

Graph Neural Networks (GNN)

Wen-Hsiao Peng

National Yang Ming Chiao Tung University (NYCU), Taiwan

July 7, 2021

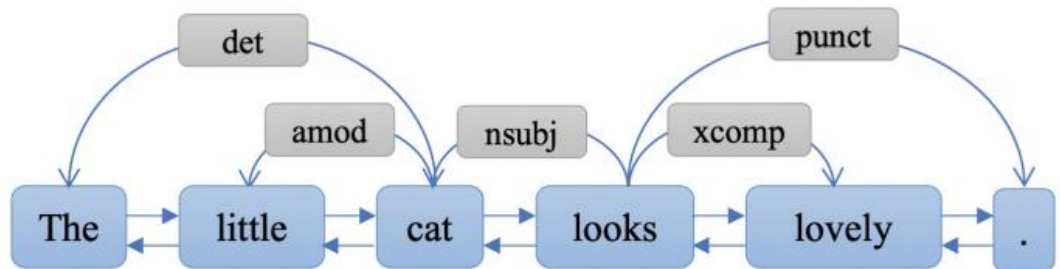
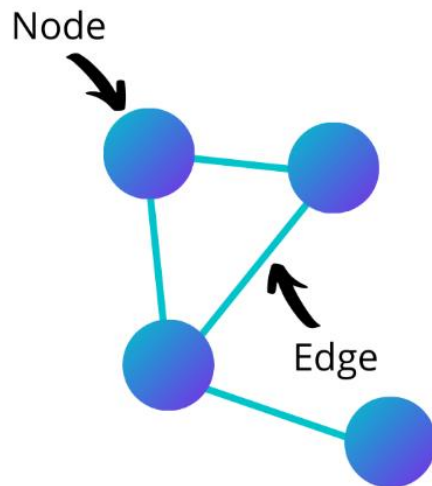
Outline

- Introduction to Graph
- Graph Signal Processing
- Graph Neural Networks (GNNs)
 - Spectral-based Convolutional Graph Neural Networks
 - Spatial-based Convolutional Graph Neural Networks
 - Recurrent Graph Neural Networks (RecGNNs)
 - Graph Autoencoders (GAEs)
 - Spatial-temporal Graph Neural Networks (STGNNs)
- Applications

Introduction to Graph

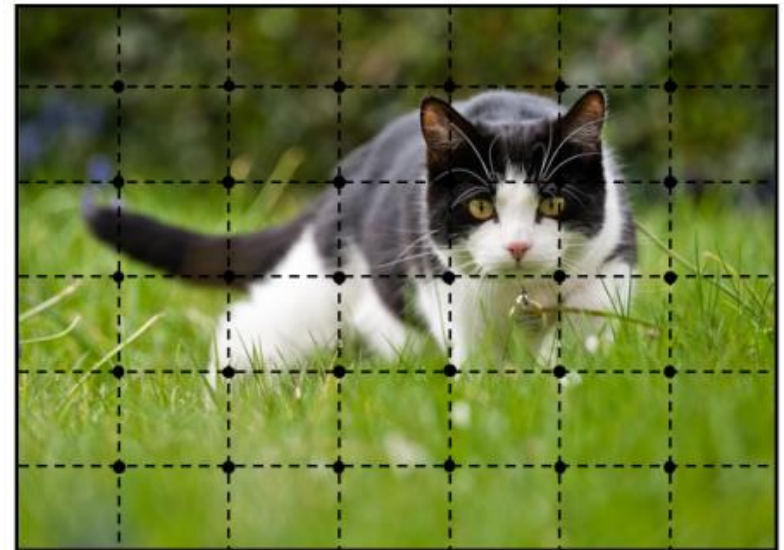
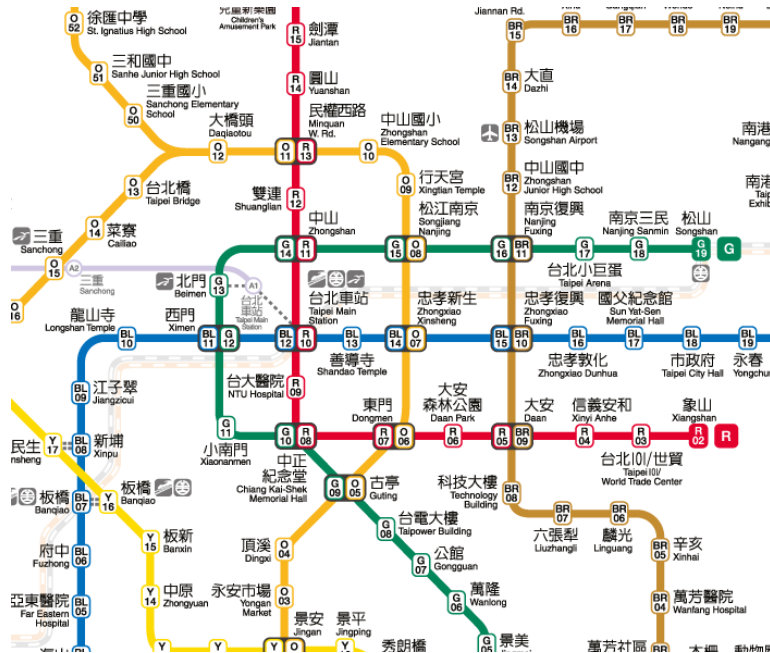
Definition of Graph

- A collection of nodes \mathcal{N} and edges \mathcal{E}
 - **Nodes** contain **data** of interest
 - **Edges** specify **structure**, i.e. how data are related



Words (**nodes**) in a sentence (**graph**)

Graph: More Examples



Roads (**nodes**) in a map (**graph**) Pixels (**nodes**) in an image (**graph**)

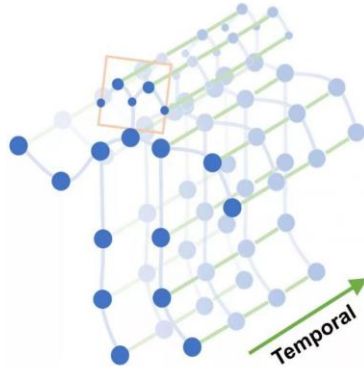
Graph structure can be **irregular**!

Why Graph?

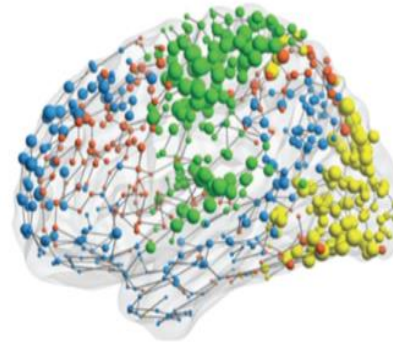
- Many real-world data come in the form of **graphs**, where **data** (nodes) are related by a **network** (edges)



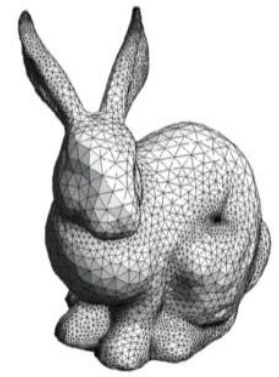
Social networks



Skeleton



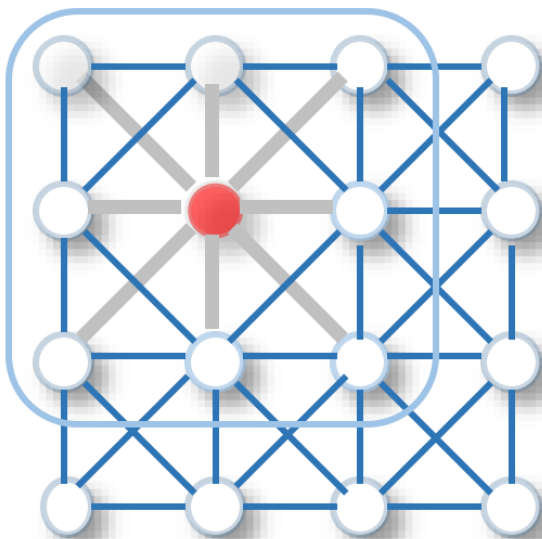
Functional networks



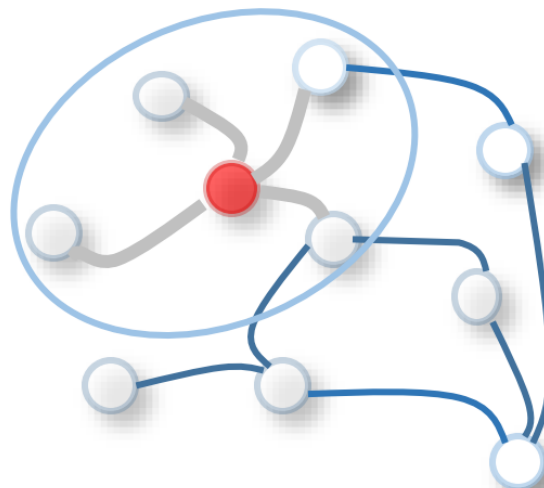
3D shapes

2-D vs. Graph Convolution

Neighbors of a node are
ordered and **fixed** in size



Neighbors of a node are
unordered and **variable** in size



Applying 2-D convolution to graph data becomes difficult!

2-D Convolution

- **Locality** – kernels with a **compact** support
- **Stationarity** – **space-invariant** kernels
- **Multi-scale** – **convolution** with **stride** and **pooling**

Suitable for data (e.g. images, videos) that have these properties!

Graph Convolution

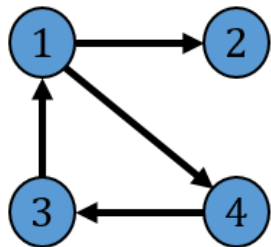
- How to define the notion of **convolution on graph**?
 - Graph signal processing (spectral graph theory)
- How to **downsample and pool graph data**?
 - Clustering on graph (graph theory)

Outline

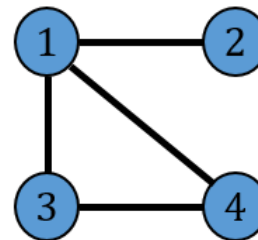
- Introduction to Graph
- **Graph Signal Processing**
- Graph Neural Networks (GNNs)
 - Spectral-based Convolutional Graph Neural Networks
 - Spatial-based Convolutional Graph Neural Networks
 - Recurrent Graph Neural Networks (RecGNNs)
 - Graph Autoencoders (GAEs)
 - Spatial-temporal Graph Neural Networks (STGNNs)
- Applications

Terminology

- A graph is represented as $G = (V, E)$ where
 - V is the set of nodes, $\{v_1, v_2, \dots, v_n\}$
 - E is the set of edges, $\{e_{ij} \mid e_{ij} = (v_i, v_j)\}$
- e_{ij} : an edge pointing from v_j to v_i
- Undirected graph: $e_{ij} \in E \rightarrow e_{ji} \in E$
- Neighborhood $N(\cdot)$ of node v : $\{u \in V \mid (u, v) \in E\}$



Directed Graph



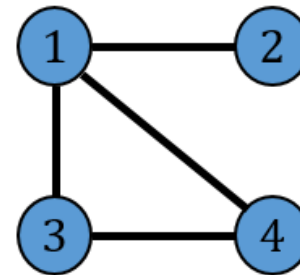
Undirected Graph

Adjacent and Degree Matrices

- Adjacency matrix $A \in R^{n \times n}$

$A_{ij} = 1$ if $e_{ij} \in E$, $A_{ij} = 0$ otherwise

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



- Degree matrix $D \in R^{n \times n}$

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

$$D_{ii} = \sum_{j=1}^n A_{ij}$$

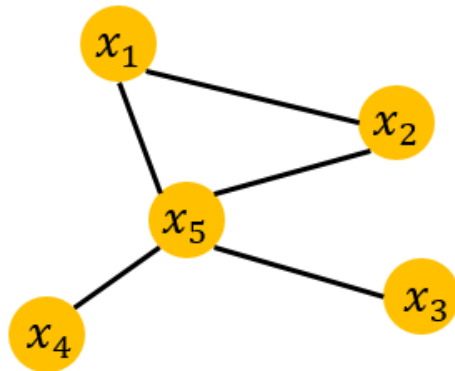
Graph Fourier Transform

- Graph convolution by **graph Fourier transform**.

$$\mathbf{x} *_G \mathbf{g} = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{x}) \odot \mathcal{F}(\mathbf{g}))$$

- $\mathbf{x} \in \mathbb{R}^n$ – nodes of graph
- $\mathbf{g} \in \mathbb{R}^n$ – filter
- $*_G$ – graph convolution
- \mathcal{F} – graph Fourier transform
- \mathcal{F}^{-1} – inverse graph Fourier transform

