Introduction to Graph Neural Networks

Alina Lazar[1] and Xiangyang Yu[2]

[1]School of Computer Science, Information and Engineering Technology, Youngstown State University

[2]Physics and X-Ray Science Computing Group, Lawrence Berkeley National Laboratory

alazar@ysu.edu, xju@lbl.gov

slido

Join at slido.com
#1152677

ⓘ Start presenting to display the joining instructions on this slide.

Tutorial Demo

https://github.com/alinutzal/IntrotoGNN

Part 1

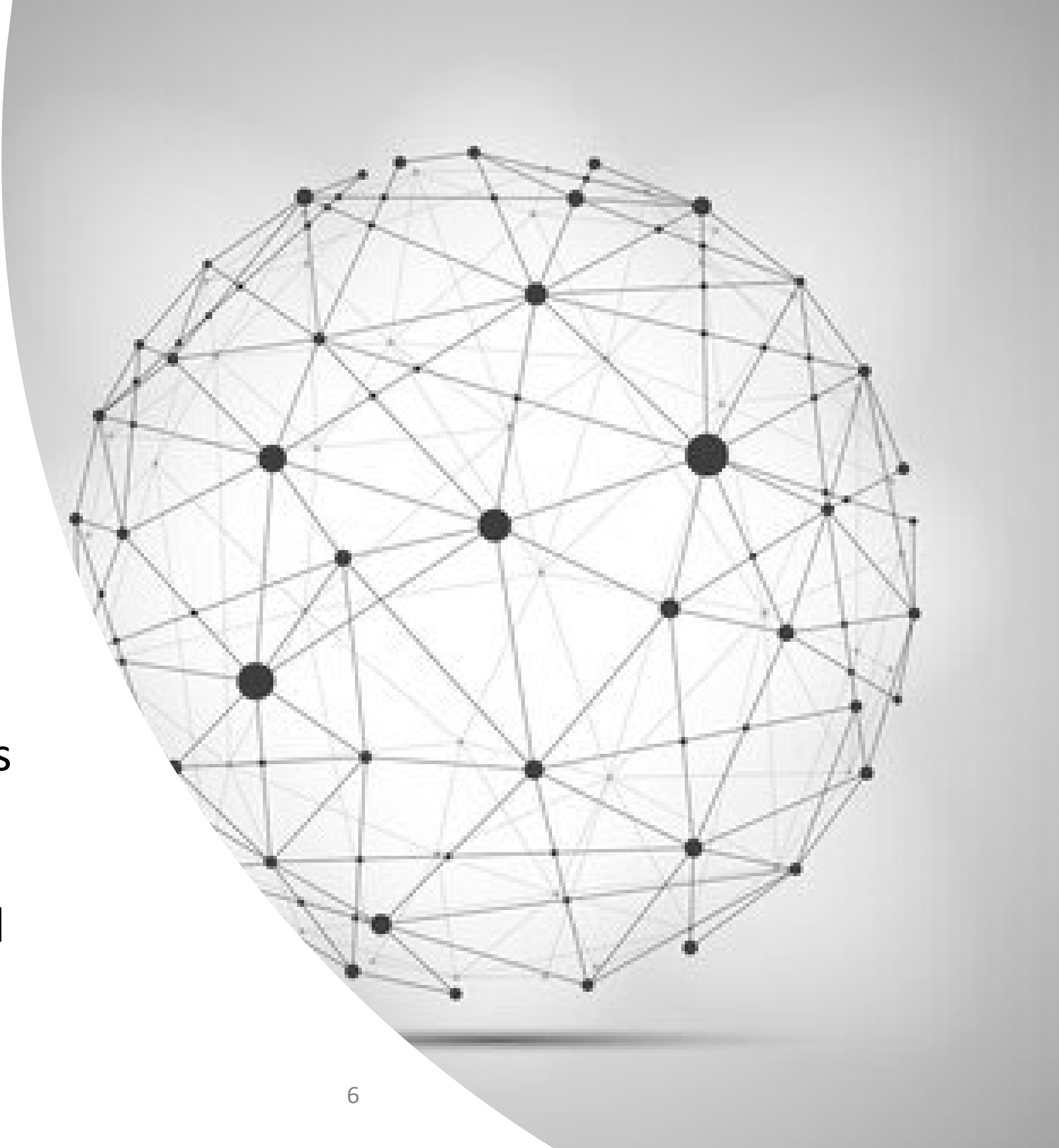
- Graphs and graph structured data
- Node, edge and graph level tasks
- Simple graph neural networks (GNN)
- Graph convolutional neural networks (GCNNs)

Part 2 – Hands-on

- GNN Frameworks
- Introduction to Graphs used in the tutorial
- Hands-on tutorial:
  - Node classification
  - Edge classification

Overview

- **Graph Neural Networks** (GNNs) are a subset of deep learning methods
- GNNs make useful predictions on **graph representations**
- Many **practical applications** from areas such as physics simulations, object detection and recommendation systems
- GNNs are one of **fastest growing** and most active research topic
- No prior knowledge of GNNs is required



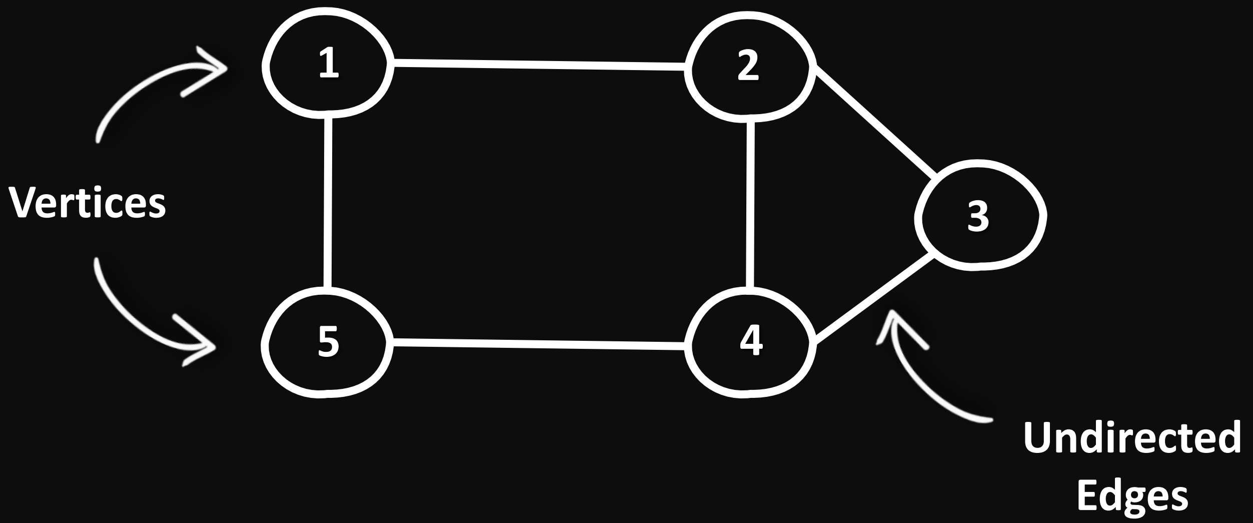
slido

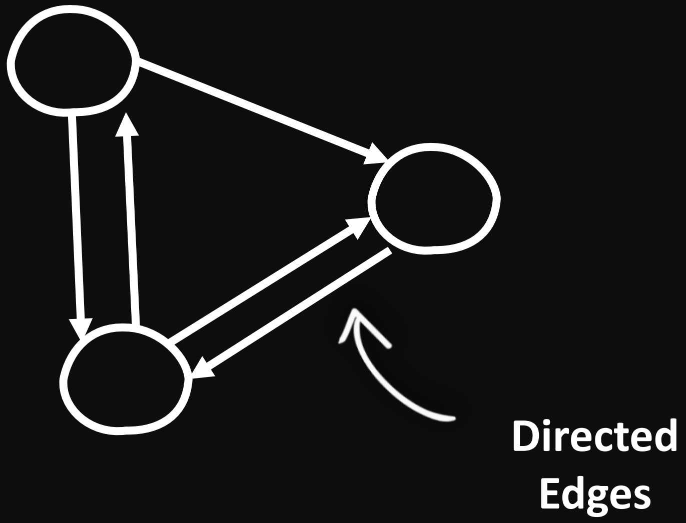
What is your background?

ⓘ Start presenting to display the poll results on this slide.

Graph Representation

- Structure
  - Nodes/Vertices V
  - Edges/Links E
- Graph $G = (V, E)$
- V = {1, 2, 3, 4, 5}
- E = { (1, 2), (2, 3), (2, 4), (3, 4), {4, 5)}
- Types
  - Undirected
  - Directed

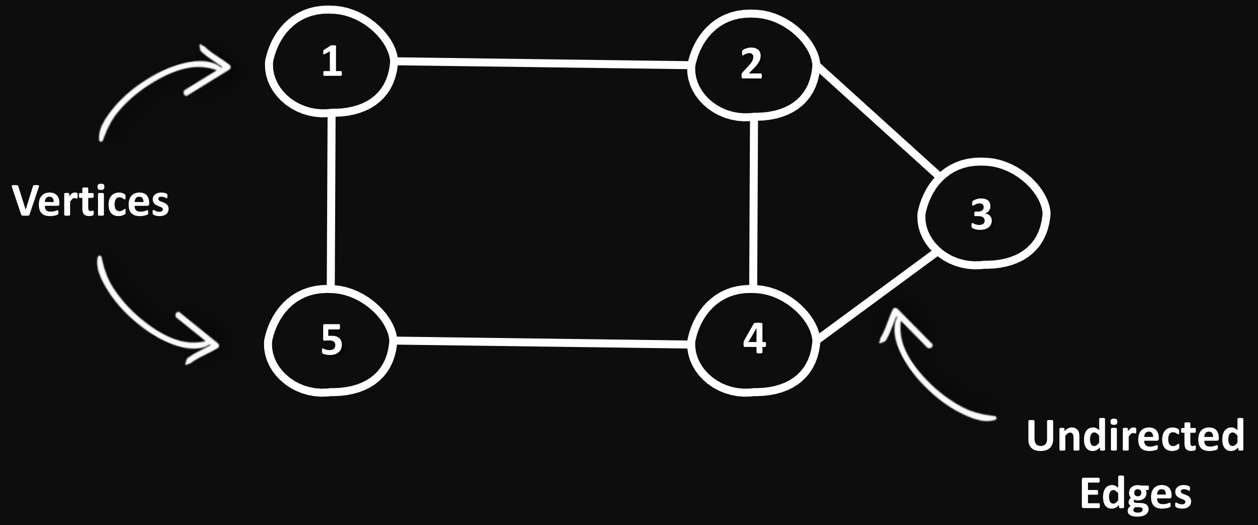




Adjacency Matrix

- $A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

- $A_{i,j} = A_{j,i}$ - A is a symmetric matrix



Representing Problems as Graphs

Chemical Compounds (a)

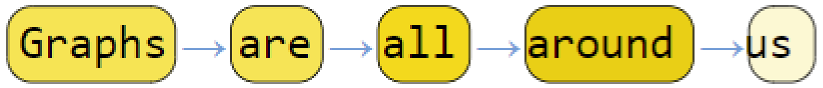
Images (b)

Text (c)

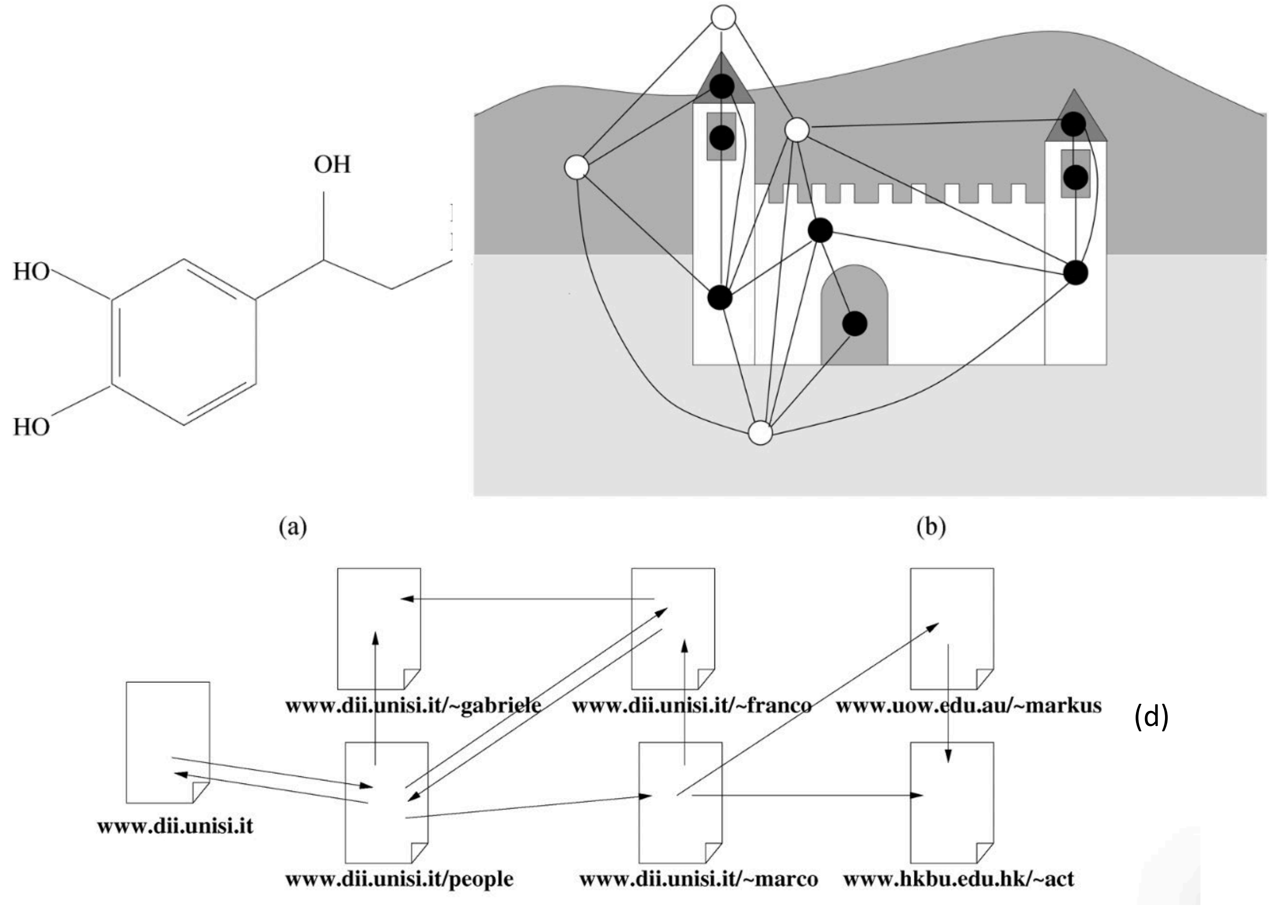
Web Pages (d)

