Learning with Graphs

A Gentle Introduction to Graph Neural Networks

Dr. Pedro CASAS

AIT Austrian Institute of Technology @Vienna

Data Science & AI

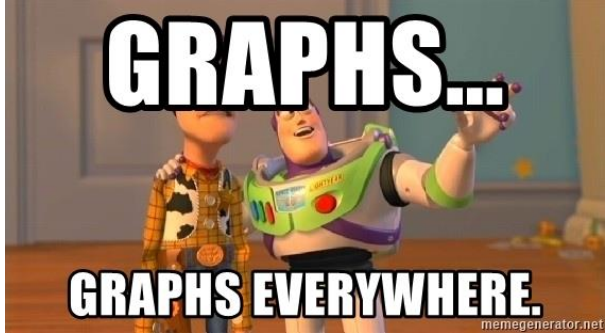
10th TMA PhD School

June 27, 2022

based on ***ML with Graphs***, Jure Leskovec, Stanford University

AI4NETS

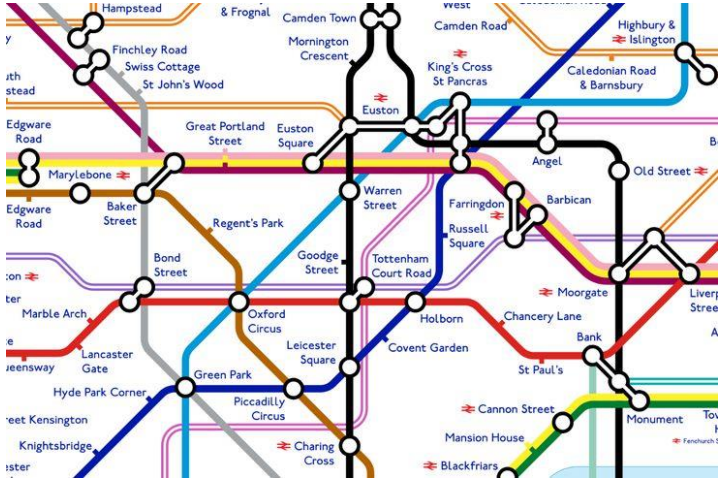
Learning with Graphs – Why Graphs?



- ***Graphs*** are all around us
- Real world ***objects are often defined*** in terms of their **connections to other things**
- A ***set of objects***, and the ***connections between them***, are naturally expressed as a ***graph***

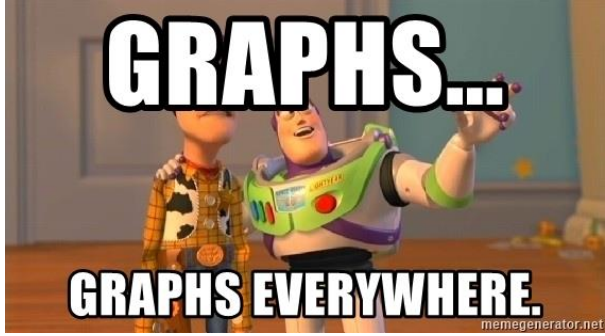
computer networks

social networks

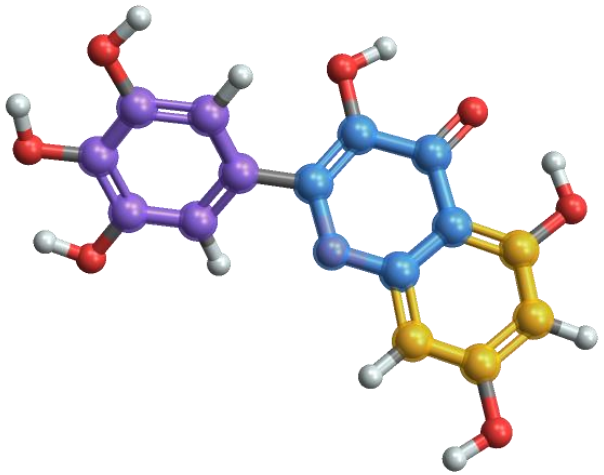


transport networks

Learning with Graphs – Why Graphs?

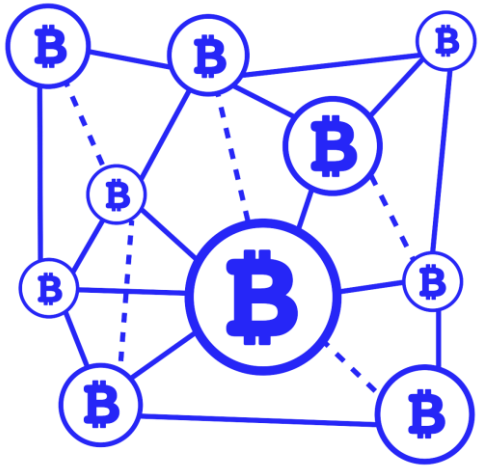


- ***Graphs*** are all around us
- Real world ***objects are often defined*** in terms of their **connections to other things**
- A ***set of objects***, and the ***connections between them***, are naturally expressed as a ***graph***
- ***Graphs*** are a ***general language*** for ***describing and analyzing entities with relations/interactions***



molecules

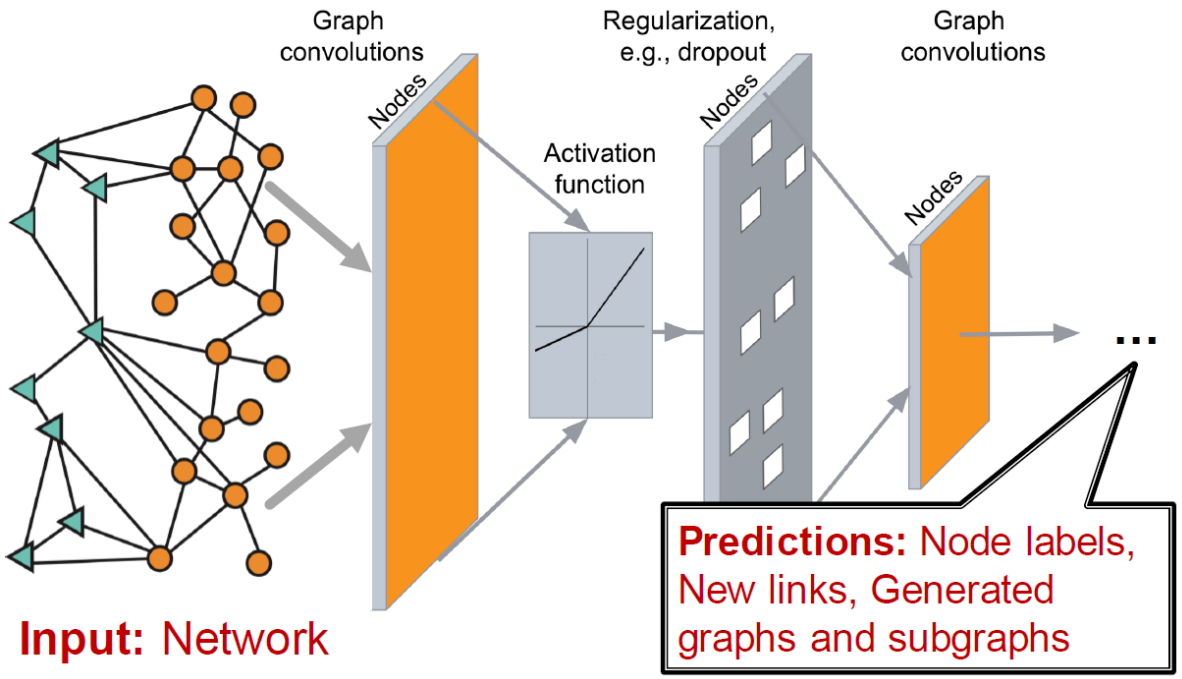
networks of neurons



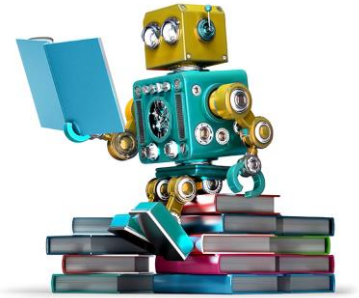
transaction graphs

Learning with Graphs – How do we Use them?

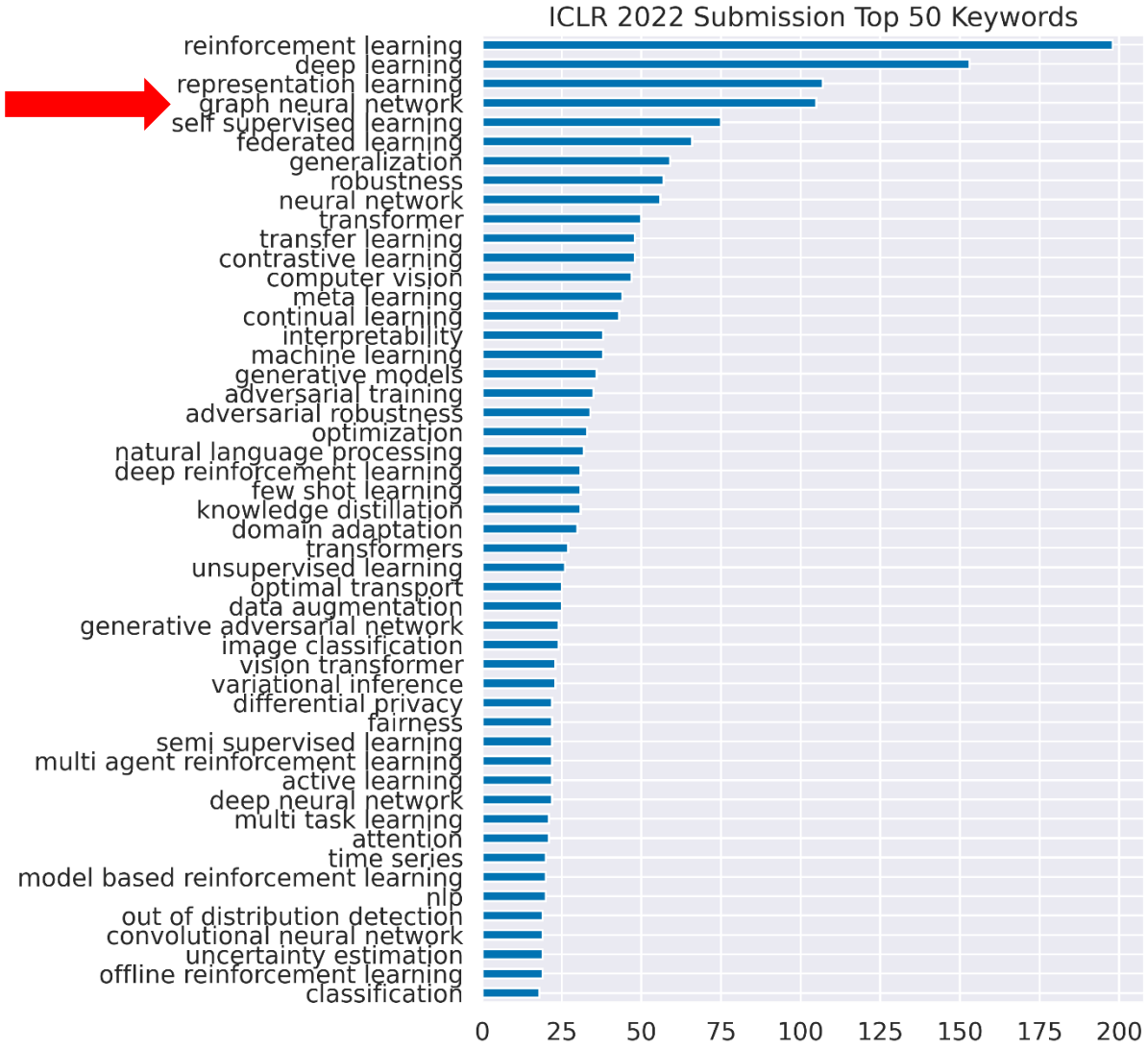
- ***Complex domains*** have a ***rich relational structure***…
- …which can be ***represented as a relational graph***
- ***Empirical results*** – by explicitly ***modeling relationships*** we achieve ***better performance***
- ***Deep Learning in Graphs*** – Graph Neural Networks (GNNs)
- Unique ability to learn and ***generalize over graph-structured data…***
- …enabled ***groundbreaking applications*** in fields where data are represented as graphs

