UE21CS343BB2

Topics in Deep Learning

Dr. Shylaja S S
Director of Cloud Computing & Big Data (CCBD), Centre for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

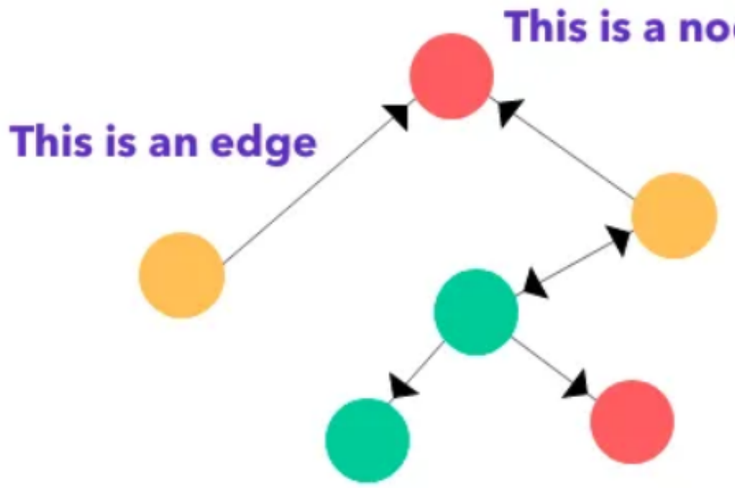
Aryan Sharma,
Teaching Assistant

UE21CS343BB2: Topics in Deep Learning

Introduction to GNNs

What is a Graph?

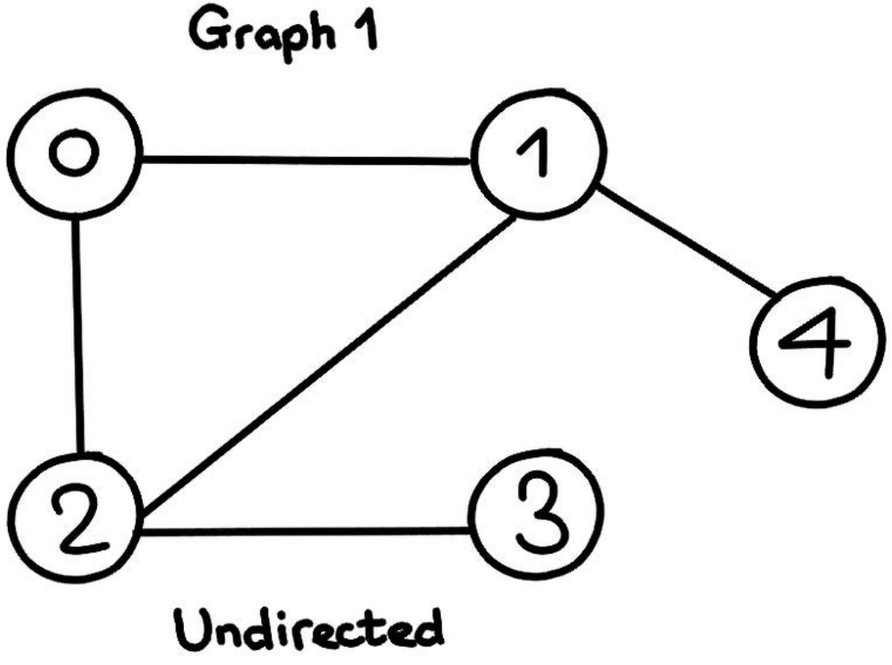
A graph is a data structure comprising of nodes (vertices) and edges connected together to represent information with no definite beginning or end.

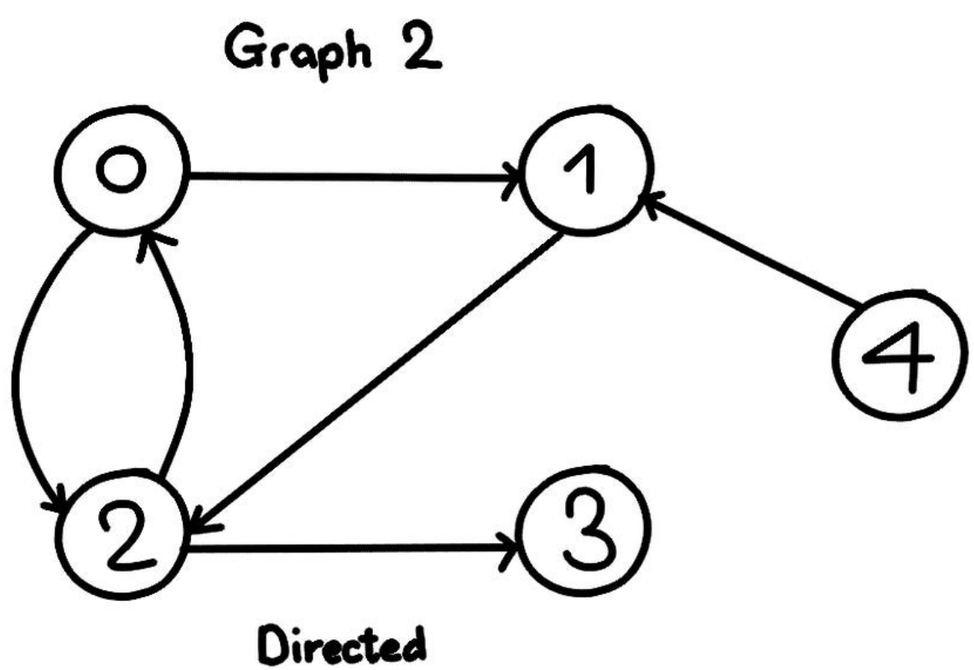


All the nodes occupy an arbitrary position in space, usually clustered according to similar features when plotted in 2D (or even nD) space.

What is a Graph?

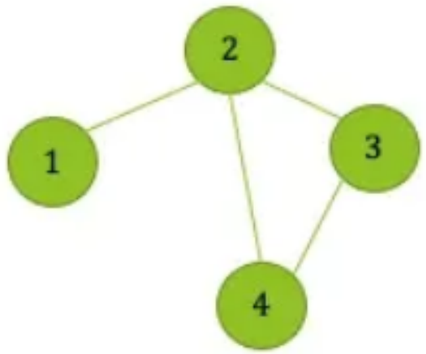
The two different kinds of graphs are directed (connection direction matters between nodes) and undirected (connection order doesn't matter).
Directed graphs can be unidirectional or bidirectional in nature.





What is a Graph?

As we know, a graph can be drawn as -



It is mathematically, however represented as -

$$G = (V, E)$$
$$V = \{1, 2, 3, 4\}$$
$$E = \{\,(1,2), (2,3), (2,4), (3,4)\}$$

Graphs and where to find them (some examples)

Images as graphs

Another way to think of images is as graphs with regular structure, where each pixel represents a node and is connected via an edge to adjacent pixels. Each non-border pixel has exactly 8 neighbors, and the information stored at each node is a 3-dimensional vector representing the RGB value of the pixel.

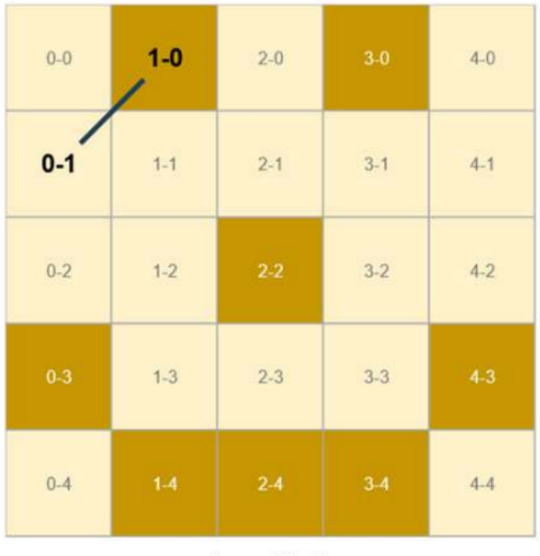
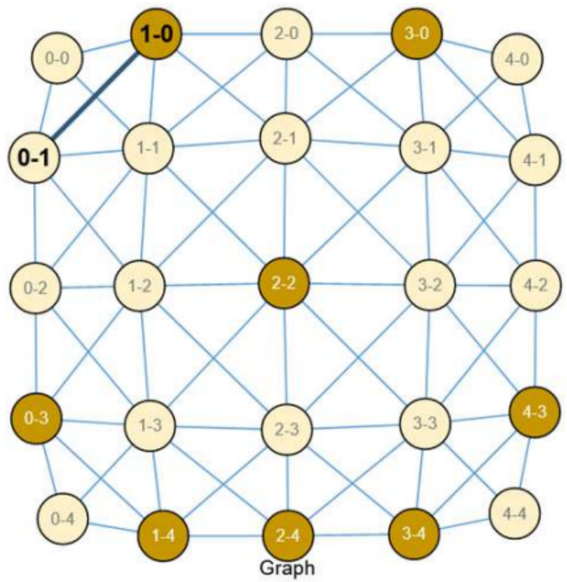


