1. Basics of deep learning ✓

2. Deep learning for graphs

3. Graph Convolutional Networks

4. GNNs subsume CNNs and Transformers

Content

- **Local network neighborhoods:**
  - Describe aggregation strategies
  - Define computation graphs

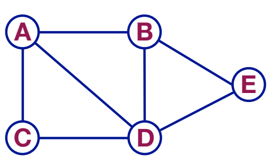
- **Stacking multiple layers:**
  - Describe the model, parameters, training
  - How to fit the model?
  - Simple example for unsupervised and supervised training

Setup

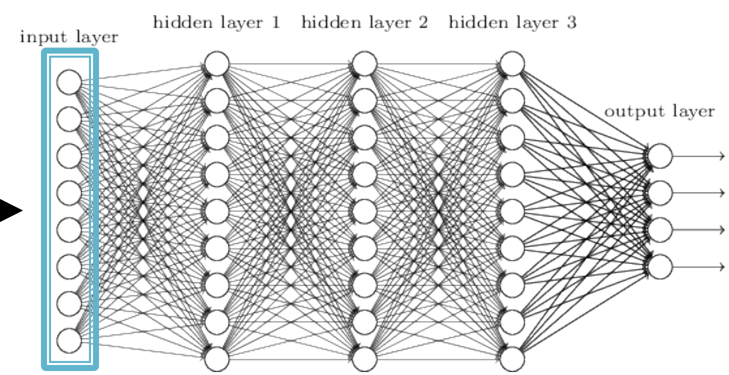
- **Assume we have a graph G:**
  - V is the **vertex set**
  - $\boldsymbol{A}$ is the **adjacency matrix** (assume binary)
  - $\boldsymbol{X} \in \mathbb{R}^{m\times|V|}$ is a matrix of **node features**
  - v: a node in V; $N(v)$: the set of neighbors of v.
  - **Node features:**
    - Social networks: User profile, User image
    - Biological networks: Gene expression profiles, gene functional information
    - When there is no node feature in the graph dataset:
      - Indicator vectors (one-hot encoding of a node)
      - Vector of constant 1: [1, 1, …, 1]

A Naïve Approach

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



	A	B	C	D	E	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

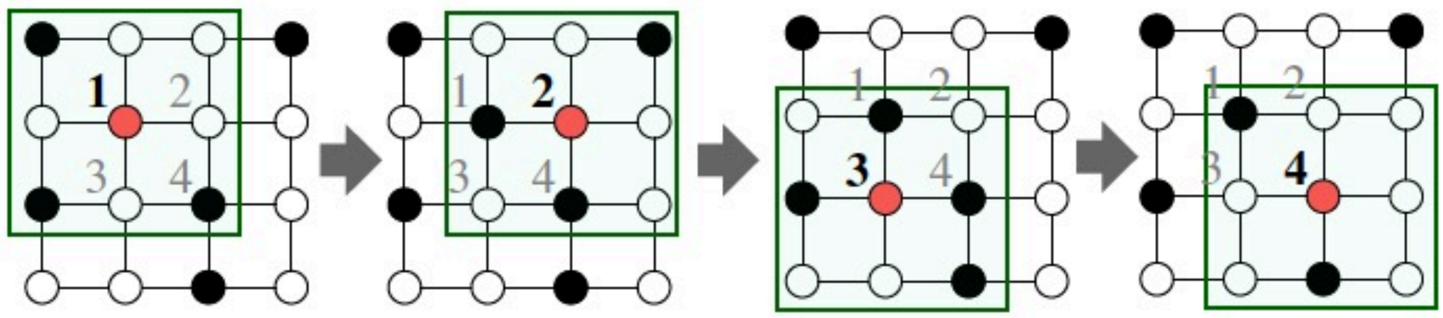


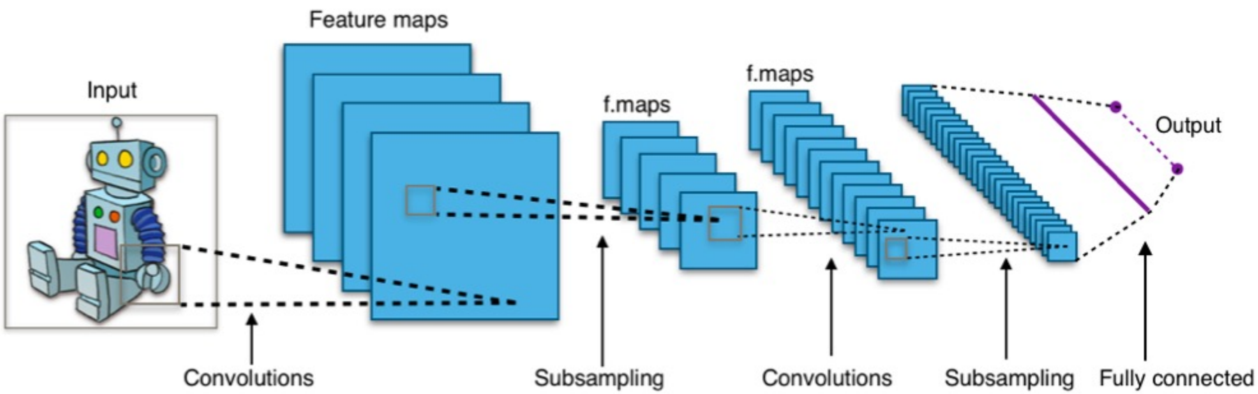
?