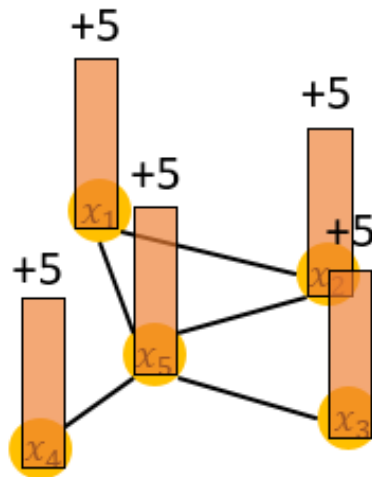
Graph Frequency



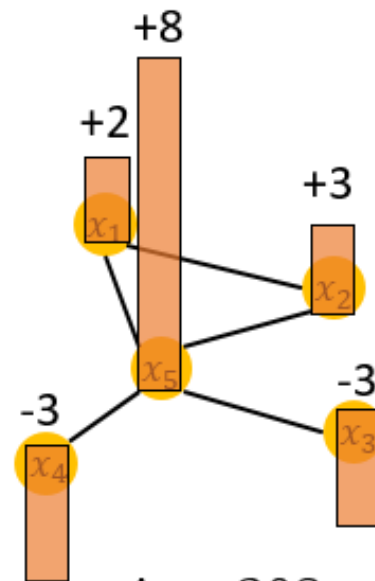
$$\Delta_G = (x_1 - x_2)^2 + (x_1 - x_5)^2 + \dots$$

$$= \sum_{(u,v) \in E} (x_u - x_v)^2$$

Example:



$$\Delta_G = 0$$

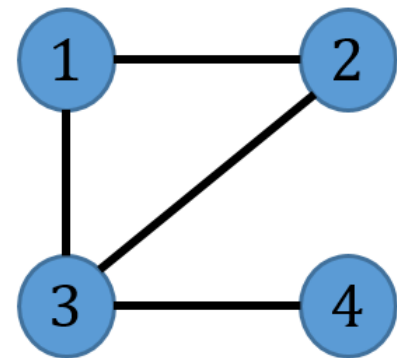


$$\Delta_G = 308$$

Graph Laplacian

- Laplacian matrix $L = D - A$ for undirected graphs

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



$$L = \begin{bmatrix} 2 & -1 & -1 & 0 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix} \Rightarrow \frac{1}{2} \mathbf{x}^T L \mathbf{x} = \sum_{(u,v) \in E} (x_u - x_v)^2$$

Eigen-decomposition of Graph Laplacian

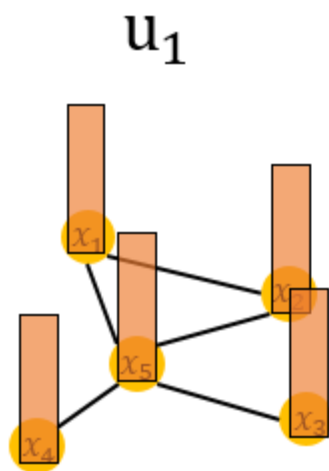
- L is **real, symmetric**, and **positive semidefinite**
 - $L = U\Lambda U^T$
 - $U = [u_1, u_2, \dots, u_n]$, $u_i \in R^n$ and $u_i^T u_j = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$
 - $\Lambda = \text{diag}([\lambda_1, \lambda_2, \dots, \lambda_n])$, $\lambda_i \in R$ and $\lambda_1 < \lambda_2 < \dots < \lambda_n$
- Graph frequency of eigenvectors

$$u_i^T L u_i = u_i^T \lambda_i u_i = \lambda_i u_i^T u_i = \lambda_i$$

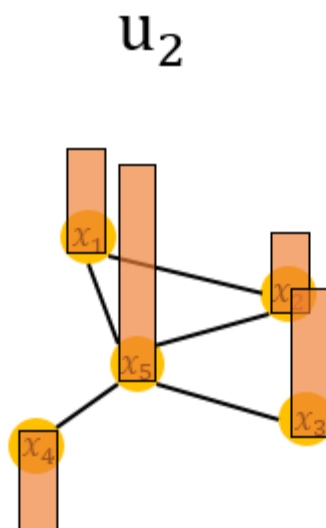
Graph Basis: Eigenvectors of L

$$\mathbf{u}_i^T L \mathbf{u}_i = \mathbf{u}_i^T \lambda_i \mathbf{u}_i = \lambda_i \mathbf{u}_i^T \mathbf{u}_i = \lambda_i$$

$$\lambda_1 < \lambda_2 < \dots < \lambda_n$$

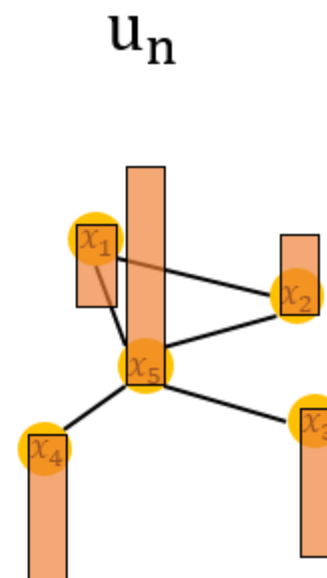


$$\Delta_G = \lambda_1 = 0$$



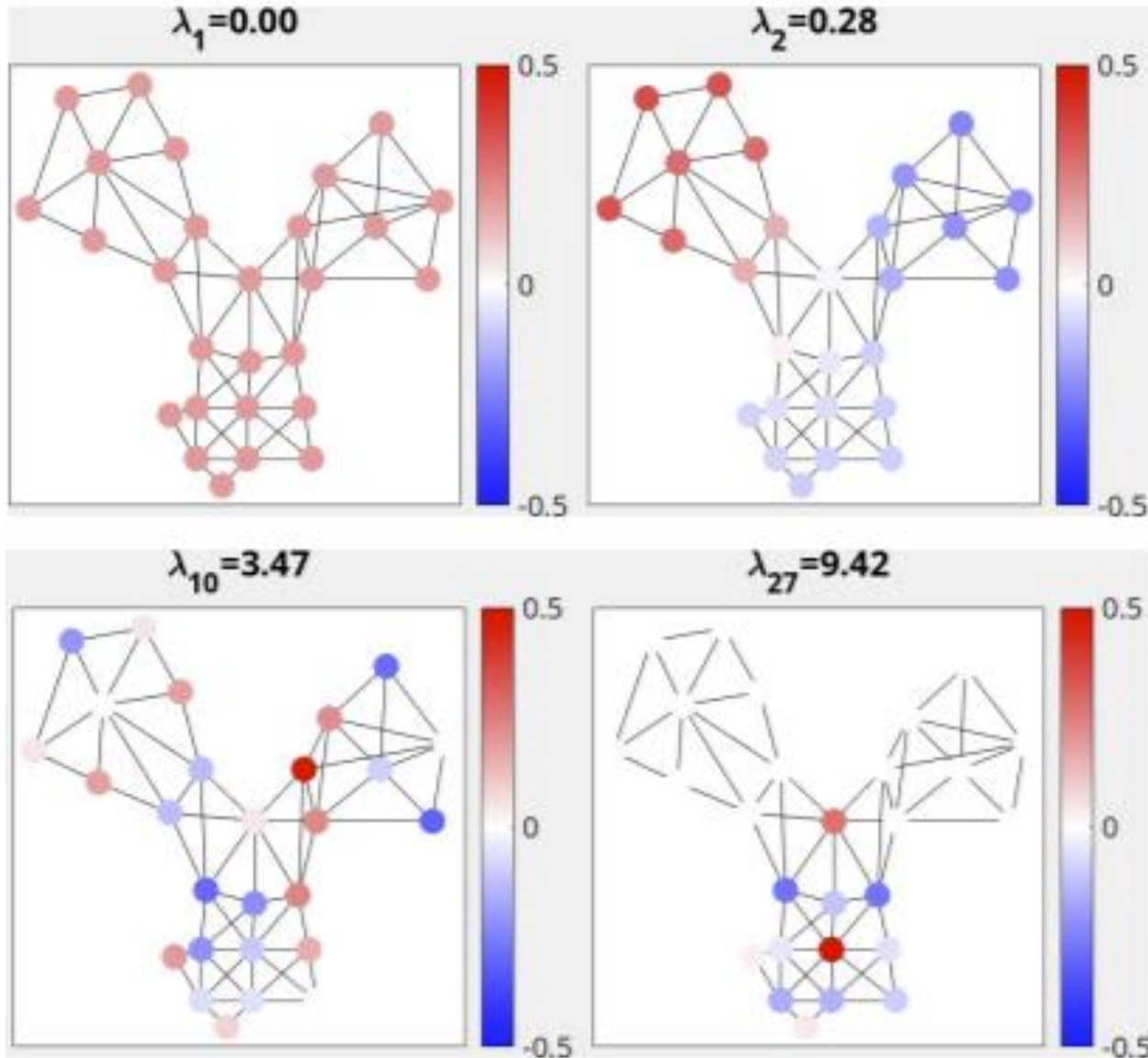
$$\Delta_G = \lambda_2$$

...



$$\Delta_G = \lambda_n$$

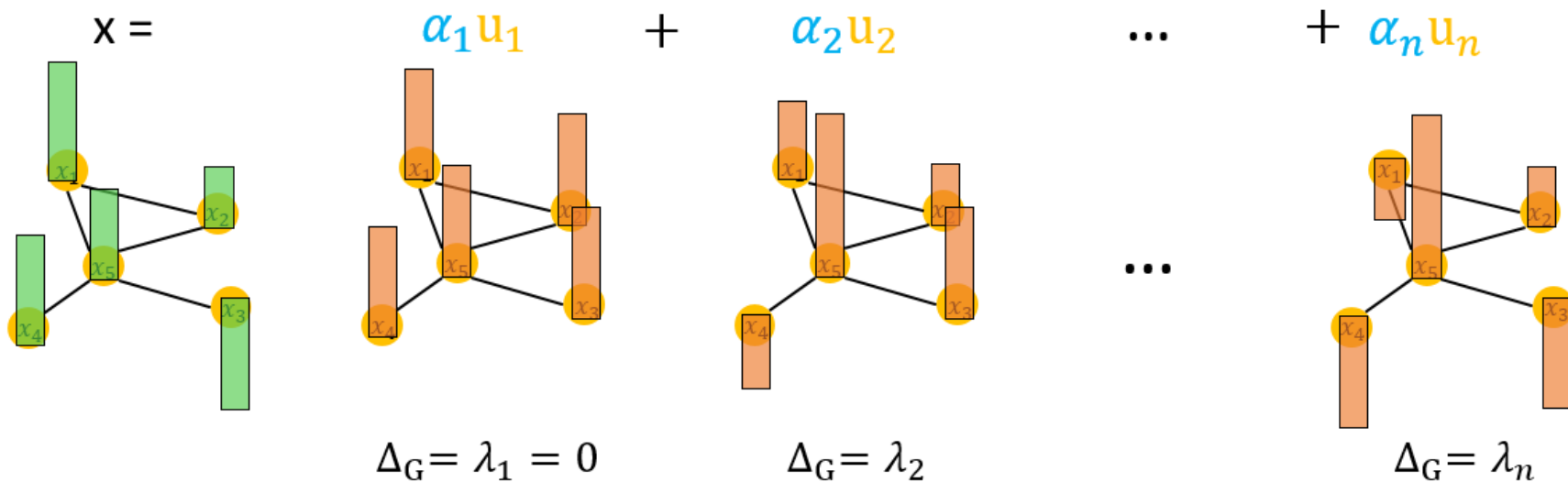
Visualization of Eigenvectors



Graph Transform Coefficients

$$\mathbf{x} = \mathbf{I}\mathbf{x} = \underbrace{\mathbf{U}\mathbf{U}^T}_{\text{Graph basis}} \mathbf{x}$$

Graph basis Graph Transform Coeff.



Graph Convolution

- Graph Fourier transform

$$\mathcal{F}(\mathbf{x}) = \hat{\mathbf{x}} = \mathbf{U}^T \mathbf{x} \quad \text{and} \quad \mathcal{F}^{-1}(\hat{\mathbf{x}}) = \mathbf{U} \hat{\mathbf{x}}$$

- Spectral graph convolution

$$\begin{aligned} \mathbf{x} *_G \mathbf{g} &= \mathcal{F}^{-1}(\mathcal{F}(\mathbf{x}) \odot \mathcal{F}(\mathbf{g})) \\ &= \mathbf{U}(\mathbf{U}^T \mathbf{x} \odot \mathbf{U}^T \mathbf{g}) \\ &= \mathbf{U} \mathbf{g}_\theta \mathbf{U}^T \mathbf{x} \end{aligned}$$

where

$$\mathbf{g}_\theta = \text{diag}(\mathbf{U}^T \mathbf{g}) \text{ is learnable}$$

Spectral-based Graph Convolution

- Inspired theoretically by **graph signal processing**
- Limited to operate on **undirected graphs**
- **Prohibitively expensive** due to **eigenvector decomposition**
- **Poor generalization** to new graphs (i.e. eigenvectors are completely different for different graphs)

Outline

- Introduction to Graph
- Graph Signal Processing
- **Graph Neural Networks (GNNs)**
 - Spectral-based Convolutional Graph Neural Networks
 - **Spatial-based Convolutional Graph Neural Networks**
 - Recurrent Graph Neural Networks (RecGNNs)
 - Graph Autoencoders (GAEs)
 - Spatial-temporal Graph Neural Networks (STGNNs)
- Applications

ChebNet: Chebyshev Spectral GNN (1/2)

- Graph convolution

$$\mathbf{X} *_G \mathbf{g} = \mathbf{U} \mathbf{g}_\theta \mathbf{U}^T \mathbf{X}$$

- Approximates \mathbf{g}_θ by Chebyshev polynomials

$$\mathbf{g}_\theta \approx \sum_{i=0}^K \theta_i T_i(\tilde{\Lambda})$$

- $\tilde{\Lambda} = 2 \Lambda / \lambda_{\max} - \mathbf{I}$
- θ_i are learnable
- $T_i(\tilde{\Lambda}) = 2\tilde{\Lambda}T_{i-1}(\tilde{\Lambda}) - T_{i-2}(\tilde{\Lambda})$,
- $T_0(\tilde{\Lambda}) = \mathbf{I}$ and $T_1(\tilde{\Lambda}) = \tilde{\Lambda}$

ChebNet: Chebyshev Spectral GNN (2/2)

- It can be shown that

$$\mathbf{x} *_G \mathbf{g}_\theta = \mathbf{U} \left(\sum_{i=0}^K \theta_i T_i(\tilde{\Lambda}) \right) \mathbf{U}^T \mathbf{x} = \sum_{i=0}^K \theta_i T_i(\tilde{\mathbf{L}}) \mathbf{x}$$

