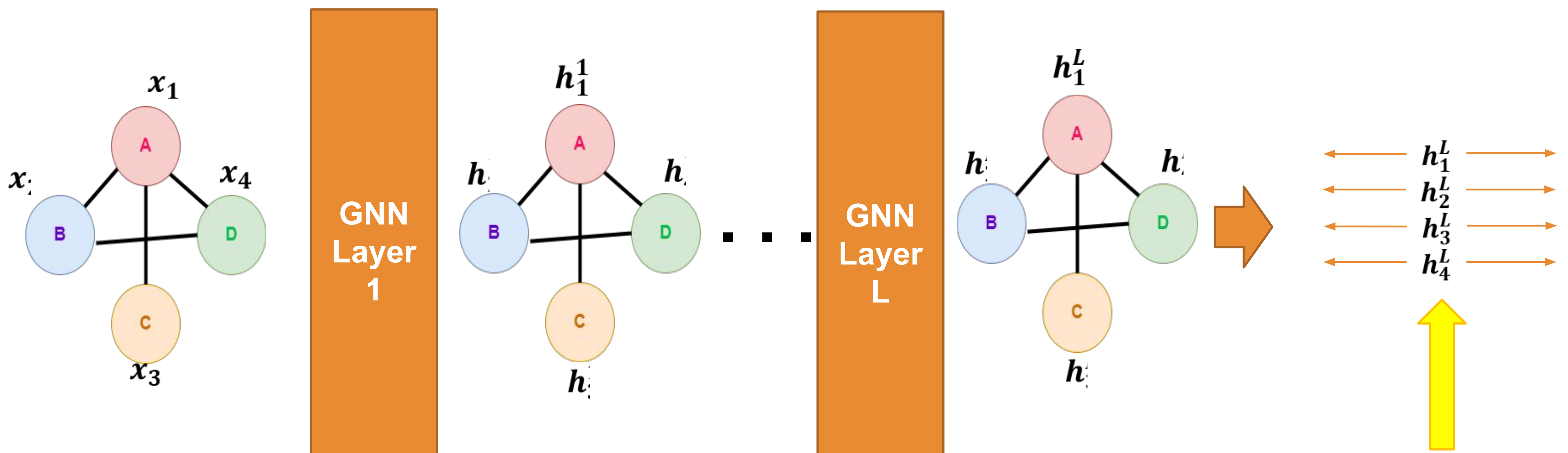


$$m_i = AGG(h_j; j \in N_i) \;\Rightarrow\; h_i = \sigma(H(G(x_i), m_i))$$

Well, it's done, one layer of GNN has been formulated!!

Note:
All these steps are performed on all the nodes in the graph simultaneously.

Stacking GNN layers together



Final learned representation for the nodes A,B,C,D. This can be used for the downstream tasks.

Where h_i^l represents the embedding h of node i in the l^{th} layer.