Image Pixels

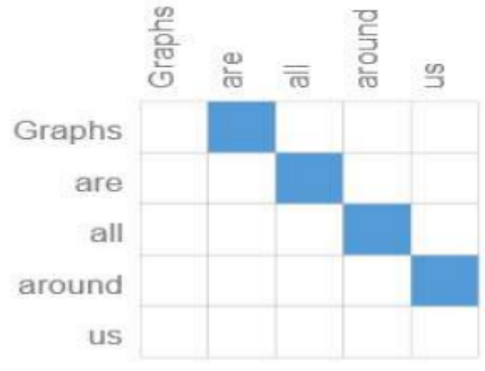


Graph

Graphs and where to find them (some examples)

Text as graphs

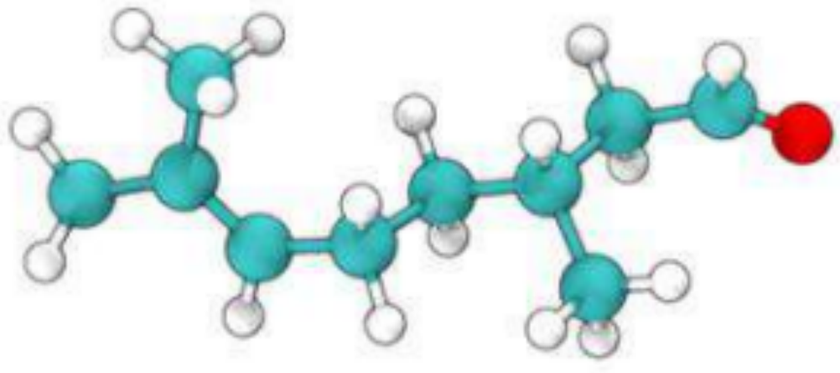
We can digitize text by associating indices to each character, word, or token, and representing text as a sequence of these indices. This creates a simple directed graph, where each character or index is a node and is connected via an edge to the node that follows it.

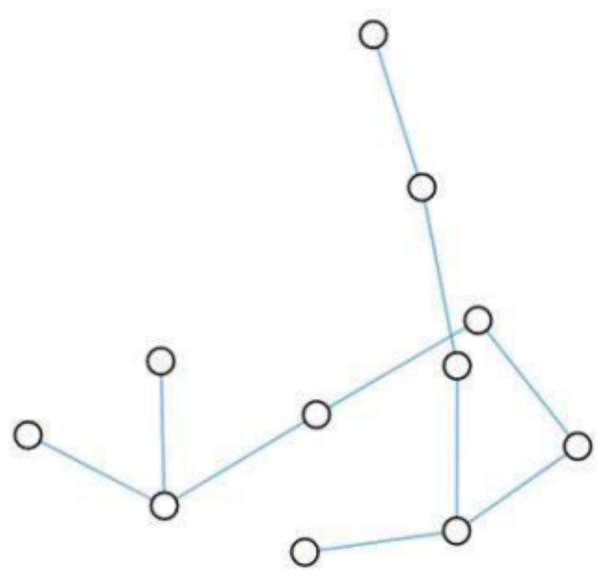


Graphs and where to find them (some examples)

Molecules as graphs

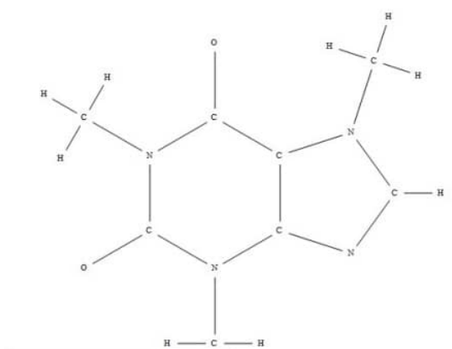
It’s a very convenient and common abstraction to describe this 3D object as a graph, where nodes are atoms and edges are covalent bonds.



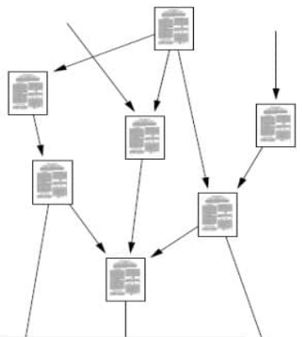


Topics in Deep Learning

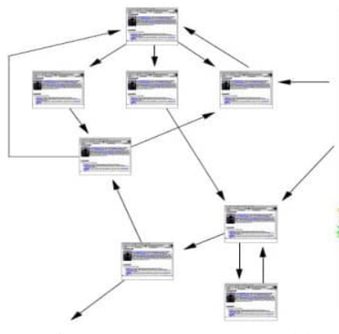
Graphs and where to find them (some examples)



Molecules

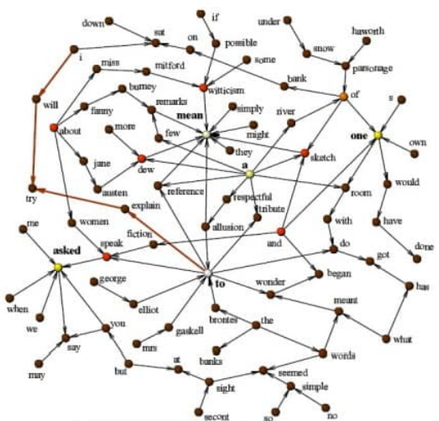


Knowledge

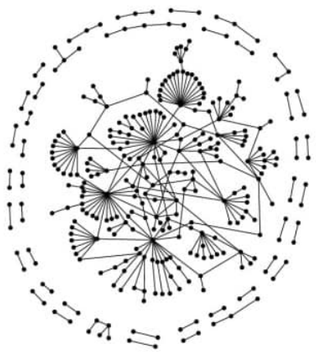


Information

Brain/neurons



Genes

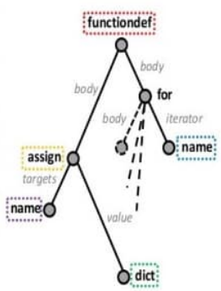


Communication

```
def encode(obj):
    """
    Encode a (possibly nested)
    dictionary containing complex values
    into a form that can be serialized
    using JSON.
    """
    e = {}
    for key,value in obj.items():
        if isinstance(value,dict):
            e[key] = encode(value)
        elif isinstance(value,complex):
            e[key] = {'type' : 'complex',
                      'r' : value.real,
                      'i' : value.imag}
    return e
```

```
import ast
tree = ast.parse("...")
...
```

Software

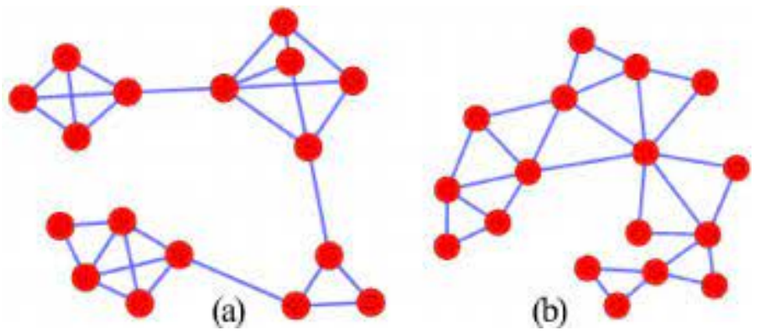


Social

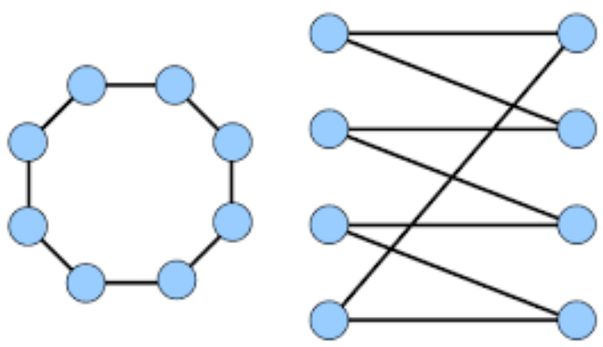
Ok, but why GNNs?

Traditional neural networks struggle with graph-structured data due to:

Fixed input size: Graphs can have varying numbers of nodes, unlike images with fixed dimensions.

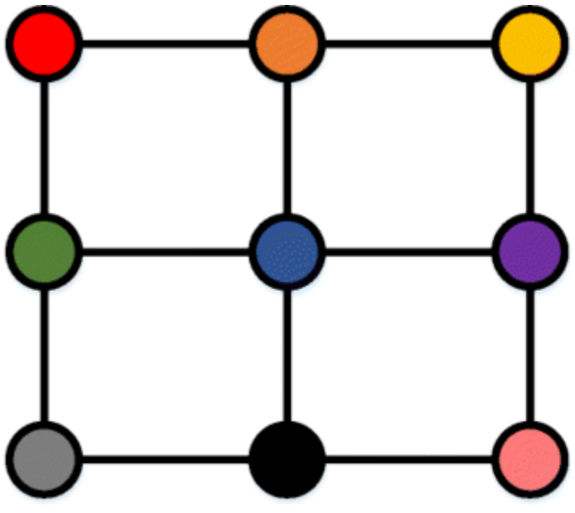


Order invariance: The order of nodes shouldn't affect analysis, unlike sequential data.