- $\tilde{\mathbf{L}} = 2 \mathbf{L} / \lambda_{\max} - \mathbf{I}$
- θ_i are learnable
- $T_i(\tilde{\mathbf{L}}) \mathbf{x}$ is localized in space (local feature extraction)
- $O(Kn) \Rightarrow O(n)$

Graph Convolutional Network (GCN)

- A **first-order approximation of ChebNet** by assuming $K = 1$, $\lambda_{max} = 2$, $\theta = \theta_0 = -\theta_1$

$$\text{ChebNet: } \mathbf{x} *_G \mathbf{g}_\theta = \sum_{i=0}^K \theta_i T_i(\tilde{\mathbf{L}}) \mathbf{x}$$

$$\text{GCN: } \mathbf{x} *_G \mathbf{g}_\theta = \theta \left(\mathbf{I}_n + \underbrace{\mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}}_{\bar{\mathbf{A}}} \right) \mathbf{x}$$

- $\bar{\mathbf{A}} \mathbf{x}$: message-passing and aggregation

Multi-channel GCN (1/2)

- GCN with **multi-channel input/output** and **non-linear activation** f

$$H = f(\bar{A}X\Theta)$$

- $X = [\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_d] \in R^{n \times d}$ are d input channels on graph
- $\Theta = [\theta_1, \theta_2, \dots, \theta_s] \in R^{d \times s}$ with θ_i being 1×1 Conv.
- $H = [\tilde{h}_1, \tilde{h}_2, \dots, \tilde{h}_s] \in R^{n \times s}$ are s output channels on graph

Multi-channel GCN (2/2)

- Node output h_v is a **weighted sum** of node **input** \tilde{x}_v and **its neighbors** $\tilde{x}_u, u \in N(v)$

$$H^T = f(\Theta^T X^T \bar{A})$$

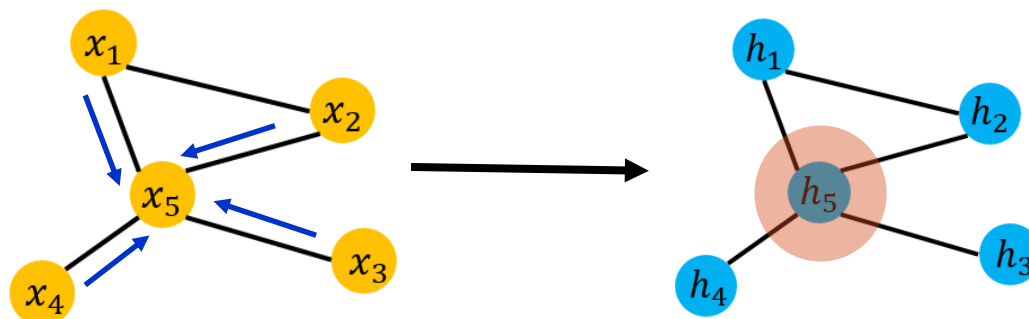
$$\underbrace{h_v}_{\text{Output node } v} = f(\Theta^T \underbrace{\sum_{u \in \{N(v) \cup v\}} \bar{A}_{v,u} x_u}_{\text{Message-passing and aggregation}})$$

- $x_u \in \mathbb{R}^d$ is node input u
- $h_v \in \mathbb{R}^s$ is node output v

Visualization of Multi-channel GCN

$$\underline{h_v} = f(\Theta^T \underbrace{\sum_{u \in \{N(v) \cup v\}} \bar{A}_{v,u} \tilde{x}_u}_{\text{Message-passing and aggregation}})$$

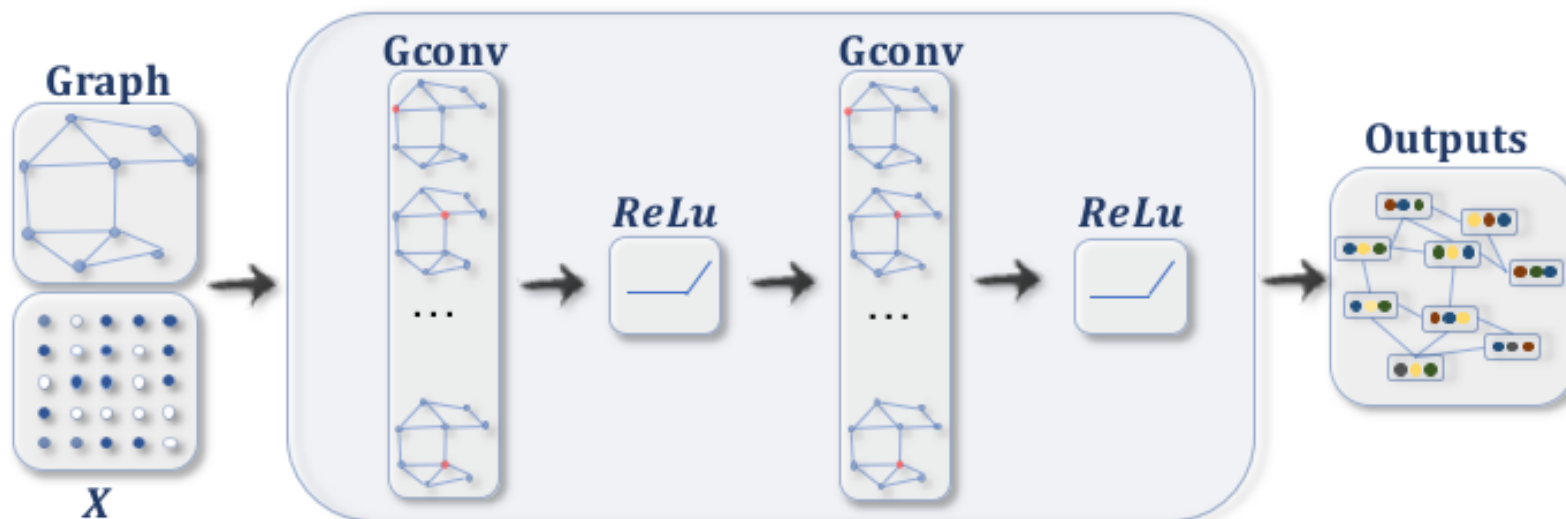
Output node v



Message-passing and aggregation

Message-passing and aggregation is localized on graph!

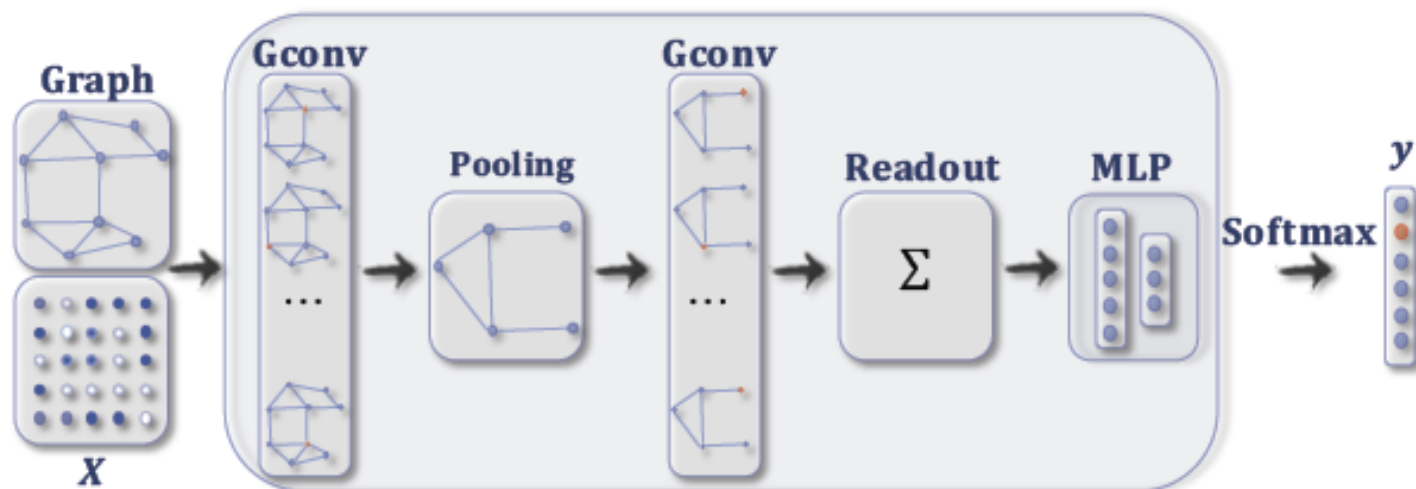
Composition of GCN Layers



- Each **node input** is a **vector** $x \in R^d$.
- **Graph input** is a **matrix** $X \in R^{n \times d}$, n is the number of nodes.

Receptive field increases with the number of layers!

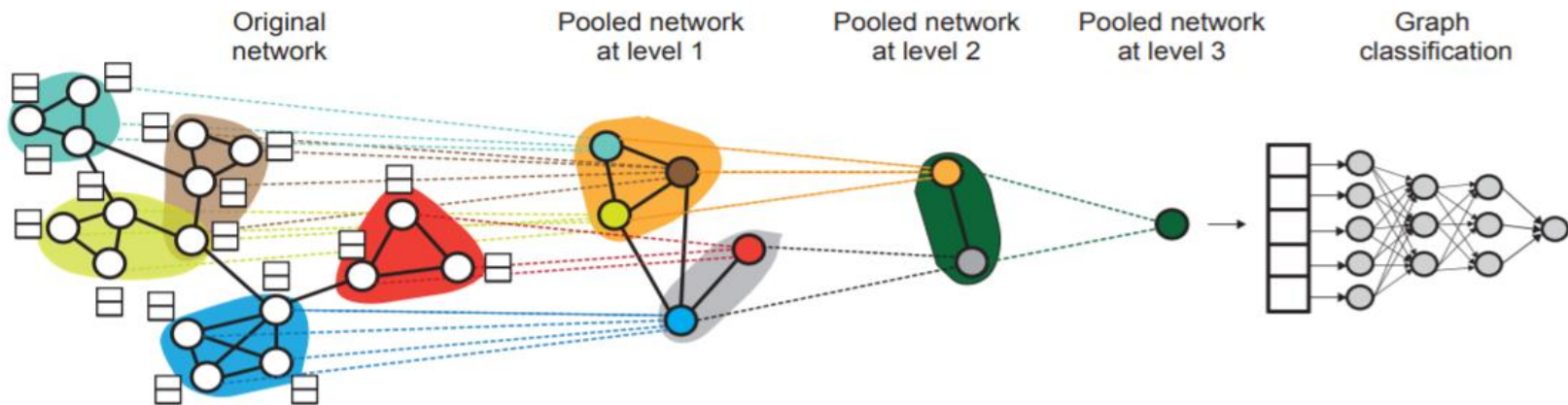
GCN with Pooling and Readout



- **Pooling** coarsens a graph into sub-graphs with node representations capturing **higher graph-level representations**
- **Readout** summarizes the final graph by **sum/mean of latent representations** of sub-graphs

Graph Pooling

- Pooling can be achieved by **clustering** the nodes based on **structure information only**



DiffPool

- To learn **assignment matrix** S for **soft clustering** by involving **both feature and structure** info. with GCN

$$S^{(l)} = \text{softmax}(\text{GNN}_{l,\text{pool}}(A^{(l)}, X^{(l)})) \in R^{n_l \times n_{l+1}}$$

- n_l is the number of input nodes at layer l
- n_{l+1} is the number of output clusters for layer $l + 1$
- Softmax is performed row-wise on $S^{(l)}$
- $S_{ij}^{(l)}$ indicates the probability of node i being clustered to j

Pooling with Assignment Matrix

- GNN for feature extraction and learning assignment matrix

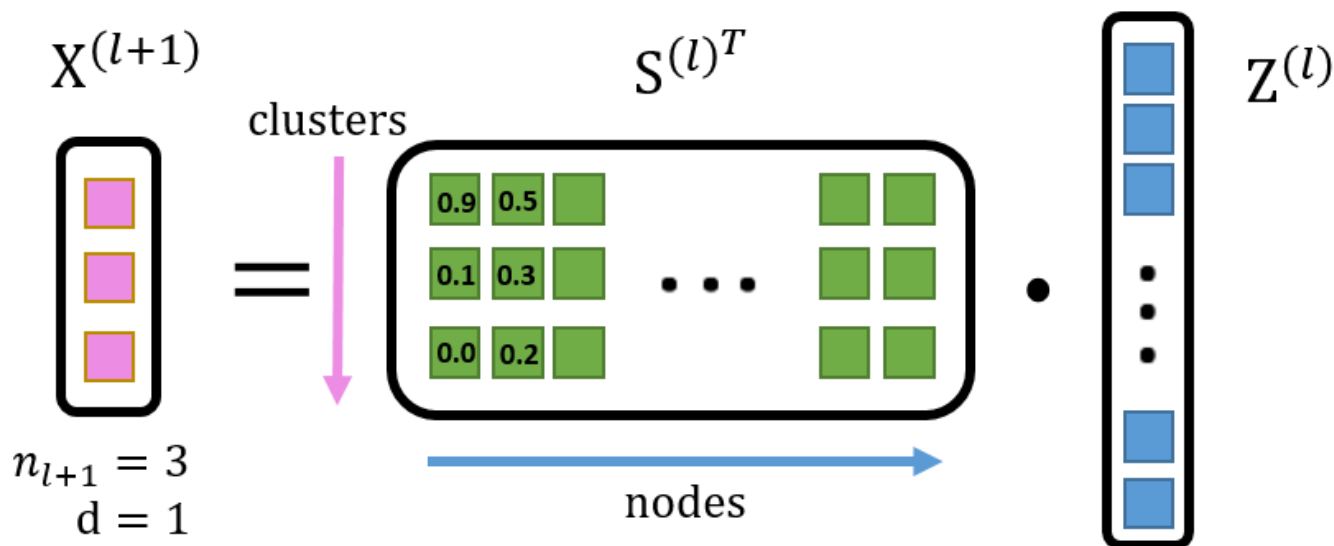
$$Z^{(l)} = \text{GNN}_{l,\text{embed}}(A^{(l)}, X^{(l)}) \in R^{n_l \times d}$$

$$S^{(l)} = \text{softmax}(\text{GNN}_{l,\text{pool}}(A^{(l)}, X^{(l)})) \in R^{n_l \times n_{l+1}}$$