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

Idea: Convolutional Networks

CNN on an image:

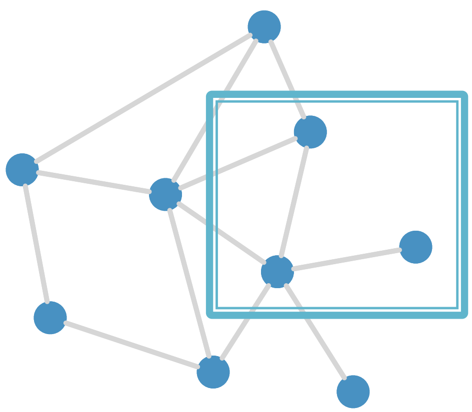




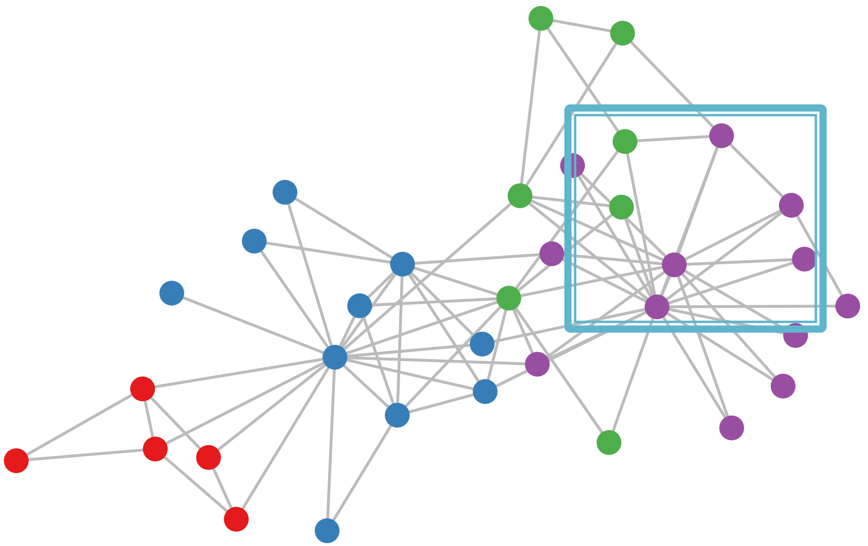
Goal is to generalize convolutions beyond simple lattices
Leverage node features/attributes (e.g., text, images)

Real-World Graphs

But our graphs look like this:



or this:



- There is no fixed notion of locality or sliding window on the graph
- Graph is permutation invariant

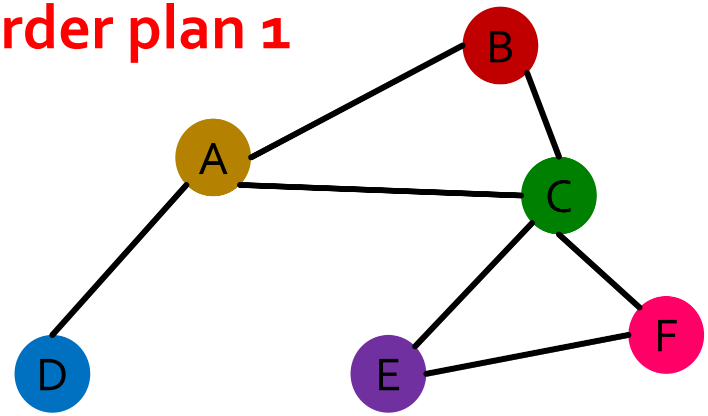
Permutation Invariance

- **Graph does not have a canonical order of the nodes!**
- We can have many different order plans.

