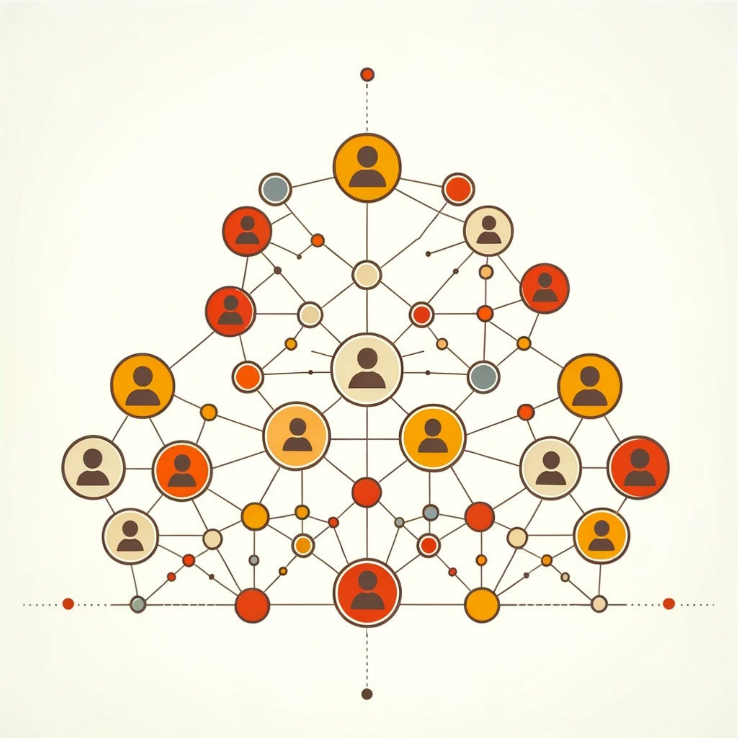
Lecture 18

Graph Neural Network- Part 1

Graphs are structured data

- Many real-world datasets come in the form of graphs.

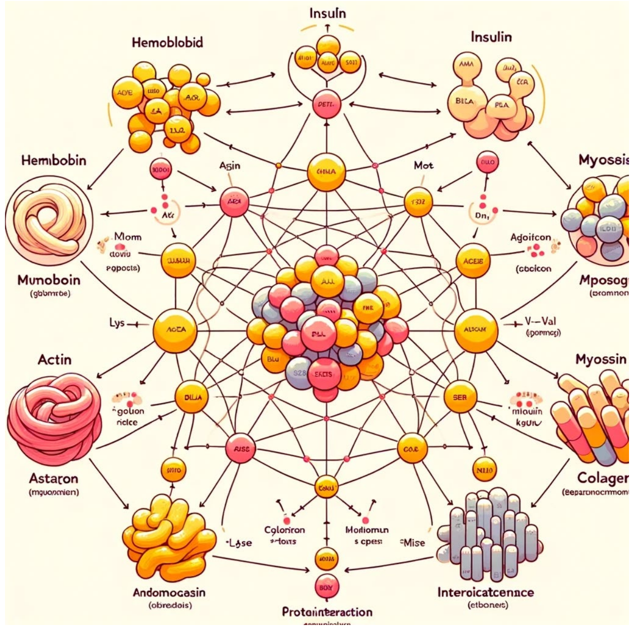
- Social networks



Graphs are structured data

- Many real-world datasets come in the form of graphs.

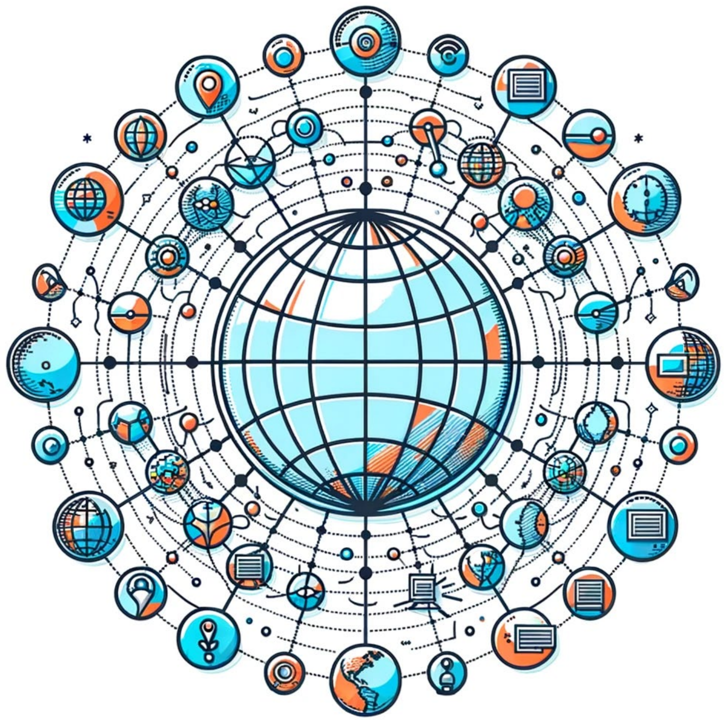
- Social networks
- Protein-interaction networks



Graphs are structured data

- Many real-world datasets come in the form of graphs.