$i = 1,2,3,4$

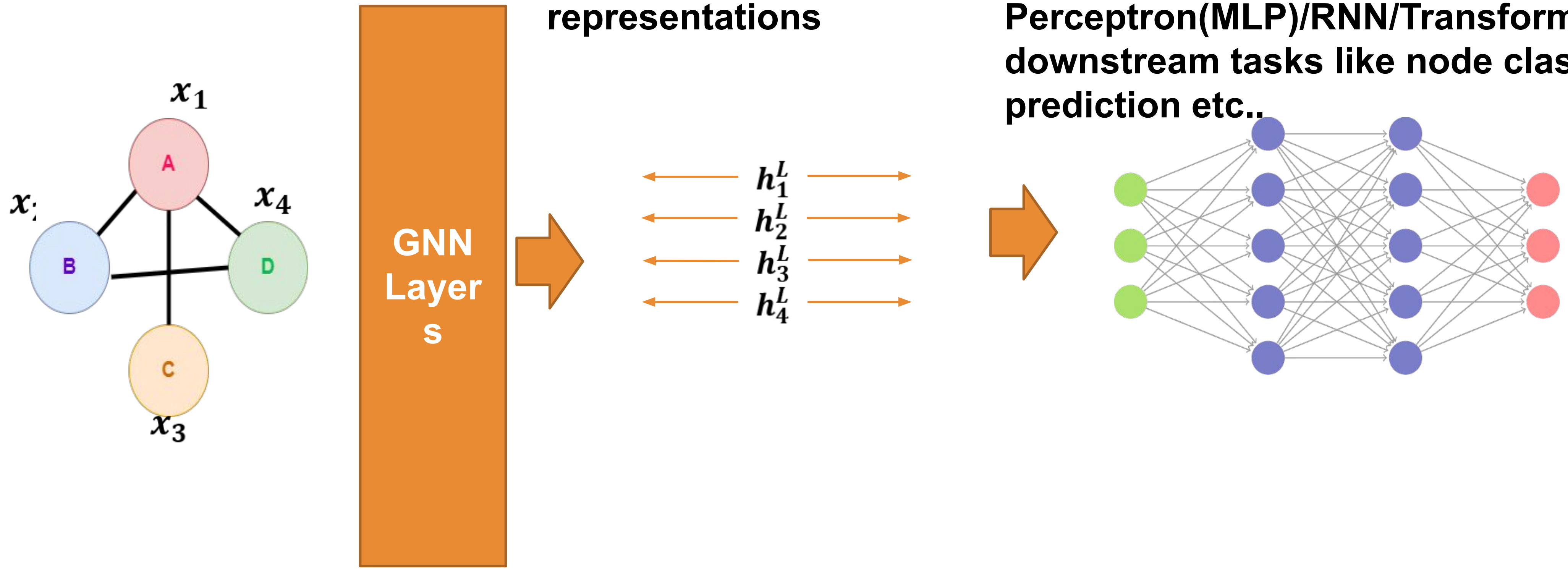
$l = 1,2,3...L$

Graph Neural Networks as an End-to-End Solution

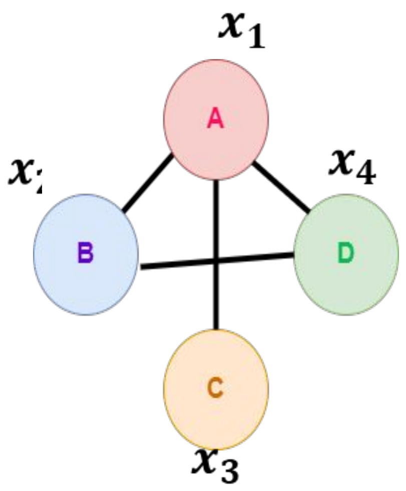
Input Graph

Learned representations

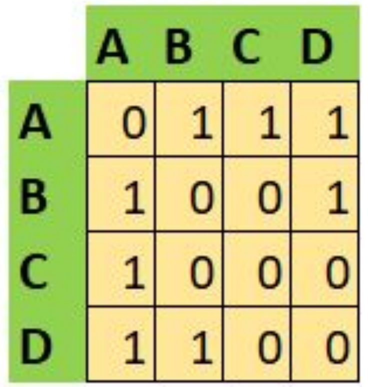
Multilayer Perceptron(MLP)/RNN/Transformer/Autoencoders fo
downstream tasks like node classification, link prediction etc..



Achieving GNN layer operation using Adjacency Matrix (a simple approach)



Graph G



	A	B	C	D
A	0	1	1	1
B	1	0	0	1
C	1	0	0	0
D	1	1	0	0

Adjacency matric for graph G

References

1. W. L. Hamilton, Graph Representation Learning Textbook.
(freely available : https://www.cs.mcgill.ca/~wlh/grl_book/)
2. CS224: Machine Learning with Graphs 2021, Lecture Notes (link: http://web.stanford.edu/class/cs224w/)
3. Petar Velickovic, Theoretical Foundations of Graph Neural Networks
(link: https://www.youtube.com/watch?v=uF53xsT7mjc&t=2113s)
4. Deep Graph Library Tutorials (link: https://docs.dgl.ai/guide/index.html)
5. PyTorch Geometric Tutorials
(link: https://pytorch-geometric.readthedocs.io/en/latest/notes/colabs.html)
6. Aleksa Gordic, GNN research paper explanations
(link: https://www.youtube.com/playlist?list=PLBoQnSflObckArGNhOcNg7IQG_f0ZlHF5)
7. Learning on Graphs Conference, GNN oriented conference (link: https://logconference.org/)
8. LoGaG: Learning on Graphs and Geometry Reading Group (link:
https://hannes-stark.com/logag-reading-group)