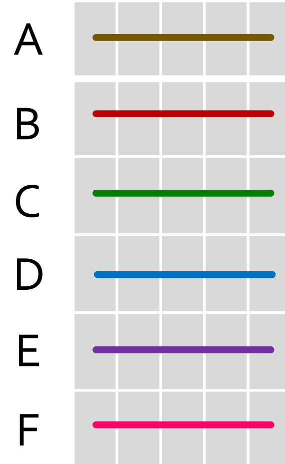
Permutation Invariance

- **Graph does not have a canonical order of the nodes!**

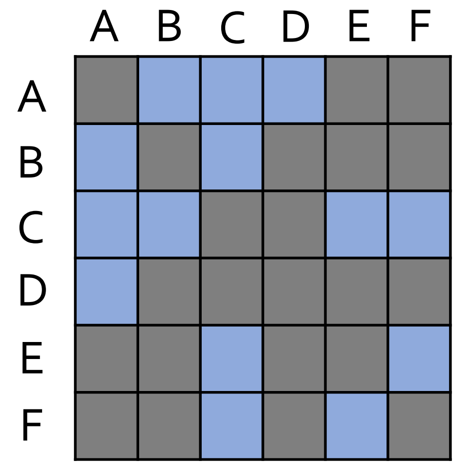
Order plan 1



Node features X_1



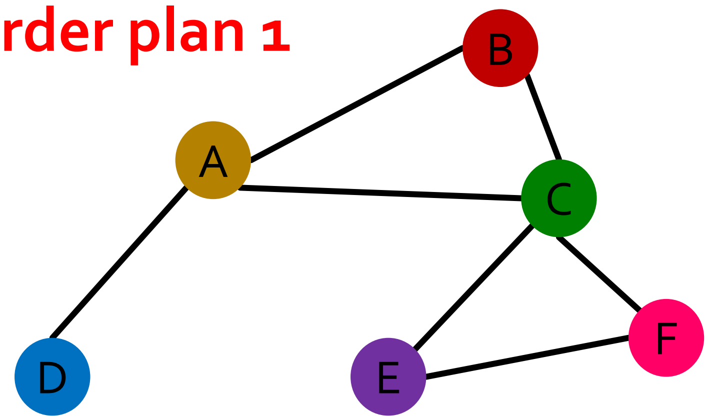
Adjacency matrix A_1



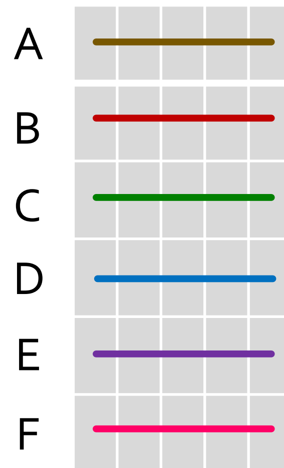
Permutation Invariance

- **Graph does not have a canonical order of the nodes!**

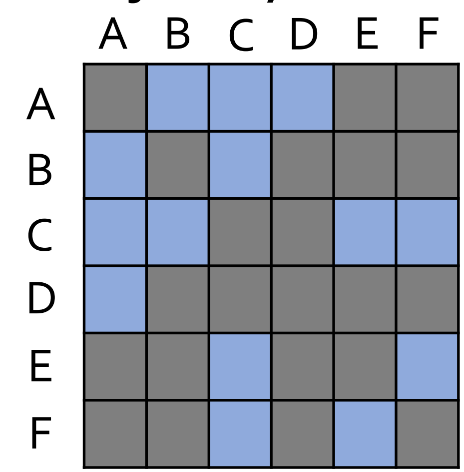
Order plan 1



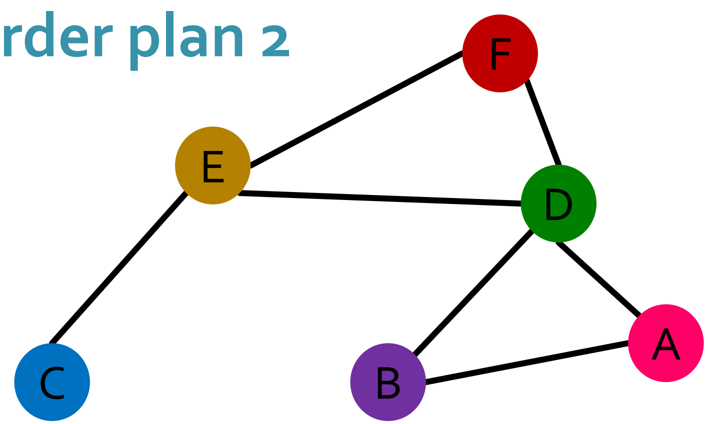
Node features X_1



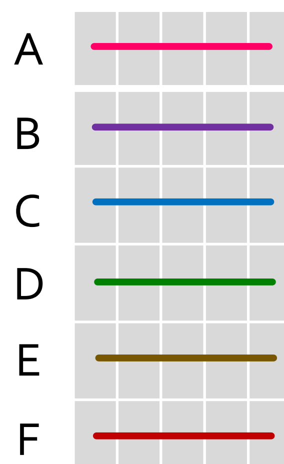