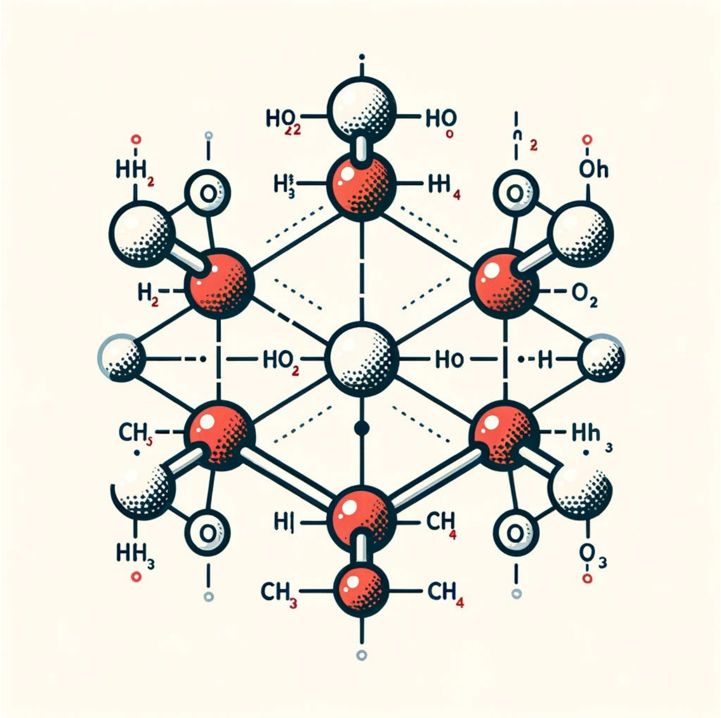
- social networks
- protein-interaction networks
- The World Wide Web



Graphs are structured data

- Many real-world datasets come in the form of graphs.

- social networks
- protein-interaction networks
- The World Wide Web
- Molecules



Images are graphs

- Images are graphs, where each pixel represents a node and is connected via an edge to adjacent pixels

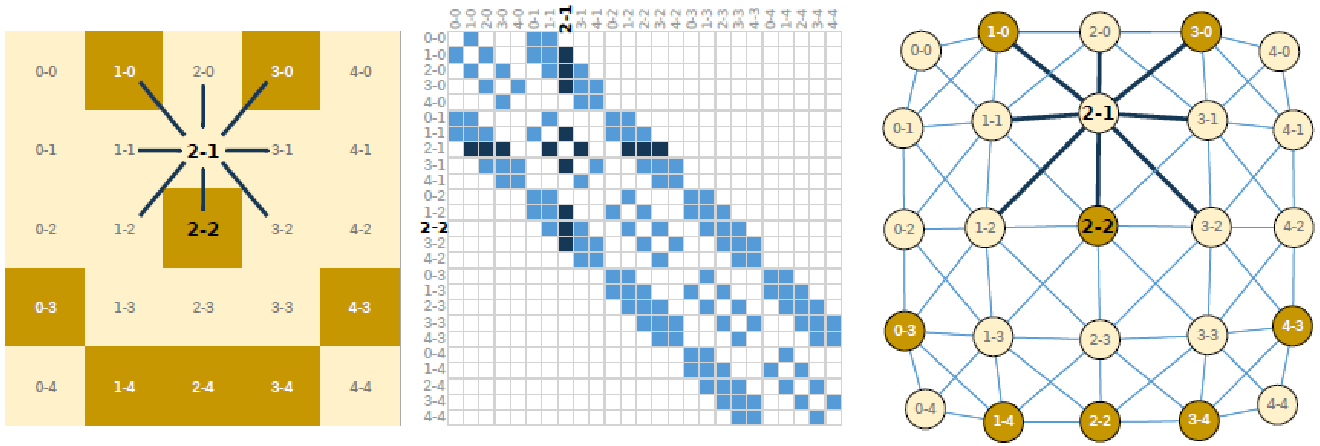


Image Pixels Adjacency Matrix Graph

Image by Sanchez-lengeling

Text as graphs

- Each token is a node and is connected via an edge to the node that preceding it.

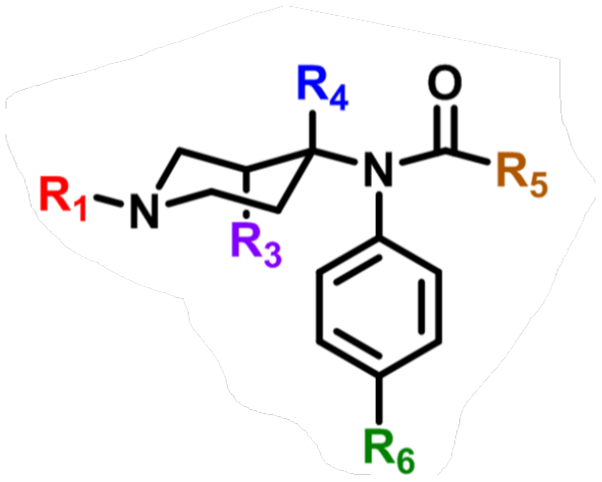
Texts are graphs

Tasks

- Graph-level task
- Node-level task
- Edge-level task

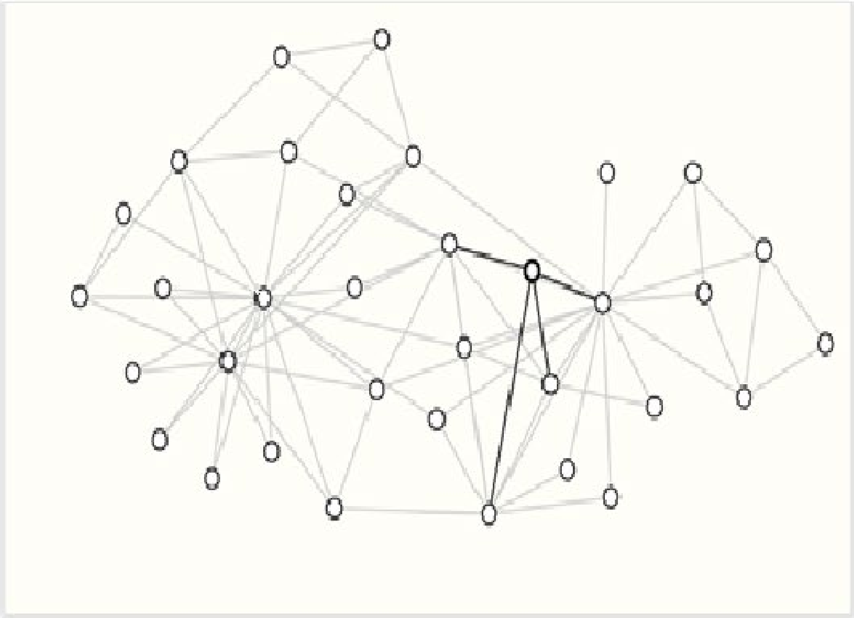
Graph-level task

- Predict the property of an entire graph.
- Predict whether a molecule will bind to a receptor or not.