Times Series

Sensors



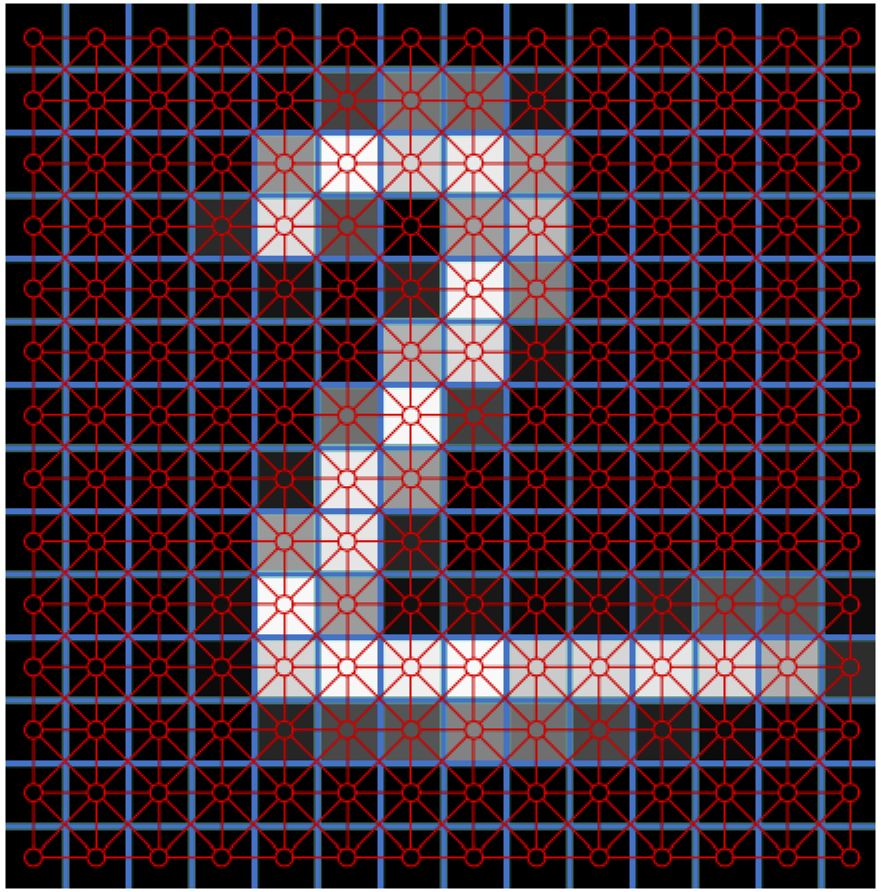
(c)



(d)

Image Source:"The Graph Neural Network Model," in IEEE Transactions on Neural Networks

Images as Graphs



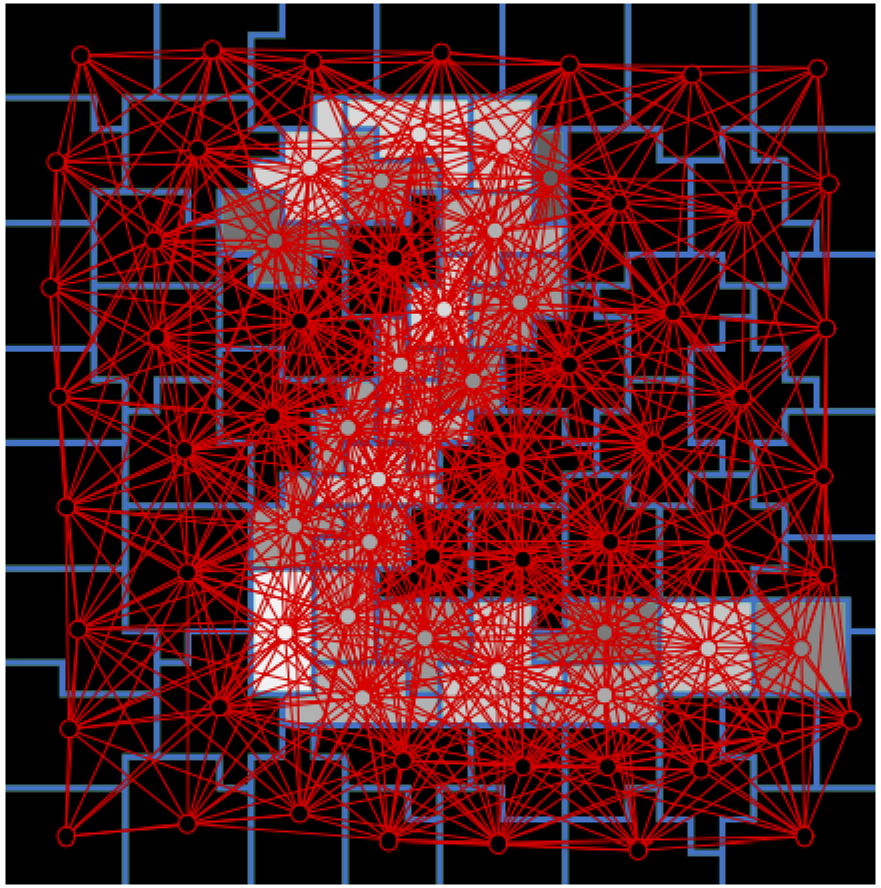


Image source:https://arxiv.org/pdf/1611.08402.pdf

Text as Graphs

*In mathematics, **graph** theory is the study of graphs, which are mathematical structures used to model pairwise relations between objects. A **graph** in this context is made up of vertices, also called nodes or points, which are connected by edges, also called links or lines. A distinction is made between undirected graphs, where edges link two vertices symmetrically, and directed graphs, where edges link two vertices asymmetrically. Graphs are one of the principal objects of study in discrete mathematics.*

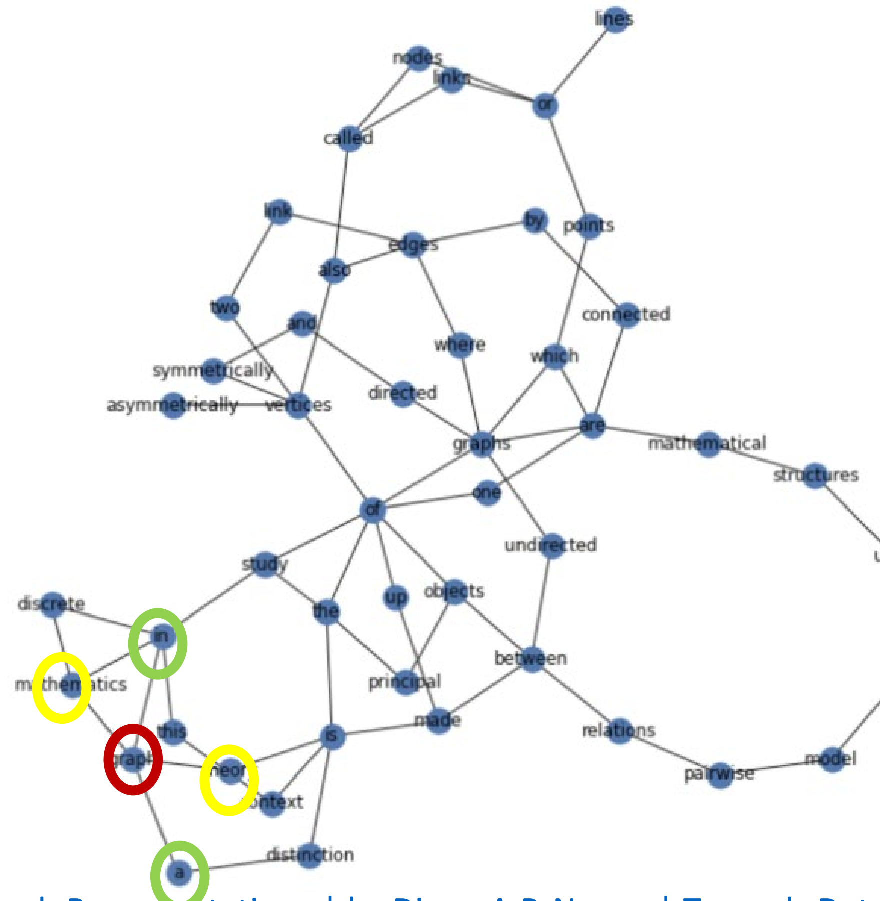
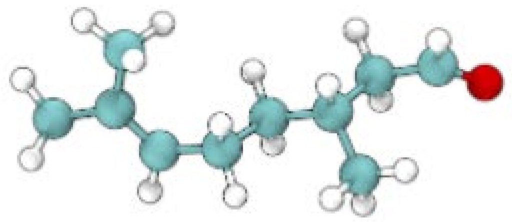
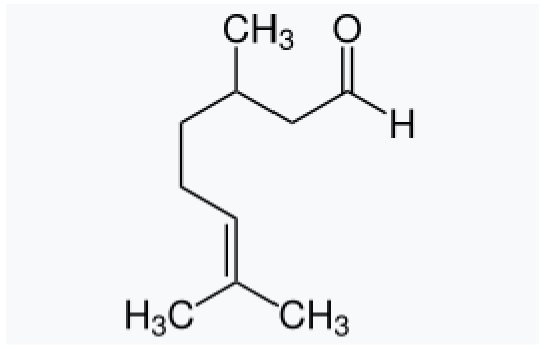
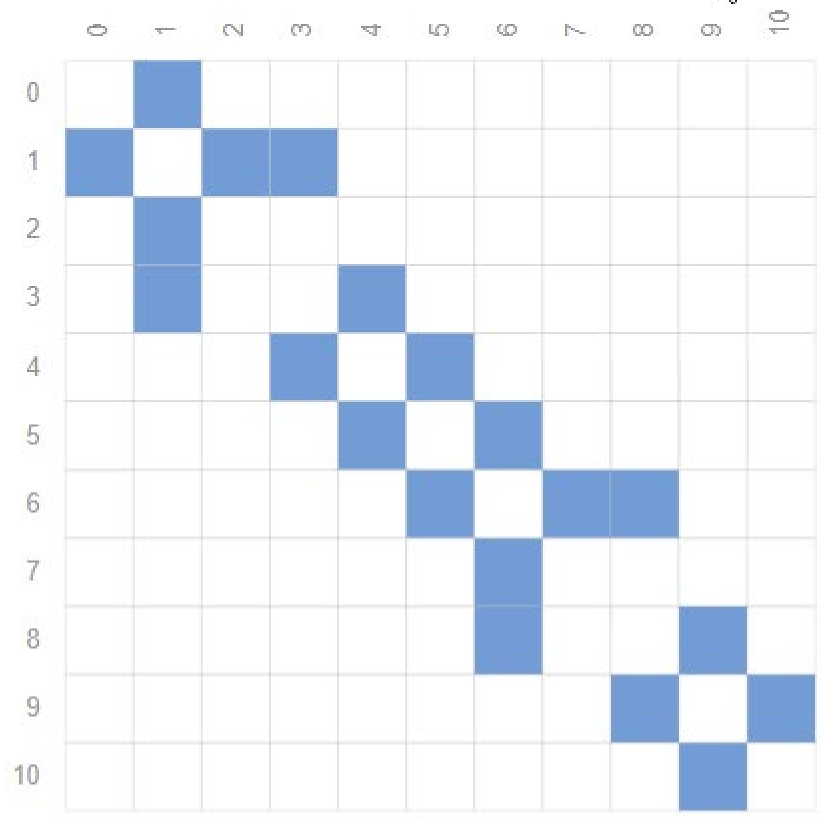


Image source: Structuring Text with Graph Representations | by Diogo A.P. Nunes | Towards Data Science

Molecules as Graphs







	0	1	2	3	4	5	6	7	8	9	10
0		■									
1	■		■	■							
2		■									
3		■			■						
4				■		■					
5					■		■				
6						■		■	■		
7							■				
8							■			■	
9									■		■
10										■	

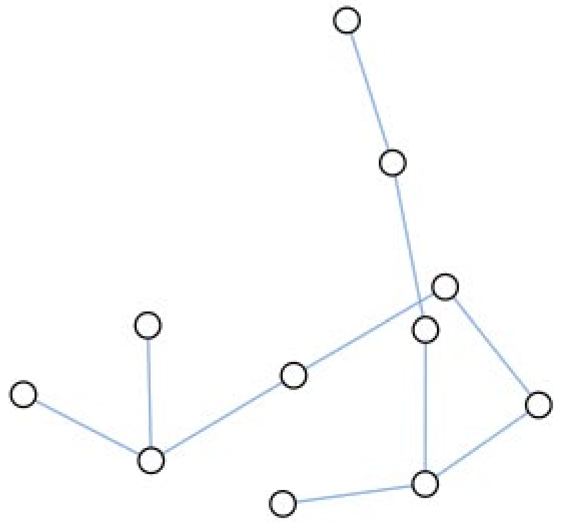


Image source: [A Gentle Introduction to Graph Neural Networks (distill.pub)]

