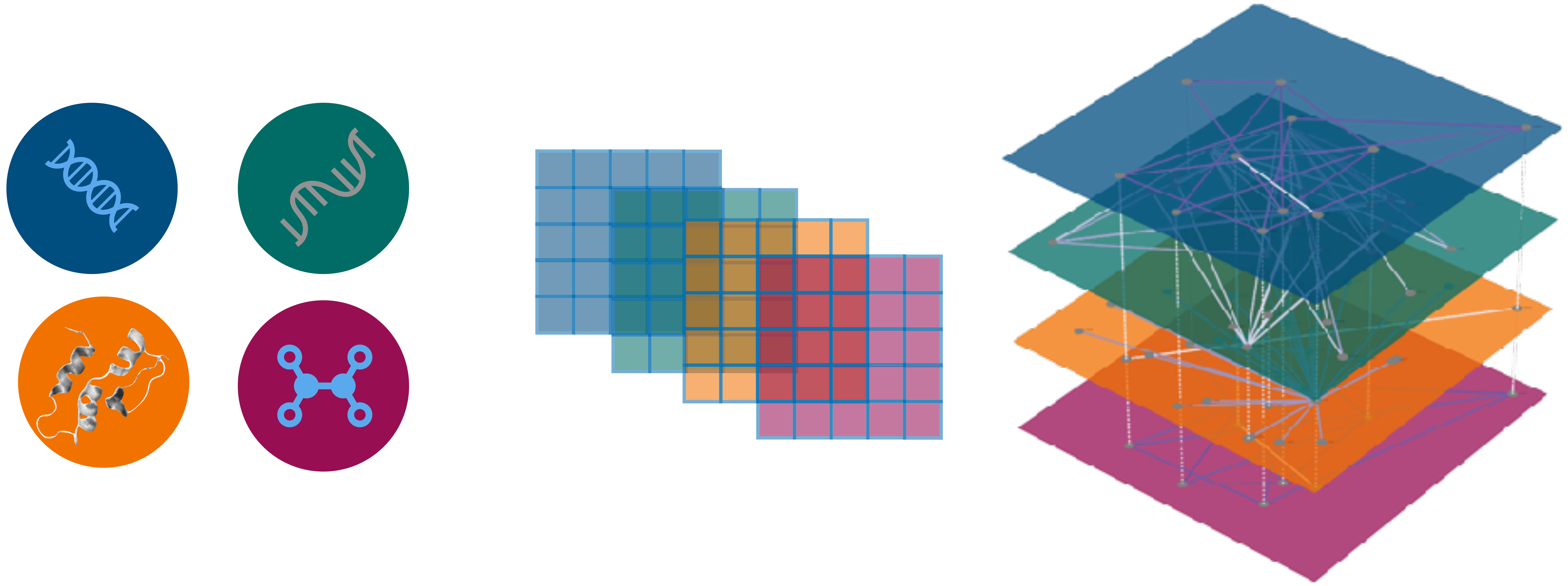


# Graph Neural Networks

## AI-guided Protein Science

Alberto Santos — Multi-omics Network Analytics (MoNA)

The Novo Nordisk Foundation
Center for Biosustainability

# Limitations of Shallow Embeddings

- **Complexity grows** with the number of nodes ($O(|V|)$) — problem with large graphs