- Pooling

$$X^{(l+1)} = S^{(l)T} Z^{(l)} \in R^{n_{l+1} \times d} \quad \# \text{ Pooled data}$$

$$A^{(l+1)} = S^{(l)T} A^{(l)} S^{(l)} \in R^{n_{l+1} \times n_{l+1}} \quad \# \text{ Sub-graph adjacency matrix}$$



Outline

- Introduction to Graph
- Graph Signal Processing
- **Graph Neural Networks (GNNs)**
 - Spectral-based Convolutional Graph Neural Networks
 - **Spatial-based Convolutional Graph Neural Networks**
 - Recurrent Graph Neural Networks (RecGNNs)
 - Graph Autoencoders (GAEs)
 - Spatial-temporal Graph Neural Networks (STGNNs)
- Applications

Spatial-based Graph Convolution

- Define **spatial graph convolution** based on **node's spatial relations**
- Convolve a node's representation with its neighbors' to update the node representation
- The idea is the same as **message passing + aggregation**

Neural Network for Graphs

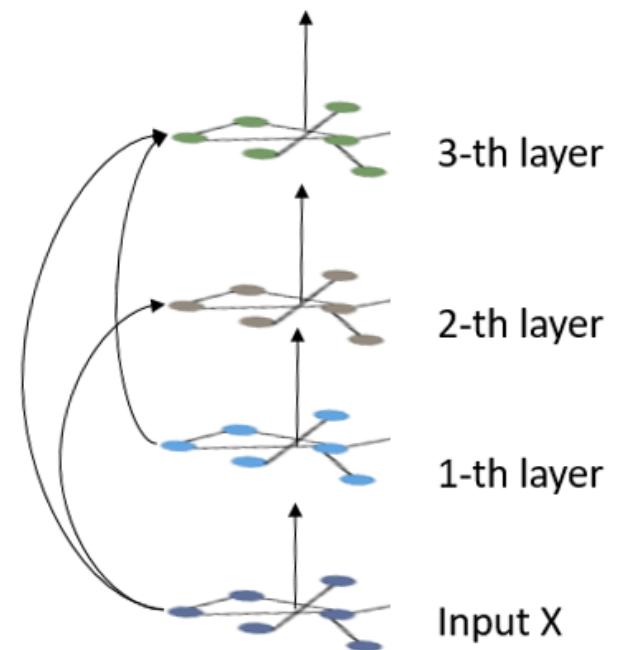
- Update node v at layer k with its input x_v and its neighbors $h_u^{(i)}$, $u \in N(v)$ from all the previous hidden layers $i, i = 1, 2, \dots, k - 1$

$$h_v^{(k)} = f(\mathbf{W}^{(k)T} \mathbf{x}_v + \sum_{u \in N(v)} \sum_{i=1}^{k-1} \Theta^{(ki)T} h_u^{(i)})$$

f - activation function

$h_v^{(k)}$ - the k -th latent representation of node v

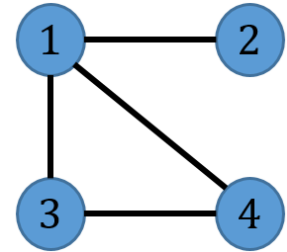
$\Theta^{(ki)}$ - the 1×1 conv. of the k -th layer from the i -th layer



Diffusion CNN (1/2)

- Graph convolution as a **diffusion process**

$$\mathbf{H}^{(k)} = f(\mathbf{W}^{(k)} \odot \mathbf{P}^k \mathbf{X})$$



- From a node's perspective \rightarrow summing information from its neighbors with the transition matrix **P**

$$\mathbf{P} = \mathbf{D}^{-1}\mathbf{A} \in \mathbb{R}^{n \times n}$$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}, \mathbf{D} = \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}, \mathbf{P} = \begin{bmatrix} 0 & 0.33 & 0.33 & 0.33 \\ 1 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0.5 \\ 0.5 & 0 & 0.5 & 0 \end{bmatrix}$$

Diffusion CNN (2/2)

- Concatenate multi-step propagation results with **variable-size receptive fields**

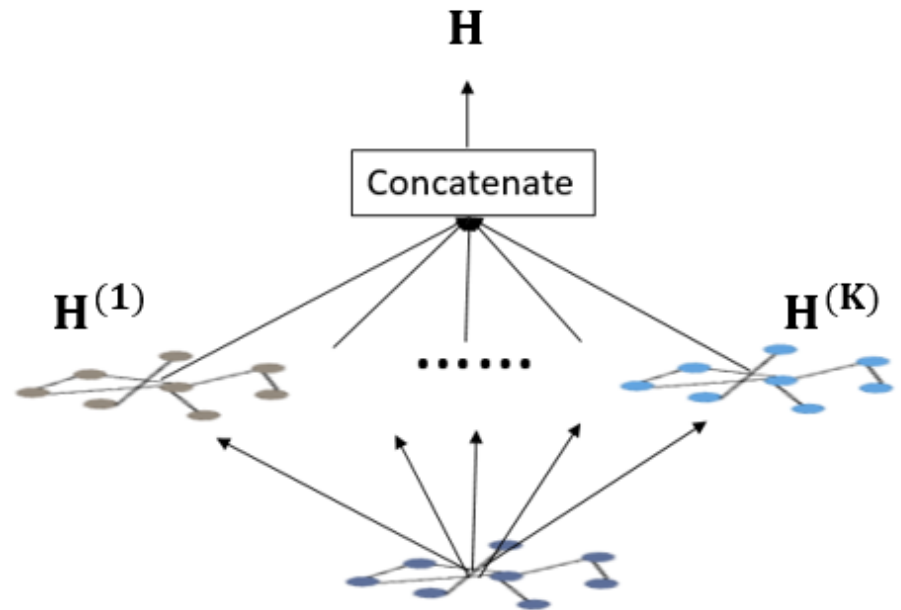
$$\mathbf{H}^{(k)} = f(\mathbf{W}^{(k)} \odot \mathbf{P}^k \mathbf{X})$$

$\mathbf{P}^1 \Rightarrow$ diffuse 1 time

$\mathbf{P}^2 \Rightarrow$ diffuse 2 times

...

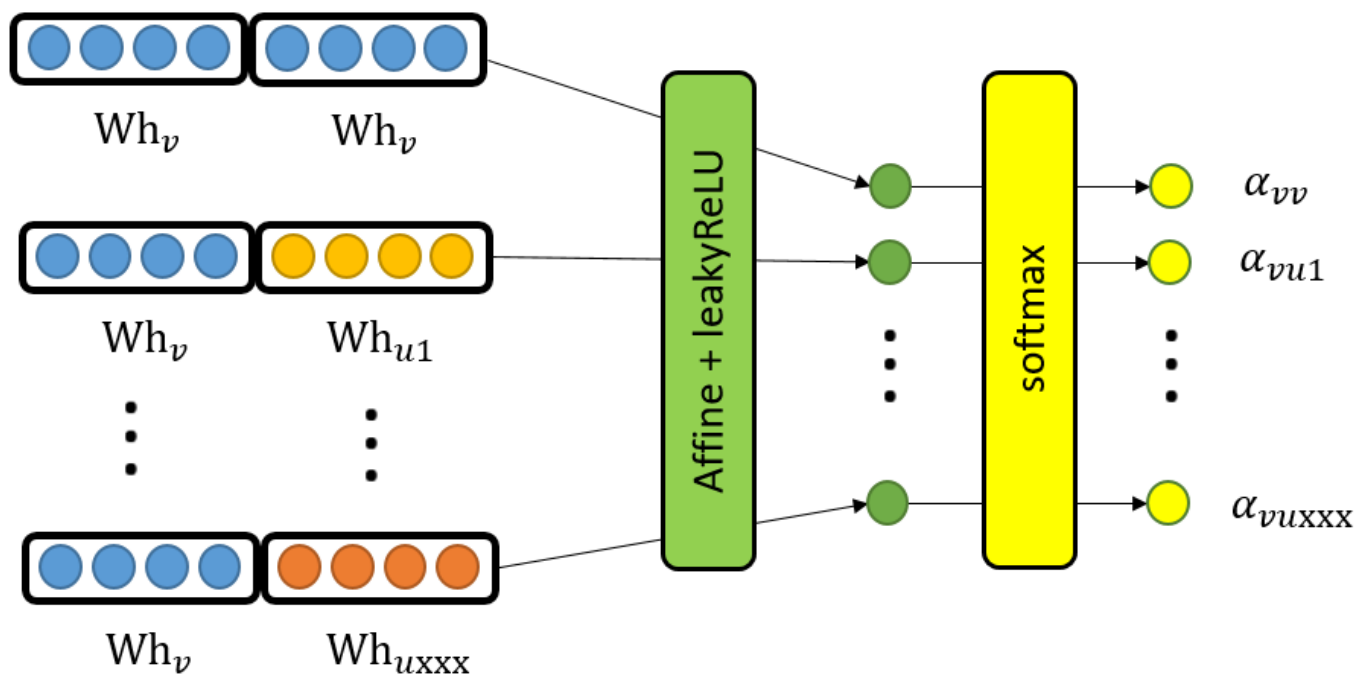
$\mathbf{P}^K \Rightarrow$ diffuse K times



Graph Attention Network

$$\mathbf{h}_v^{(k)} = \sigma \left(\sum_{u \in \mathcal{N}(v) \cup v} \underline{\alpha_{vu}^{(k)}} \mathbf{W}^{(k)} \mathbf{h}_u^{(k-1)} \right)$$

Attention weights



Comparison with Spectral-based GNN

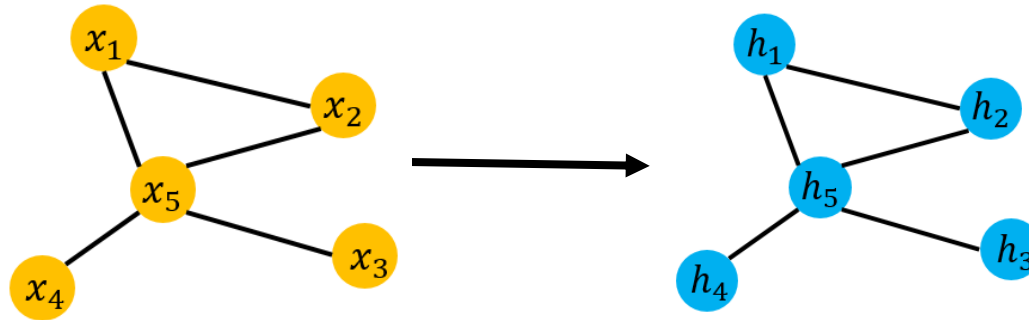
- Rely primarily on message passing and aggregation
- Scale to large graphs easily
- Perform computation on a batch of nodes, not whole graph
- Share weights easily across different locations and structures
- Generalize easily to different graphs, e.g. directed graphs

Outline

- Introduction to Graph
- Graph Signal Processing
- **Graph Neural Networks (GNNs)**
 - Spectral-based Convolutional Graph Neural Networks
 - Spatial-based Convolutional Graph Neural Networks
 - **Recurrent Graph Neural Networks (RecGNNs)**
 - Graph Autoencoders (GAEs)
 - Spatial-temporal Graph Neural Networks (STGNNs)
- Applications

Recurrent Feature Extraction

- Task: To extract node feature in a **recurrent** manner



- A node **exchanges information** with its neighbors

$$h_v^{(t)} = \sum_{u \in N(v)} f_w(x_v, x_{(v,u)}^e, x_u, h_u^{(t-1)})$$

Node features

Node inputs

Node features

Recurrent Feature Extraction

- RecGNNs

$$h_v^{(t)} = \sum_{u \in N(v)} f_w(x_v, x_{(u,v)}^e, x_u, h_u^{(t-1)})$$

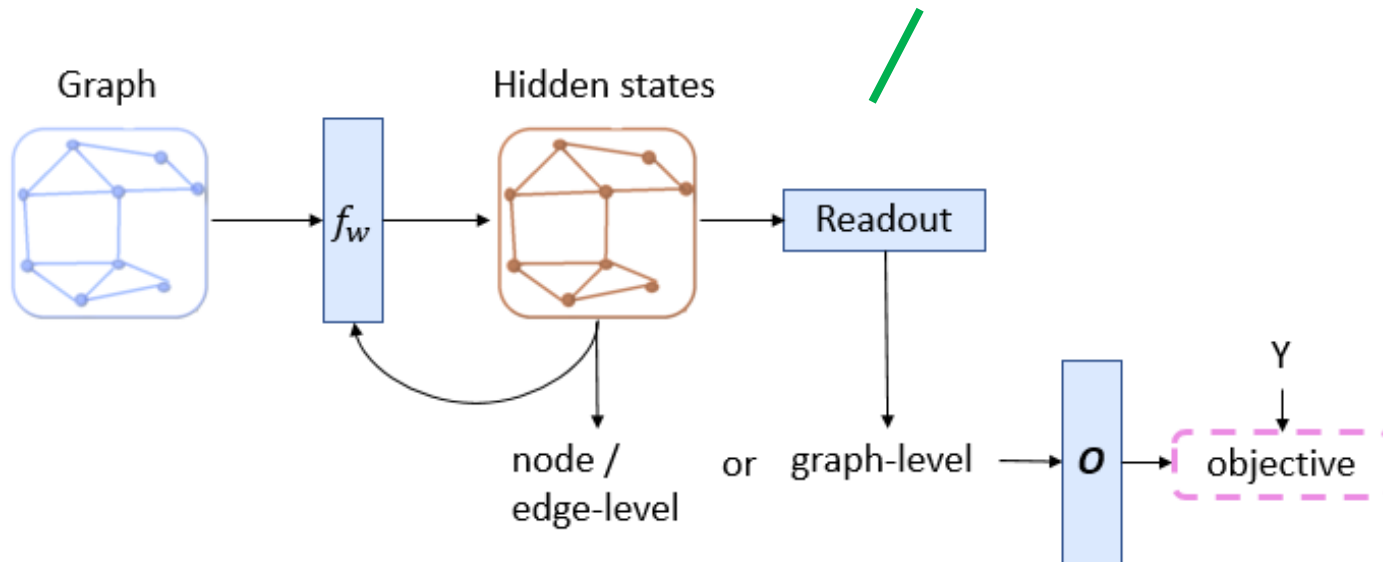
- f_w must be **contractive** for convergence