Social Networks as Graphs

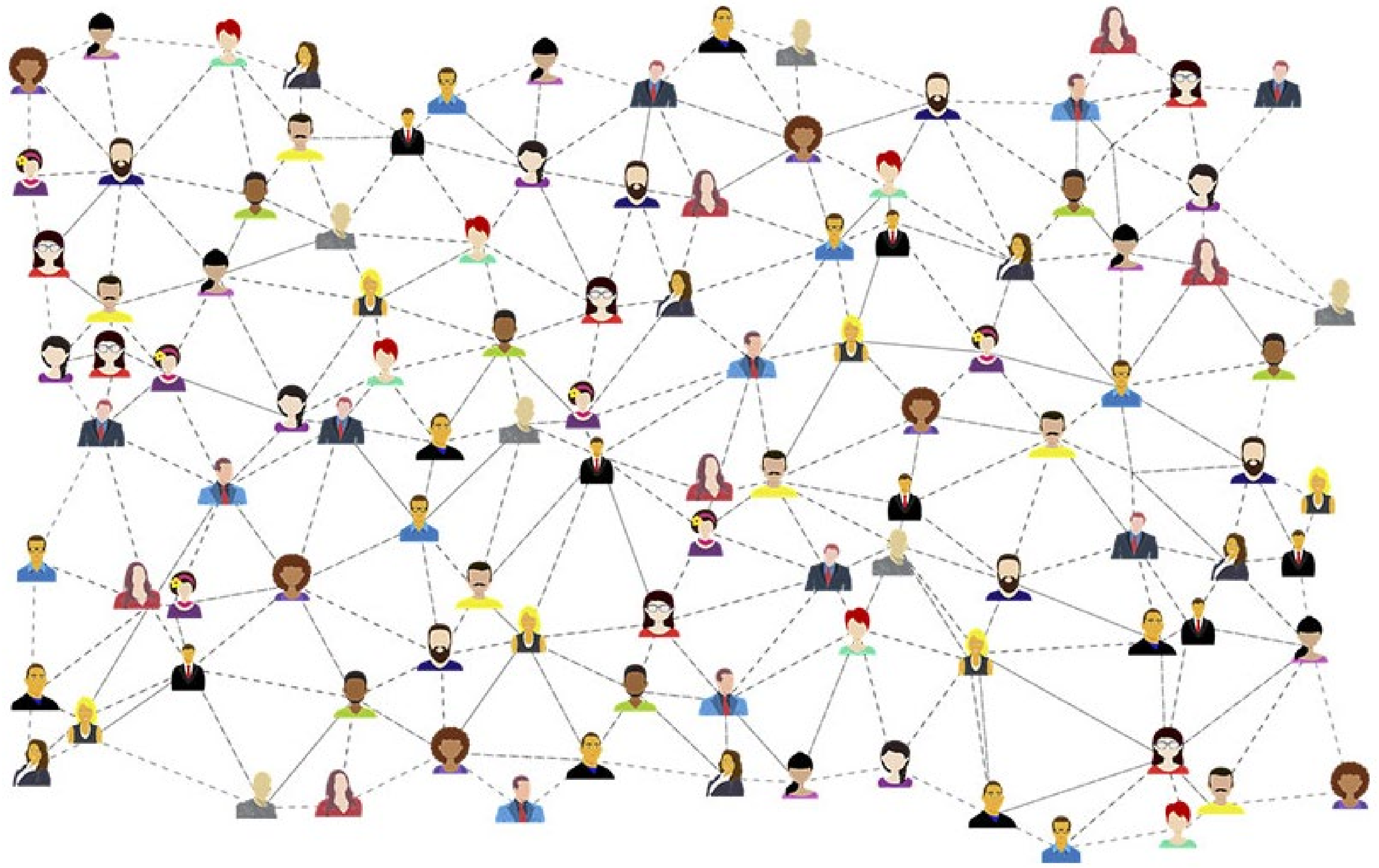


Image source: [2.6 million+ Stunning Free Images to Use Anywhere (pixabay.com)]

Citation Networks as Graphs

Patterns of Citations in German Political Science

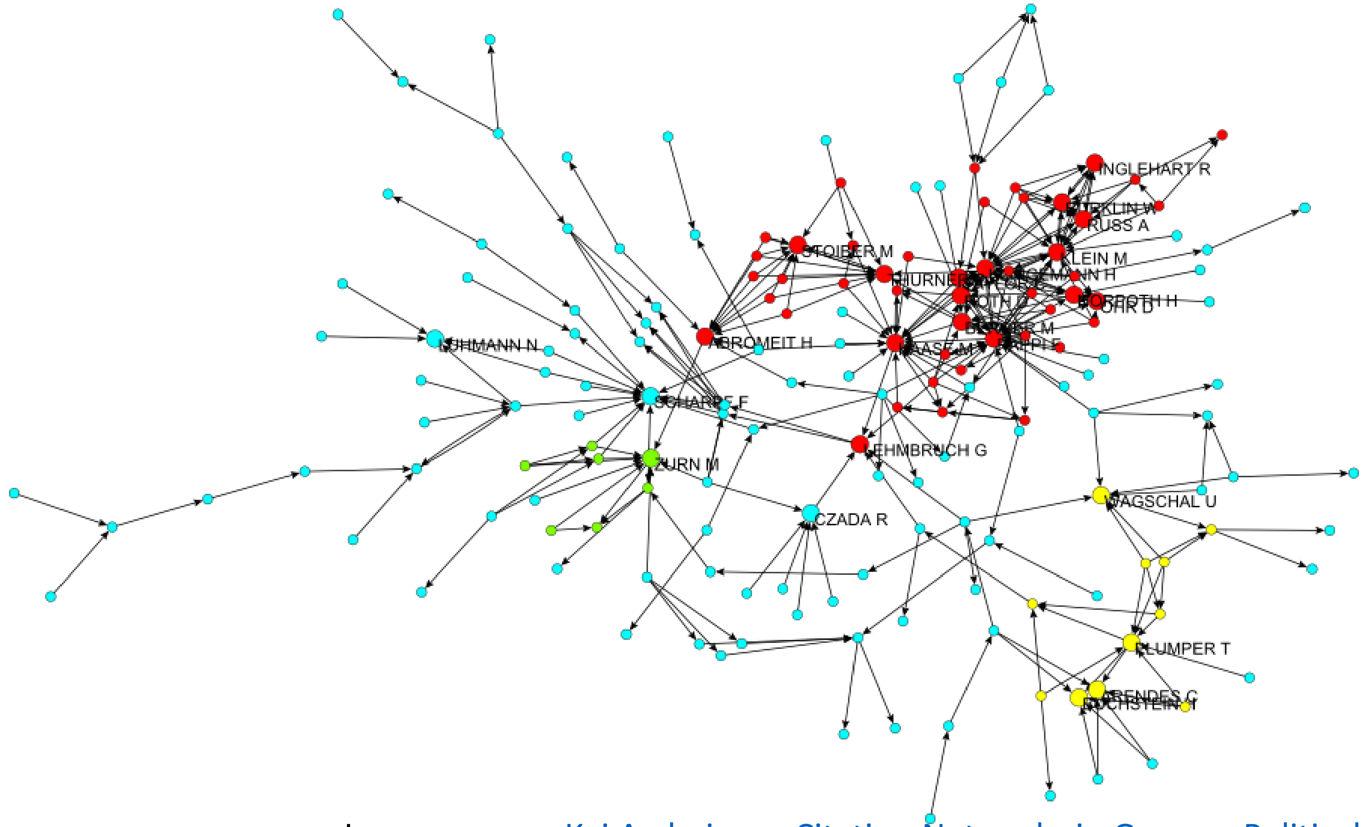


Image source: [Kai Arzheimer: Citation Networks in German Political Science (kai-arzheimer.com)]

Other Examples of Graph Representations

- Objects in visual scenes
- Machine learning models
- Programming code
- Computer networks
- Time series
- Mathematical equations
- Physical phenomena

Machine Learning

Deep Learning – Neural Networks

Using Graphs in Machine Learning

- How to **represent graphs** to be compatible with neural networks?
- Graphs have up to four types of **information**: nodes, edges, global-context and connectivity
- **Nodes** can be represented by a node feature matrix *N*
- Representing a graph's **connectivity** as an adjacency matrix drawbacks
  - Very sparse adjacency matrices
  - There are many adjacency matrices representing the same connectivity
- A memory-efficient way of representing sparse matrices is to use **adjacency or edge lists**

What types of problems have graph structured data?

Tasks on graph-structured data can be grouped into three main groups:

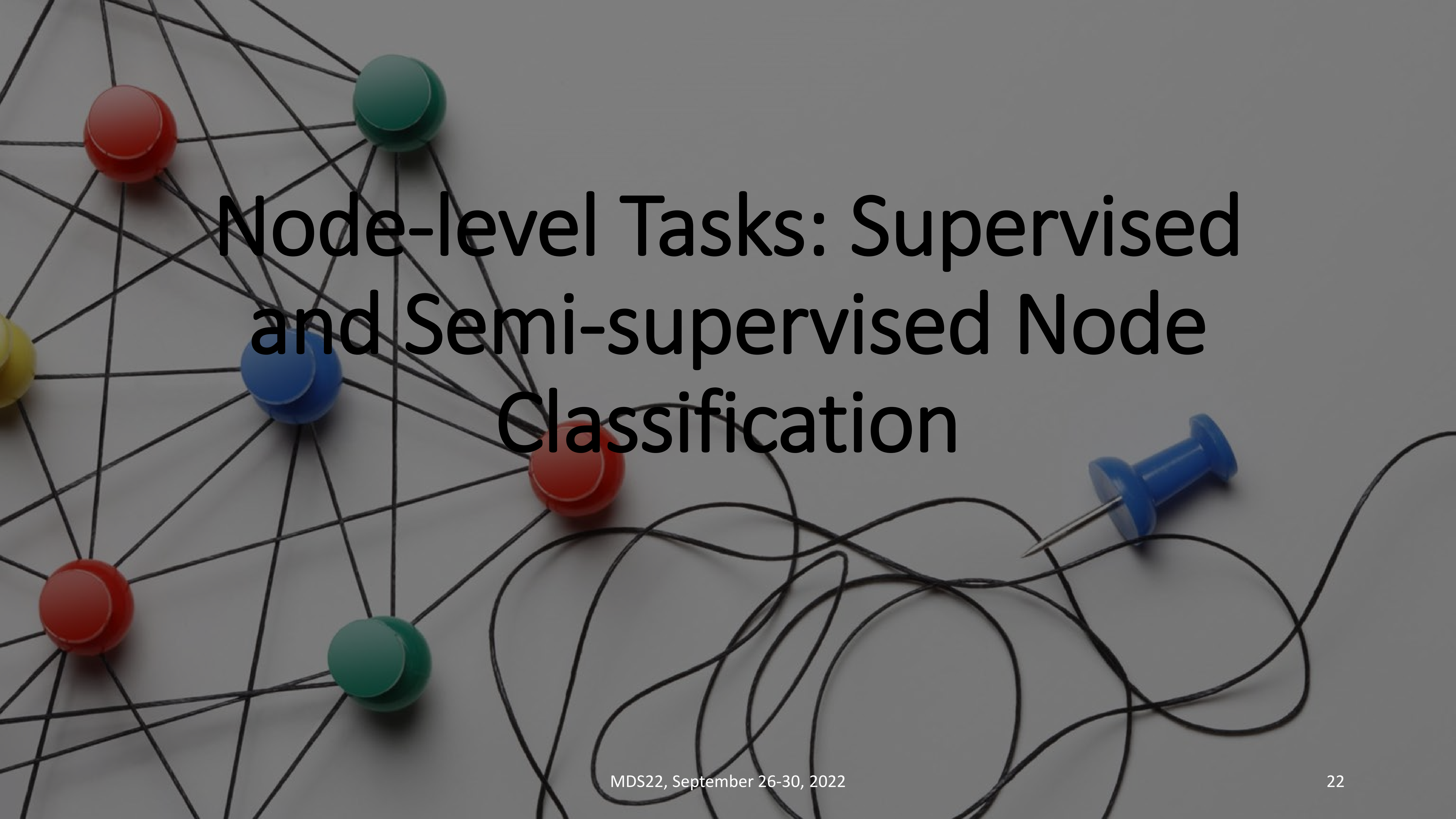
Node-level	Edge-level	Graph-level

What level the prediction task is performed.

slido

Audience Q&A Session

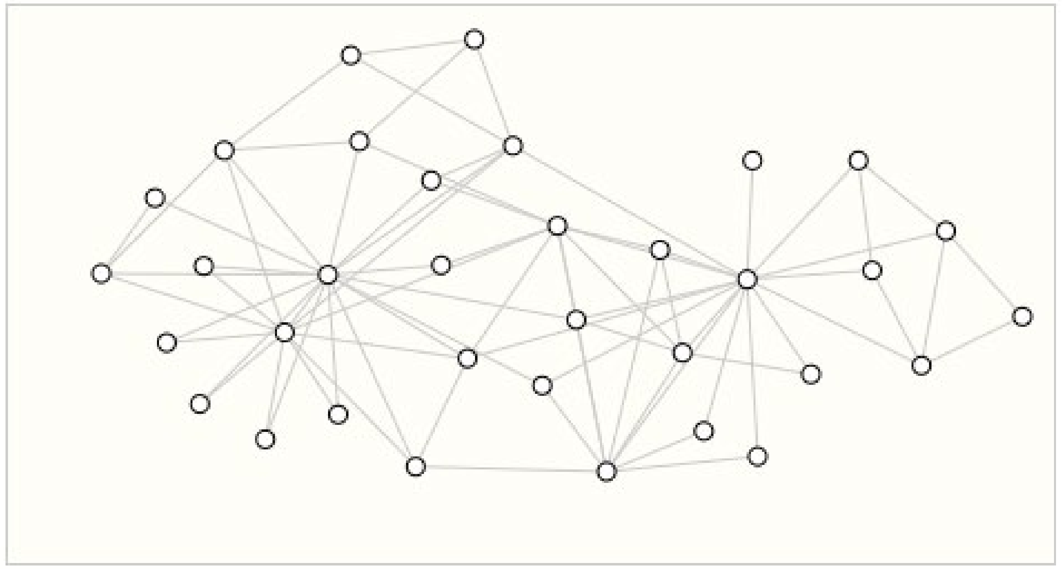
ⓘ Start presenting to display the audience questions on this slide.



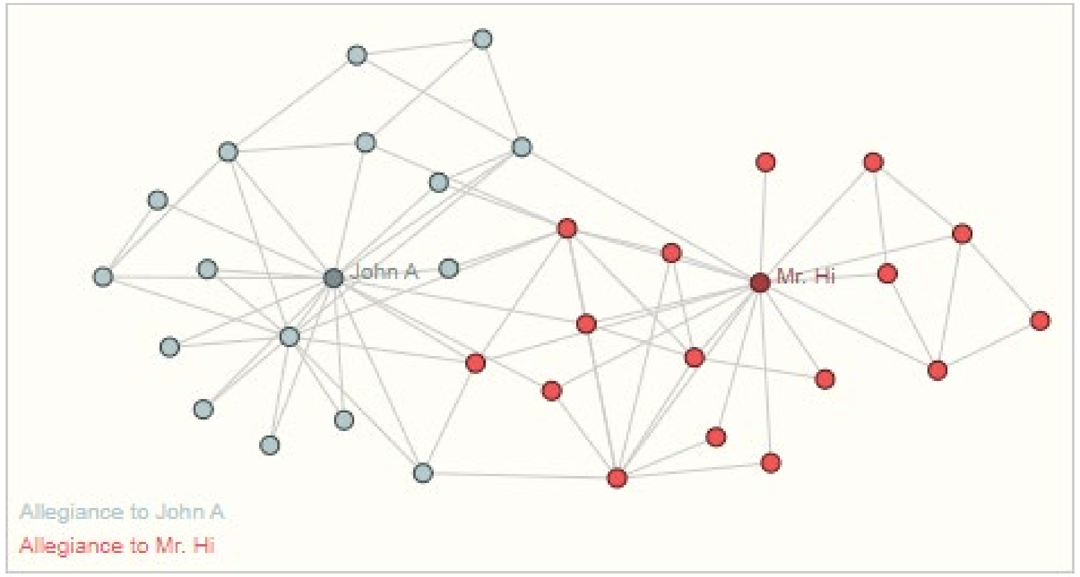
Node-level Tasks: Supervised and Semi-supervised Node Classification

Node Level Tasks

Node-level tasks predict a label for each node within a graph.