Adjacency matrix A_1



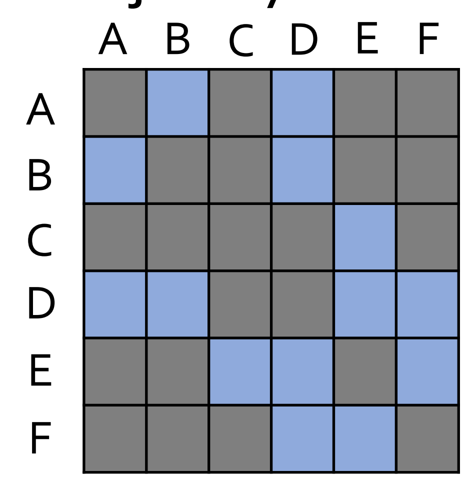
Order plan 2



Node features X_2



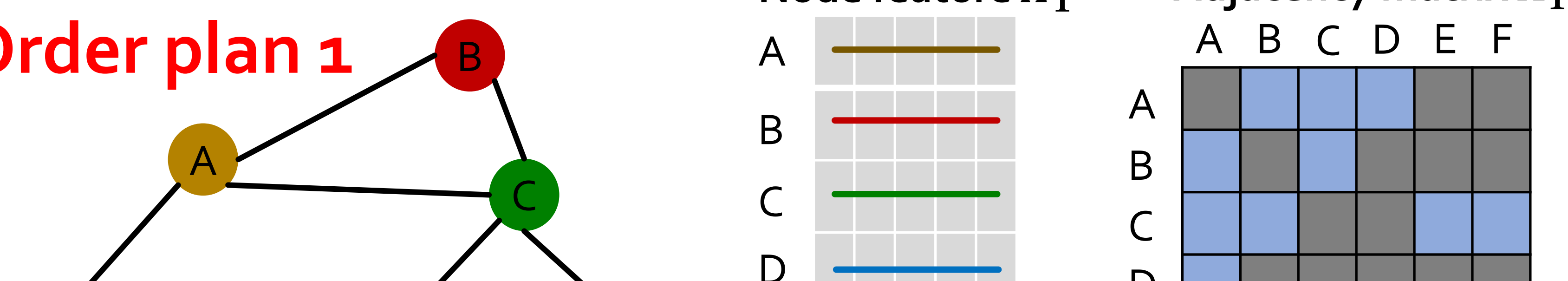
Adjacency matrix A_2



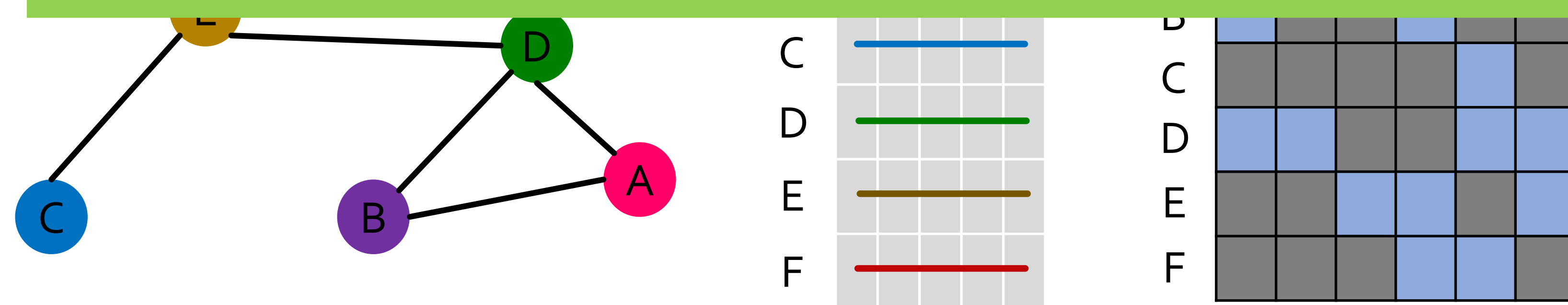
Permutation Invariance

- **Graph does not have a canonical order of the nodes!**

Order plan 1



Graph and node representations should be the same for Order plan 1 and Order plan 2



Permutation Invariance

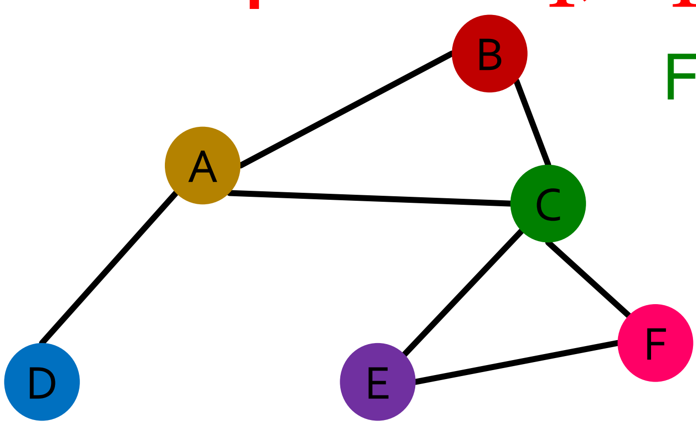
What does it mean by “graph representation is same for two order plans”?

