

CSCI 1470/2470
Spring 2024

Ritambhara Singh

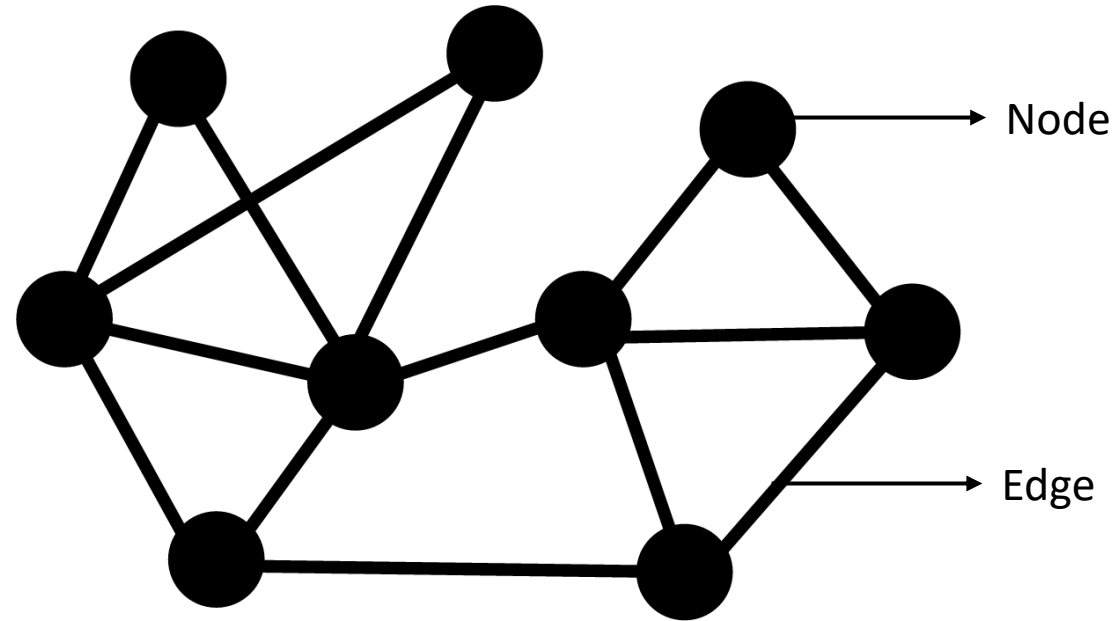
April 26, 2024
Friday

Graph Neural Networks

Deep Learning



General language for describing entities with relations/interactions



Graph

Majority of real-world data can be described using graphs



Image credit: [Medium](#)

Social Networks

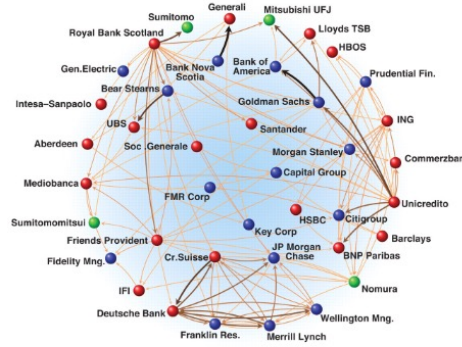


Image credit: [Science](#)

Economic Networks

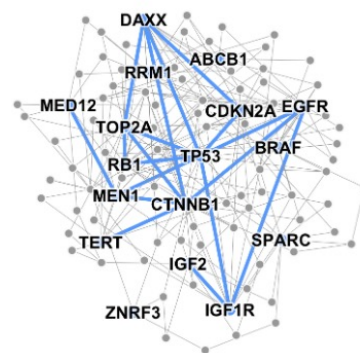


Image credit: [Lumen Learning](#)

Communication Networks



Citation Networks



Disease Pathways

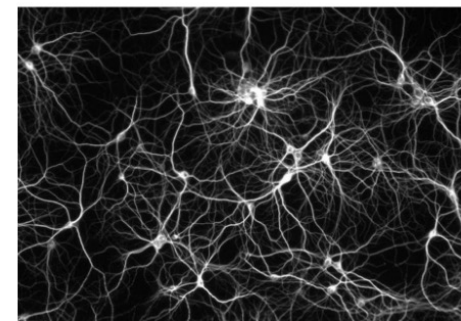
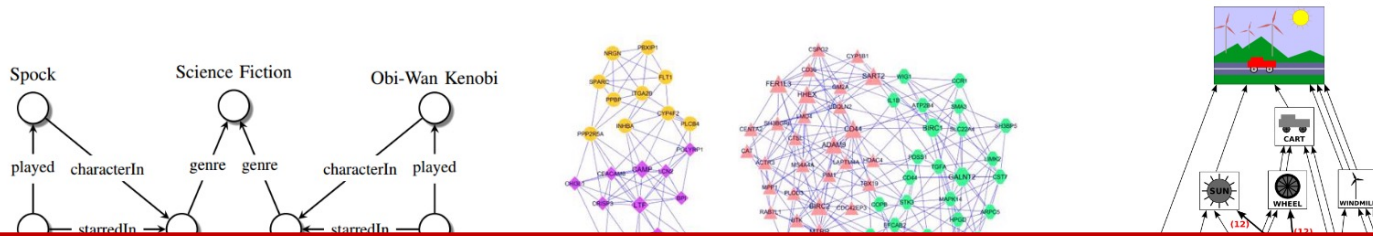


Image credit: [The Conversation](#)

Networks of Neurons

Majority of real-world data can be described using graphs



By explicitly modeling relationships we achieve better performance!

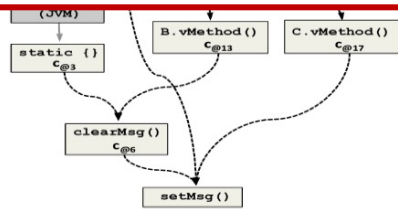


Image credit: [ResearchGate](#)

Code Graphs

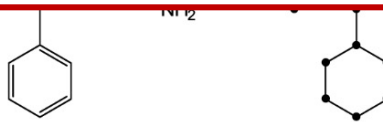


Image credit: [MDPI](#)

Molecules

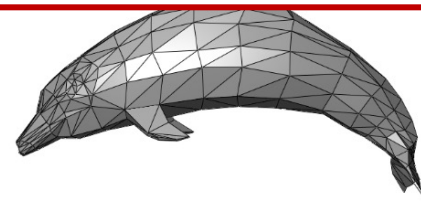
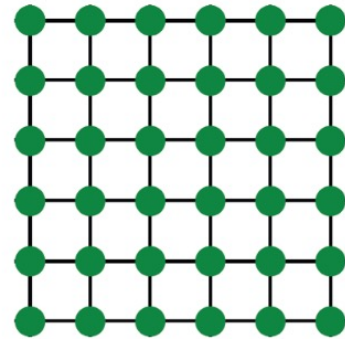


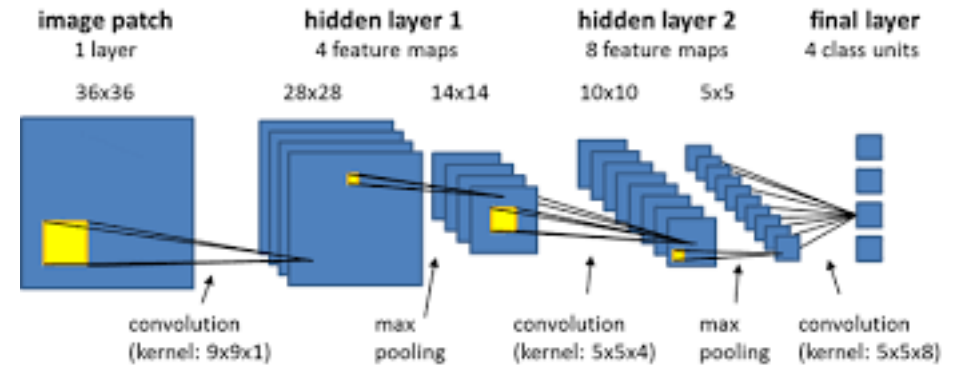
Image credit: [Wikipedia](#)

3D Shapes

Existing deep learning models designed for grids and sequences

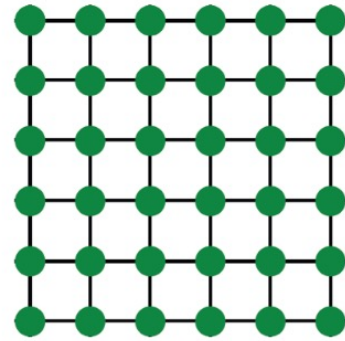


Images

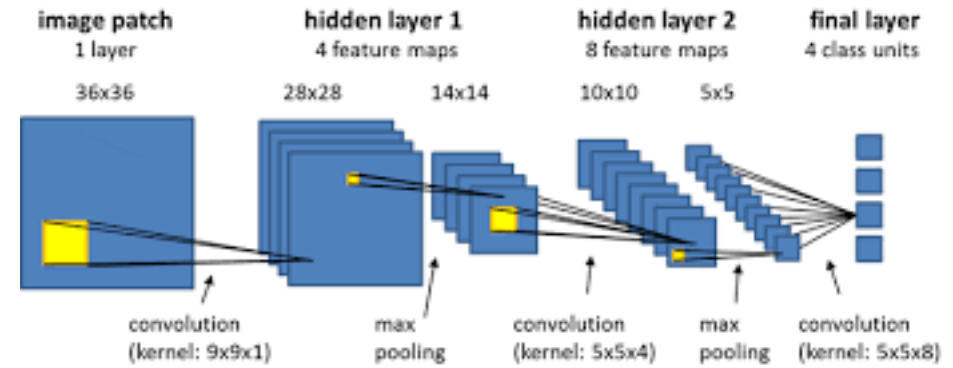


Convolutional Neural Network (CNN)

Existing deep learning models designed for grids and sequences

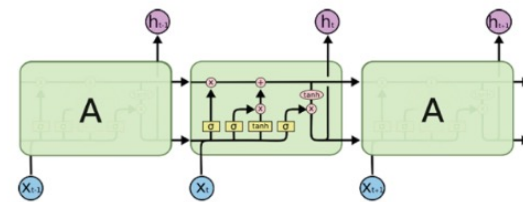


Images



Convolutional Neural Network (CNN)

Text/Speech



Recurrent Neural Network (RNN)

What's hard about NNs for graphs?

- Let's think about the problem of **summarizing** a data item
- We've done this for different types of data already:
 - Grids: Pass information from the neighboring pixels
 - Sequences: Pass information along the sequence (or, use a Transformer with positional embeddings)
- In both cases: there is an obvious way to **direct** the computation toward a single end point
 - Grid: center of the grid
 - Sequences: the end of the sequence

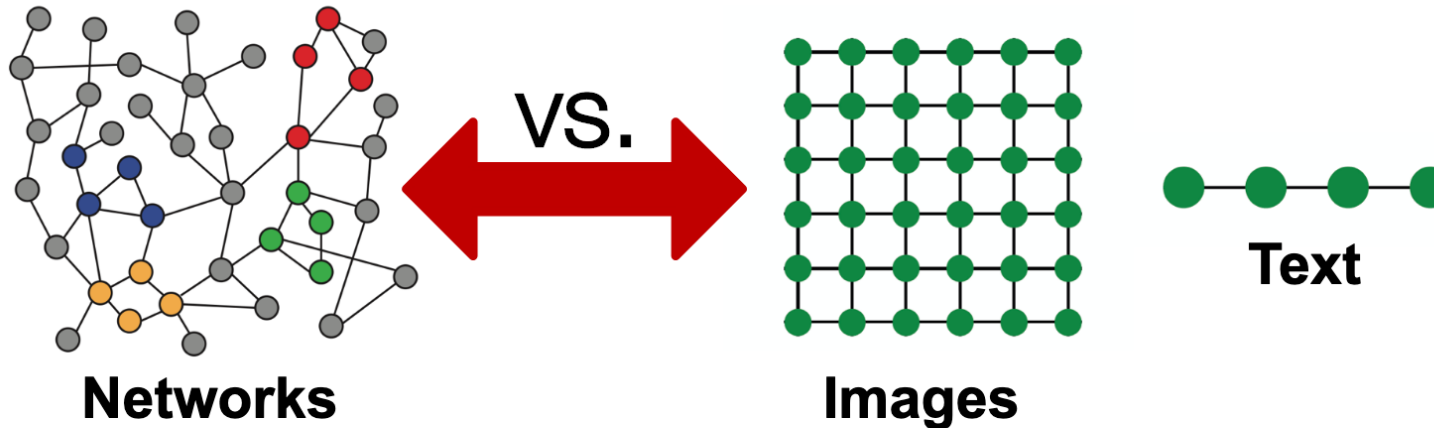
What's hard about NNs for graphs?

Networks are complex.

What's hard about NNs for graphs?

Networks are complex.

- Arbitrary size and complex topological structure (*i.e.*, no spatial locality like grids)



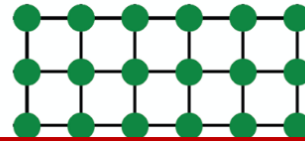
What's hard about NNs for graphs?

Networks are complex.

- Arbitrary size and complex topological structure (*i.e.*, no spatial locality like grids)



VS.



How to pass information along the nodes of the graphs?

Networks

Images

- No fixed node ordering or reference point

An alternate take on summarization

- If we look at our architectures for summarizing sequences and grids, they are actually doing two important things at once:
 1. Passing information along relationships between individual elements
 2. Aggregating per-element information into a summarized description
- Let's design a graph network that does both of these things

Today's goal – learn about graph-based neural networks (GNNs)

(1) Motivation

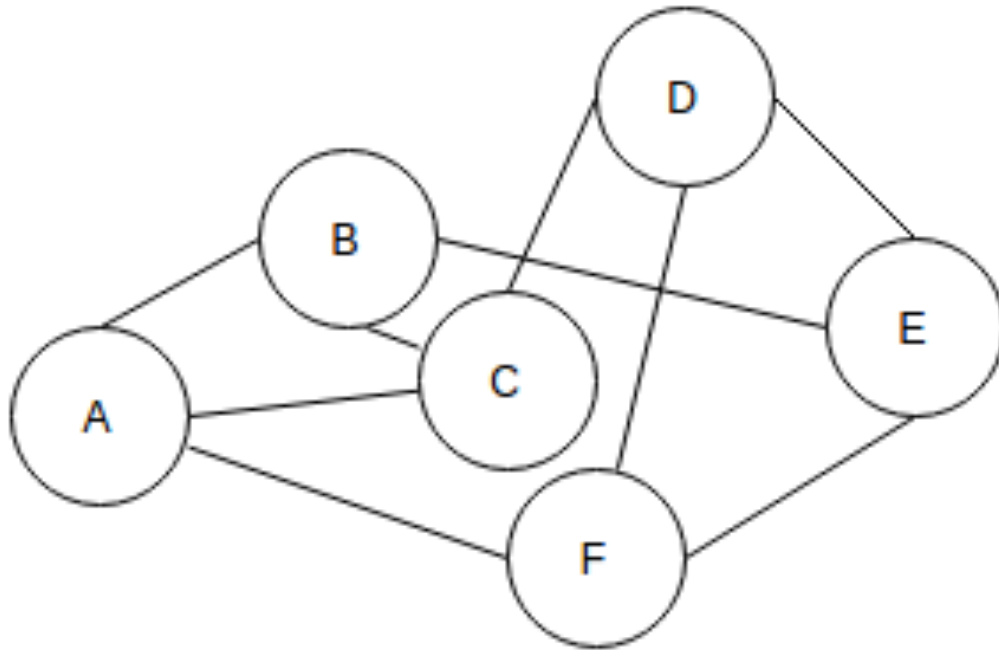
(2) **Passing information in Graphs (for NNs)**

(3) How to use GNNs?