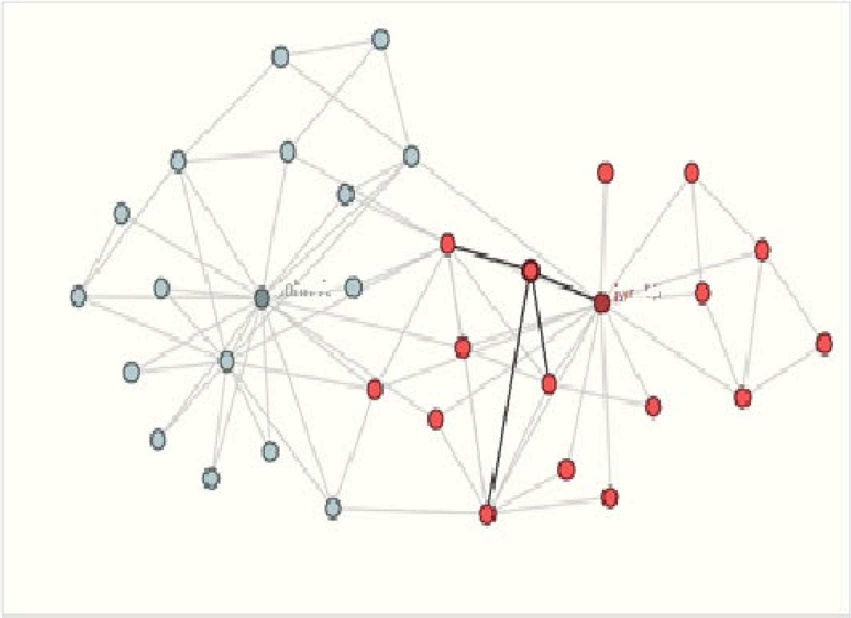


Node-level task

- Predicting the identity or role of each node within a graph.



Input: graph with unlabled nodes



Output: graph node labels

Edge-level task

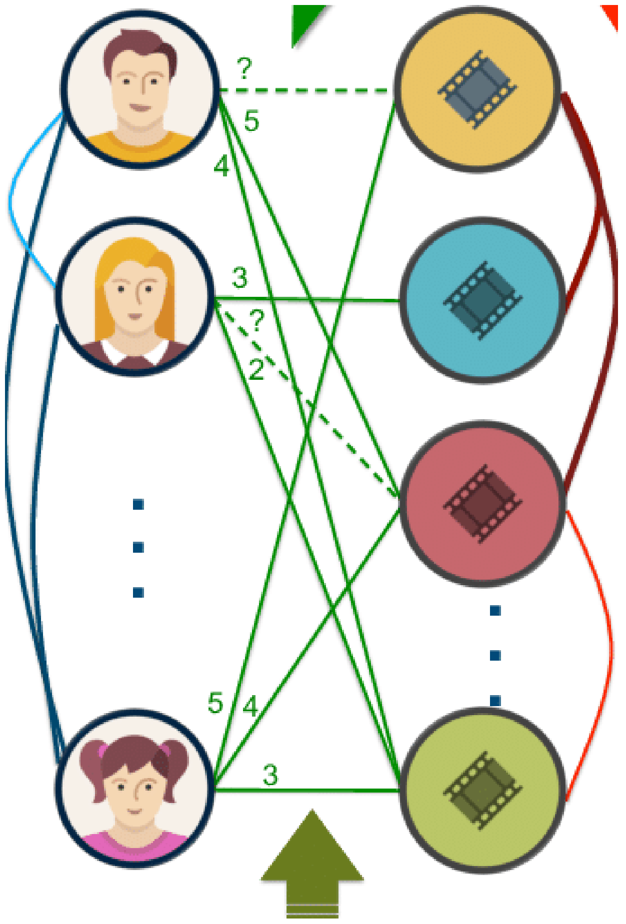
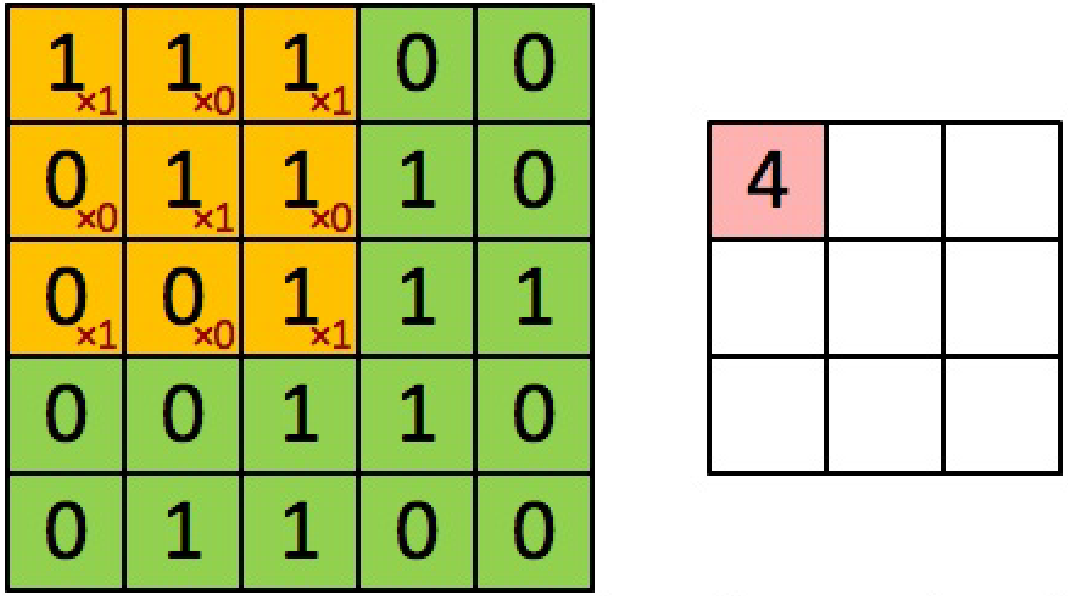