- Consider we learn a function f that maps a graph $G = (\boldsymbol{A}, \boldsymbol{X})$ to a vector $\mathbb{R}^d$ then

$$f(\boldsymbol{A}_1, \boldsymbol{X}_1) = f(\boldsymbol{A}_2, \boldsymbol{X}_2)$$

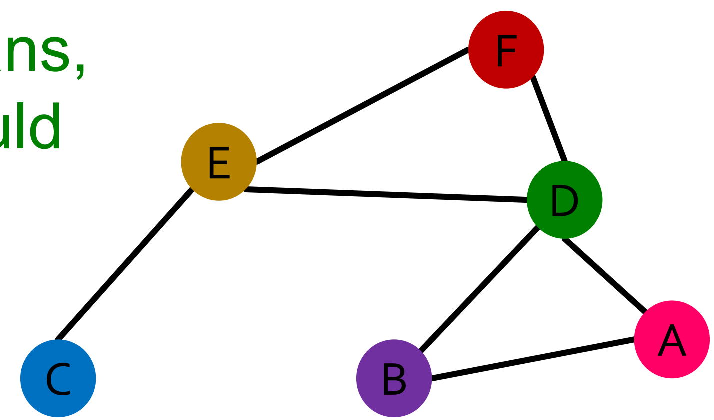
$\boldsymbol{A}$ is the adjacency matrix
$\boldsymbol{X}$ is the node feature matrix

Order plan 1: $\boldsymbol{A}_1, \boldsymbol{X}_1$



For two order plans, output of f should be the same!

Order plan 2: $\boldsymbol{A}_2, \boldsymbol{X}_2$



Permutation Invariance

What does it mean by "graph representation is same for two order plans"?

- Consider we learn a function f that maps a graph $G = (\boldsymbol{A}, \boldsymbol{X})$ to a vector $\mathbb{R}^d$.
- Then, if $f(\boldsymbol{A}_i, \boldsymbol{X}_i) = f\left(\boldsymbol{A}_j, \boldsymbol{X}_j\right)$ for any order plan i and j, we formally say f is a **permutation invariant function**.

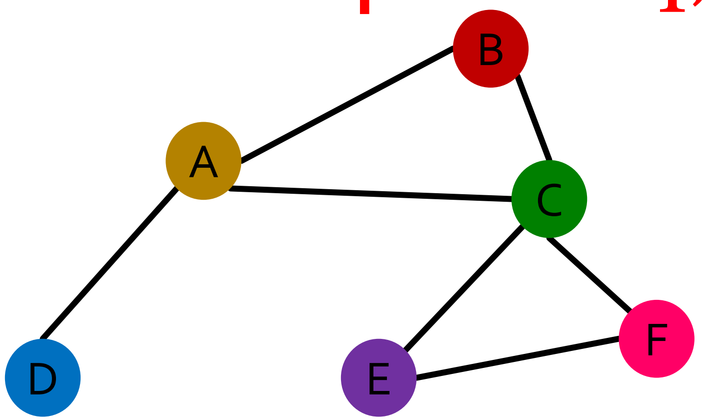
$\boldsymbol{A}$ is the adjacency matrix
$\boldsymbol{X}$ is the node feature matrix

For a graph with m nodes, there are $m!$ different order plans.

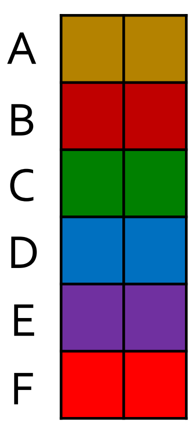
Permutation Equivariance

Similarly for node representation: We learn a function f that maps nodes of G to a matrix $\mathbb{R}^{m\times d}$.

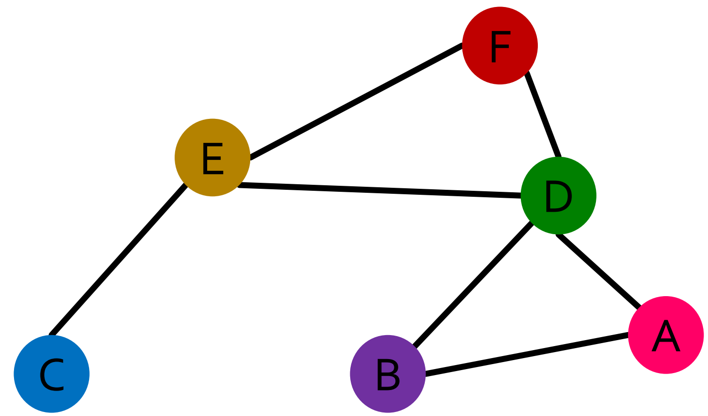
Order plan 1: $\boldsymbol{A}_1, \boldsymbol{X}_1$



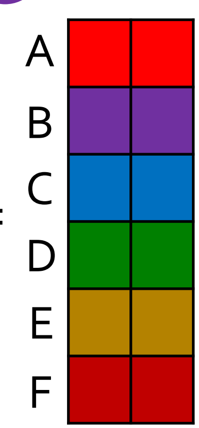
$f(\boldsymbol{A}_1, \boldsymbol{X}_1) =$