- They **do not use node features**

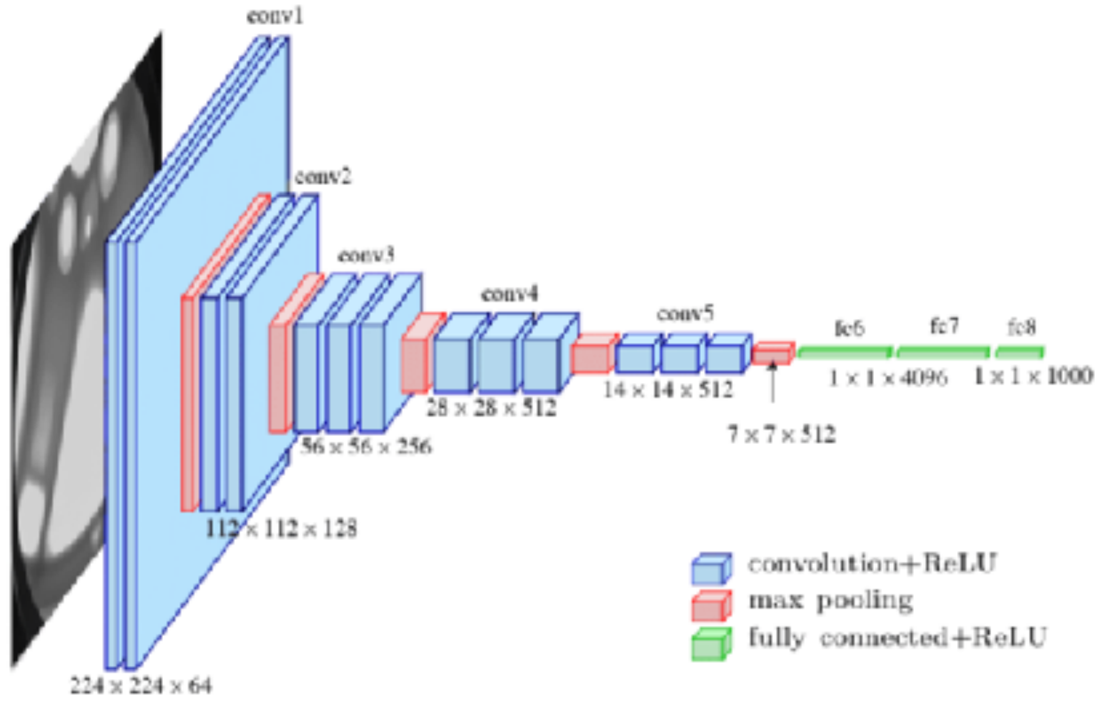
- They are **transductive** — can only generate embeddings for nodes present during the training phase (no generalizable to unseen nodes)

# More Complex Encoder Models

- **Graph Neural Networks** — a deep learning framework for representing graph data
  - Incorporate both local and global context
  - Allows node features (e.g., expression)
  - Inductive
- Different to **traditional DL**