(4) Graph Attention Networks

Why is passing information necessary?

- I.e. why not just compute some **descriptor on the nodes**, then aggregate them?



↑
Node features:
numerical
values that
describe the
node



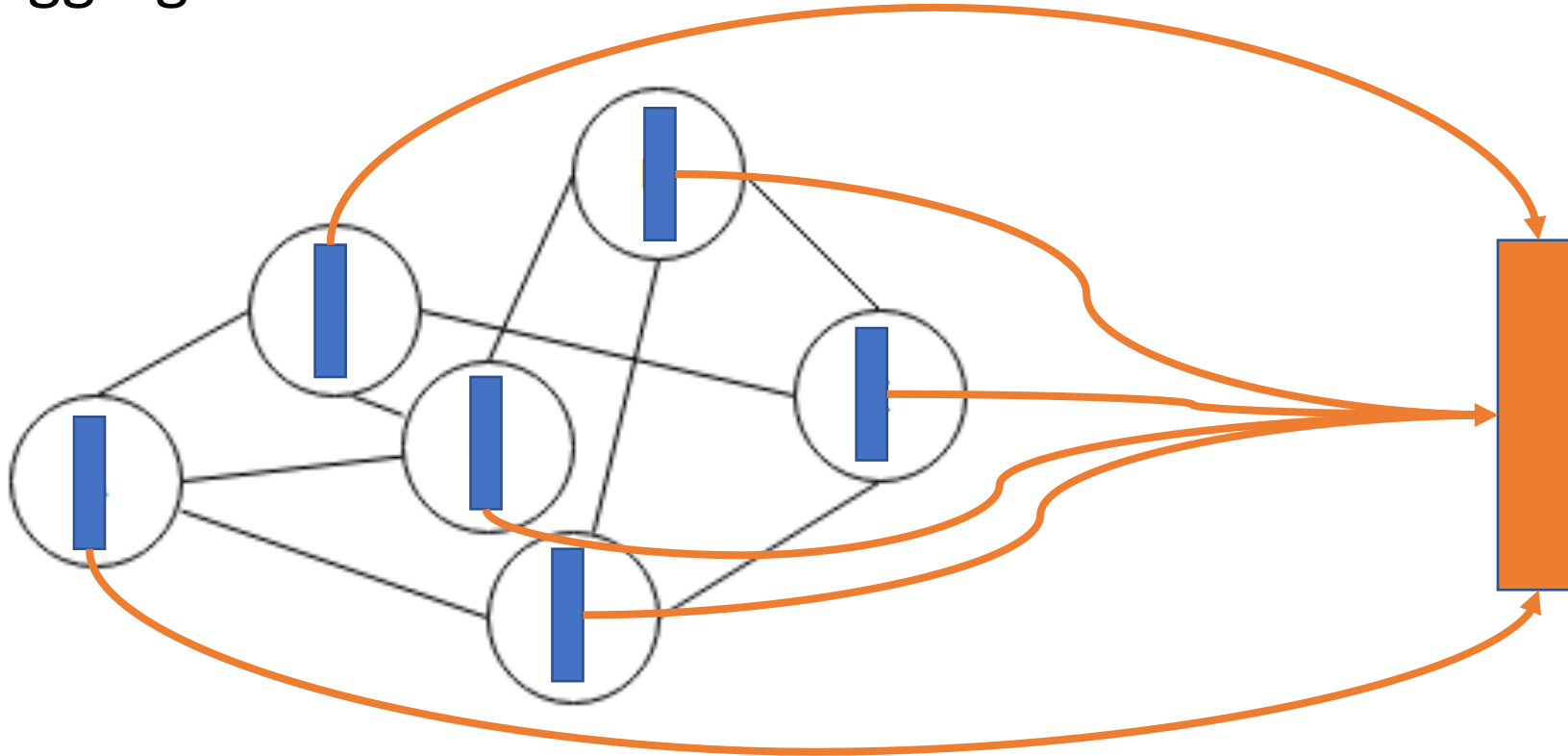
Image credit: [Medium](#)

Social Networks |

Node features: [24 (age), 1 (if employed),...]

Why is passing information necessary?

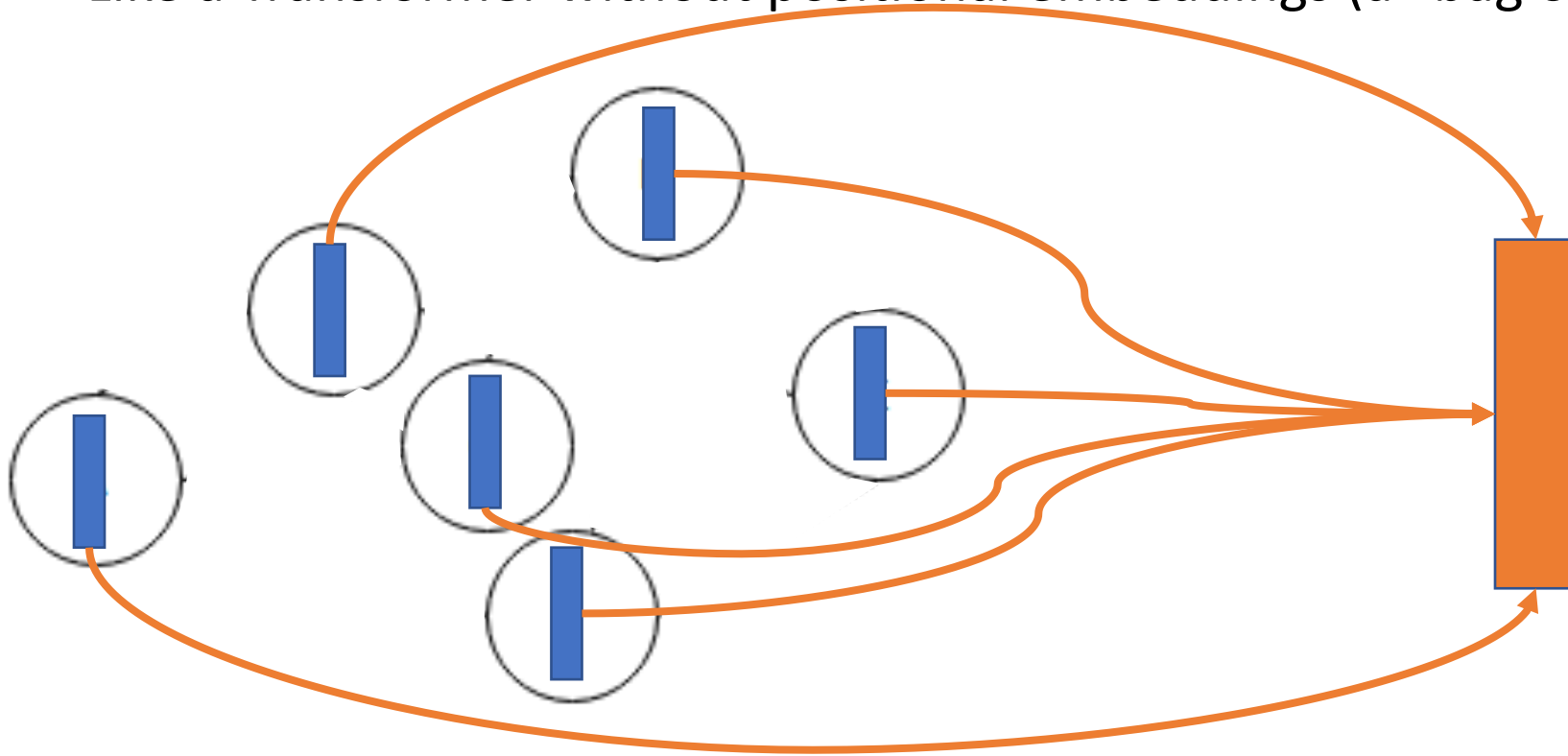
- I.e. why not just compute some descriptor on the nodes, then aggregate them?



What is the issue here?

Why is passing information necessary?

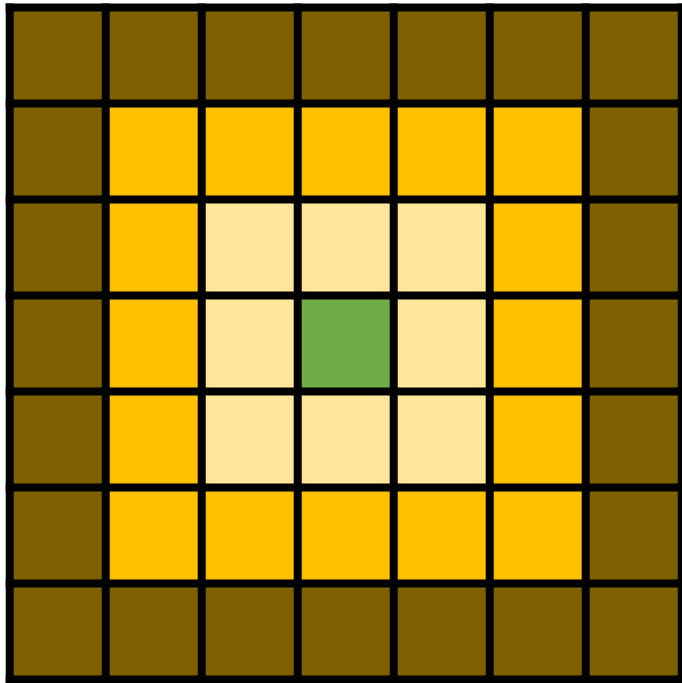
- This totally ignores the structure of the data!
 - Like a Transformer without positional embeddings (a “bag of nodes”)



So, how to pass information along edges?

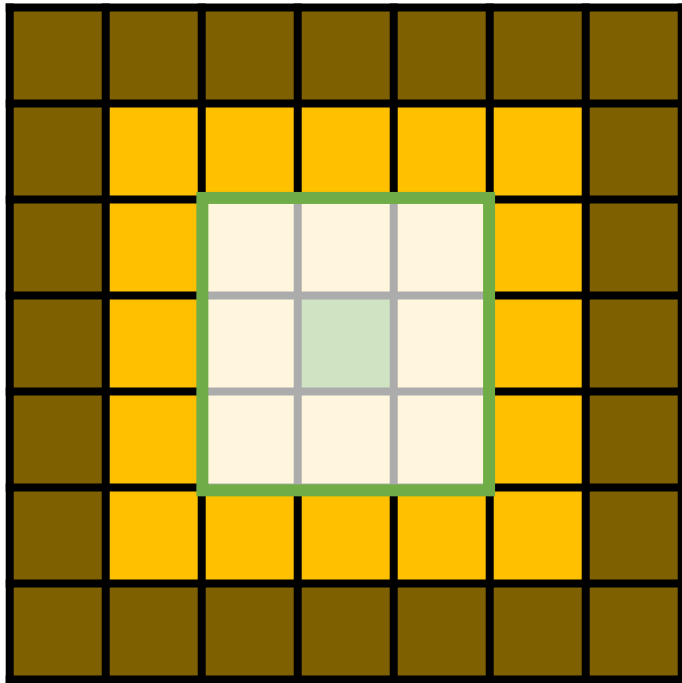
- We're going to generalize an operation we've already seen, so that it operates on graphs...

For NNs on Graphs, we generalize *convolution*



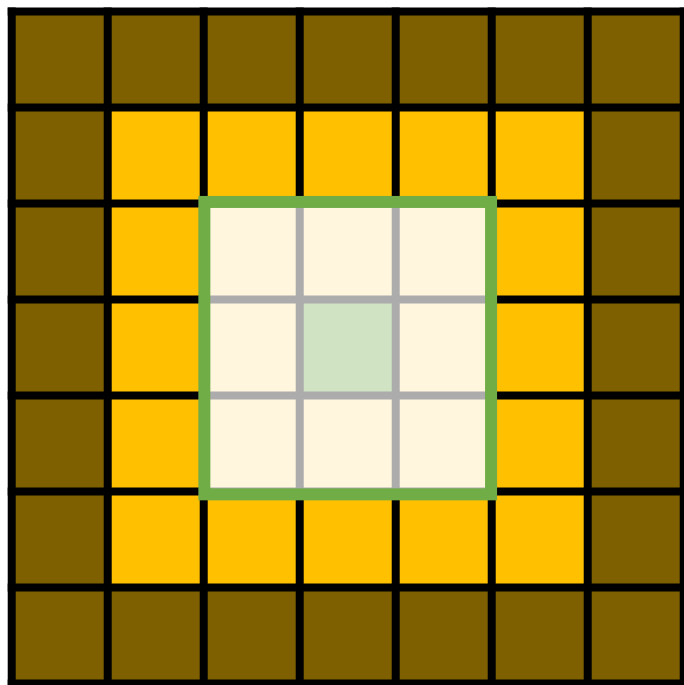
One **pixel** in an Image

For NNs on Graphs, we generalize *convolution*

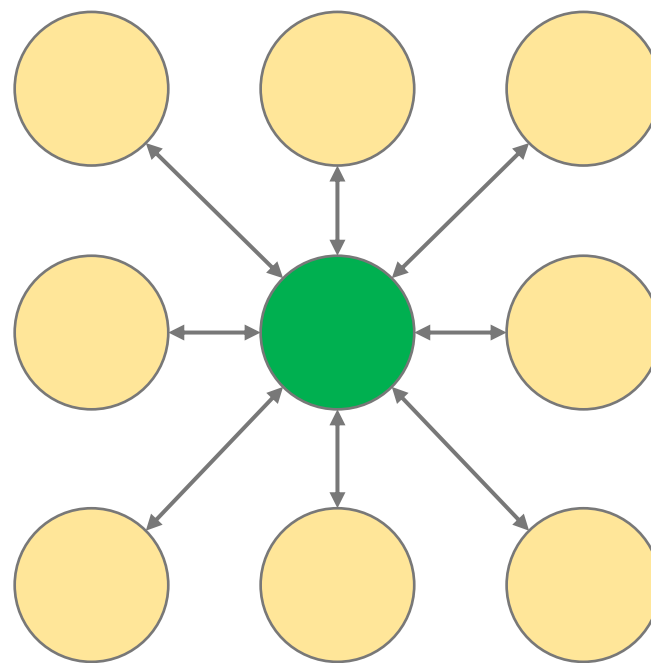


Convolution aggregates information from
the immediate neighborhood of a pixel

For NNs on Graphs, we generalize *convolution*



Convolution aggregates information from
the immediate neighborhood of a pixel



Another interpretation of
this aggregation

Looks like a graph, no?

What other information do we need here?