Image by Shobeir Fakhraei

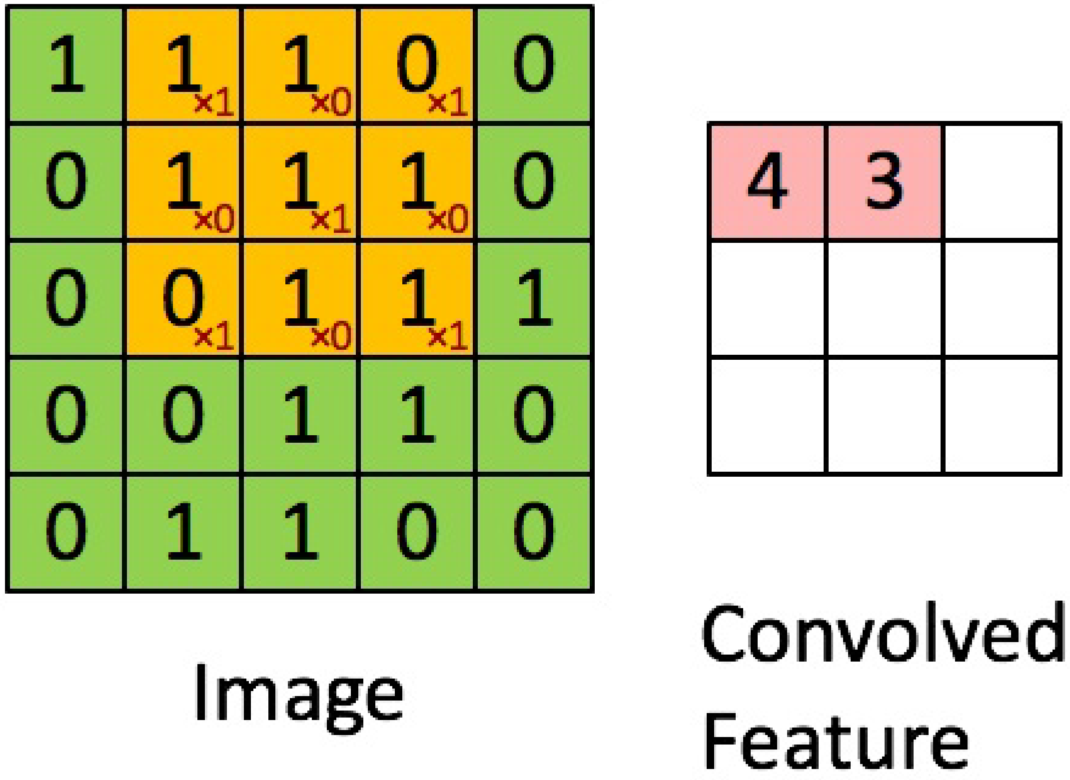
CNN as GNN



Image

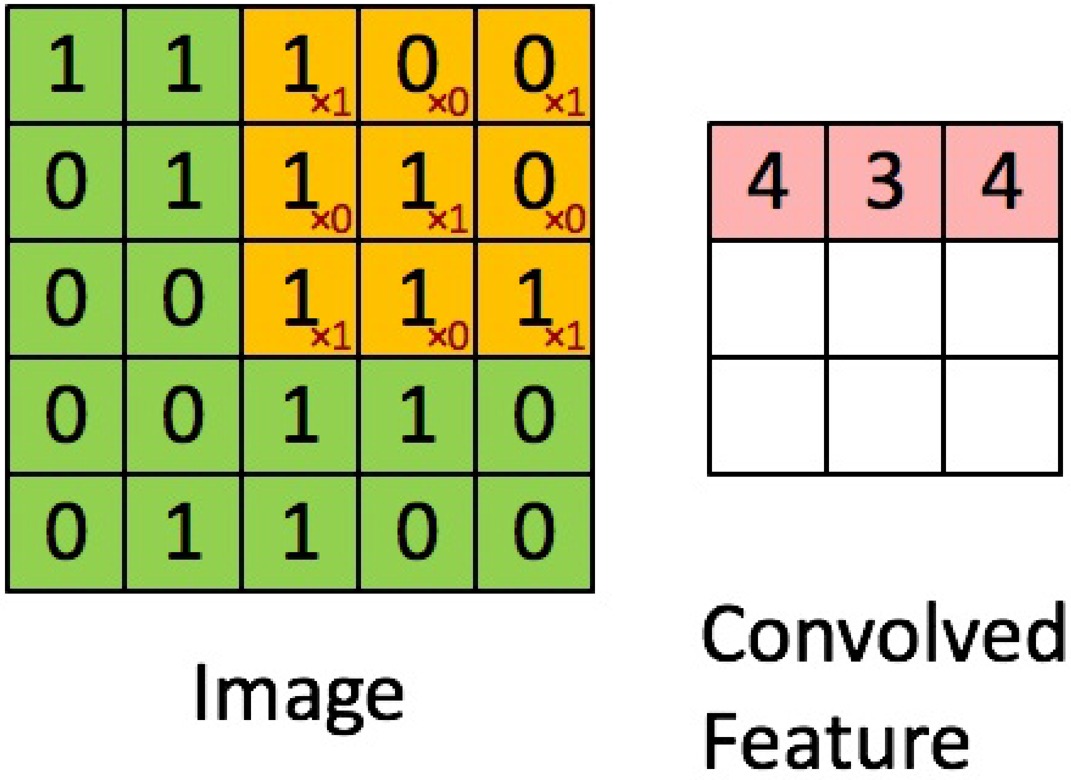
Convolved Feature

CNN as GNN

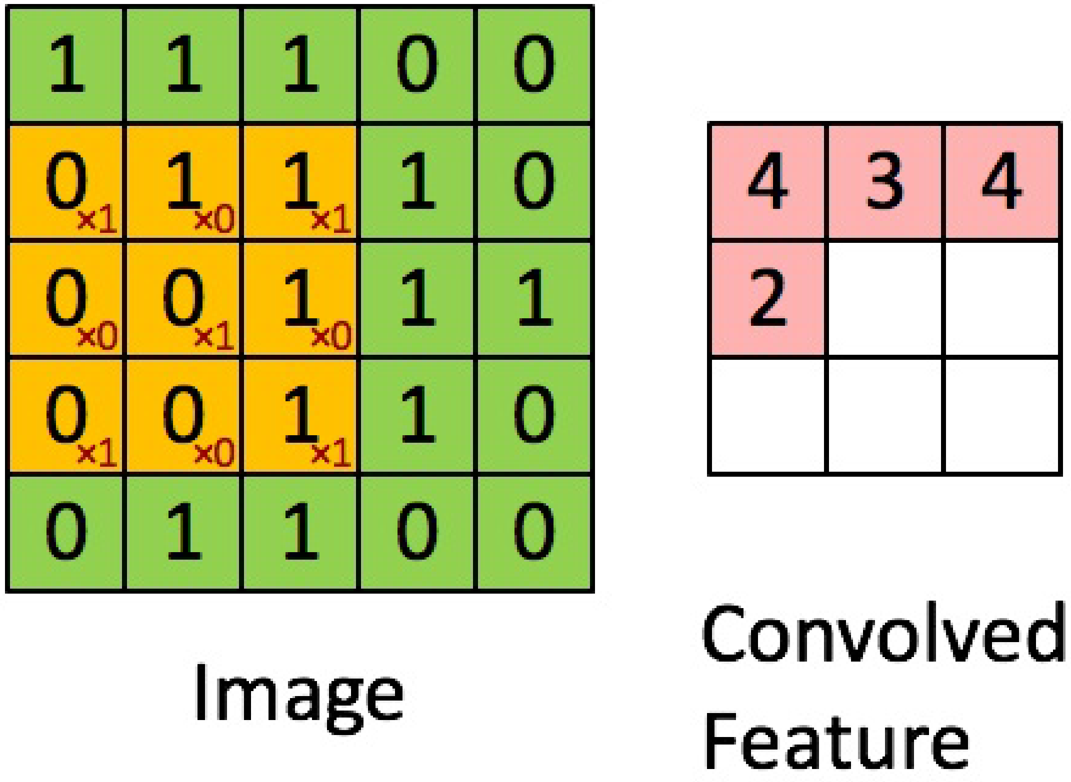


1 1 ×1 1 ×0 0 ×1 0
0 1 ×0 1 ×1 1 ×0 0
0 0 ×1 1 ×0 1 ×1 1
0 0 1 1 0
0 1 1 0 0
Image
4 3
Convolved Feature

CNN as GNN


1 1 1 ×1 0 ×0 0 ×1