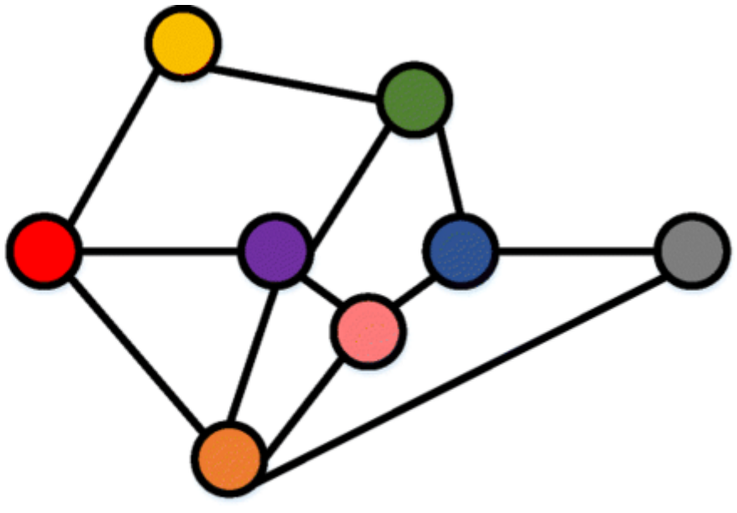
Ok, but why GNNs?

Traditional neural networks, designed for Euclidean data like images, falter when faced with the inherent variability and relational structure of graphs. GNNs address these limitations, paving the way for effective graph analysis.



CNN

In Euclidean Space

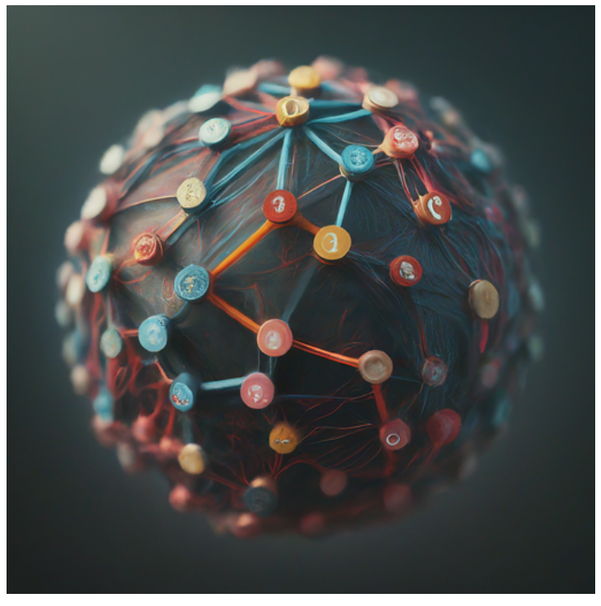


GNN

In Non-Euclidean Space

Core Concepts of GNNs

- **Message Passing:** Nodes exchange information with their neighbors via messages passed along edges.
- **Aggregation:** Each node aggregates messages from its neighbors to update its own representation.
- **Node Embeddings:** Nodes are encoded into low-dimensional vectors capturing their features and relationships.



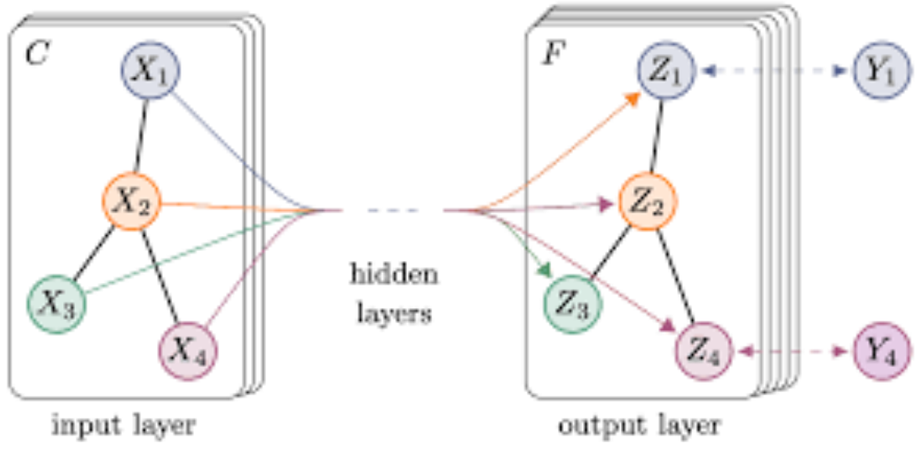
These core concepts form the foundation of GNNs. Message passing allows information to flow through the graph, aggregation incorporates neighborhood information, and node embeddings condense rich data into a usable format.

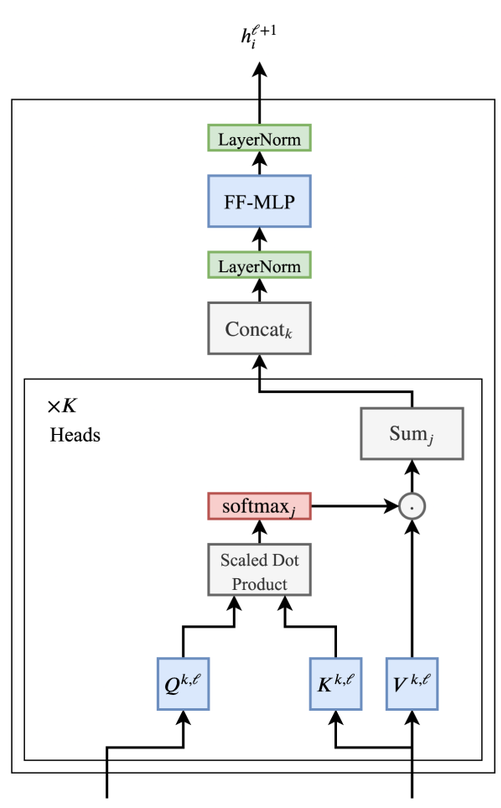
Popular GNN Architectures

- **Graph Convolutional Networks (GCNs):** Inspired by CNNs, GCNs aggregate messages from local neighborhoods.
- **Recurrent GNNs (e.g., Gated Recurrent Units - GRUs):** Utilize recurrent mechanisms for information propagation over longer sequences.
- **Transformer-based GNNs:** Leverage attention mechanisms to focus on relevant parts of the graph.

Popular GNN Architectures

Different GNN architectures cater to specific tasks and data characteristics.
GCNs excel at localized information extraction, while recurrent GNNs handle sequential data within graphs, and transformer-based GNNs capture long-range dependencies.





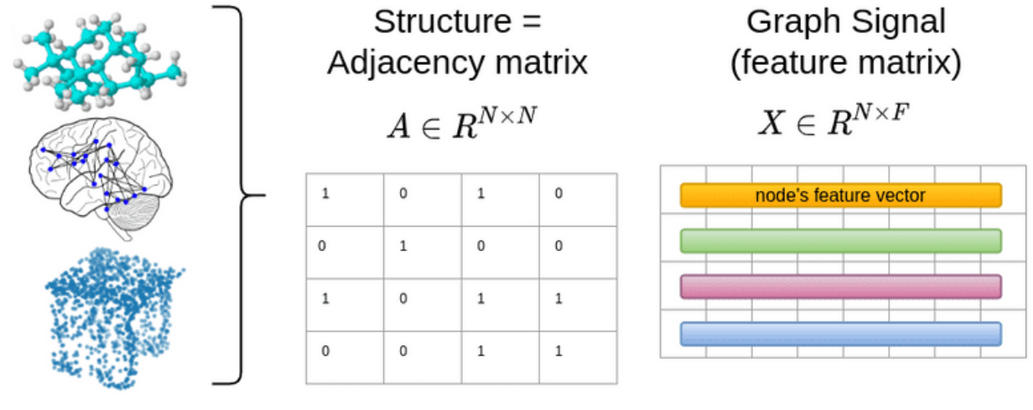
The Starting Point: Graph Representation

- **Graph (G):** A collection of nodes (entities) and edges (relationships) connecting them.
- **Node Features (X):** Numerical vectors representing attributes associated with each node.
- **Adjacency Matrix (A):** A matrix where each entry A[i,j] indicates if there's an edge between nodes i and j.

G = (V, E) - Set of nodes V and set of edges E

X = $[x_1, ..., x_N]$ - Node feature matrix

A = $[a_{ij}]$ - Adjacency matrix



The Starting Point: Graph Representation

Imagine a citation network: nodes are research papers, edges indicate citations, and node features might include publication year, author names, and keywords.

The GNN operates on this graph representation, extracting knowledge from nodes, edges, and their interconnections.

The adjacency matrix is crucial, capturing the network structure and allowing information flow between connected nodes.

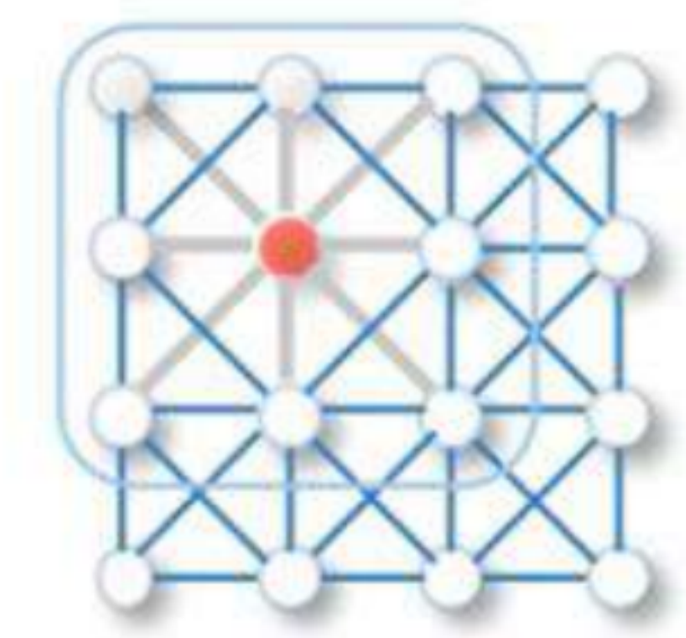
The node feature matrix encodes individual node characteristics, providing valuable insights for the GNN.