For NNs on Graphs, we generalize *convolution*

Inputs:
Node features +
Adjacency matrix

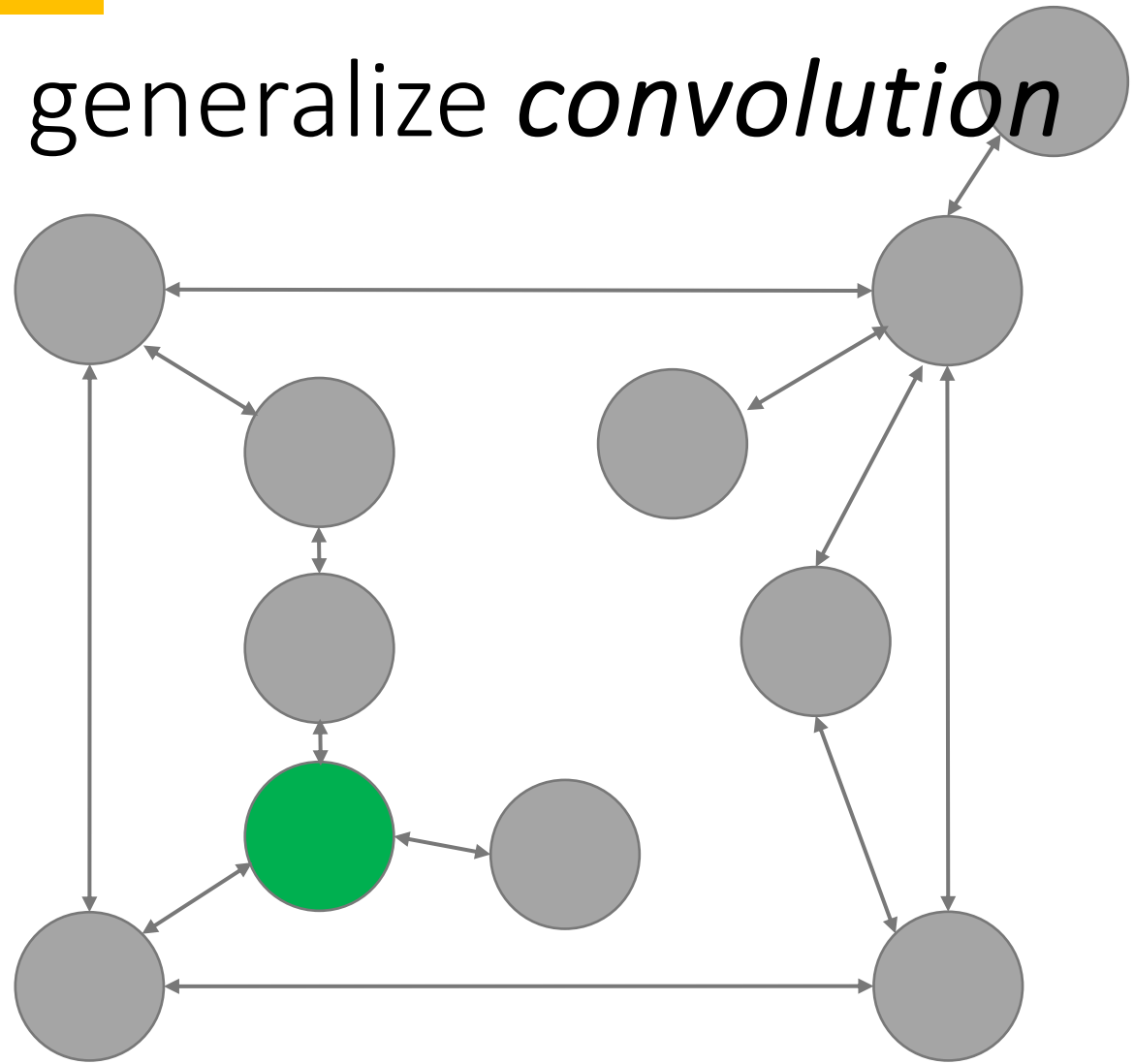


Image credit: [Medium](#)

Social Networks |

Node features: [24 (age), 1 (if employed),...]

Adjacency matrix:
$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$



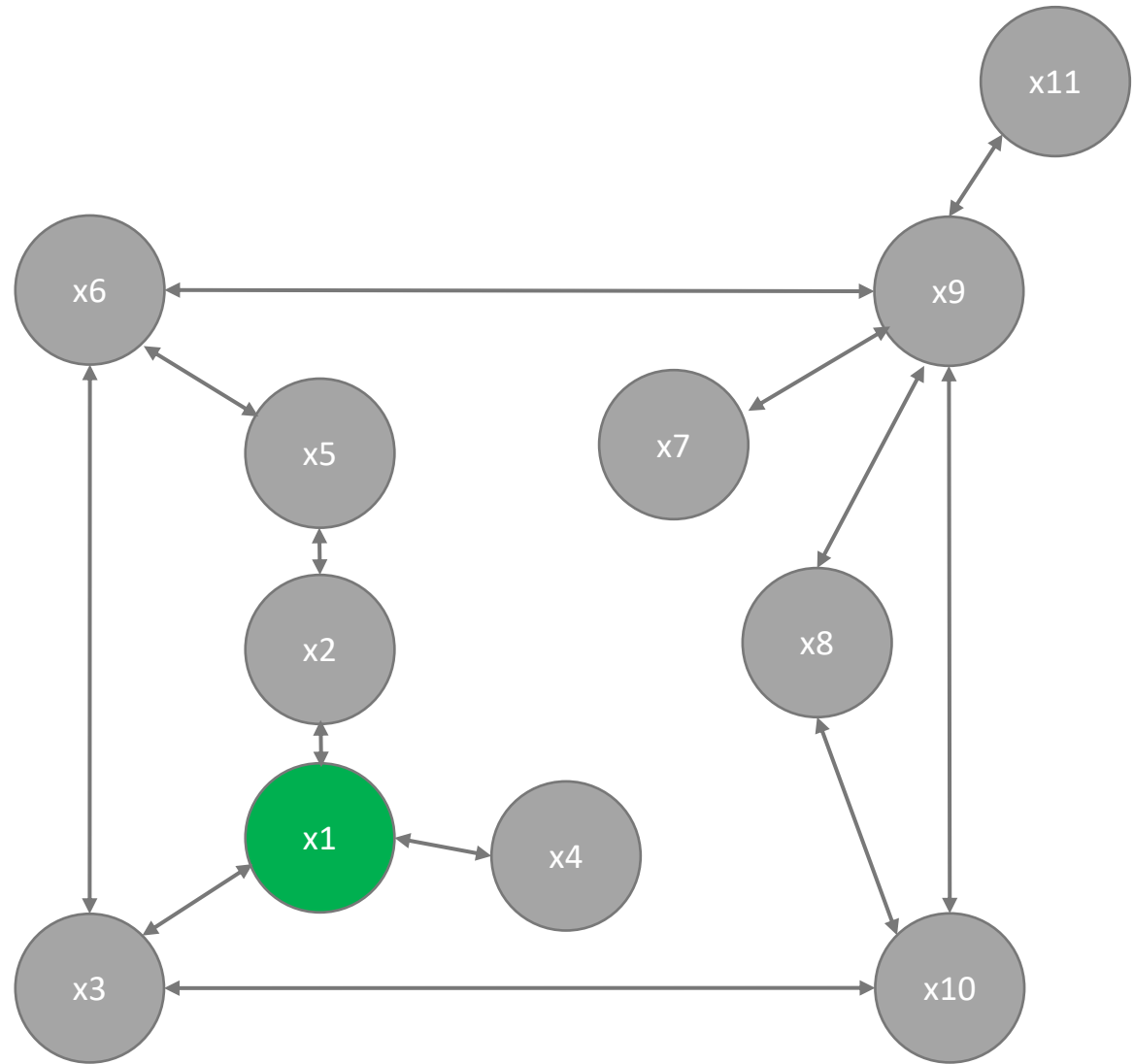
A more general graph

How do we define convolution on *this*?

Message Passing

We start by mapping the features \mathbf{x} at each node to a hidden state vector \mathbf{h}

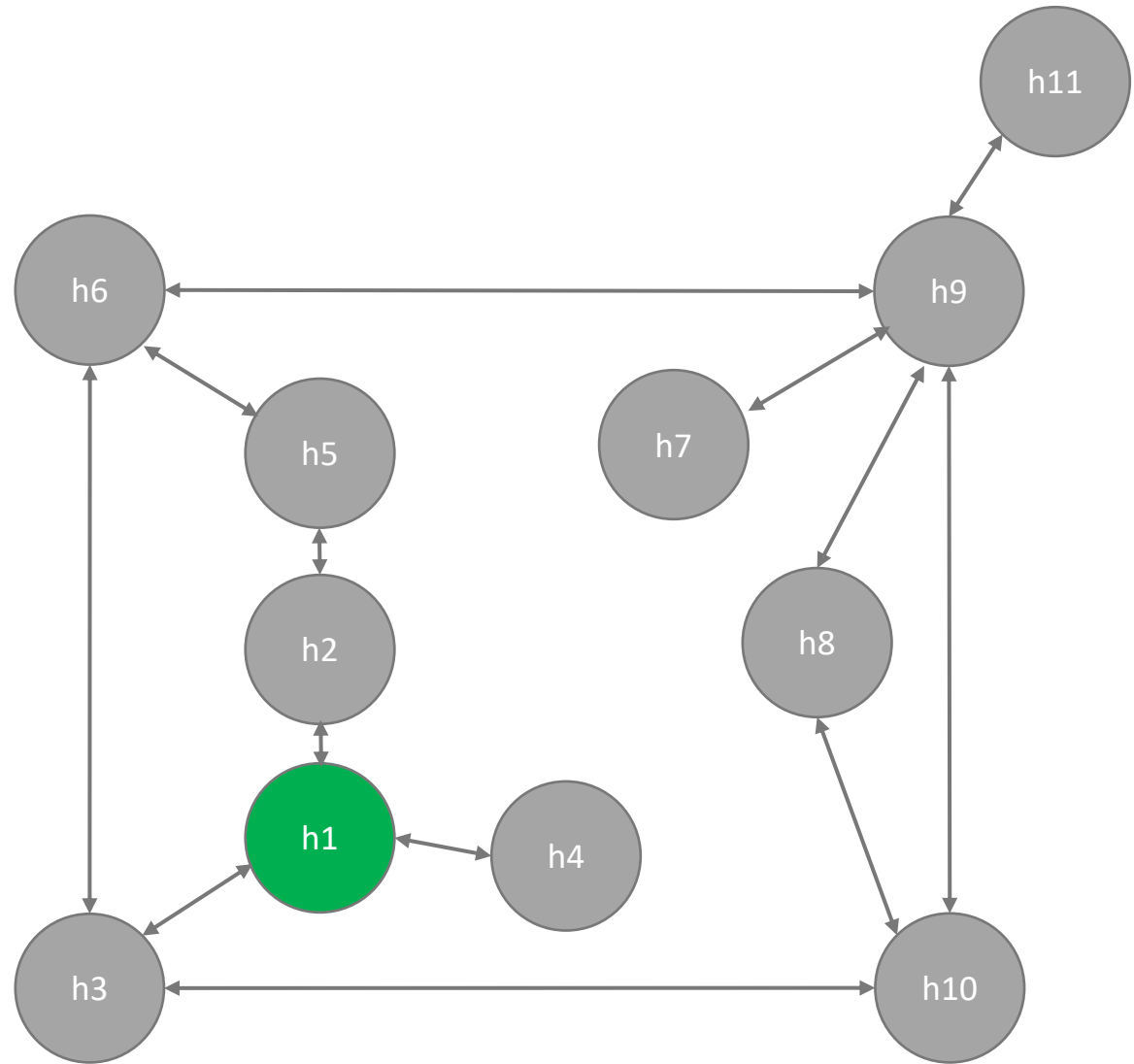
$$f_{\text{init}}(\mathbf{x}) = \mathbf{h}$$



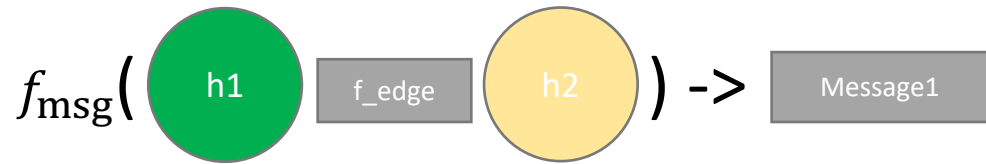
Message Passing

We start by mapping the features \mathbf{x} at each node to a hidden state vector \mathbf{h}

