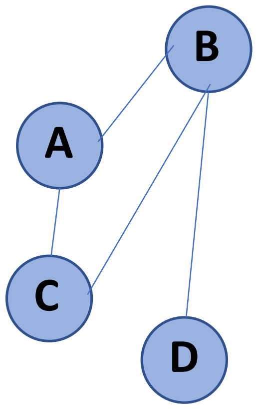
Graph Neural Networks

NIC LAB
Chen-Hao Hsiao

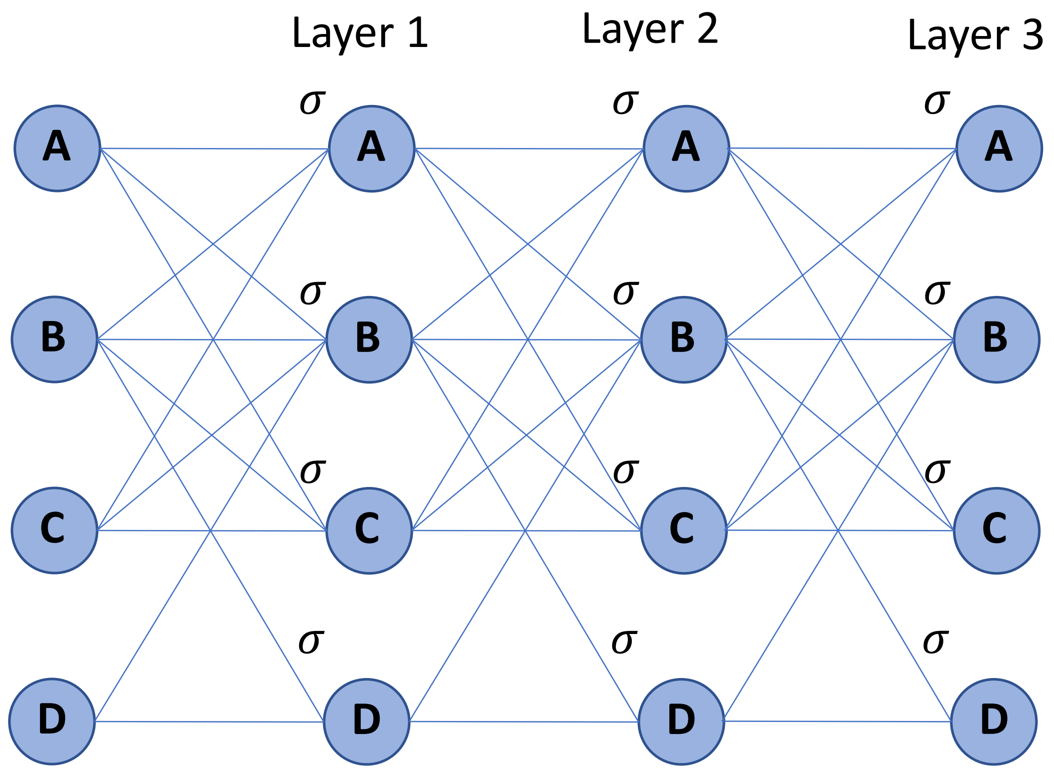
Outline

- Graph Neural Networks
- GCN
- GraphSAGE
- GN Block
- How powerful are GNNs

What is GNN?



Use graph structure as the NN architecture



What can GNN do?

- Node classification
- Graph classification → **Graph Representation Learning**
- Edge prediction

Graph Convolutional Network(GCN)

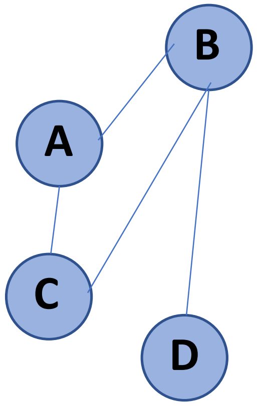
- Used for semi-supervised learning (Node classification)
- We have a undirected graph $G = (V, E)$ with $|V| = \mathrm{N}$, each node has a D-dimensional input feature, some nodes have C-dimensional labels
- Propagation rule:
  1. Input Feature X (a $\mathrm{N} \times \mathrm{D}$ matrix), depth L
  2. Set $H^{(0)} = X$
  3. Compute $H^{(l+1)} = \sigma(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$ where $\tilde{A} = A + I_N$,$\widetilde{D}_{ii} = \sum_j \tilde{A}_{ij}$
  4. Output $H^{(L)}$
- $W^{(l)}$is a $h_l \times h_{l+1}$ trainable matrix (hidden dimension) where $h_0 = D$
- Use cross entropy as loss function and do GD to train weight matrix W

