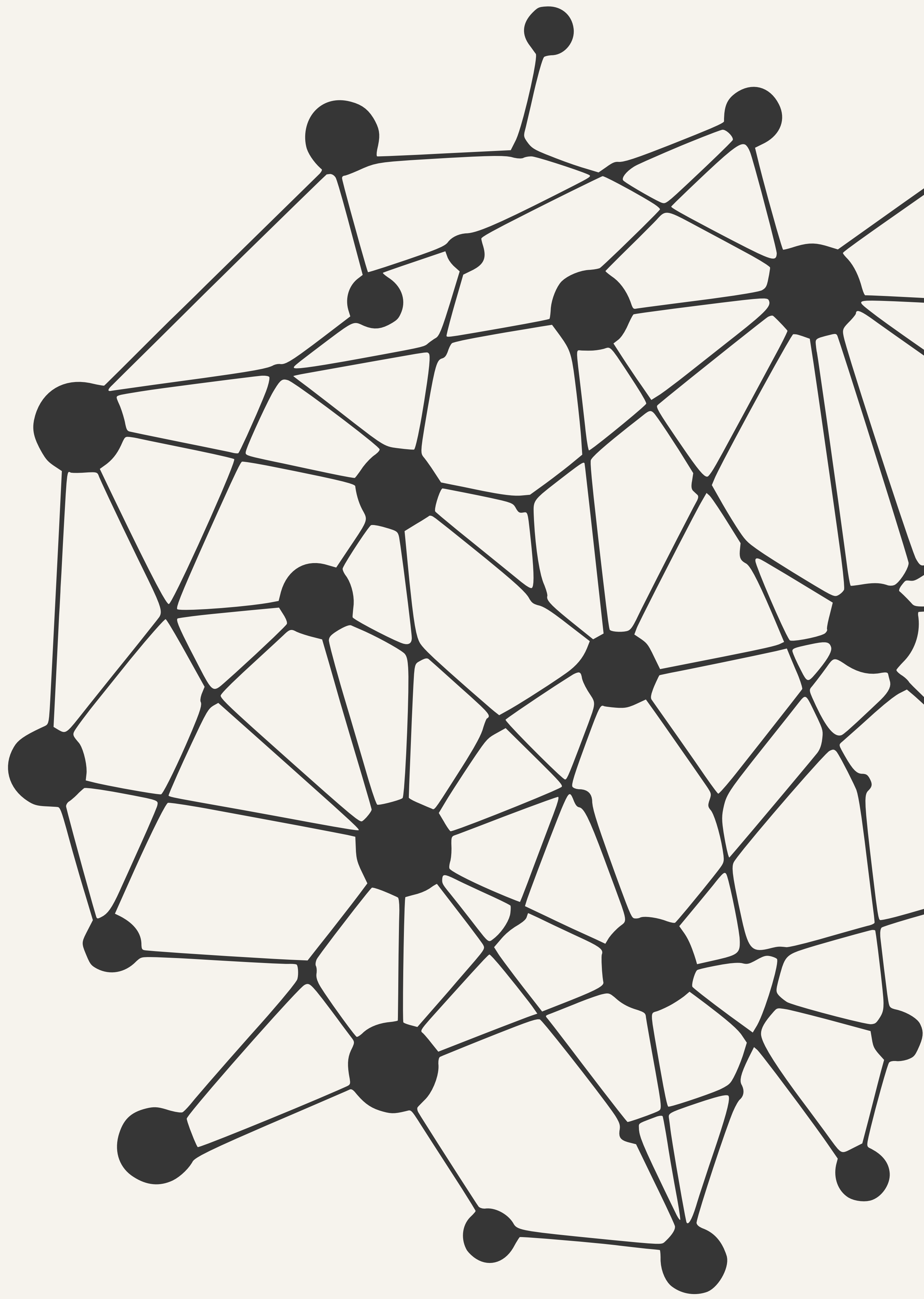
GRAPH NEURAL NETWORKS

An introduction to GNNs and GraphNeuralNetowrks.jl

x



THE PROJECT

GraphNeuralNetworks.jl is a **Julia**-based graph neural network library built on Flux.jl. Key features include standard graph convolutional layers, batched graph computations, and customizable layer creation. It supports CUDA for GPU acceleration and seamlessly integrates with Graphs.jl
The toolkit covers diverse machine learning tasks at node, edge, and graph levels, including complex graphs like heterogeneous and temporal graphs.

GraphNeuralNetworks.jl is an **open source** project, so if you like it and want to contribute in any way you're welcome!