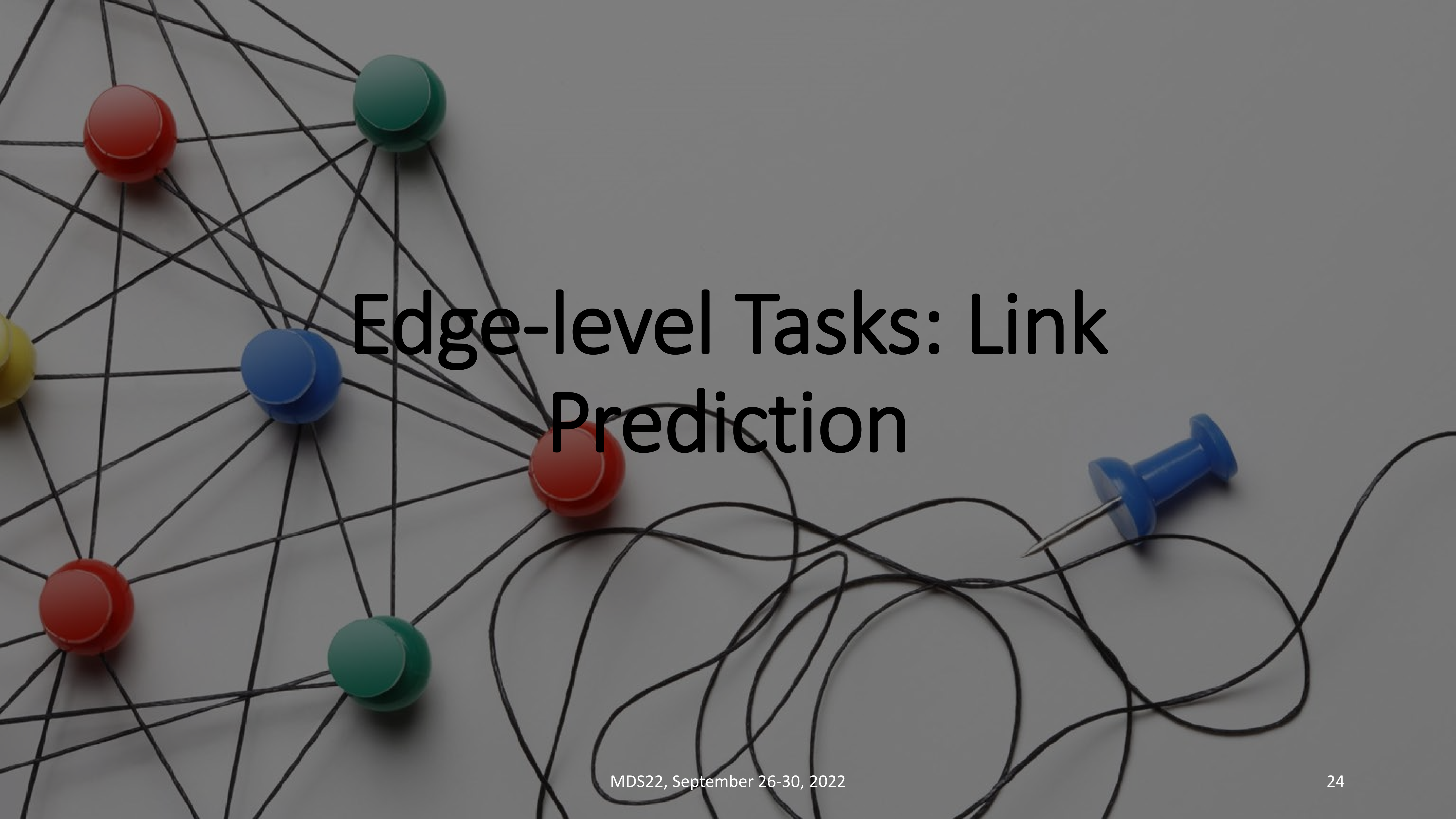


Input: graph with unlabled nodes



Output: graph node labels

Image source: [A Gentle Introduction to Graph Neural Networks (distill.pub)]



Edge-level Tasks: Link Prediction

Edge Level Tasks

- The most common edge-level task in GNN is link prediction.
- Link prediction means that given a graph, we want to predict whether there will be/should be an edge between two nodes or not.
- In a social network, this is used to propose new friends to you.
- The output prediction is done by performing a similarity metric on the pair of node features, which should be 1 if there should be a link, and otherwise close to 0.

Image Scene Understanding

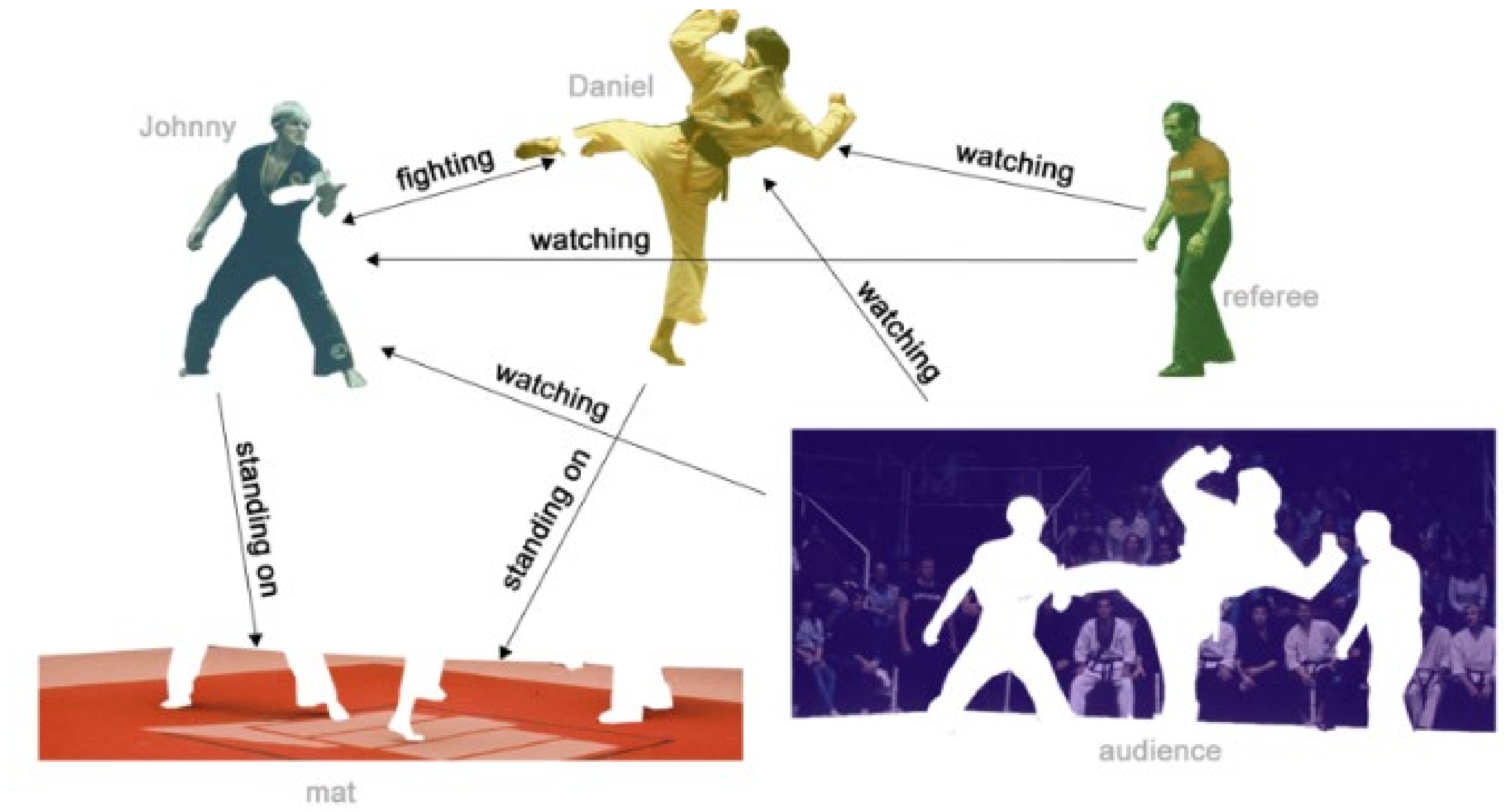
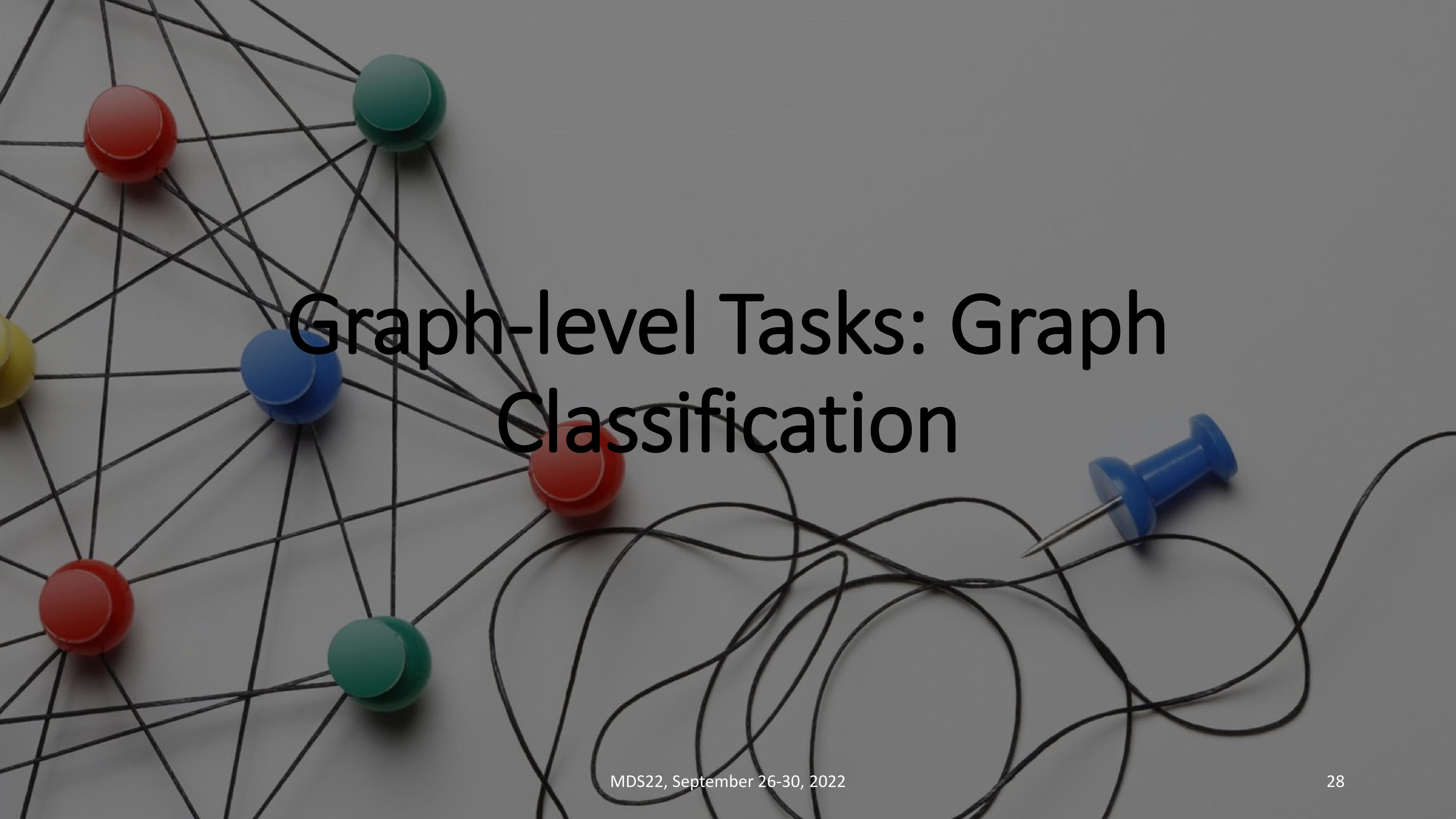