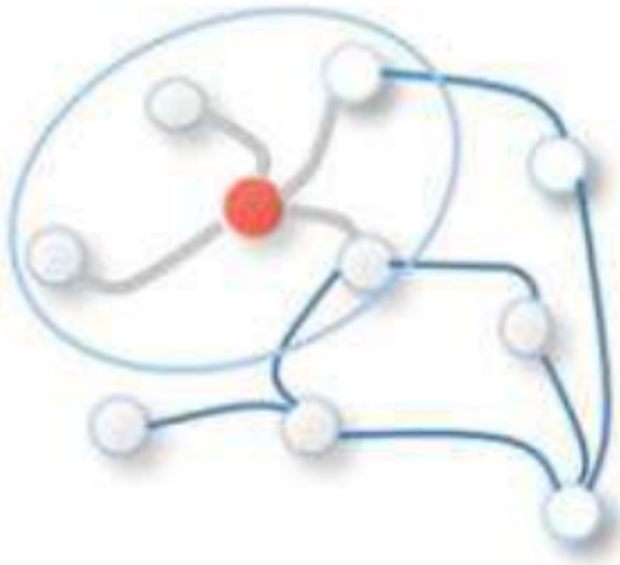
The Message Passing Mechanism: Sharing Information

- **Message Function:** Takes node features and edge features as input, outputs a message vector.
- **Message Passing:** Each node sends a message to its neighbors, summarizing its own information.
- **Neighborhood Aggregation:** Each node combines messages from neighbors and its own features to update its representation.

This is the heart of the GNN! Each node acts as a "messenger," sending a message to its neighbors based on its own features and the edge connecting them. Neighbors then aggregate these messages with their own features, updating their understanding of themselves and their local context.

The Message Passing Mechanism: Sharing Information





Similar to filter in CNN

The Message Passing Mechanism: Sharing Information

Message function: $m_{ij} = f(h_i, h_j, e_{ij})$

Aggregation function: $h_i^{(t+1)} = g(h_i^{(t)}, \text{aggregate}(m_{ji}, \forall j \in N(i)))$

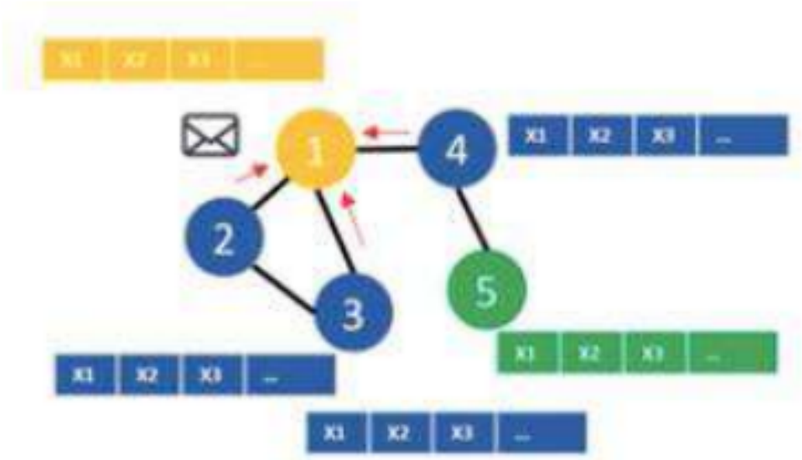
$h_i^{(t)}$: Node representation at layer t, $N(i)$: Neighbors of node i.

The message function can be a simple neural network, but its complexity depends on the specific GNN architecture.

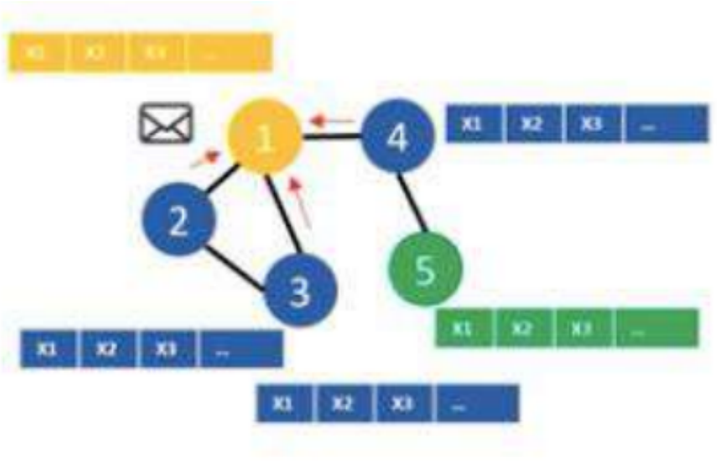
The aggregation function can involve averaging, summing, or more sophisticated methods like attention mechanisms.

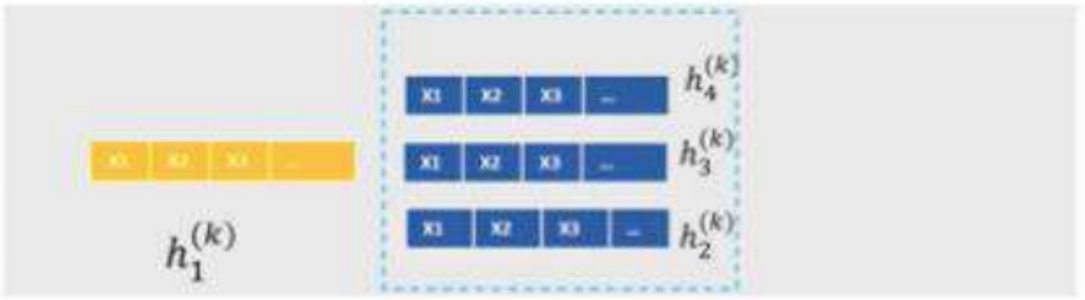
These choices impact how effectively nodes capture information from their neighborhoods.

The Message Passing Mechanism: Sharing Information



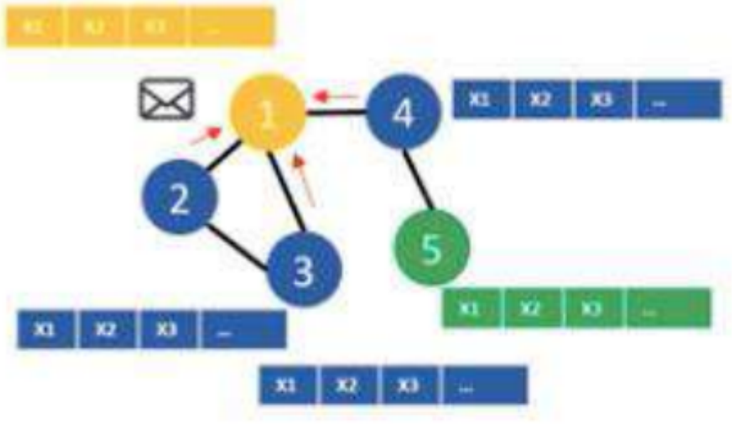
The Message Passing Mechanism: Sharing Information