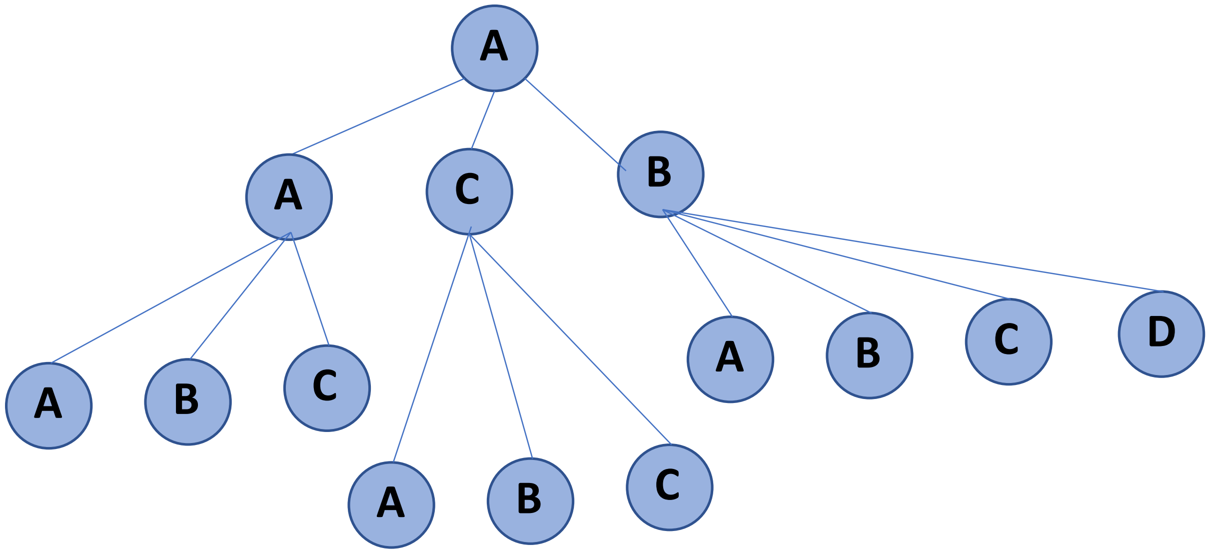
Convolution?

Propagation rule:

1. Input Feature X (a $\mathrm{N} \times \mathrm{D}$ matrix), depth L
2. Set $H^{(0)} = X$
3. Compute $H^{(l+1)} = \sigma(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)})$ where $\tilde{A} = A + I_N, \widetilde{D}_{ii} = \sum_j \tilde{A}_{ij}$
4. Output $H^{(L)}$

Aggregate with neighbors and itself





GraphSAGE Algorithm

initialize representations as features

K = "search depth"

aggregate information from neighbors

$\mathbf{h}^0_v \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;

for $k = 1...K$ **do**

  for $v \in \mathcal{V}$ **do**

    $\mathbf{h}^k_{\mathcal{N}(v)} \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}^{k-1}_u, \forall u \in \mathcal{N}(v)\})$;

    $\mathbf{h}^k_v \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}^{k-1}_v, \mathbf{h}^k_{\mathcal{N}(v)})\right)$

