What is Graph Neural Networks (GNN)?

Exploring the World of GNNs

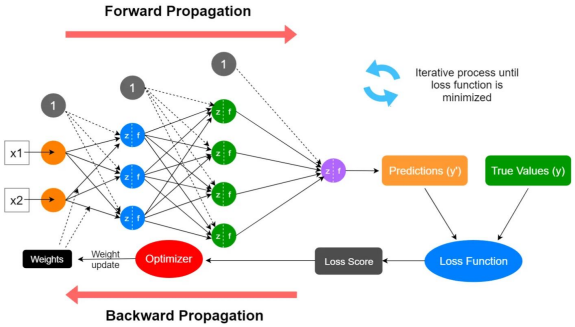
Tao Chen

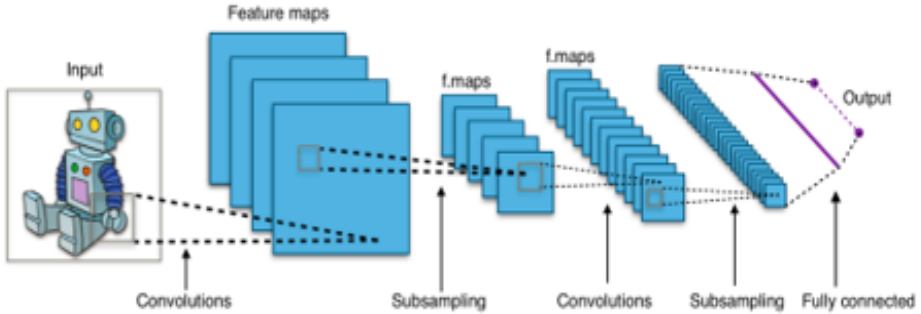
chentao.hfut@mail.hfut.edu.cn

Hefei University of Technology,
Department of Computer and Information Technology

Convolutional Neural Network

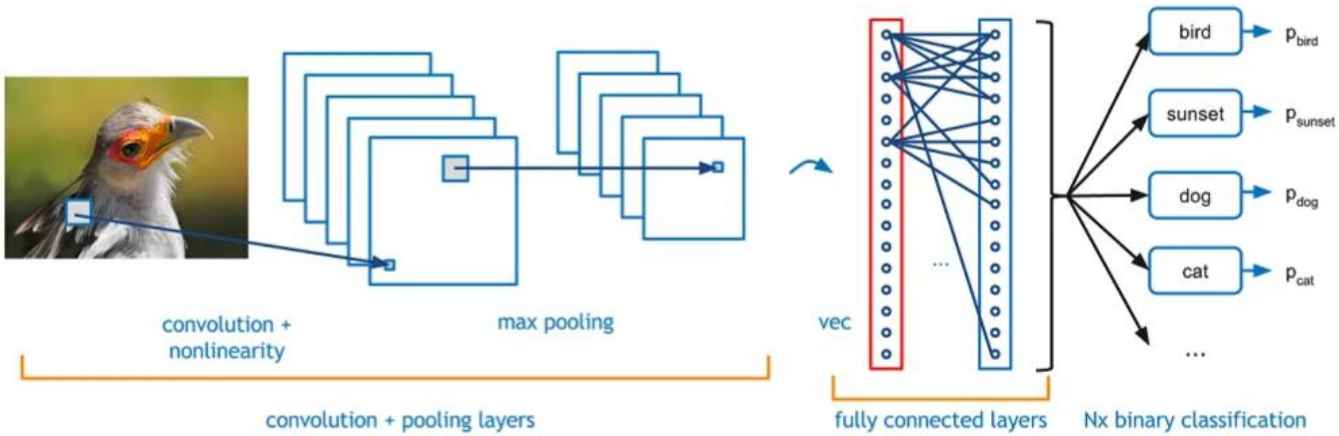
- Neural networks: A neural network is a computational model inspired by the structure of the human brain.
- Convolutional Neural networks: Convolutional neural network (CNN) gains great success on Euclidean data.
  - Image classification, object detection, machine translation.





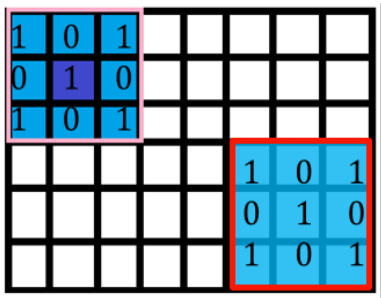
Convolutional Neural Network

- What is the key strength of CNNs?
  - CNNs excel at learning **local stationary structures** via localized convolution filters and building them into complex, multi-scale hierarchical structures.



Convolutional Neural Network

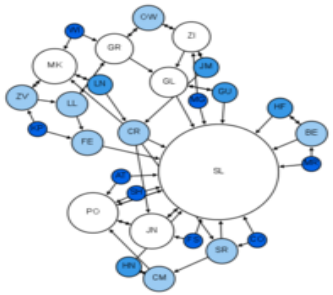
- Localized convolutional filters are **translation/shift-invariant** [1].
  - Which can recognize identical features independently of their spatial locations.
- One interesting problem is how to generalize convolution to a non-Euclidean domain, e.g., graph?
  - Irregular structure of graph poses challenges for defining convolution.