Order plan 2: $\boldsymbol{A}_2, \boldsymbol{X}_2$



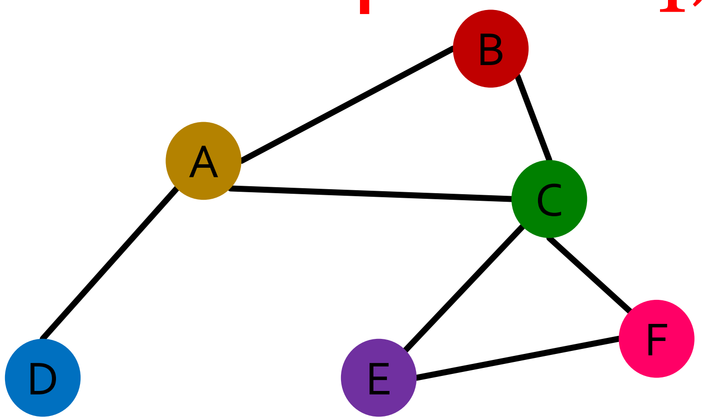
$f(\boldsymbol{A}_2, \boldsymbol{X}_2) =$



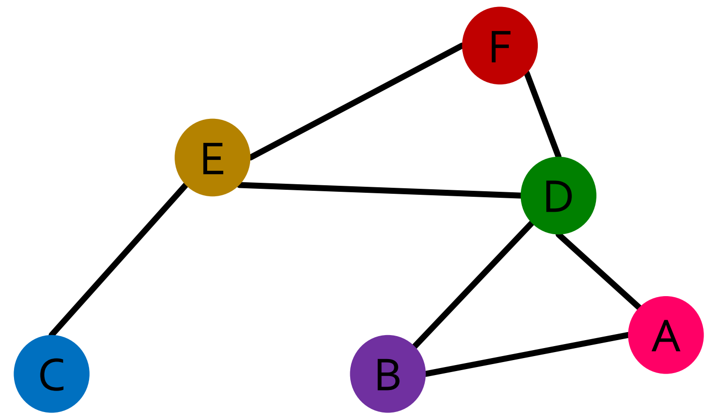
Permutation Equivariance

Similarly for node representation: We learn a function f that maps nodes of G to a matrix $\mathbb{R}^{m \times d}$.

Order plan 1: A_1, X_1

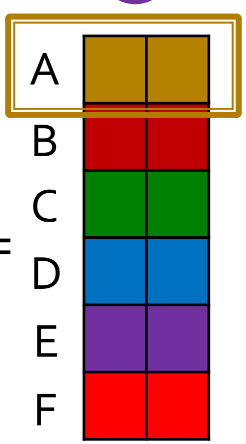


Order plan 2: A_2, X_2

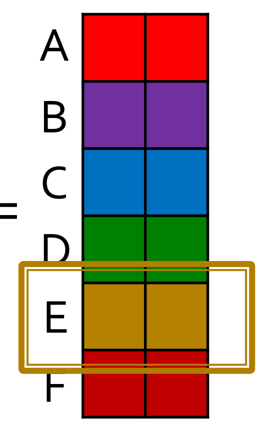


Representation vector of the brown node A

$$f(\boldsymbol{A}_1, \boldsymbol{X}_1) =$$



$$f(\boldsymbol{A}_2, \boldsymbol{X}_2) =$$



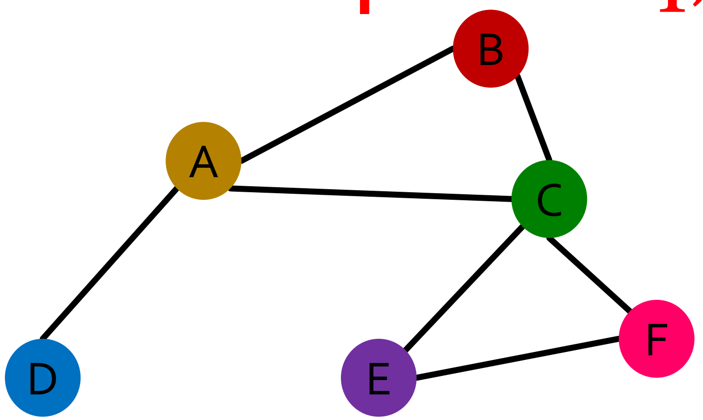
Representation vector of the brown node E

For two order plans, the vector of node at the same position is the same!

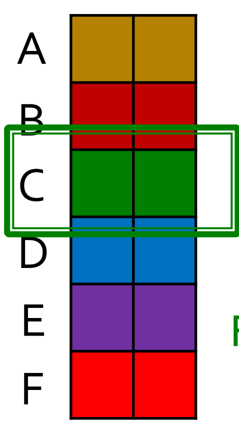
Permutation Equivariance

Similarly for node representation: We learn a function f that maps nodes of G to a matrix $\mathbb{R}^{m\times d}$.