$$\|f_w(x) - f_w(y)\| < \|x - y\| \text{ for any } x, y$$

- Number of iterations for convergence is unknown

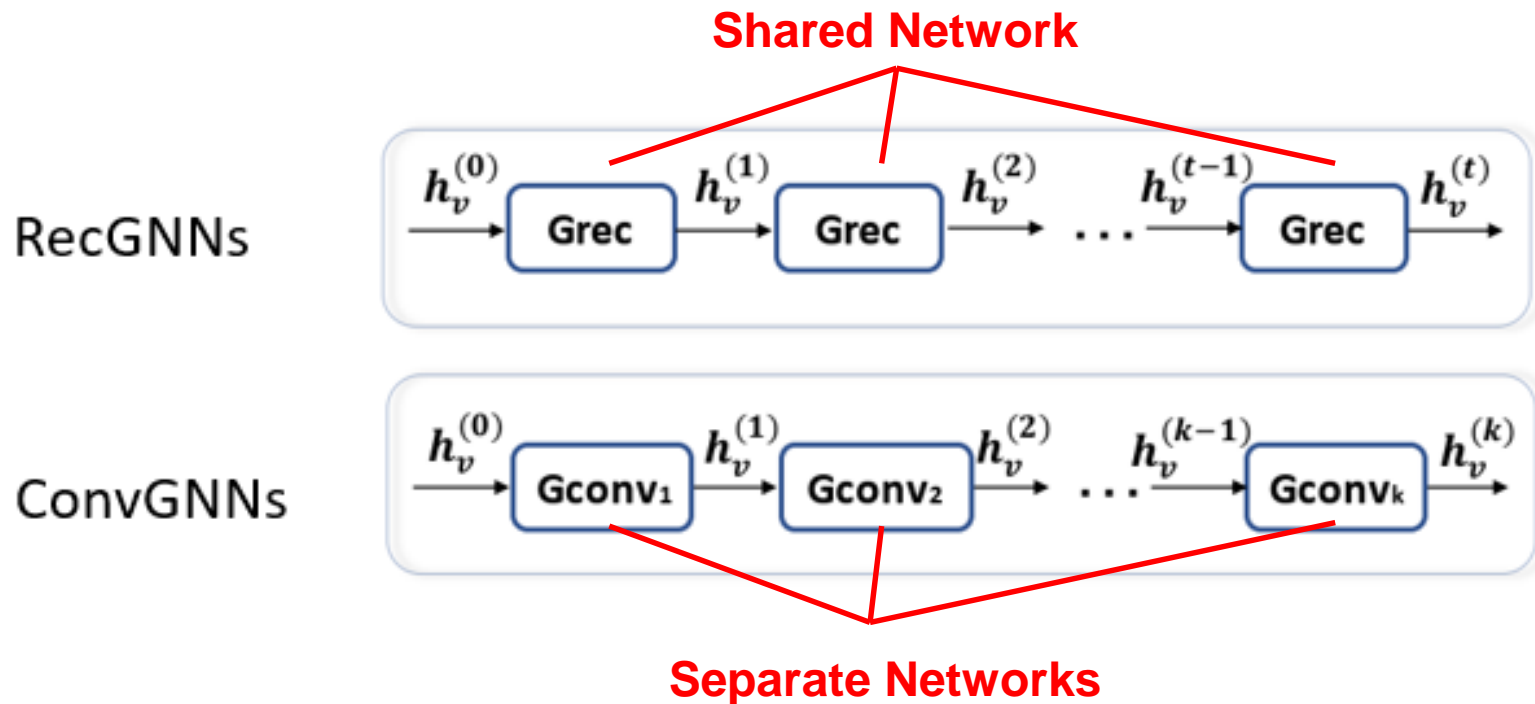
Training RecGNNs

1. Randomly initialize hidden states and networks.
 2. Recurrently update hidden states until convergence.
 3. Minimize training objective by updating f_w and o networks.
 4. Repeat step 2 and 3.
- generate graph-level representation
based on hidden stats i.e. sum, concatenate



RecGNNs vs. ConvGNNs

- RecGNNs – propagation is inefficient (t is unknown)
- ConvGNNs – controllable running time (k is known)



Gated Recurrent Neural Networks

- A **fixed number of layers** with training done by BPTT

$$h_v^{(t)} = GRU \left(h_v^{(t-1)}, \sum_{u \in N(v)} W h_u^{(t-1)} \right)$$

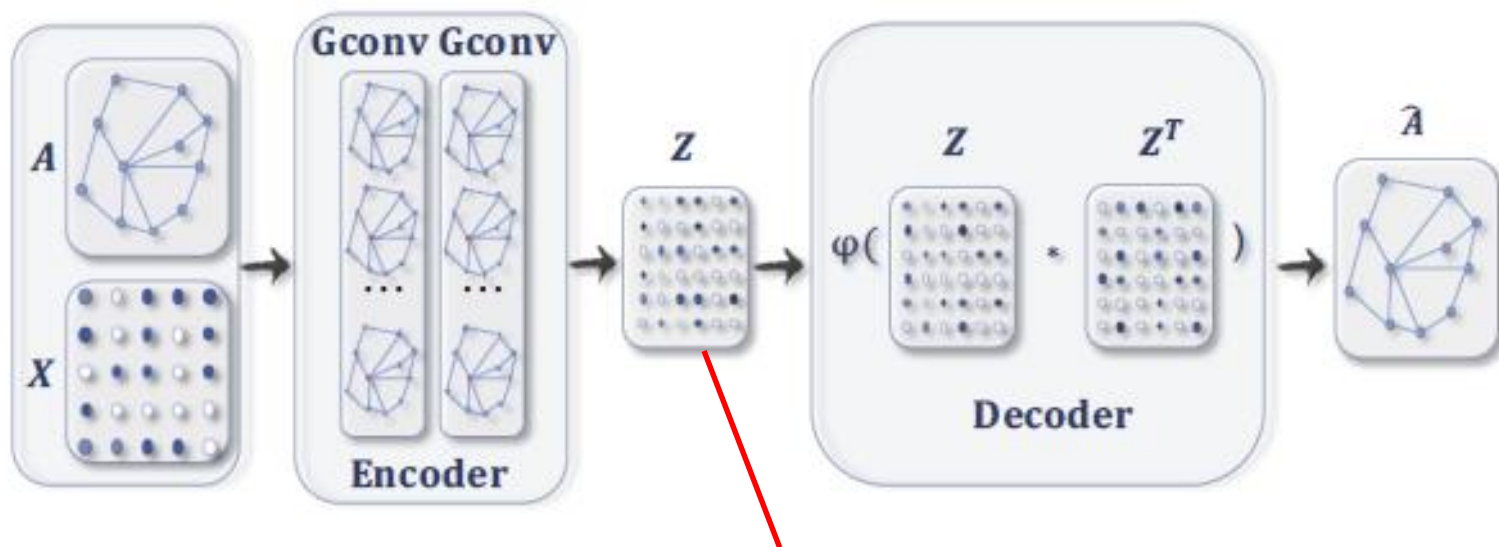
- $h_v^{(0)} = x_v$ is node input

Outline

- Introduction to Graph
- Graph Signal Processing
- **Graph Neural Networks (GNNs)**
 - Spectral-based Convolutional Graph Neural Networks
 - Spatial-based Convolutional Graph Neural Networks
 - Recurrent Graph Neural Networks (RecGNNs)
 - **Graph Autoencoders (GAEs)**
 - Spatial-temporal Graph Neural Networks (STGNNs)
- Applications

Graph Autoencoders (GAE)

- **Task:** To learn **network embeddings** (i.e. node representations) for nodes that preserve **graph topological structure**



Network embeddings

Graph Encoder and Decoder

- Encoder (GNN, MLP, etc) – to **learn network embeddings** by **structure and feature** information
- Decoder (CNN, MLP, etc) – to **predict links between nodes** for reconstructing **adjacency or PPMI matrix**

Positive Pointwise Mutual Information Matrix

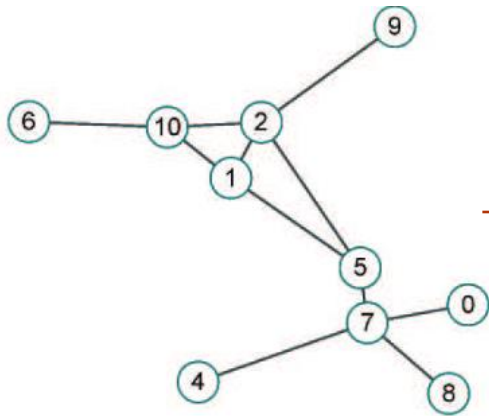
- Capture **mutual information between nodes** by random walks as a specification of the graph structure

$$\mathbf{PPMI}_{v_1, v_2} = \max(\log(\frac{\text{count}(v_1, v_2) \cdot |D|}{\text{count}(v_1)\text{count}(v_2)}), 0)$$

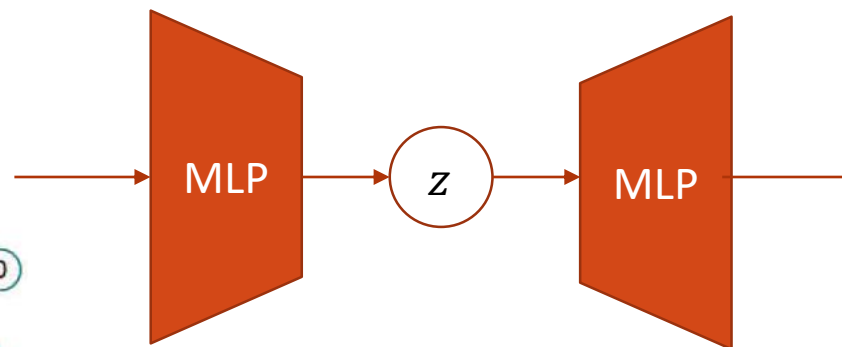
- $\text{count}(v_1)$: the **frequency of node v_1** being visited in sampled random walks
- $\text{count}(v_1, v_2)$: the frequency of **co-occurrence of nodes v_1 and v_2** in sampled random walks

Deep Neural Network for Graph Representations

Adjacency matrix



Denoising autoencoder



PPMI matrix

\vdots	\vdots	\vdots	\vdots	\vdots
\dots	0.05	0	0.7	\dots
\dots	0.1	0.3	0	\dots
\dots	0	0.2	0	\dots
\vdots	\vdots	\vdots	\vdots	\vdots

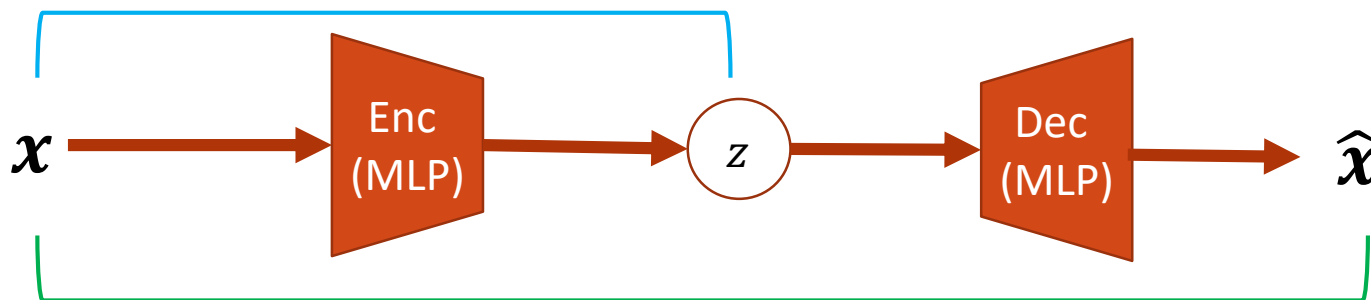
MLP: multi-layer perceptron

Structural Deep Network Embedding

- To preserve the node **first-order** and **second-order** proximity

$$L_{1st} = \sum_{(v,u) \in E} A_{v,u} ||enc(\mathbf{x}_v) - enc(\mathbf{x}_u)||^2$$