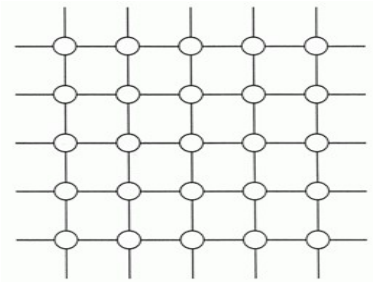


X-Shape

Template Matching

1 0 1
0 1 0
1 0 1

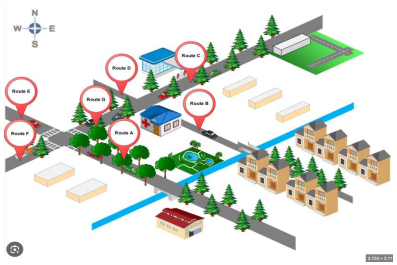


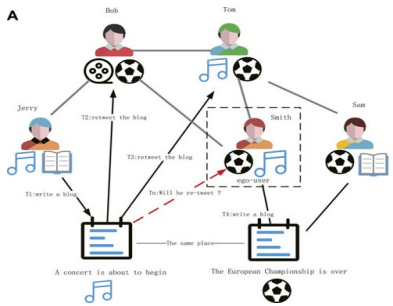


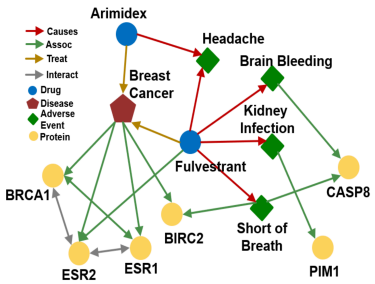
Convolution is well defined in Euclidean data, grid-like network.

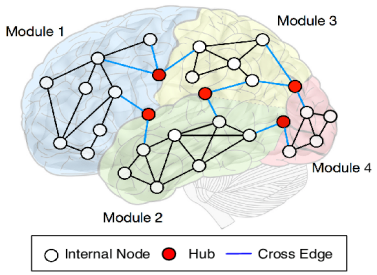
Non-Euclidean Data

- Diverse Applications of Non-Euclidean Data Structures.
  - **Transportation Systems**: Urban traffic flow analysis for smart city planning.
  - **Social Networks**: Facebook's social graph mapping friendships and interactions.
  - **Knowledge Graphs**: Google's Knowledge Graph in search engine algorithms.
  - **Brain Topology**: The neurons in the brain and their connections form a non-Euclidean structure.





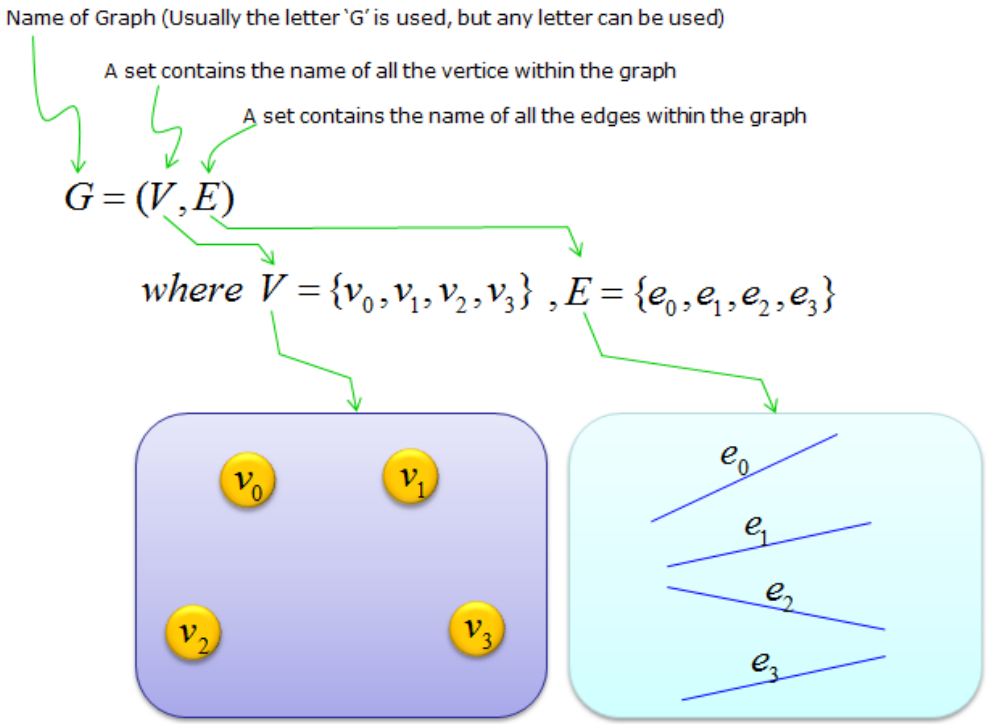




In real-world scenarios, there is a plethora of non-Euclidean data, which is why we have introduced Graph Neural Networks.

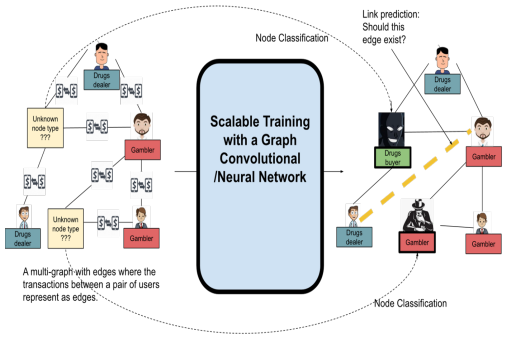
Preliminaries for GNNs

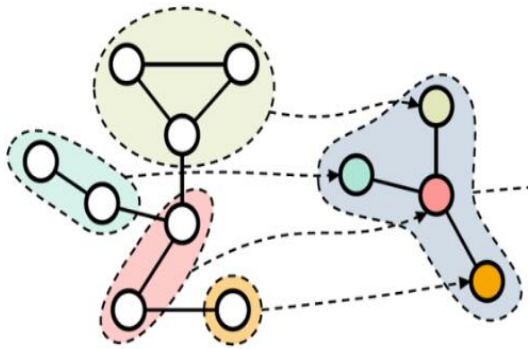
- The graph notation $G = (V, E)$
  - V is the set of vertices (or nodes).
  - E is the set of edges (or links) between vertices.