Order plan 1: $\boldsymbol{A}_1, \boldsymbol{X}_1$

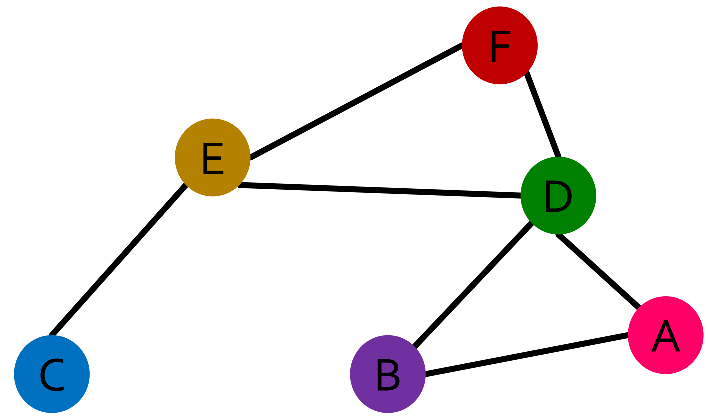


$f(\boldsymbol{A}_1, \boldsymbol{X}_1) =$

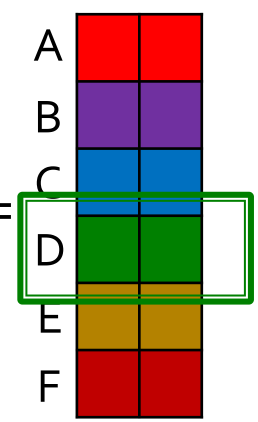


Representation vector of the brown node C

Order plan 2: $\boldsymbol{A}_2, \boldsymbol{X}_2$



$f(\boldsymbol{A}_2, \boldsymbol{X}_2) =$



Representation vector of the brown node D

For two order plans, the vector of node at the same position is the same!

