AIMS CDT SIGNAL PROCESSING

Introduction to graph neural networks

Dr. Dorina Thanou

EPFL

21.10.2021

Outline

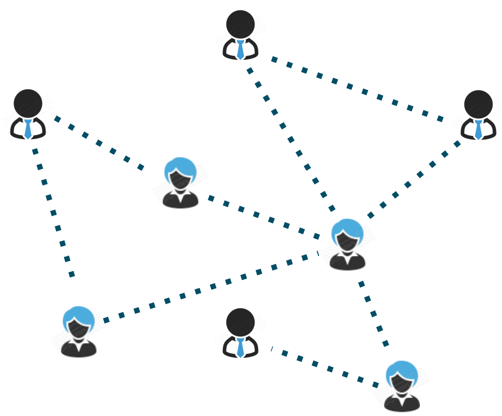
- Motivation: Why Graph Neural Networks?

- Basic definitions on graphs

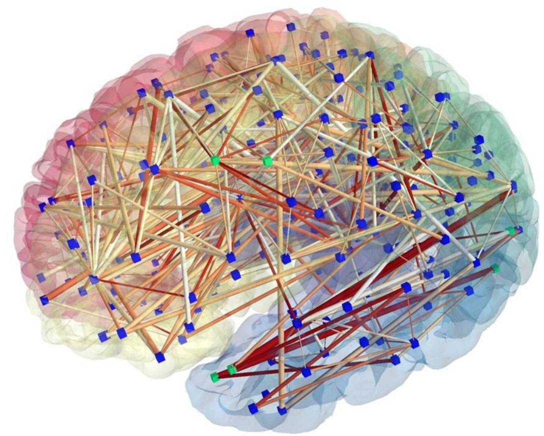
- Partial historical overview
  - Graph CNN (main focus)
  - Graph autoencoders (briefly)

- Applications
  - Naturally graph-structured data
  - Images

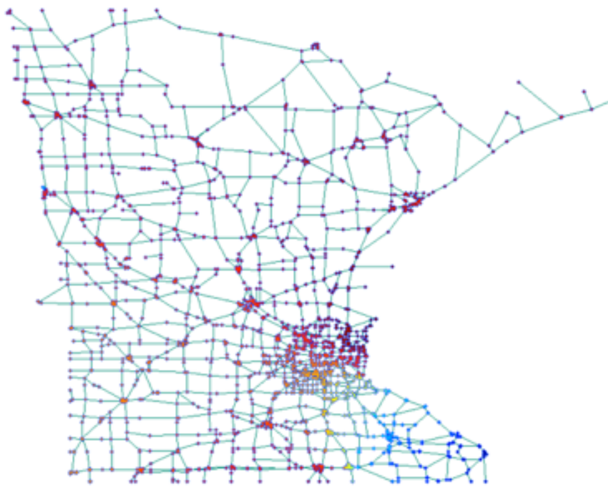
Network data are pervasive



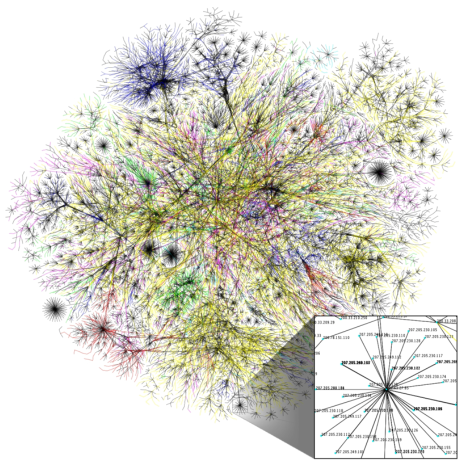
Social networks



Biological networks

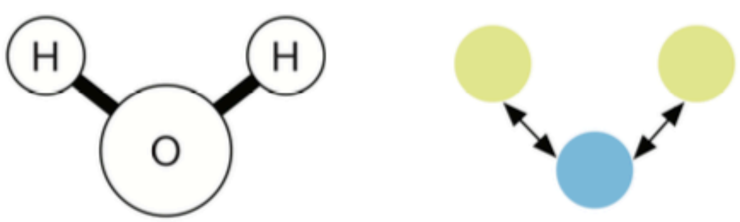


Transportation networks



Communication networks

n-body system

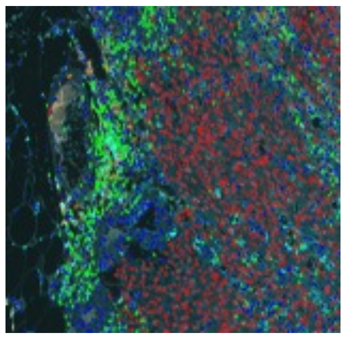


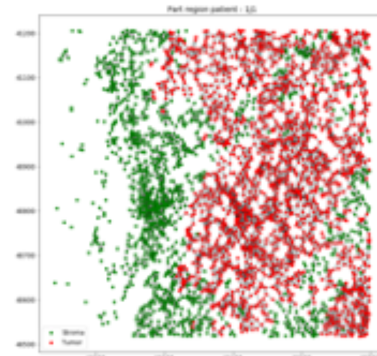
molecule

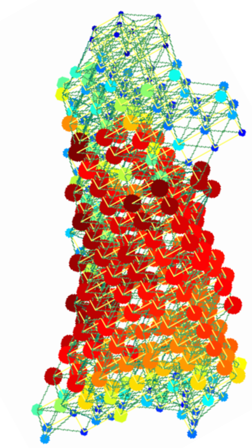
Graphs provide a mathematical representation for describing and modeling complex systems

Why graph-based modelling?

- Provide a different perspective of data

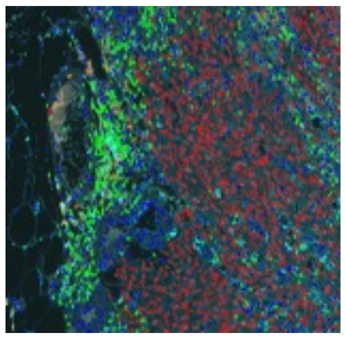


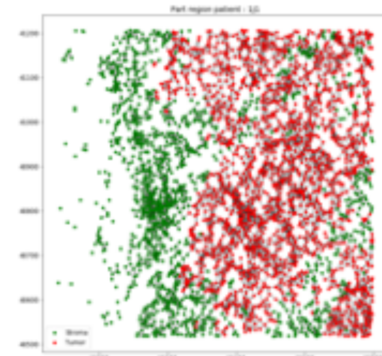


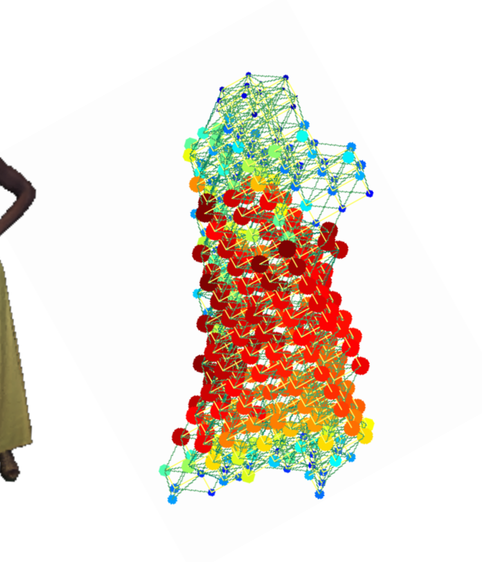


Why graph-based modelling?

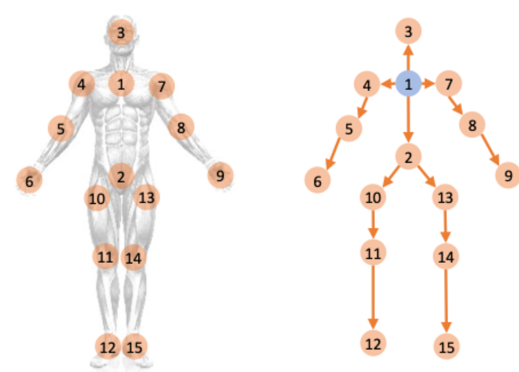
- Provide a different perspective of data





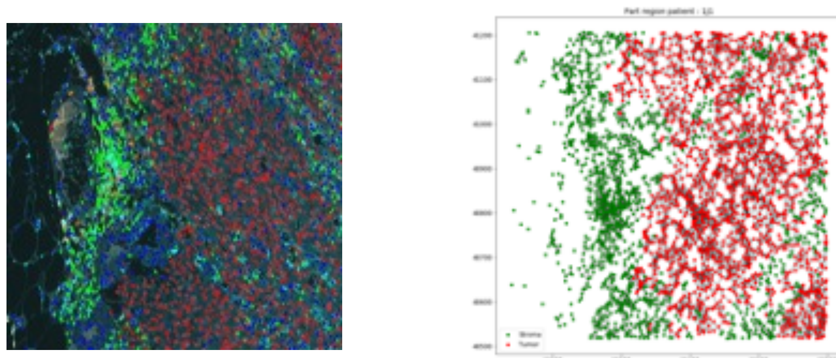


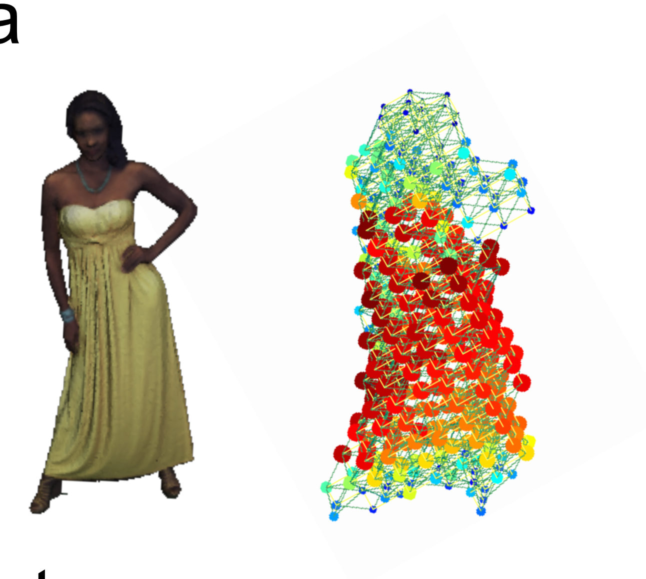
- Impose relational inductive bias in data



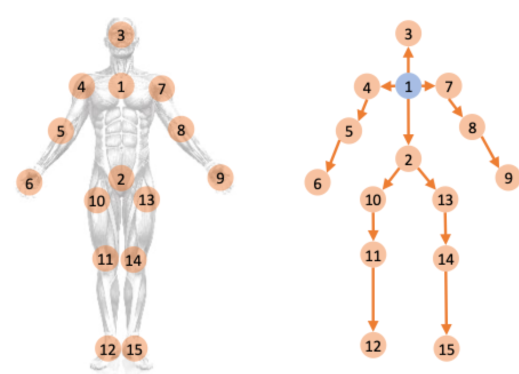
Why graph-based modelling?