More information on the website:
https://github.com/CarloLucibello/GraphNeuralNetworks.jl

THIS GUIDE

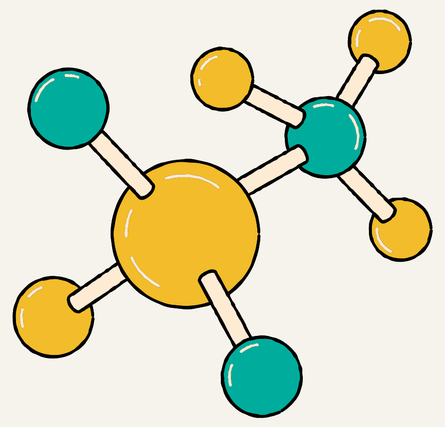
It’s for those who do not know and want to discover the world of Graph Neural Networks with the use of a library (written in Julia).
You will find parts of **theory** followed by practical examples with **code snippets**.
This is not intended to be a comprehensive guide but rather a **quick start guide** with both theory and practice.

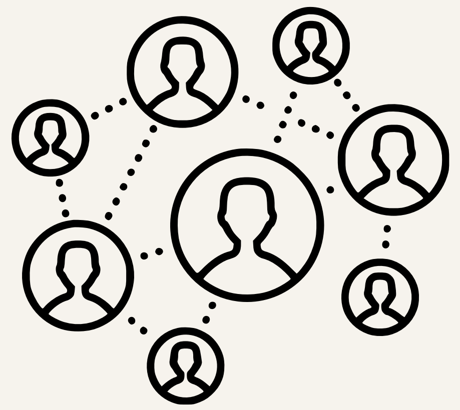
For further information, tutorials and examples see the Github repository (https://github.com/CarloLucibello/GraphNeuralNetworks.jl) and the **documentation**

GRAPHS ARE EVERYWHERE

Today GNNs (Graph Neural Networks) find application in many fields, below we present some of them.

GNNs find extensive applications in **chemistry**. Computational chemistry commonly models molecules as graphs, where atoms represent nodes and atomic bonds represent edges. These models can estimate pharmacological activities, chemical properties, and other molecular characteristics.





GNNs in **recommendation systems** (as in social networks) leverage graphs to enhance recommendation accuracy. In a recommendation graph, nodes can represent users, and edges can indicate relationships between users.

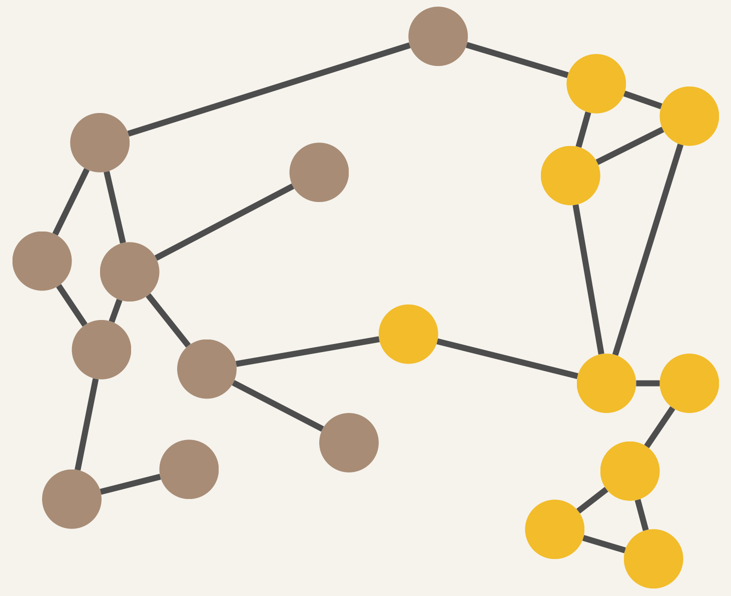
In NLP(Natural Language Processing), particularly in **text** sentiment analysis, GNNs are used to explore syntactic and semantic relationships among words in text. GNNs can consider the context of words and capture complex correlations, enhancing sentiment understanding and analysis.

GNNs can be applied in many other fields!

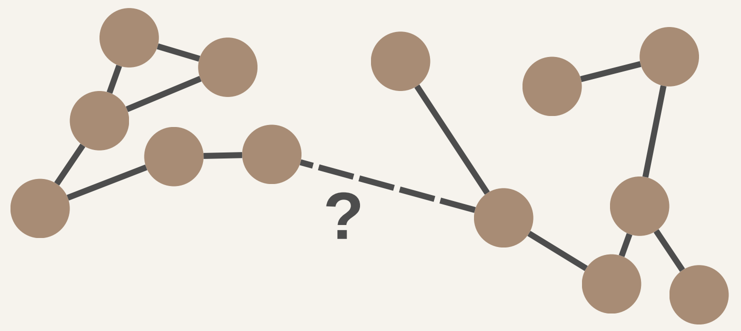
WHAT PREDICTIONS?