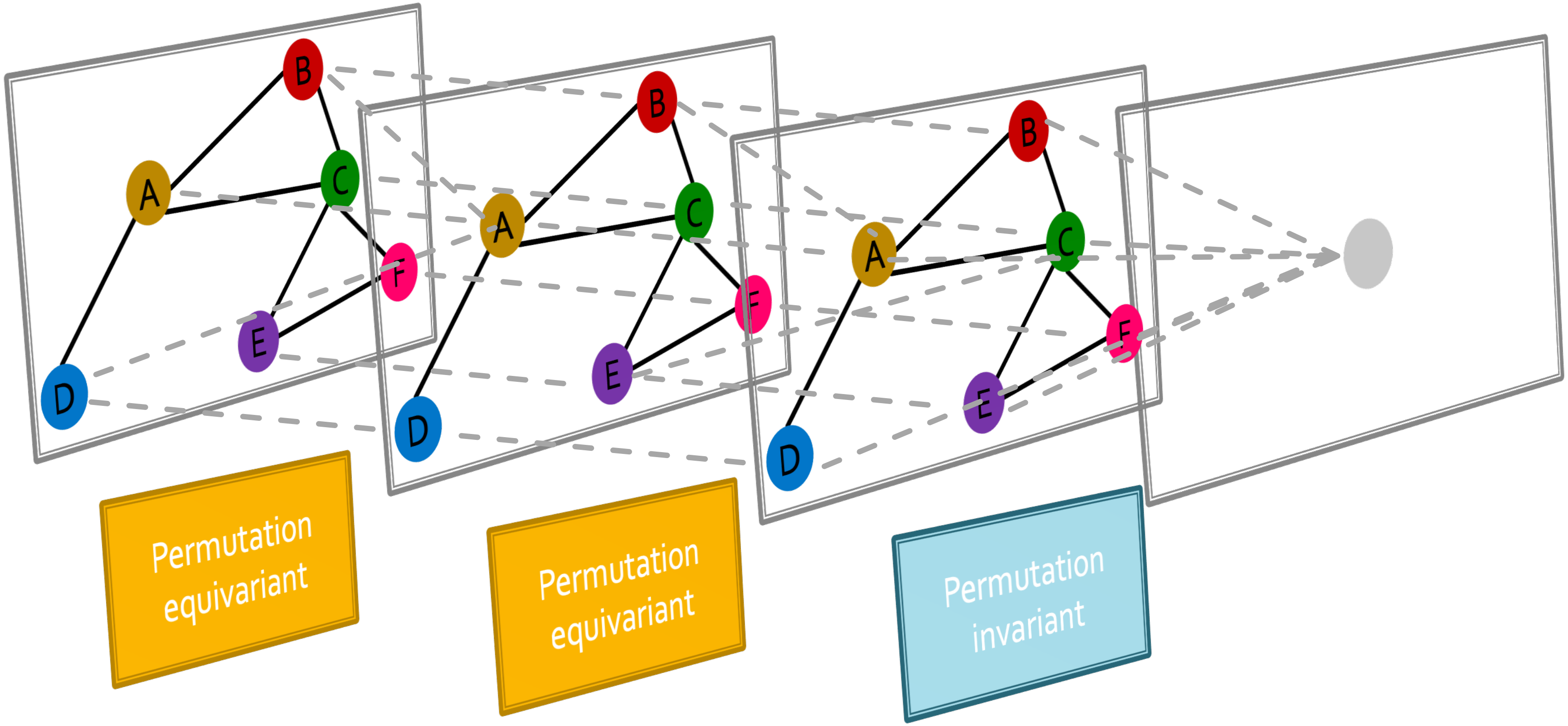
Permutation Equivariance

For node representation

- Consider we learn a function f that maps a graph $G = (\boldsymbol{A}, \boldsymbol{X})$ to a matrix $\mathbb{R}^{m \times d}$
  - graph has m nodes, each row is the embedding of a node.
- Similarly, if this property holds for any pair of order plan i and j, we say f is a **permutation equivariant function**.

Graph Neural Network Overview

- **Graph neural networks consist of multiple permutation equivariant / invariant functions.**

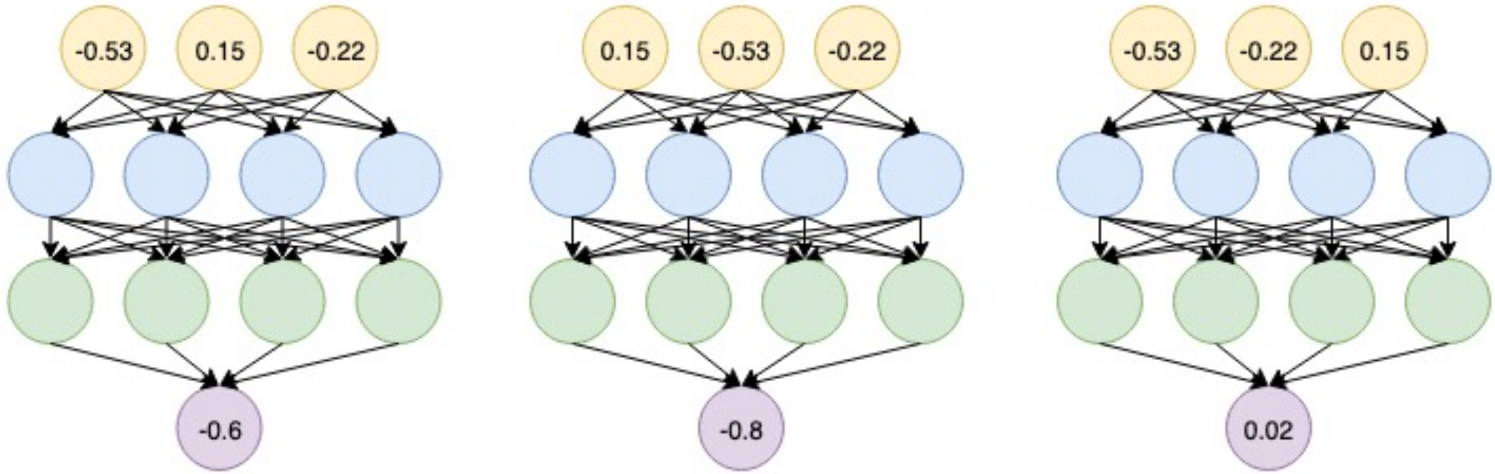


Graph Neural Network Overview

Are other neural network architectures, e.g., MLPs, permutation invariant / equivariant?

- **No.**

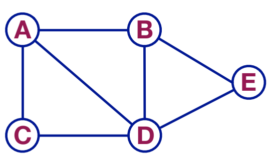
Switching the order of the input leads to different outputs!



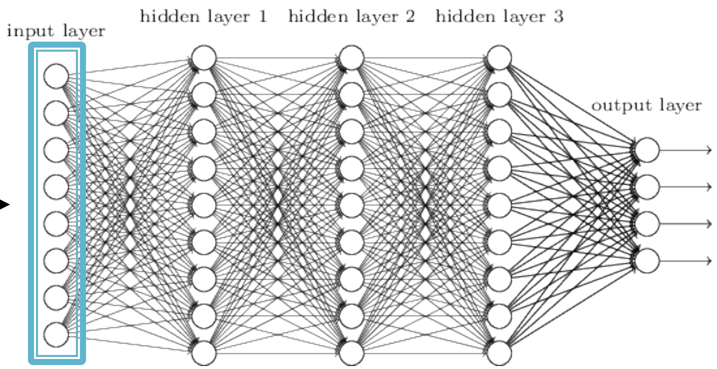
Graph Neural Network Overview

Are other neural network architectures, e.g., MLPs, permutation invariant / equivariant?

- **No.**



	A	B	C	D	E	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

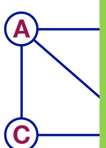


?

This explains why **the naïve MLP approach fails for graphs**!

Graph Neural Network Overview

- Are any neural network architecture, e.g.,



Next: Design graph neural networks that are permutation invariant / equivariant by passing and aggregating information from neighbors!

?