$$f_{\text{init}}(\mathbf{x}) = \mathbf{h}$$

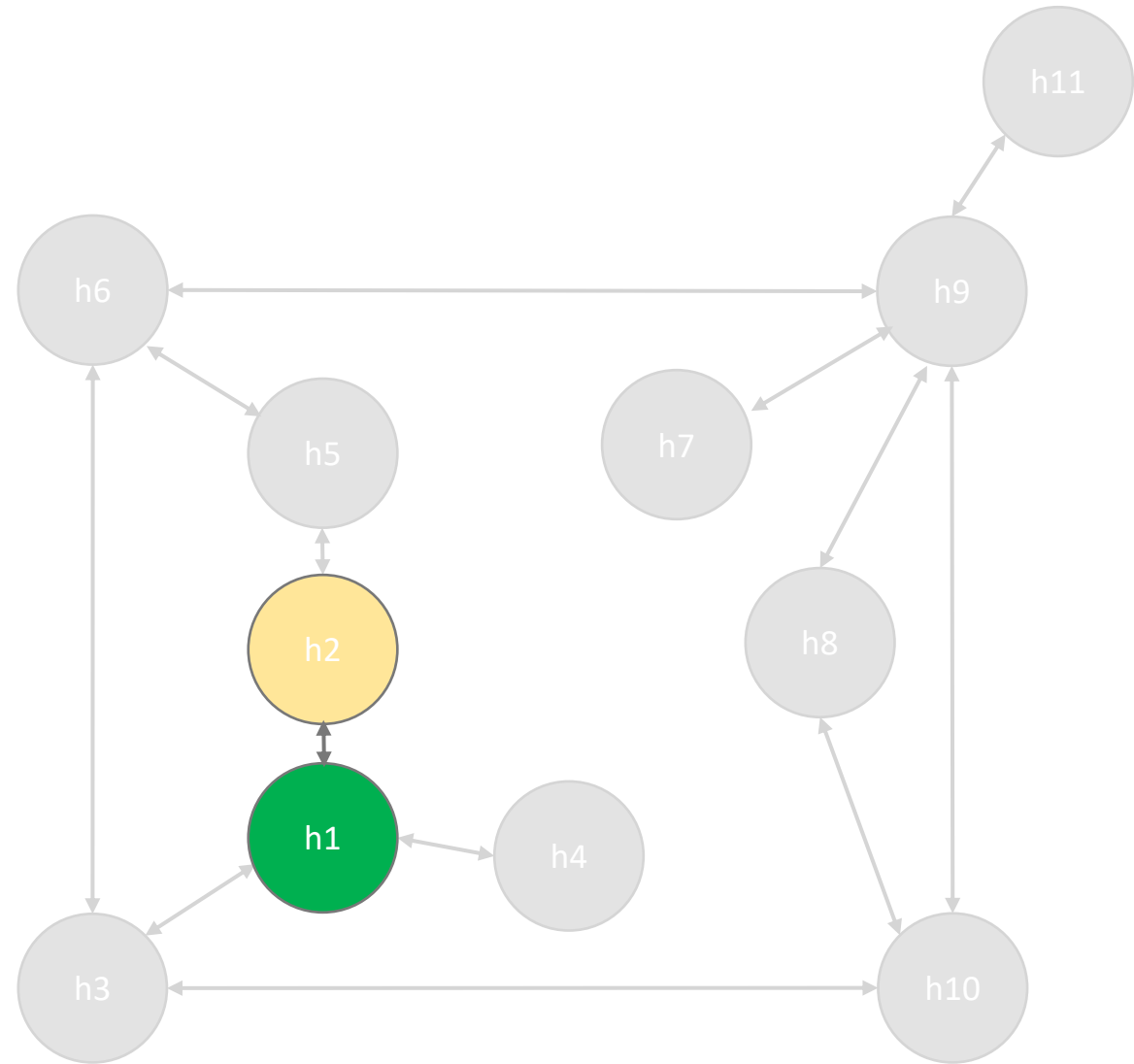


Message Passing

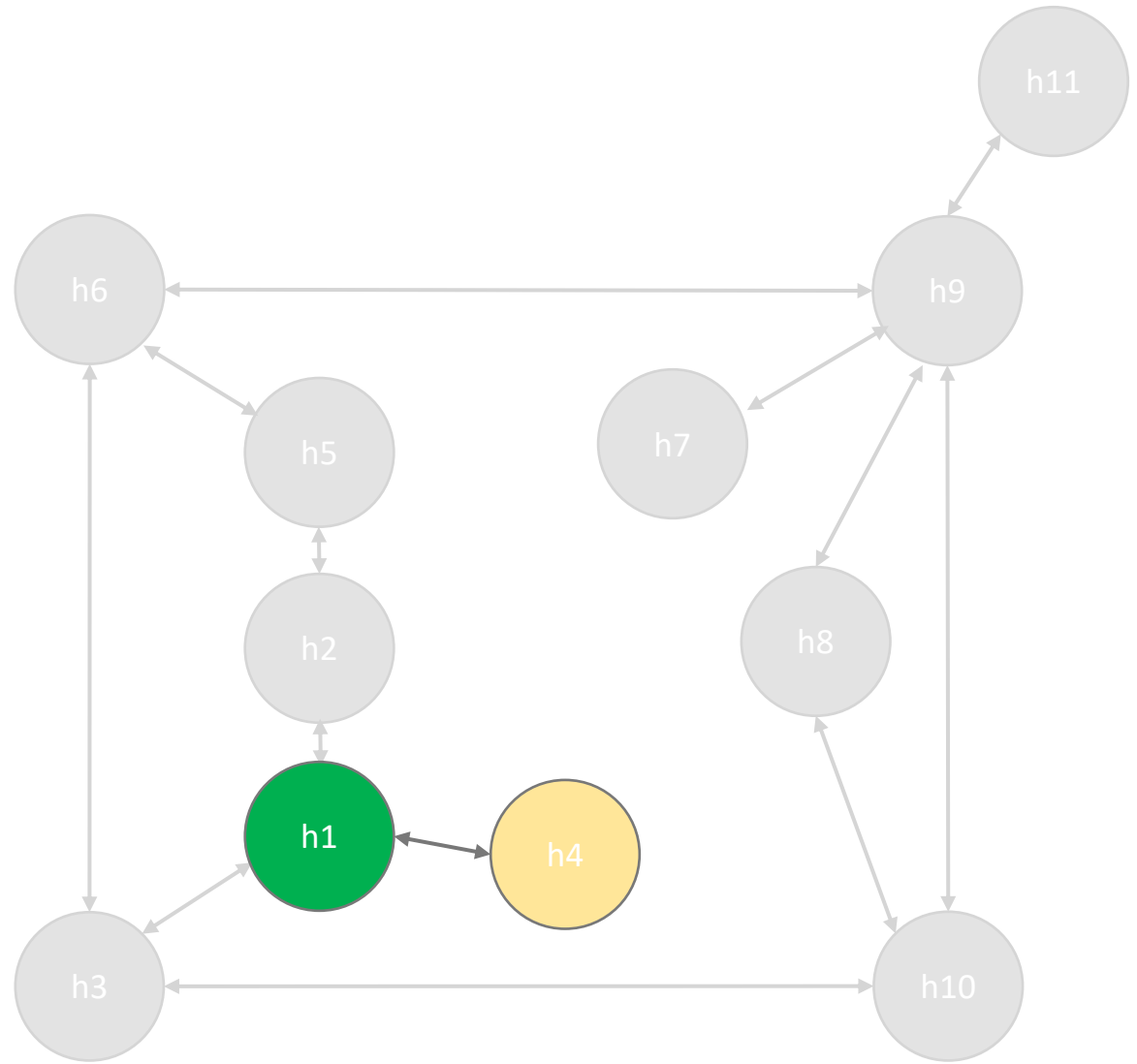
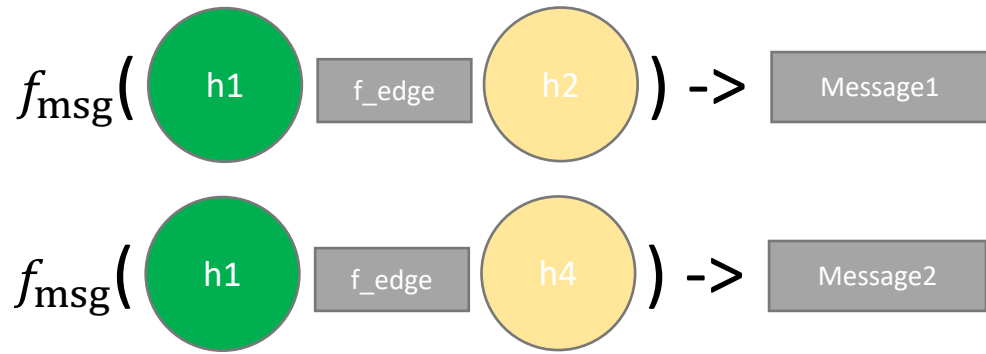


We use a neural network f_{msg} to compute a **message** between two nodes in the graph

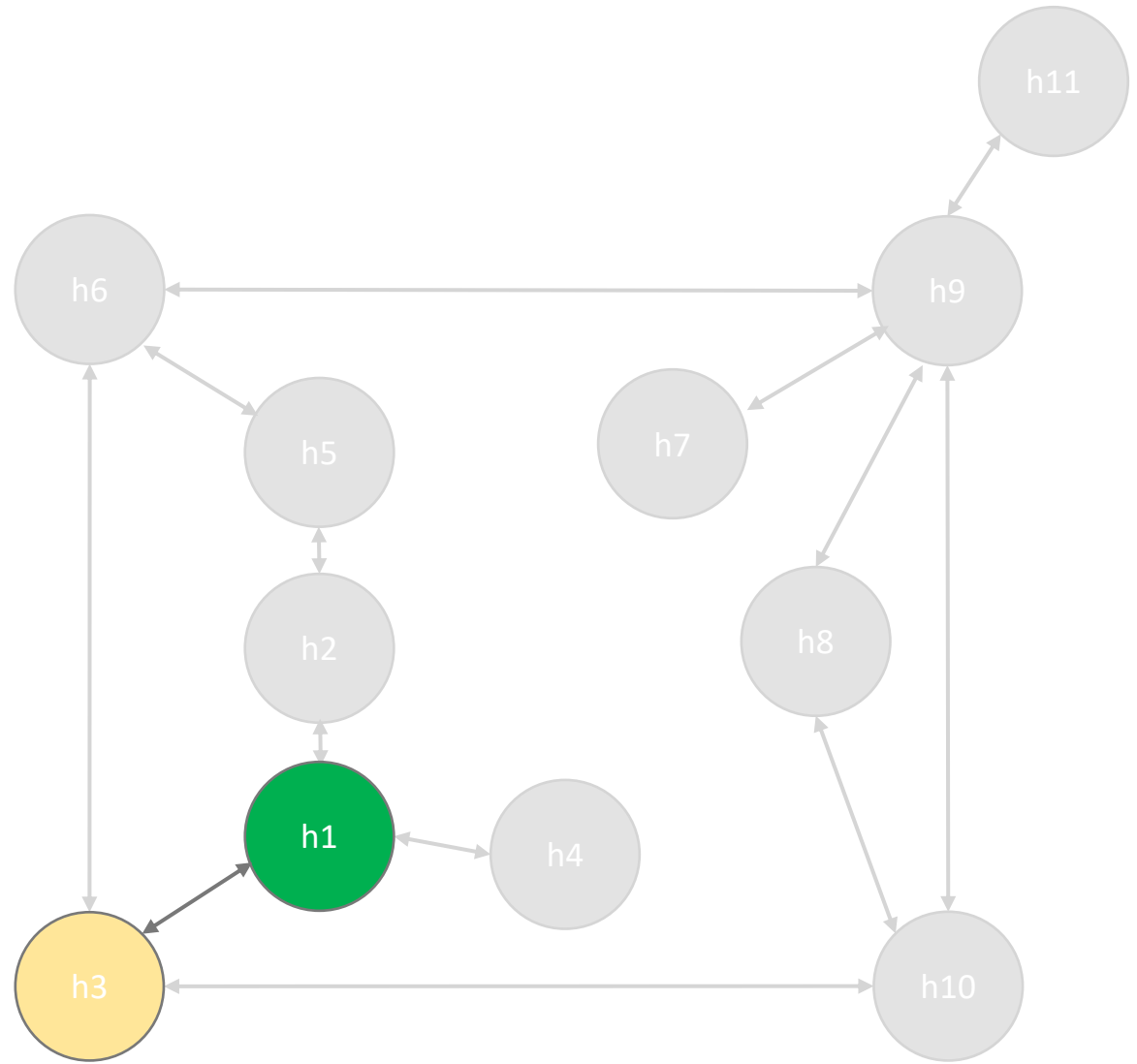
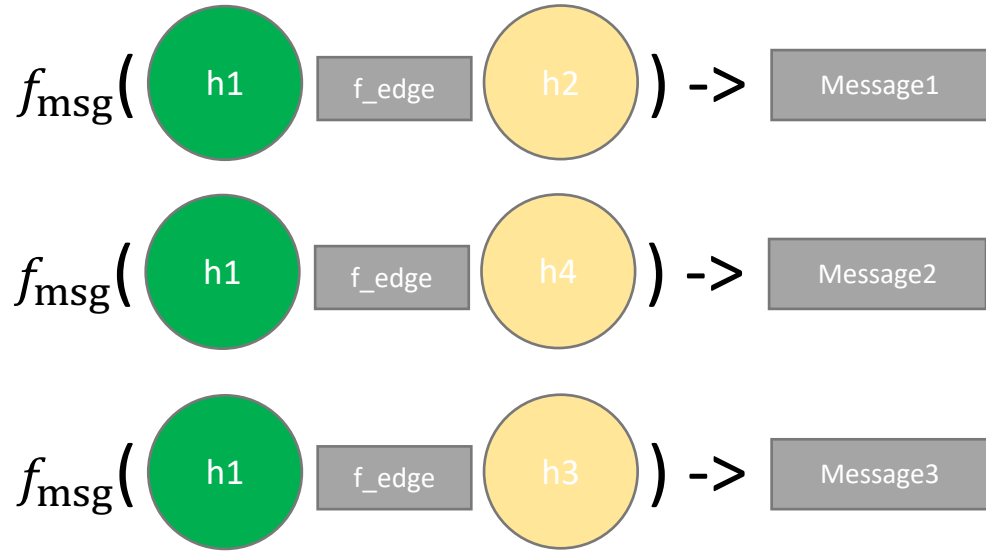
- Inputs
 - Hidden states for each node
 - Features along the edge
- Output
 - Message vector