Ok, graphs can be anywhere! But what kind of tasks can we carry out with a GNN?

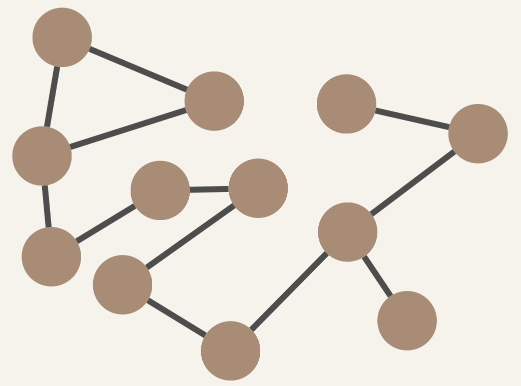
Node predictions:
We aim to predict node properties. A classic example is the Zachary's Karate Club problem (see tutorial in the documentation on GitHub). Node-level predictions are analogous to the concept of image segmentation, where we attempt to label the role of each pixel in an image, here we want to classify the nodes.



Edge predictions:
We want to predict the presence or properties of a link between two nodes. For instance, in a social network, we might predict the formation of a friendship between two nodes (users) based on some information about the two nodes.



Graph predictions:
We aim to predict a property of the entire graph. If we have a graph representing the structure of molecules, we can predict global chemical properties of the molecule, such as solubility, reactivity, or other relevant molecular characteristics.



GNN VS FNN

FNNs (Feedforward Neural Networks) work well with regularly structured data, such as images or text sequences, while GNNs are designed to handle data with a more complex and relational structure, like graphs.
This is why GNNs are said to be an extension of FNNs.

However there are some important **differences**:

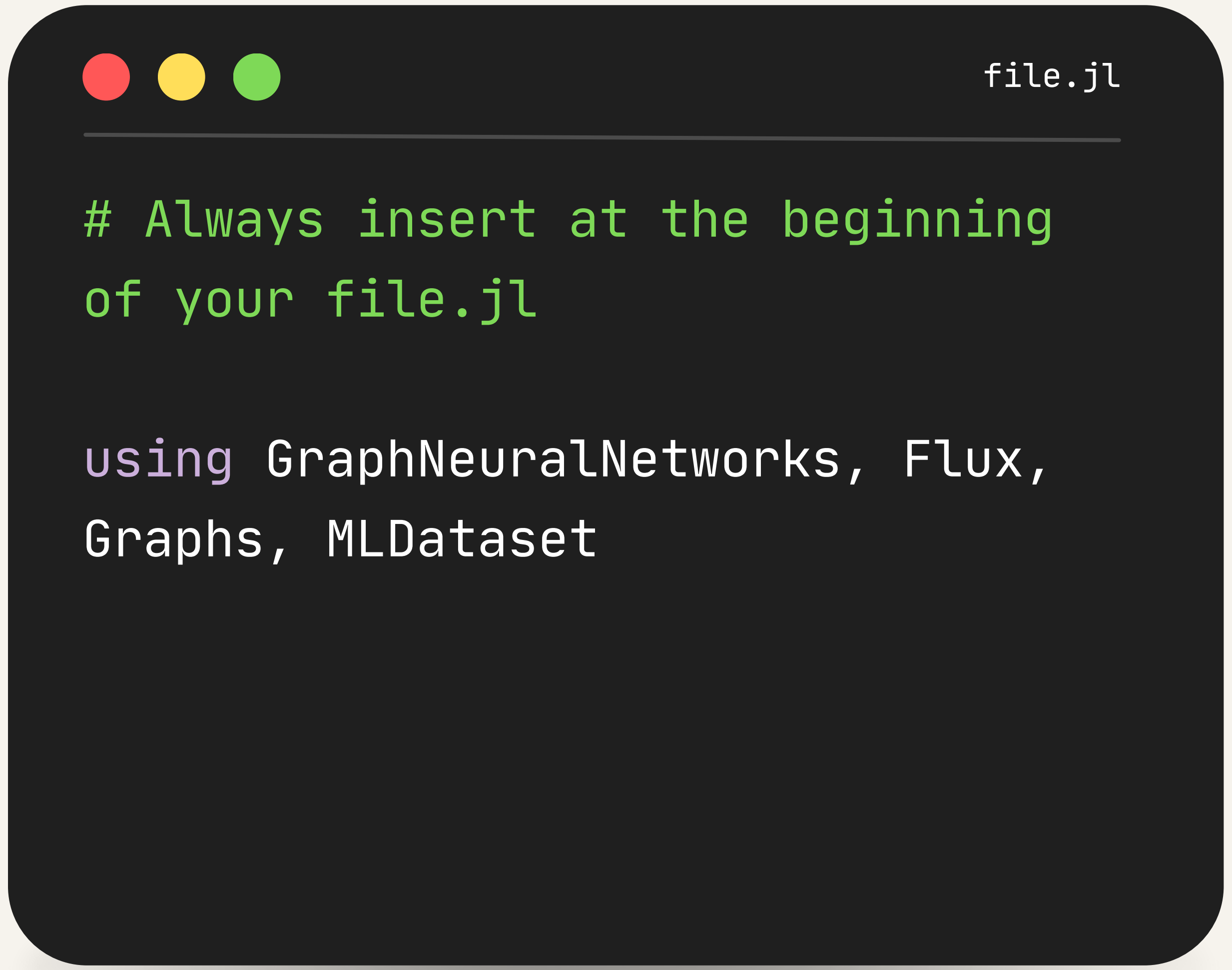
- **Variable dimensions**: FNNs require fixed-size inputs (for example, an image with fixed dimensions of 32x32 pixels). GNNs handle graphs with different sizes that can have a variable number of nodes and edges.