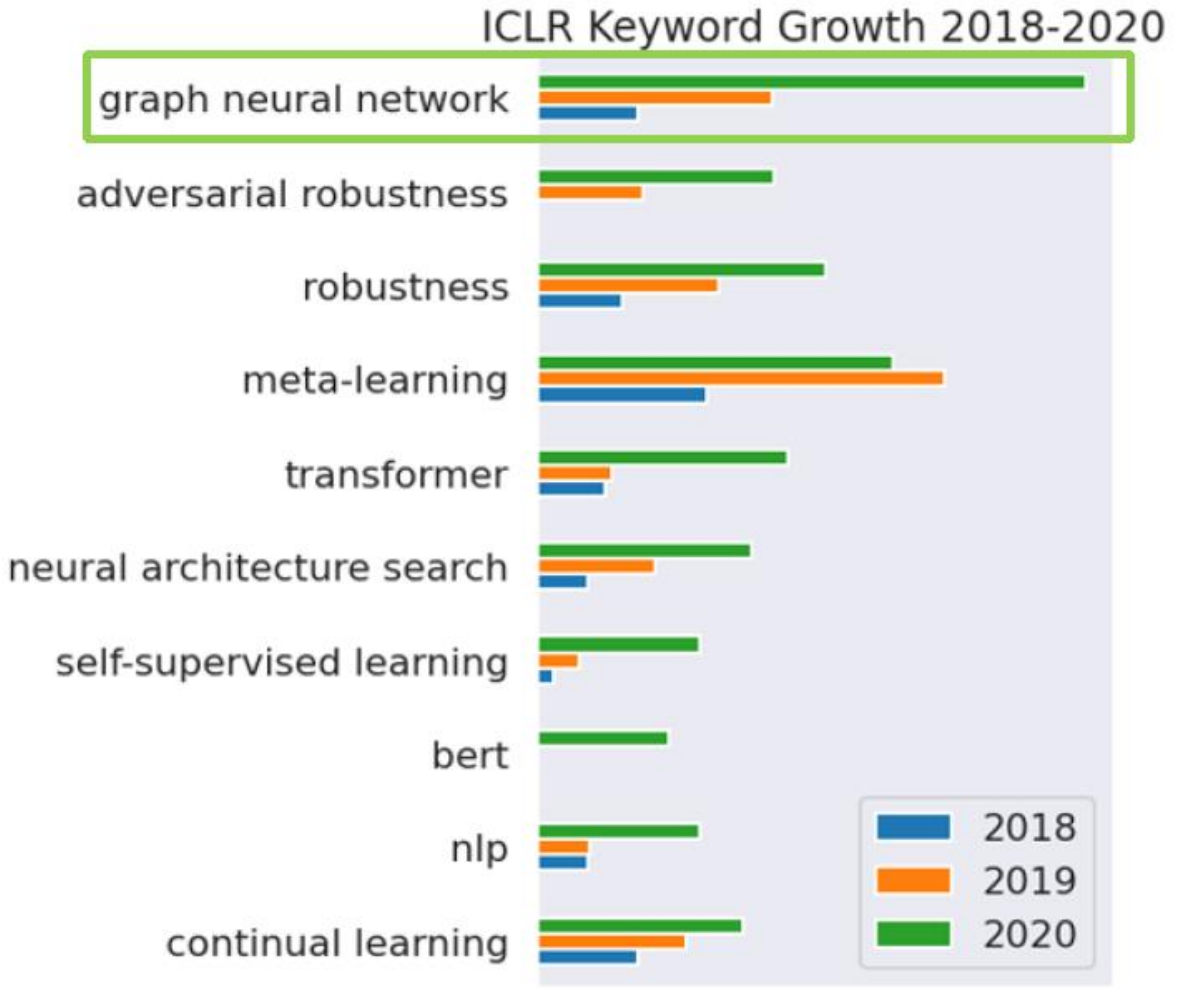


GNNs are one of the Hottest Sub-fields in ML Today

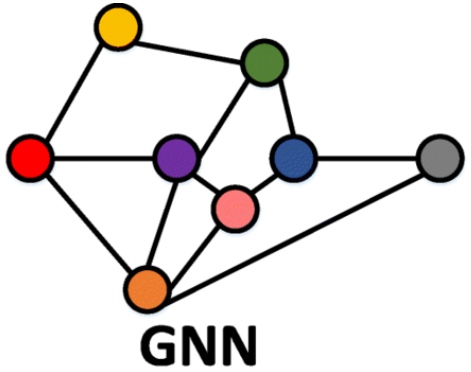


- About 4% of ICLR 2022 submitted papers using GNNs





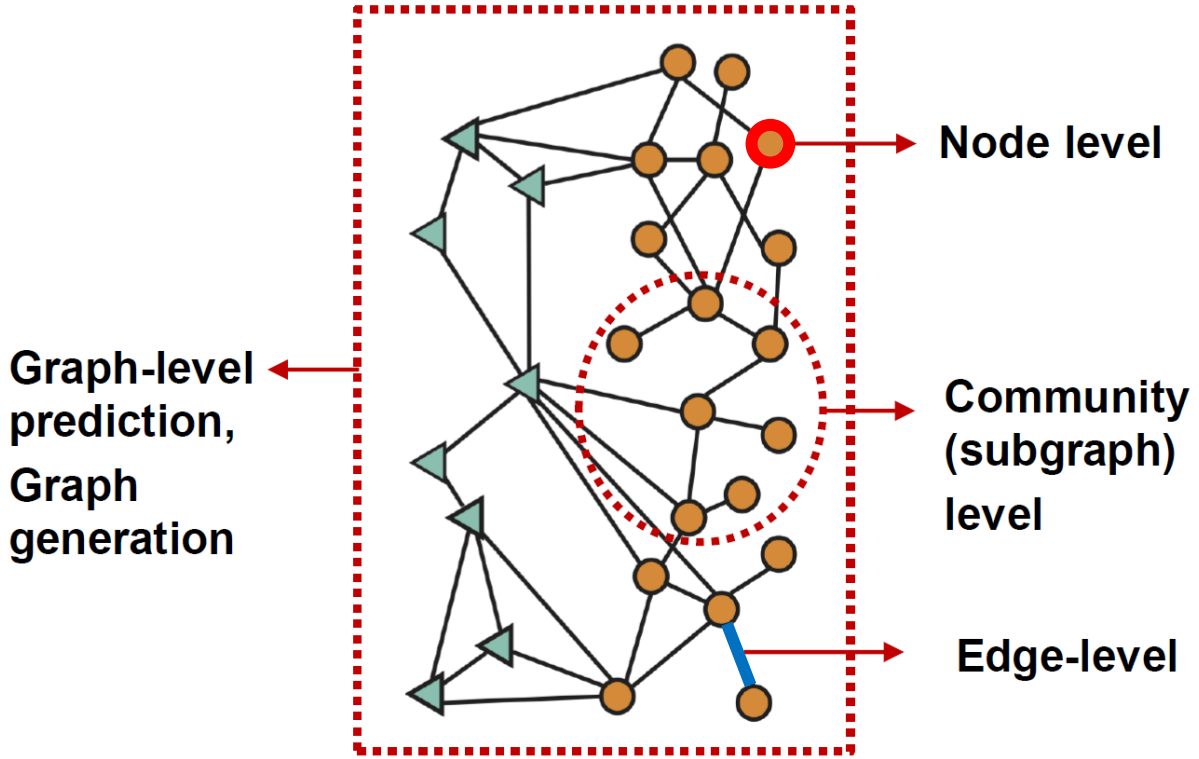
GNNs in a Nutshell – Classic Graph ML Tasks



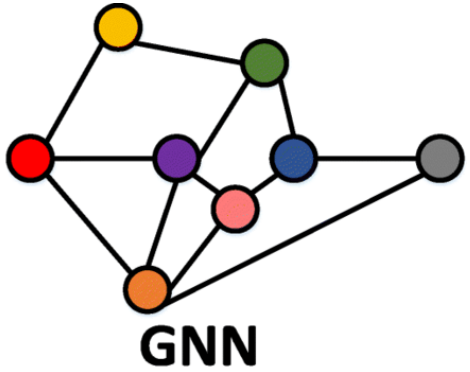
- ***Graph Neural Networks (GNNs)*** are a class of ***deep learning methods*** designed to ***perform inference on data described by graphs***

- GNNs are ***neural networks that can be directly applied to graphs*** to tackle standard types of ***tasks at the node-level***, ***edge-level***, and ***graph-level***

- We define a graph ***G(V,E)***
  - ***V*** is the set of ***n nodes*** or ***vertices***
  - ***E*** is the set of ***links*** or ***edges***
- Adjacency matrix ***A*** with dimensions ***(n x n)***
- Matrix of node features $\boldsymbol{X} \in \mathbb{R}^{(n \times m)}$
- ***Many types of graphs:*** directed, undirected, bipartite, weighted, heterogeneous, etc.