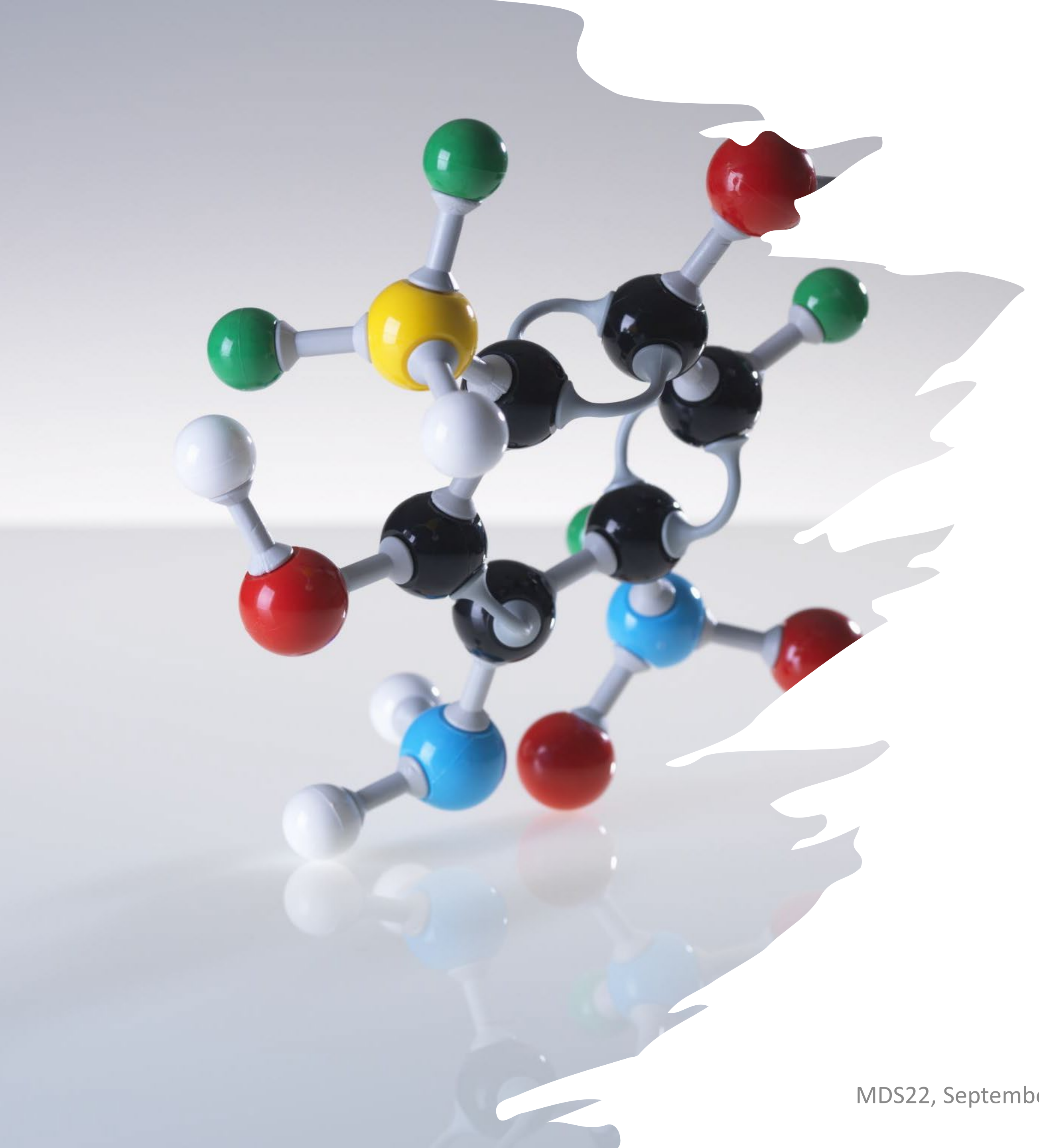
Image source: [A Gentle Introduction to Graph Neural Networks (distill.pub)]

Transform Edge Level to Node Level

- An edge prediction task on a graph ***G*** can be phrased as a node-level prediction on ***G***'s dual.
- To obtain ***G***'s dual, convert nodes to edges (and edges to nodes).



Graph-level Tasks: Graph Classification



Graph-level tasks: Graph classification

- **Classify** an entire graph instead of single nodes or edges
- Given a dataset of multiple graphs, classify individual graphs based on **structural graph properties**
- The most common task for graph classification is **molecular property prediction**
- Each atom is linked to a node, and edges in the graph are the bonds between atoms

Graph-level tasks: Graph classification

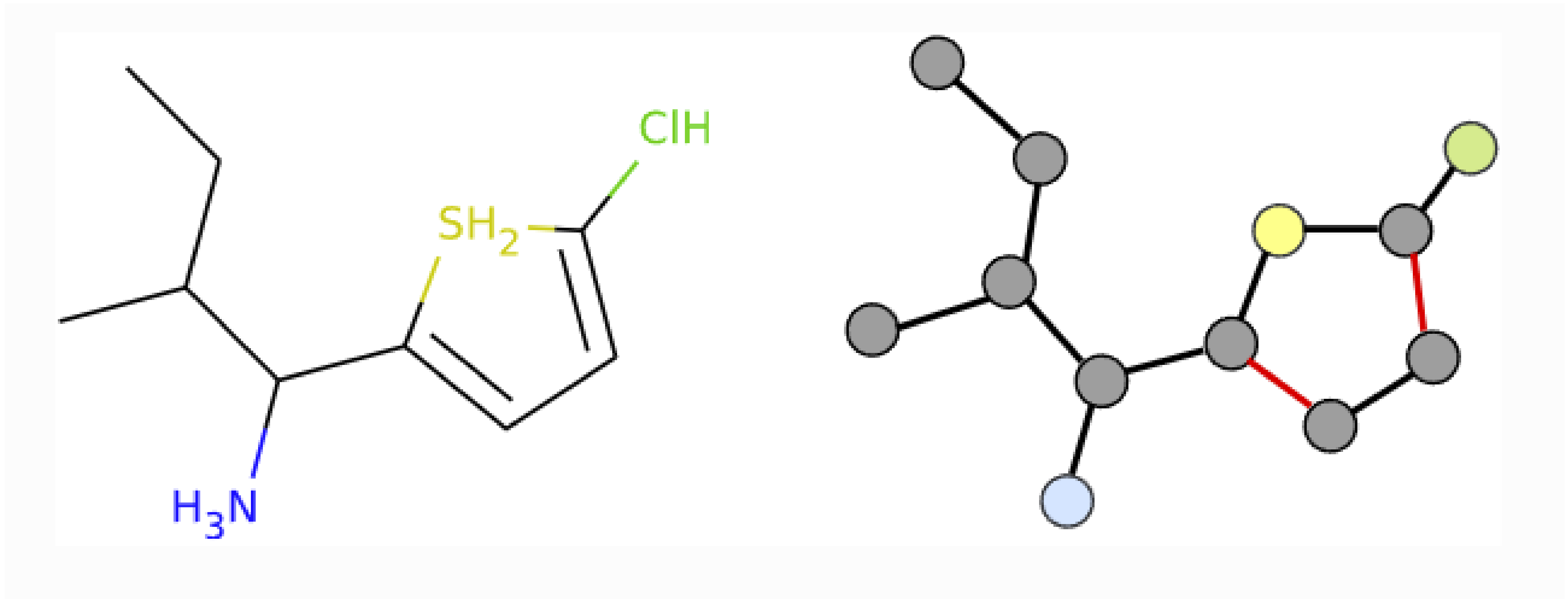


Image source: A Gentle Introduction to Graph Neural Networks (distill.pub)

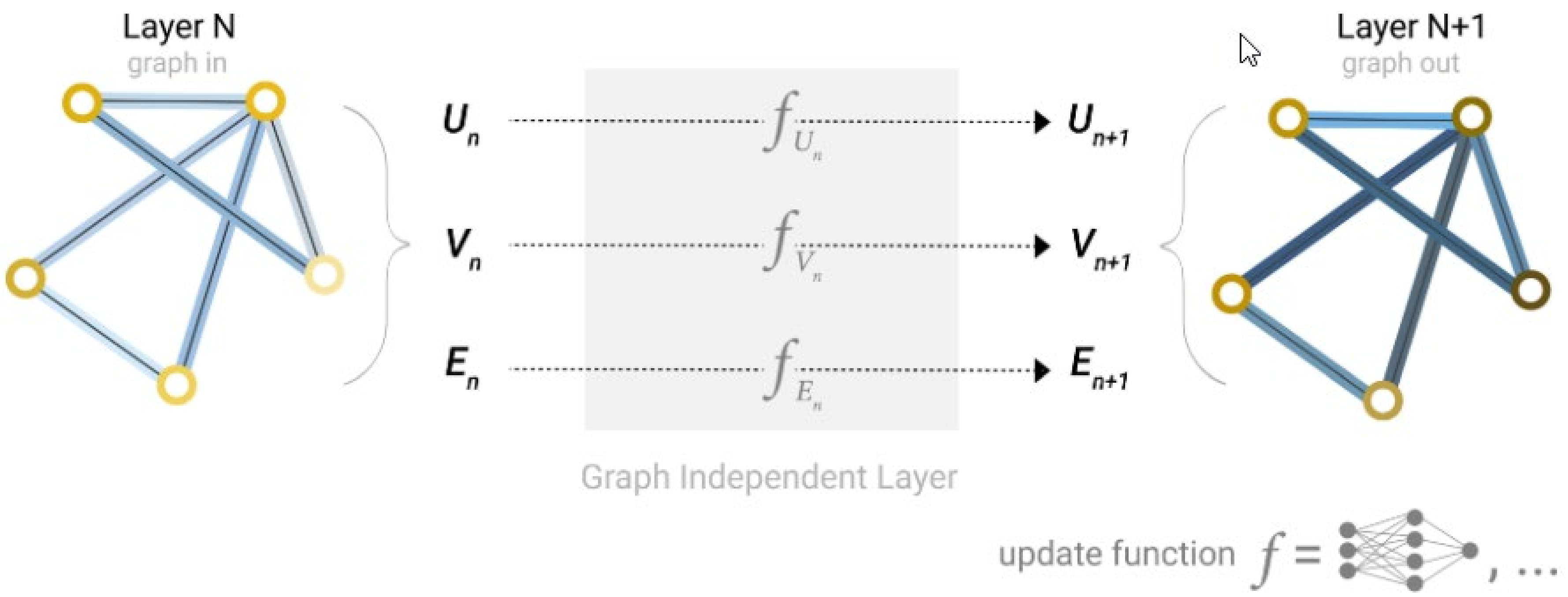
Graph Neural Networks History

- Sperduti and Starita first applied neural networks to directed acyclic graphs in **1997**
- The notion of GNNs was initially outlined by Gori, Monfardini, and Scarselli in **2005**
- These early studies (RecGNNs) learn a target node's representation by propagating neighbor information in an iterative manner until a stable fixed point is reached
- In **2009**, Micheli first addressed graph mutual dependence by architecturally compose non-recursive layers while inheriting ideas of message passing from RecGNNs
- The first prominent research on spectral-based ConvGNNs was presented by Bruna in **2014**
- The Graph Convolutional Networks were developed by Kipf and Welling in **2016**

Graph Neural Networks

- A GNN is an **optimizable transformation** on all attributes of the graph (nodes, edges, global-context) that preserves **graph symmetries**
- The Graph Nets architecture schematics introduced by Battaglia et al. (2014)
- To build GNNs, researcher use the “message passing neural network” framework proposed by Gilmer et al. (2020)
- GNNs adopt a “graph-in, graph-out” architecture

The Simplest GNN

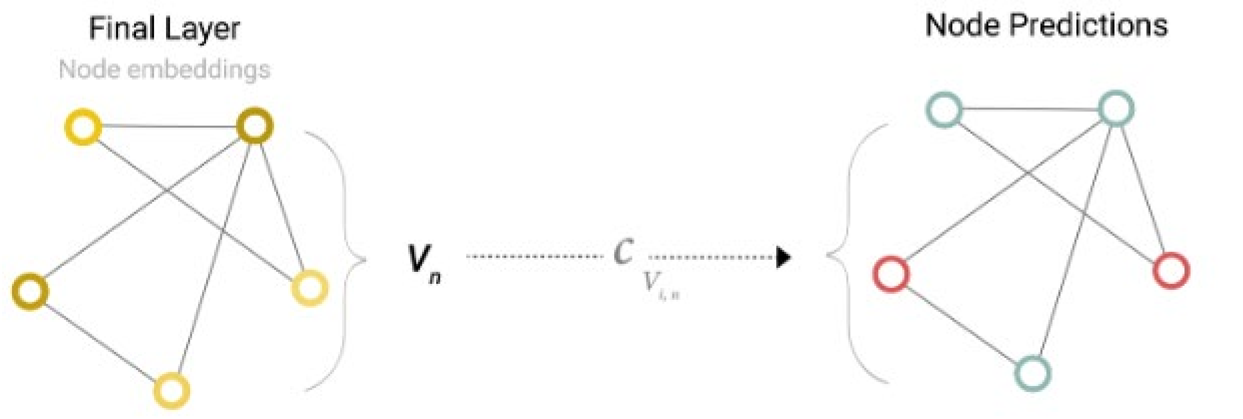


A single layer of a simple GNN. A graph is the input, and each component (V,E,U) gets updated by a MLP to produce a new graph. Each function subscript indicates a separate function for a different graph attribute at the n-th layer of a GNN model.

Image source: [A Gentle Introduction to Graph Neural Networks (distill.pub)]

The Simplest GNN

- The task is to make **binary predictions** on nodes
- For each **node embedding**, apply a linear classifier
- In the **Karate club** example, use the number of meetings between people to determine the alliance to Mr. Hi or John H.



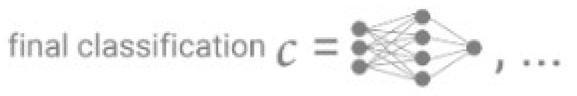


Image source: [A Gentle Introduction to Graph Neural Networks (distill.pub)]()

GNN Predictions by Pooling Information

- If the information in the graph is stored in edges, but no information is stored in nodes, but still need to make predictions on nodes.
- How to collect information from edges and give them to nodes for prediction? This can be done by *pooling*.
  1. For each item to be pooled, *gather* each of their embeddings and concatenate them into a matrix.
  2. The gathered embeddings are then *aggregated*, usually via a reduce operation.
- The pooling technique serves as the building block for constructing more sophisticated GNN models.