first-order



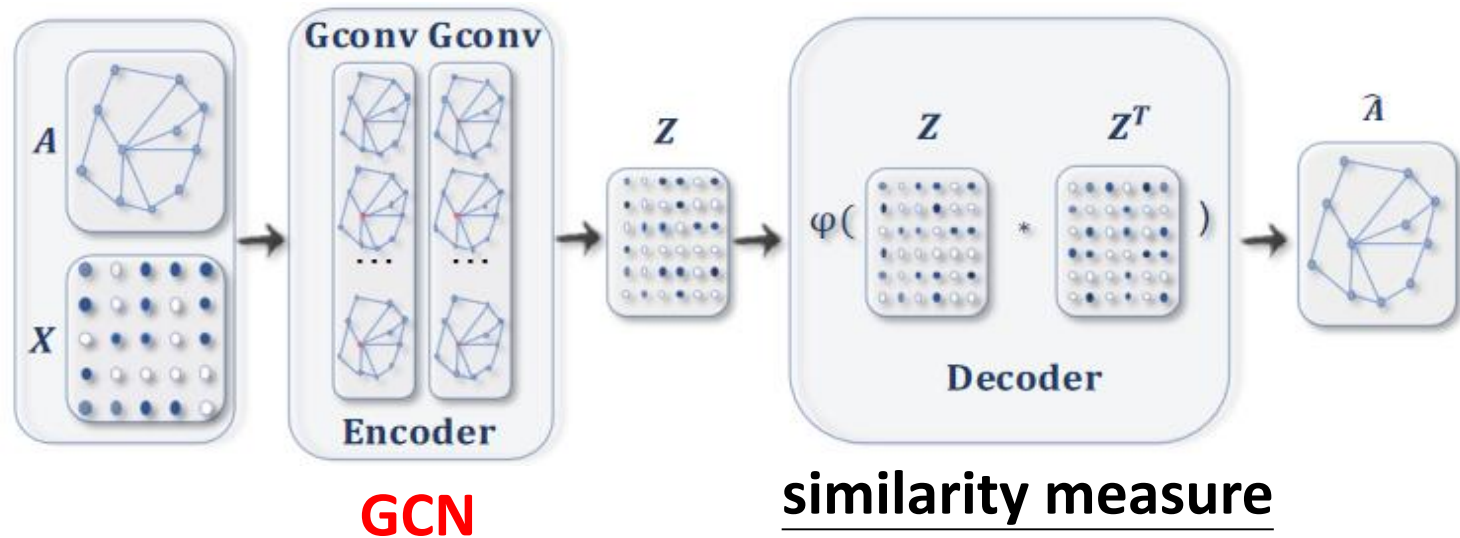
Second-order

$$L_{2nd} = \sum_{v \in V} ||(dec(enc(\mathbf{x}_v)) - \mathbf{x}_v) \odot \mathbf{b}_v||^2$$

$$b_{v,u} = 1 \quad \text{if } A_{v,u} = 0$$
$$b_{v,u} = \beta > 1 \quad \text{if } A_{v,u} = 1$$

Graph Autoencoder (GAE*)

- GAE* adopts **2-layered GCN** to leverage both **structure** and **feature information**



$$\hat{\mathbf{A}}_{v,u} = \text{dec}(\mathbf{z}_v, \mathbf{z}_u) = \sigma(\mathbf{z}_v^T \mathbf{z}_u)$$

Variational Graph Autoencoder

- To learn the **generative distribution** of graph
- Optimize the variational lower bound L

$$L = E_{q(\mathbf{Z}|\mathbf{X}, \mathbf{A})} [\log p(\mathbf{A}|\mathbf{Z})] - KL[q(\mathbf{Z}|\mathbf{X}, \mathbf{A}) || p(\mathbf{Z})]$$

- $q(\mathbf{Z}|\mathbf{X}, \mathbf{A}) = \prod_{i=1}^n q(z_i|\mathbf{X}, \mathbf{A})$, $q(z_i|\mathbf{X}, \mathbf{A}) = N(z_i|\mu_i, \text{diag}(\sigma_i^2))$
- $p(\mathbf{Z}) = \prod_{i=1}^n p(z_i)$, $p(z_i) = N(z_i|0, \mathbf{I})$
- $p(\mathbf{A}_{ij} = 1|z_i, z_j) = \text{dec}(z_i, z_j) = \sigma(z_i^T z_j)$

Outline

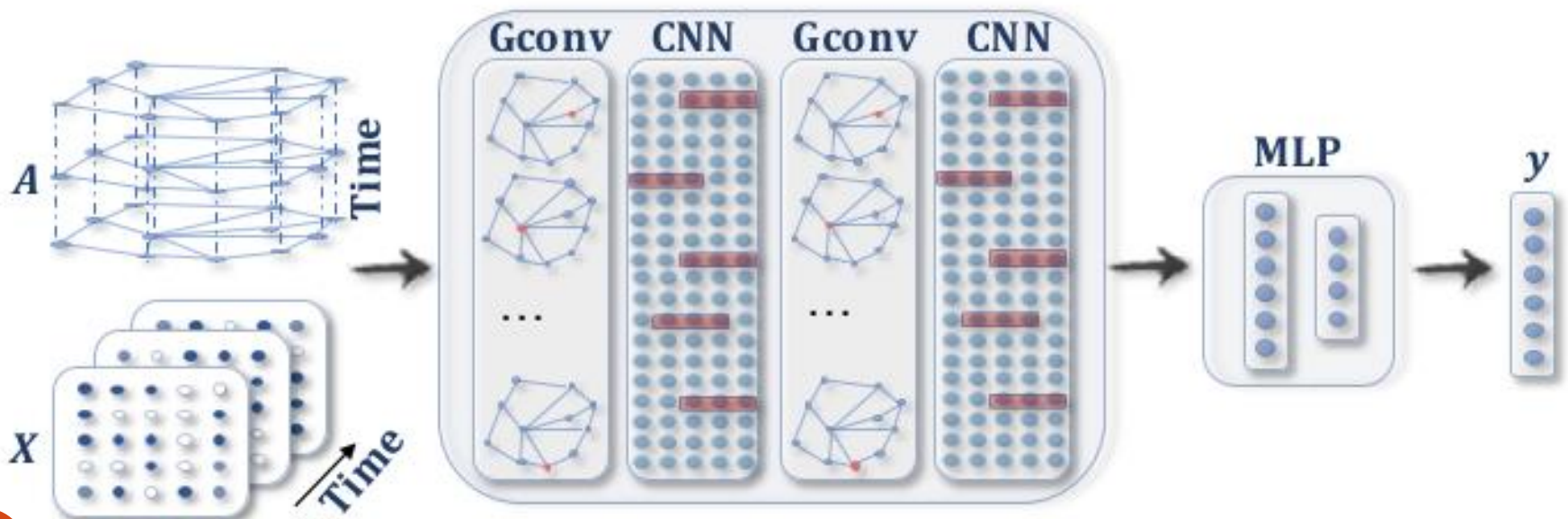
- Introduction to Graph
- Graph Signal Processing
- **Graph Neural Networks (GNNs)**
 - Spectral-based Convolutional Graph Neural Networks
 - Spatial-based Convolutional Graph Neural Networks
 - Recurrent Graph Neural Networks (RecGNNs)
 - Graph Autoencoders (GAEs)
 - **Spatial-temporal Graph Neural Networks (STGNNs)**
- Applications

Spatial-temporal Graph Neural Networks

- Real-world graphs are often **dynamic**
- Spatial-temporal Graph Neural Networks (STGNNs)
 - To capture **spatial dynamics via GNN**
 - To capture **temporal dynamics via RNN or CNN**

CNN-based STGNNs

- Apply GCN to aggregate spatial information at individual time instances
- Apply **1D convolution** to **co-located nodes across time instances** to aggregate temporal information



RNN-based STGNNs

- Recap on RNN

$$H^{(t)} = \sigma(WX^{(t)} + UH^{(t-1)} + b),$$

- $X^{(t)} \in R^{n \times d}$ is the node feature matrix at time t

- RNN-based GCN

$$H^{(t)} = \sigma(Gconv(X^{(t)}, A; W) + Gconv(H^{(t-1)}, A; U) + b)$$

Outline

- Introduction to Graph
- Graph Signal Processing
- Graph Neural Networks (GNNs)
 - Spectral-based Convolutional Graph Neural Networks
 - Spatial-based Convolutional Graph Neural Networks
 - Recurrent Graph Neural Networks (RecGNNs)
 - Graph Autoencoders (GAEs)
 - Spatial-temporal Graph Neural Networks (STGNNs)
- Applications

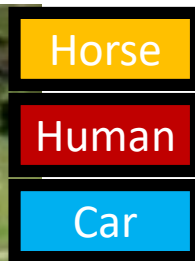
Weakly-Supervised Image Semantic Segmentation Using Graph Convolutional Networks

Shun-Yi Pan, Cheng-You Lu, Shih-Po Lee, and Wen-Hsiao Peng
National Yang Ming Chiao Tung University, Taiwan

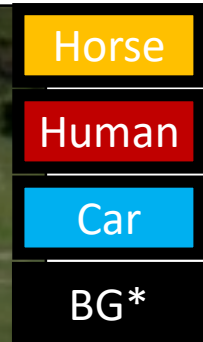
IEEE International Conference on Multimedia and Expo (ICME), July 2021.

Task

- To classify pixels in images into semantic classes with **image-level annotations**



Predict



Input

Output

Common Solutions

- Produce **pseudo labels (PLs)** as ground-truth labels
- Use **Class Activation Map (CAM)** to generate PLs