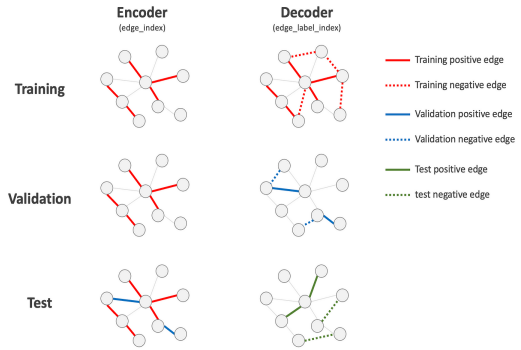


Typical Tasks for GNN

- Diverse Applications of Non-Euclidean Data Structures.
  - **Node classification**: Assigning labels to nodes based on their features and connections. It's commonly used in social network analysis and bioinformatics.
  - **Graph classification**: In this task, entire graphs (as units) are classified into different categories. This is particularly useful in chemical compound analysis and pattern recognition in various kinds of networks.
  - **Edge classification**: Predicting types or properties of edges in a graph, significant in knowledge graphs and recommender systems.

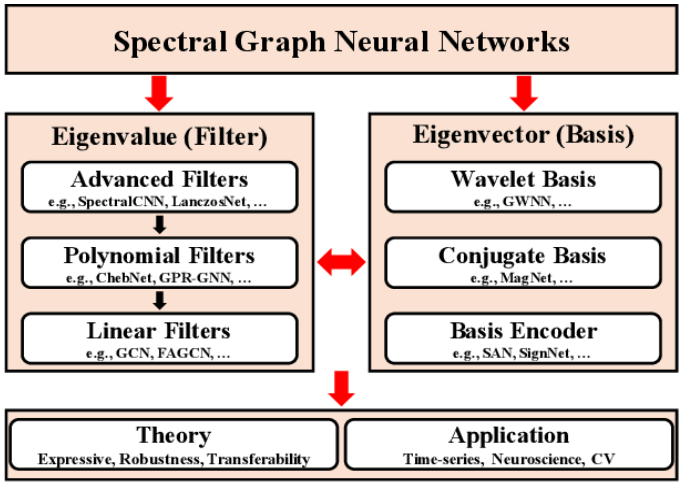


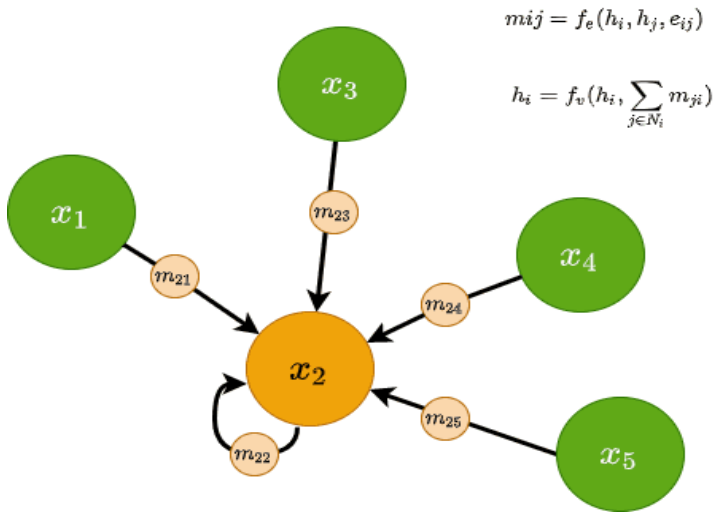




Typical Methods for GNN

- Diverse Applications of Non-Euclidean Data Structures.
  - **Spectral GNN methods**: These methods leverage the graph theory, particularly eigenvalues and eigenvectors of the graph Laplacian, to analyze graph structure.
  - **Spatial GNN methods**: Spatial methods directly operate on the graph nodes and their neighbors, focusing on local neighborhood information and making them well-suited for large and dynamic graphs.



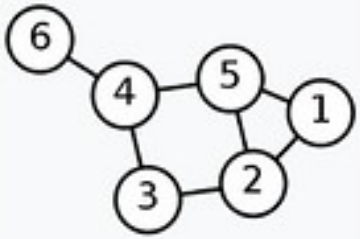


Spectral GCN Methods

- A graph $G = (V, E, W, X)$
  - V is node set with $n = |V|$, E is edge set, and $W \in R^{n \times n}$ is the weighted adjacency matrix.
  - Each node is associated with d features, and $X \in R^{n \times d}$ is the feature matrix of nodes, each column of X is a signal defined over nodes.
- Graph Laplacian
  - $L = D - W$, where D is a diagonal matrix with $D_{ij} = \sum_i W_{ij}$.
  - Graph Laplacian Normalization.

$$L = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} \tag{1}$$

where I is the identity matrix.