0 1 1 ×0 1 ×1 0 ×0
0 0 1 ×1 1 ×0 1 ×1
0 0 1 1 0
0 1 1 0 0
4 3 4
Image
Convolved Feature

CNN as GNN



1 1 1 0 0
0 ×1 1 ×0 1 ×1 1 0
0 ×0 0 ×1 1 ×0 1 1
0 ×1 0 ×0 1 ×1 1 0
0 1 1 0 0
Image
4 3 4
2
Convolved Feature

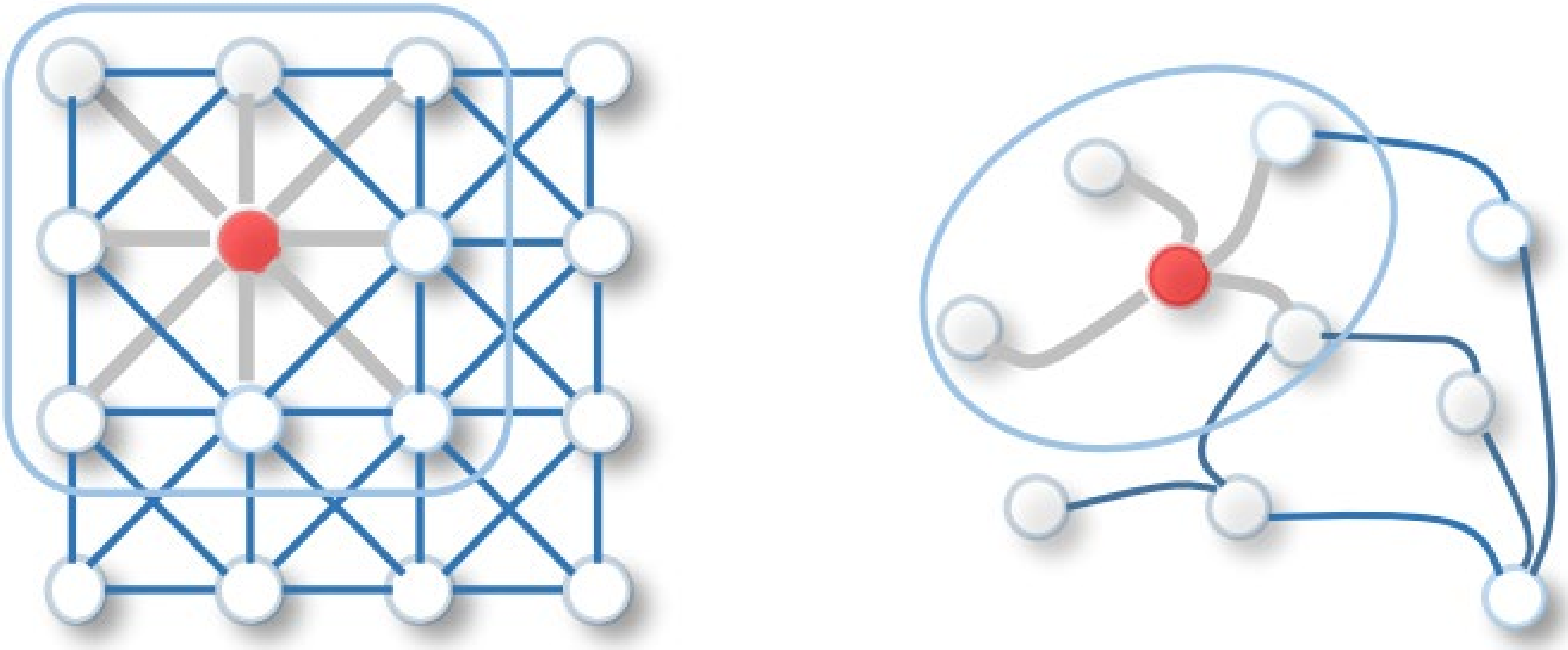


Image by Zonghan Wu et al

Convolution

This operation is called convolution.