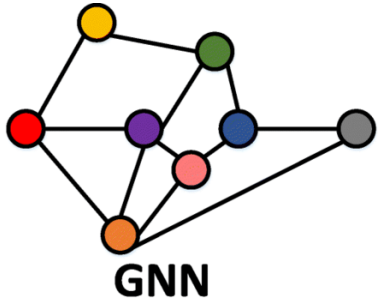


GNNs in a Nutshell – Examples of Tasks on Graphs



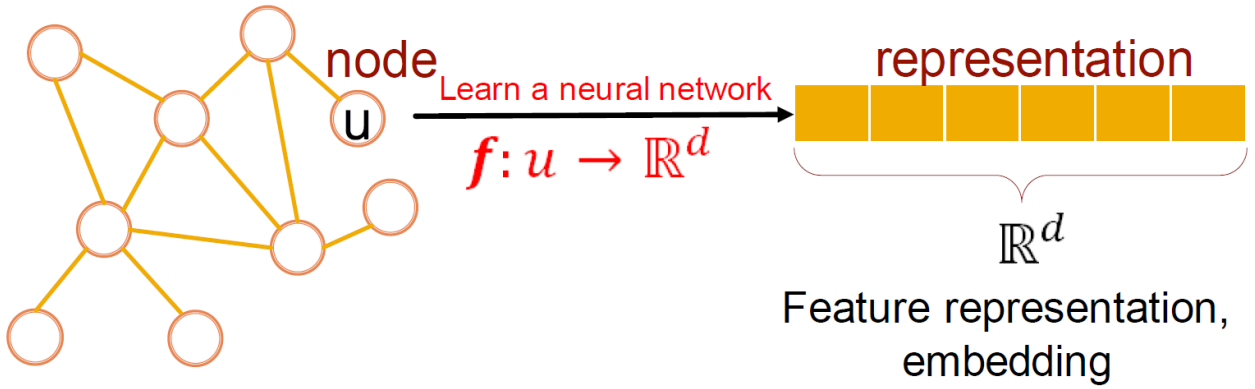
- ***Node Classification***
  - ***predict a type (label) of a given node***, by looking at the labels of the neighbors
  - usually trained in a semi-supervised way, with only a part of the graph being labeled
  - E.g., predict amino–acid sequences (e.g., ***DeepMind's AlphaFold***)
- ***Link Prediction***
  - understand the relationship between nodes in graphs
  - ***predict whether there is a connection between two nodes***
  - E.g., infer social interactions in social networks, recommendation systems
- ***Graph Classification and Prediction***
  - ***classify the complete graph into different categories***
  - similar problems to image classification
  - E.g., molecule property prediction (protein is an enzyme or not), social analysis, Travel Time Estimation (e.g., ***Google Maps***)
- ***Graph Clustering***
  - detect if nodes form a certain community
  - vertex clustering, graph clustering (similarity between graphs)
  - E.g., identification of communities, anomaly detection

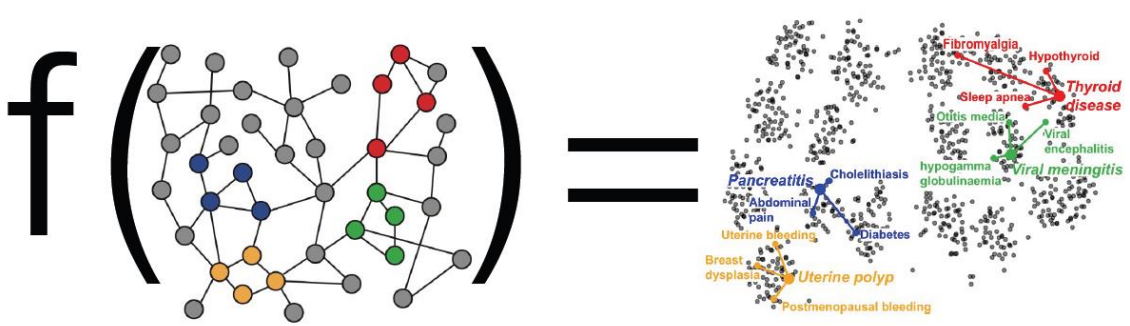
GNNs in a Nutshell – Node Embeddings



- How do we ***learn on graphs?*** → ***(deep) embeddings***

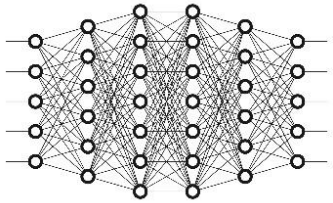
- The notion of ***node embeddings:*** map nodes to *d*-dimensional embeddings such that similar nodes in the network are embedded close together



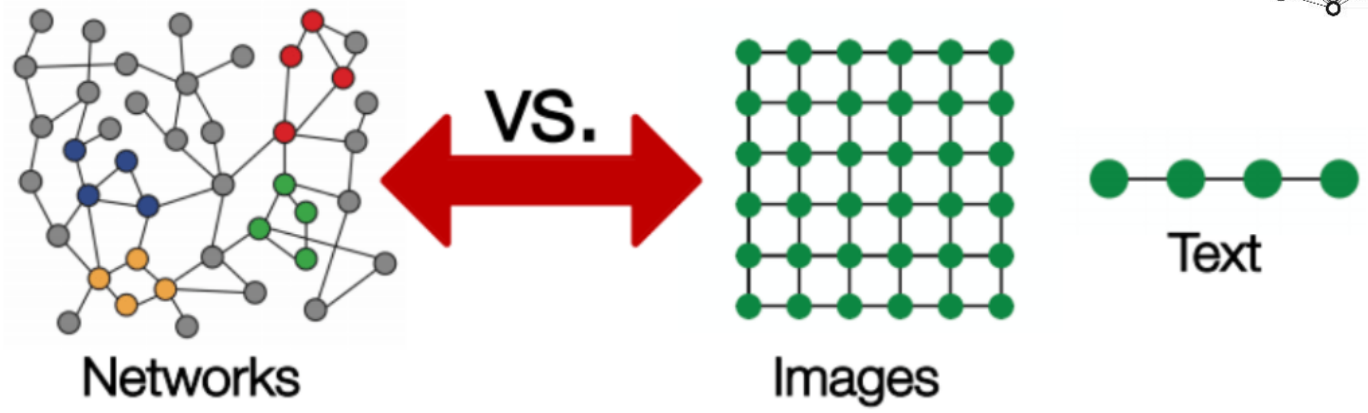


- The exercise is therefore how to ***learn*** these ***mapping functions f ?*** Embeddings should keep the structure of the graph, and incorporate nodes' neighboring properties

Why not Traditional *Deep Learning?*

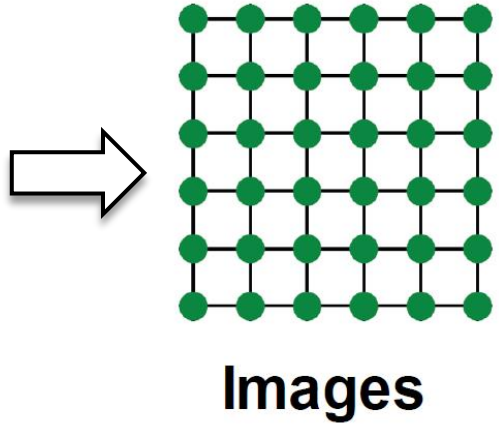


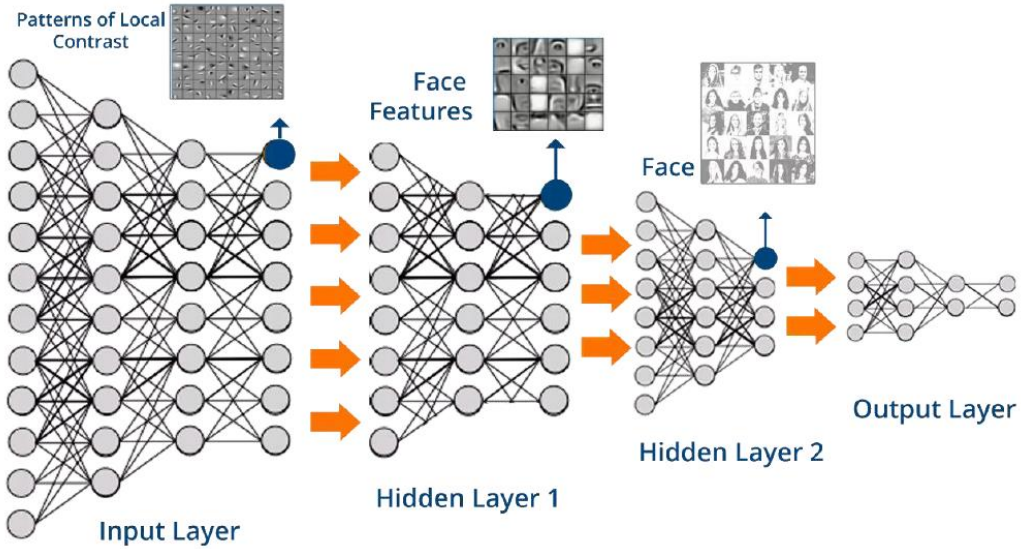
- ***Deep Learning*** is ***designed*** to specific, structured, ***simple types of graphs***: ***grids*** and ***sequences***

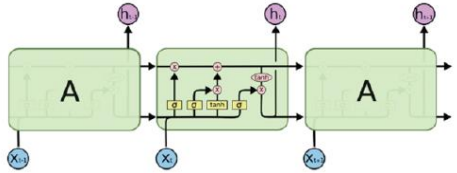


- Graphs have ***no spatial locality as grids***

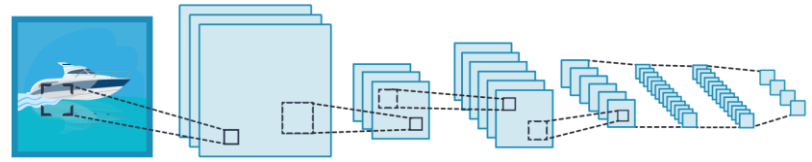
- ***No fixed node ordering*** as sequences