Labeled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

Graph Fourier Transform

- Fourier basis of graph G
  - The complete set of orthonormal eigenvectors $\{u_l\}_{l=0}^{n-1}$ of L, ordered by its non-negative eigenvalues $\{\lambda_l\}_{l=0}^{n-1}$.
  - Graph Laplacian could be diagonalized as

$$L = U\Lambda U^T \tag{2}$$

  where $U = [u_0, ..., u_{n-1}]$ and $\Lambda = diag([\lambda_0, ..., \lambda_{n-1}])$

- Graph Fourier transform
  - Graph Fourier transform of a signal $x \in R^n$ is defined as

$$\hat{x} = U^T x \tag{3}$$

  - Graph Fourier inverse transform is

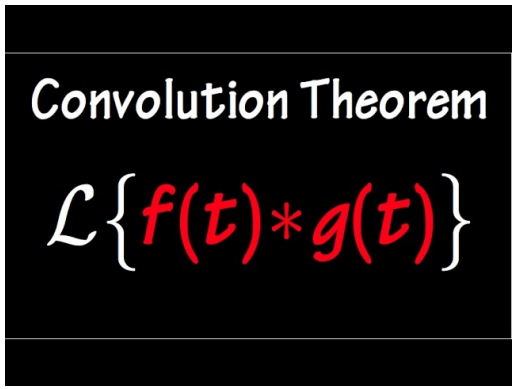
$$x = U\hat{x} \tag{4}$$

Convolution in spectral domain

- Convolution theorem
  - The Fourier transform of a convolution of two signals is the point-wise product of their Fourier transforms
- According to the convolution theorem, given a signal x as input and the other signal y as filter, graph convolution $*_G$ can be written as:

$$x *_G y = U\left(\left(U^T x\right) \cdot \left(U^T y\right)\right) \tag{5}$$

Here, the convolution filter in the spectral domain is $U^T y$.

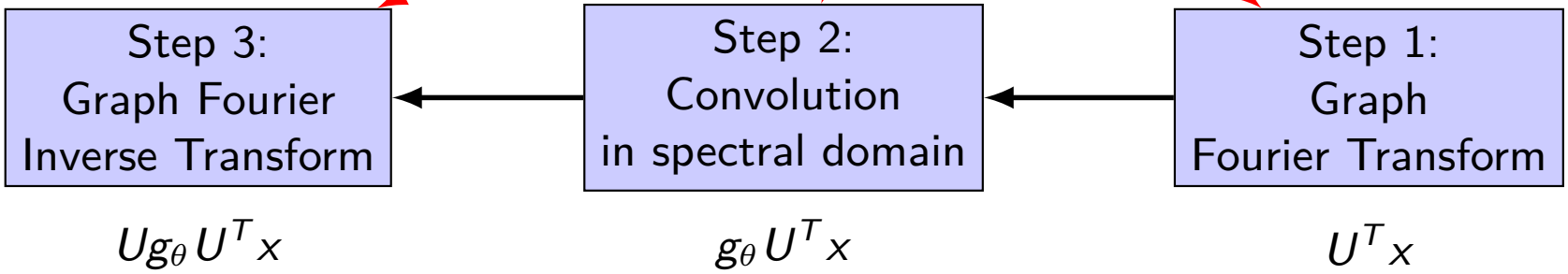


Convolution in spectral domain

- Let $\boldsymbol{U}^T y = [\boldsymbol{\theta_0}, \cdots, \boldsymbol{\theta}_{n-1}]^T$ and $g_\theta = \mathrm{diag}([\theta_0, \cdots, \theta_{n-1}])$, we have
  - The Fourier transform of a convolution of two signals is the point-wise product of their Fourier transforms

$$\boldsymbol{x} *_G \boldsymbol{y} = U\left(\left(U^T \boldsymbol{x}\right) \odot \left(U^T \boldsymbol{y}\right)\right)$$

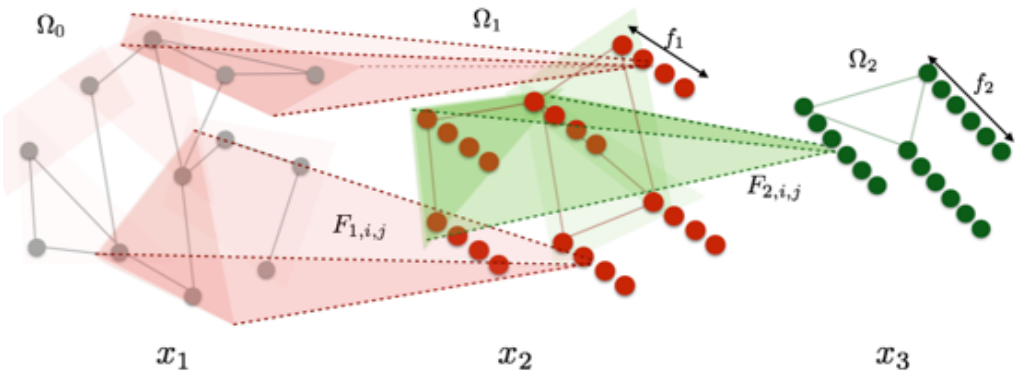
$$x *_G y = U g_\theta U^T x$$



Spectral Graph CNN

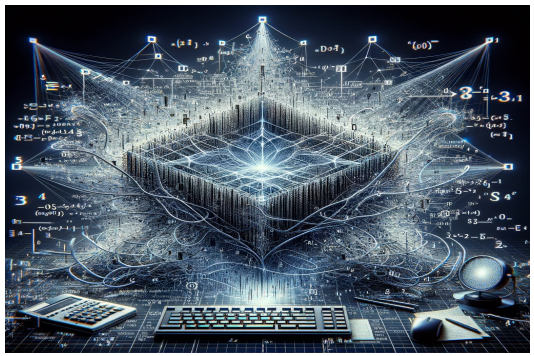
- Spectral Graph CNN

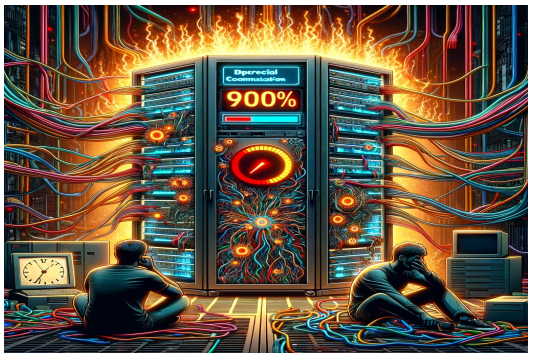
$$x_{k+1,j} = h\left(\sum_{i=1}^{f_k} U F_{k,i,j} U^T x_{k,i}\right), \qquad j = 1, \cdots, f_{k+1} \tag{6}$$