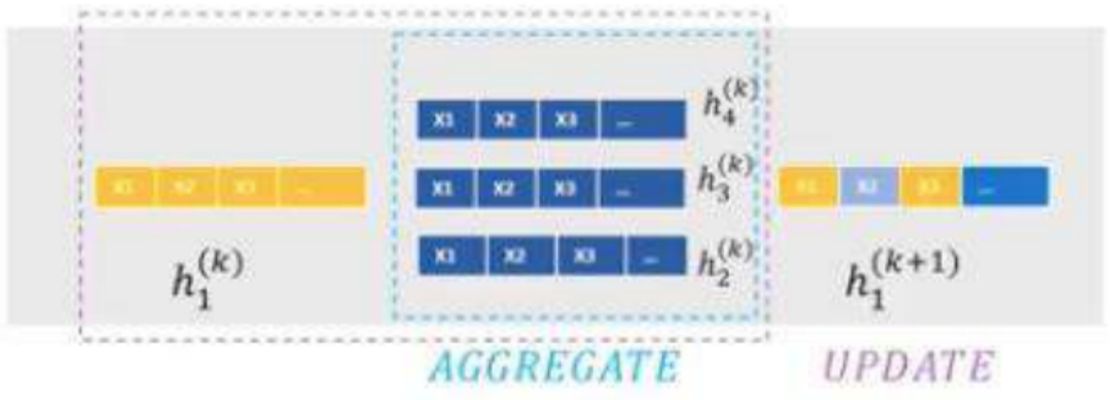




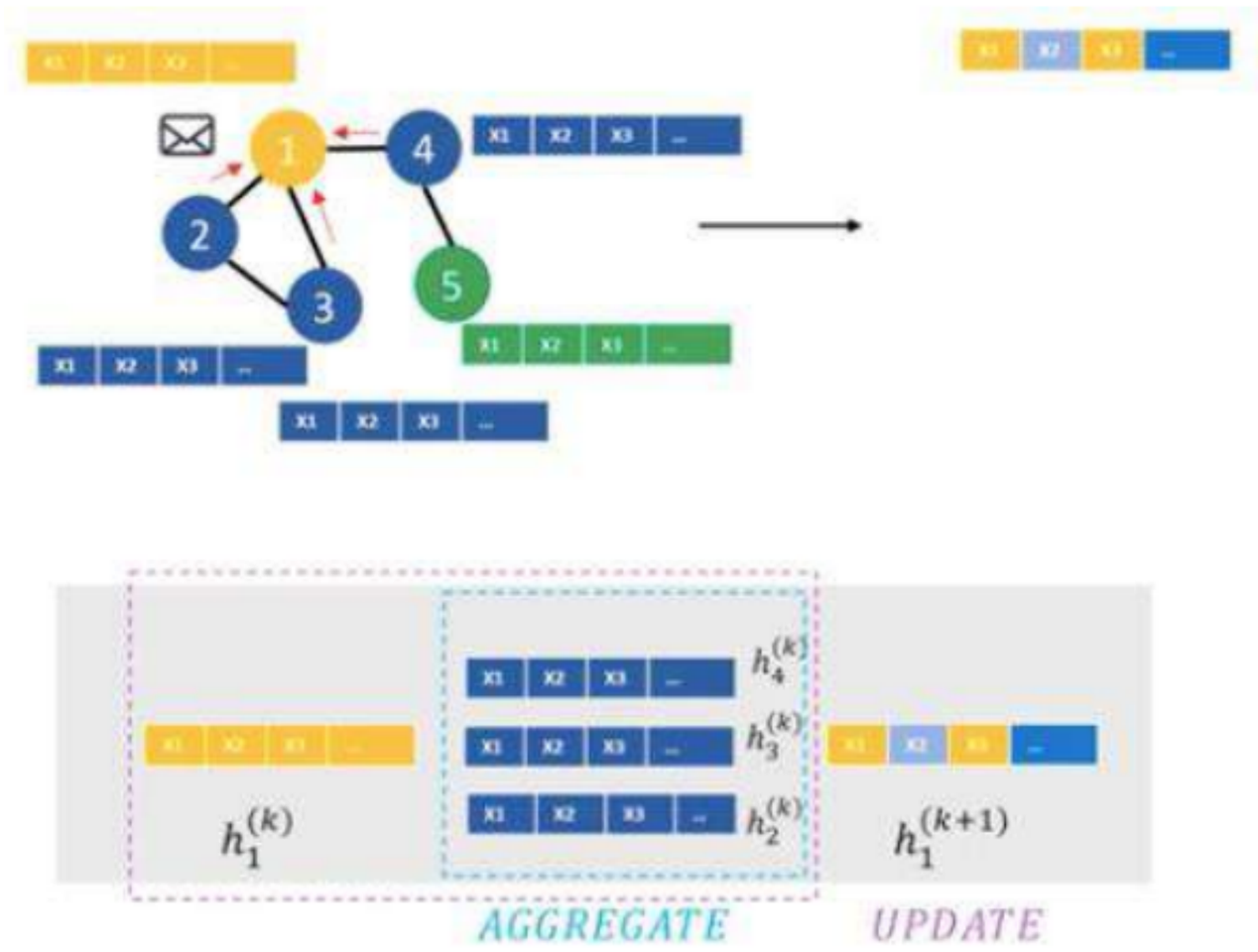
AGGREGATE

The Message Passing Mechanism: Sharing Information

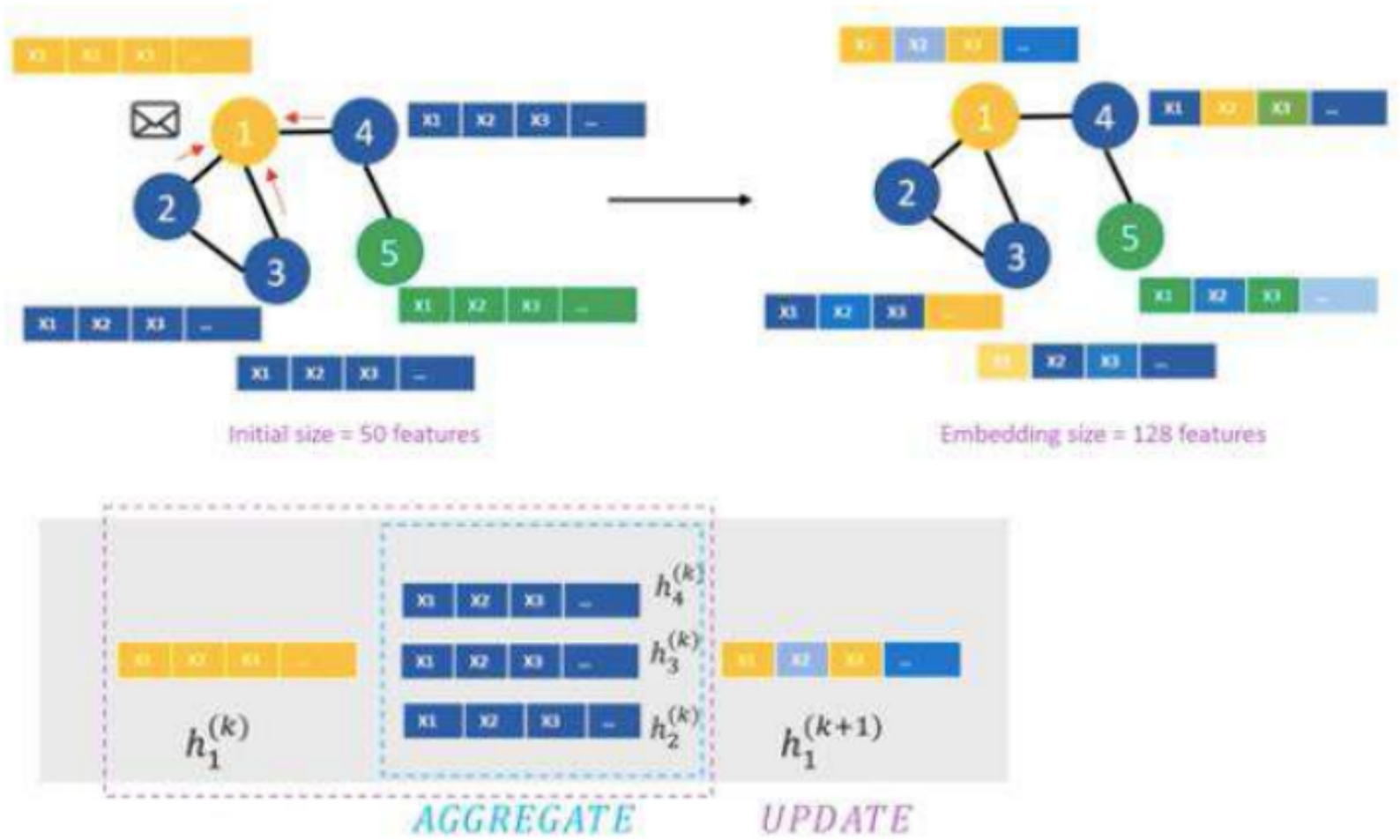




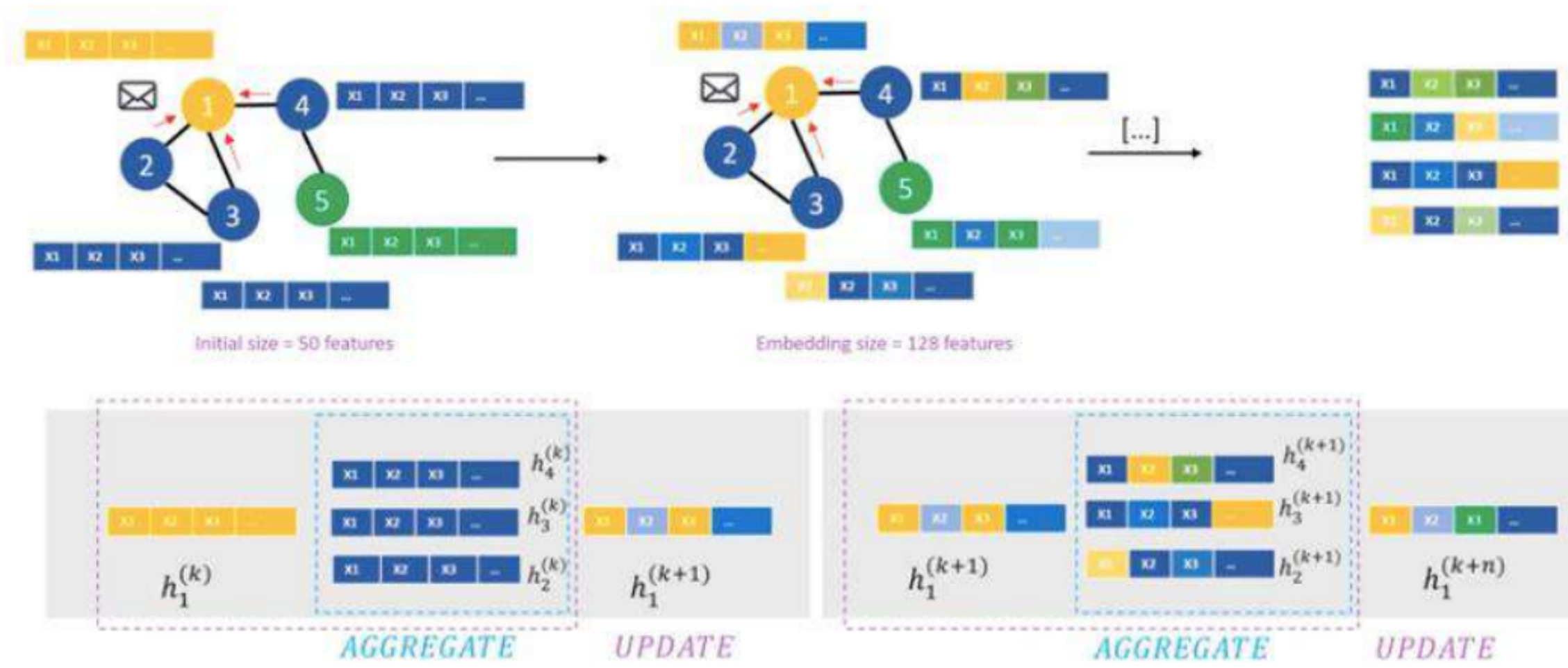
The Message Passing Mechanism: Sharing Information