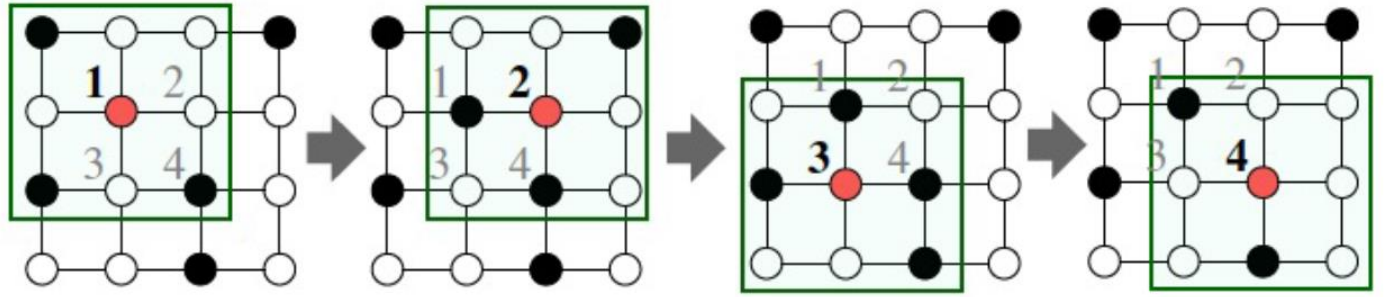


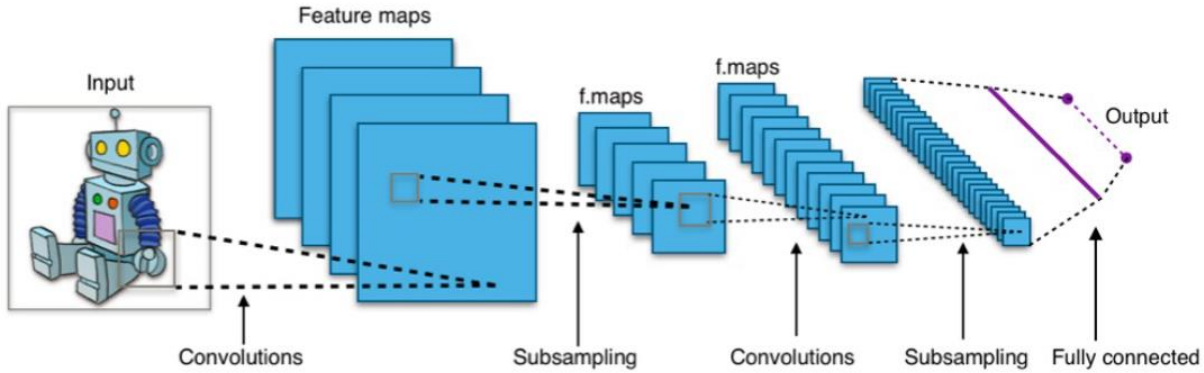


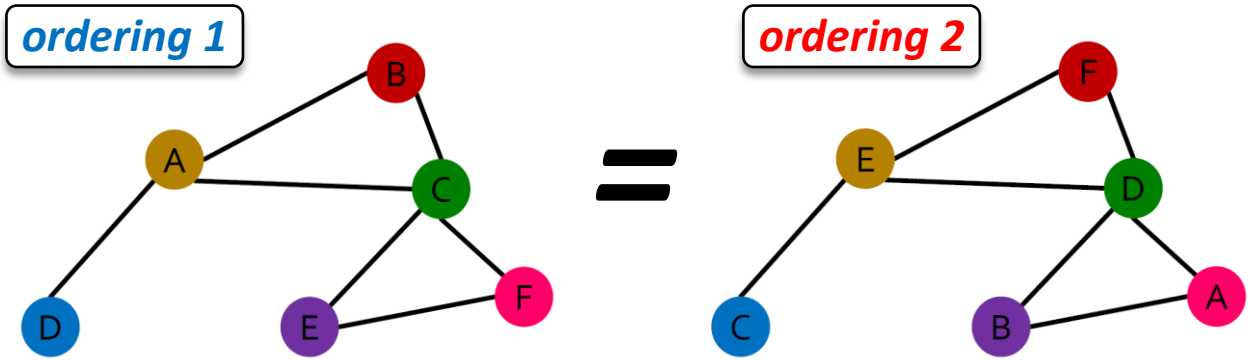
Generalizing Convolutions to Graphs



- ***CNN on images:*** convolution takes a little sub-patch of the image around a pixel (***node***) and ***aggregates information*** from its ***neighbors*** and ***itself***
- The goal is to ***generalize convolutions beyond simple lattices…***
- …but as we said, there is ***no fixed notion of locality*** or sliding window ***on the graph***
- And ***graphs are permutation invariant!***

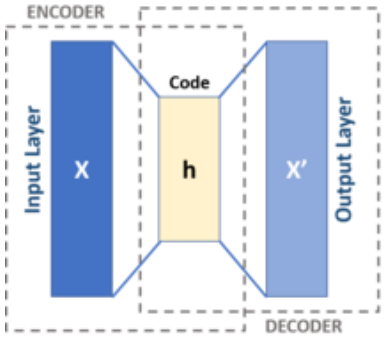




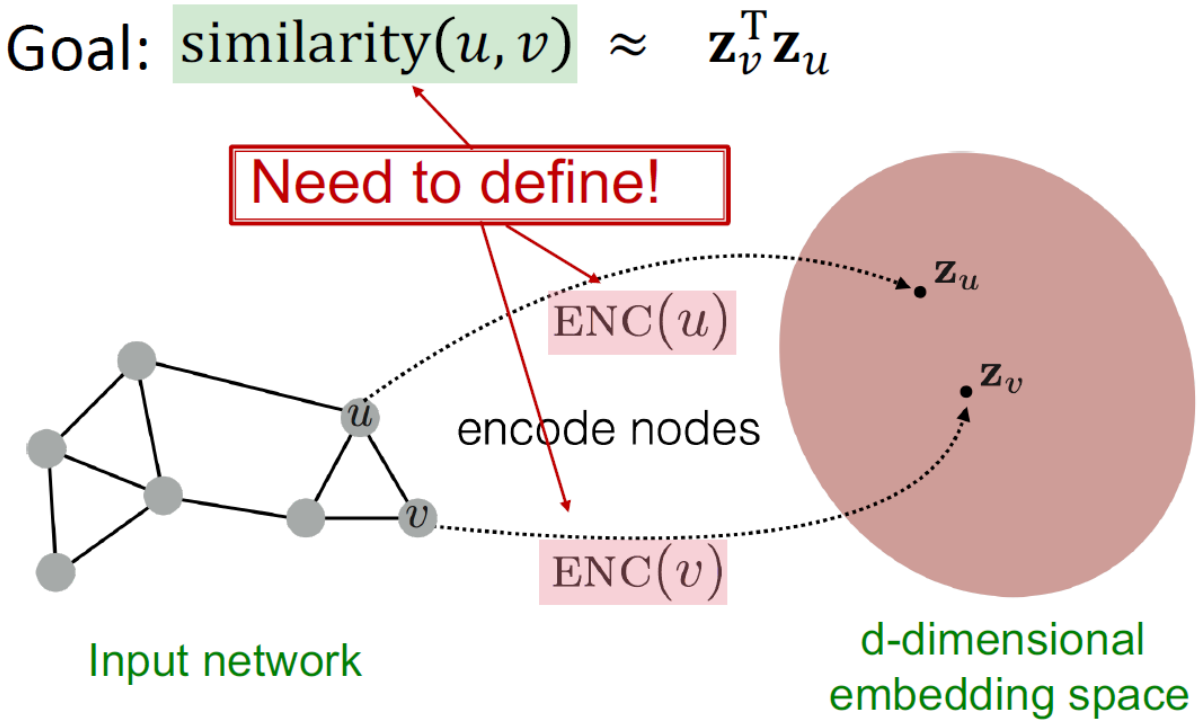


- Graphs ***do not have a canonical order*** of the nodes
- Graph and node ***representations*** should be the ***same*** for ***ordering 1*** and ***ordering 2***

Building the Embeddings – the Encoding Function

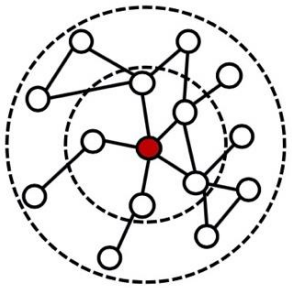


- ***Recap:*** GNNs basically consists in ***encoding the graph*** in the form of vectors and then using this encoding ***to make predictions***
- ***Encoder:*** take a graph and ***learn an embedding for every node of the graph***
- ***Decoder:*** use the learned embeddings and make predictions
- ***Training:*** feed embeddings into any loss function and run stochastic gradient descent to train weights



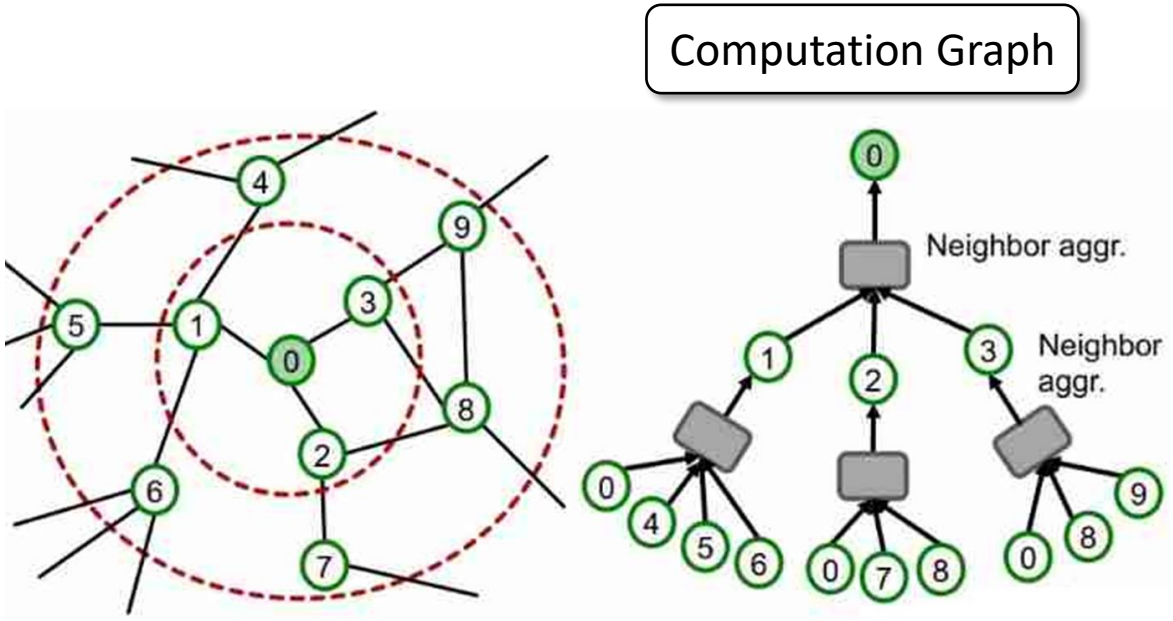
- let ***u*** and ***v*** be two nodes on the graph
- x_u and x_v their corresponding node feature vectors
- the ***encoding function*** **ENC(*u*)** y **ENC(*v*)** convert the feature vectors to z_u and z_v in the embedding space
- the ***decoding function*** is simply the similarity between nodes: ***similarity***$(u, v) \approx z^T_v \cdot z_u$
- ***challenge*** → come up with ***the encoder function***