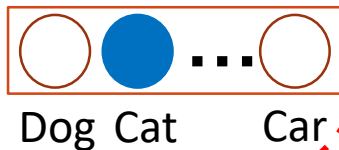
Training



Input image



Loss



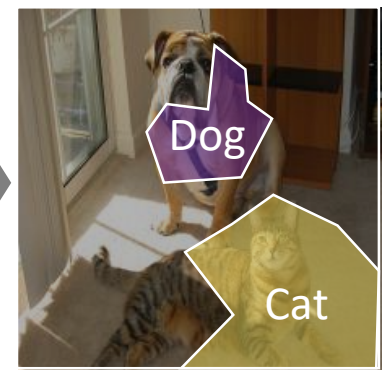
Dog Cat Car

Inference



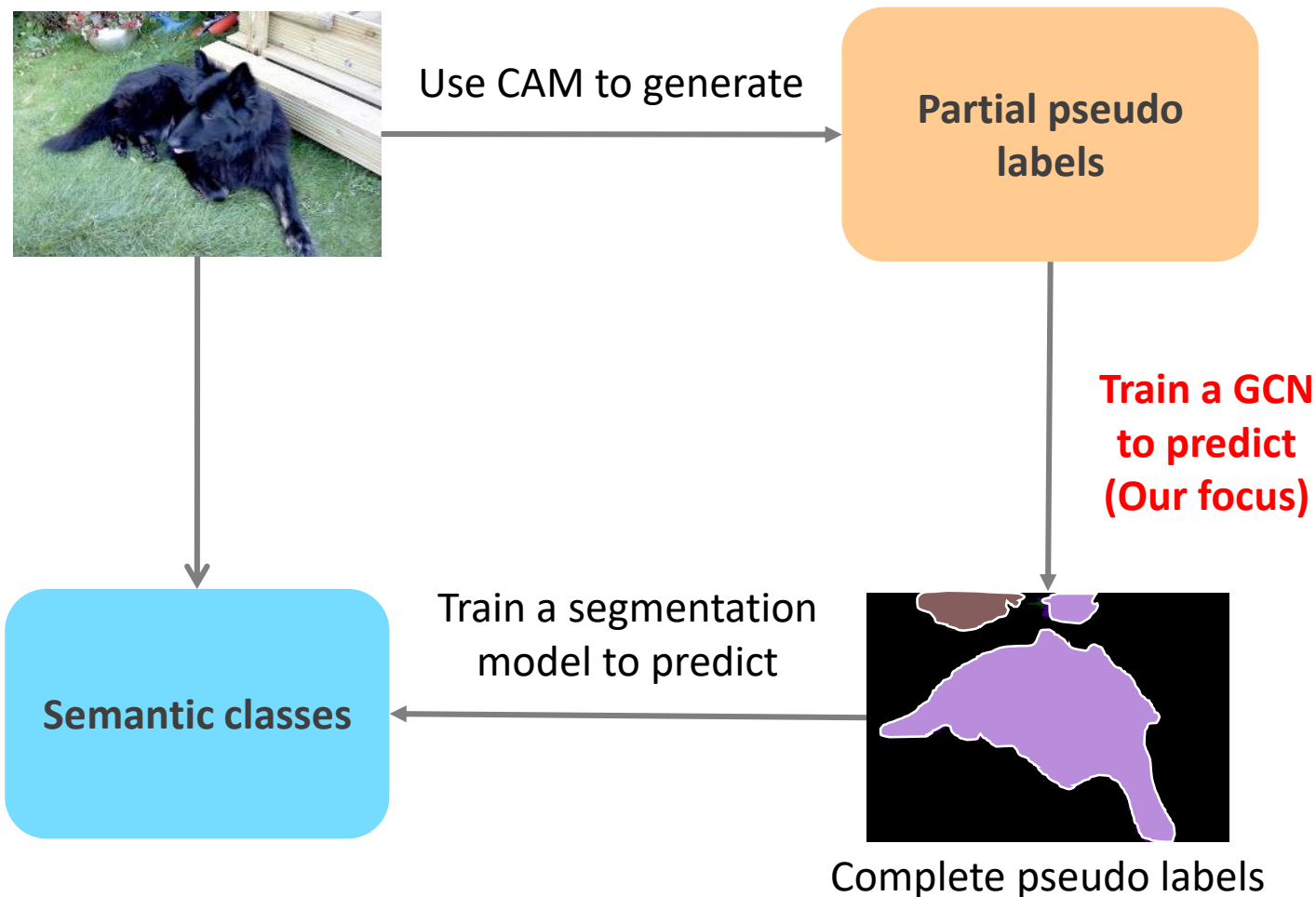
CAM

Threshold



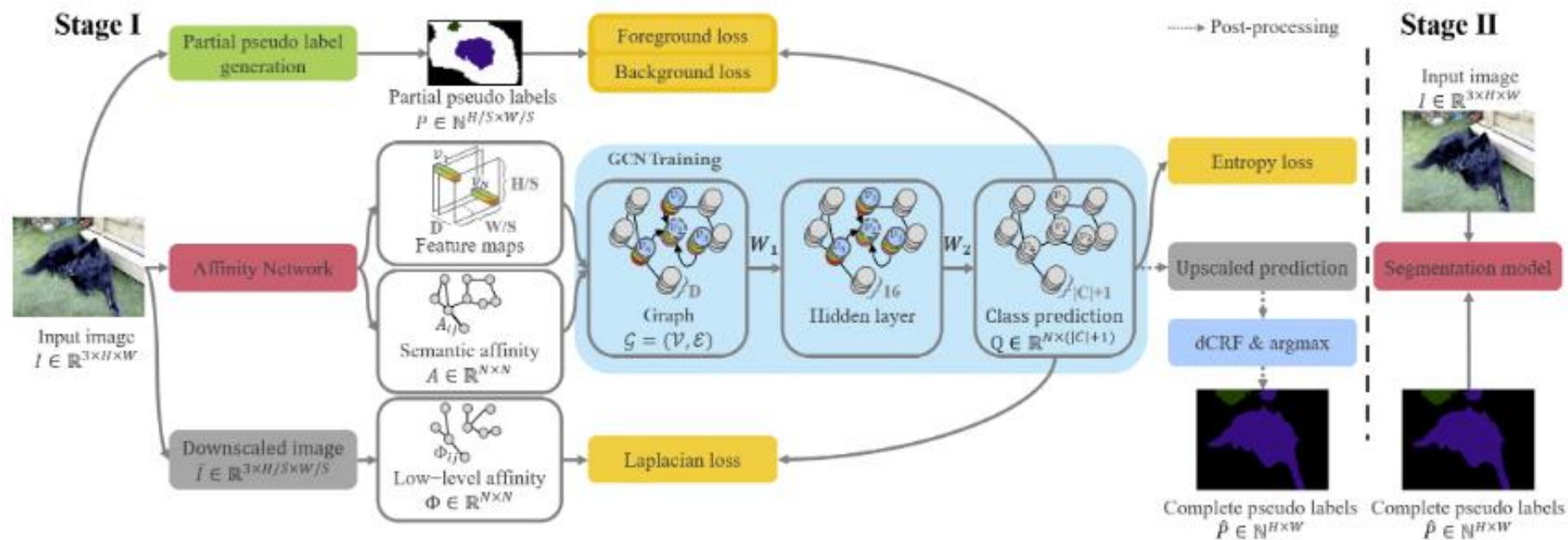
Pseudo labels

Proposed Method



GCN for Label Propagation

- Use Affinity Network to generate **features of nodes and edges**
- Train a GCN to generate **complete PLs** from **partial PLs**



Loss functions

- Foreground and background loss (**for pixels with pseudo labels**)

$$\ell_{fg} = -\frac{1}{|V_{fg}|} \sum_{i \in V_{fg}} \log(q_i)_{p_i}, \ell_{bg} = -\frac{1}{|V_{bg}|} \sum_{i \in V_{bg}} \log(q_i)_{p_i}$$

- Laplacian loss (**for all pixels**)

$$\ell_{lp} = \frac{1}{2|V|} \sum_{i \in V} \sum_{j \in V} \Phi_{i,j} \|q_i - q_j\|_2^2,$$
$$\Phi_{i,j} = \begin{cases} \exp\left(-\frac{\|\bar{I}_i - \bar{I}_j\|^2}{2\sigma_1^2} - \frac{\|\bar{f}_i - \bar{f}_j\|^2}{2\sigma_2^2}\right) & \text{if } j \in N_i \\ 0 & \text{otherwise} \end{cases}$$

- Entropy loss (**for unlabeled pixels**)

$$\ell_{ent} = -\frac{1}{|V_{ig}|} \sum_{i \in V_{ig}} \sum_{c \in \bar{C}} (q_i)_c \log(q_i)_c$$

Results on PASCAL VOC 2012

IRNet
(Baseline)



WSGCN
(Ours)



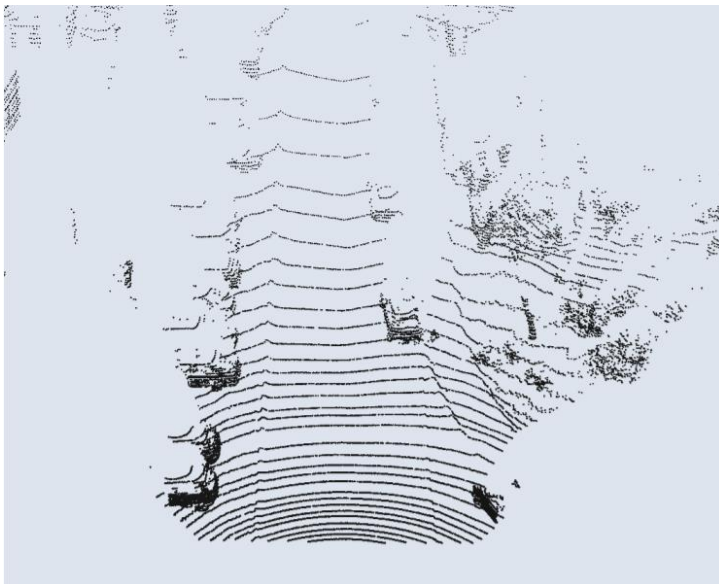
Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud

Weijing Shi and Ragunathan (Raj) Rajkumar
Carnegie Mellon University

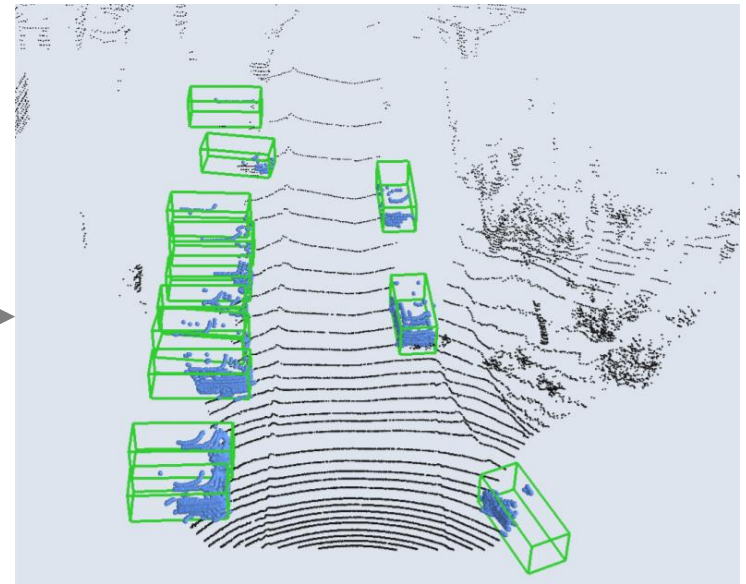
IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

Task Specification

- To predict a 3D bounding box around group of 3D points representing 3 classes (car, pedestrian, cyclist)



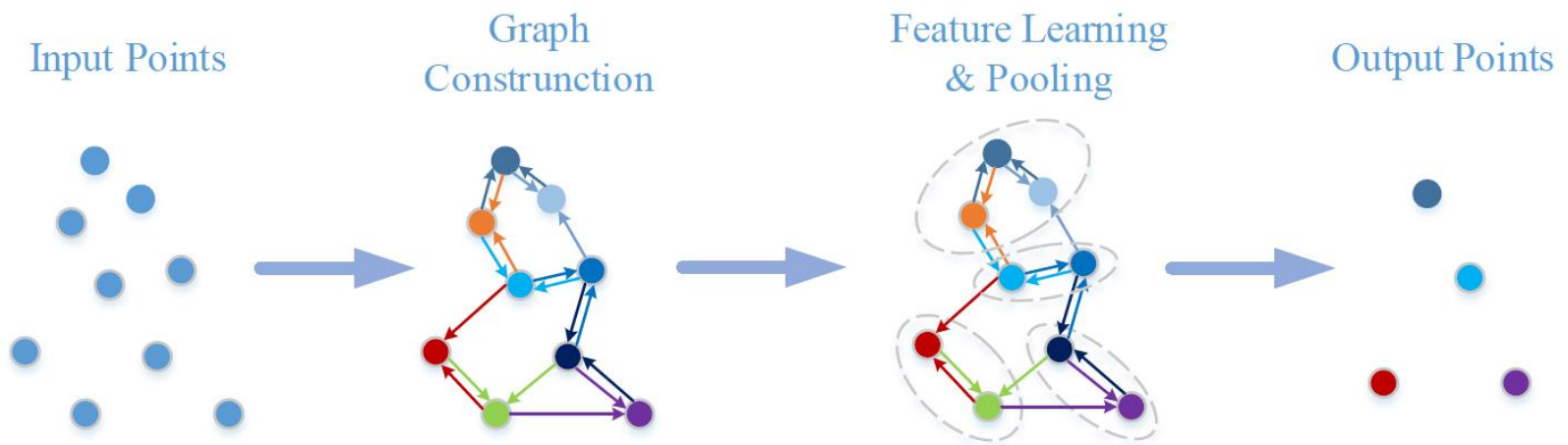
Input



Output
(only car class is shown)

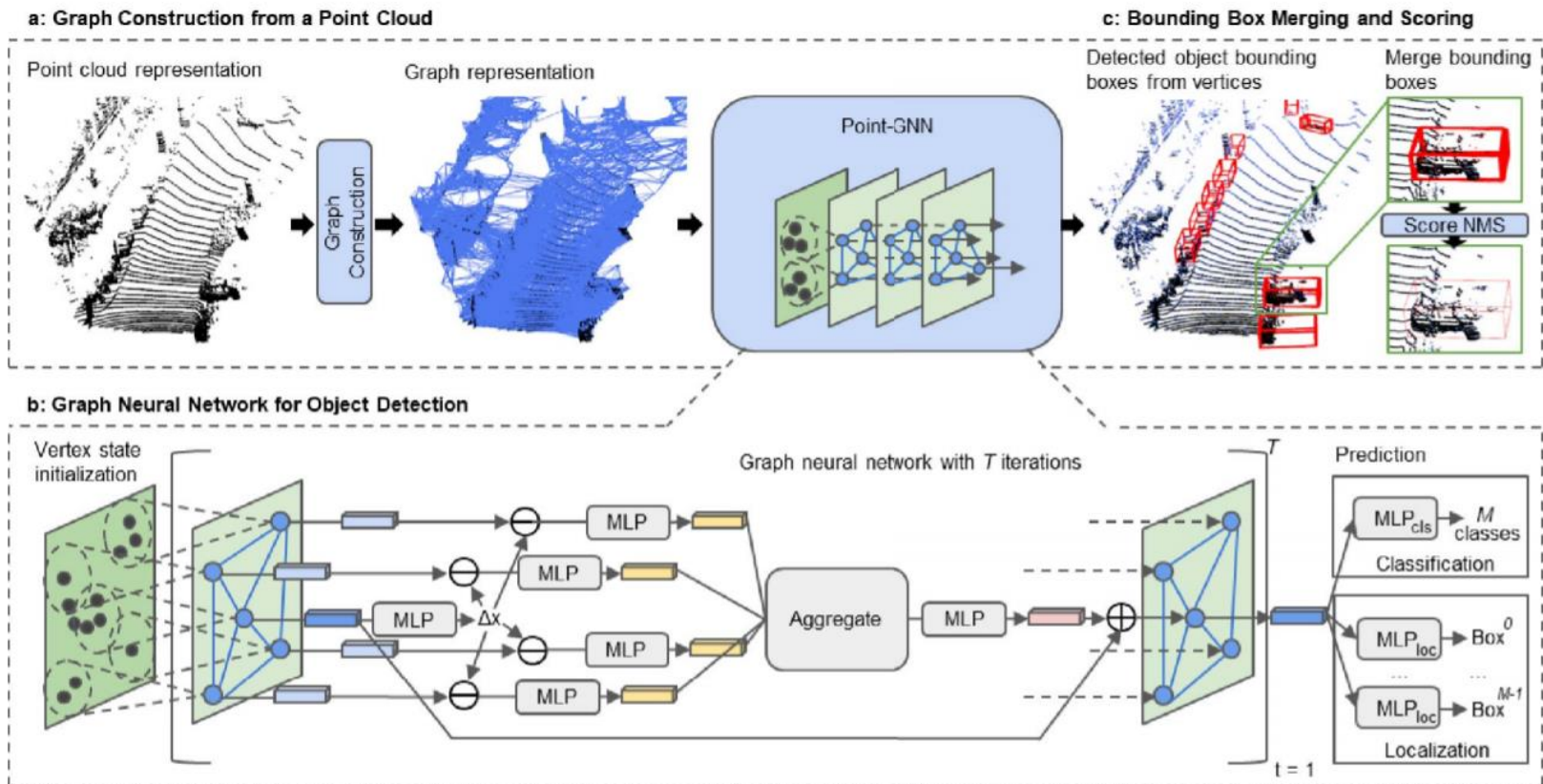
GNN in 3D Point Cloud

- Each point in a point cloud is a vertex of a graph
- Directed edges are formed for the graph
- Feature learning is performed in spatial domain