$$s(t) = \int x(a)w(t-a)da$$

The convolution operation is typically denoted with an asterisk:

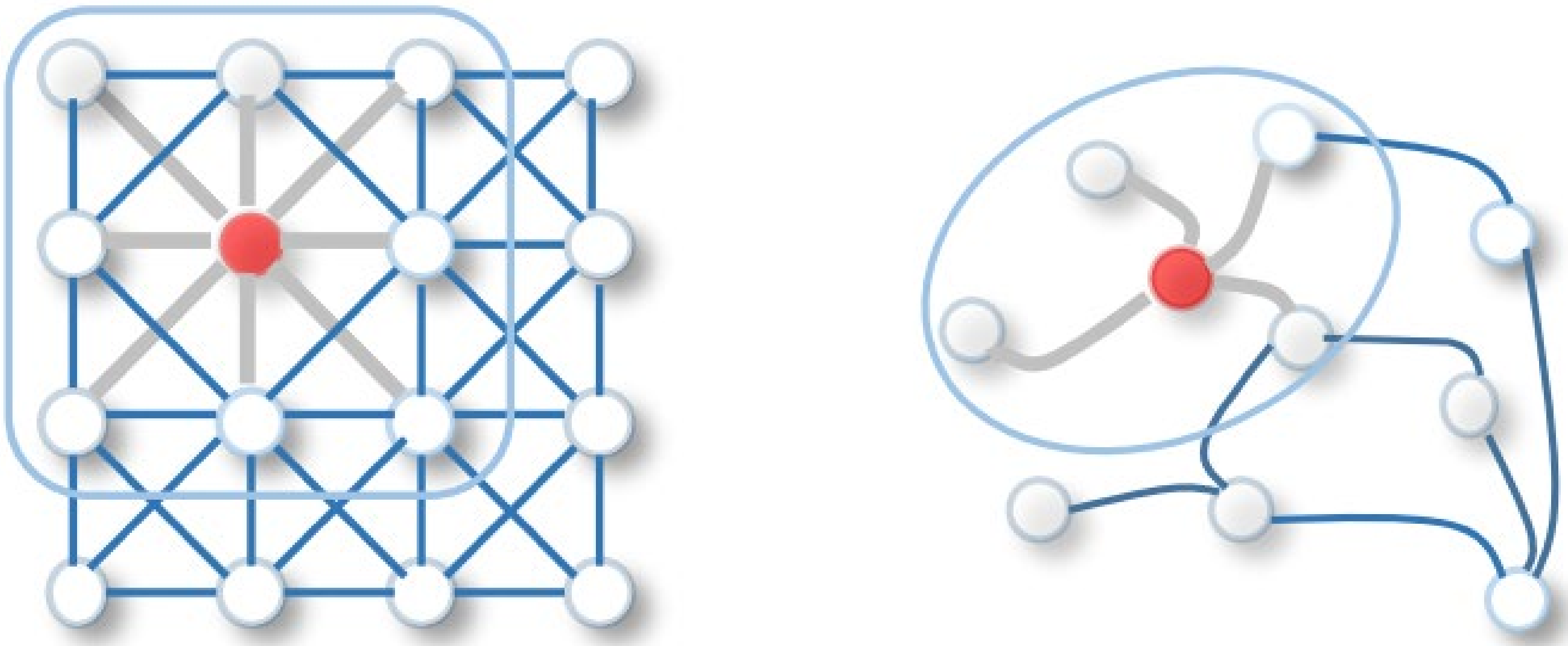
$$s(t) = (x * w)(t)$$

Discrete convolution

If we now assume that x and w are defined only on integer t, we can define the discrete convolution:

$$s[t] = (x * w)(t) = \sum_{a=-\infty}^{\infty} x[a]w[t-a]$$

Convolution on Graphs



Definition of a Graph

A graph G can be defined as a set of vertices V and edges E, along with an adjacency matrix A.

Graph Notation

$$G = (V, E, A)$$

- V: Vertices or Nodes
- E: Edges, representing connections between vertices
- A: Adjacency Matrix, indicating the presence (1) or absence (0) of an edge between vertex pairs

The adjacency matrix is a binary matrix indicating whether pairs of vertices are adjacent.

- $A_{ij} = 1$ if there is an edge between vertex i and vertex j
- $A_{ij} = 0$ otherwise

Laplacian of a Graph

The Laplacian matrix L of a graph provides insights into the graph's structure, including its connectivity and the presence of clusters.

Laplacian Matrix

$L = D - A$

- D: Degree Matrix, a diagonal matrix with the degree of each vertex along the diagonal
- A: Adjacency Matrix, as defined above

Degree Matrix and Degree of a Vertex

The degree matrix D is a square matrix where each diagonal element d_i represents the degree of the corresponding vertex i.

Degree Matrix Notation

$$D = \begin{bmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n \end{bmatrix}_{n \times n}$$

Degree of a Vertex

$$d_i = \sum_j A_{ij}$$

- The degree d_i is the sum of the elements in the i-th row of A, representing the number of edges connected to vertex i.