Graph Convolutional Networks (GCNs)

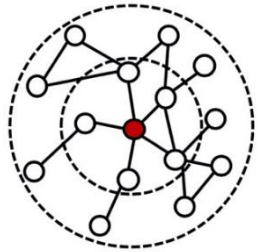


- For simplicity, let us consider a ***simple type of GNN – Graph Convolutional Network (GCN)***
- We are looking for an encoder function which should be capable of:
  - ***Integrating locality information*** (local graph neighborhoods)
  - ***Aggregating information***
  - ***Stacking multiple layers*** (computation) – deep graph encoders

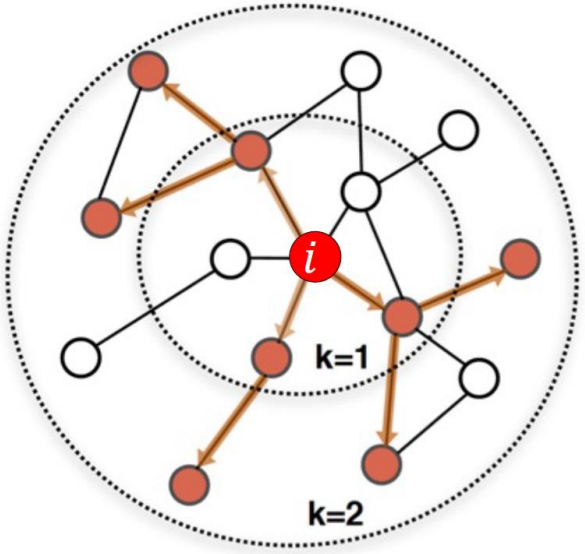
- ***Locality information*** → the ***neighborhood of a node*** defines ***a computation graph*** (directed graph where nodes correspond to mathematical operations – acts as a ***functional description of a computation***)
- ***Key idea*** → generate ***node embeddings*** based on local network neighborhoods.



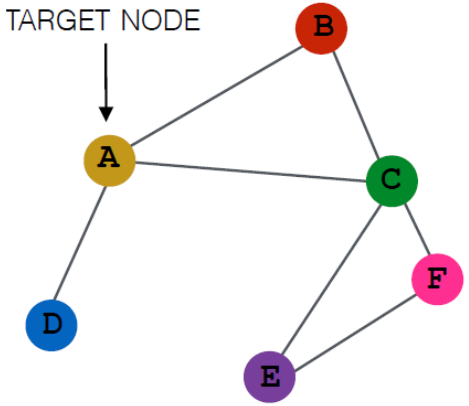
Graph Convolutional **Networks** *(GCNs)*



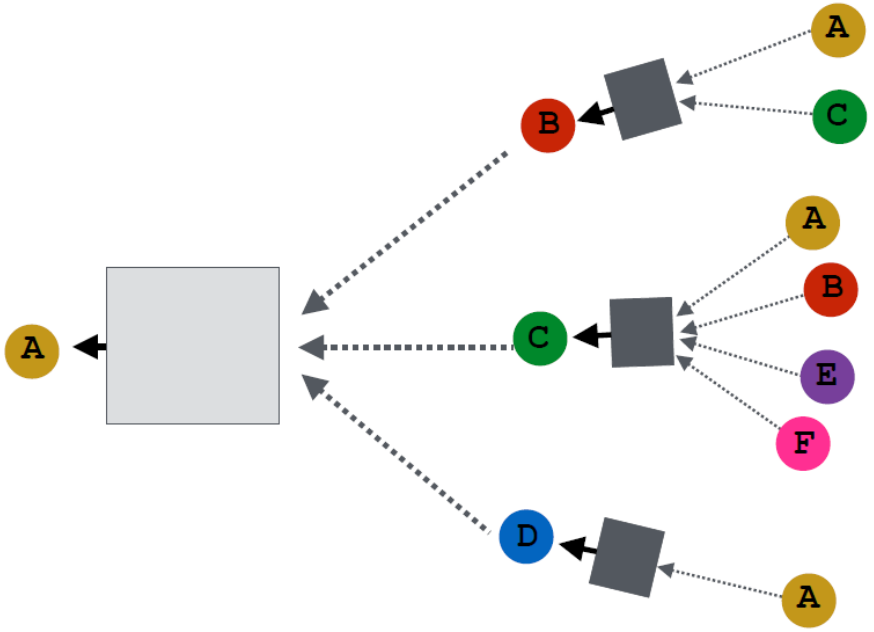
- ***Key idea*** → generate node embeddings based on local network neighborhoods (***k*** hops), using ***computation graphs***



k *hops for nearest neighbors*



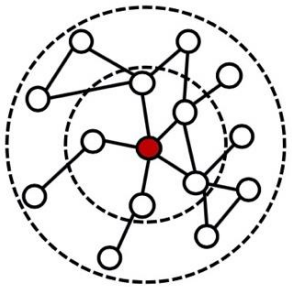
input graph embedding for ***node A***



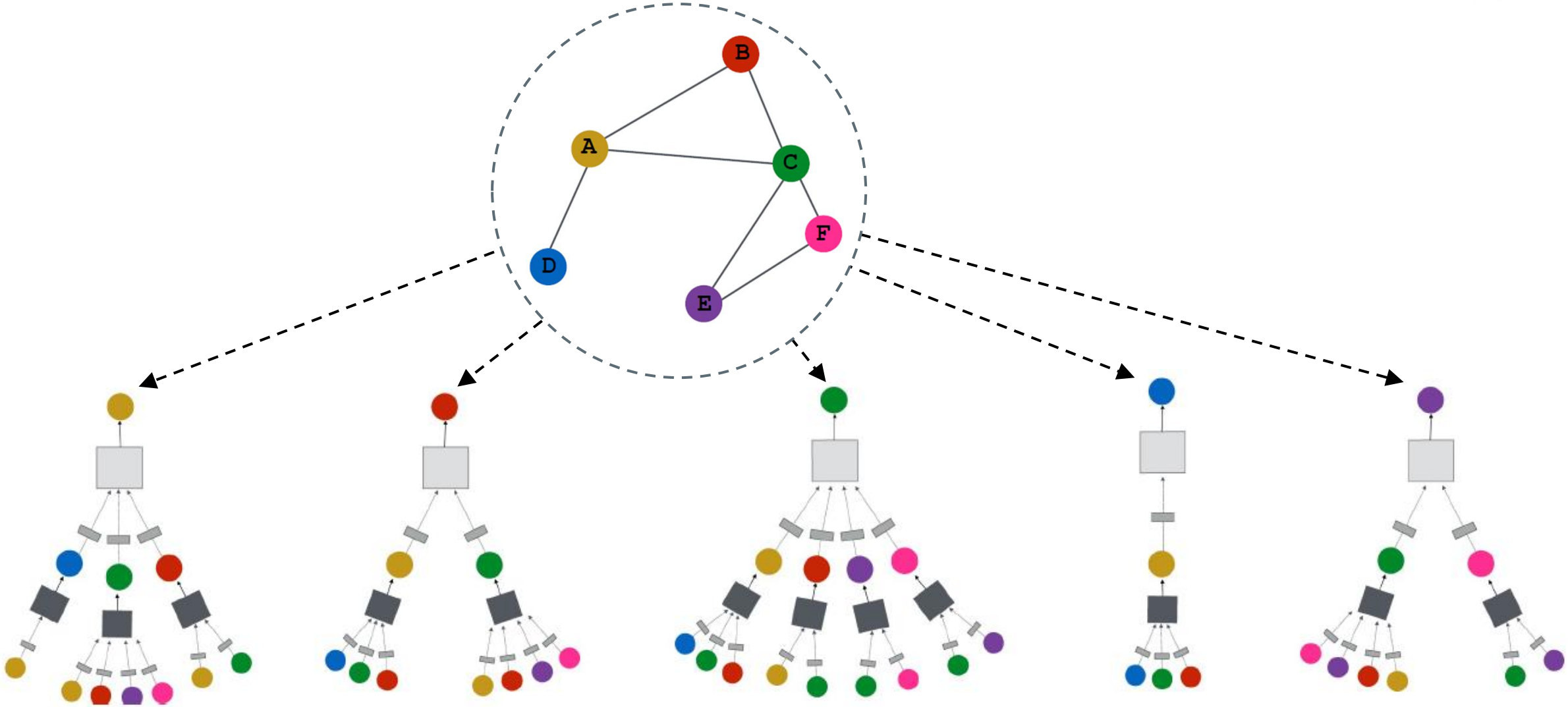
computation graph for ***node A***

- Now that we have the ***local information coded in a graph structure***, we do ***data aggregation***

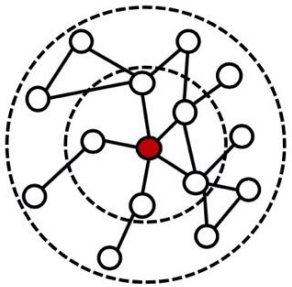
Graph Convolutional Networks (GCNs)