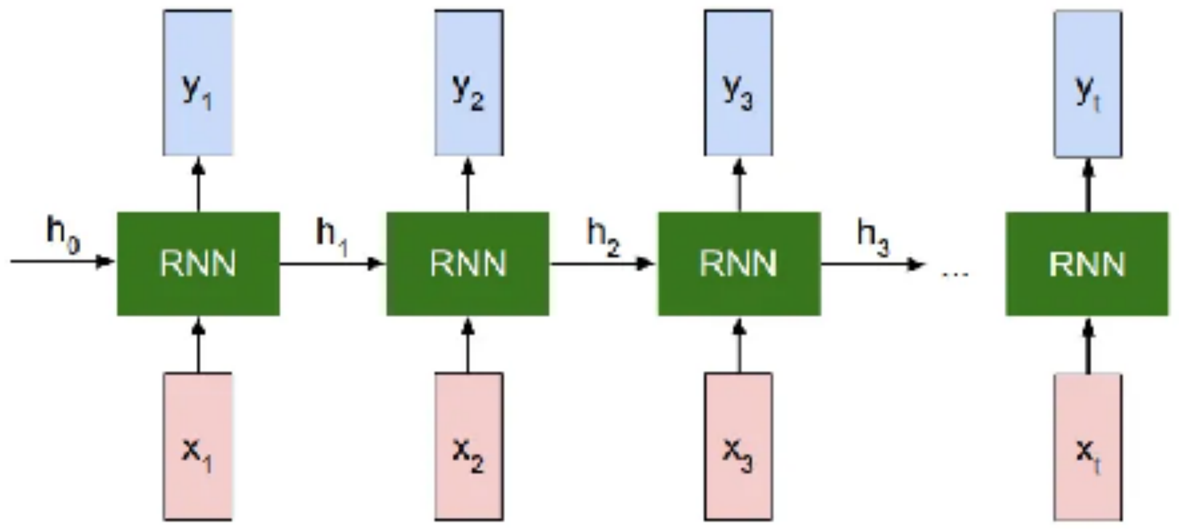
**Grid structures**

CNNs



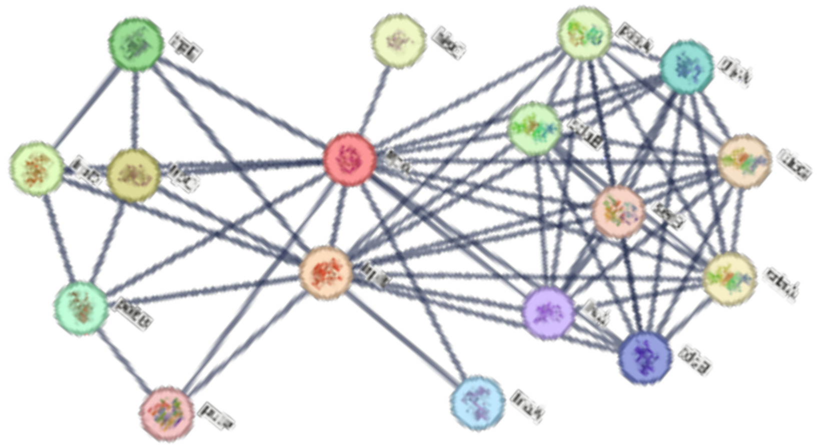
**Sequences**

RNNs



**Graph structures**

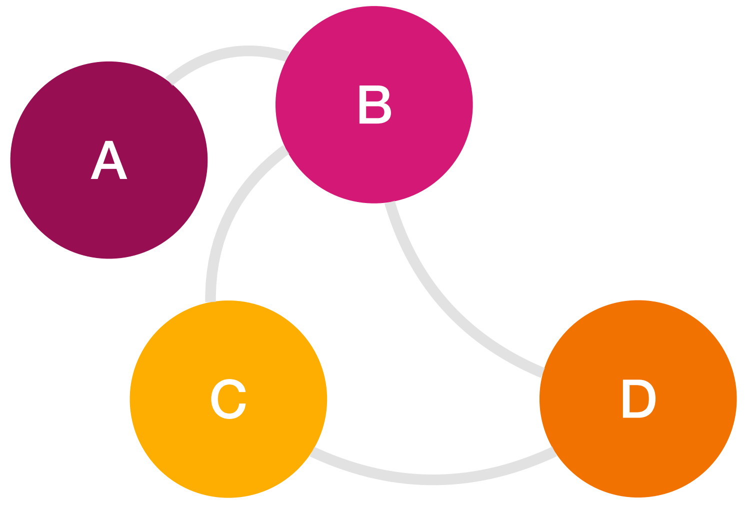
GNNs



?

# Using the Adjacency Matrix

- Why not using the Adjacency Matrix ***A***?
  - Assumptions for DL
    - **Permutation invariance**: same output regardless of the order of its input.
    - **Permutation equivariance**: the output reflects the permutation applied to the input.
  - ***A*** has an arbitrary order of nodes



| | A | B | C | D |
|---|---|---|---|---|
| A | 0 | 1 | 0 | 0 |
| B | 1 | 0 | 1 | 1 |
| C | 0 | 1 | 0 | 1 |
| D | 0 | 1 | 1 | 0 |