- **Permutation invariant**: If we change the order of inputs in an FNN (for example, the pixels of an image), we change the meaning of the input and consequently the output. With GNNs, since the order of nodes in a graph doesn’t alter the relationships among them, we say that GNN is permutation-invariant.

- **Non-euclidean space**: The pixels of an input image in an FNN can be represented on a grid of coordinates (x,y) and it's possible to calculate the Euclidean distance between them. In a GNN, the edges don't have coordinates, so it doesn't make sense to calculate the distance between two nodes in a classical sense.

THE PAKAGES

As mentioned GraphNeuralNetworks.jl is a library written in Julia, so you need to install Julia on your computer first. Below are some of the main libraries useful for GNNs



file.jl

```julia
# Always insert at the beginning
of your file.jl

using GraphNeuralNetworks, Flux,
Graphs, MLDataset
```

GraphNeuralNetworks
A graph neural network library written in Julia

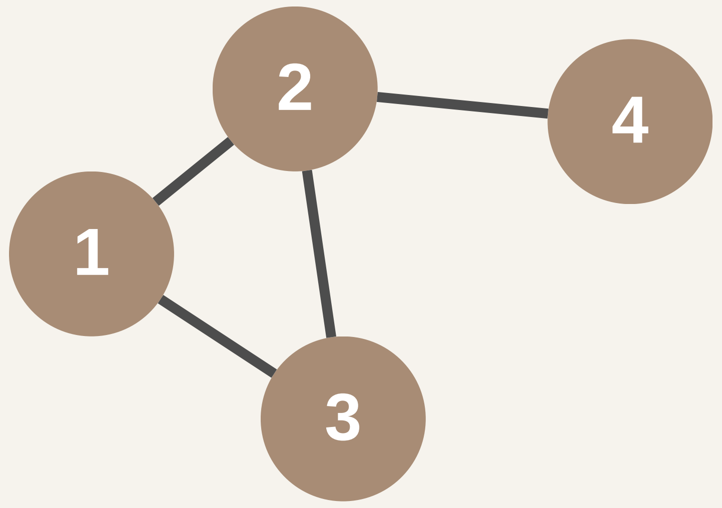
Flux
Deep learning library in Julia

Graphs
Library for graph manipulation and analysis in Julia.

MLDatasets
Provides access to common datasets for machine learning in Julia.

Those are just the main libraries for working with GNNs; as you progress, you might need others!