- ***Every node*** defines a ***computation graph*** based on its neighborhood



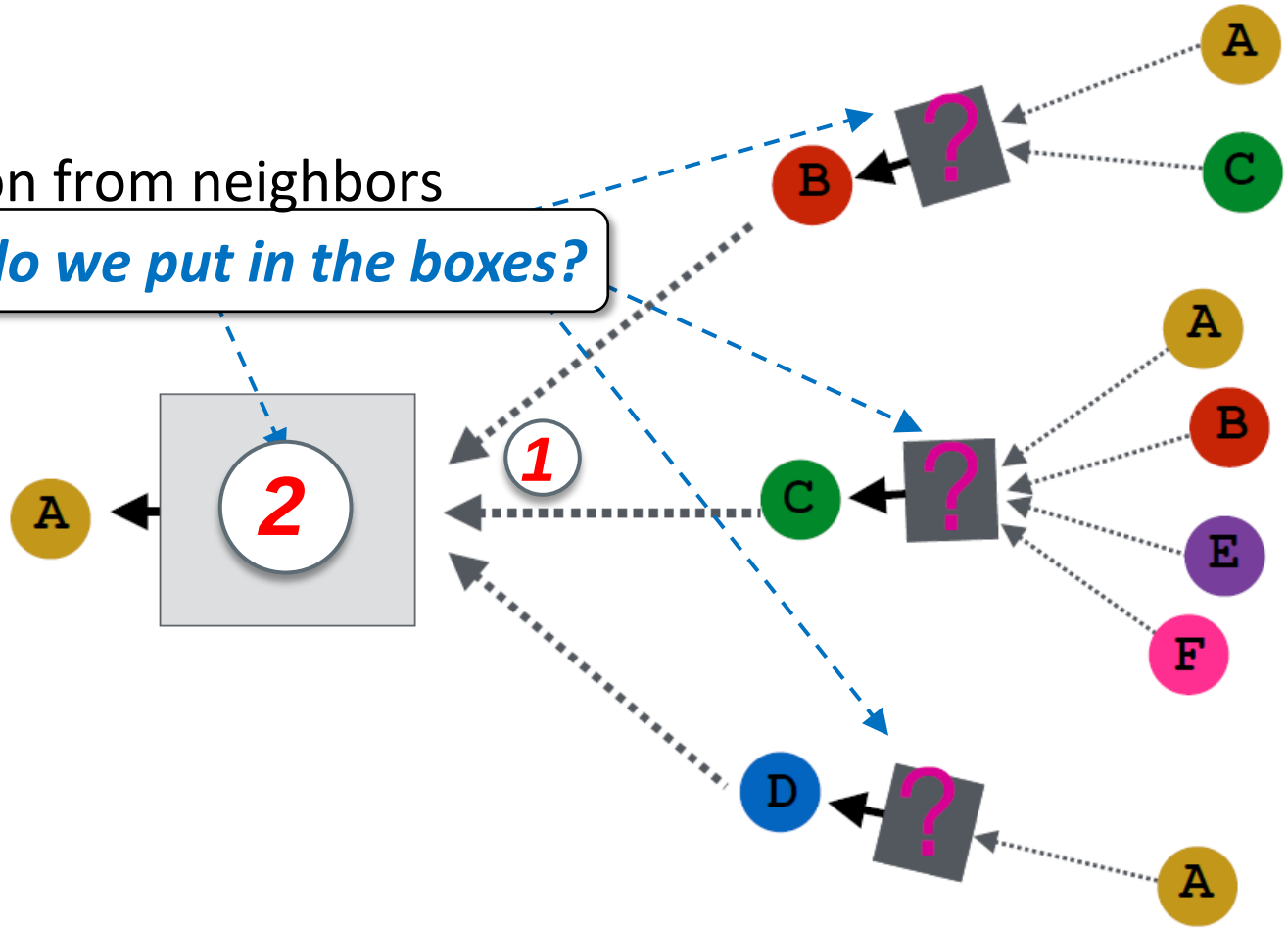
Neighborhood Aggregation with Deep Encoders



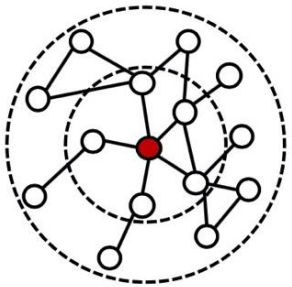
- Nodes aggregate information from their neighbors and apply *neural networks*

(1) Basic aggregation: average information from neighbors

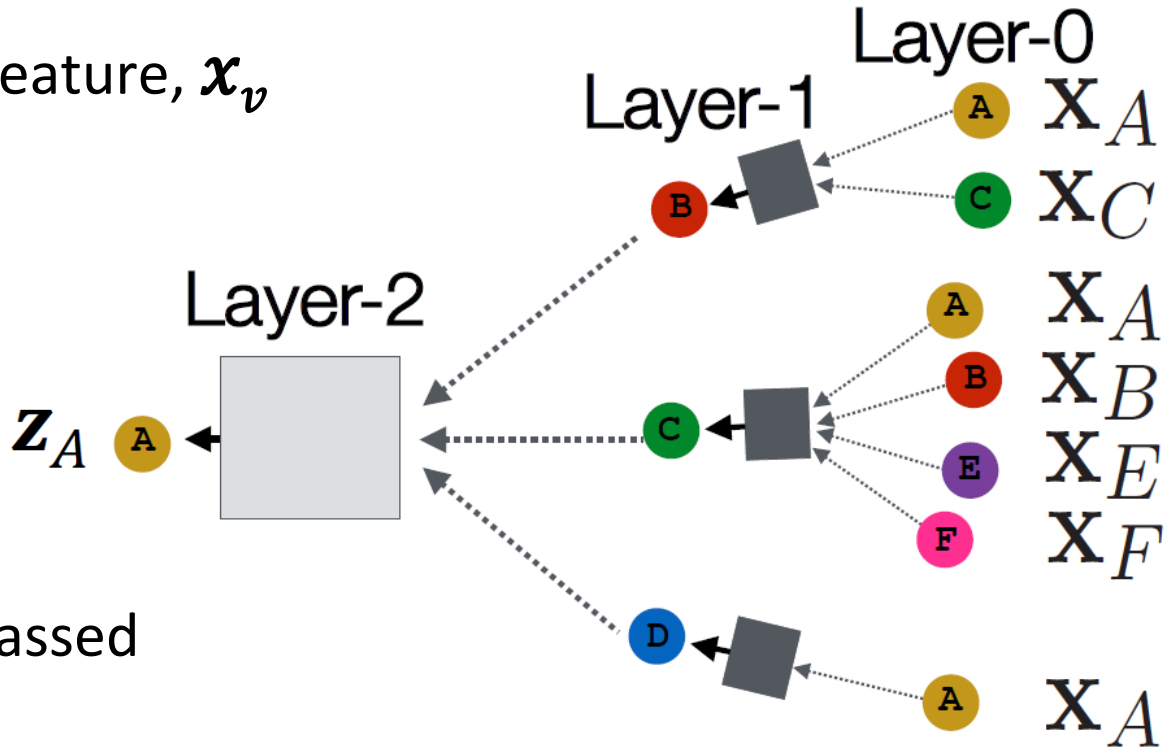
(2) Apply neural network



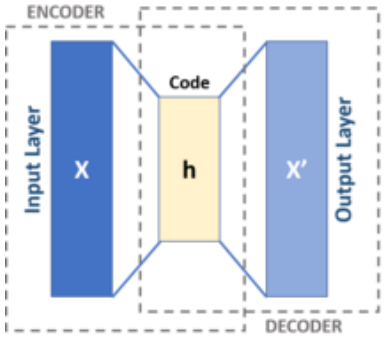
Deep Encoders – Many Layers



- Model can be of ***arbitrary depth*** – depends on the k hops for NNs
- Nodes (v) have ***embeddings at each layer k*** → $\mathrm{h}_v^{(k)}$
  - Layer – 0 embedding of node v is its input feature, x_v
  - Layer – k embedding gets information from nodes that are k hops away
- Example:
  - x_A and x_C are the inputs at Layer-0
  - Both feature vectors are aggregated and passed through an activation function in Layer-1
  - And then passed to the next Layer-2



Formulation of the Deep Encoder



$$\mathbf{h}_v^0 = \mathbf{x}_v$$

initial layer 0 embeddings are equal to node features

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right), \quad \forall k > 0$$

k^{th} layer embedding of v

non-linearity

average of neighbor's previous layer embeddings

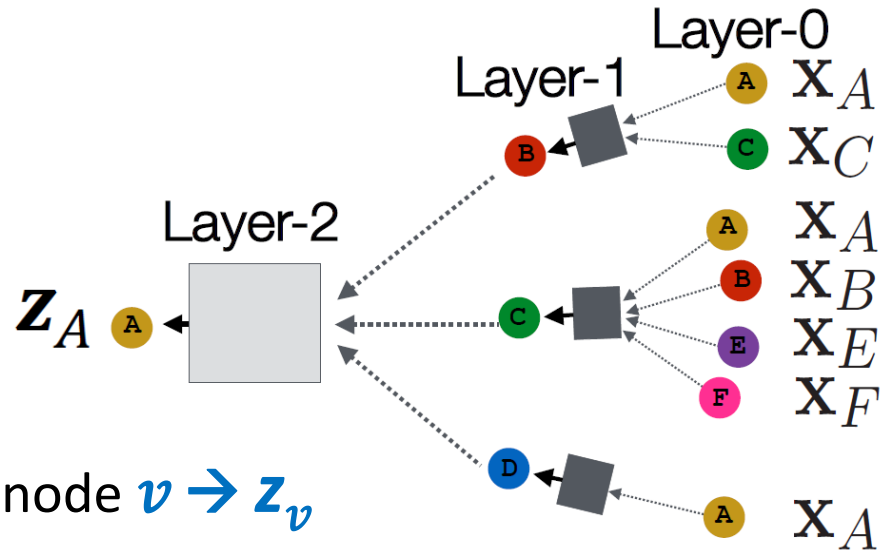
previous layer embedding of v

$$k \in \{1, .., K\}$$

total number of layers

$$z_v = \mathbf{h}_v^{last}$$

- Initial Layer-0 embeddings are equal to node features
- ***N(v)*** is the set of neighboring nodes (embeddings)
- Non-linearity (***activation function***) σ → e.g., ReLU
- Matrices W_k and B_k are the ***trainable weights***
- After ***K*** layers of aggregation, we obtain the embedding for node $v \rightarrow z_v$



Training the Model to Generate Embeddings

- To train the model (parameters), we need to define a ***loss function on the embeddings***
- We can feed the embeddings into any loss function and run *SGD* to train the weights
- Training can be ***supervised*** or ***unsupervised***
- ***Supervised:*** train model for supervised task, e.g., node classification, using ***node labels y***

$$\min_{\Theta} \mathcal{L}(\boldsymbol{y}, f(\boldsymbol{z}_v))$$

$\boldsymbol{z}_v$: node embeddings

e.g., ***L2 norm*** if ***y*** are real numbers, ***Cross Entropy (CE)*** for categorical ***y***