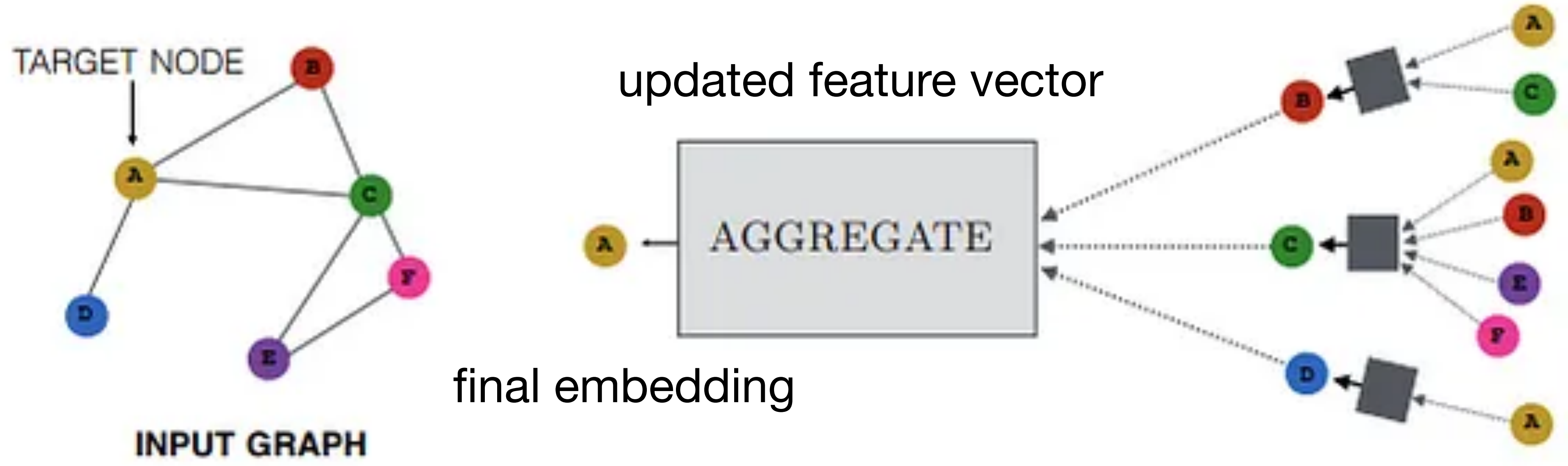
| | B | A | D | C |
|---|---|---|---|---|
| A | 1 | 0 | 0 | 0 |
| B | 0 | 1 | 1 | 1 |
| C | 1 | 0 | 1 | 0 |
| D | 1 | 0 | 0 | 1 |

# Neural Message Passing

- Enables information exchange and aggregation among nodes capturing the dependencies and interactions among them
- The information collected from neighbours is used to update the representation of the node

At each iteration (layer) of the message passing in a GNN, a hidden embedding $h_u^{(k)}$ is updated for each node *u* based on information gathered from its graph neighbourhood N(u).

As these iterations progress, each node embedding contains more and more global information