THE GRAPH



The main elements are:

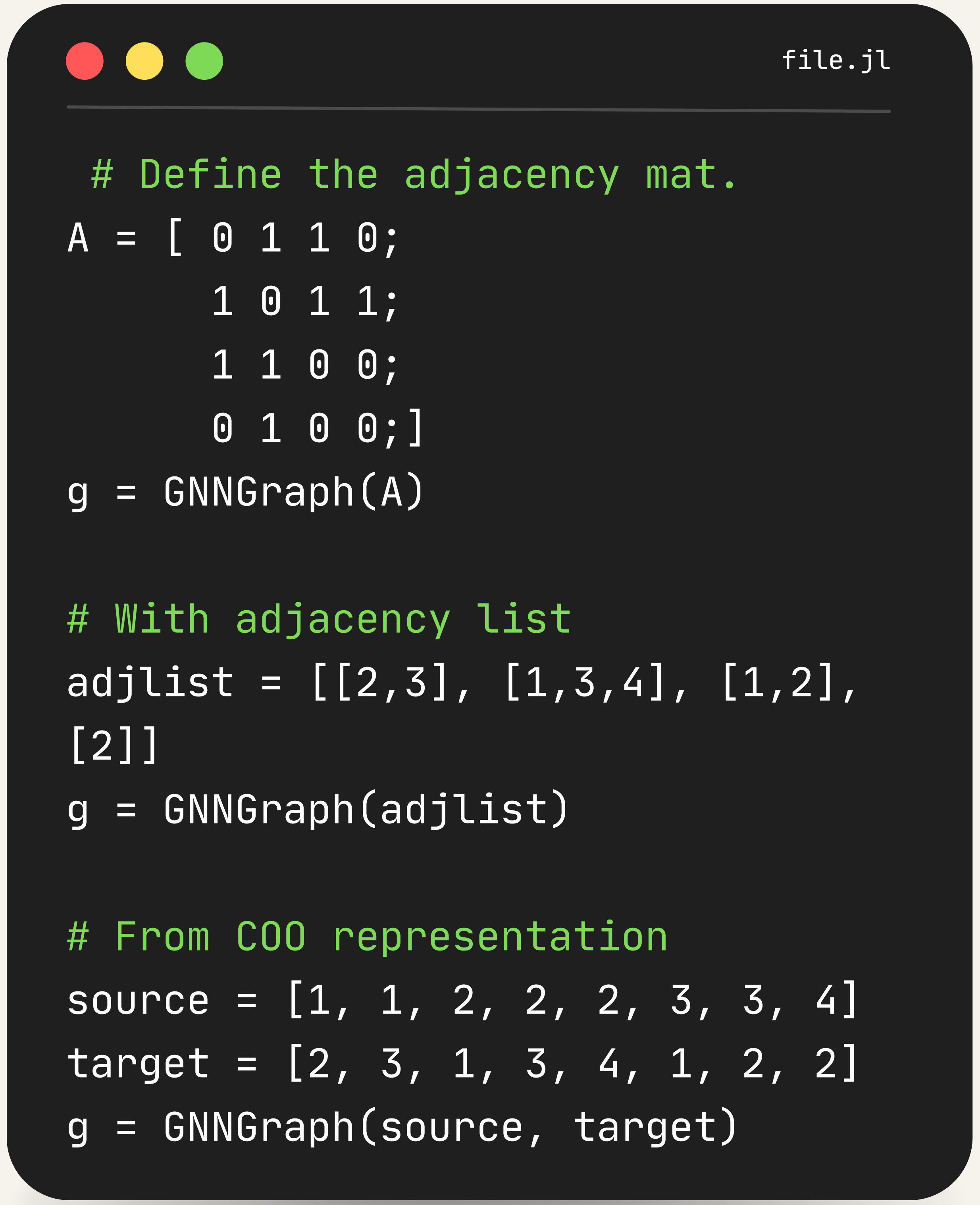
- The **graph** (often denoted by 'g')
- The **nodes** or vertices (numbered for convenience)
- **Archs** (can be unidirectional if they have arrows or bidirectional)

We can create the graph in various ways:
With adjacency matrix, adjacency list, CCO representation

	V1	V2	V3	V4
V1	0	1	1	0
V2	1	0	1	1
V3	1	1	0	0
V4	0	1	0	0

Adjacency matrix A 4x4, dim ~ n^2

The matrix is symmetric because the graph is bidirectional.
On the diagonal there are 0 because there are no self loops.
If I have many nodes, this is not the best way to represent a graph because it would be sparse matrix and takes up a lot of space.