- Provide a different perspective of data

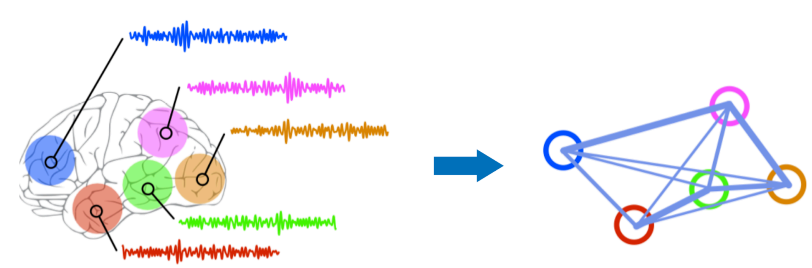




- Impose relational inductive bias in data

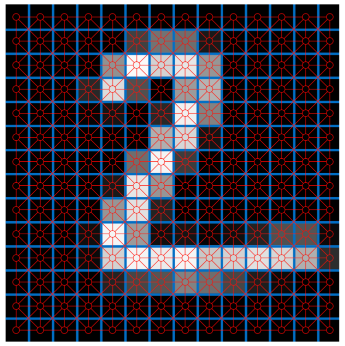


- Lead to knowledge discovery



Typical learning tasks on graph-structured data

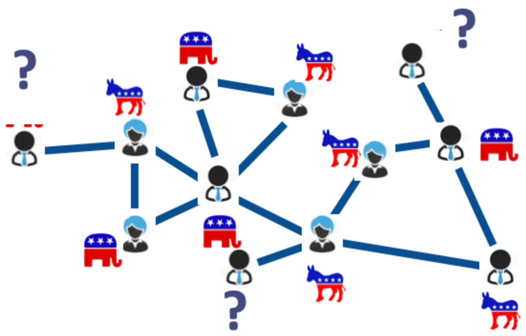
- Graph-wise classification



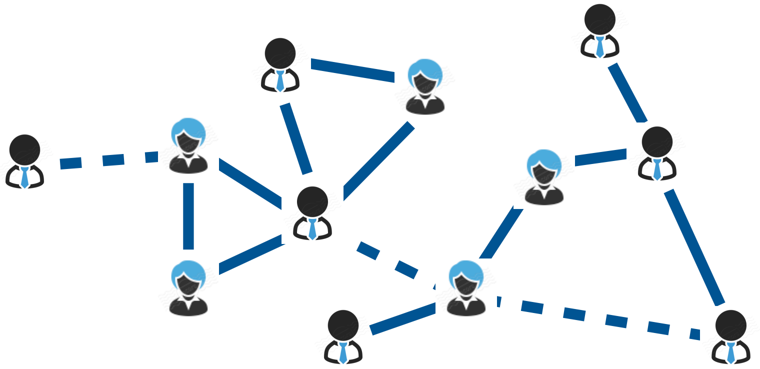
is it a 2?
is it a 4?

condition?
no condition?

- Vertex-wise classification/inference of missing values

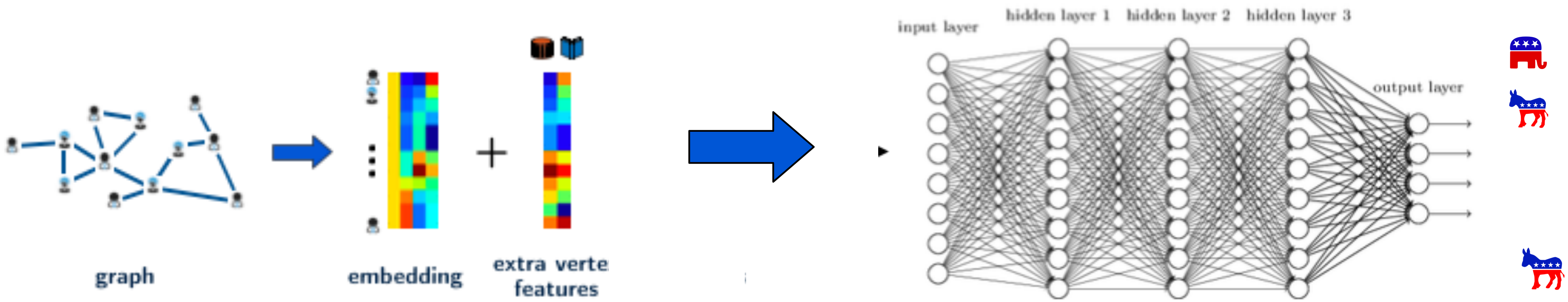


- Link prediction



A naive approach

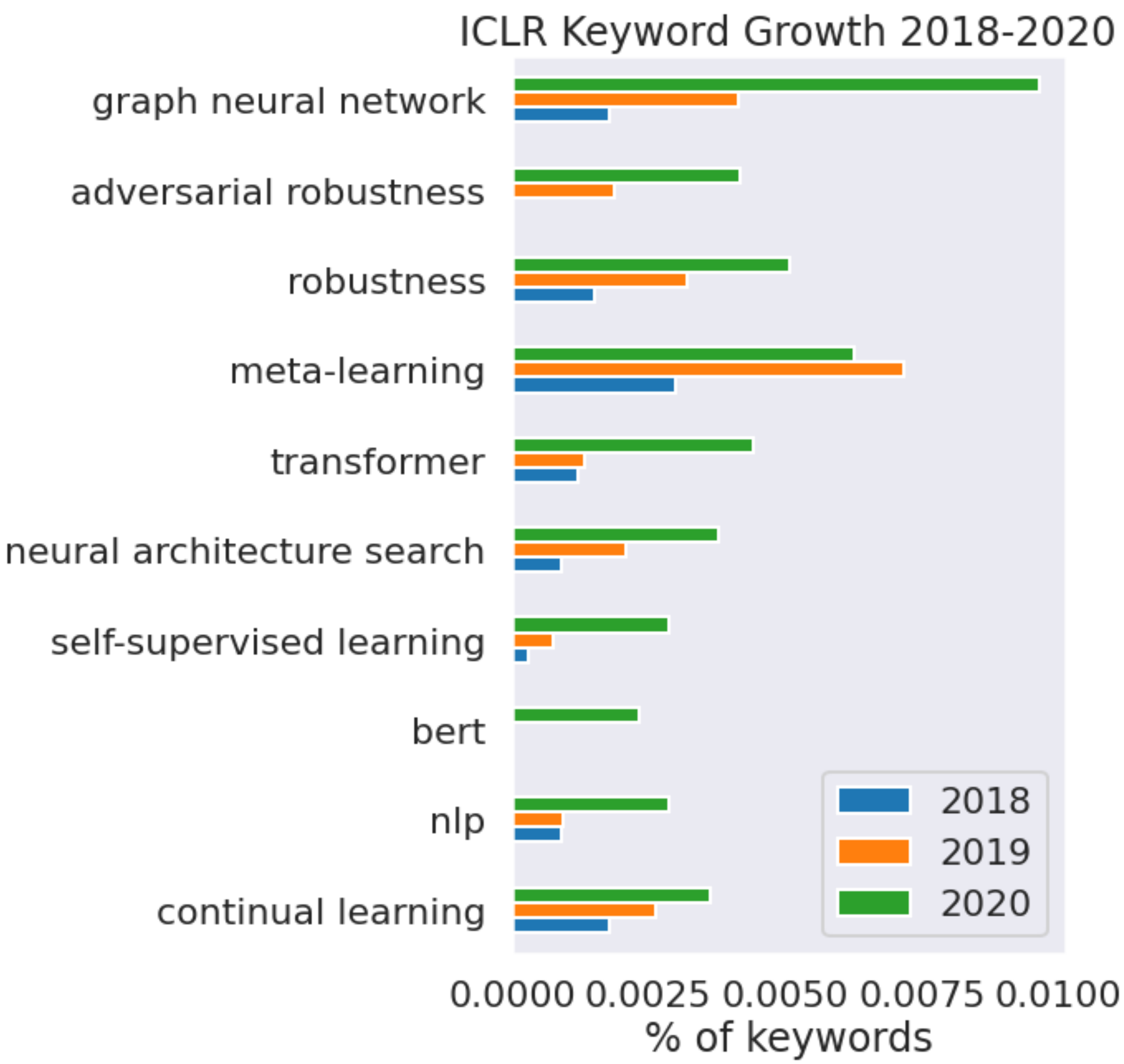
- Embed graph and features into a Euclidean space
- Feed them into a deep neural net

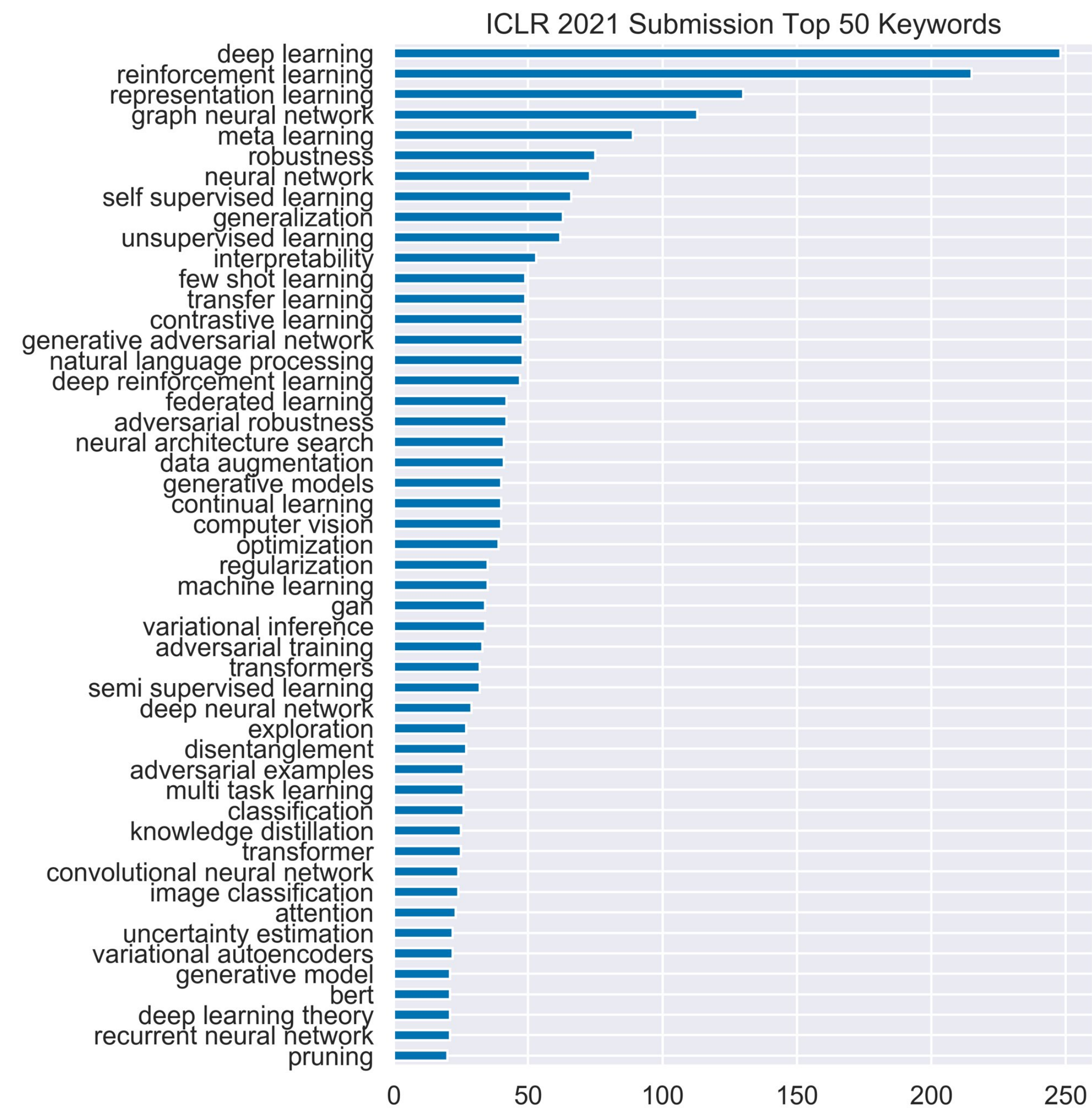


- Issues with that:
  - O(N) parameters
  - Not applicable to graphs of different sizes
  - Not invariant to node ordering

Can we do better?