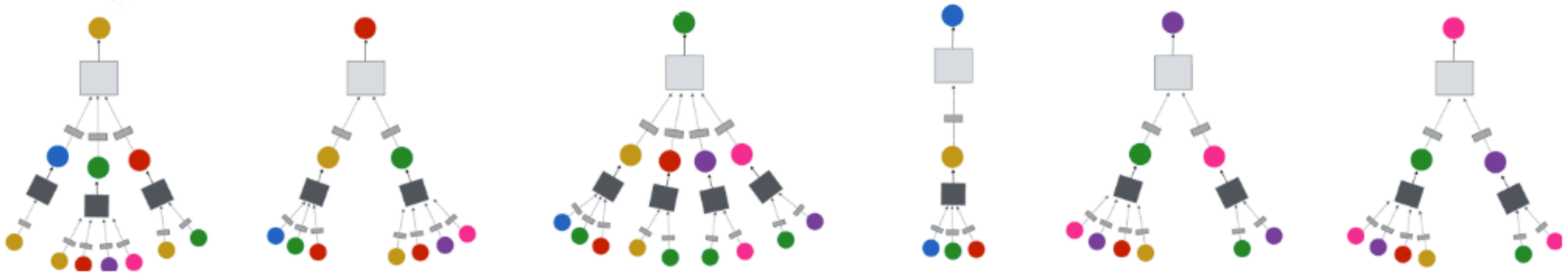


Node + feature vector

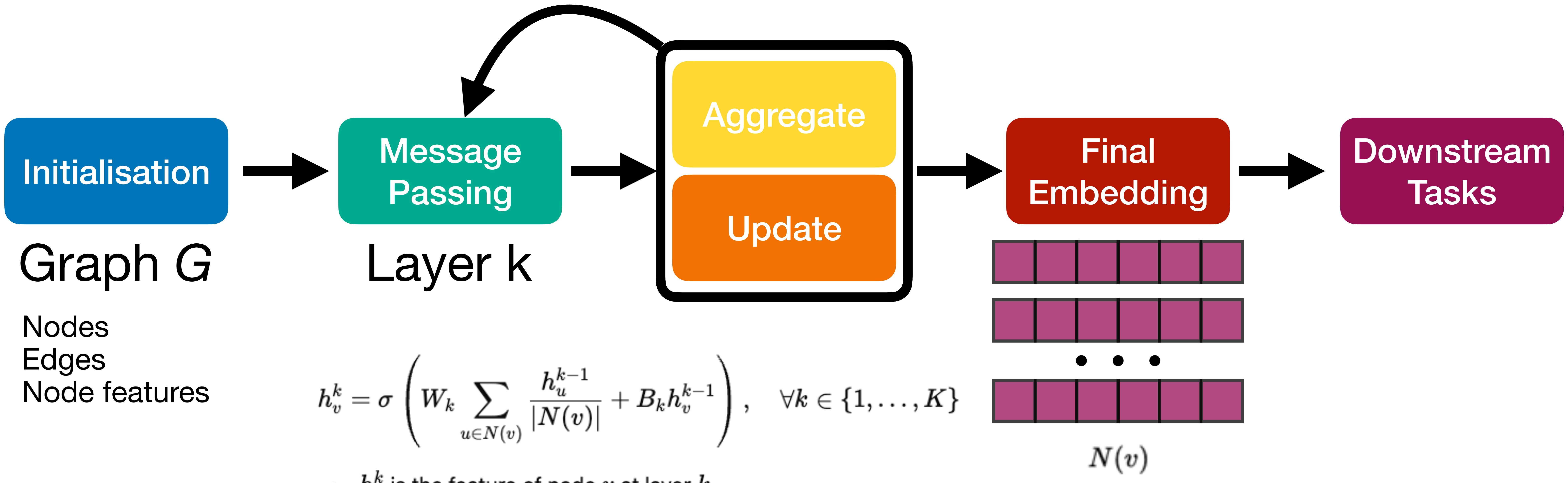
updated feature vector

final embedding

Each node defines a computational graph based on its neighbours



# GNN Workflow

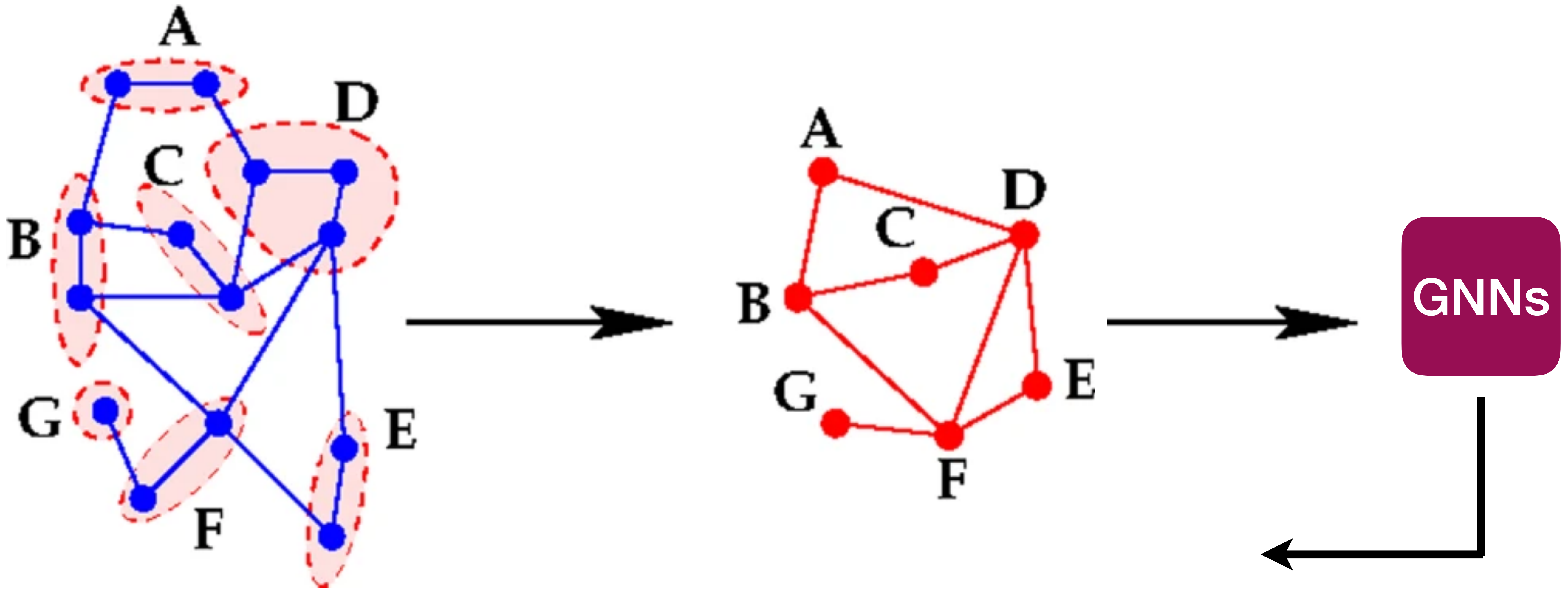


$$h_v^k = \sigma\left(W_k \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(v)|} + B_k h_v^{k-1}\right), \quad \forall k \in \{1, \ldots, K\}$$

- $h_v^k$ is the feature of node $v$ at layer $k$,
- $N(v)$ is the set of neighbors of node $v$,
- $h_u^{k-1}$ is the feature of node $u$ at layer $k-1$,
- $|N(v)|$ is the number of neighbors of node $v$,
- $W_k$ and $B_k$ are learnable weight matrices and bias at layer $k$,
- $\sigma$ is a non-linear activation function (e.g., ReLU or Sigmoid).

# Graph Pooling

- **Graph-level** tasks — Define a **pooling function** that maps a set of node embeddings to an embedding representing the full graph.
- Strategies:
  - Take the sum or mean of the node embeddings
  - Combination of LSTMs and attention to pool the node embeddings
  - Clustering or coarsening (e.g., spectral clustering)



# Variations of GNNs

- **Graph Convolutional Networks (GCN)**: A popular form of GNN where the aggregation step is similar to a convolution operation on graphs.