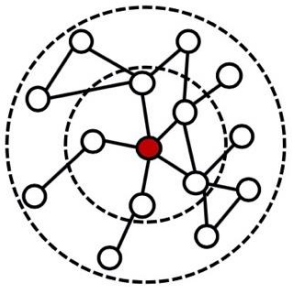
- ***Unsupervised*** → use graph structure as supervision: similar nodes have similar embeddings
  - Unsupervised loss function can be a loss based on node proximity in the graph

$$\mathcal{L} = \sum_{z_u, z_v} \mathrm{CE}(y_{u,v}, \mathrm{DEC}(z_u, z_v))$$

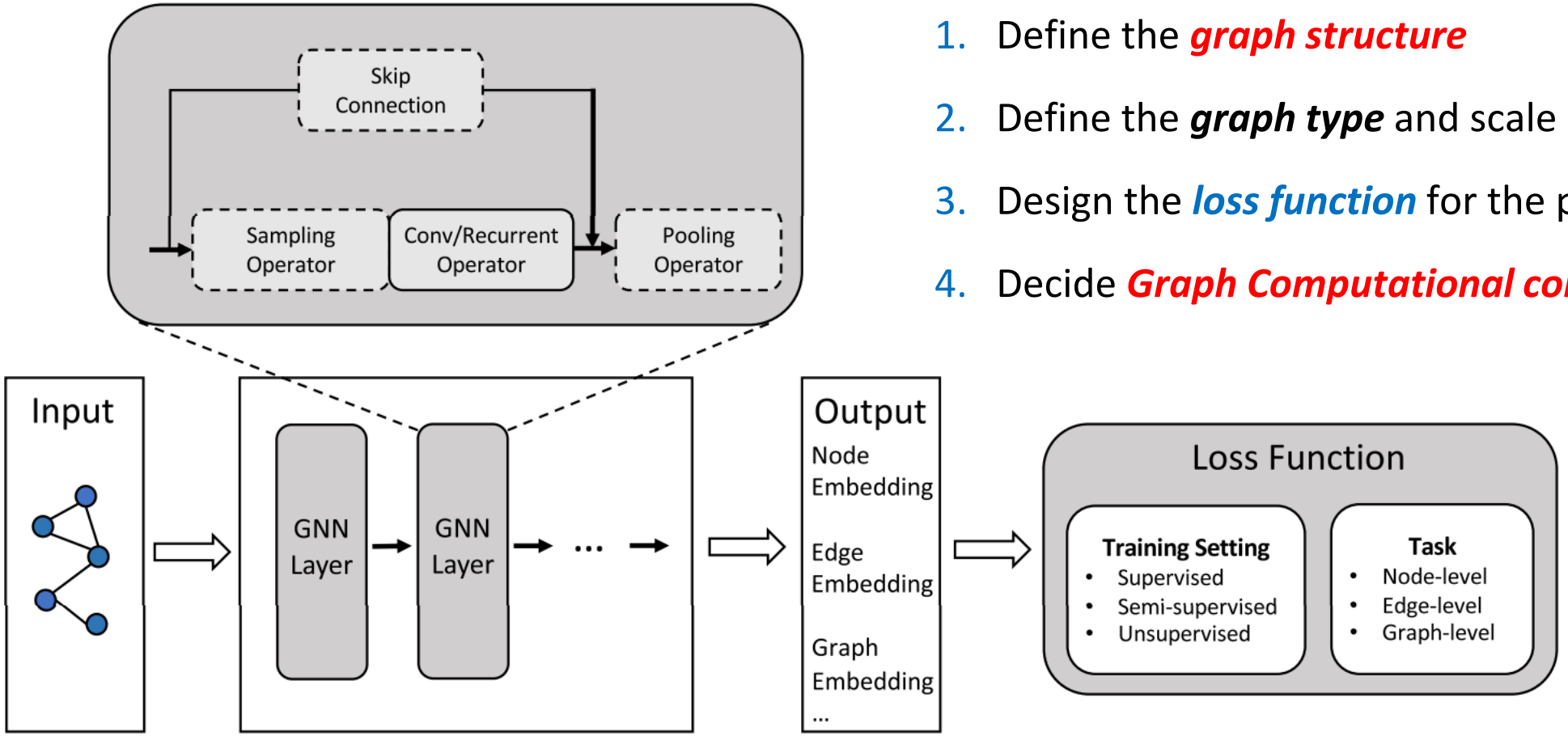
DEC: decoder (e.g., inner product)

$y_{u,v}$: node similarity function

GNN Design – Just the Tip of the Iceberg



- Many decisions and options to chose from when designing a GNN



1. Define the ***graph structure***
2. Define the ***graph type*** and scale
3. Design the ***loss function*** for the planned task
4. Decide ***Graph Computational components***

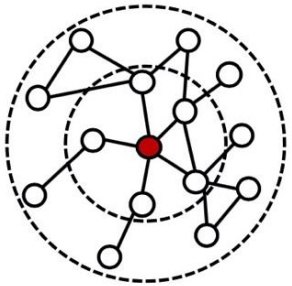
1. Find graph structure.

2. Specify graph type and scale.

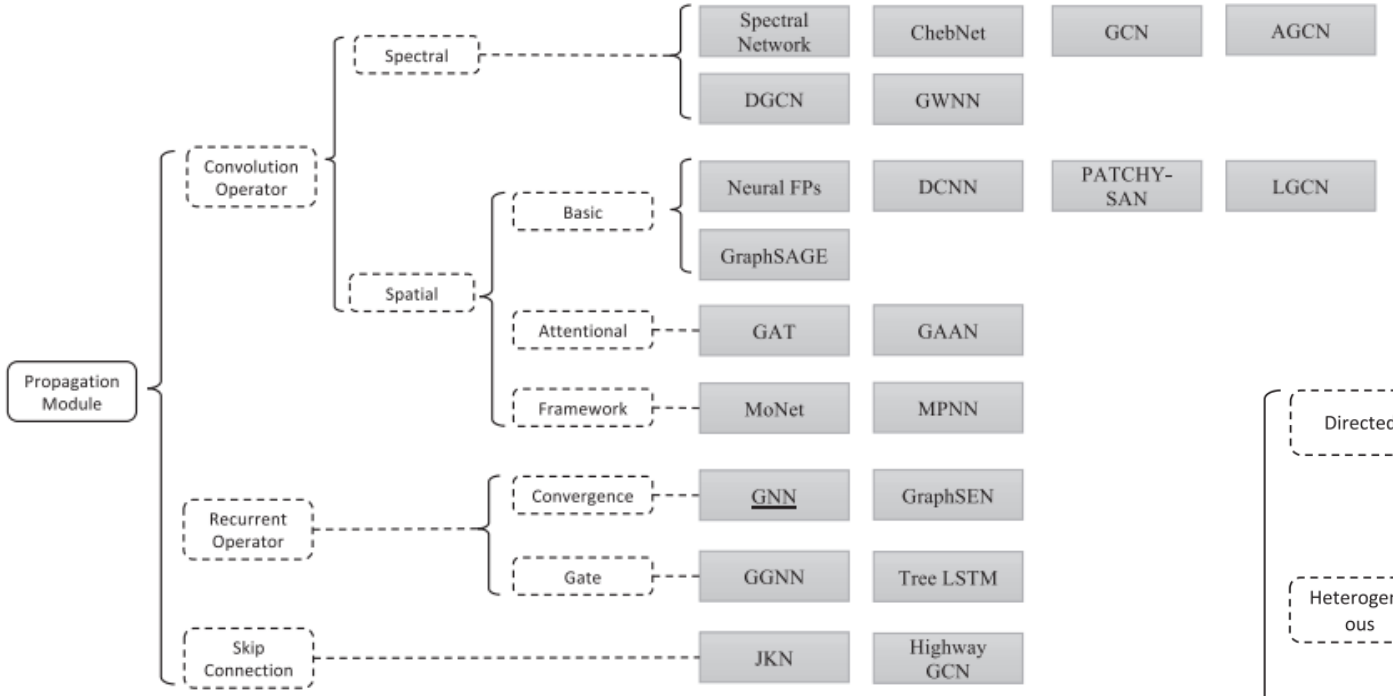
4. Build model using computational modules.

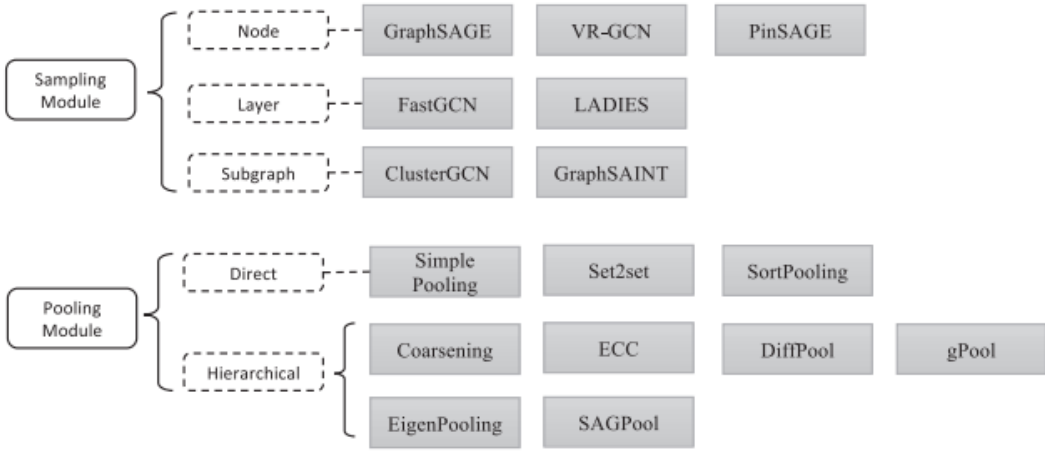
3. Design loss function.