Pooling Edge-level Features for Node Prediction

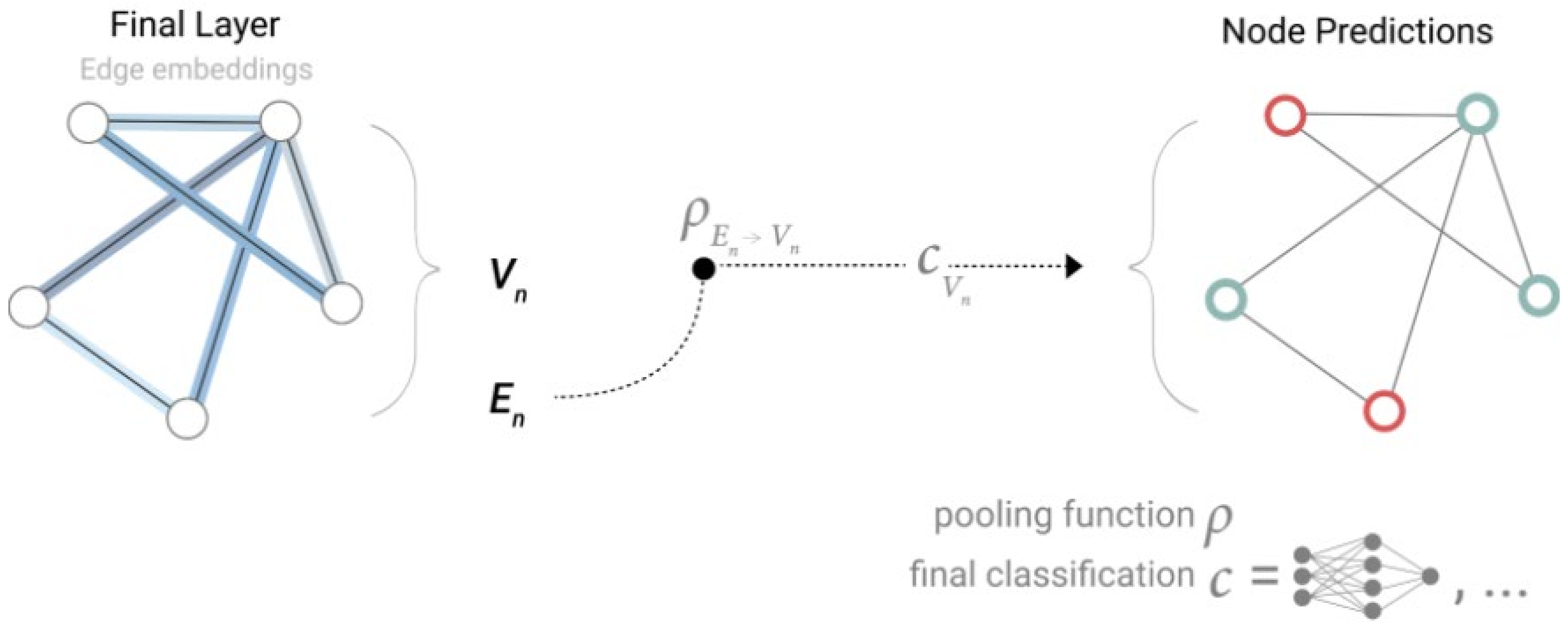


Image source: A Gentle Introduction to Graph Neural Networks (distill.pub)

Pooling Node-level Features for Edge Prediction

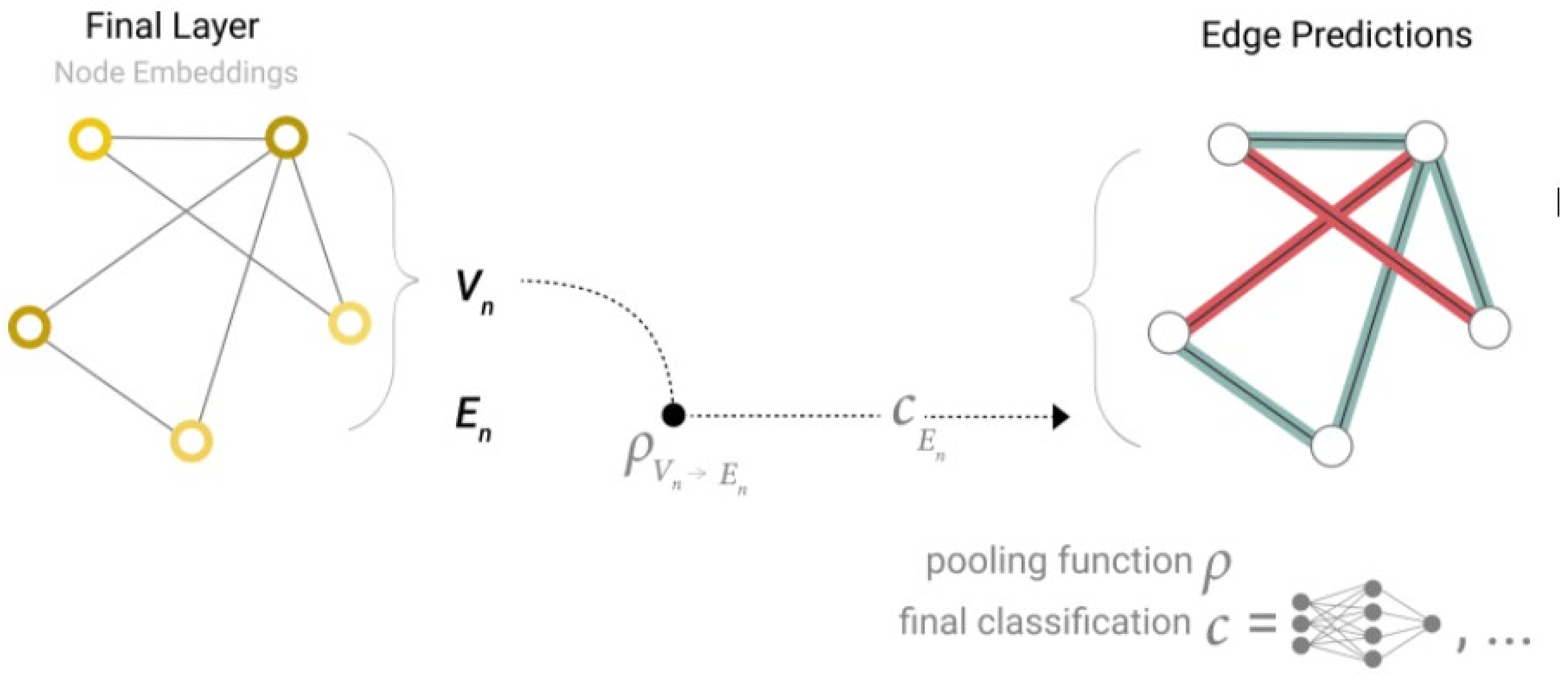


Image source: [A Gentle Introduction to Graph Neural Networks (distill.pub)]

Pooling Edge-level Features for Node Prediction

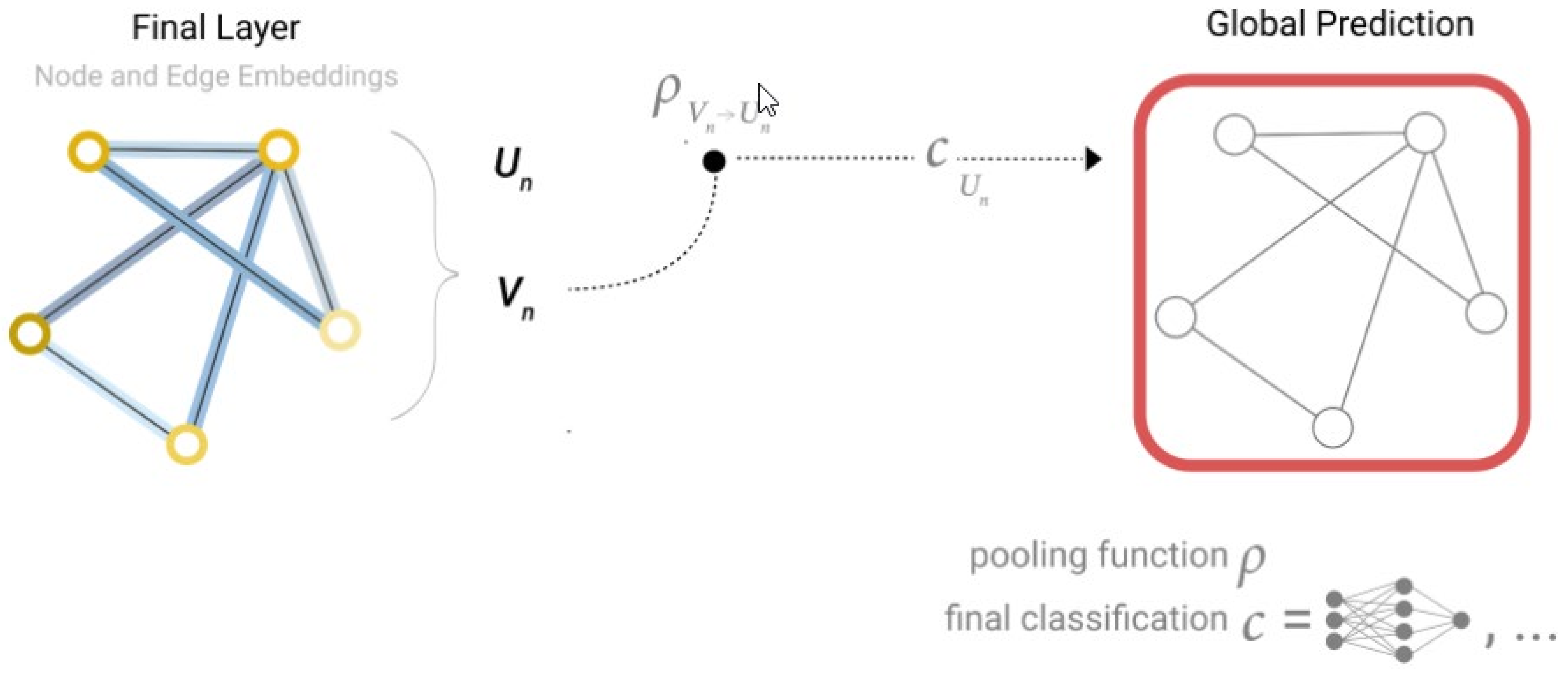


Image source: A Gentle Introduction to Graph Neural Networks (distill.pub)

Message Passing for Graph Connectivity

Message passing works in three steps:

1. For each node in the graph, *gather* all the neighboring node embeddings (or messages)
2. Aggregate all messages via an *aggregate function* (like sum).
3. All pooled messages are *passed* through an update function, usually a learned neural network.

Message Passing

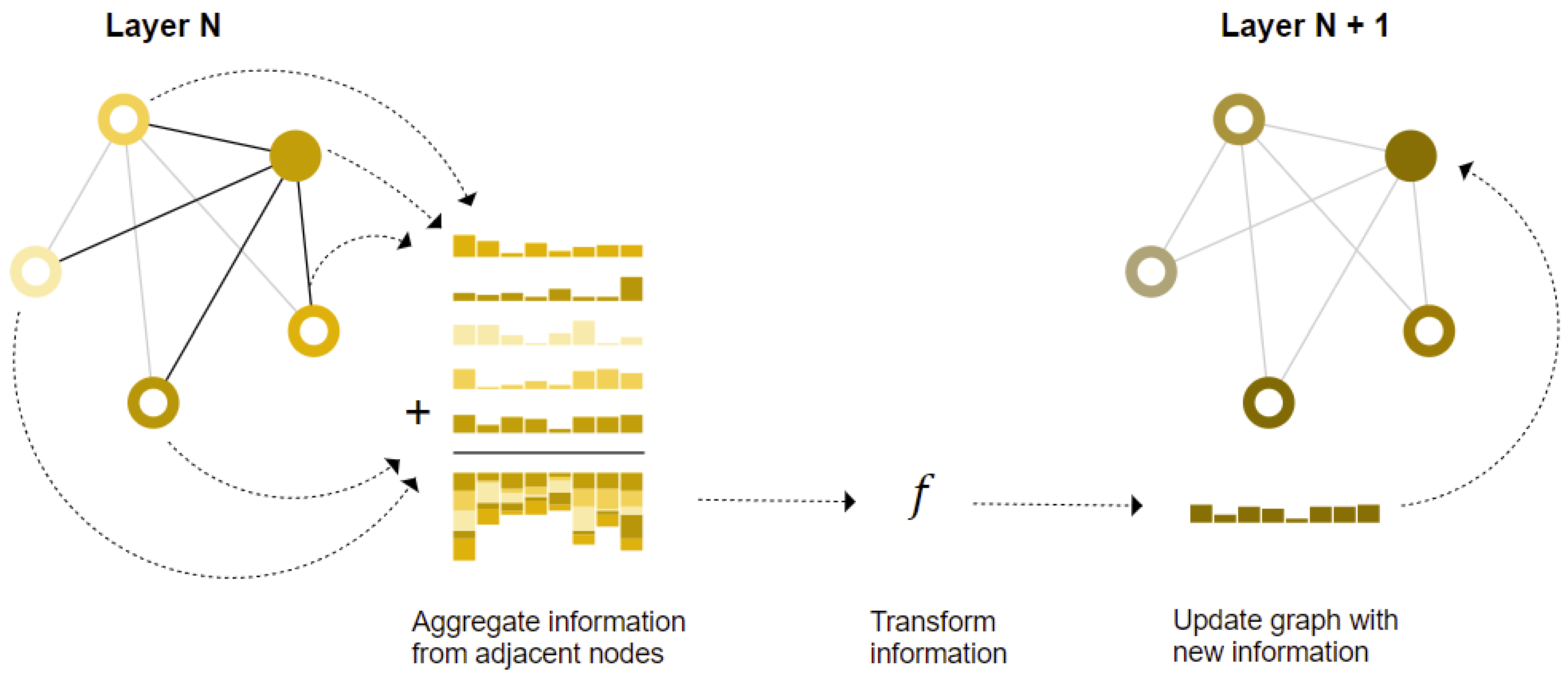

Layer N
Layer N + 1
+
f
Aggregate information from adjacent nodes
Transform information
Update graph with new information

Image source: [A Gentle Introduction to Graph Neural Networks (distill.pub)]