- **Graph Attention Networks (GAT)**: Incorporates attention mechanisms, allowing each node to weight the importance of its neighbours differently.
- **GraphSAGE**: A GNN variant that samples a fixed-size neighbourhood for each node to improve scalability for large graphs.

| Model | Best for |
|---|---|
| **GNN** | General-purpose graph tasks, node/graph classification, link prediction |
| **GCN** | Semi-supervised node classification, graph classification |
| **GAT** | Tasks where **neighbour importance varies** (e.g., node classification with varying relevance of neighbours) |
| **GraphSAGE** | **Scalable** learning on **large graphs**, node classification, link prediction |

# Generative Models

- **Variational Graph Autoencoders (VGAE)** - an extension of Variational Autoencoders (VAEs) for graphs. It uses a GNN encoder and a decoder that reconstructs the graph structure (adjacency matrix or edge probabilities) using the latent variables.

- **Graph Convolutional GAN (GraphGAN)** - a generative adversarial network (GAN) designed to generate graph-structured data. It uses a generator to create graph structures and a discriminator to differentiate between real and fake graphs.

- **Graph Recurrent Neural Networks (GRNN)** - GRNNs apply RNNs to graphs, generating node sequences or edge sequences iteratively. They can be used for generating dynamic graphs, where nodes and edges evolve over time.