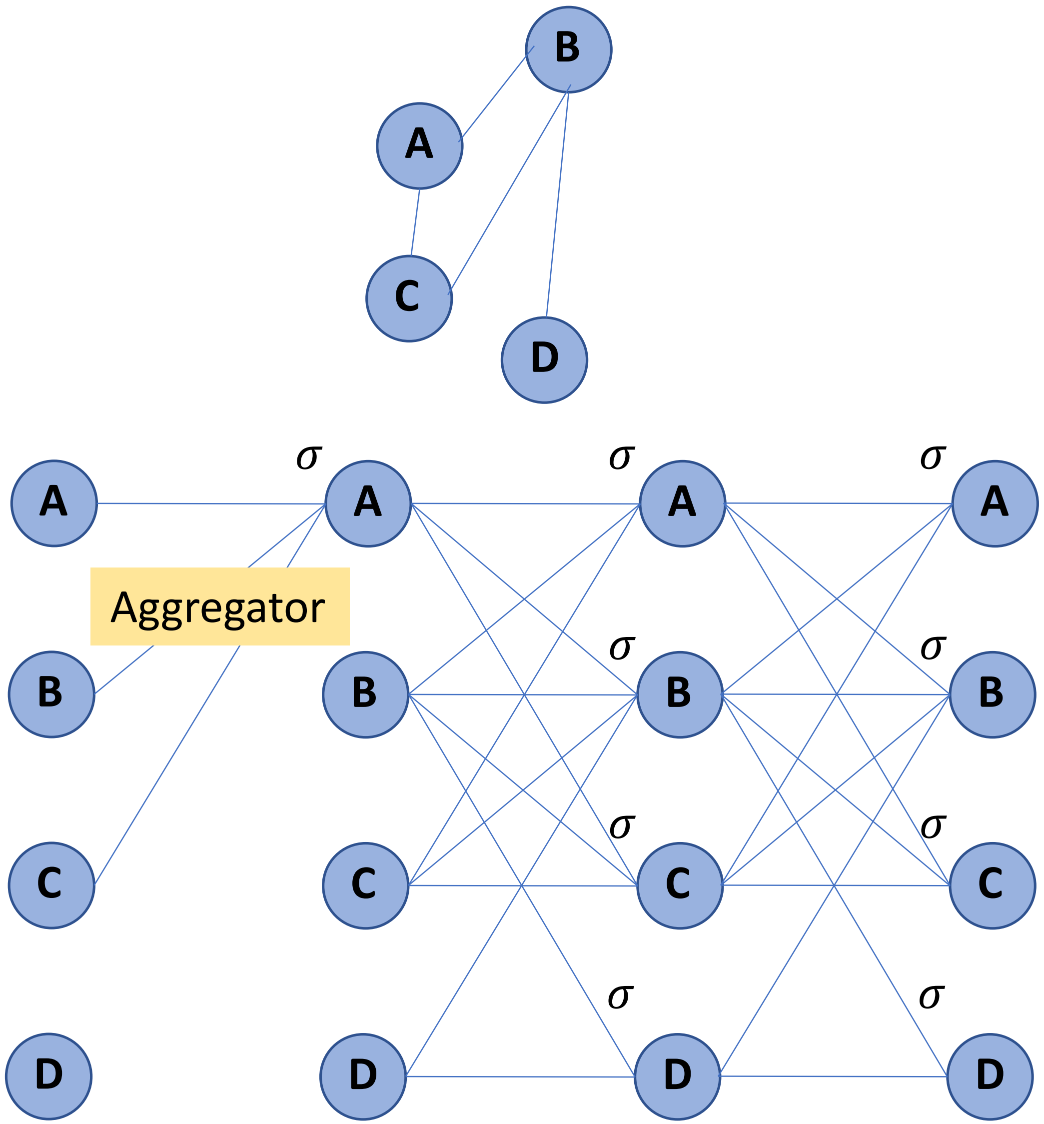
  end

  $\mathbf{h}^k_v \leftarrow \mathbf{h}^k_v / \|\mathbf{h}^k_v\|_2, \forall v \in \mathcal{V}$

end

$\mathbf{z}_v \leftarrow \mathbf{h}^K_v, \forall v \in \mathcal{V}$

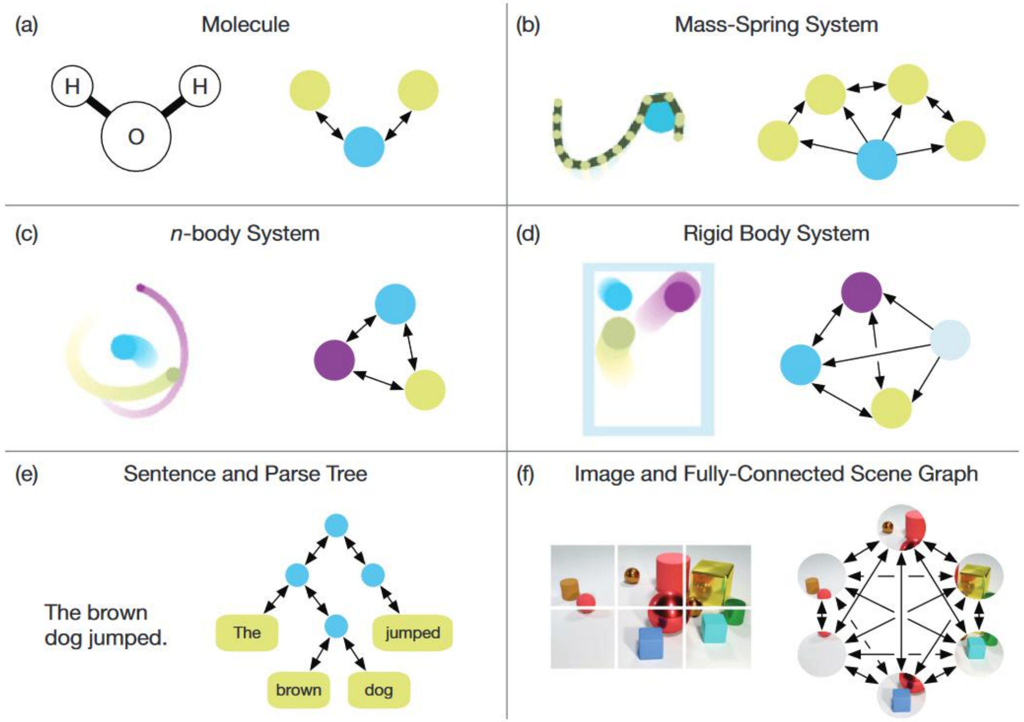
concatenate neighborhood info with current representation and propagate