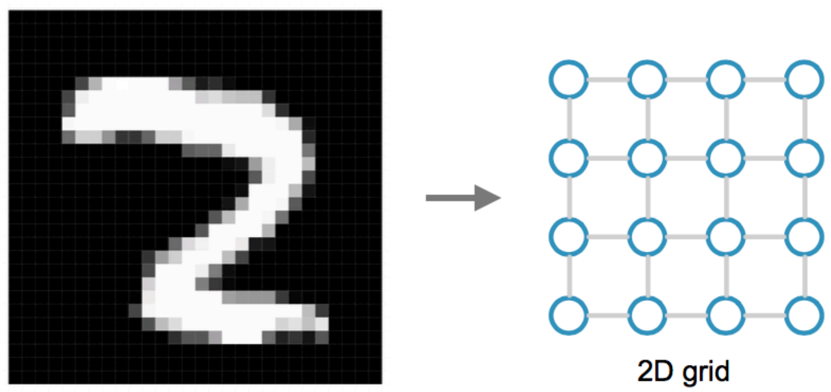
GNN: A growing trend



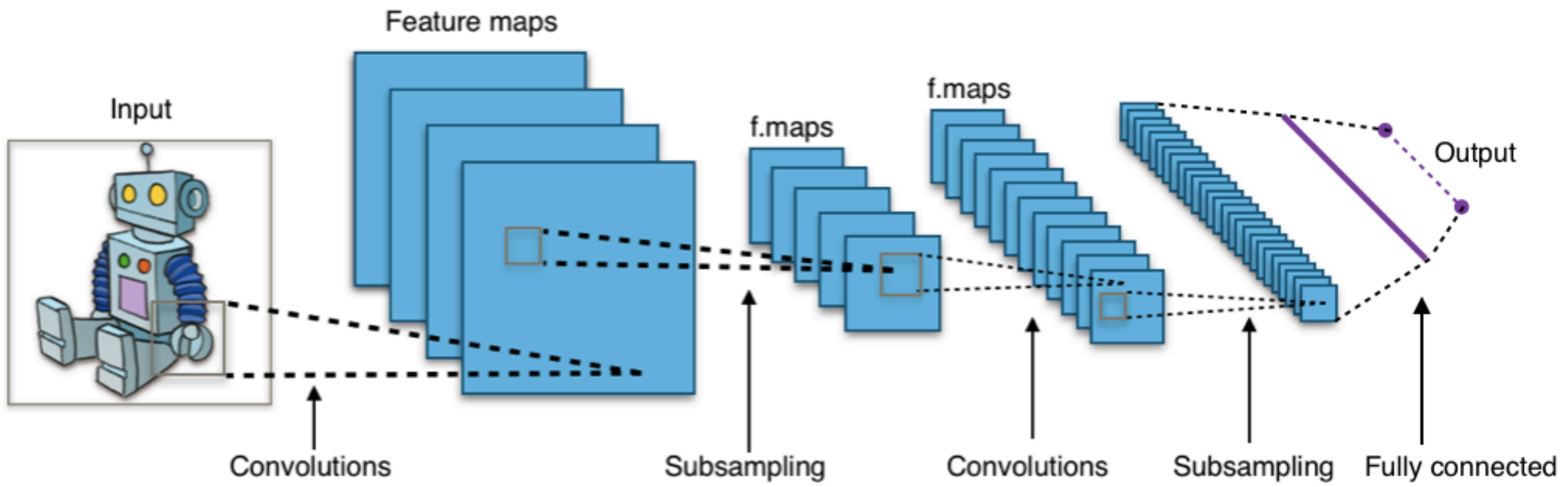


CNN on Euclidean data (I)

- **Main assumption:** data lives on a grid



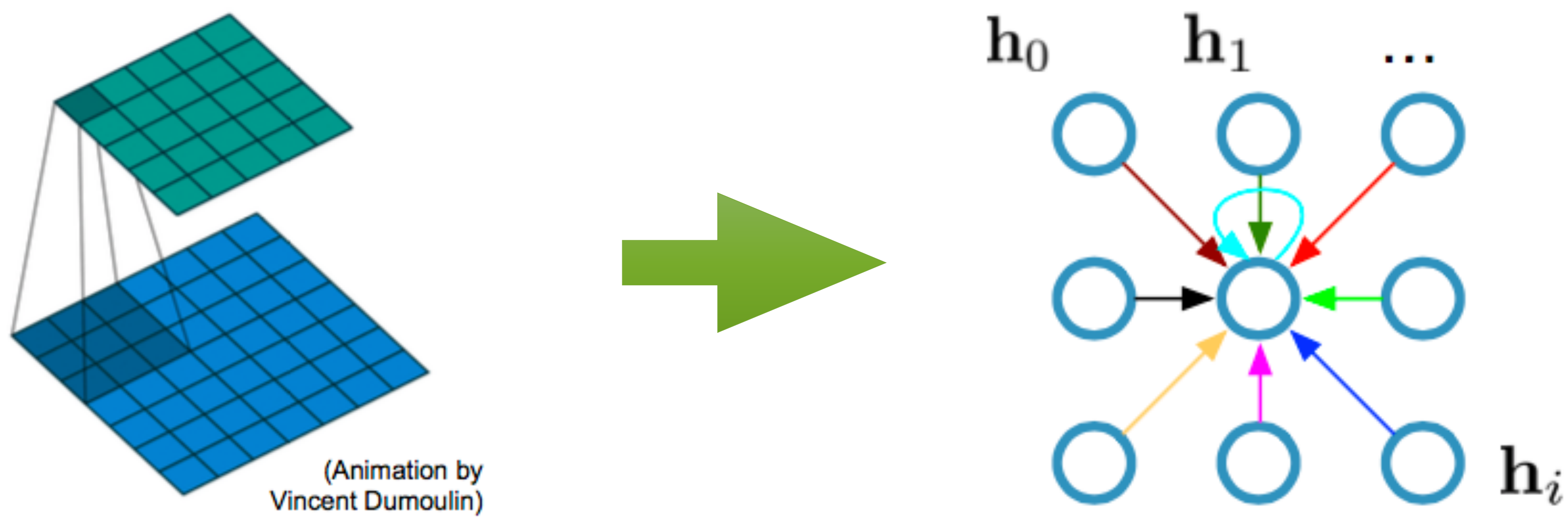
- Typical CNN architecture:



https://en.wikipedia.org/wiki/File:Typical_cnn.png

CNN on Euclidean data (II)

- Single CNN layer with 3x3 filter:



$$h_4^{(l+1)} = \sigma(\sum_{i=0}^{8} W_i^{(l)} h_i^{(l)})$$

Non-linearity   Filter weights

How can we extend CNN on graphs?

- Desirable properties

  - **Convolution:** how to achieve translation invariance
  - **Localization:** what is the notion of locality
  - **Graph pooling:** how to downsample on graphs
  - **Efficiency:** how to keep the computational complexity low
  - **Generalization:** how to build models that generalize to unseen graphs

Outline

- Motivation: Why Graph Neural Networks?

- **Basic definitions on graphs**
  - Convolution
  - Pooling

- Partial historical overview
  - Graph CNN (main focus)
  - Graph autoencoders (briefly)

- Applications
  - Naturally graph-structured data
  - Images

Graphs and signals on graphs

- **Irregular domain:** connected, undirected, weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$

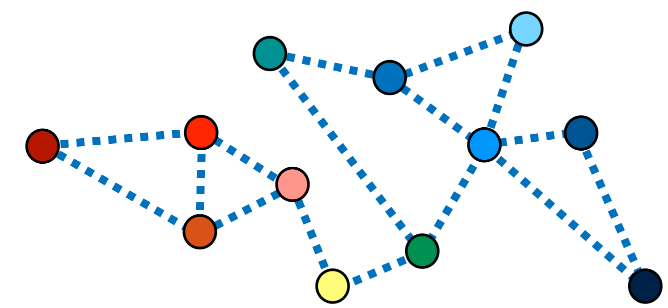
- **Graph description:**

  - Degree matrix D: diagonal matrix with sum of weights of incident edges
  - Laplacian matrix L: $L = D - W$, $L = \chi \Lambda \chi^T$
    - Complete set of orthonormal eigenvectors $\chi = [\chi_0, \chi_1, ..., \chi_{N-1}]$
    - Real, non-negative eigenvalue $0 = \lambda_0 < \lambda_1 <= \lambda_2 <= ... <= \lambda_{N-1}$

- **Graph signal:** a function $y : \mathcal{V} \rightarrow \mathbb{R}$ that assigns real values to each vertex of the graph

Graphs and signals on graphs

- **Irregular domain:** connected, undirected, weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$



- **Graph description:**
  - Degree matrix D: diagonal matrix with sum of weights of incident edges
  - Laplacian matrix L: $L = D - W$, $L = \chi \Lambda \chi^T$
    - Complete set of orthonormal eigenvectors $\chi = [\chi_0, \chi_1, ..., \chi_{N-1}]$
    - Real, non-negative eigenvalue $0 = \lambda_0 < \lambda_1 <= \lambda_2 <= ... <= \lambda_{N-1}$

- **Graph signal:** a function $y : \mathcal{V} \to \mathbb{R}$ that assigns real values to each vertex of the graph

Frequency on the graph

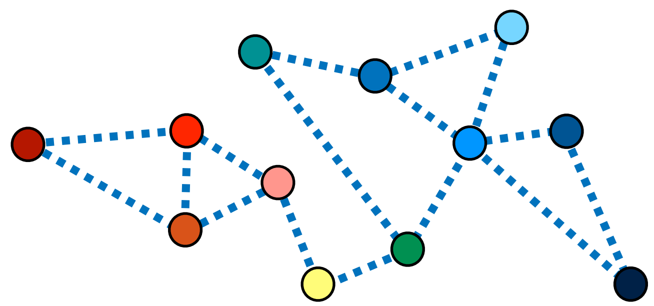
- Global smoothness on the graph

$$y^T L y = \sum_{m \in \mathcal{V}} \sum_{n \in \mathcal{N}_m} W_{m,n} [y(m) - y(n)]^2$$

Frequency on the graph

- Global smoothness on the graph

$$y^T L y = \sum_{m \in \mathcal{V}} \sum_{n \in \mathcal{N}_m} W_{m,n}[y(m) - y(n)]^2$$