# GNN Python Libraries

PyG

https://pytorch-geometric.readthedocs.io/

DGL DEEP GRAPH LIBRARY

https://www.dgl.ai/

Spektral

https://graphneural.network/

Jraph

https://github.com/google-deepmind/jraph

# GNN Python Libraries

PyG

https://pytorch-geometric.readthedocs.io/

DGL DEEP GRAPH LIBRARY

https://www.dgl.ai/

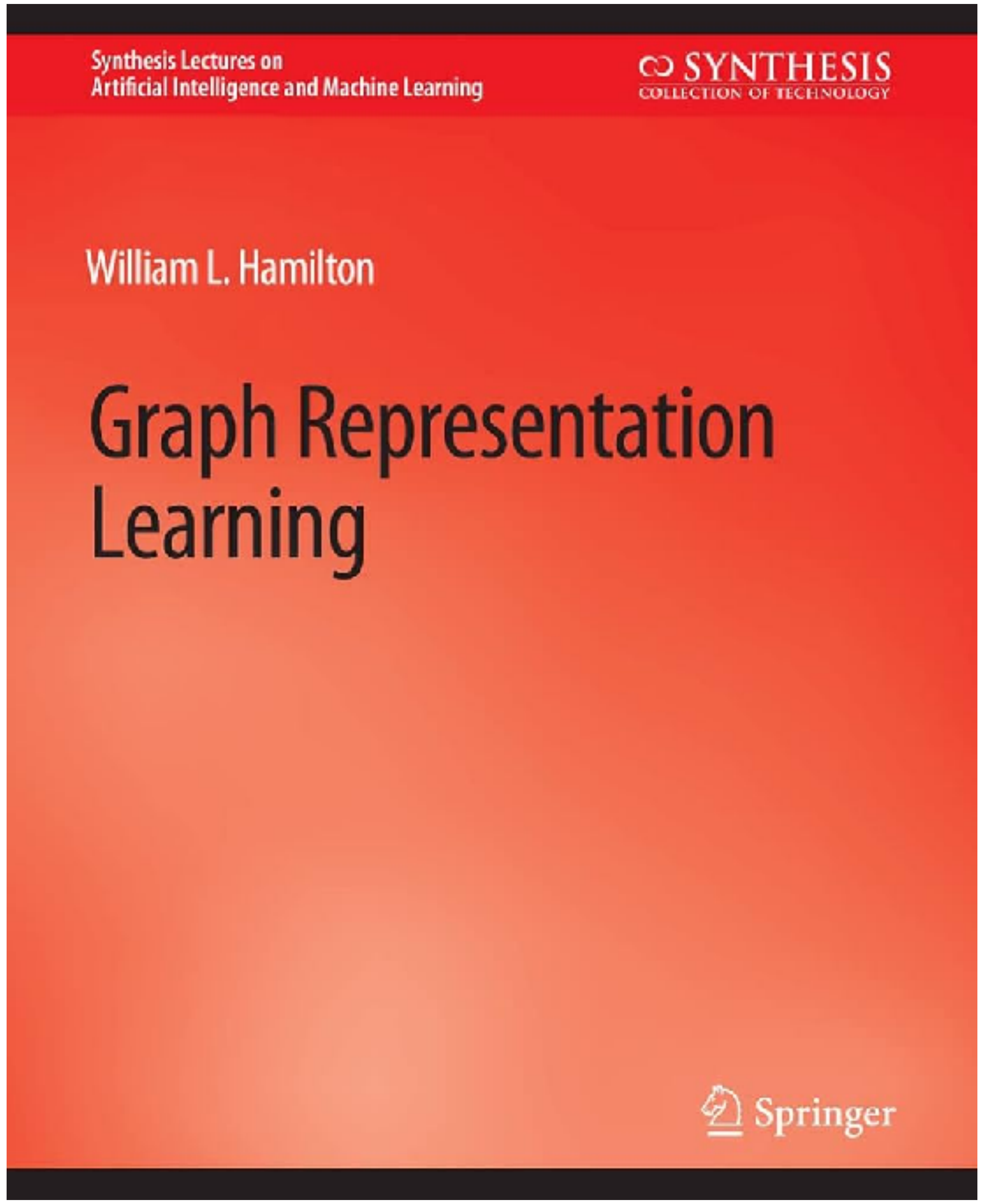
Spektral

https://graphneural.network/

Jraph

https://github.com/google-deepmind/jraph

# Acknowledgements



Synthesis Lectures on
Artificial Intelligence and Machine Learning