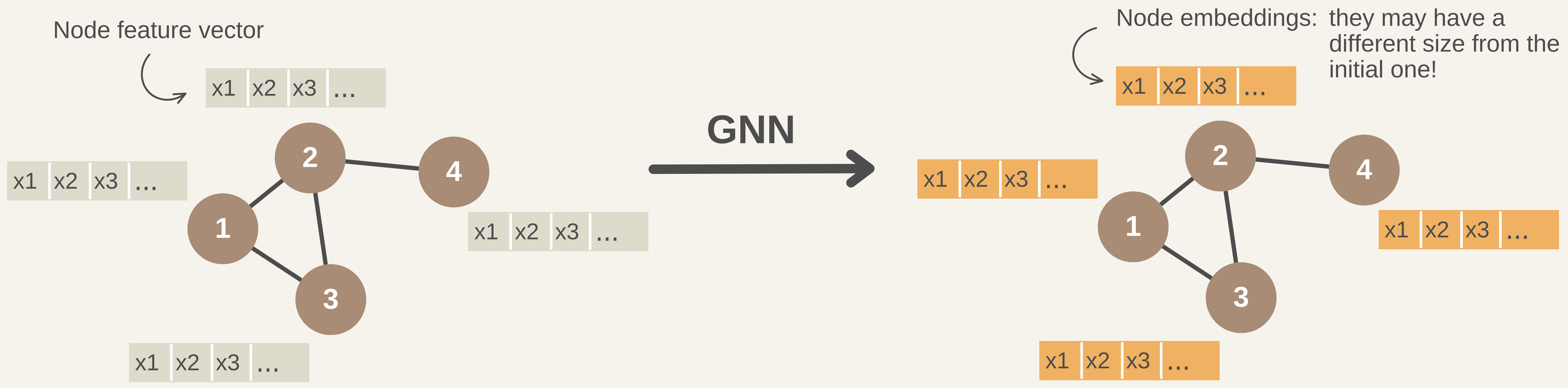
```julia
 # Define the adjacency mat.
A = [ 0 1 1 0;
      1 0 1 1;
      1 1 0 0;
      0 1 0 0;]
g = GNNGraph(A)

# With adjacency list
adjlist = [[2,3], [1,3,4], [1,2],
[2]]
g = GNNGraph(adjlist)

# From COO representation
source = [1, 1, 2, 2, 2, 3, 3, 4]
target = [2, 3, 1, 3, 4, 1, 2, 2]
g = GNNGraph(source, target)
```

FEATURE VECTORS

....

REPRESENTATION LEARNING

Graph Neural Networks (GNNs) focus on **Representation Learning**, aiming to capture relevant information from near (and not only) nodes and edges within a graph. Through iterative processing in layers, such as layers based on Multi-Layer Perceptron (MLP), GNNs acquire a deeper and more meaningful understanding of the global structure of the graph. The main goal is to generate representations, known as **node embeddings**, that compactly and informatively capture the features and relationships of nodes within the graph context.
These representations are essential for subsequent tasks such as classification, property prediction, or other learning activities involving graph structures.