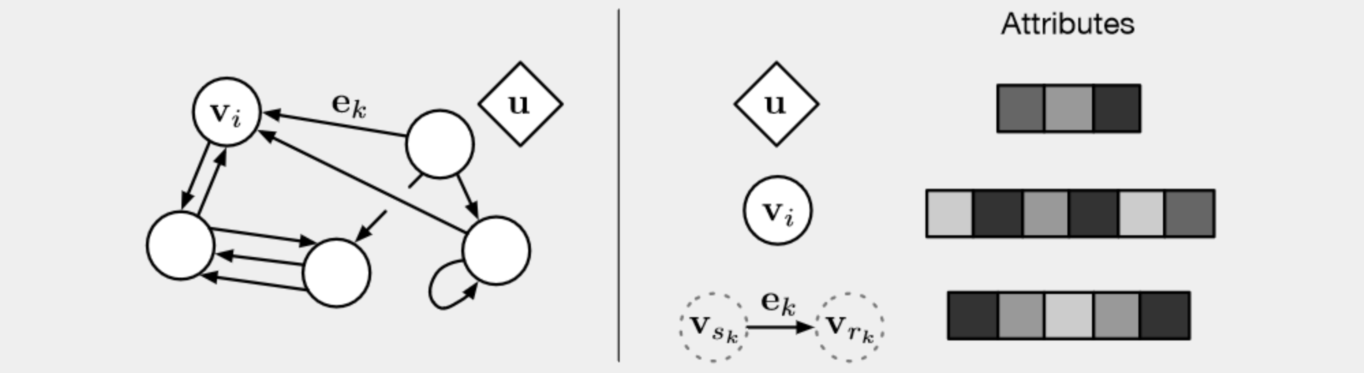


Main difference with GCN:
Sampling
Aggregation

GN-Definition of Graph

$$G = (\boldsymbol{u}, V, E)$$





Here we use "graph" to mean a directed, attributed multi-graph with a global attribute. In our terminology, a node is denoted as $\mathbf{v}_i$, an edge as $\mathbf{e}_k$, and the global attributes as $\mathbf{u}$. We also use s_k and r_k to indicate the indices of the sender and receiver nodes (see below), respectively, for edge k. To be more precise, we define these terms as:

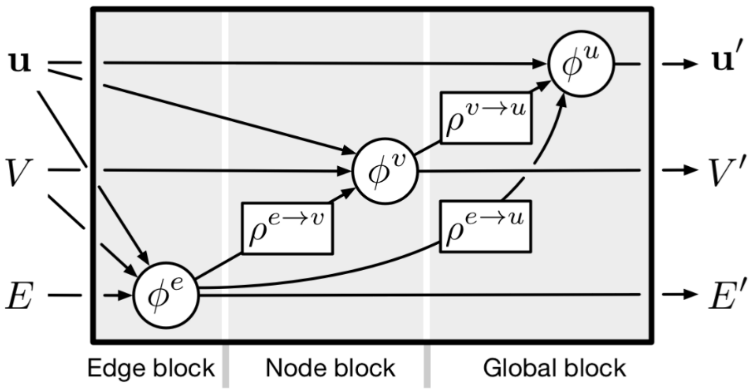
Directed : one-way edges, from a "sender" node to a "receiver" node.
Attribute : properties that can be encoded as a vector, set, or even another graph.
Attributed : edges and vertices have attributes associated with them.
Global attribute : a graph-level attribute.
Multi-graph : there can be more than one edge between vertices, including self-edges.