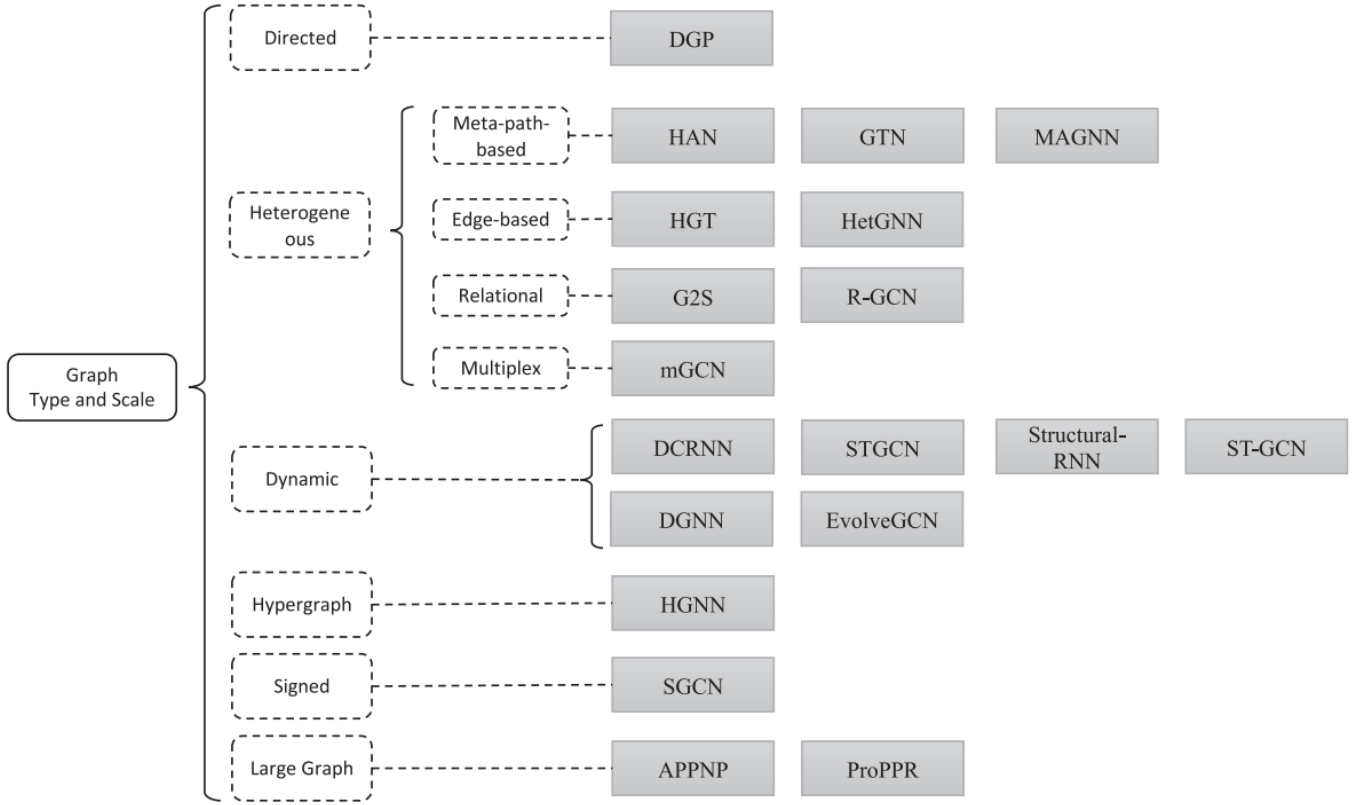
GNN Design – Computational Components



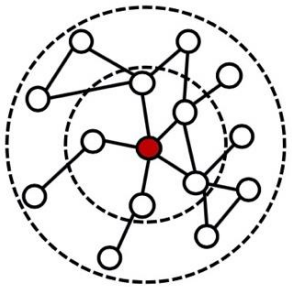
- ***Propagation modules***
- ***Sampling modules***
- ***Pooling modules***
- Variations depending on type of graph



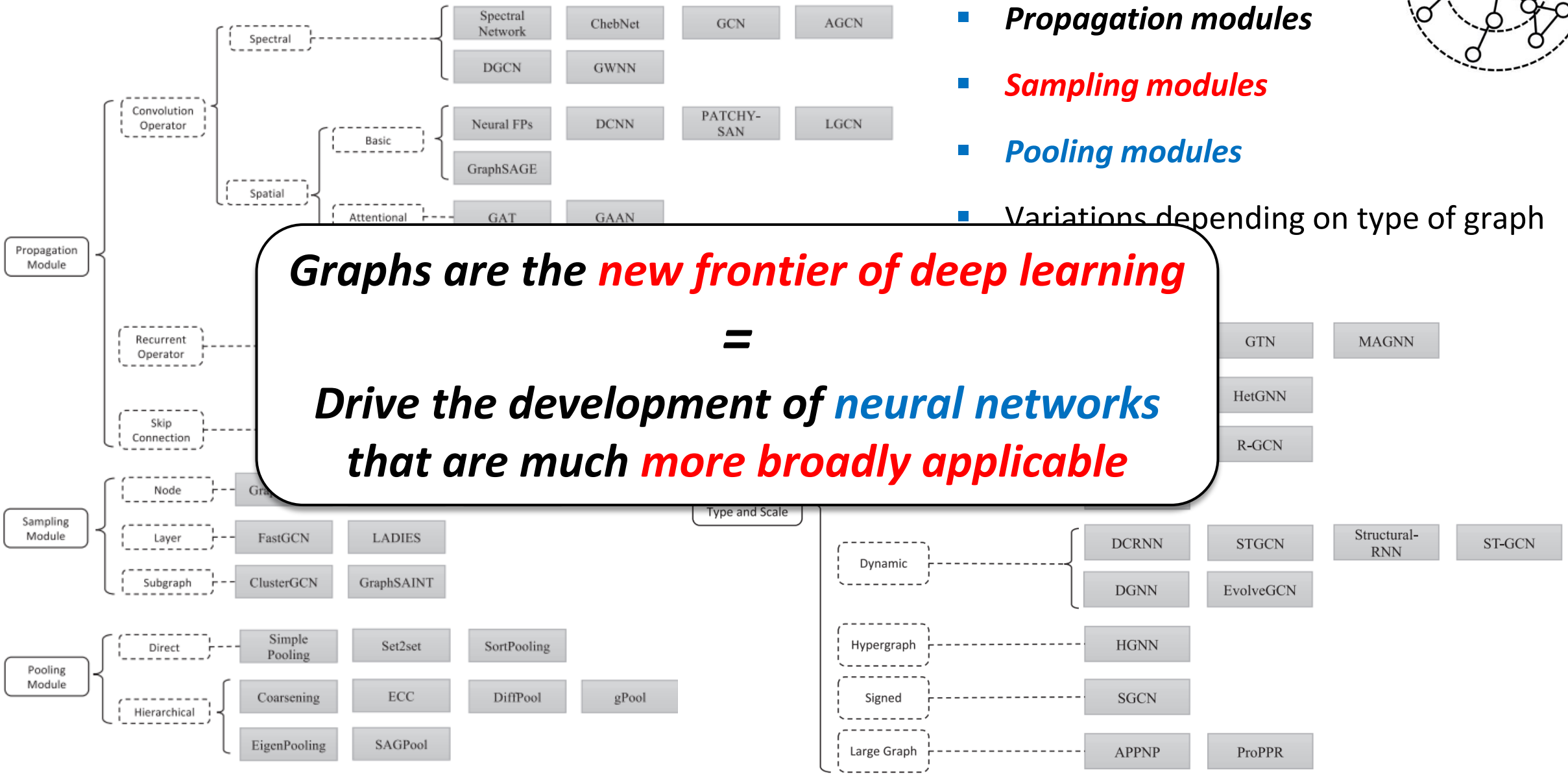




GNN Design – Computational Components



- ***Propagation modules***
- ***Sampling modules***
- ***Pooling modules***
- Variations depending on type of graph



Graphs are the new frontier of deep learning

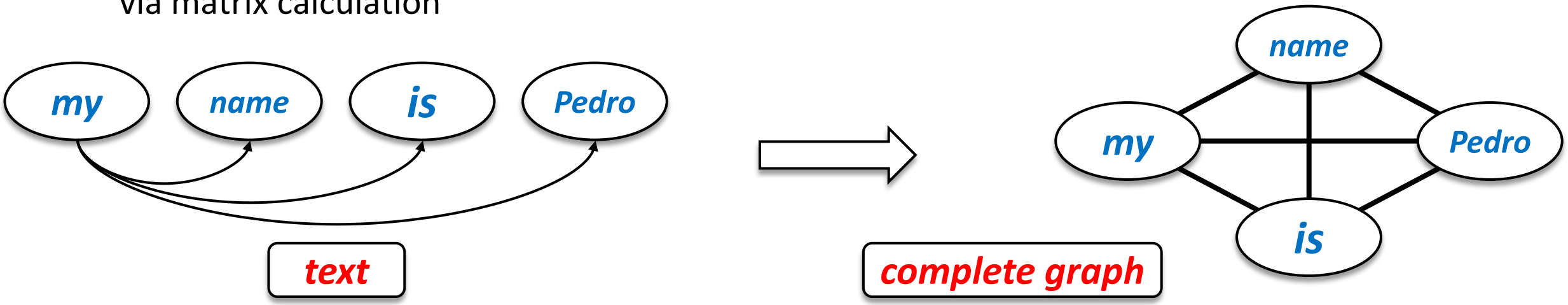
=

Drive the development of neural networks that are much more broadly applicable

GNN vs CNN vs Transformers

- ***Recap:** CNNs* can be seen as a ***special GNN with fixed neighbor size and ordering***:
  - The size of the filter is pre-defined for a CNN
  - The advantage of GNN is it processes arbitrary graphs
  - CNNs are NOT permutation equivariant → switching the order of pixels will leads to different outputs

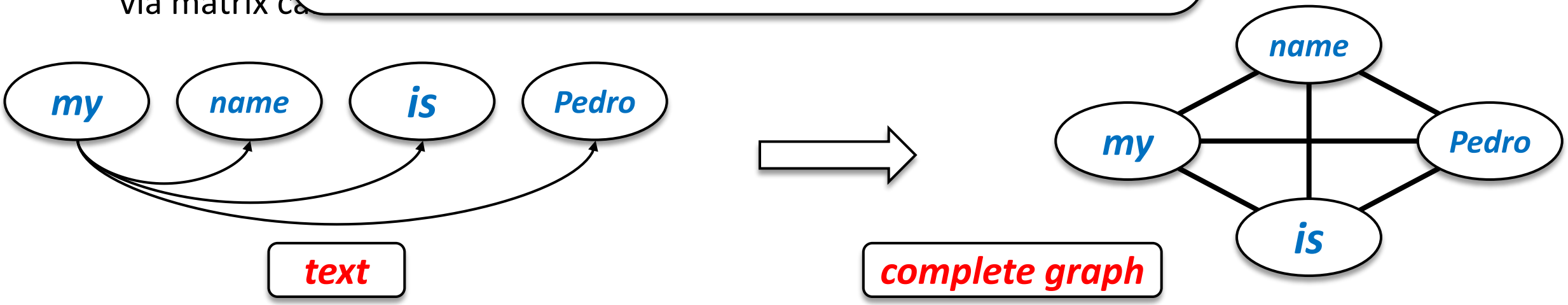
- ***Transformers:*** the most popular architecture to handle sequential data (text)
  - ***Fully rely on self-attention:*** every token (word) attends to all the other tokens via matrix calculation



GNN vs CNN vs Transformers

- ***Recap:** CNNs* can be seen as a ***special GNN with fixed neighbor size and ordering***:
  - The size of the filter is pre-defined for a CNN
  - The advantage of GNN is it processes arbitrary graphs
  - CNNs are NOT permutation equivariant → switching the order of pixels will leads to diff[illegible]
- ***Transformers:*** [illegible]xt)
  - ***Fully rely o*** [illegible] tokens via matrix ca[illegible]

Transformer** layer can be seen as a **special GNN** that runs **on a fully-connected** **"word" graph



GNNet – 1st Graph Neural Networking Workshop @CoNEXT 2022

- ***Networks are graphs*** ☺…the success of GNNs in recent years creates a great opportunity for a wide range of networking problems
- ***GNNet*** targets the ***application of Graph Neural Network*** (GNN) technology to ***networking problems***
- Help building a strong community among those of us interested in what GNN can bring to networking
- Include a Special Session featuring the ***best solutions*** from the ***BNN-UBP GNNet challenge 2022***

Albert Cabellos

Pere Barlet-Ros

Franco Scarselli

Pedro Casas

AUSTRIAN INSTITUTE OF TECHNOLOGY
TOMORROW TODAY

Thanks

Dr. Pedro Casas
Data Science & Artificial Intelligence
AIT Austrian Institute of Technology @Vienna

pedro.casas@ait.ac.at
http://pcasas.info