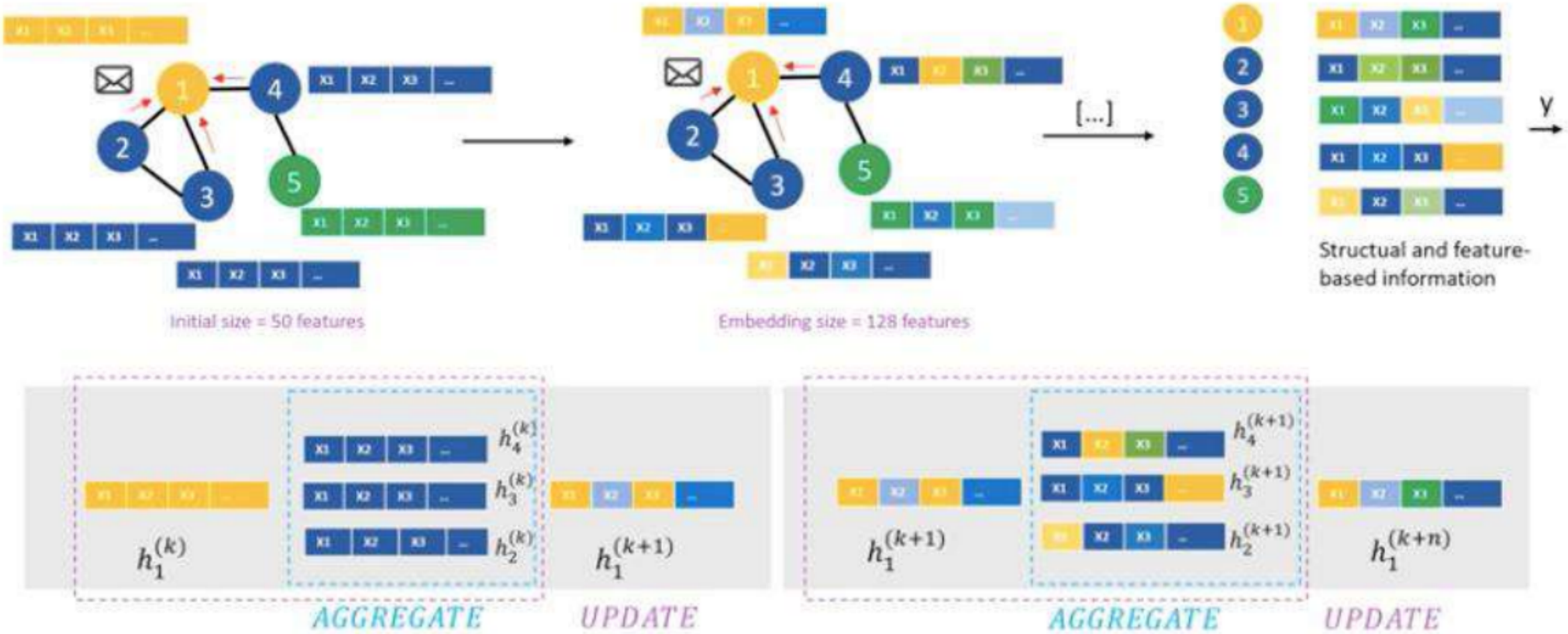
The Message Passing Mechanism: Sharing Information



The Message Passing Mechanism: Sharing Information



The Message Passing Mechanism: Sharing Information



Iterative Refinement: Layers and Embeddings

Multiple Layers: The message passing and aggregation steps are repeated for multiple layers.
Refined Representations: With each layer, node representations become richer, incorporating information from larger neighborhoods.
Final Embeddings: After the final layer, each node has a low-dimensional embedding capturing its essence and relationships within the graph.
Similar to traditional neural networks, GNNs often have multiple layers. Each layer refines the node representations by incorporating information from a wider neighborhood. After the final layer, each node has a "fingerprint" called an embedding, encapsulating its unique characteristics and position within the graph.

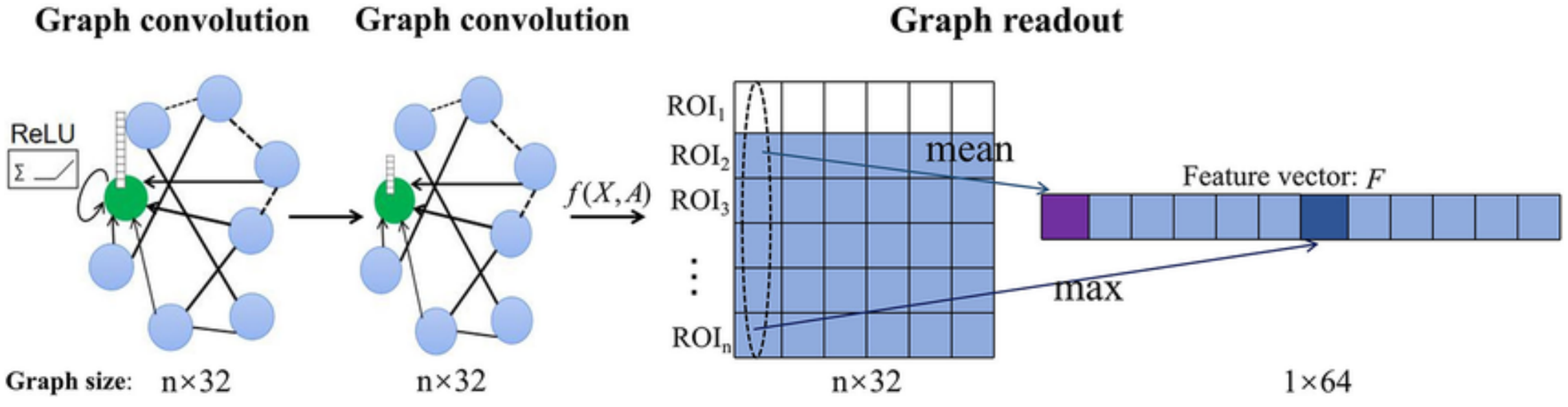
Iterative Refinement: Layers and Embeddings

The choice of activation functions and the number of layers can significantly impact the expressiveness and performance of the GNN. Selecting appropriate settings depends on the specific task and data characteristics.

Readout Function

After several layers, we have individual node embeddings capturing local information. But for many tasks, we need a single output representing the entire graph.

This is where the readout function comes in. It takes the node embeddings and summarizes them into a single vector that reflects the overall graph properties.



Readout Function

Common readout functions include:
Summation: Simply averages the embeddings of all nodes.
MaxPooling: Takes the maximum value across all node embeddings.
Attention-based: Uses learned weights to focus on the most relevant nodes in the graph.

The choice of readout function depends on the specific task and data characteristics.
For example, summation might work well for tasks like graph property prediction, while attention can be useful for identifying influential nodes.

Readout Function

This final output vector can then be used for various tasks:
Classification: Predict a class label for the entire graph (e.g., fraudulent transaction network).
Regression: Predict a continuous value related to the graph (e.g., protein-protein interaction strength).
Link Prediction: Predict the likelihood of a link existing between two nodes.
Remember, the readout function plays a crucial role in transforming individual node information into a meaningful graph-level output.

Putting It All Together

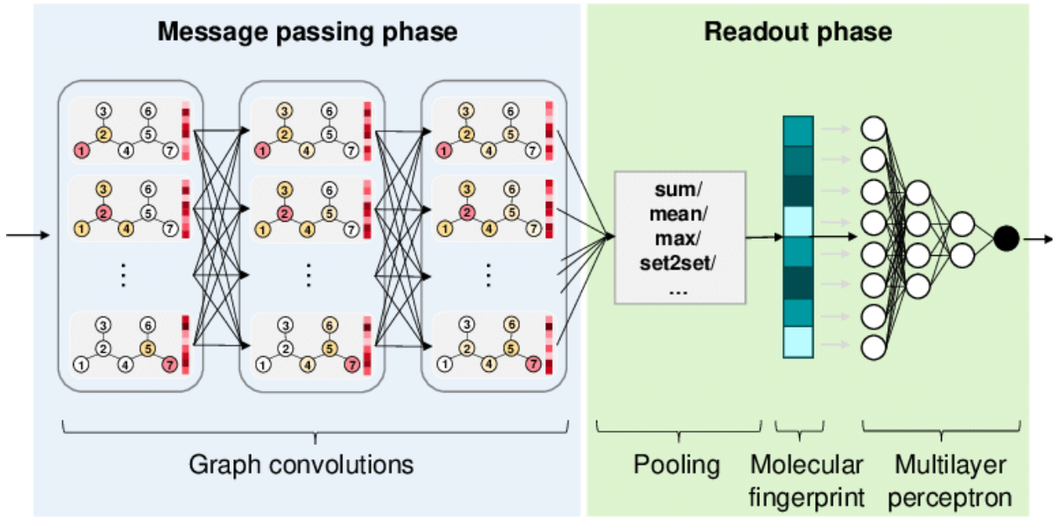
Lets Recap.