SYNTHESIS
COLLECTION OF TECHNOLOGY

William L. Hamilton

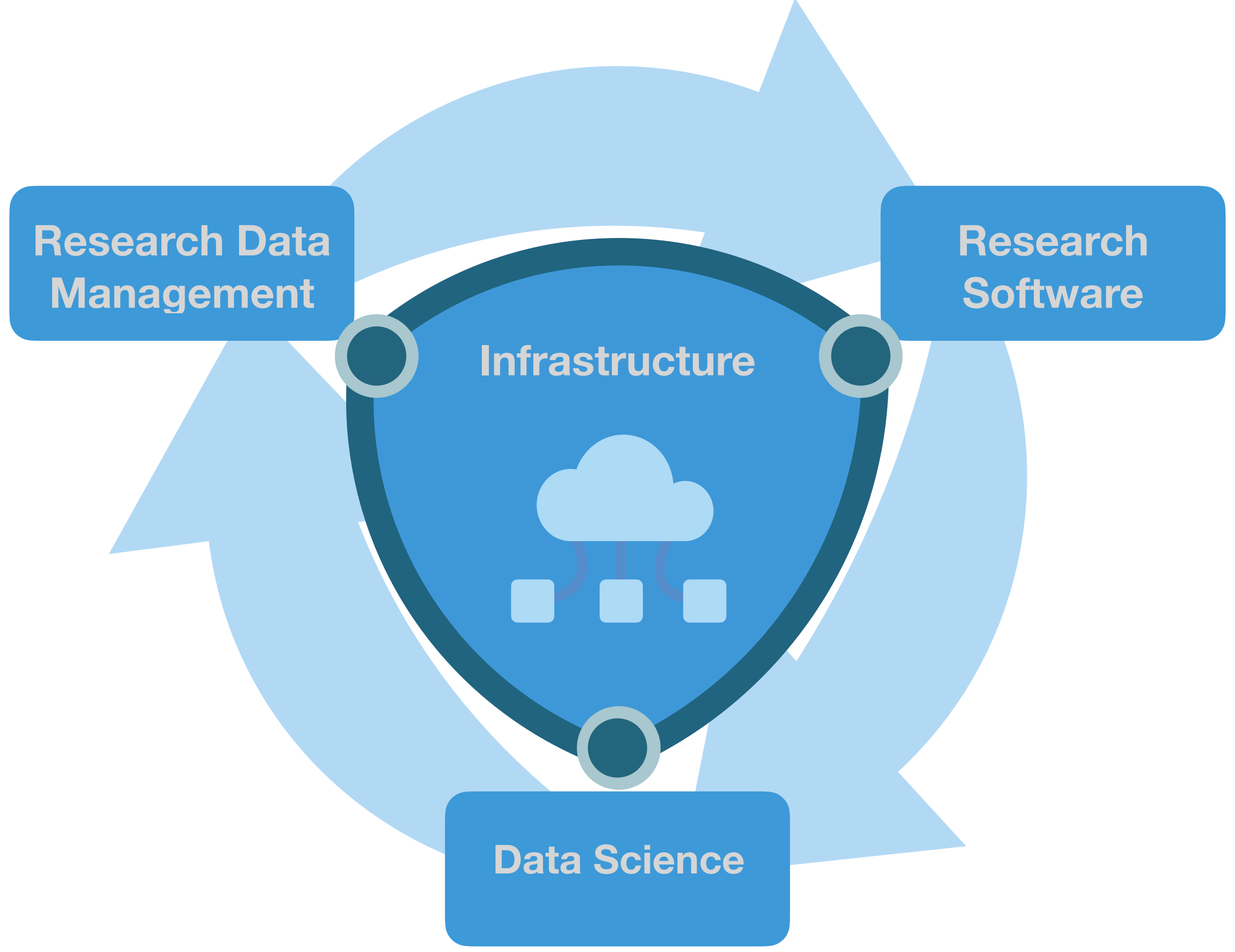
Graph Representation
Learning

Springer

# Thank you

## Multi-omics Network Analytics Research Group

## Informatics Platform



The Novo Nordisk Foundation
Center for Biosustainability

novo nordisk fonden

PIONEER CENTRE FOR ARTIFICIAL INTELLIGENCE

✉ albsad@dtu.dk

@albsantosdel

https://github.com/Multiomics-Analytics-Group

https://multiomics-analytics-group.github.io/

The Novo Nordisk Foundation
Center for Biosustainability