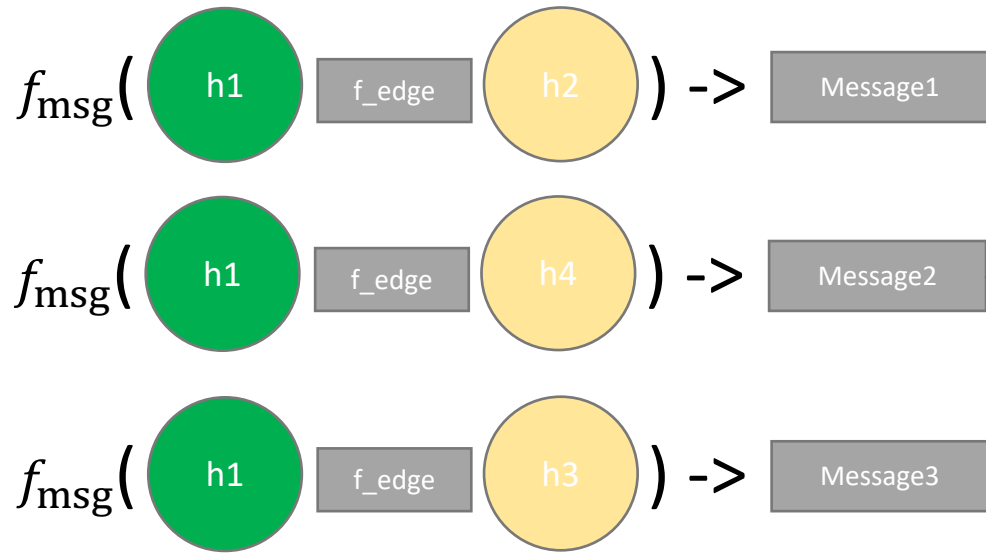
Message Passing



Message Passing

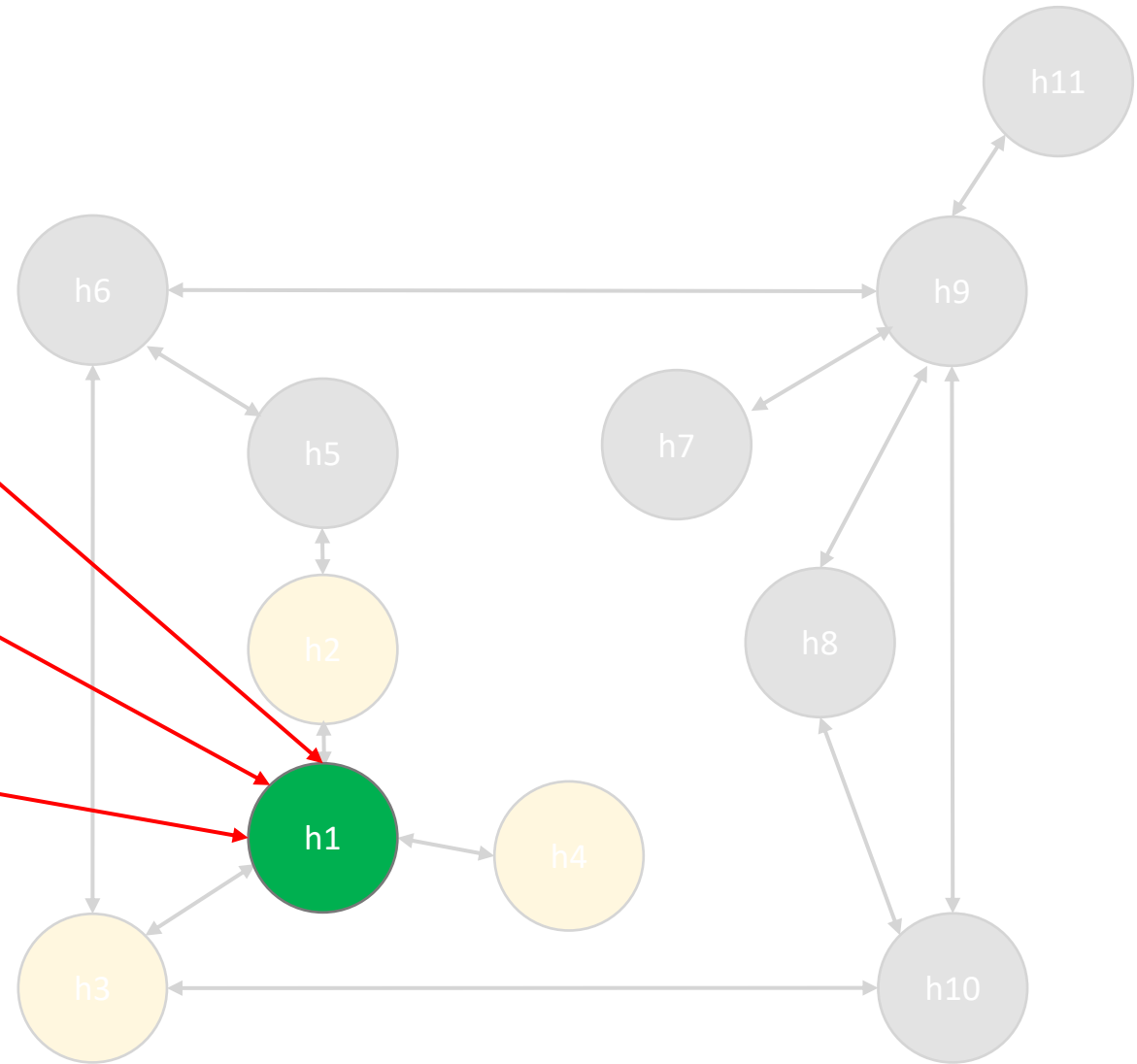


Message Passing



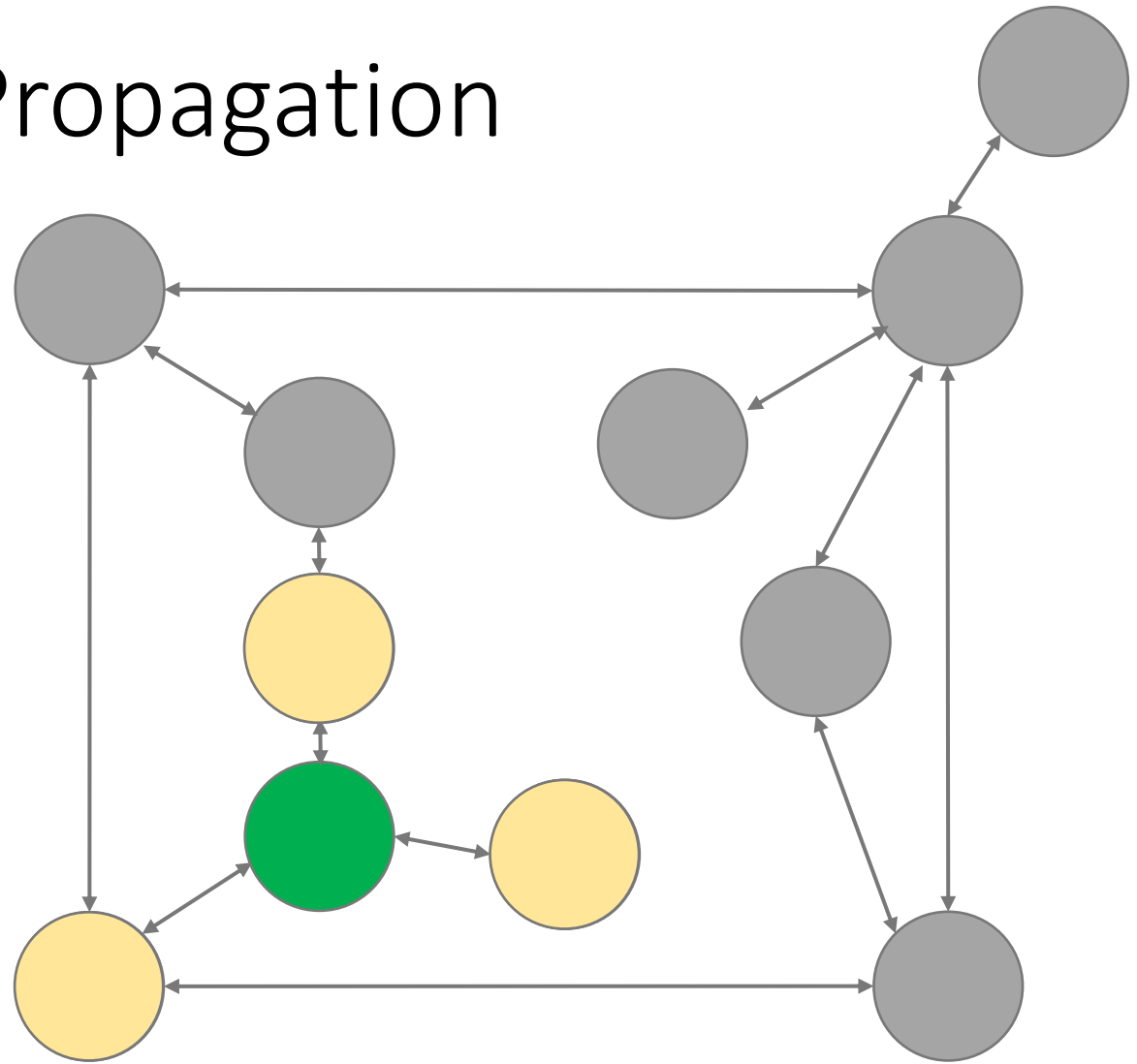
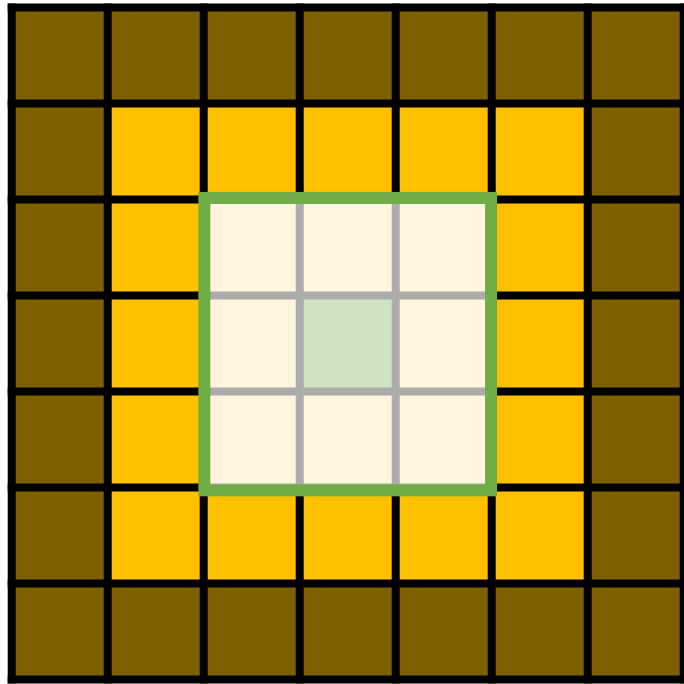
Aggregate all the messages to compute the output hidden state vector for \mathbf{h}_1

(e.g. sum up the messages)



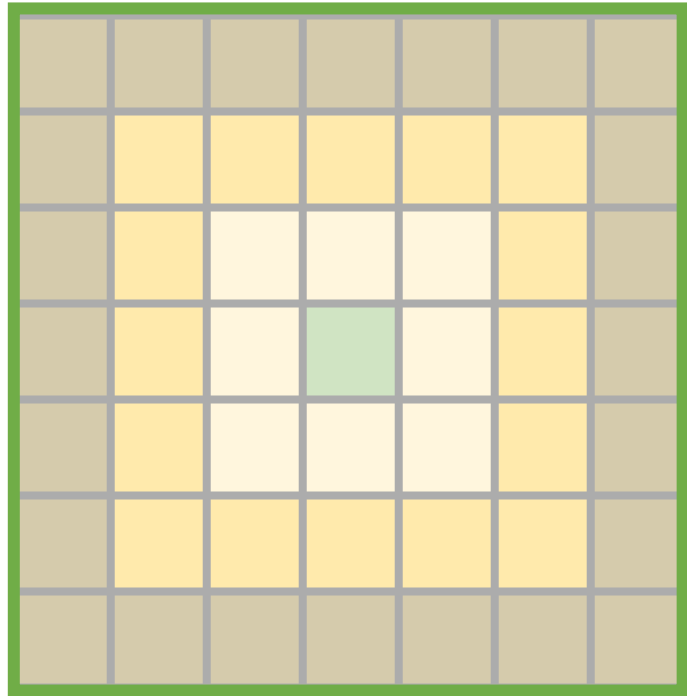
NOTE: The aggregation function can differ leading to different formulations of GNNs

Multiple Rounds of Propagation

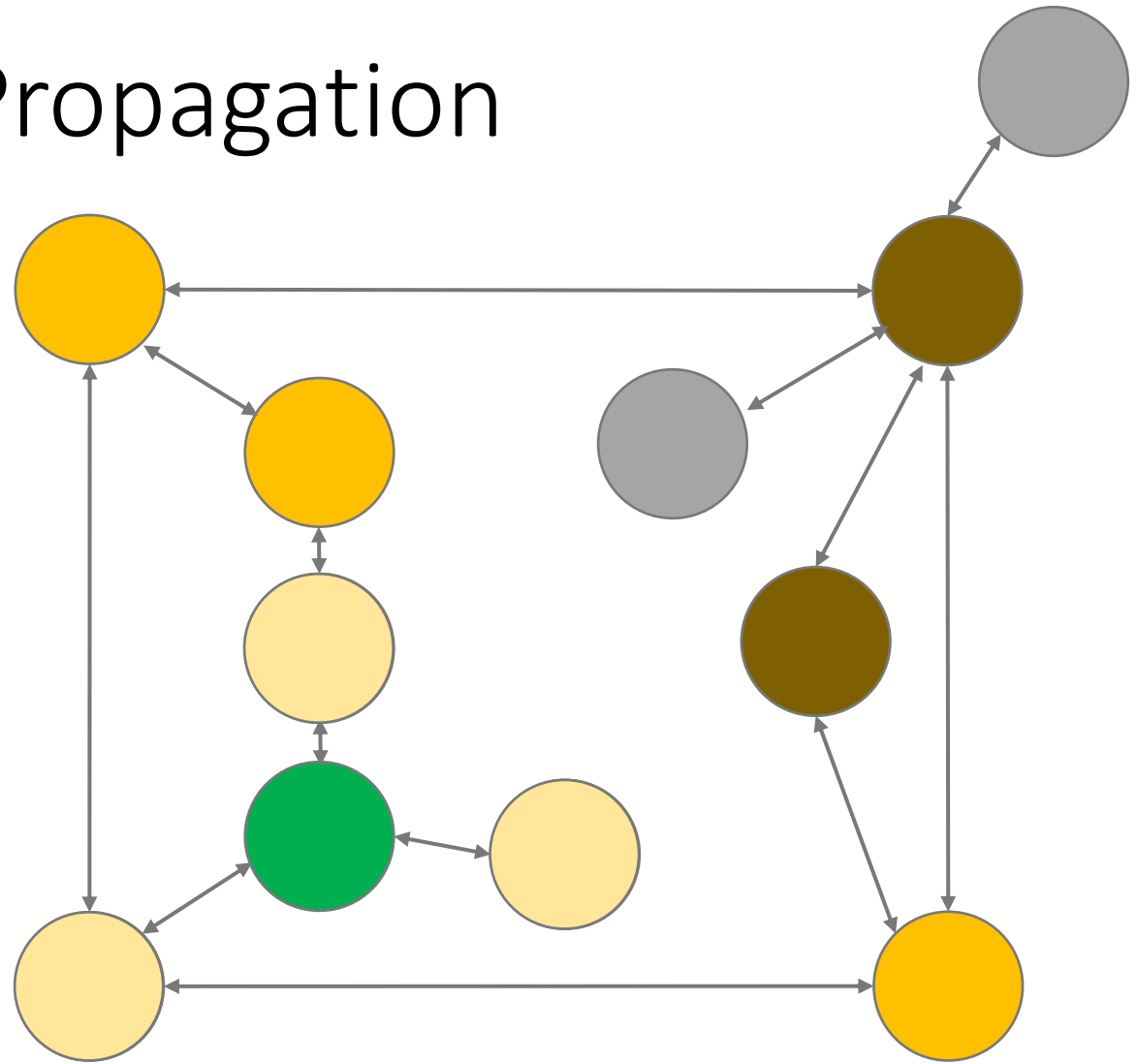


Just as multiple steps of convolution increases the “receptive field” of an image convolutional network...

Multiple Rounds of Propagation



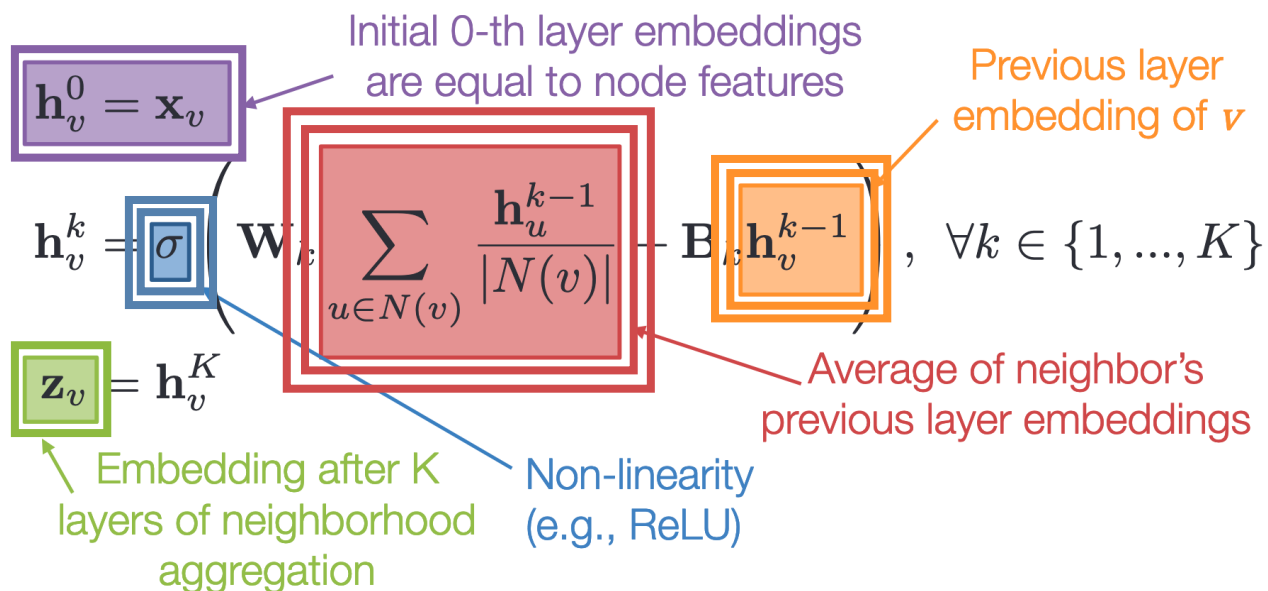
Just as multiple steps of convolution increases the “receptive field” of an image convolutional network...