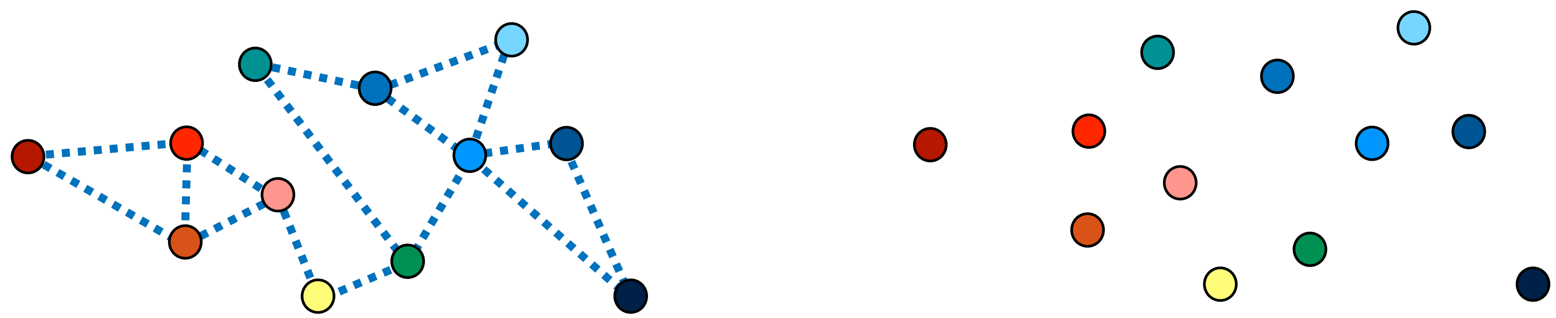


$y^T \mathcal{L}_1 y = 0.15$

Frequency on the graph

- Global smoothness on the graph

$$y^T L y = \sum_{m \in \mathcal{V}} \sum_{n \in \mathcal{N}_m} W_{m,n}[y(m) - y(n)]^2$$

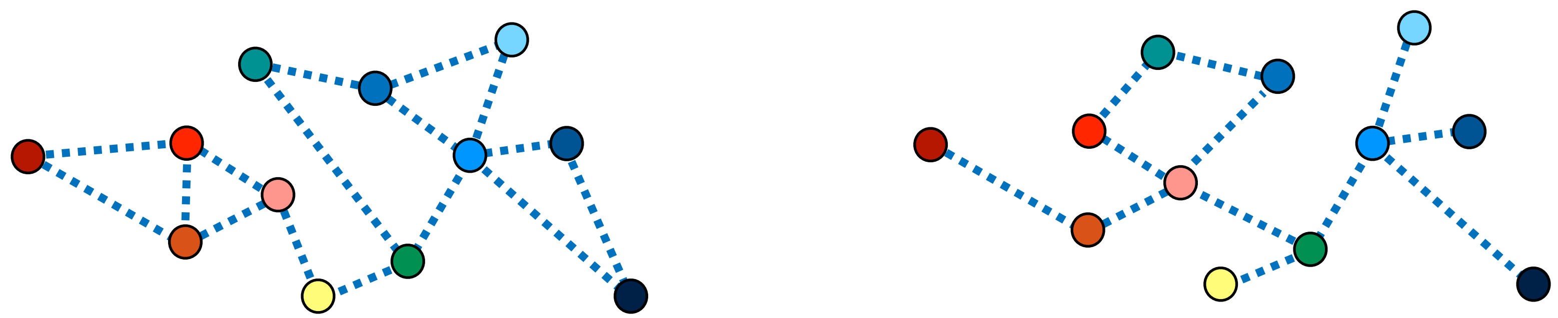


$y^T \mathcal{L}_1 y = 0.15$

Frequency on the graph

- Global smoothness on the graph

$$y^T L y = \sum_{m \in \mathcal{V}} \sum_{n \in \mathcal{N}_m} W_{m,n}[y(m) - y(n)]^2$$



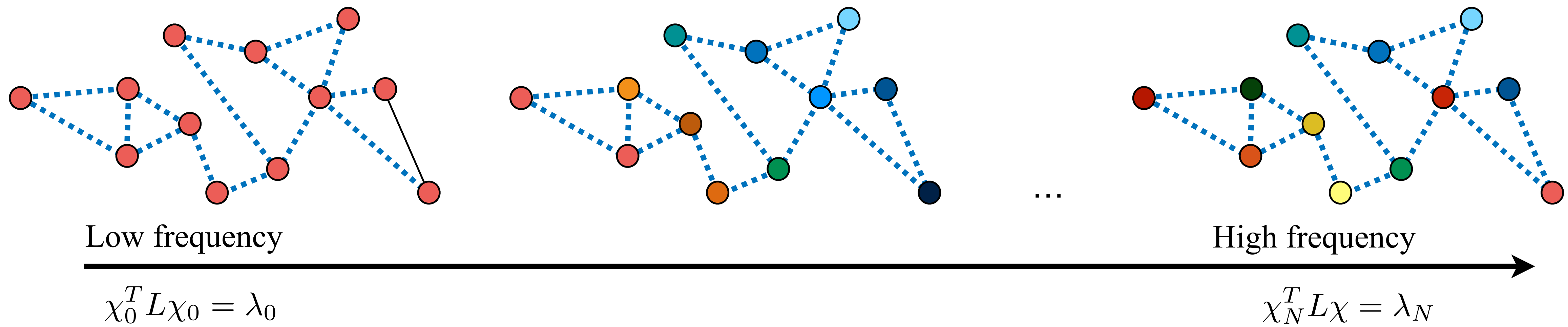
$y^T \mathcal{L}_1 y = 0.15$ $\qquad$ $y^T \mathcal{L}_2 y = 1.8$

Frequency on the graph

- Global smoothness on the graph

$$y^T L y = \sum_{m \in \mathcal{V}} \sum_{n \in \mathcal{N}_m} W_{m,n}[y(m) - y(n)]^2$$

- The eigenvectors of the Laplacian permit to define a harmonic (Fourier) analysis of graph signals

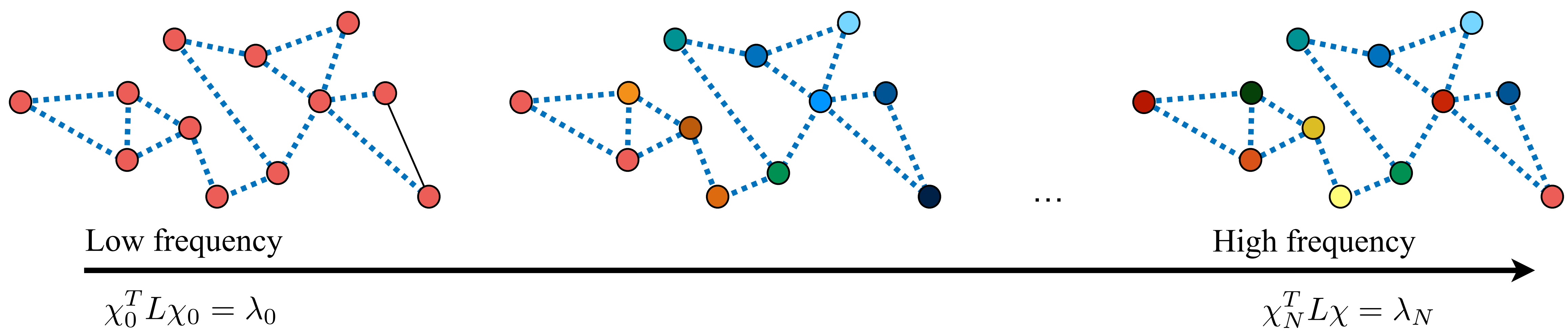


Frequency on the graph

- Global smoothness on the graph

$$y^T L y = \sum_{m \in \mathcal{V}} \sum_{n \in \mathcal{N}_m} W_{m,n}[y(m) - y(n)]^2$$

- The eigenvectors of the Laplacian permit to define a harmonic (Fourier) analysis of graph signals



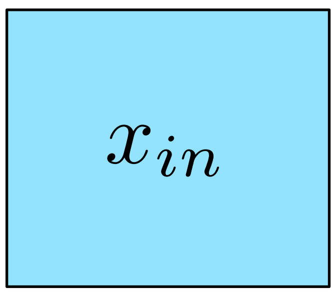
GFT $$\hat{y}(\lambda_\ell) = \langle y, \chi_\ell \rangle = \sum_{n=1}^{N} y(n)\chi_\ell^*(n), \quad \ell = 0, 1, ..., N-1$$