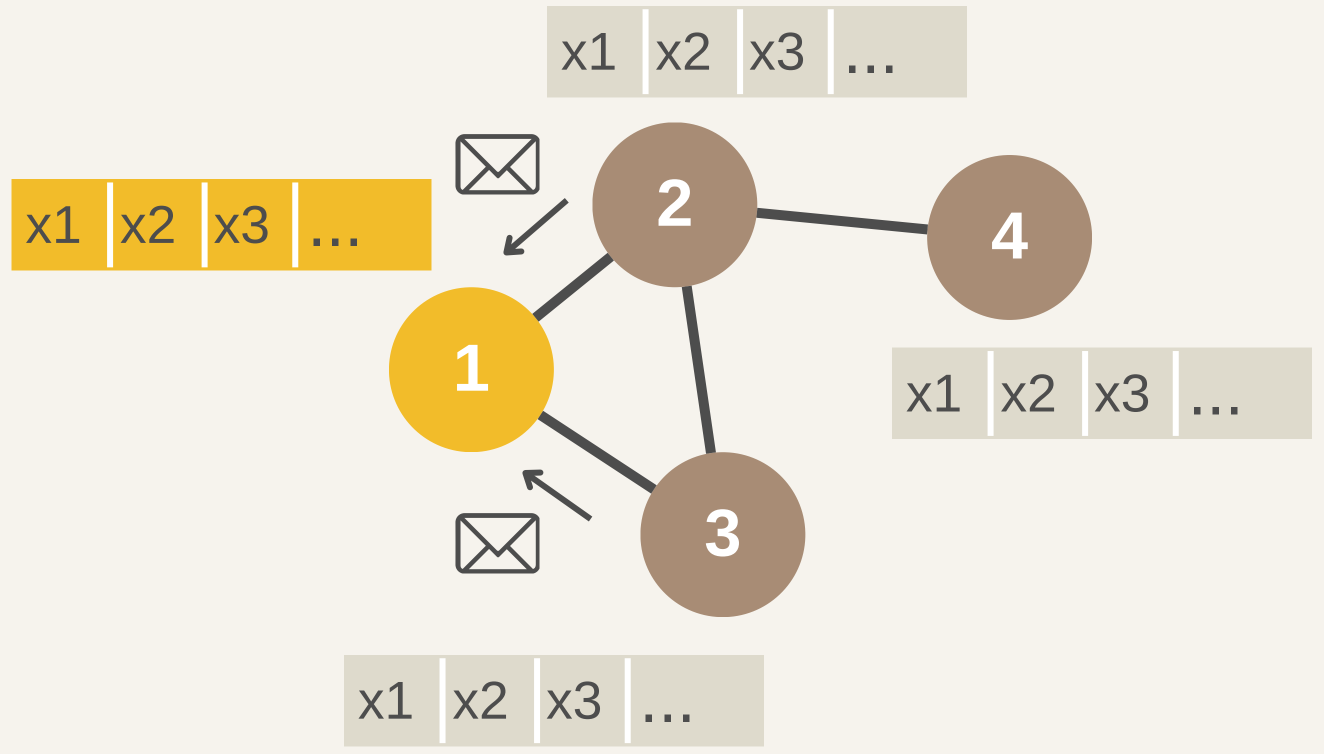
MESSAGE PASSING

We have mentioned that the GNN must create a node embedding that can encapsulate as much information as possible about the graph and its neighboring nodes. But how do the nodes communicate with each other and exchange information? This is where the **message passing layer** comes into play, which has two main tasks:

Aggregate: It involves collecting and combining information from neighboring nodes. This step involves processing messages coming from the incoming edges.

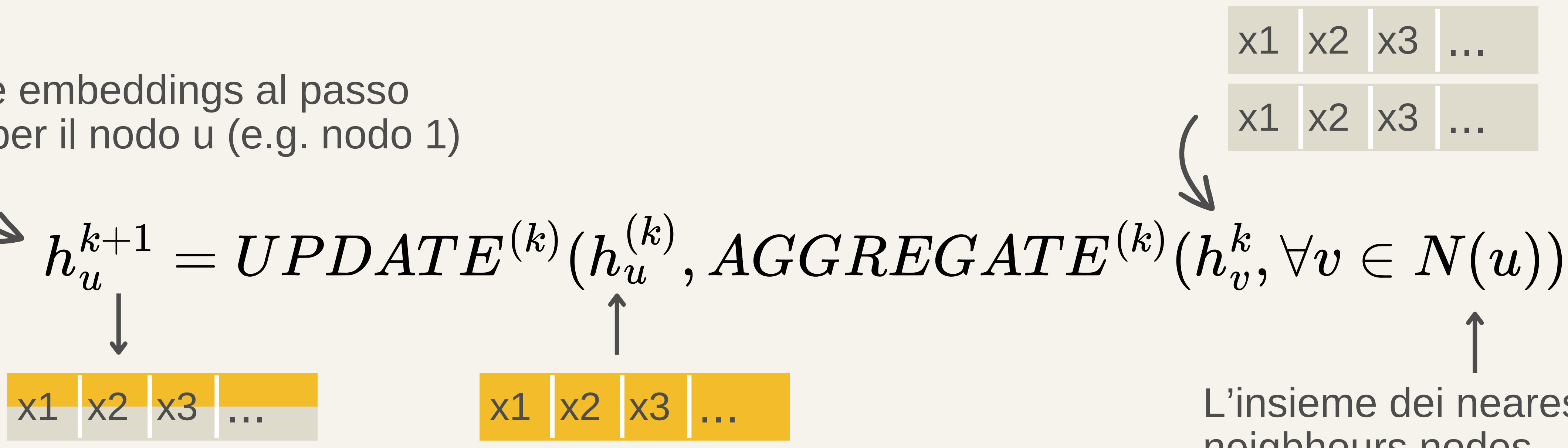
Update: it is performed on the nodes themselves and updates their representation by incorporating the aggregated information.

if we focus on node 1:



Node embeddings al passo k+1 per il nodo u (e.g. nodo 1)

$$h_u^{k+1} = UPDATE^{(k)}(h_u^{(k)}, AGGREGATE^{(k)}(h_v^k, \forall v \in N(u))$$