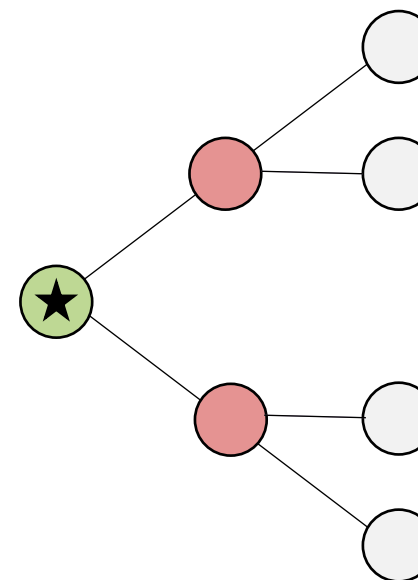
...multiple steps of message passing increases the range of information spread in the graph



Example: GraphSAGE formulation



W and B are learnable parameters



Today's goal – learn about graph-based neural networks (GNNs)

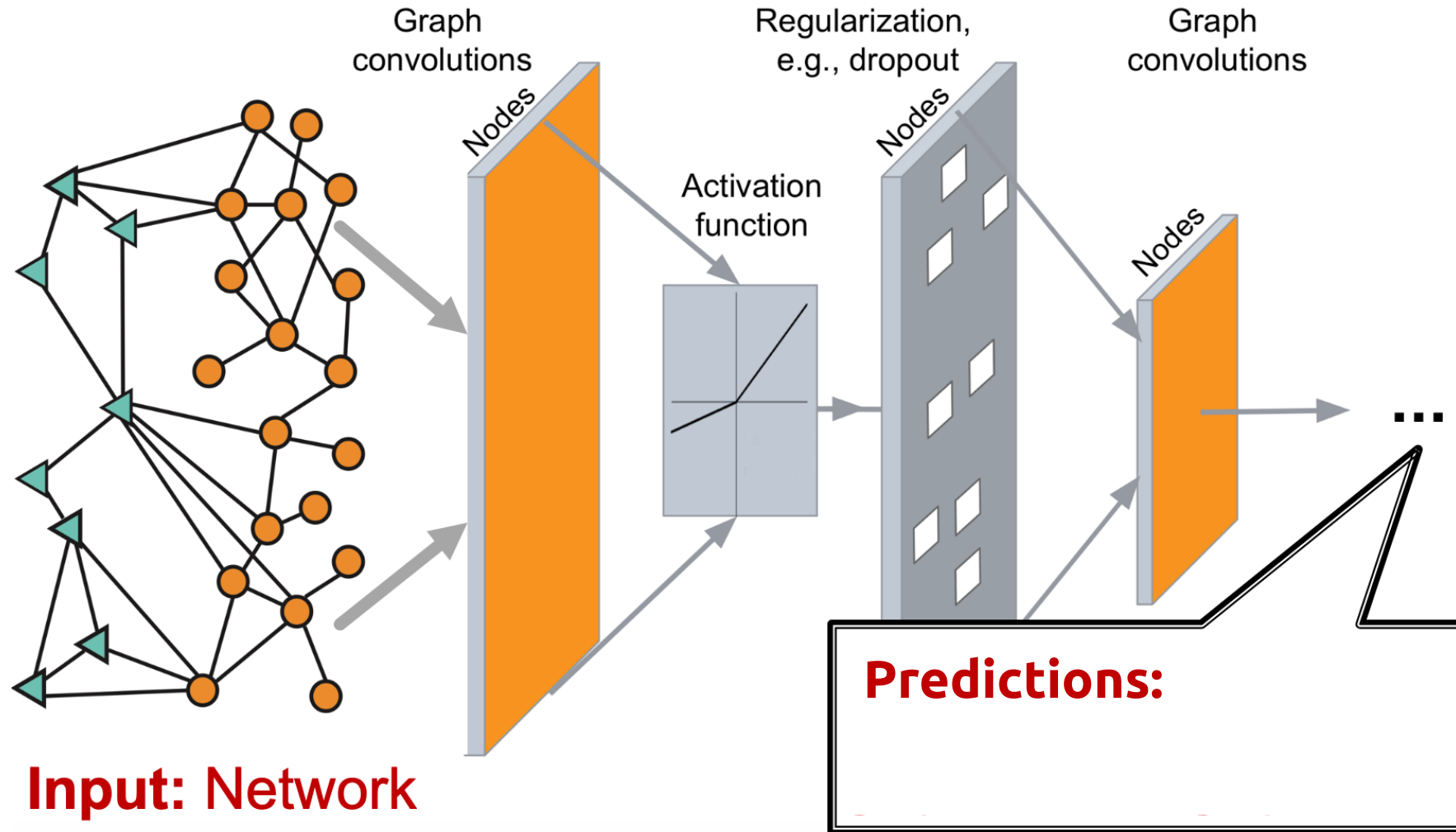
(1) Motivation

(2) Passing information in Graphs (for NNs)

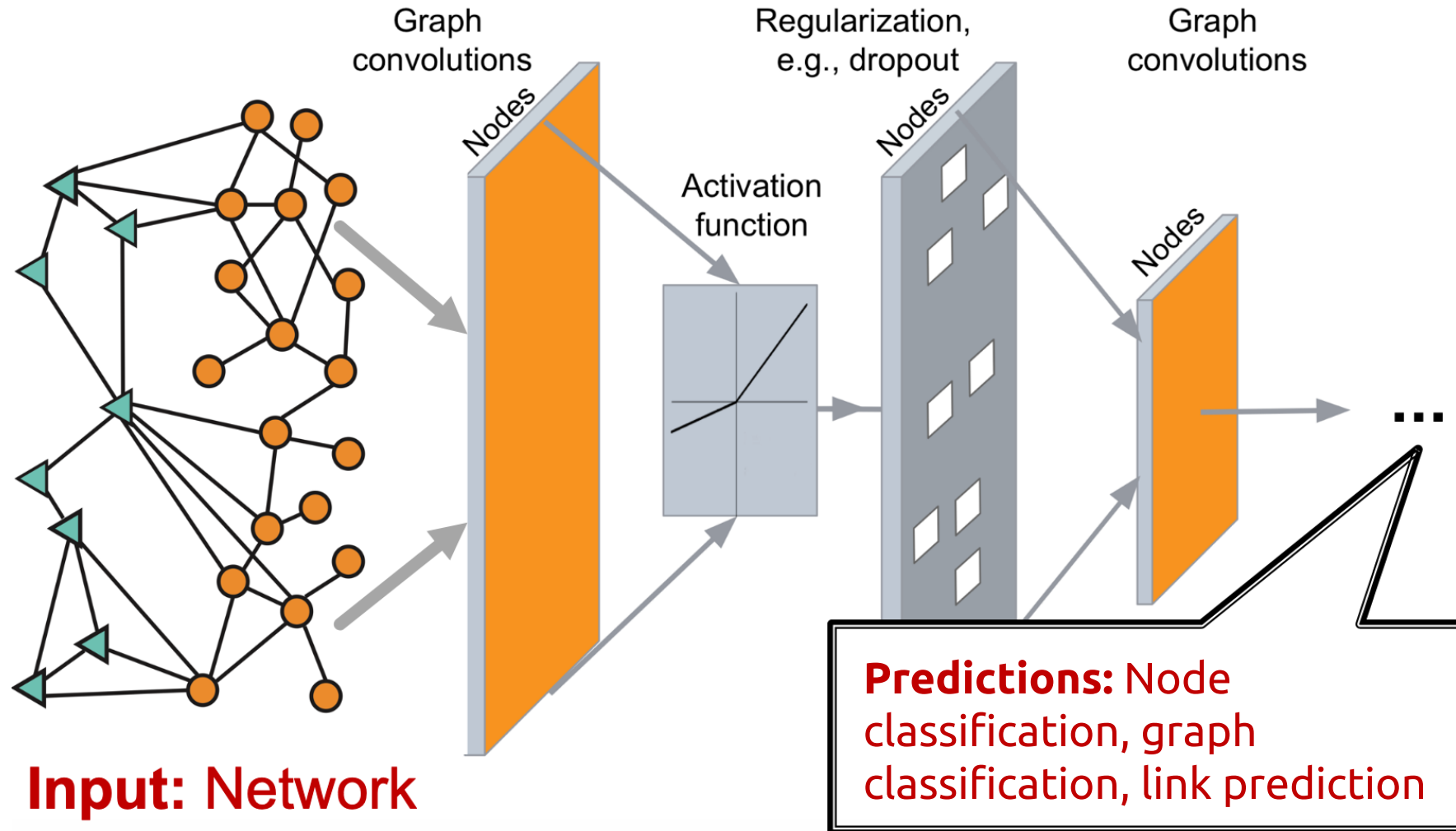
(3) How to use GNNs?

(4) Graph Attention Networks

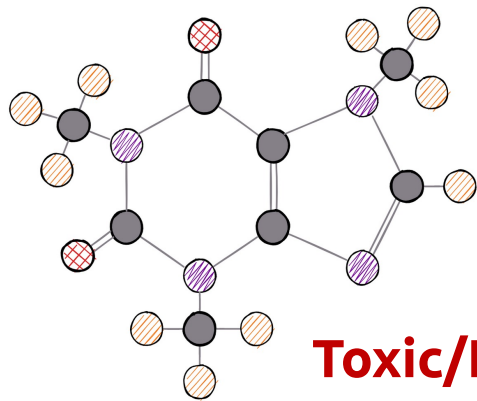
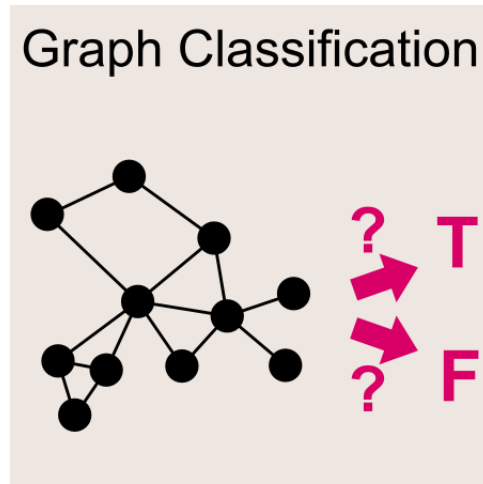
Deep learning framework for graphs



Deep learning framework for graphs

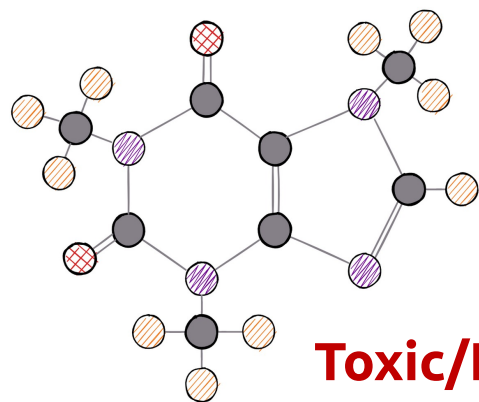
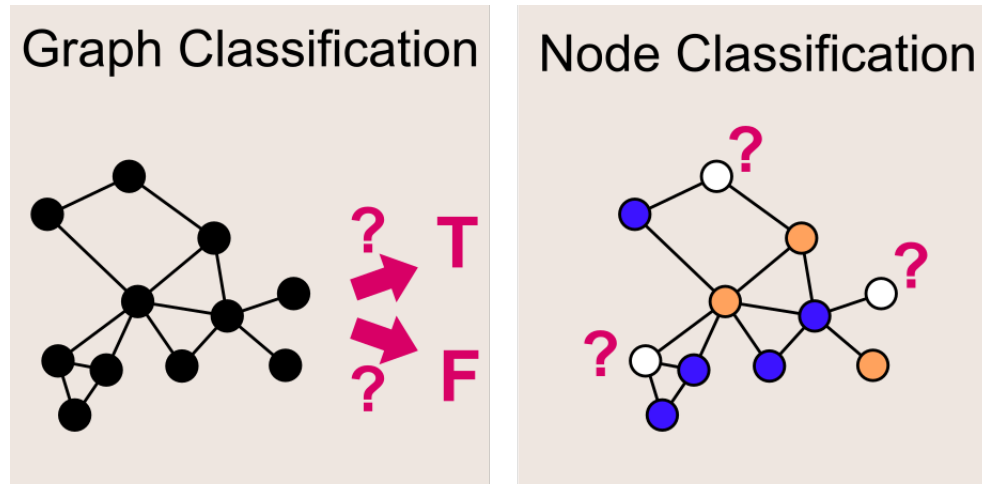


Prediction tasks using GNNs



Toxic/Not toxic?

Prediction tasks using GNNs



Toxic/Not toxic?

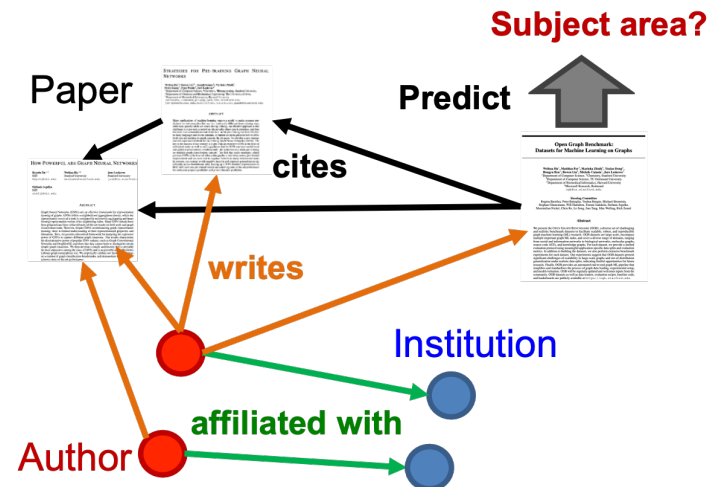
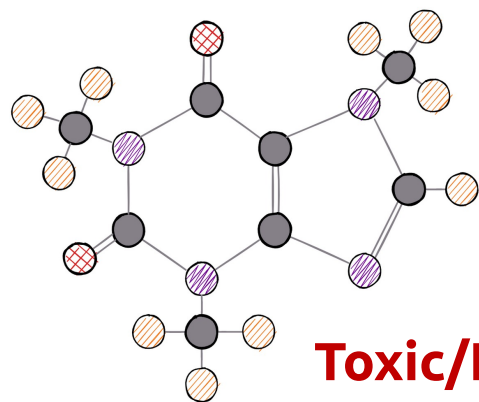
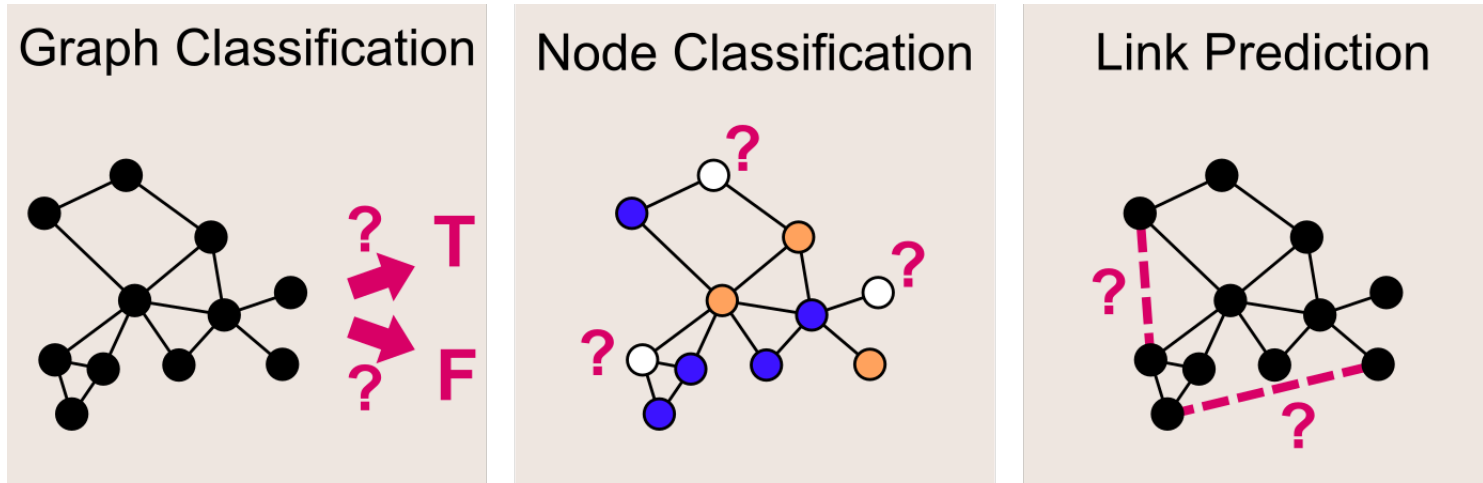


Image courtesy: <https://towardsdatascience.com/graph-convolutional-networks-deep-99d7fee5706f>

<https://memgraph.com/docs/mage/algorithms/machine-learning-graph-analytics/graph-classification-algorithm>

<https://aws.amazon.com/blogs/machine-learning/graph-based-recommendation-system-with-neptune-ml-an-illustration-on-social-network-link-prediction-challenges/>

Prediction tasks using GNNs



Toxic/Not toxic?