Figure 2 shows a variety of different types of graphs corresponding to real data that we may be interested in modeling, including physical systems, molecules, images, and text.

GN-Algorithm

Algorithm 1 Steps of computation in a full GN block.

function GRAPHNETWORK(E, V, $\mathbf{u}$)
  for $k \in \{1 \dots N^e\}$ **do**
    $\mathbf{e}'_k \leftarrow \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right)$ ▷ 1. Compute updated edge attributes
  end for
  for $i \in \{1 \dots N^n\}$ **do**
    let $E'_i = \{(\mathbf{e}'_k, r_k, s_k)\}_{r_k=i,\ k=1:N^e}$
    $\bar{\mathbf{e}}'_i \leftarrow \rho^{e\to v}\left(E'_i\right)$ ▷ 2. Aggregate edge attributes per node
    $\mathbf{v}'_i \leftarrow \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right)$ ▷ 3. Compute updated node attributes
  end for
  let $V' = \{\mathbf{v}'\}_{i=1:N^v}$
  let $E' = \{(\mathbf{e}'_k, r_k, s_k)\}_{k=1:N^e}$
  $\bar{\mathbf{e}}' \leftarrow \rho^{e\to u}\left(E'\right)$ ▷ 4. Aggregate edge attributes globally
  $\bar{\mathbf{v}}' \leftarrow \rho^{v\to u}\left(V'\right)$ ▷ 5. Aggregate node attributes globally
  $\mathbf{u}' \leftarrow \phi^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right)$ ▷ 6. Compute updated global attribute
  return $(E', V', \mathbf{u}')$
end function



Full GN block

Main Contribution:
GN library

How powerful are GNNs?

From previous examples, modern GNNs follow a neighborhood aggregation strategy:

$$a_v^{(k)} = \text{AGGREGATE}^{(k)}\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right), \quad h_v^{(k)} = \text{COMBINE}^{(k)}\left(h_v^{(k-1)}, a_v^{(k)}\right)$$

For graph classification task:

$$h_G = \text{READOUT}\left(\left\{h_v^{(K)} \mid v \in G\right\}\right)$$

Intuitively, the most powerful GNN maps two nodes to the same location only if they have identical subtree structures with identical features on the corresponding nodes.

Definition 1 (Multiset). A multiset is a generalized concept of a set that allows multiple instances for its elements. More formally, a multiset is a 2-tuple $X = (S, m)$ where S is the *underlying set* of X that is formed from its *distinct elements*, and $m : S \to \mathbb{N}_{\geq 1}$ gives the *multiplicity* of the elements.

Compare with WL isomorphism test

Lemma 2. *Let G_1 and G_2 be any non-isomorphic graphs. If a graph neural network $\mathcal{A} : \mathcal{G} \to \mathbb{R}^d$ following the neighborhood aggregation scheme maps G_1 and G_2 to different embeddings, the Weisfeiler-Lehman graph isomorphism test also decides G_1 and G_2 are not isomorphic.*

Theorem 3. *Let $\mathcal{A} : \mathcal{G} \to \mathbb{R}^d$ be a GNN following the neighborhood aggregation scheme. With sufficient iterations, $\mathcal{A}$ maps any graphs G_1 and G_2 that the Weisfeiler-Lehman test of isomorphism decides as non-isomorphic, to different embeddings if the following conditions hold:*

a) $\mathcal{A}$ aggregates and updates node features iteratively with

$$h_v^{(k)} = \phi\left(h_v^{(k-1)}, f\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right)\right) \quad \text{or} \quad h_v^{(k)} = f\left(\left\{h_v^{(k-1)}, h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right)$$

where the functions f, which operates on multisets, and ϕ are injective.

b) $\mathcal{A}$'s graph-level readout, which operates on the multiset of node features $\left\{h_v^{(k)}\right\}$, is injective.