Filtering on graphs

- Filtering in the spectral domain with a transfer function $\hat{g}(\cdot)$

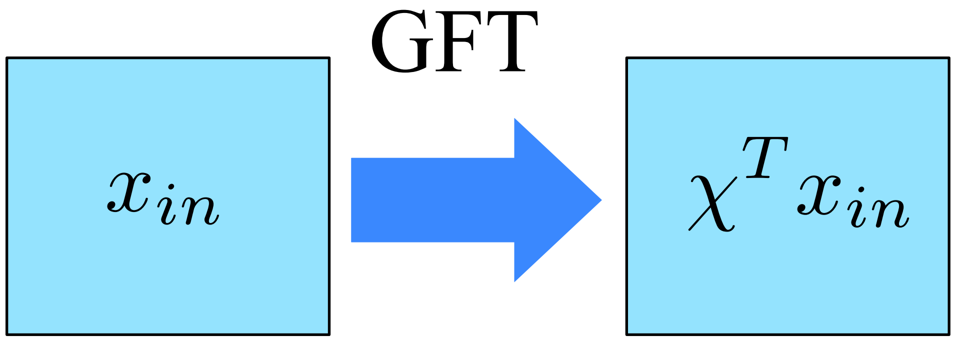
Filtering on graphs

- Filtering in the spectral domain with a transfer function $\hat{g}(\cdot)$



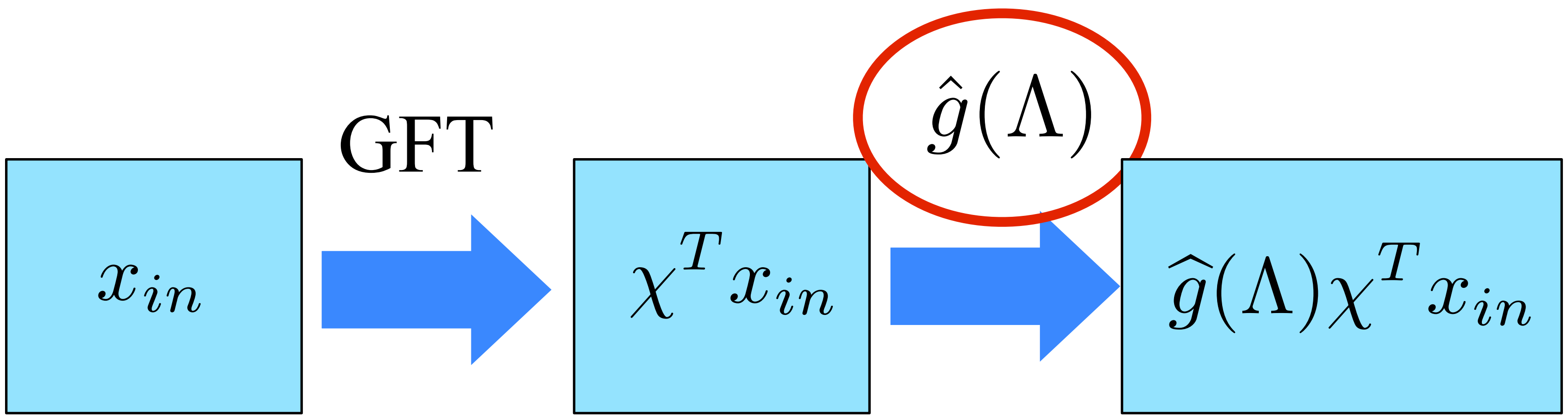
Filtering on graphs

- Filtering in the spectral domain with a transfer function $\hat{g}(\cdot)$



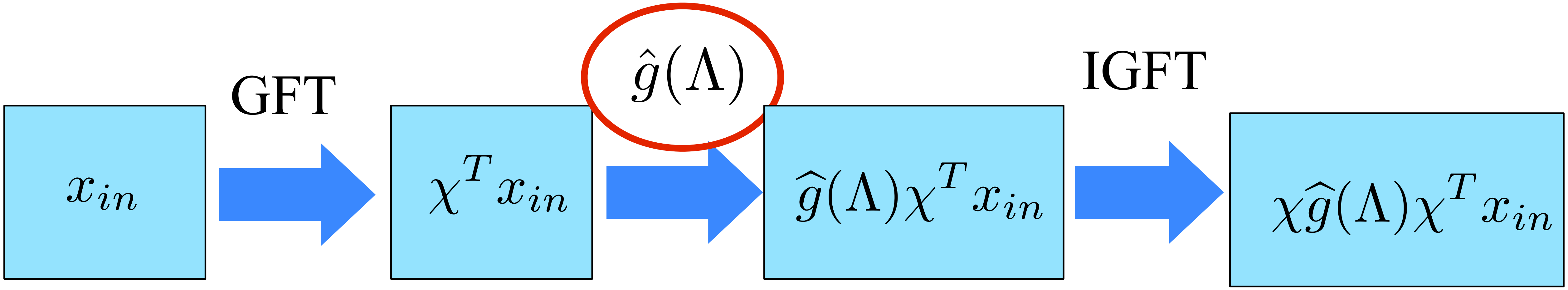
Filtering on graphs

- Filtering in the spectral domain with a transfer function $\hat{g}(\cdot)$



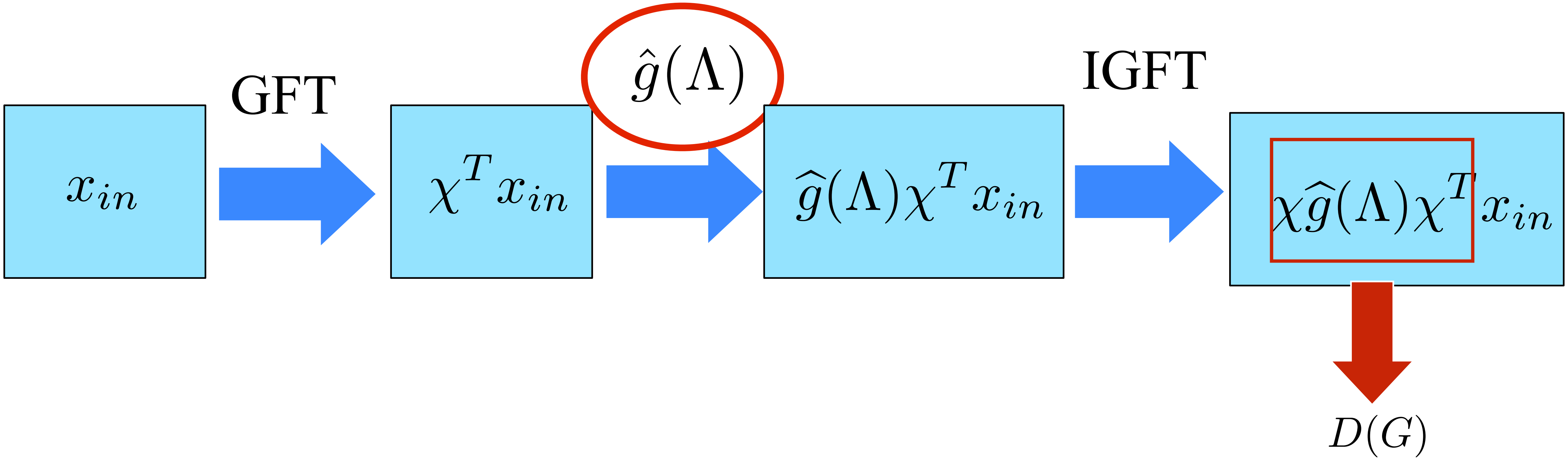
Filtering on graphs

- Filtering in the spectral domain with a transfer function $\hat{g}(\cdot)$



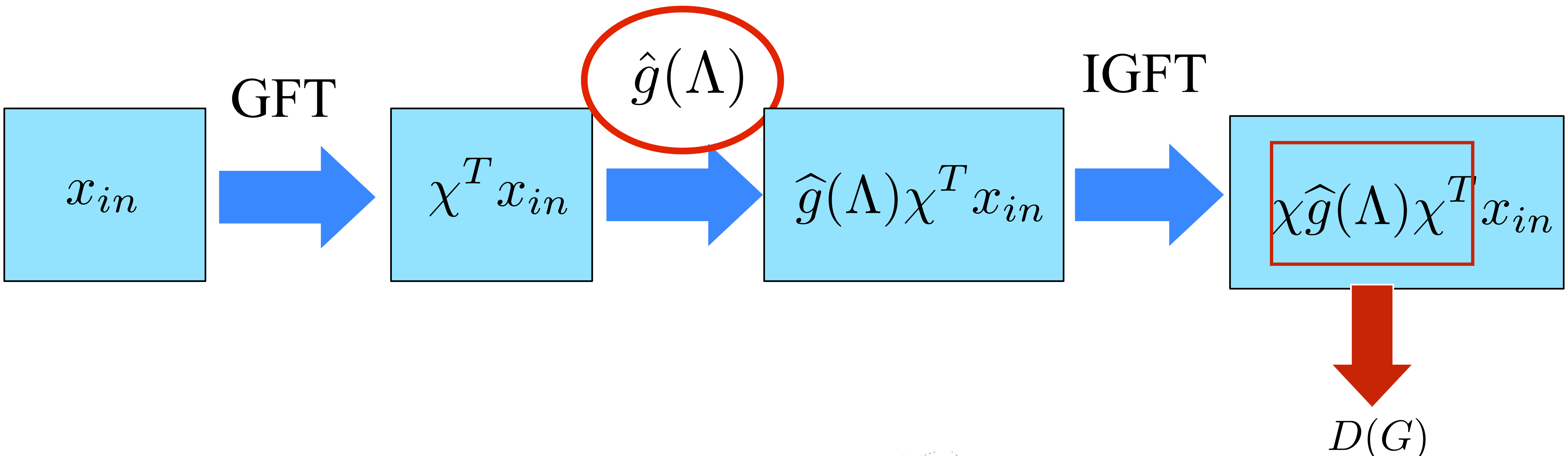
Filtering on graphs

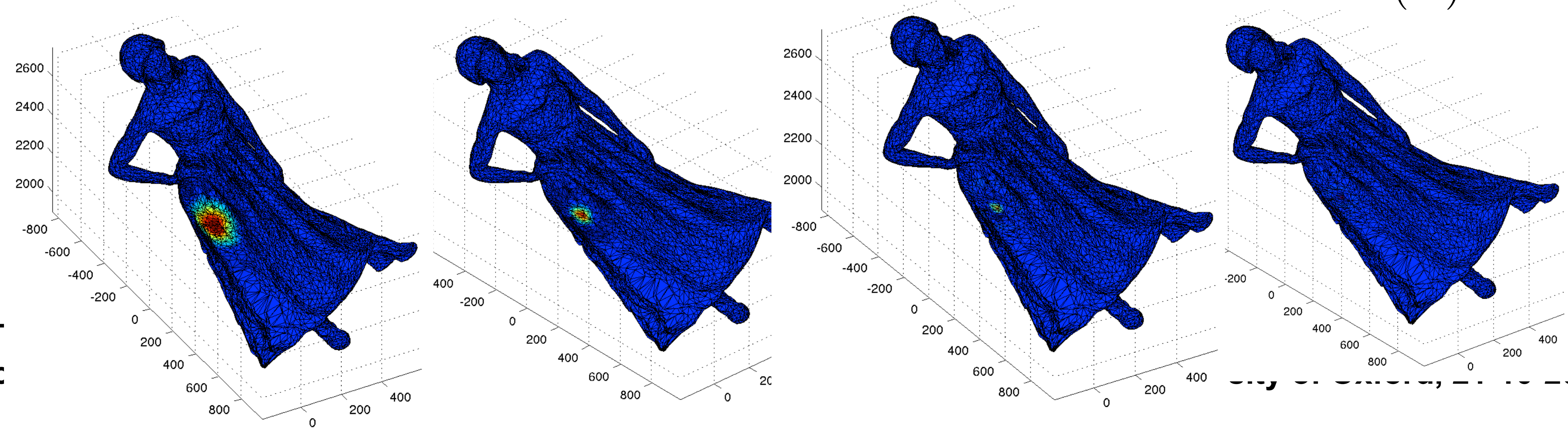
- Filtering in the spectral domain with a transfer function $\hat{g}(\cdot)$



Filtering on graphs

- Filtering in the spectral domain with a transfer function $\hat{g}(\cdot)$





Spectral convolution on graphs

Classical convolution

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

Convolution on graphs

Spectral convolution on graphs

Classical convolution

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

Convolution on graphs

Spectral convolution on graphs

Classical convolution

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

$$\widehat{(f * g)}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega)$$

Convolution on graphs

Spectral convolution on graphs

Classical convolution

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

$$\widehat{(f * g)}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega)$$

Convolution on graphs

Spectral convolution on graphs

Classical convolution

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau) g(\tau) d\tau$$

$$\widehat{(f * g)}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega)$$

Convolution on graphs

$$\widehat{(f * g)}(\lambda) = \big((\chi^T f) \circ \hat{g}\big)(\lambda)$$

Spectral convolution on graphs

Classical convolution

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

$$\widehat{(f * g)}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega)$$

Convolution on graphs

$$\widehat{(f * g)}(\lambda) = \big((\chi^T f) \circ \hat{g}\big)(\lambda)$$

Spectral convolution on graphs

Classical convolution

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

$$\widehat{(f * g)}(\omega) = \hat{f}(\omega) \cdot \hat{g}(\omega)$$

Convolution on graphs

$$f * g = \chi \hat{g}(\Lambda)\chi^T f = \hat{g}(L) f$$

$$\widehat{(f * g)}(\lambda) = \big((\chi^T f) \circ \hat{g}\big)(\lambda)$$

K-hops localization on graphs

K-hops localization on graphs

$$f * g = \chi \hat{g}(\Lambda) \chi^T f = \hat{g}(L) f$$

K-hops localization on graphs

$$f * g = \chi \hat{g}(\Lambda) \chi^T f = \hat{g}(L) f$$

$$\hat{g}_\theta(\lambda) = \sum_{j=0}^{K} \theta_j \lambda^j, \ \theta \in \mathbb{R}^{K+1}$$