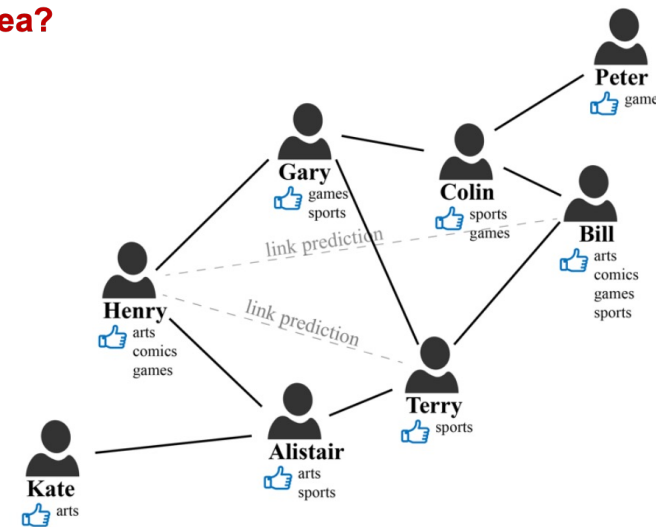
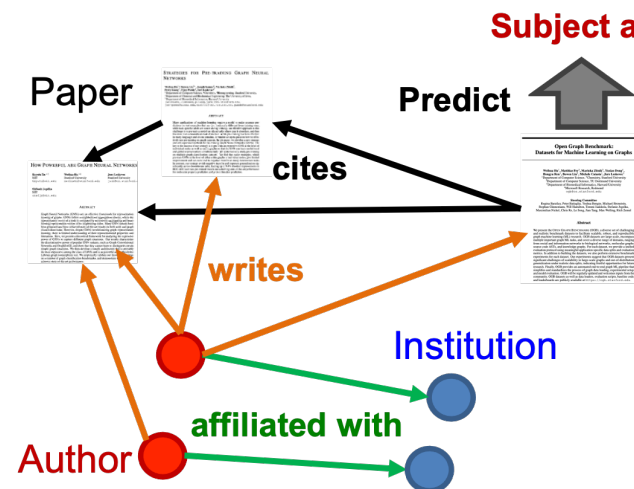
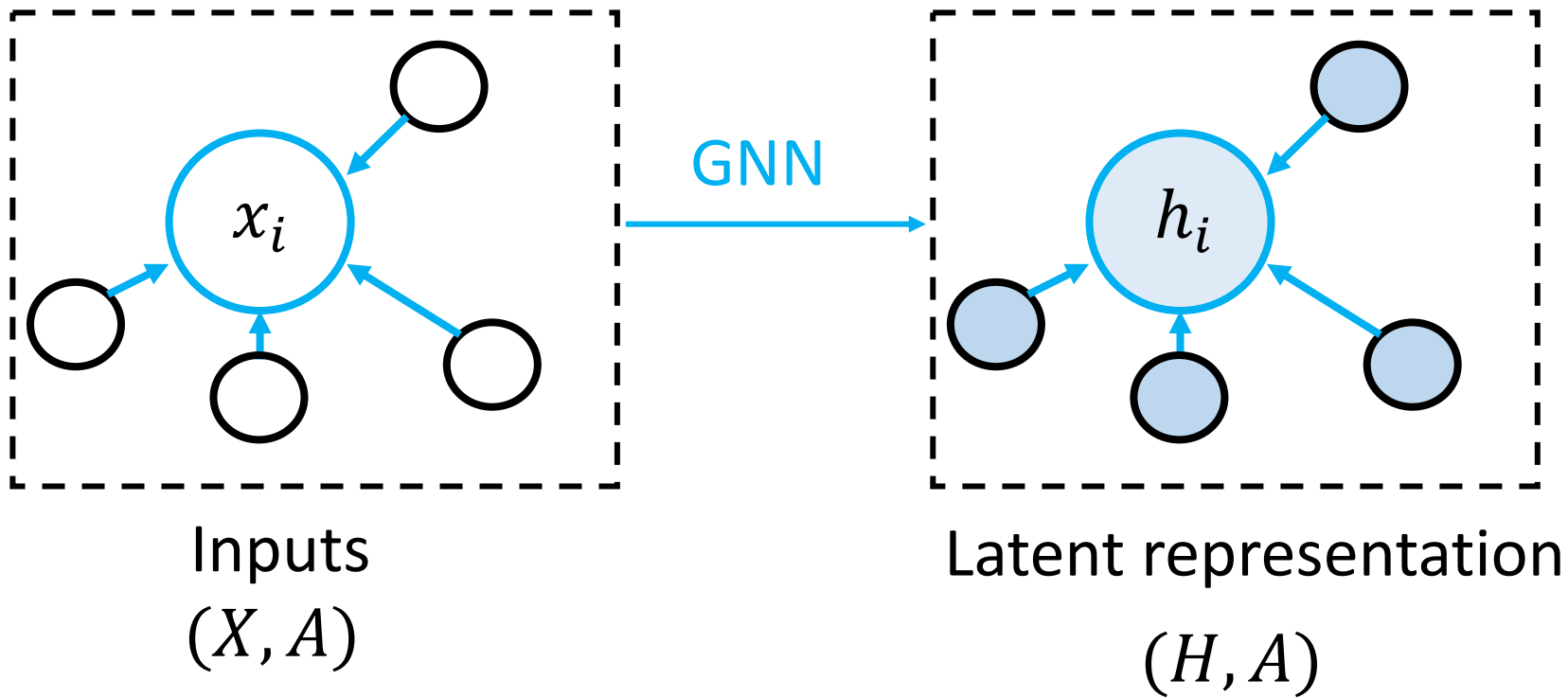


Image courtesy: <https://towardsdatascience.com/graph-convolutional-networks-deep-99d7fee5706f>

<https://memgraph.com/docs/mage/algorithms/machine-learning-graph-analytics/graph-classification-algorithm>

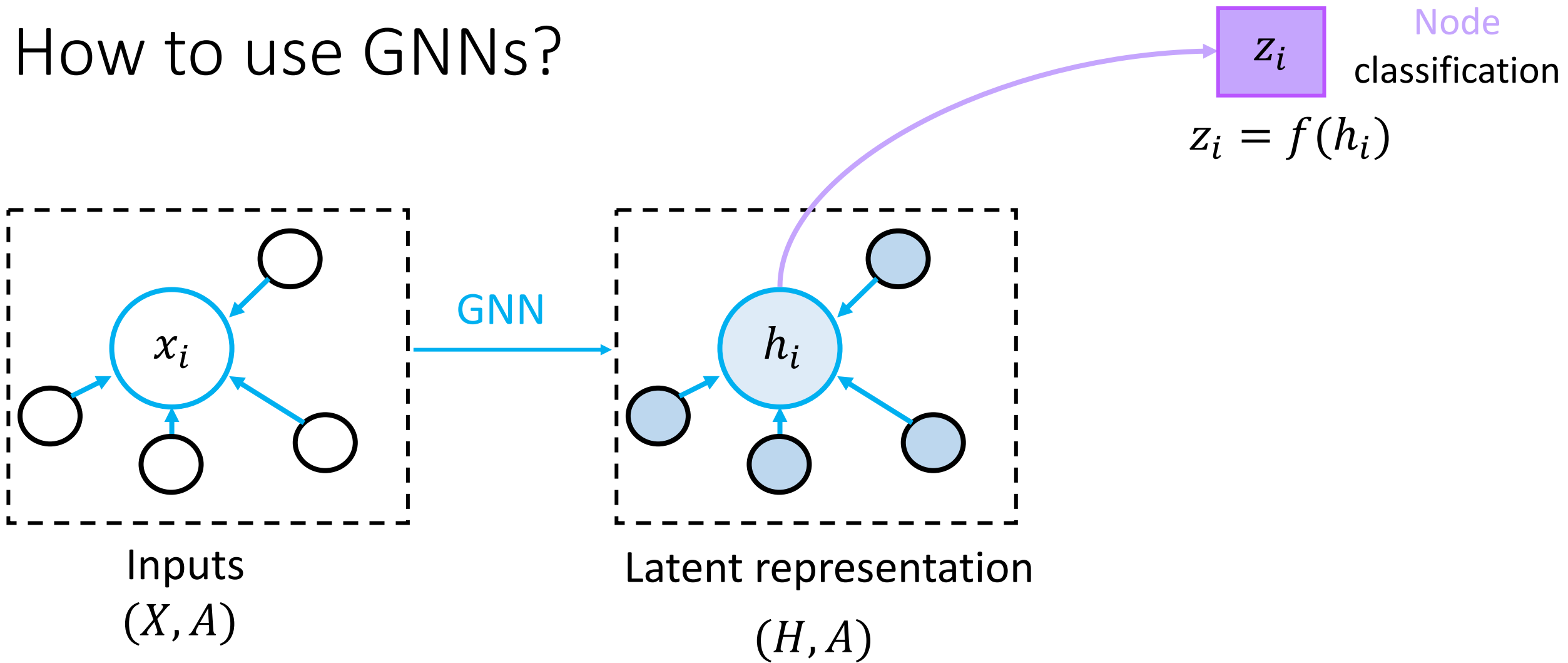
<https://aws.amazon.com/blogs/machine-learning/graph-based-recommendation-system-with-neptune-ml-an-illustration-on-social-network-link-prediction-challenges/>

How to use GNNs?

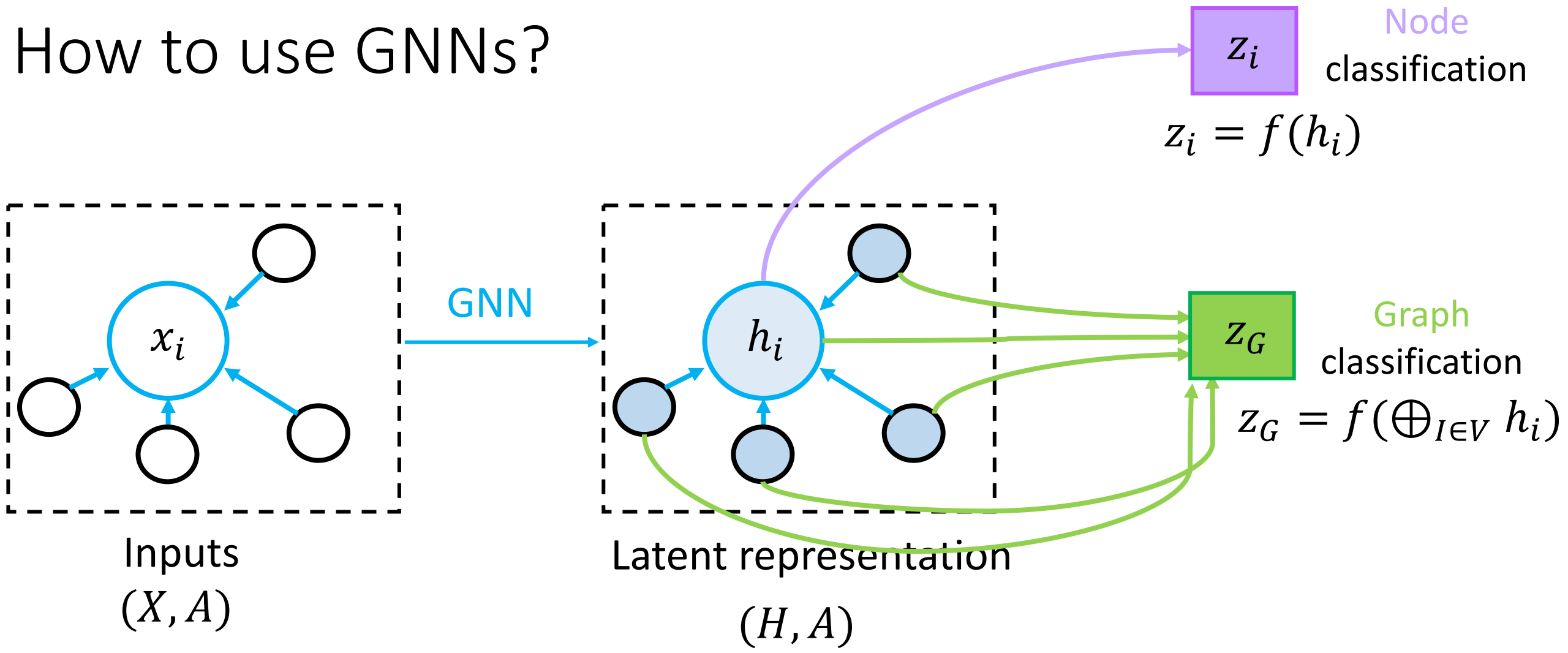


Think-pair-share:
Given the learned representation of the nodes, what do we do next to perform:
(1) Node classification,
(2) graph classification,
(3) link prediction

How to use GNNs?

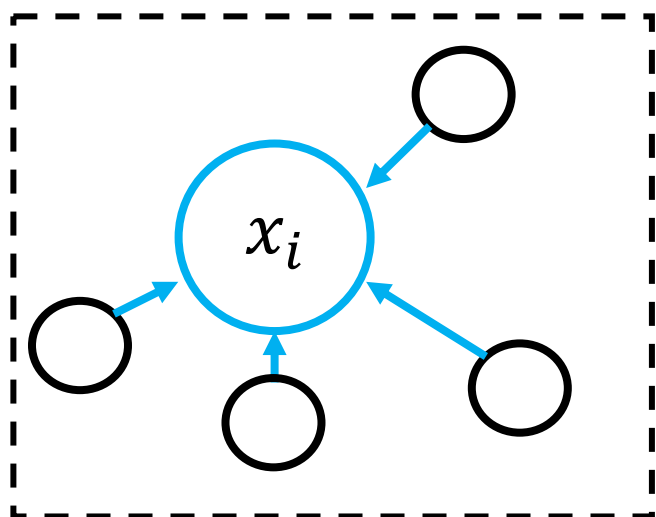


How to use GNNs?



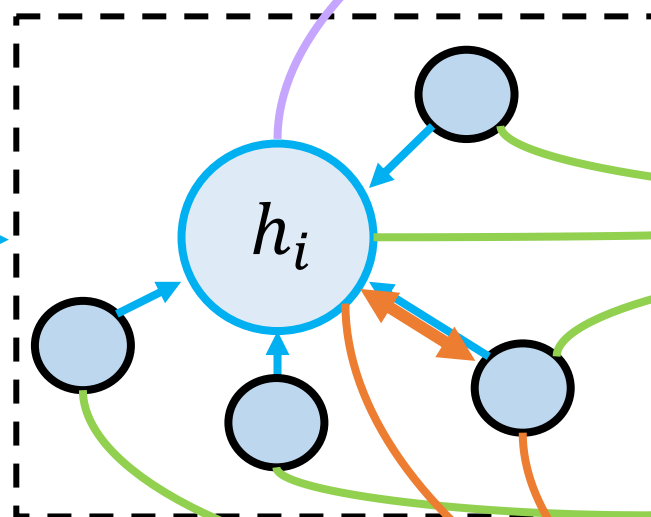
How to use GNNs?

Any questions?