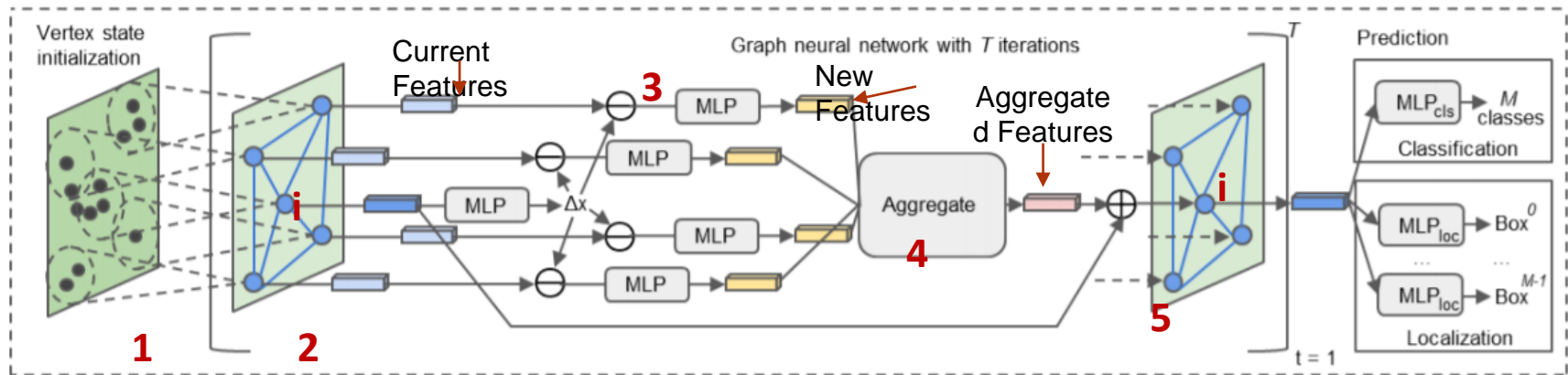


3D Object Detection in 3D Point Cloud

- Point-GNN



3D Object Detection in 3D Point Cloud



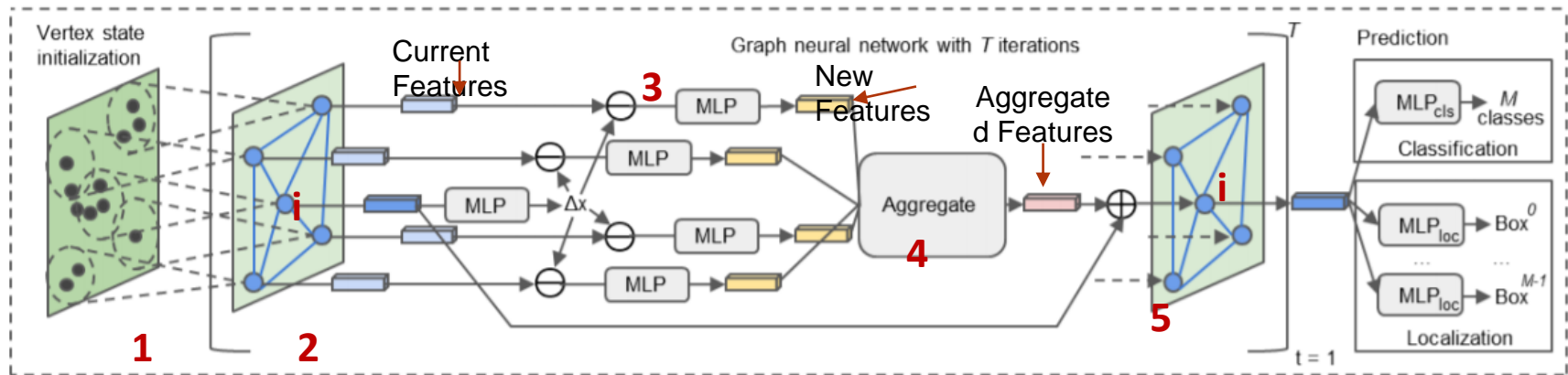
1. Point Cloud down-sampling to reduce initial point cloud size
2. Graph formation by connecting a point to its neighbors within a fixed radius r .

$$Graph\ G = (P, E)$$

P = Vertices = points in point cloud

$$E = \text{Edges} = \{(p_i, p_j) \mid \|x_i - x_j\|_2 < r\}$$

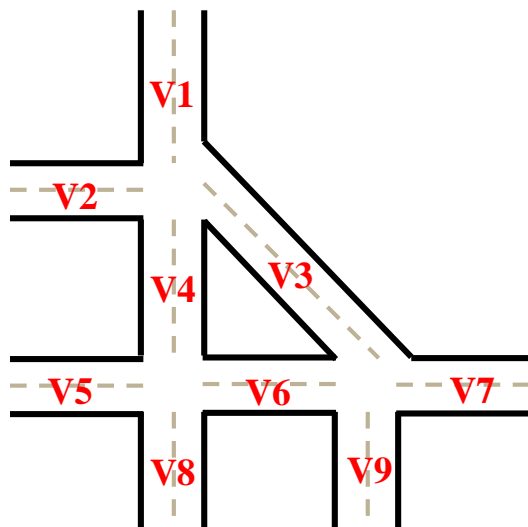
3D Object Detection in 3D Point Cloud



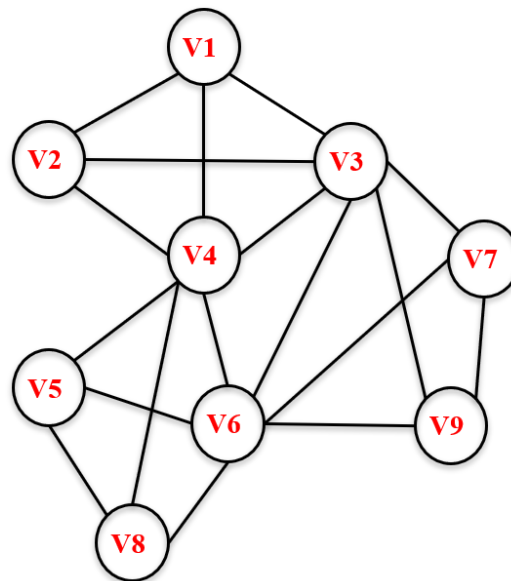
- 3.** Each neighborhood vertex's MLP's input is its **current features** and an offset generated by MLP of the center vertex **i**. The output is learned **new features**
- 4.** Feature aggregation stage aggregates **new features** of neighboring vertices and produce new **aggregated features** for center vertex **i**. **Aggregated features** become **new current features** of center point
- 5.** Graph is updated with new features for each vertex. The loop follows until specific iterations T

Traffic Prediction Using STGNN

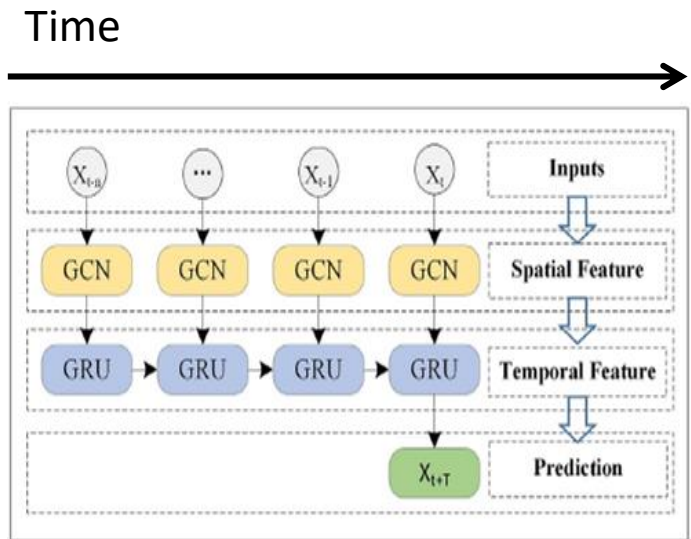
- To predict traffic conditions on urban roads based on time-series data



Map



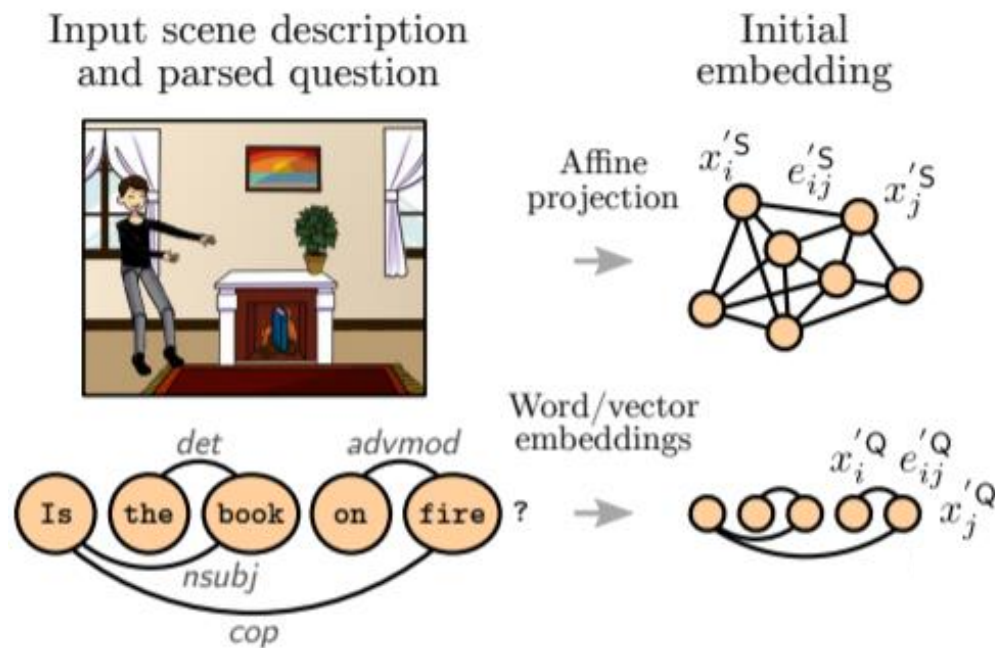
Graph Topology



STGCNN

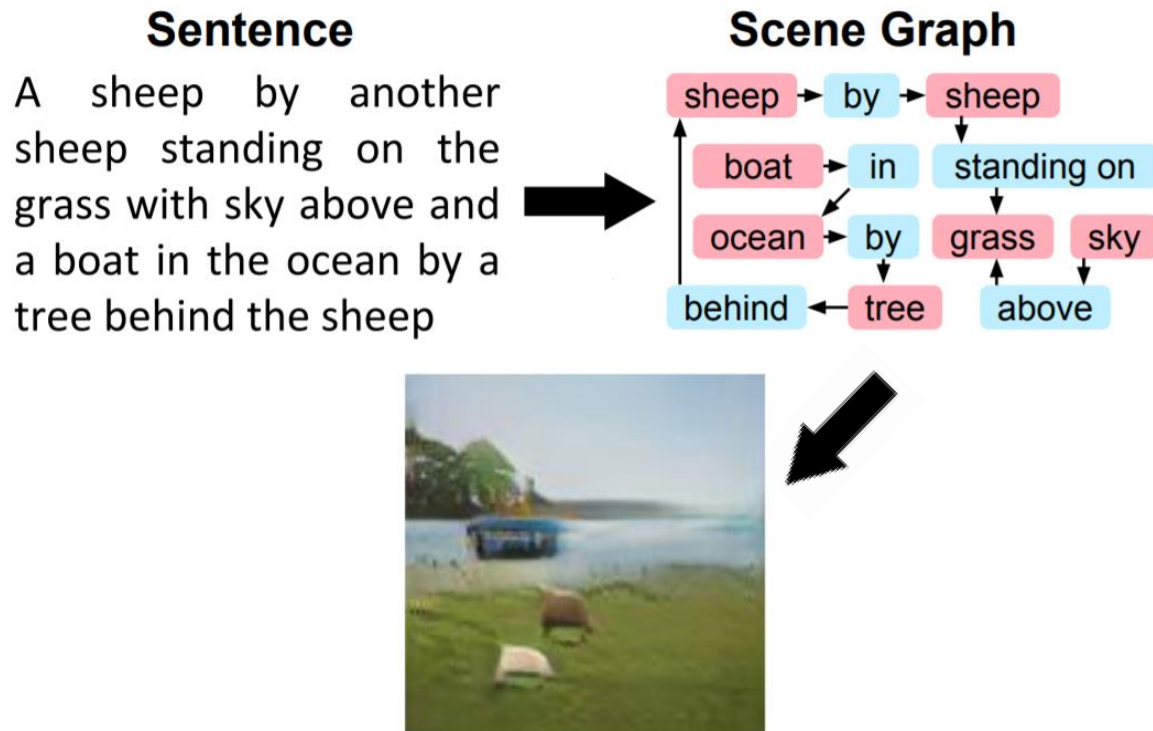
Visual Question Answering

- To answer questions based on description of an image
- Encode the descriptions of scene and question as graphs
- Scene – **node**: objects; **edge**: spatial arrangement
- Question – **node**: words; **edge**: syntactic dependencies



Text to Image Synthesis

- To generate images from natural language descriptions
- Convert sentences to scene graphs
- Generate images from scene graphs



Thank you for your attention

Reference

- Zhou et al., “Graph Neural Networks: A Review of Methods and Applications,” arxiv, 2019.
- Wu et al., “A Comprehensive Survey on Graph Neural Networks,” arxiv, 2019.
- Xu et al., “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention,” in ICML, 2015.
- Ying et al., “Hierarchical Graph Representation Learning with Differentiable Pooling,” in NIPS, 2018.
- Cao et al., “deep neural networks for learning graph representations,” in AAAI, 2016.
- Ortega et al., “Graph Signal Processing: Overview, Challenges and Applications,” in Proc. IEEE, 2018.
- Zhou et al., “T-gcn: A temporal graph convolutional network for traffic prediction,” in TITS, 2019.
- Guo et al., “Attention based spatial-temporal graph convolutional networks for traffic flow forecasting,” in AAAI, 2019.
- Wang et al., “Zero-shot recognition via semantic embeddings and knowledge graphs,” in CVPR, 2018.
- Teney et al., “Graph-structured representations for visual question answering,” in CVPR, 2017.
- Johnson et al., “Image generation from scene graphs,” in CVPR, 2018.
- Zhou et al., “Learning deep features for discriminative localization,” in CVPR, 2016.
- Ahn et al., “Weakly supervised learning of instance segmentation with interpixel relations,” in CVPR, 2019.
- Guo et al., “Deep Learning for 3D Point Clouds: A Survey,” in IEEE TPAMI, 2020.
- Shi et al., “Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud,” in CVPR, 2020.