K-hops localization on graphs

$$f * g = \chi \hat{g}(\Lambda) \chi^T f = \hat{g}(L) f$$

$$\hat{g}_\theta(\lambda) = \sum_{j=0}^{K} \theta_j \lambda^j, \ \theta \in \mathbb{R}^{K+1}$$ ➡

K-hops localization on graphs

$$f * g = \chi \hat{g}(\Lambda) \chi^T f = \hat{g}(L) f$$

$$\hat{g}_\theta(\lambda) = \sum_{j=0}^{K} \theta_j \lambda^j, \ \theta \in \mathbb{R}^{K+1}$$

$\Rightarrow$

$$\hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j L^j$$

K-hops localization on graphs

$$f * g = \chi \hat{g}(\Lambda)\chi^T f = \hat{g}(L) f$$

$$\hat{g}_\theta(\lambda) = \sum_{j=0}^{K} \theta_j \lambda^j, \ \theta \in \mathbb{R}^{K+1}$$

→

$$\hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j L^j$$

↓

K-hops localization on graphs

$$f * g = \chi \hat{g}(\Lambda) \chi^T f = \hat{g}(L) f$$

$$\hat{g}_\theta(\lambda) = \sum_{j=0}^{K} \theta_j \lambda^j, \ \theta \in \mathbb{R}^{K+1}$$

→

$$\hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j L^j$$

↓

Localization within K-hop neighborhood

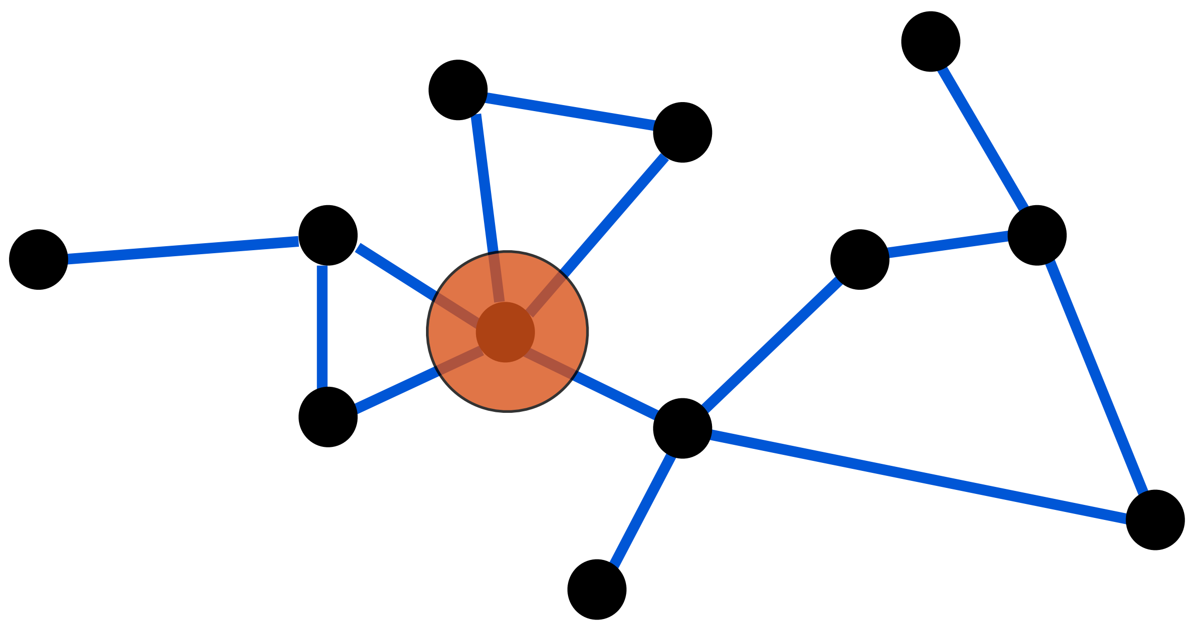
K-hops localization on graphs

$$f * g = \chi \hat{g}(\Lambda) \chi^T f = \hat{g}(L) f$$

$$\hat{g}_\theta(\lambda) = \sum_{j=0}^{K} \theta_j \lambda^j, \; \theta \in \mathbb{R}^{K+1}$$

$$\hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j L^j$$

Localization within K-hop neighborhood

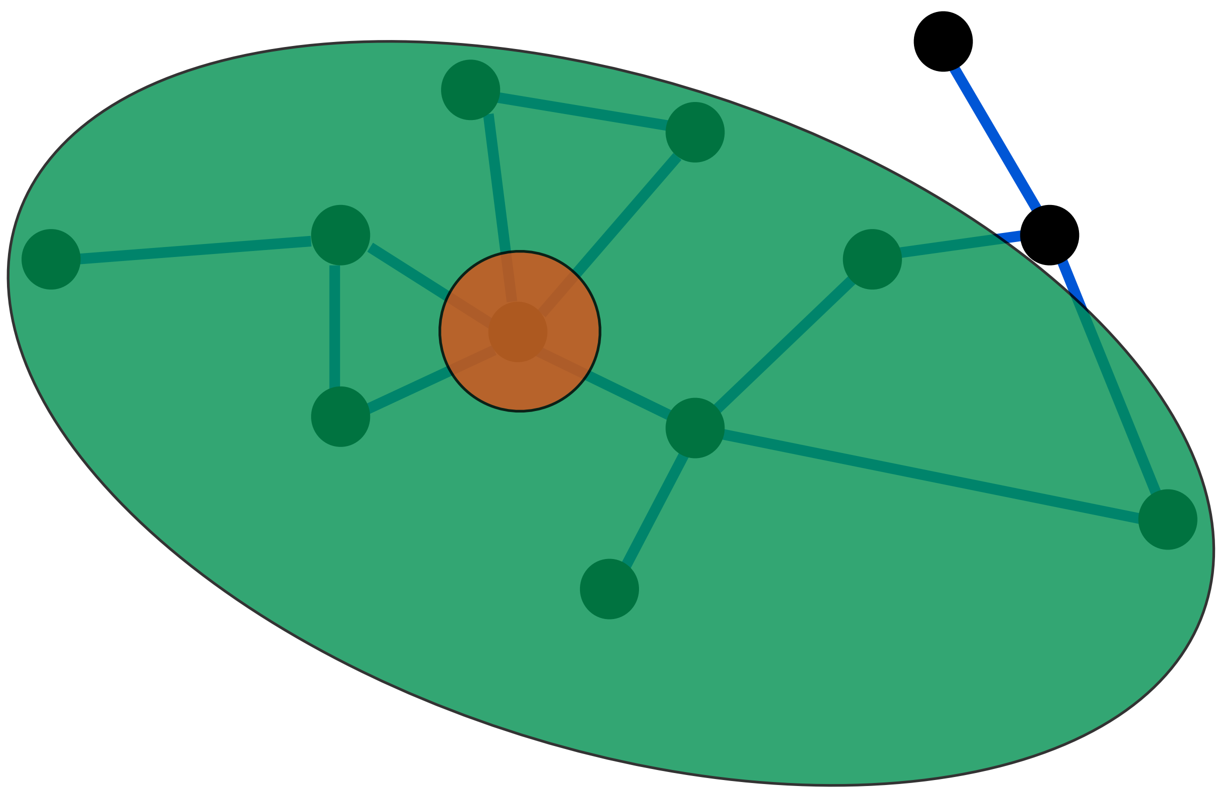


K-hops localization on graphs

$$f * g = \chi \hat{g}(\Lambda) \chi^T f = \hat{g}(L) f$$

$$\hat{g}_\theta(\lambda) = \sum_{j=0}^{K} \theta_j \lambda^j, \ \theta \in \mathbb{R}^{K+1}$$

$$\hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j L^j$$



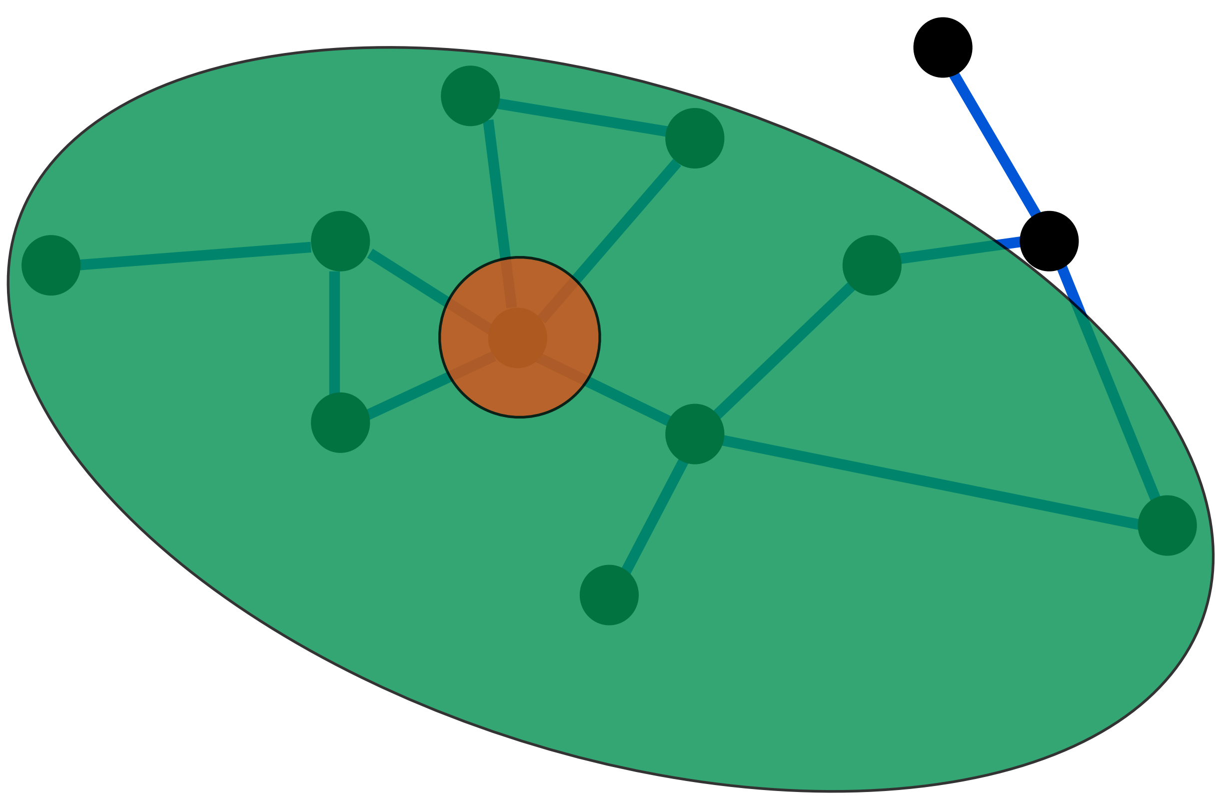
Localization within K-hop neighborhood

K-hops localization on graphs

$$f * g = \chi \hat{g}(\Lambda)\chi^T f = \hat{g}(L) f$$

$$\hat{g}_\theta(\lambda) = \sum_{j=0}^{K} \theta_j \lambda^j, \ \theta \in \mathbb{R}^{K+1}$$

$$\hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j L^j$$



Localization within K-hop neighborhood

- **Convolution on graphs:**
  - spectrally motivated
  - spatially implemented

K-hops localization on graphs

$$f * g = \chi \hat{g}(\Lambda) \chi^T f = \hat{g}(L) f$$

$$\hat{g}_\theta(\lambda) = \sum_{j=0}^{K} \theta_j \lambda^j, \ \theta \in \mathbb{R}^{K+1}$$