Inputs
(X, A)

GNN



Latent representation
(H, A)

z_i Node classification

$$z_i = f(h_i)$$

z_G Graph classification

$$z_G = f(\bigoplus_{I \in V} h_i)$$

$z_{i,j}$ Link prediction

$$z_{i,j} = f(h_i, h_j, e_{i,j})$$

Today's goal – learn about graph-based neural networks (GNNs)

(1) Motivation

(2) Passing information in Graphs (for NNs)

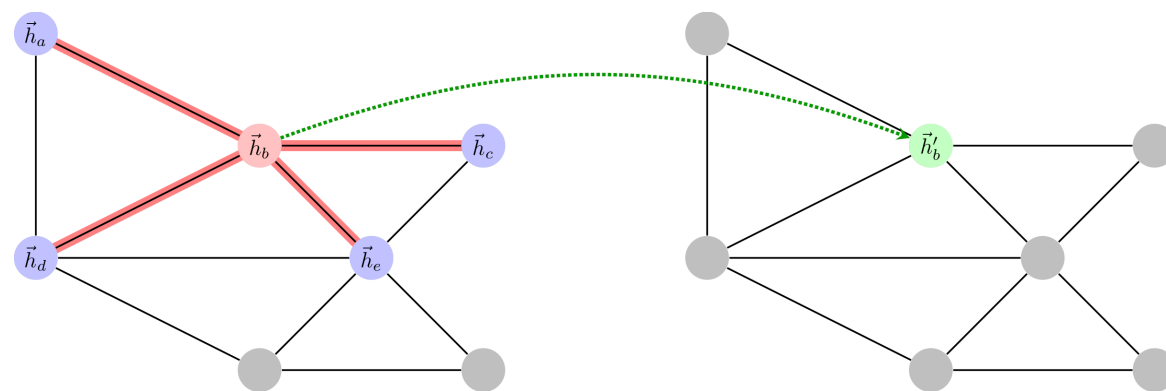
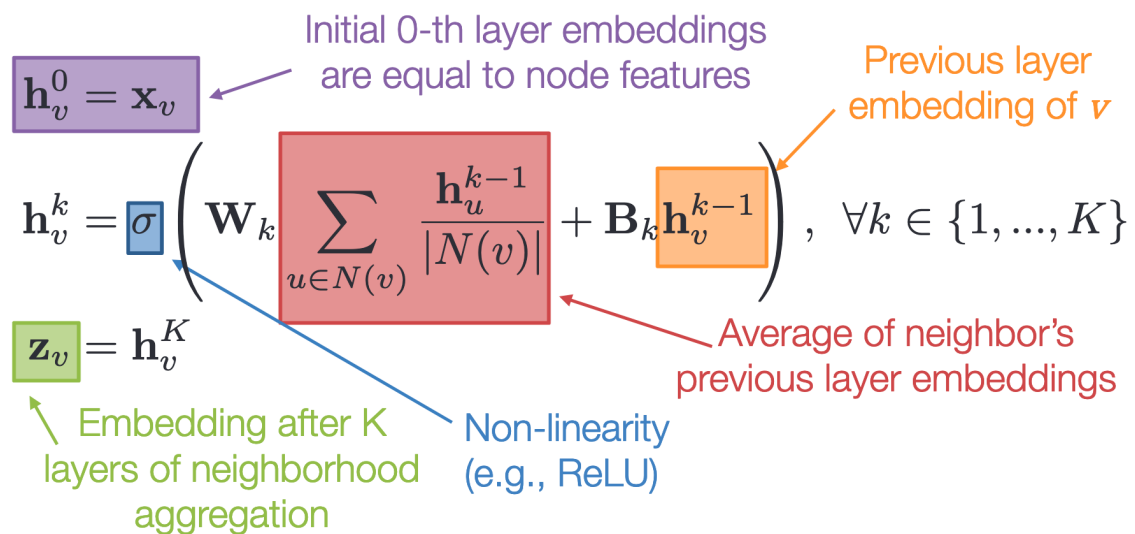
(3) How to use GNNs?

(4) Graph Attention Networks

Our formulation so far

We give equal importance to all nodes through normalized aggregation and learn the weights implicitly during training

What if we could attend to different nodes based on their relevance (explicit weighting) ?



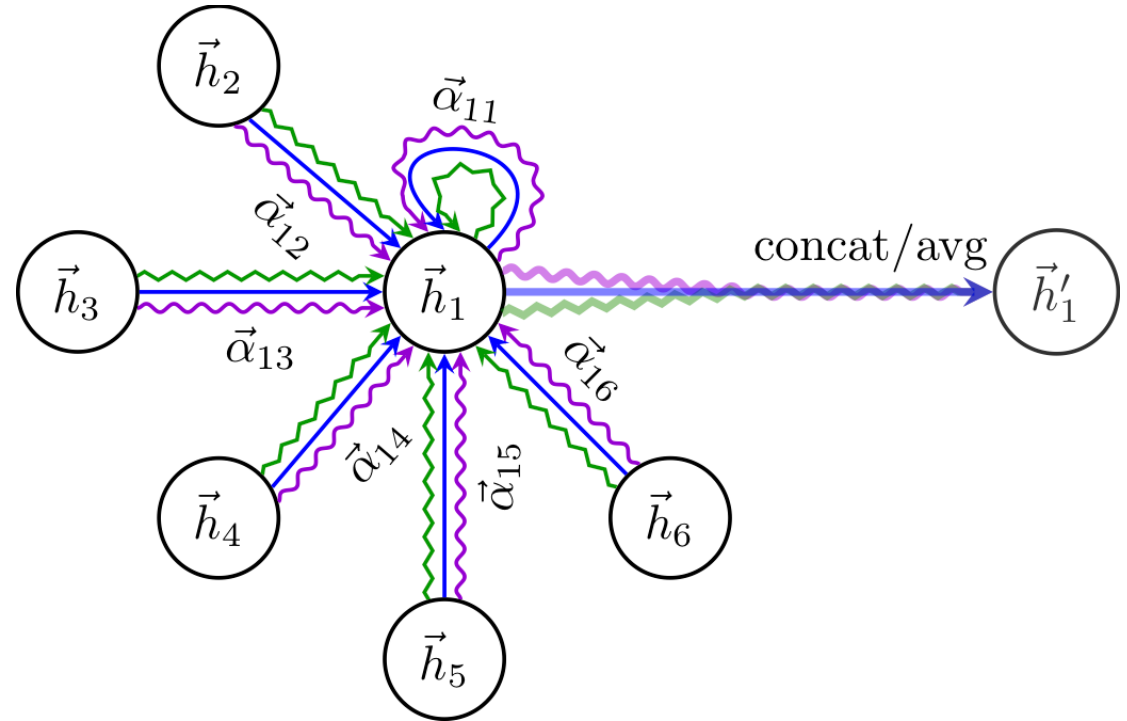
Graph Attention Networks (GAT)

Let's walk through attention steps:

(1) Alignment score $e_{ij} = a(\vec{h}_i, \vec{h}_j)$

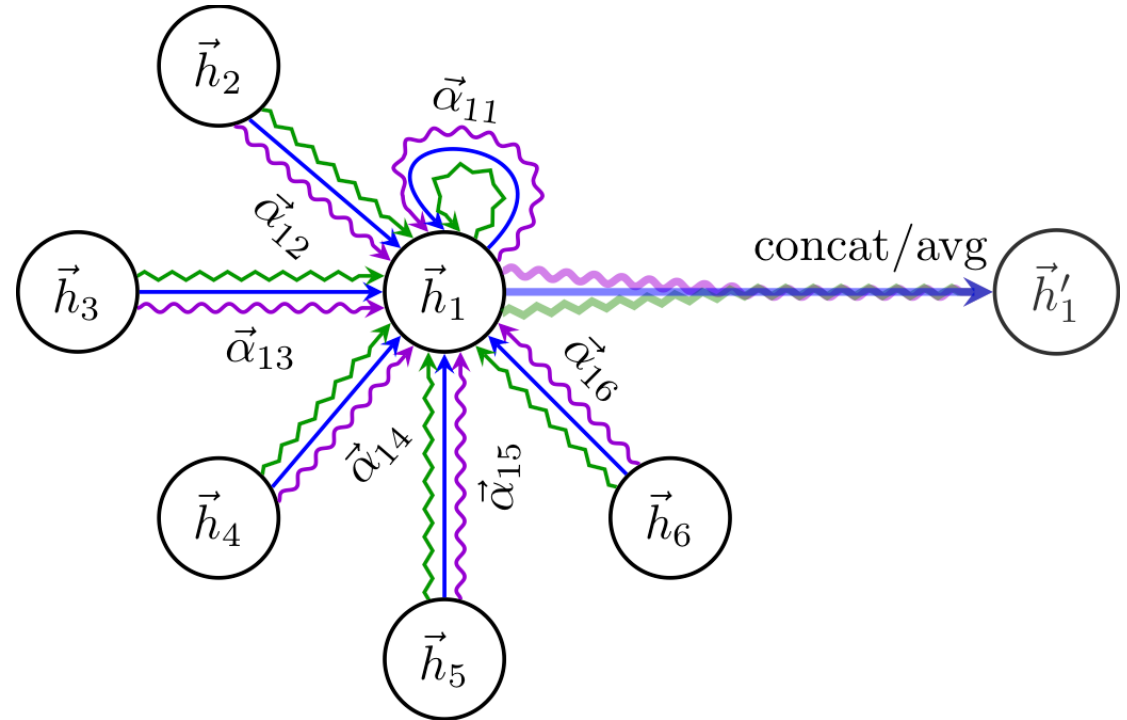
(2) Softmax
$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} \exp(e_{ik})}$$

(3) Weighted aggregation
$$\vec{h}'_i = \parallel_{k=1}^K \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}^k \vec{h}_j \right)$$



Graph Attention Networks (GAT)

Method	PPI
Random	0.396
MLP	0.422
GraphSAGE-GCN (Hamilton et al., 2017)	0.500
GraphSAGE-mean (Hamilton et al., 2017)	0.598
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612
GraphSAGE-pool (Hamilton et al., 2017)	0.600
GraphSAGE*	0.768
Const-GAT (ours)	0.934 ± 0.006
GAT (ours)	0.973 ± 0.002

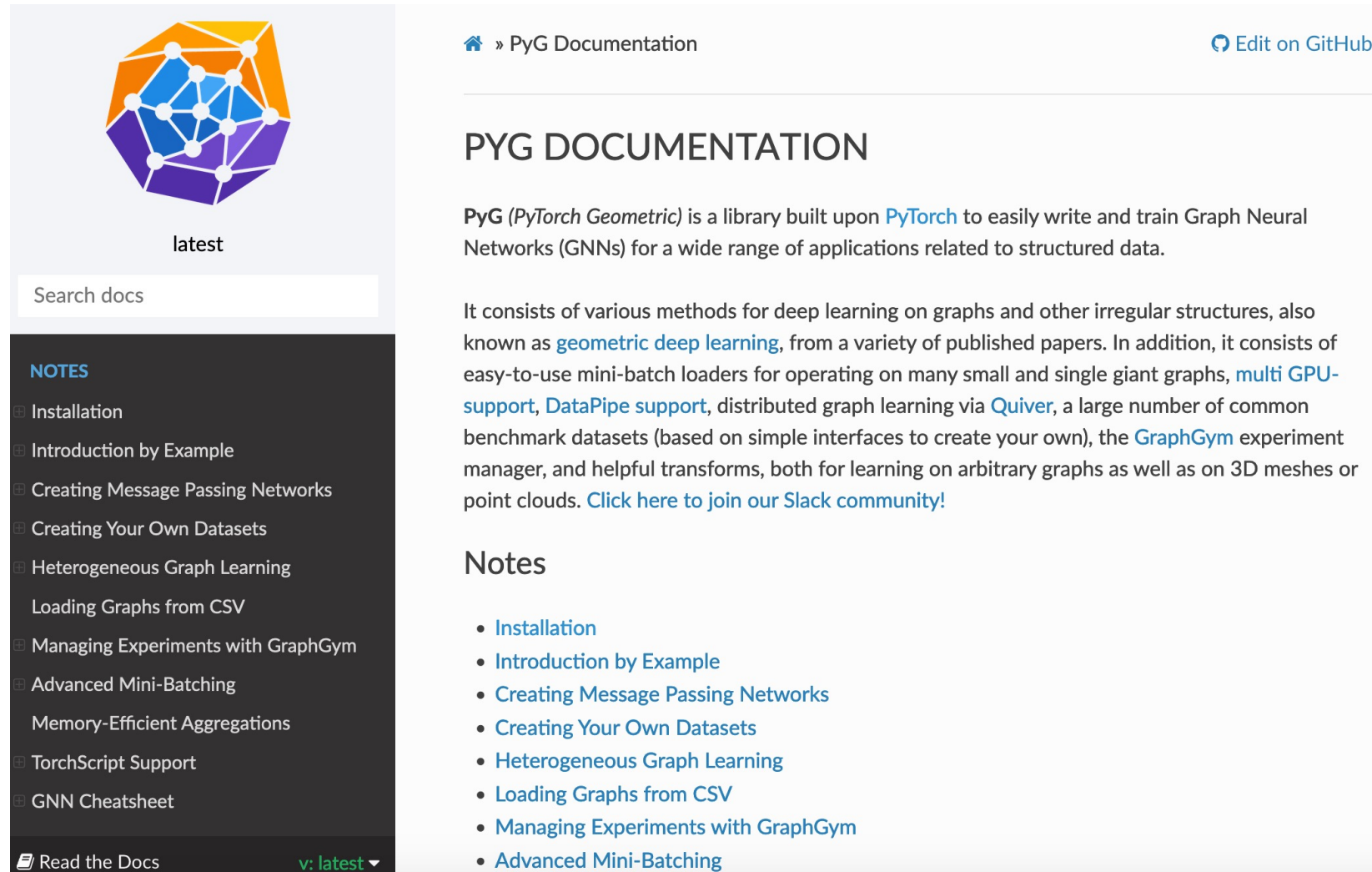



Protein-Protein Interaction dataset:

- consists of graphs corresponding to different human tissue
- contains 20 graphs for training, 2 for validation and 2 for testing.

How to implement a GNN?

<https://pytorch-geometric.readthedocs.io/en/latest/>




latest

Search docs

NOTES

- Installation
- Introduction by Example
- Creating Message Passing Networks
- Creating Your Own Datasets
- Heterogeneous Graph Learning
 - Loading Graphs from CSV
- Managing Experiments with GraphGym
- Advanced Mini-Batching
 - Memory-Efficient Aggregations
- TorchScript Support
- GNN Cheatsheet

Read the Docs v: latest

» PyG Documentation [Edit on GitHub](#)

PYG DOCUMENTATION

PyG (*PyTorch Geometric*) is a library built upon [PyTorch](#) to easily write and train Graph Neural Networks (GNNs) for a wide range of applications related to structured data.

It consists of various methods for deep learning on graphs and other irregular structures, also known as [geometric deep learning](#), from a variety of published papers. In addition, it consists of easy-to-use mini-batch loaders for operating on many small and single giant graphs, [multi GPU-support](#), [DataPipe support](#), distributed graph learning via [Quiver](#), a large number of common benchmark datasets (based on simple interfaces to create your own), the [GraphGym](#) experiment manager, and helpful transforms, both for learning on arbitrary graphs as well as on 3D meshes or point clouds. [Click here to join our Slack community!](#)

Notes

- [Installation](#)
- [Introduction by Example](#)
- [Creating Message Passing Networks](#)
- [Creating Your Own Datasets](#)
- [Heterogeneous Graph Learning](#)
- [Loading Graphs from CSV](#)
- [Managing Experiments with GraphGym](#)
- [Advanced Mini-Batching](#)

Recap

Graph-based Neural Networks (GNNs)

Motivation

Message Passing

GraphSAGE formulation

Practical aspects of GNNs

Prediction tasks

How to use GNNs

Graph Attention Networks

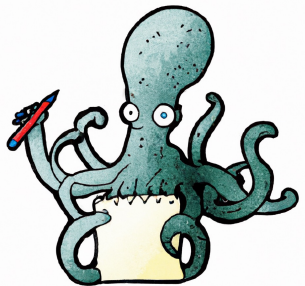
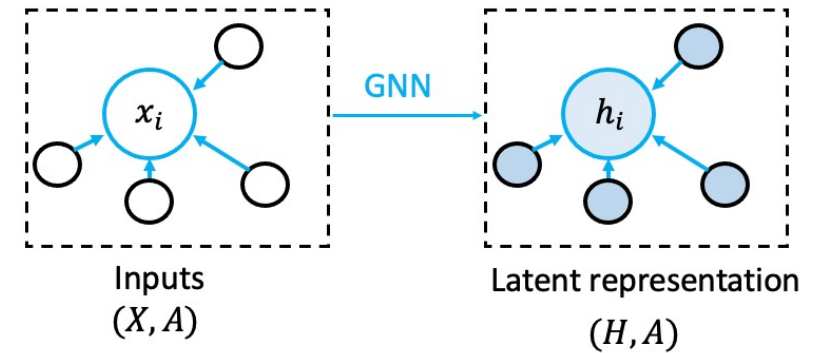


Image credit: [Medium](#)

Social Networks



See you on
DL Day(s)! 😊



Extra reading material

- [Summary of graph convolution using graph Fourier transform](#)
- [Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering](#)
- [Semi-Supervised Classification with Graph Convolutional Networks](#)