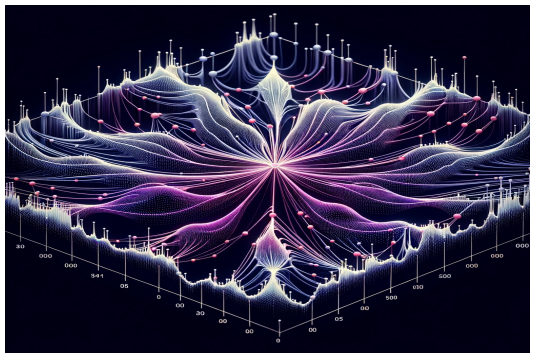


Limitation of Spectral Graph CNN

- The limitations of spectral GCNs are summarized as follows:
  - **Eigen-decomposition**: Spectral GCNs require the eigen-decomposition of the Laplacian matrix, which is a computationally intensive task.
  - **High Computational Costs**: The operations involved in spectral GCNs, particularly eigen-decomposition, are resource-intensive, leading to high computational costs.
  - **Non-localization in Vertex Domain**: Spectral GCNs lack localization, meaning that the filters applied are global and not restricted to the immediate neighborhood of a vertex, which can lead to inefficiencies in capturing localized patterns in the data.

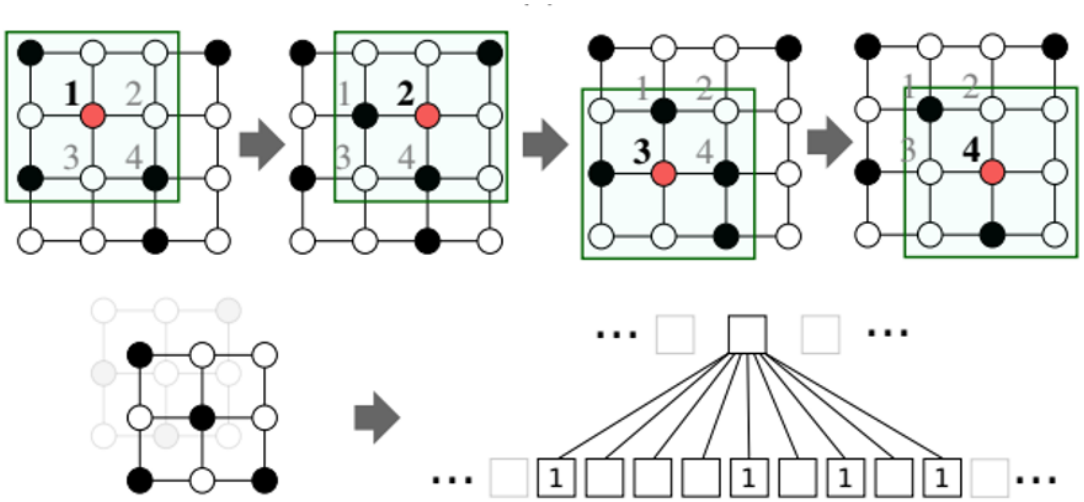


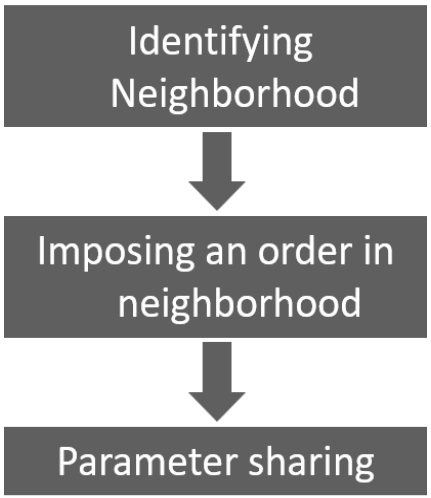




Spatial Graph CNN

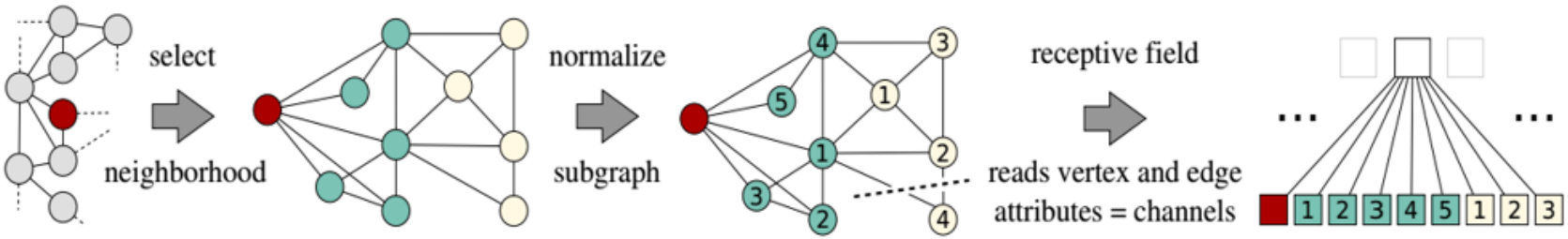
- Analogy[2]
  - What can we learn from the architecture of a standard convolutional neural network?





Spatial Graph CNN

- Analogy
  - Select a fixed number of nodes as neighbors for each node, based on a specific proximity metric.
  - Establish an order for the nodes using the proximity metric as a basis.
  - Implement parameter sharing across the nodes.