Approximate the injective function using UAT

Lemma 4. *Assume $\mathcal{X}$ is countable. There exists a function $f : \mathcal{X} \to \mathbb{R}^n$ so that $h(X) = \sum_{x \in X} f(x)$ is unique for each finite multiset $X \subset \mathcal{X}$. Moreover, any multiset function g can be decomposed as $g(X) = \phi\left(\sum_{x \in X} f(x)\right)$ for some function ϕ.*

So they derived a new message passing method and it is at least good as WL

$$h_v^{(k)} = \mathrm{MLP}^{(k)}\left(h_v^{(k-1)} + \sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}\right)$$

Conclusions

- GNN model performs well on many tasks like node classification, edge prediction, graph classification…
- There are variants of GNN structure, but modern GNNs follow the aggregation scheme
- We can explore more GNN scheme in the future
- We can do some theoretical analysis on node classification (semi-supervised learning problem)

References

- Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." *arXiv preprint arXiv:1609.02907* (2016).
- Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." *Advances in Neural Information Processing Systems*. 2017.
- Peter W. Battaglia, et al. " Relational inductive biases, deep learning, and graph networks . " *arXiv preprint arXiv:1806.01261*
- Xu, Keyulu, et al. "How Powerful are Graph Neural Networks?." *arXiv preprint arXiv:1810.00826* (2018).

Appendix

- Spectral Graph Convolutions:

  Signal $x \in \mathbb{R}^N$ (a scalar for every node) on graph multiply $g_\theta = diag(\theta)$ parameterized by $\theta \in \mathbb{R}^N$ in the Fourier domain

  - First define Fourier transform on graph:

  Since the basis of continuous Fourier transform is the eigen function of the Laplace operator

  $\mathcal{F}\{f(t)\} = \hat{f}(\omega) = < f, e^{i\omega t} > = \int f(t)e^{-i\omega t}dt$

  We similarly define the basis of the Fourier transform on graph be the eigenvector of it's Laplacian

  $L = I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}} = U\Lambda U^T$

  Now for any function $f \in \mathbb{R}^N$ defined on the vertices of G :

  $\hat{f}(\lambda_l) = < f, u_l > = \sum_{i=1}^{N} f(i)u_l{}^*(i)$

$$\begin{pmatrix} \hat{f}(\lambda_1) \\ \hat{f}(\lambda_2) \\ \vdots \\ \hat{f}(\lambda_N) \end{pmatrix} = \begin{pmatrix} u_1(1) & u_1(2) & \dots & u_1(N) \\ u_2(1) & u_2(2) & \dots & u_2(N) \\ \vdots & \vdots & \ddots & \vdots \\ u_N(1) & u_N(2) & \dots & u_N(N) \end{pmatrix} \begin{pmatrix} f(1) \\ f(2) \\ \vdots \\ f(N) \end{pmatrix}$$

  $\hat{f} = U^T f$, also we can find out that $\mathcal{F}^{-1}\{\hat{f}\} = U\hat{f}$

Appendix

- Spectral Graph Convolutions:

  Signal $x \in \mathbb{R}^N$ (a scalar for every node) on graph multiply $g_\theta = diag(\theta)$ parameterized by $\theta \in \mathbb{R}^N$ in the Fourier domain

  - Now define convolution:

  $f * h = \mathcal{F}^{-1}\{\hat{f}\hat{h}\}$, let $\widehat{H} = diag(\hat{h}(\lambda_1), \hat{h}(\lambda_2), \ldots, \hat{h}(\lambda_N))$, then $\mathcal{F}^{-1}\{\hat{f}\hat{h}\} = U\widehat{H}U^T f$

  So $x * g = U g_\theta U^T f$, if we view g_θ as $g_\theta(\Lambda)$ and approximate it with truncated expansion in terms of Chebyshev polynomials up to K-th order, $g_{\theta'}(\Lambda) = \sum_{i=1}^{K} \theta'_i T_i(\tilde{\Lambda})$ with

  $\tilde{\Lambda} = \frac{2}{\lambda_{max}}\Lambda - I_N$, we have $x * g \approx \sum_{i=1}^{K} \theta'_i T_i(\tilde{L})$ with $\tilde{L} = \frac{2}{\lambda_{max}} L - I_N$

  In GCN, they use the first 2 terms of approximation and assume $\lambda_{max} \approx 2$, $\alpha = \theta'_0 = -\theta'_1$

  $x * g \approx \alpha\left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right) x$

  Do renormalization trick $I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \rightarrow \widetilde{D}^{-\frac{1}{2}} \tilde{A} \widetilde{D}^{-\frac{1}{2}}$, where $\tilde{A} = A + I_N$, $\widetilde{D}_{ii} = \sum_j \tilde{A}_{ij}$