L'insieme dei nearest neigbhours nodes

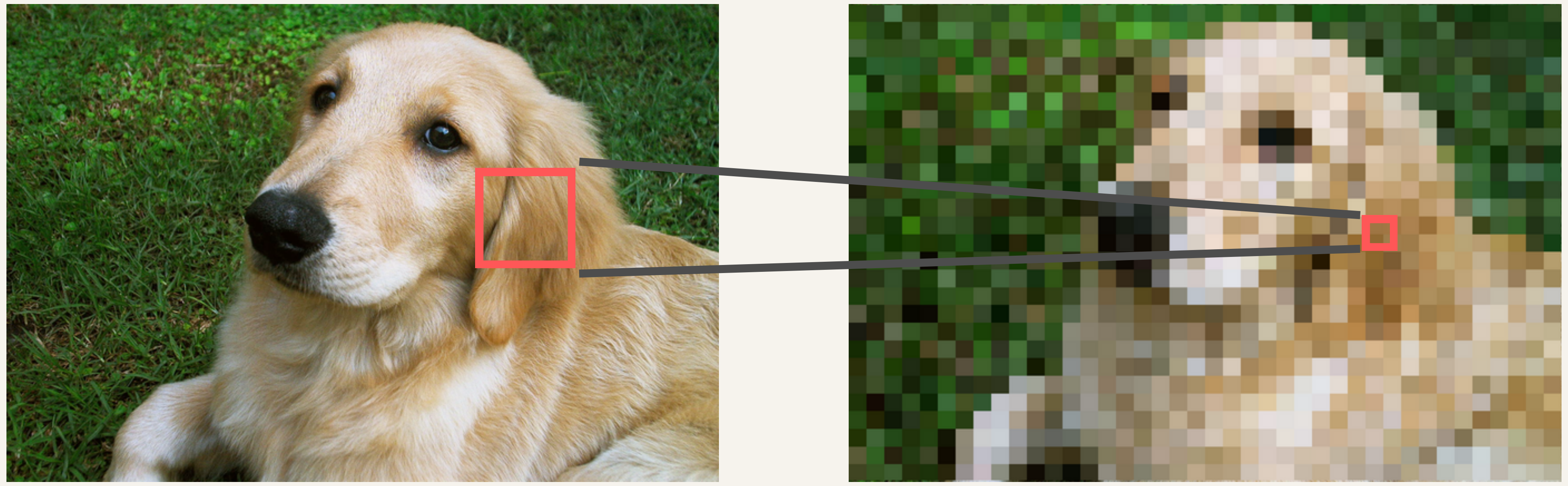
(In this example k=1, meaning each node is taking the information of its nearest neighbor nodes.)

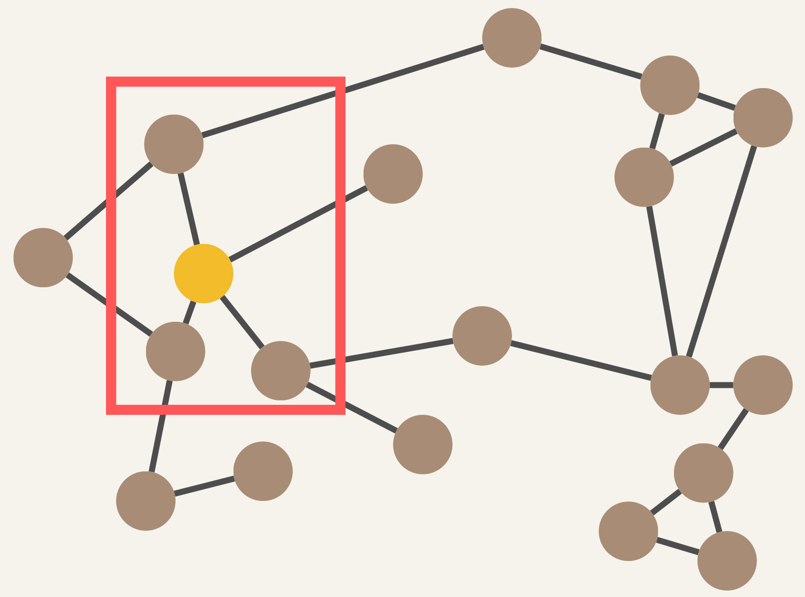
COMPUTATIONAL GRAPH

....

CONVOLUTIONS

In Convolutional Neural Networks (CNNs), the convolution operation applied to an image utilizes a kernel to examine small areas of the image at a time, capturing local relationships among pixels.
This process enables the extraction of relevant features, such as lines, shapes, or patterns, from different regions of the image. Similarly, in Graph Neural Networks (GNNs), the message passing layers aggregate information from neighboring nodes using an approach akin to the convolution kernel in CNNs.
The message passing layers scrutinize the neighbors of each node, capturing local relationships in the graph and updating the representations of the nodes (node embeddings).





to continue.......