We've explored the key steps of a GNN architecture:

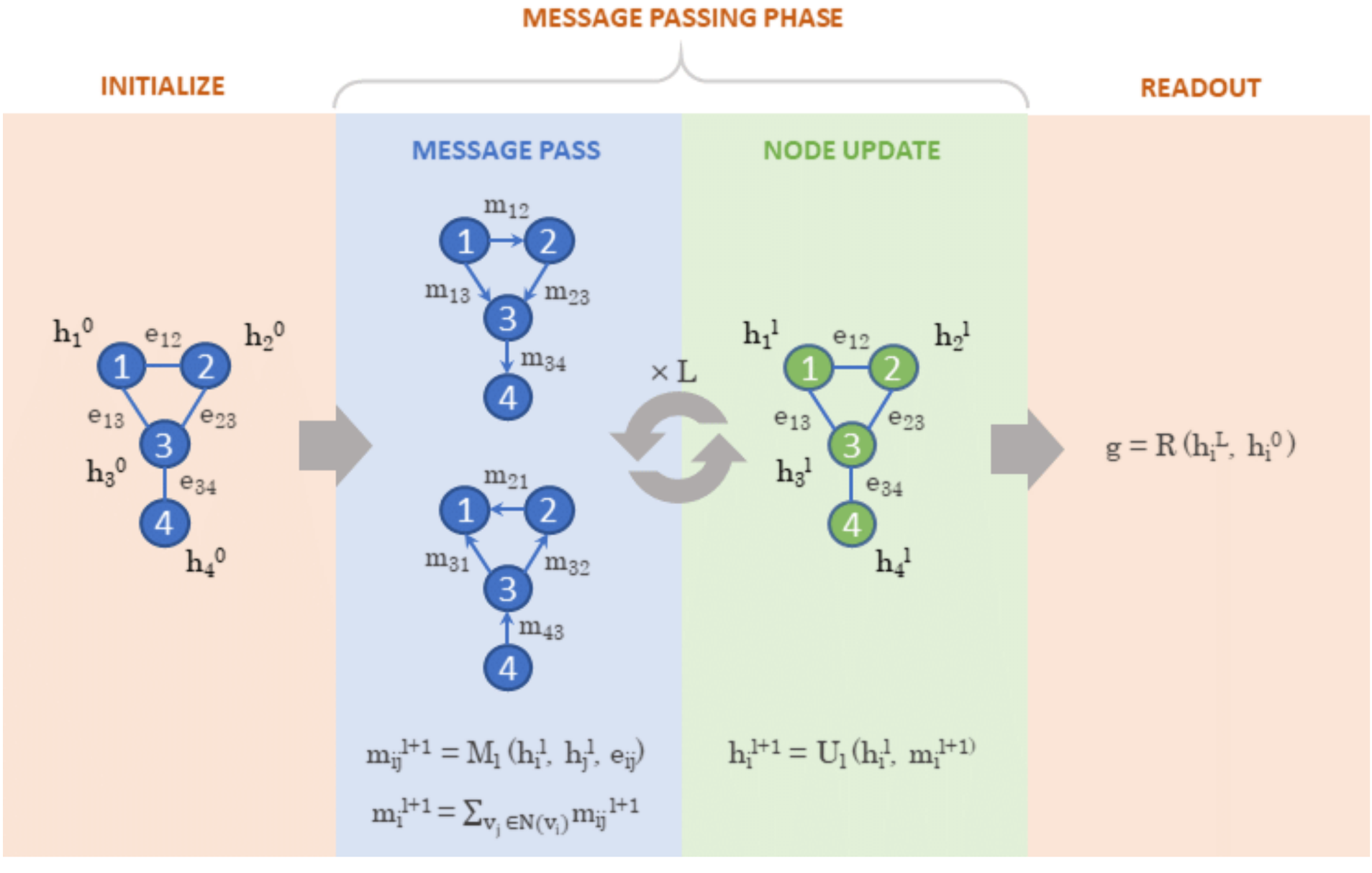
Message passing: Nodes exchange information with neighbors.

Aggregation: Nodes update their representations based on received messages.

Readout: A function summarizes node embeddings into a single output vector.



Putting It All Together



Some Examples

Prediction Examples-

Node Classification: Imagine a social network analysis GNN. The final output vector could be used to predict the overall sentiment of the network (positive, negative, or neutral).

Link Prediction: A GNN trained on protein-protein interaction data could use the output vector to predict if two specific proteins are likely to interact.

Community detection: Identifying groups of nodes that are highly interconnected within a network.

Some Examples

Prediction Examples-

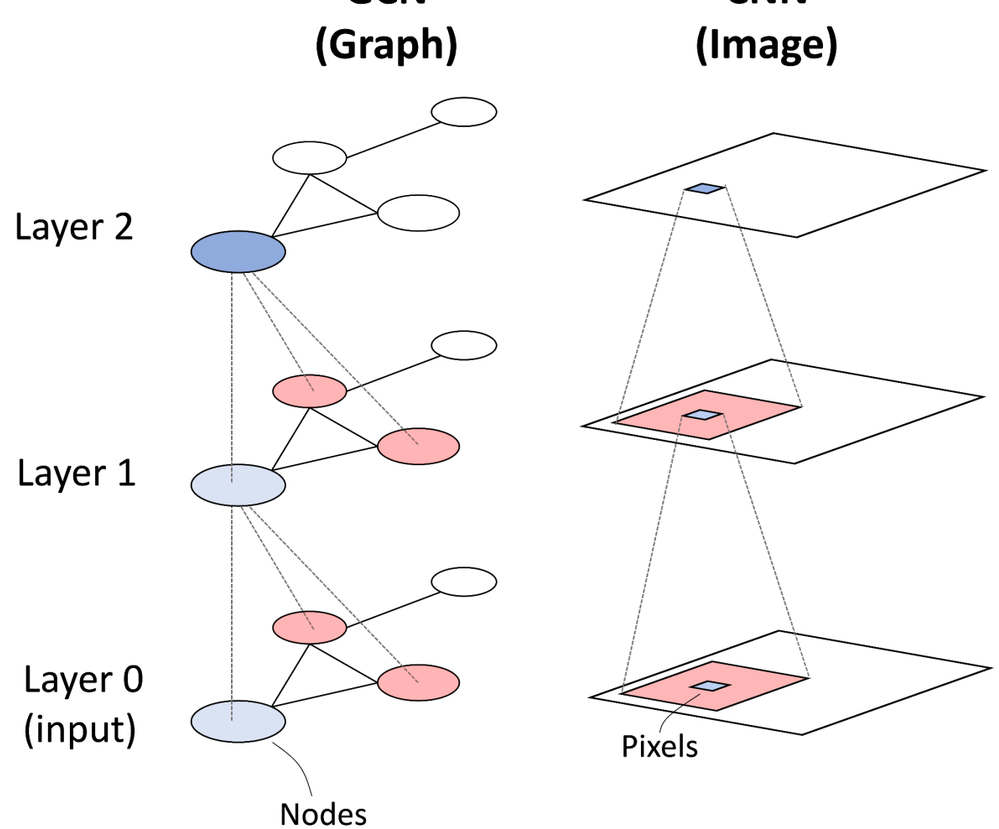
Anomaly detection: Detecting unusual patterns in a network that may indicate fraudulent activity or other issues.

Molecular property prediction: Predicting the properties of molecules based on their structure and interactions with other molecules.

Scene graph generation: Analyzing images to understand the relationships between objects and people within a scene.

Graph Convolutional Networks (GCNs): A Specific Kind of GNN

GCNs are a specific type of Graph Neural Network (GNN) inspired by Convolutional Neural Networks (CNNs).
They excel at analyzing grid-like data structures like images or protein interaction networks.



Graph Convolutional Networks (GCNs): A Specific Kind of GNN

Key advantages of GCNs-

- Efficiency: Handle large graphs efficiently due to their focused architecture.
- Interpretability: Easier to understand their decision-making process due to simpler message passing mechanisms.
- Domain-specific: Particularly effective for grid-like data, making them valuable tools in specific fields like image processing and bioinformatics.

Graph Convolutional Networks (GCNs): A Specific Kind of GNN

GCNs use a unique convolution operation for message passing between nodes.
Imagine each node "convolving" its features with its neighbors', capturing the local structure of the grid.

Feature	GCNs	GNNs
Message passing	Convolution-based, efficient for grid-like data	More general, flexible message passing functions
Layers	Often use same type of layer in each iteration	Can use diverse layer types for more flexibility
Generalizability	More focused on grid-like structures	More generalizable to various graph types

Message Passing in GCNs

- GCNs use a convolution operation inspired by CNNs. This operation considers the node's own features and aggregates information from its immediate neighbors in the grid-like structure.
- GNNs, on the other hand, can employ more flexible message passing functions that may not rely on convolution and can consider information from further neighbors.