→

$$\hat{g}_\theta(L) = \sum_{j=0}^{K} \theta_j L^j$$

↓

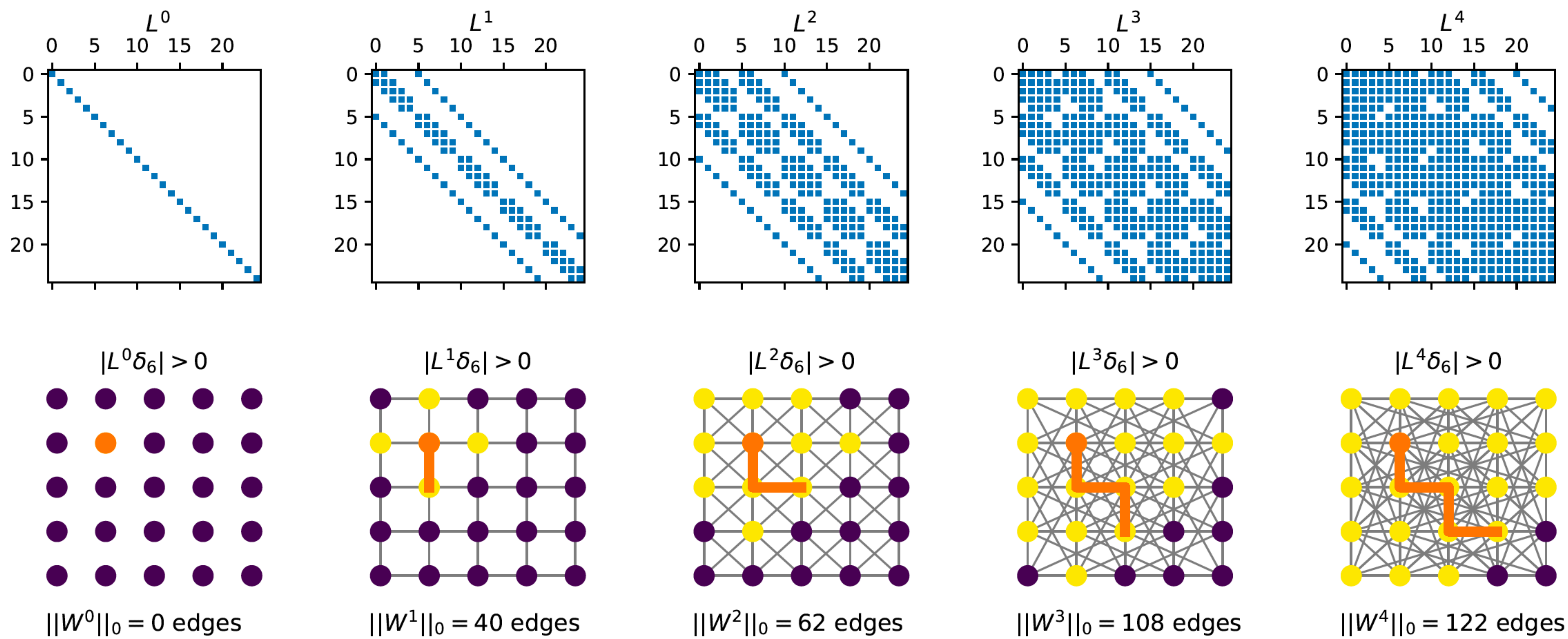
Localization within K-hop neighborhood

↓

- **Convolution on graphs:**
  - spectrally motivated
  - spatially implemented

Powers of the graph Laplacian

L^k defines the k-neighborhood



Localization: $d_{\mathcal{G}}(v_i, v_j) > K$ implies $(L^K)_{ij} = 0$

Slide made by M. Defferrard

Spatial convolution on graphs

Classical convolution

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau) g(\tau) d\tau$$

Convolution on graphs

$$(f * g)_n = \sum_{m \in \mathcal{V}} f(m) g(m,n)$$

function value at neighbors

- Defined as a weighted sum of function values at neighboring nodes

Spatial convolution on graphs

Classical convolution

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau) g(\tau) d\tau$$

Convolution on graphs

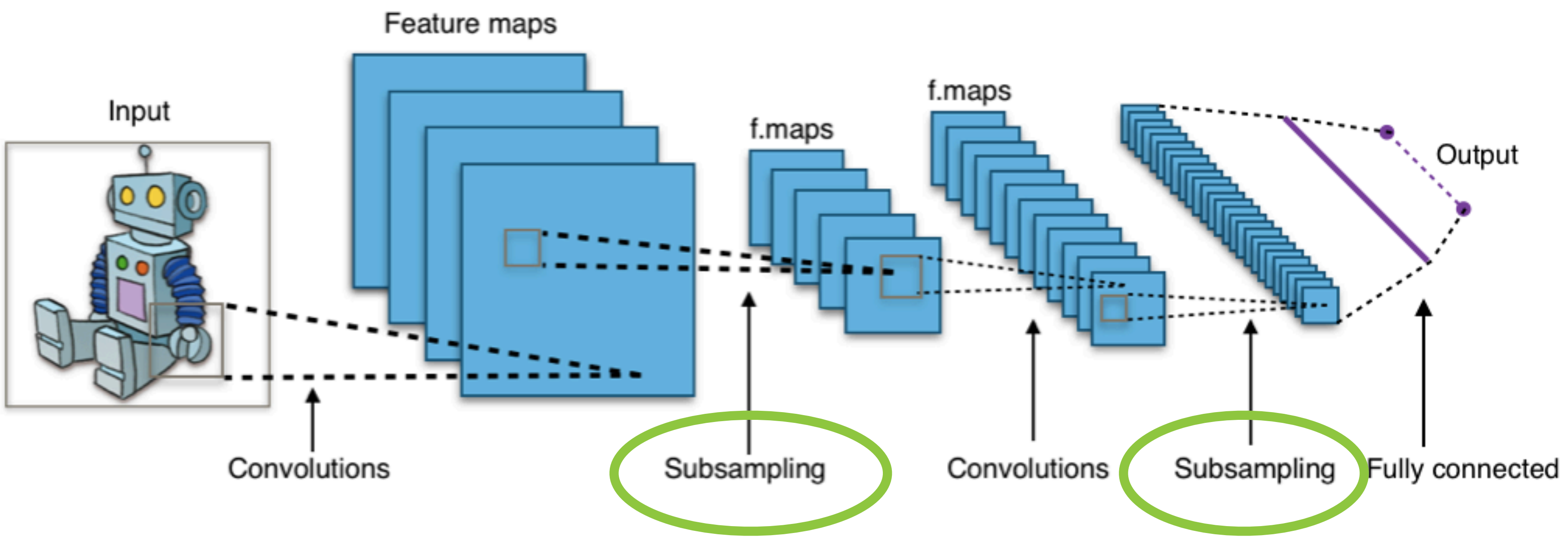
$$(f * g)_n = \sum_{m \in \mathcal{V}} f(m) g(m, n)$$

function value at neighbors

weighting function/nodes' similarity

- Defined as a weighted sum of function values at neighboring nodes

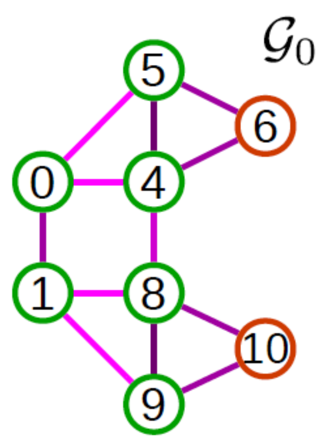
How to define pooling on graphs?



- A relatively open question, with ongoing research
- Methods can be grouped in three main categories:
  - topology based pooling
  - global pooling
  - hierarchical pooling

Topology based pooling

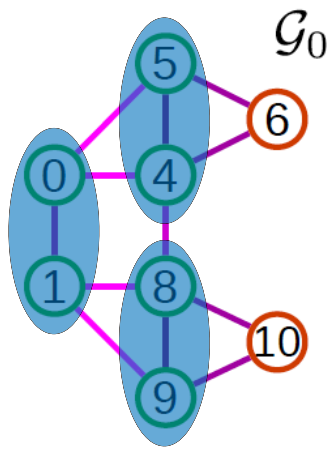
- Multi-scale graph coarsening: no features involved



- Graclus algorithm (Dhillon et al. 2007)

Topology based pooling

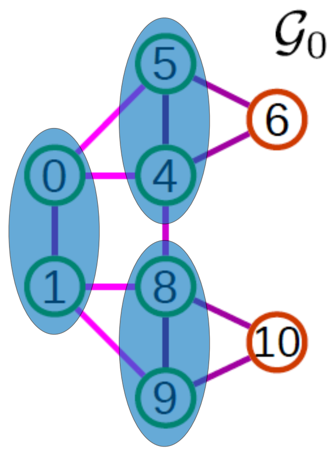
- Multi-scale graph coarsening: no features involved



- Graclus algorithm (Dhillon et al. 2007)

Topology based pooling

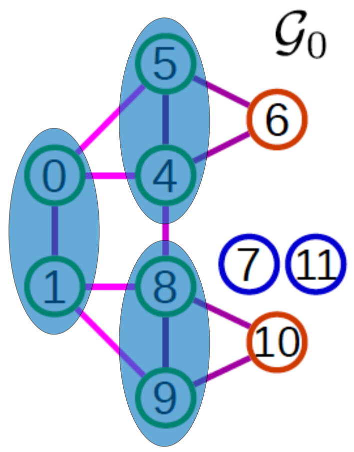
- Multi-scale graph coarsening: no features involved



- Graclus algorithm (Dhillon et al. 2007)
  - Local greedy way of merging vertices that minimises the normalised cut

Topology based pooling

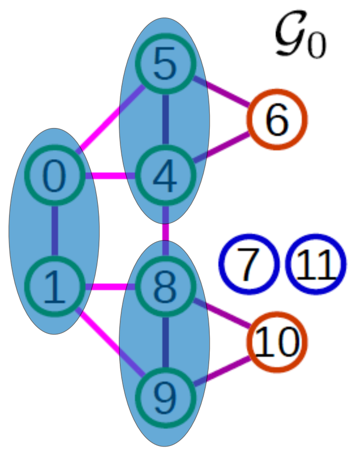
- Multi-scale graph coarsening: no features involved



- Graclus algorithm (Dhillon et al. 2007)
  - Local greedy way of merging vertices that minimises the normalised cut

Topology based pooling

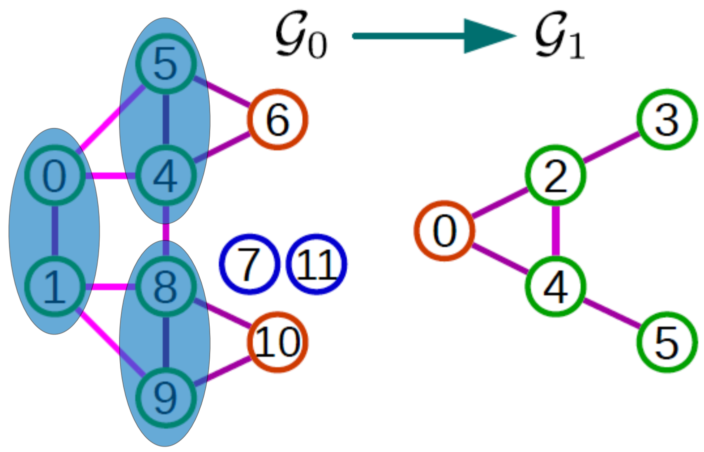
- Multi-scale graph coarsening: no features involved