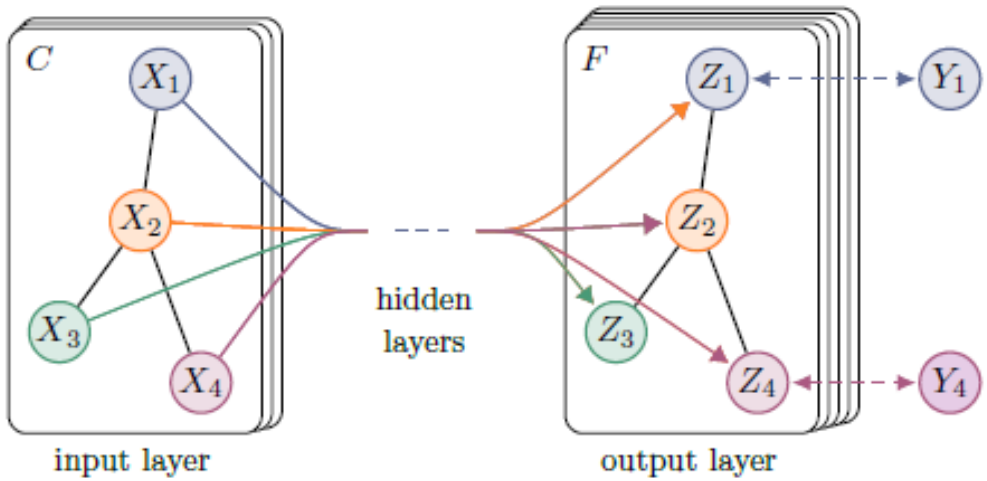


Step 1: Identifying Neighborhood	Step 2: Imposing an order in Neighborhood	Step 3: Parameter Sharing

Spatial Graph CNN

- GCN: Graph Convolution Network
  - Aggregating information from neighborhood via a normalized Laplacian matrix.
  - Parameters sharing is from feature transformation.

$$Z = f(X, A) = \mathrm{softmax}\left(\hat{A}\,\mathrm{ReLU}\left(\hat{A}XW^{(0)}\right)W^{(1)}\right) \qquad (7)$$



Framework of Graph Neural Network

- GNNs offer us a **flexible and effective** framework for both node classification and graph classification tasks
  - Leveraging graph structure and node feature to learn a representation vector of each node h_v, or the entire graph h_G.
  - Neighborhood aggregation strategy: iteratively updates a node's representation by aggregating its neighbors' representations.
- GNN framework: **Aggregate + Combine**

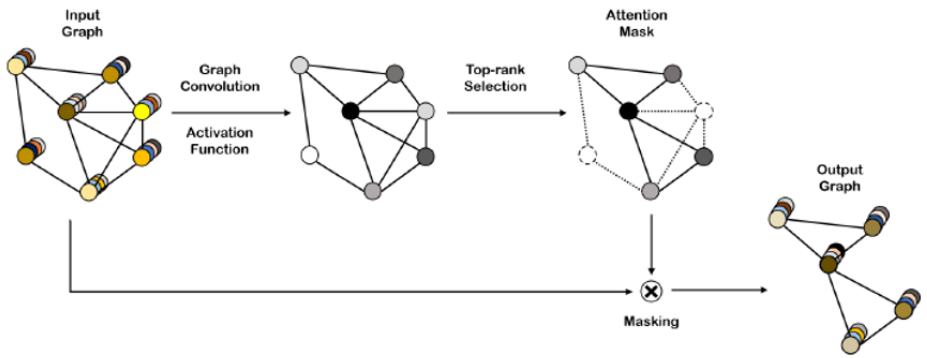
$$a_v^{(k)} = \text{AGGREGATE}^{(k)}\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right) \tag{8}$$

$$h_v^{(k)} = \text{COMBINE}^{(k)}\left(h_v^{(k-1)}, a_v^{(k)}\right) \tag{9}$$

$h_v^{(k)}$ is the feature vector of node at the k-th layer/iteration of GNN, and $h_v^{(0)} = X_v$. $\mathcal{N}(v)$ is a set of nodes adjacent to v.

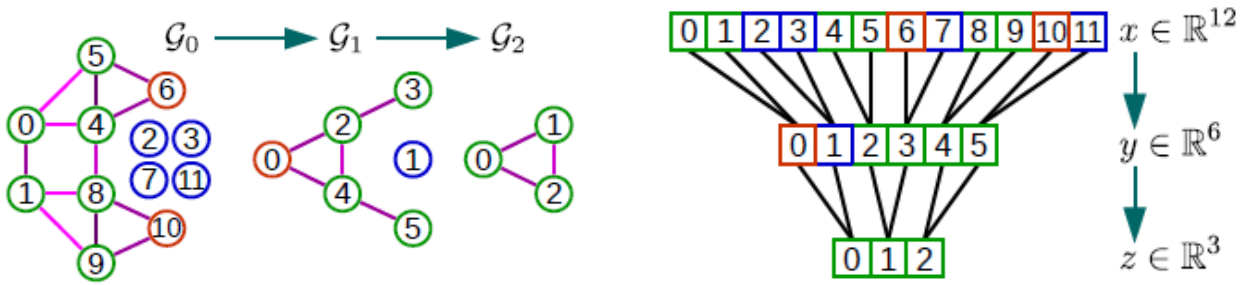
Graph pooling via node selection[3]

- Graph coarsening: Graph coarsening aims to reduce the size of a graph by merging similar nodes or edges, resulting in a coarser, simpler graph.
  - **Node Selection Strategies**: Nodes can be selected based on various criteria like node importance, feature similarity, or structural roles in the graph.
  - **Preserving Graph Structure**: An important aspect of node selection is to preserve the essential structural and feature-related information of the original graph.
  - **Hierarchical Graph Representations**: Graph pooling layers can be stacked to produce hierarchical representations of graphs, capturing features at different scales.