Example

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\text{Sum} = \begin{bmatrix} \sum A_{1j} \\ \sum A_{2j} \\ \sum A_{3j} \\ \sum A_{4j} \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ 3 \\ 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Laplacian Matrix

$$L = D - A$$

Graph Cut Optimization Problem

- **Objective**: Find a cut that divides the graph into segments with minimal interconnections.
- **Minimization Target**: $\sum A_{ij}(y_i - y_j)^2$, captures the 'cut' cost.
- **Equation**: $y^T Ly$, where L is the Laplacian matrix and y is a vector indicating node segments.
- **Constraint Applied**: $y^T y = 1$, ensures non-trivial solutions.
- **Vector y**: Represents the assignment of nodes to segments, dimension $n \times 1$.
- **Solution Method**: Eigen decomposition of L identifies optimal partitioning.

Eigen Decomposition of Laplacian

- Decomposed as $L = U \Lambda U^T$
- U: Orthonormal eigenvectors
    - Eigenvectors are orthogonal and normalized
- Λ: Diagonal matrix with eigenvalues
    - Each diagonal entry is an eigenvalue that pairs with an eigenvector in U

Normalized Laplacian

- Start with standard Laplacian: $L = D - A$
  - D: Degree matrix
  - A: Adjacency matrix
- Normalized Laplacian is defined as:

$$\widetilde{L} = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}}$$

- Can be simplified to:

$$\widetilde{L} = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$$

  - I: Identity matrix

Laplacian Eigenvectors & Fourier Analysis Analogy

- **Fourier Analysis:**
- **Mathematical Definition**: Decomposition of a signal into sinusoidal components.
- **Frequency Domain**: A signal is transformed to represent it as a sum of its frequency components.
- **Basis Functions**: Sine and cosine functions serve as the basis for this transformation.
- **Orthogonality**: These basis functions are orthogonal, ensuring unique frequency representation.

Graph Laplacian Eigenvectors

- **Graph Signals**: Functions defined over the nodes of a graph.
- **Spectral Domain**: Eigenvectors of L transform graph signals into the spectral domain.
- **Eigenvector Basis**: Analogous to sines and cosines, eigenvectors form a basis for graph signals.
- **Orthogonality**: Eigenvectors are orthogonal, providing a unique spectral representation for graph signals.

Eigenvectors of Laplacian

- **Low Eigenvalues**: Correspond to "low-frequency" eigenvectors.
  - These eigenvectors change slowly over the graph.
  - Represent large-scale, smooth structures in the graph.
- **High Eigenvalues**: Correspond to "high-frequency" eigenvectors.
  - These eigenvectors change rapidly between connected nodes.
  - Capture fine details or irregularities in the graph.
- **Ordering**: Eigenvalues (and eigenvectors) are ordered from lowest to highest.

Fourier Functions

- The following is the eigen-decomposition of graph Laplacian,

$$L = U^T \Lambda\, U$$

$$U = [\boldsymbol{u_1}, \ldots, \boldsymbol{u_n}]$$

$\Lambda_{ii} = \lambda_i$ diagonal matrix of eigenvalues (spectrum)

$$U^T U = I$$

$\boldsymbol{u_1}$ to $\boldsymbol{u_n}$ are eigenvectores of laplacian also called Fourier functions.

Fourier Transform

- The Fourier transform is projecting a signal x on the Fourier functions.

- The result is the coefficients of the Fourier series.

- The graph Fourier transform projects the input graph signal to the orthonormal space where the basis is formed by eigenvectors of the normalized graph Laplacian.

Fourier Transform

- Suppose $\boldsymbol{x} \in R^n$ is a feature vector of all nodes of a graph where x_i is the value of the i^{th} node.

- The graph Fourier transform to a signal x

$$f(x) = U^T x = \hat{x}$$

- The inverse graph Fourier transform

$$f^{-1}(\hat{x}) = U\hat{x} = UU^T x = x$$

Convolution Theorem

- The graph convolution of the input signal x with a filter $g \in \mathbb{R}^n$ is defined as:

Convolution Theorem: $f\,(\boldsymbol{x} * \boldsymbol{g}) = \big(f(x) \cdot f(g)\big)$

$$\begin{aligned}\boldsymbol{x} * \boldsymbol{g} &= f^{-1}\big(f(x) \cdot f(g)\big) \\ &= U(U^T x \cdot U^T g) \\ &= U g_\theta U^T x\end{aligned}$$

Where $g_\theta = \mathrm{diag}(U^T g)$

Vanilla Spectral GCN and ChebNet

ConvGNNs

- Spectral-based
  - Graph signal processing perspective.

- Spatial-based
  - Define graph convolutions by information propagation.

- GCN [1] bridged the gap between spectral-based approaches and spatial-based approaches.

[1] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in Proc. of ICLR, 2017.

Convolution Theorem

- The graph convolution of the input signal x with a filter $g \in \mathbb{R}^n$ is defined as:

$$\begin{aligned} x * g &= f^{-1}\big(f(x) \cdot f(g)\big) \\ &= U(U^T x \cdot U^T g) \\ &= U g_\theta U^T x \end{aligned}$$

Where $g_\theta = \mathrm{diag}(U^T g)$

Neural Network Layers

Feedforward Neural Networks (FFNN):

- Computation in layers: $h' = \sigma(Wh)$
- Each layer's output h' becomes the next layer's input.
- h_0 (initial input) is the feature vector x.

Convolutional Neural Networks (CNN):

- Layer computation uses convolution: $h' = \sigma(w * h)$
- w: Learned filters/kernels that slide over h.

Vanilla Spectral GNN

- The graph convolutional layer of Spectral CNN [*] is defined as

$$U^{\mathrm{T}} g = \begin{bmatrix} u_g^{\mathrm{T}} \\ u_{ng}^{\mathrm{T}} \end{bmatrix}$$

$$\begin{aligned} x * g &= f^{-1}\big(f(x) \cdot f(g)\big) \\ &= U(U^{\mathrm{T}} x \cdot U^{\mathrm{T}} g) \\ &= U g_\theta U^{\mathrm{T}} x \end{aligned}$$

$$H' = \sigma(U \Theta\, U^T H)$$

$$X = H^{(0)}$$

$$g_\theta = \Theta$$

$$\Theta = \begin{bmatrix} u_1^{\mathrm{T}} g & \cdots & \\ \vdots & \ddots & \vdots \\ & \cdots & u_n^{\mathrm{T}} g \end{bmatrix}$$

Θ is a diagonal matrix with learnable parameters.