Pooling and Message Passing

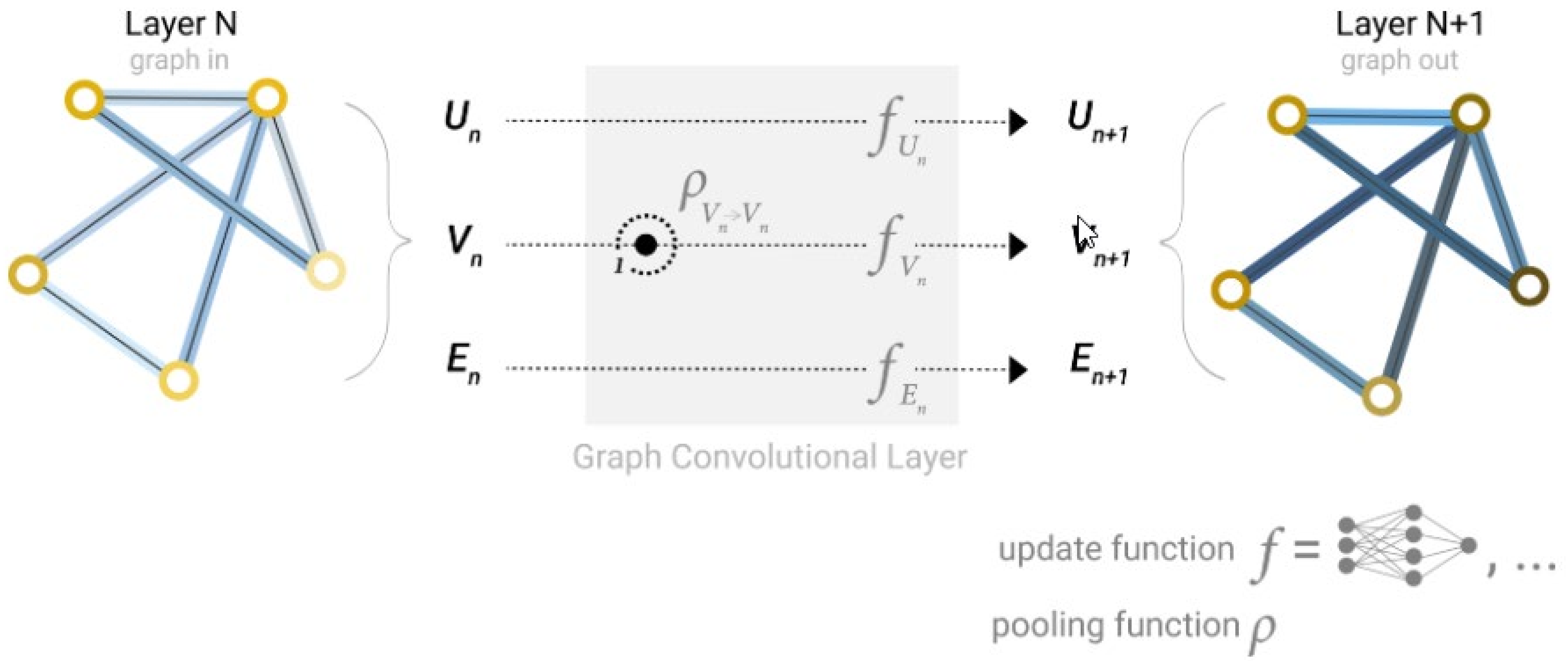


Image source: A Gentle Introduction to Graph Neural Networks (distill.pub)

Learning Edge Representations

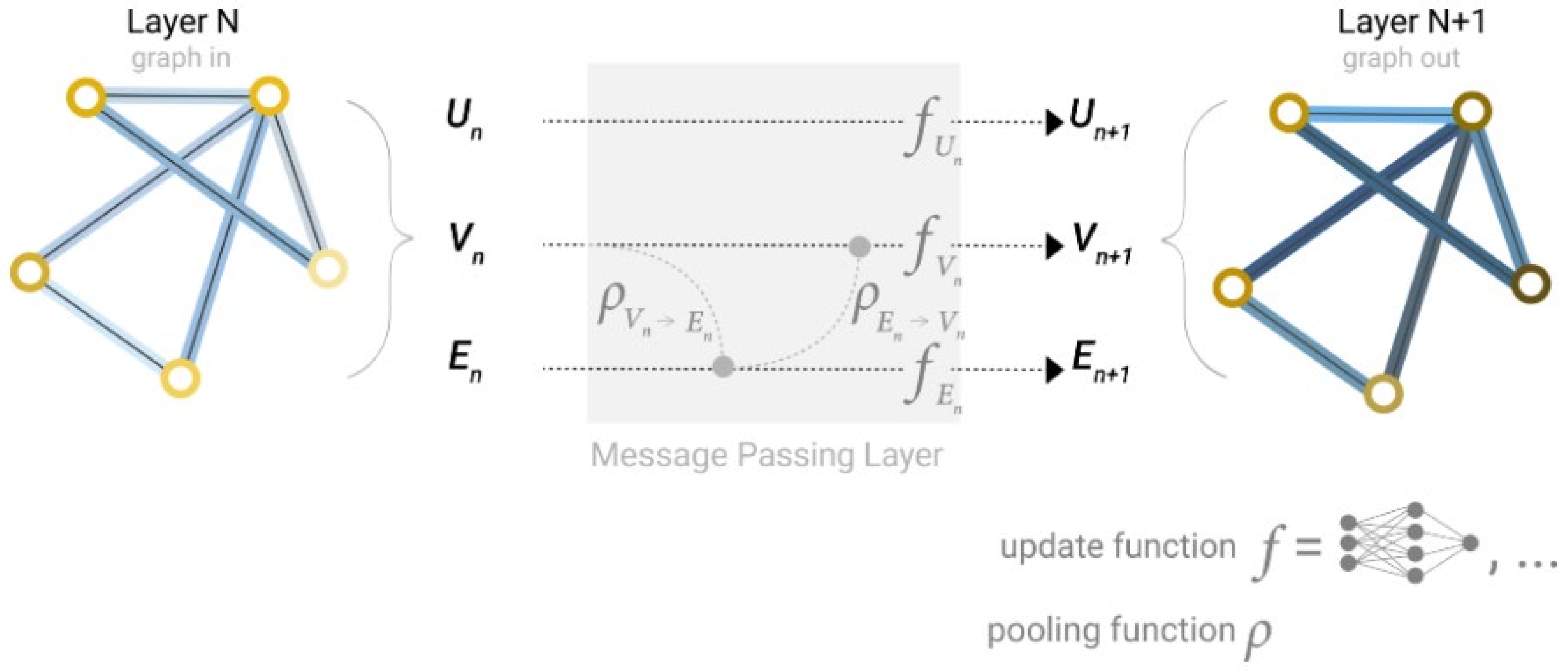


Image source: A Gentle Introduction to Graph Neural Networks (distill.pub)

Adding Global Representations

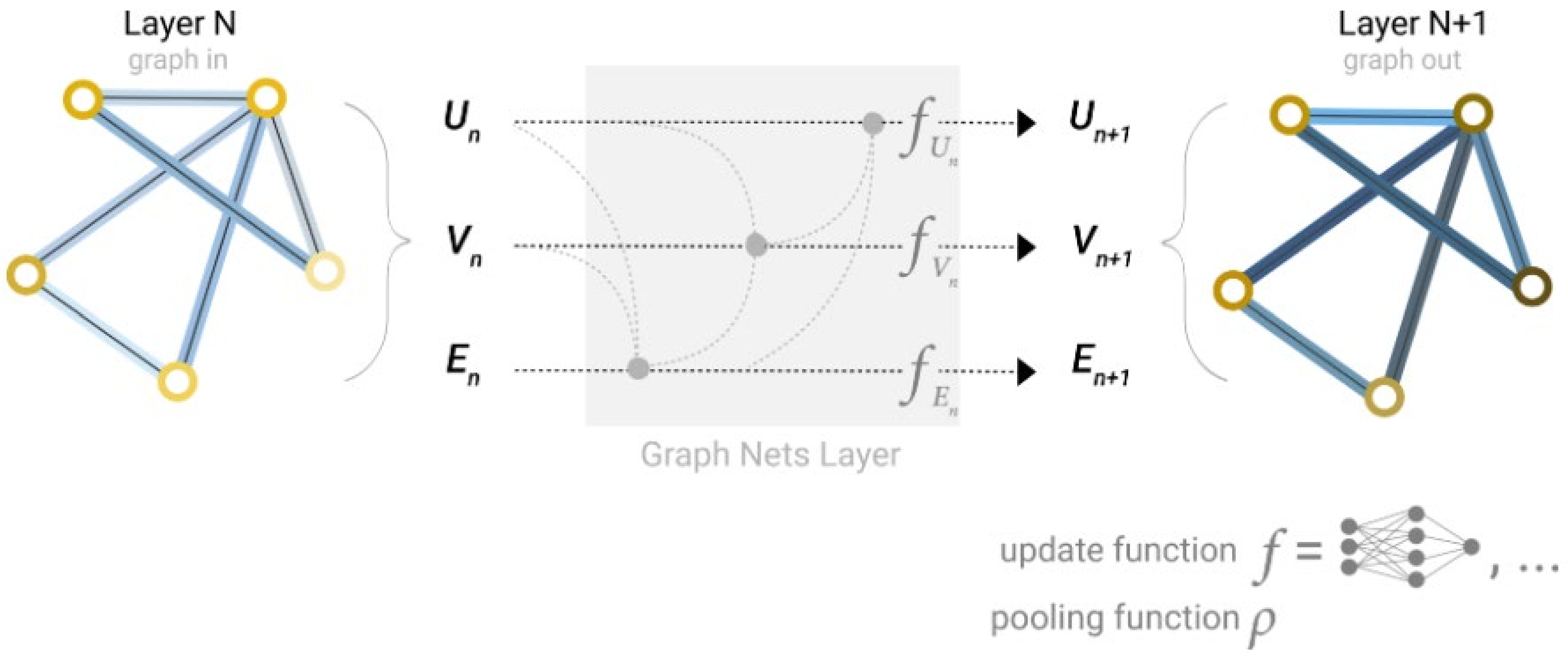


Image source: A Gentle Introduction to Graph Neural Networks (distill.pub)

Adding Global Representations

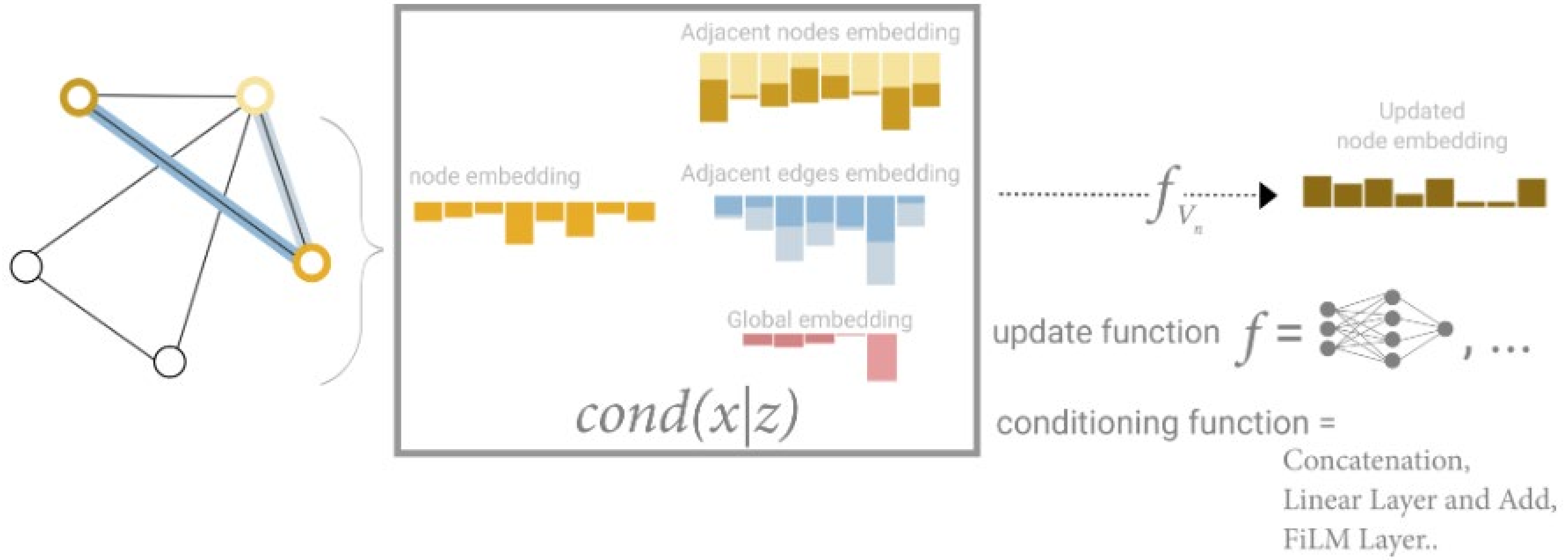
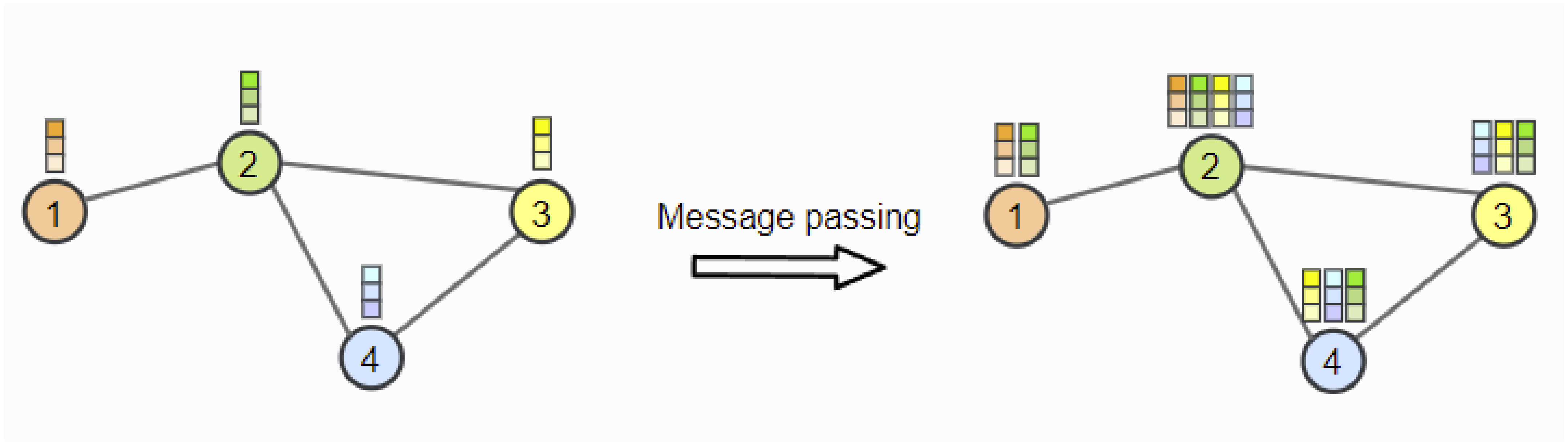


Image source: [A Gentle Introduction to Graph Neural Networks (distill.pub)]

Graph Convolutions

- Graph Convolutional Networks have been introduced by Kipf et al. in 2016 at the University of Amsterdam.
- GCNs are similar to convolutions in images in the sense that the “filter” parameters are typically shared over all locations in the graph.
- At the same time, GCNs rely on message passing methods, which means that vertices exchange information with the neighbors, and send “messages” to each other.