Graph Pooling via graph coarsening [4]

- Graph coarsening: Graph coarsening aims to reduce the size of a graph by merging similar nodes or edges, resulting in a coarser, simpler graph.
  - **Node Merging**: Nodes are often merged based on **similarity, connectivity, or other criteria**, forming super-nodes in the coarser graph.
  - **Edge Reduction**: Alongside node merging, edges may also be reduced or reweighted to reflect the new structure of the coarser graph.
  - **Application in GNNs**: In graph neural networks, coarsening is used to reduce the graph size.
  - **Impact on Performance**: Proper graph coarsening must be taken to ensure that the coarsening process does not lead to a significant loss of information.



Graph Neural Networks

- Graph neural networks (GNNs) gained remarkable success
  - Achieving state-of-the-art empirical performance in node classification, link prediction, and graph classification.
- The design of new GNNs is mostly based on empirical intuition, heuristics and experimental trial-and-error.
- We lack theoretical understanding of the properties and limitations of GNNs.
  - One fundamental problem is the expressive power of GNNs.

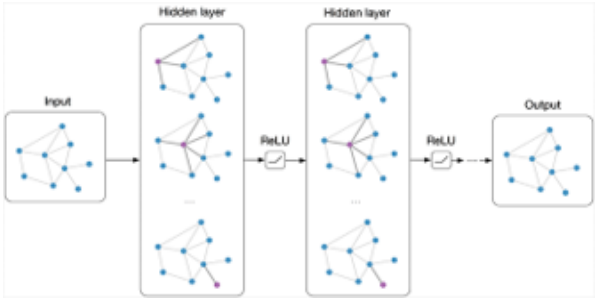
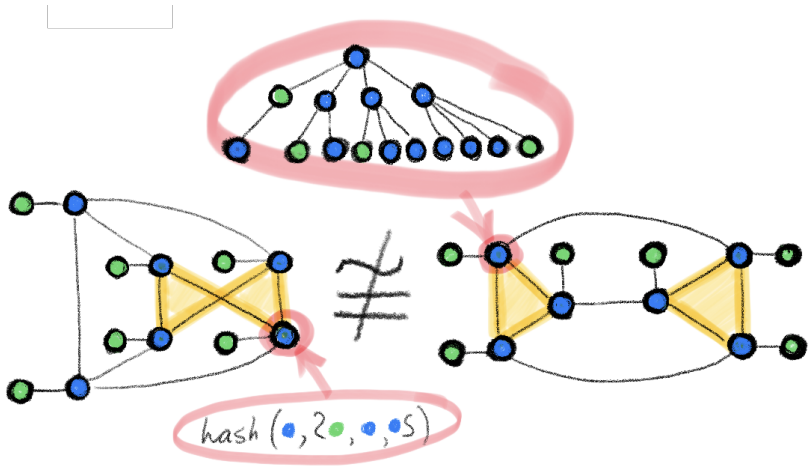


Figure 1: Node classification task

Figure 2: Graph classification task

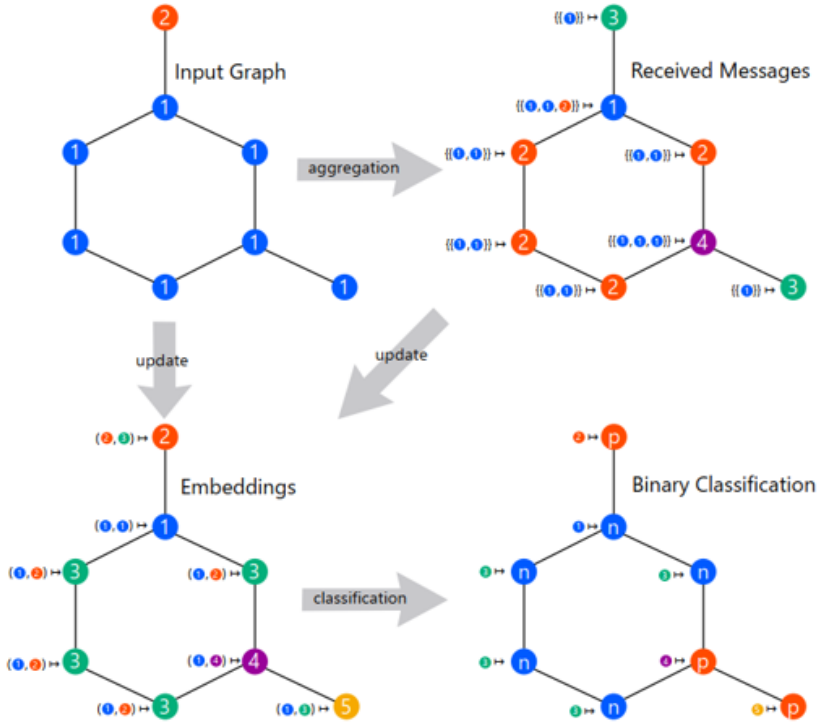
Expressive Power

- Number of possible distinct occurrences
  - Expressive power of n bits is 2^n.
    - ✓ For example, 5 bits can distinguish 32 distinct states.
    - ✓ Given no more than 32 bottles of water and one of them is poisonous, at most 5 mice are required to identify which bottle is poisonous.
- Approximation capability
  - Expressive power of 1-layer perception.
  - Multi-layer perceptron offers us a universal approximator.