[*] Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in Proc. of ICLR, 2014.

Limitation

- eigen-decomposition requires $O(n^3)$ computational complexity.

ChebNet

- Approximates the filter g_θ by Chebyshev [*] polynomials of the diagonal matrix of eigenvalues Λ.

[*] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” in Proc. of NIPS, 2016, pp. 3844–3852.

Chebyshev Polynomials of the First Kind

- $\cos(\theta) = \cos(\theta)$
- $\cos(2\theta) = 2\cos^2(\theta) - 1$
- $\cos(3\theta) = 4\cos^3(\theta) - 3\cos(\theta)$

Chebyshev Polynomials:

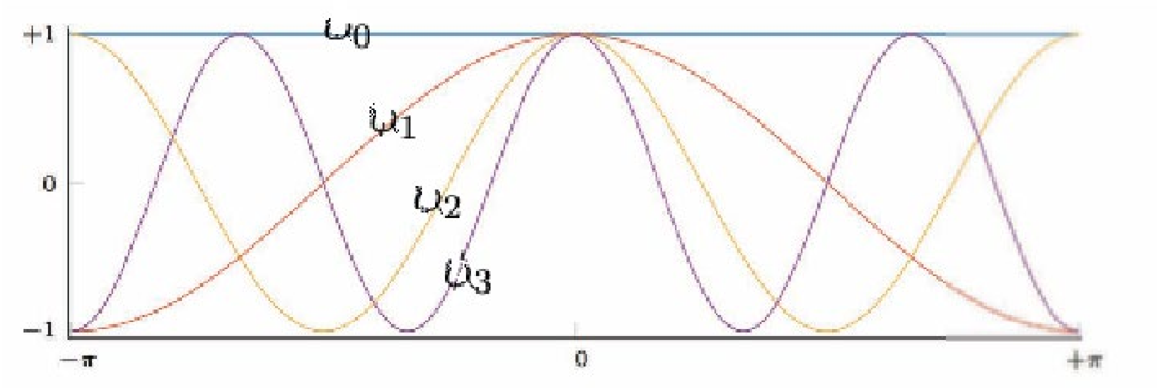
- By substituting $\cos(\theta) = x$, we obtain:
  - $T_1(x) = x$
  - $T_2(x) = 2x^2 - 1$
  - $T_3(x) = 4x^3 - 3x$

- $T_n(x)$ denotes the nth polynomial.
- Orthogonal on the interval $[-1, 1]$

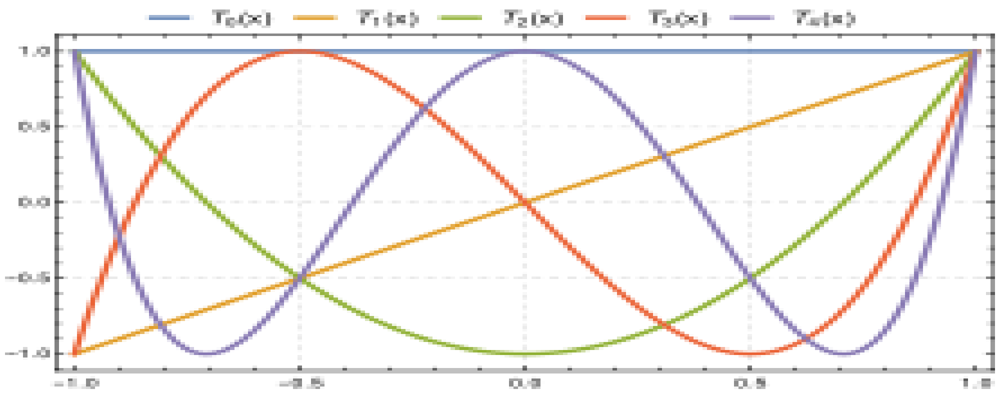
Chebyshev Polynomials of the First Kind

- $T_0(x) = 1$
- $T_1(x) = x$
- For $i \geq 2, T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x)$

$$\sum \alpha_i b_i$$



Fourier basis (eigenvectors of 1D Laplacian)



Chebyshev polynomials

ChebNet Graph Convolution

- $g_\theta = \sum_i \theta_i T_i(\tilde{\Lambda})$
- g_θ: Graph convolutional filter parameterized by θ.
- $\tilde{\Lambda}$: Scaled version of the eigenvalues of the Laplacian matrix.
- $\tilde{\Lambda} = \frac{2\Lambda}{\lambda_{max}} - I_n$
  - Normalizes the eigenvalues to fall within $[-1, 1]$.
- $x * g = U g_\theta U^T x$
- $x * g = \sum_i \theta_i U T_i(\tilde{\Lambda}) U^T x$

ChebNet Graph Convolution

- $x * g = U g_\theta U^T x$
- $x * g = \sum_i \theta_i U T_i(\tilde{\Lambda}) U^T x$

$$x * g_\theta = U(\sum_i \theta_i T_i(\tilde{\Lambda}))U^T x$$

- It is equivalent to:

$$x * g_\theta = \sum_{i=1}^{k} \theta_i T_i(\tilde{L}) x$$

ChebNet Graph Convolution

- $x * g = U g_\theta U^T x$
- $x * g = \sum_i \theta_i U T_i(\tilde{\Lambda}) U^T x$

$$x * g_\theta = U(\sum_i \theta_i T_i(\tilde{\Lambda}))U^T x$$

- It is equivalent to:

$$x * g_\theta = \sum_{i=1}^{k} \theta_i T_i(\tilde{L}) x$$

We can compute $\tilde{L}$ without the eigendecomposition of L.

The scaled Laplacian $\tilde{L}$ is

$$\tilde{L} = \frac{2L}{\lambda_{max}} - I_n$$