Graph Convolutions



Tutorial 7: Graph Neural Networks — UvA DL Notebooks v1.1 documentation (uvadlc-notebooks.readthedocs.io)

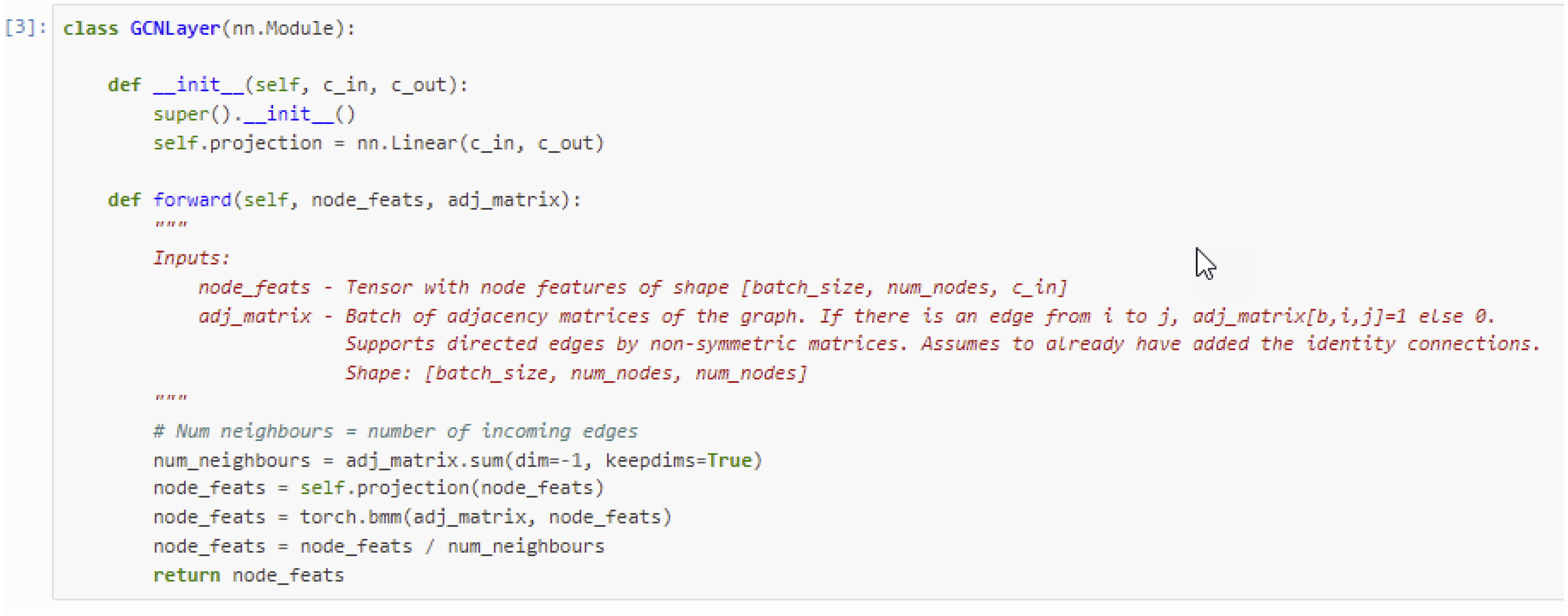
Graph Convolutions

$$H^{(l+1)} = \sigma\left(\widehat{D}^{-1/2}\hat{A}\widehat{D}^{-1/2}H^{(l)}W^{(l)}\right)$$

- $H^{(l)}$ - current features of nodes, $H^{(l+1)}$ next step features of nodes
- $W^{(l)}$ - weight parameters used to transform the input features into messages $(H^{(l)}W^{(l)})$
- Sum up the adjacency matrix and the identity matrix ($\hat{A} = A + I$), so each node sends its own message to itself
- To take the mean instead of sum, calculate the diagonal matrix ($\widehat{D}$) with D_{ii} denoting the number of neighbors node i has.
- σ represents an arbitrary activation function, and not necessarily the sigmoid (usually a ReLU-based activation function is used in GNNs).

[Tutorial 7: Graph Neural Networks — UvA DL Notebooks v1.1 documentation (uvadlc-notebooks.readthedocs.io)](uvadlc-notebooks.readthedocs.io)

Graph Convolutions – GCN Layer

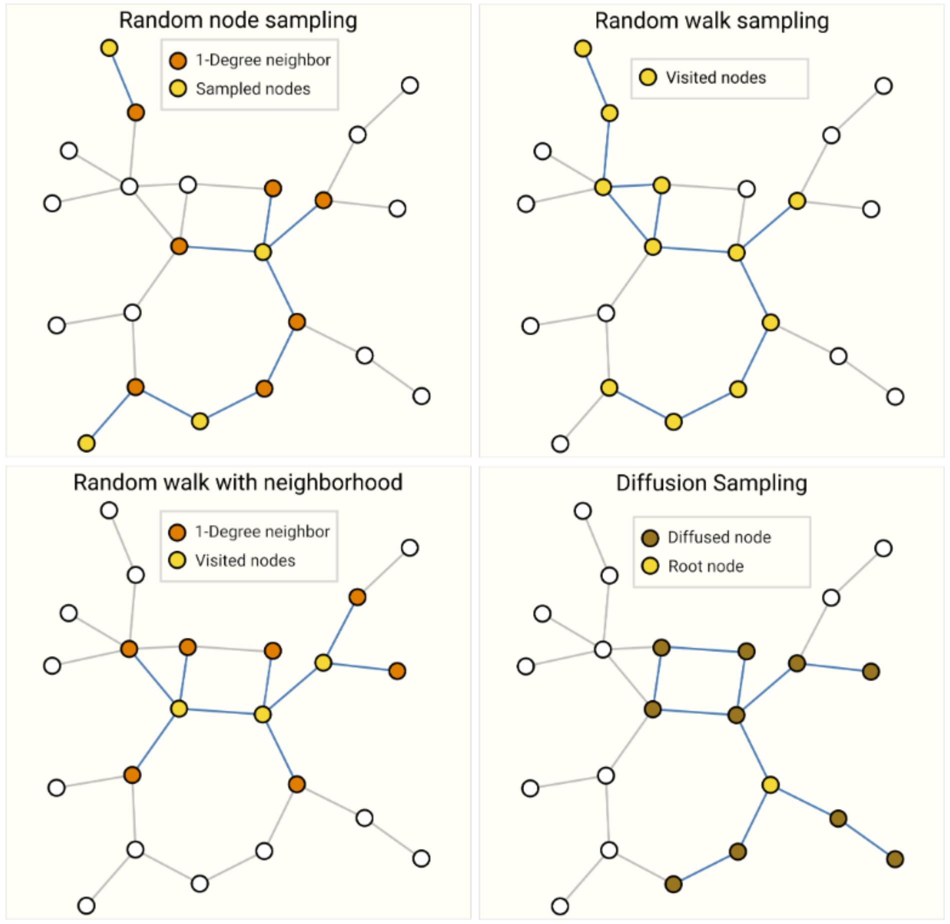


```
[3]: class GCNLayer(nn.Module):

         def __init__(self, c_in, c_out):
             super().__init__()
             self.projection = nn.Linear(c_in, c_out)

         def forward(self, node_feats, adj_matrix):
             """
             Inputs:
                 node_feats - Tensor with node features of shape [batch_size, num_nodes, c_in]
                 adj_matrix - Batch of adjacency matrices of the graph. If there is an edge from i to j, adj_matrix[b,i,j]=1 else 0.
                              Supports directed edges by non-symmetric matrices. Assumes to already have added the identity connections.
                              Shape: [batch_size, num_nodes, num_nodes]
             """
             # Num neighbours = number of incoming edges
             num_neighbours = adj_matrix.sum(dim=-1, keepdims=True)
             node_feats = self.projection(node_feats)
             node_feats = torch.bmm(adj_matrix, node_feats)
             node_feats = node_feats / num_neighbours
             return node_feats
```

Sampling Graphs and Batching in GNNs



slido

Audience Q&A Session

ⓘ Start presenting to display the audience questions on this slide.

Conclusions

- Graphs are a powerful and rich structured data type that have strengths and challenges that are very different from those of images and text.
- Great opportunity for a wide range of new problems.
- Flexible to work with complex and large datasets.
- It is difficult to find the right problem representation.
- Important practical problem can be solved.
- Opens research areas where data consists of non-Euclidean patterns and relations.

Introduction to Graph Neural Networks

Alina Lazar[1] and Xiangyang Yu[2]

[1]School of Computer Science, Information and Engineering Technology, Youngstown State University

[2]Physics and X-Ray Science Computing Group, Lawrence Berkeley National Laboratory

alazar@ysu.edu, xju@lbl.gov

Part 2 – Hands-on

- GNN Frameworks
- Introduction to Graphs used in the tutorial
- Hands-on tutorial:
    - Node classification
    - Edge classification

GNN Frameworks

Library Name	License	Stars	Programming Language	Main Contributers
PyTorch Geometric (PyG)	MIT	14.6k	Python, PyTorch	Matthias Fey
Deep Graph Library (DGL)	Apache 2.0	9.6k	Python, PyTorch, TF, MxNet	Distributed MLC
Graph Nets	Apache 2.0	5.1k	Python, TF, Sonnet	DeepMind
Spektral	MIT	2.1k	Python, TF/Keras	Daniele Grattarola
Jraph	Apache 2.0	892	Python, Jax	DeepMind
GeometricFlux.jl	MIT	276	Julia, Flux	Julia Project

Table source: https://neptune.ai/blog/graph-neural-networks-libraries-tools-learning-resources

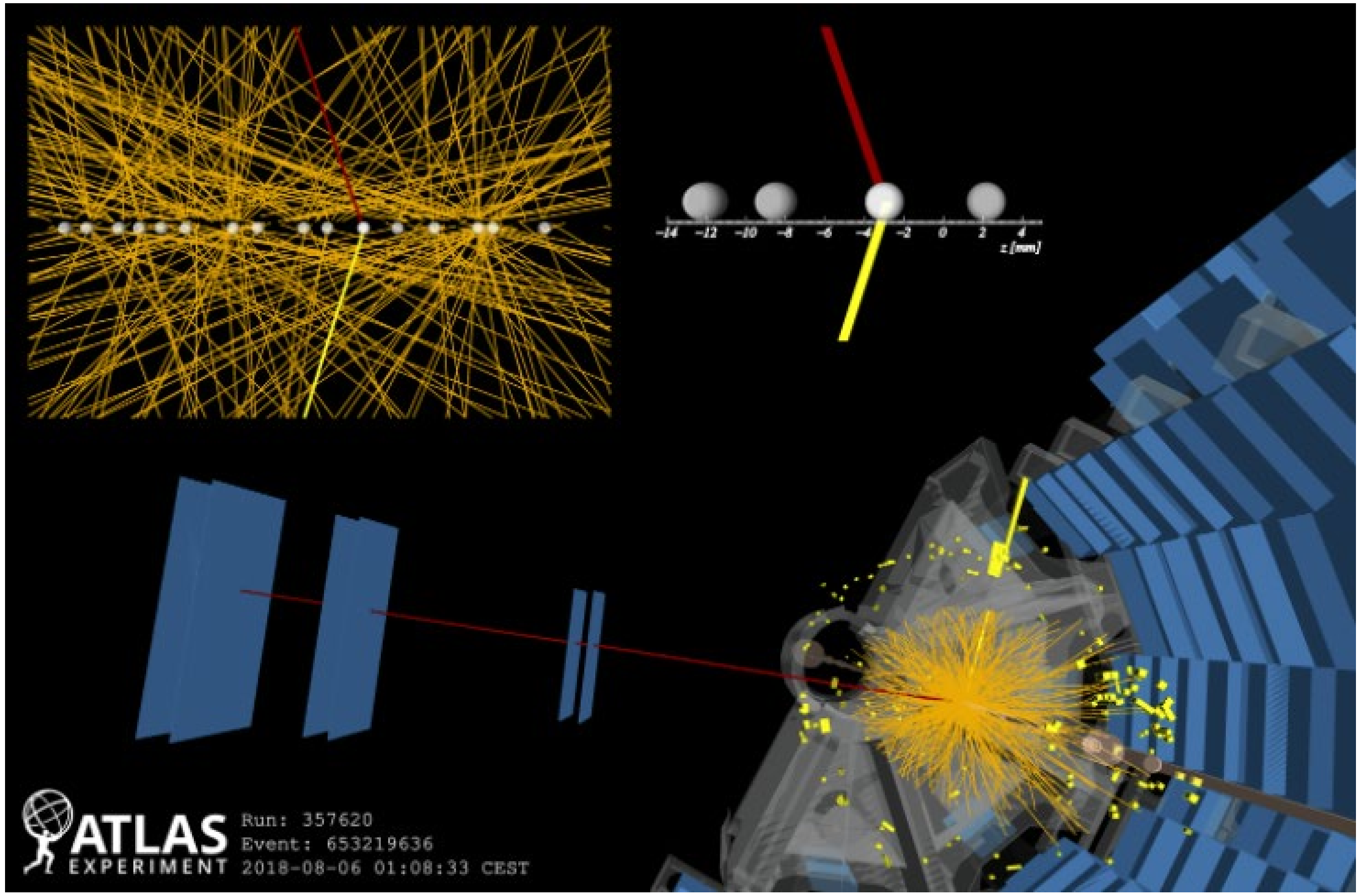
GNN Frameworks

- 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 implements easy-to-use mini-batch loaders for operating on many small and single giant graphs, multi-GPU support, DataPipe support, and distributed graph learning via Quiver.

Tutorial Demo

https://github.com/alinutzal/IntrotoGNN

HEP – Large Hadron Collider – Track Finding



Bibliography

- A. Sperduti and A. Starita, “Supervised neural networks for the classification of structures,” IEEE Trans. Neural Netw., vol. 8, no. 3, pp. 714–735, May 1997.
- M. Gori, G. Monfardini, and F. Scarselli, “A new model for learning in graph domains,” in Proc. IEEE Int. Joint Conf. Neural Netw., vol. 2, Aug. 2005, pp. 729–734.
- C. Gallicchio and A. Micheli, “Graph echo state networks,” in Proc. Int. Joint Conf. Neural Netw. (IJCNN), Jul. 2010, pp. 1–8.
- J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral networks and locally connected networks on graphs,” in Proc. ICLR, 2014, pp. 1–14.
- T. N. Kipf and M. Welling, “Semi-Supervised Classification with Graph Convolutional Networks,” arXiv [cs.LG], Sep. 09, 2016. [Online]. Available: http://arxiv.org/abs/1609.02907
- J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Message Passing Neural Networks,” in Machine Learning Meets Quantum Physics, K. T. Schütt, S. Chmiela, O. A. von Lilienfeld, A. Tkatchenko, K. Tsuda, and K.-R. Müller, Eds. Cham: Springer International Publishing, 2020, pp. 199–214.
- A. Santoro et al., “A simple neural network module for relational reasoning,” Adv. Neural Inf. Process. Syst., vol. 30, 2017.