Layer Types in GCNs

- GCNs typically use the same type of convolutional layer in each iteration, focusing on local information aggregation within the grid.

- GNNs offer more flexibility by allowing different layer types with varying message passing mechanisms and aggregation functions, potentially capturing more complex relationships.

Generalizability

- GCNs excel with **grid-like structures** due to their convolution operation and focus on local information.

- GNNs are more **generalizable** and can handle diverse graph structures, making them suitable for various data types beyond grids.

The GCN Advantage

Despite the differences, GCNs offer unique advantages in specific scenarios:

- Efficiency: Their focused message passing and layer structure make them efficient for handling large grid-like graphs.
- Interpretability: The convolution operation and reliance on local information make it easier to understand how GCNs make decisions, which can be valuable for debugging and analysis.

Choosing the Right Tool: GCNs vs. GNNs

The choice between GCNs and GNNs depends on your specific needs-

Use GCNs:

When dealing with grid-like data (images, protein structures).
When efficiency and interpretability are crucial.
When understanding local relationships within the grid is key.

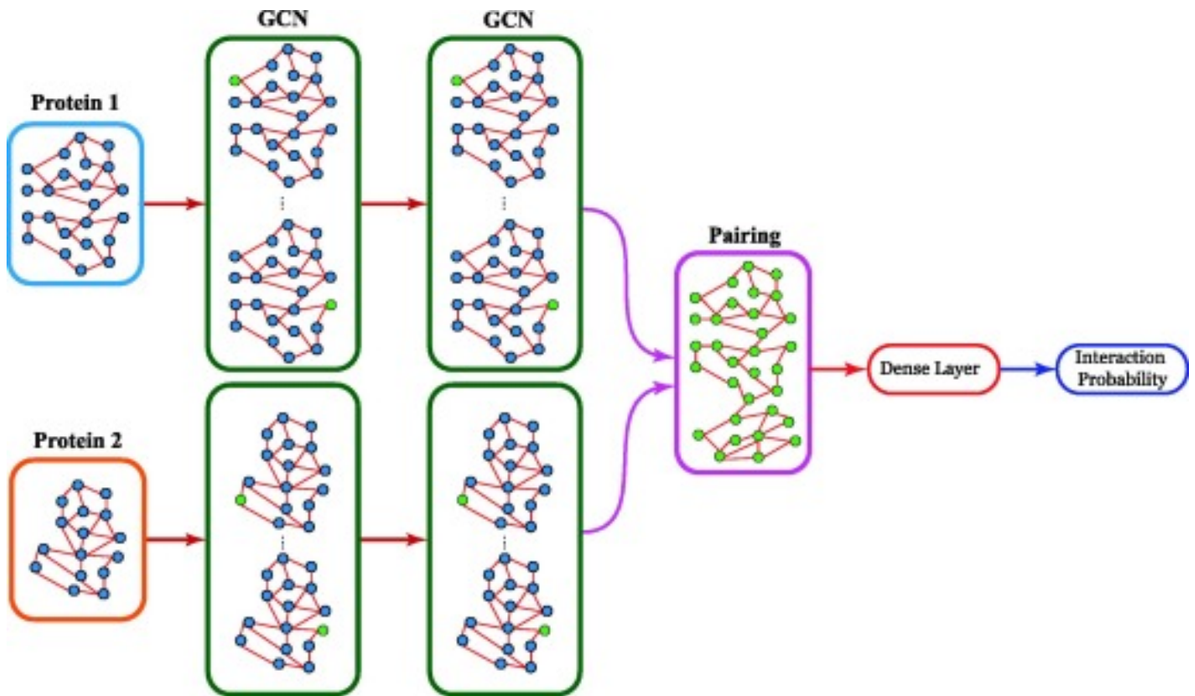
Use GNNs:

For diverse graph structures beyond grids.
When more complex relationships and information flow are important.
When flexibility and generalizability are essential.

Applications of GCNs

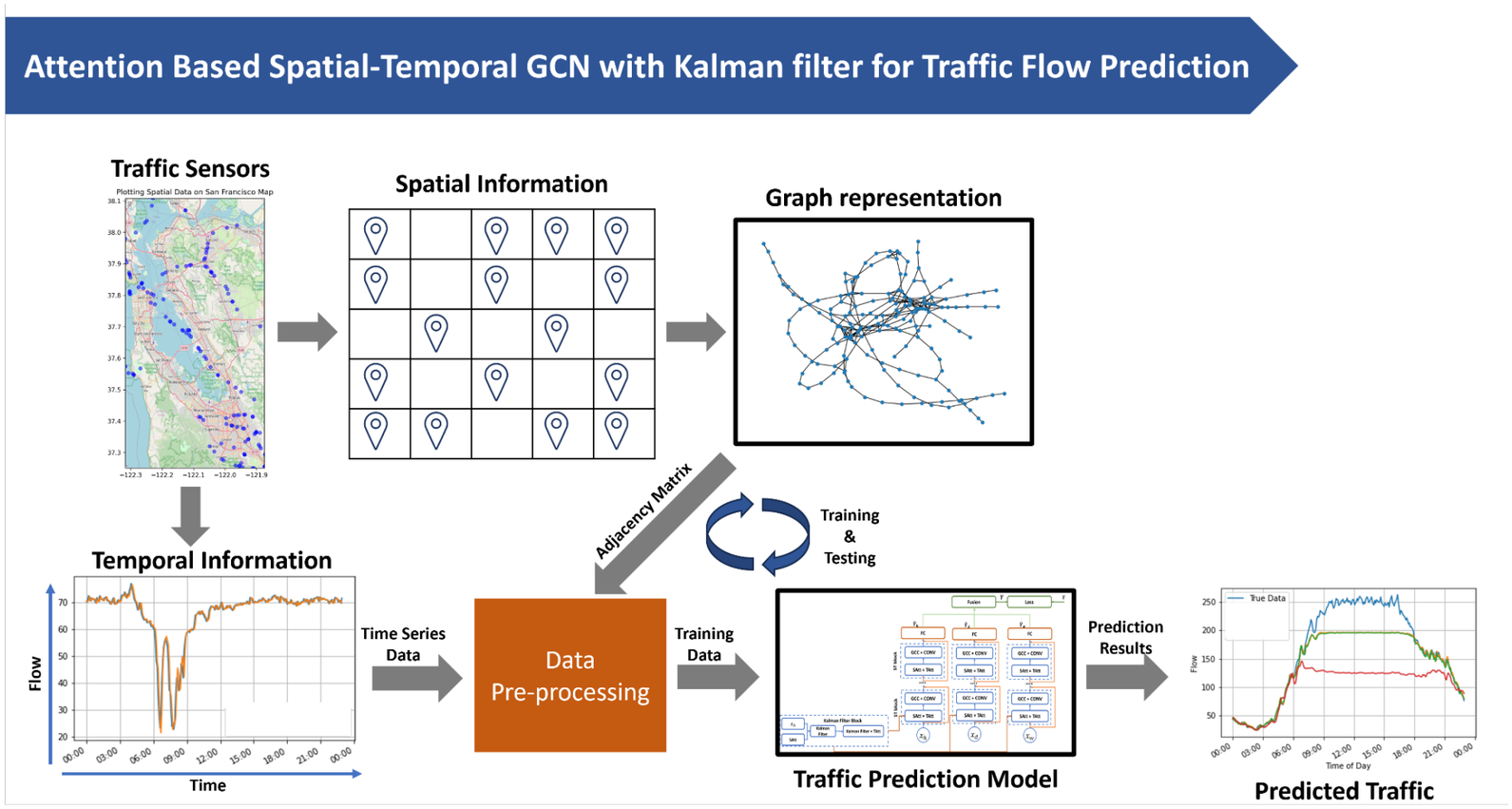
GCNs unlock power in specific domains with grid-like data:

- Image Classification: Analyze images by understanding pixel relationships, e.g., classifying objects in a scene.
- Protein Interaction Prediction: Predict how proteins interact within grid-like structures, aiding drug discovery.



Applications of GCNs

- Traffic Flow Analysis: Model traffic flow in road networks with grid-like layouts, optimizing traffic management.



Applications of GNNs

The applications of GNNs are vast and span diverse domains. From social media to drug discovery, GNNs are unlocking new possibilities for data analysis and prediction in various fields.

- Social Network Analysis: Node classification (e.g., identifying influential users), link prediction (e.g., recommending friendships).
- Drug Discovery: Predicting molecule properties (e.g., toxicity, bioactivity) for drug design.
- Traffic Flow Prediction: Forecasting traffic congestion in transportation networks.

Applications of GNNs

- Recommendation Systems: Recommending products, movies, or music based on user-item relationships.
- Protein-Protein Interaction Prediction: Understanding protein interactions in biological systems.

Acknowledgements & References

- https://medium.com/dair-ai/an-illustrated-guide-to-graph-neural-networks-d5564a551783
- Graph Neural Networks: A Review and Open Issues: https://arxiv.org/abs/2108.10733
- Introduction to Graph Neural Networks. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers, 2020.
- Applications of Graph Neural Networks (GNN) - Jonathan Hui
- https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0279604
- Graph Neural Network Applications and its Future - XenonStack: https://www.xenonstack.com/blog/graph-neural-network-applications

THANK YOU