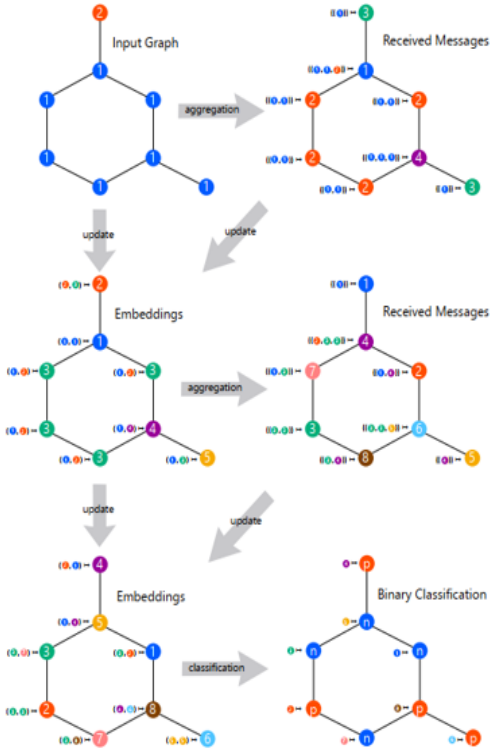
GNNs for node classification

- 1-layer GCN
  - Limited expressive power of 1-layer GCN: it cannot fully distinguish all nodes



GNNs for node classification

- 2-layer GCN
  - 2-layer GCN can fully distinguish all nodes in this toy example.
  - Depth of GNNs matters.
- Can the expressive power of GNNs be improved infinitely via improving their depth? Is there a bound?



Over-smooth issue

- Effect on Model Performance
  - As a result of over-smoothing, the performance of GCNs on tasks like node classification, graph classification, and others can significantly degrade.
  - The model might end up producing similar outputs for all nodes, reducing its ability to make accurate predictions.

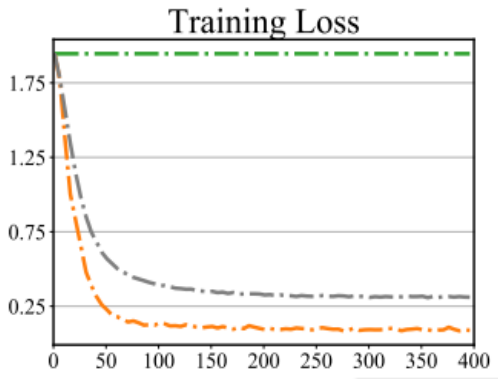


Figure 3: Node classification task

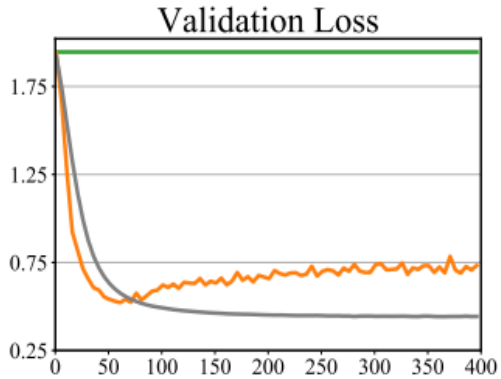


Figure 4: Graph classification task

Why deeper GNNs perform worse than their shallow counterparts?

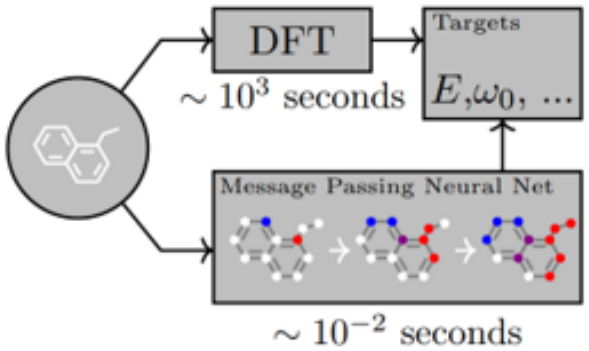
Expressive Power

- Three major scenarios
  - Node-level
    - ✓ Node classification: predict the label of nodes according to several labeled nodes and graph structure.
    - ✓ Link prediction: predict the existence or occurrence of links among pairs of nodes.
- Graph-level
  - Graph classification: predict the label of graphs via learning a graph representation using graph CNN.
- Signal-level
  - Signal classification, similar to image classification which is signal-level scenario on a grid-like network

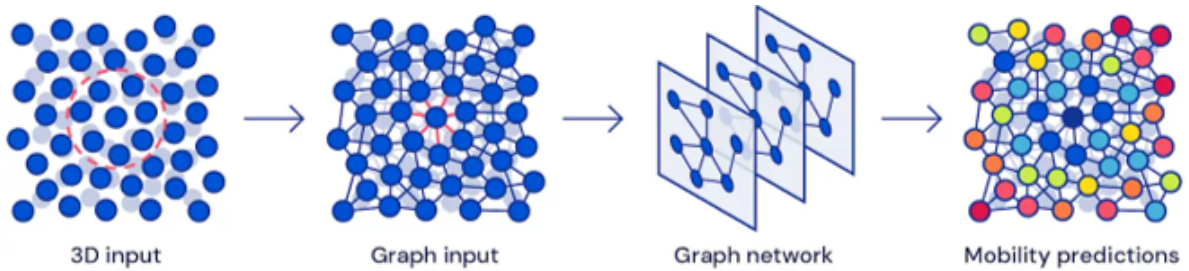
Applications in other fields

- GNNs for quantum chemistry
  - Predict the quantum properties of molecules using GNNs.
  - Traditional methods, i.e., DFT (Density Functional Theory), is computationally expensive .

- GNNs for quantum chemistry
  - Predict the quantum properties of molecules using GNNs.
  - Traditional methods, i.e., DFT (Density Functional Theory), is computationally expensive.



- Modeling dynamics of glassy systems using GNNs



[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 48, New York, New York, USA, 2016, pp. 2014–2023.

[3] J. Lee, I. Lee, and J. Kang, "Self-attention graph pooling," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97, 2019, pp. 3734–3743.

[4] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, 2018.

Thank you for your attentions!