- Graclus algorithm (Dhillon et al. 2007)
  - Local greedy way of merging vertices that minimises the normalised cut
  - Add artificial nodes to ensure two children for each vertex

Topology based pooling

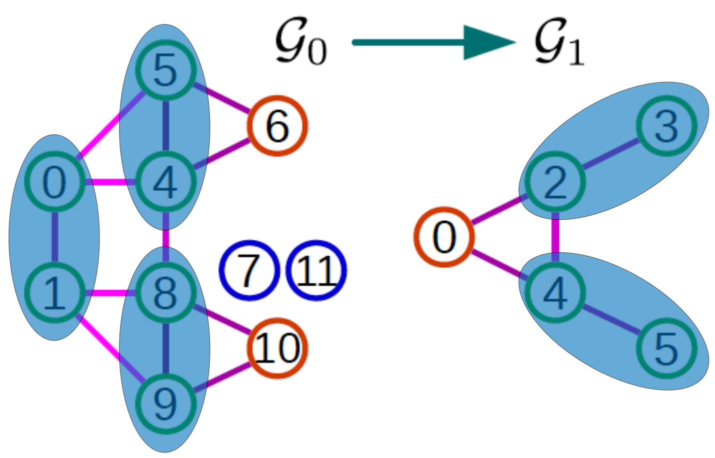
- Multi-scale graph coarsening: no features involved



- Graclus algorithm (Dhillon et al. 2007)
  - Local greedy way of merging vertices that minimises the normalised cut
  - Add artificial nodes to ensure two children for each vertex

Topology based pooling

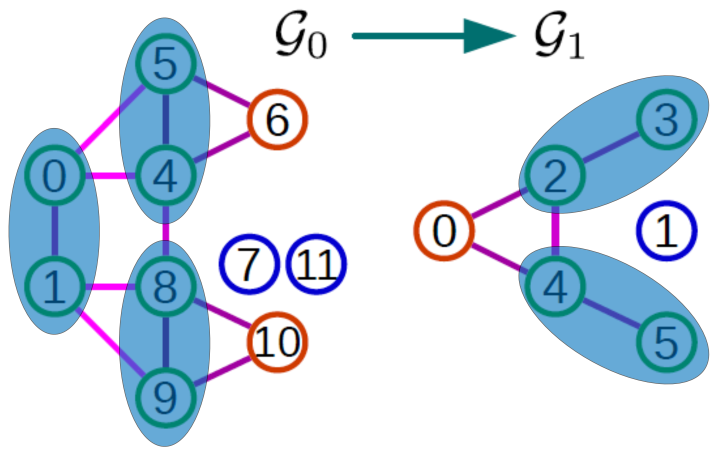
- Multi-scale graph coarsening: no features involved



- Graclus algorithm (Dhillon et al. 2007)
  - Local greedy way of merging vertices that minimises the normalised cut
  - Add artificial nodes to ensure two children for each vertex

Topology based pooling

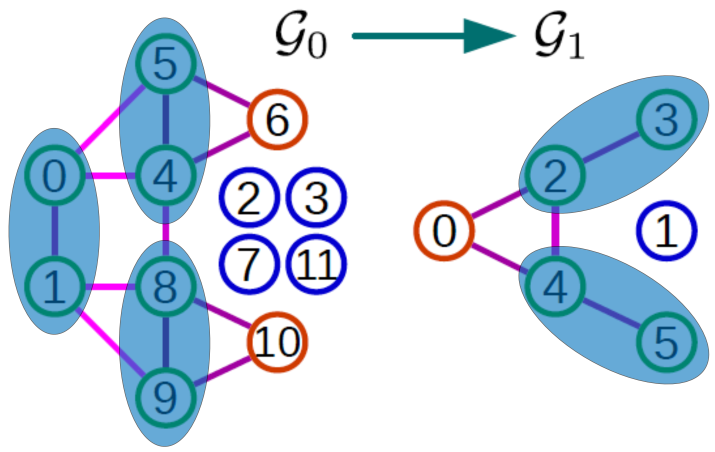
- Multi-scale graph coarsening: no features involved



- Graclus algorithm (Dhillon et al. 2007)
  - Local greedy way of merging vertices that minimises the normalised cut
  - Add artificial nodes to ensure two children for each vertex

Topology based pooling

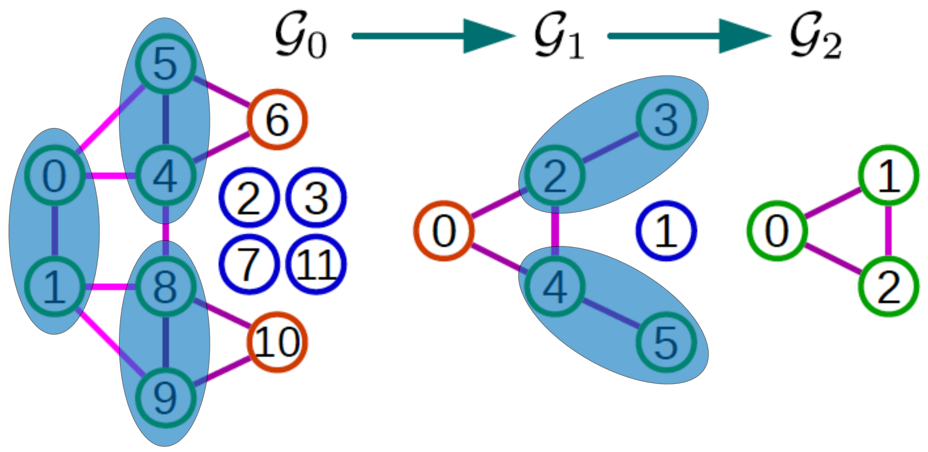
- Multi-scale graph coarsening: no features involved



- Graclus algorithm (Dhillon et al. 2007)
  - Local greedy way of merging vertices that minimises the normalised cut
  - Add artificial nodes to ensure two children for each vertex

Topology based pooling

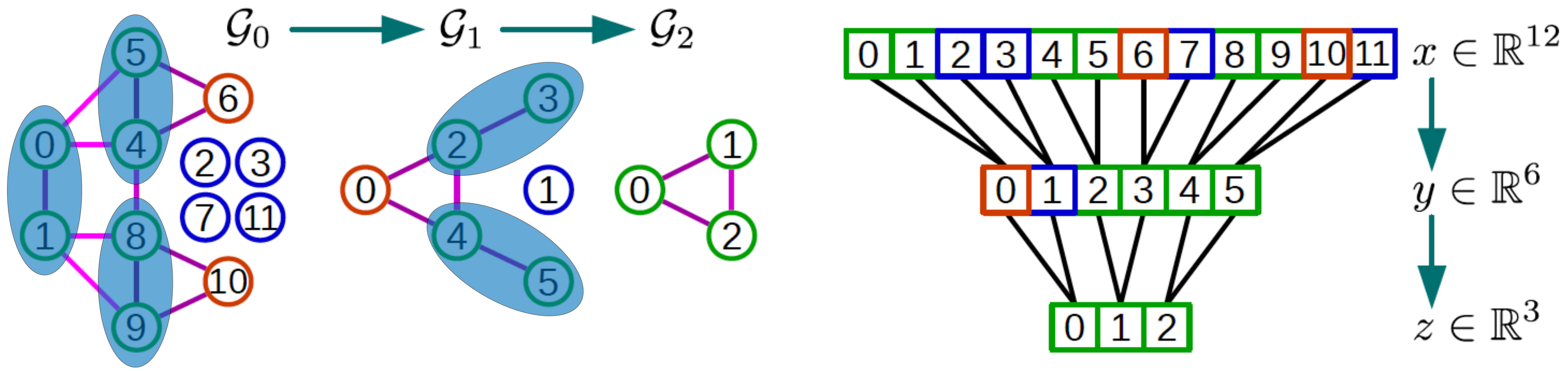
- Multi-scale graph coarsening: no features involved



- Graclus algorithm (Dhillon et al. 2007)
  - Local greedy way of merging vertices that minimises the normalised cut
  - Add artificial nodes to ensure two children for each vertex

Topology based pooling

- Multi-scale graph coarsening: no features involved



Defferrard et al. 2016

- Graclus algorithm (Dhillon et al. 2007)
    - Local greedy way of merging vertices that minimises the normalised cut
    - Add artificial nodes to ensure two children for each vertex
    - 1D grid pooling: [max(0,1), max(4,5,6), max(8,9,10)]

Global pooling

- Involves node features
- Uses sum/max or neural networks to pool all representation of nodes

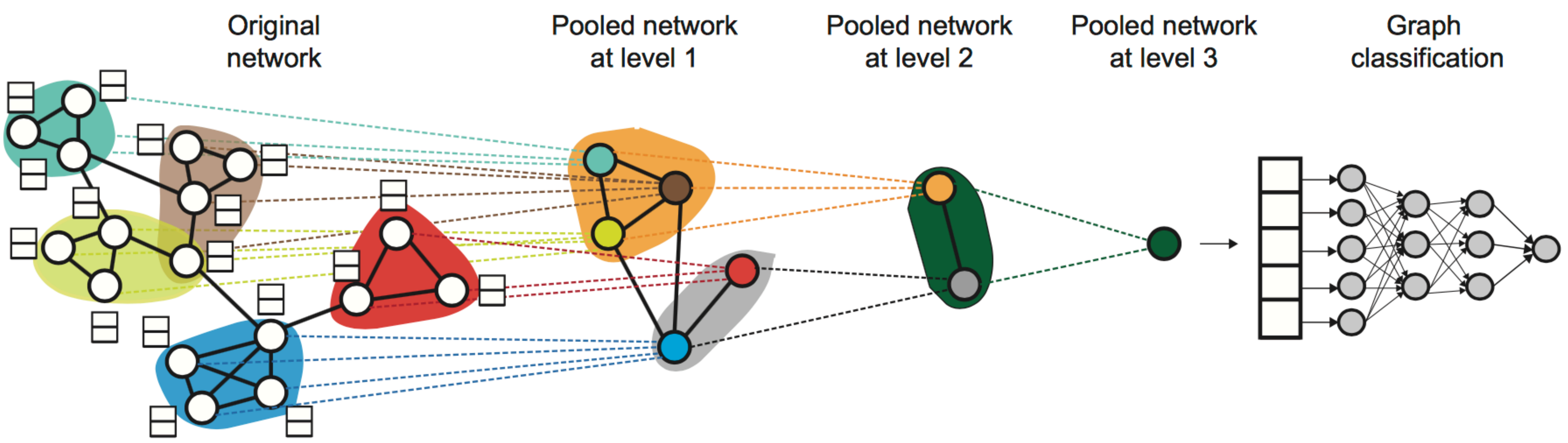
$$h_G = mean/max/sum(h_1^{(K)}, h_2^{(K)}, ..., h_N^{(K)})$$

- Also known as READOUT

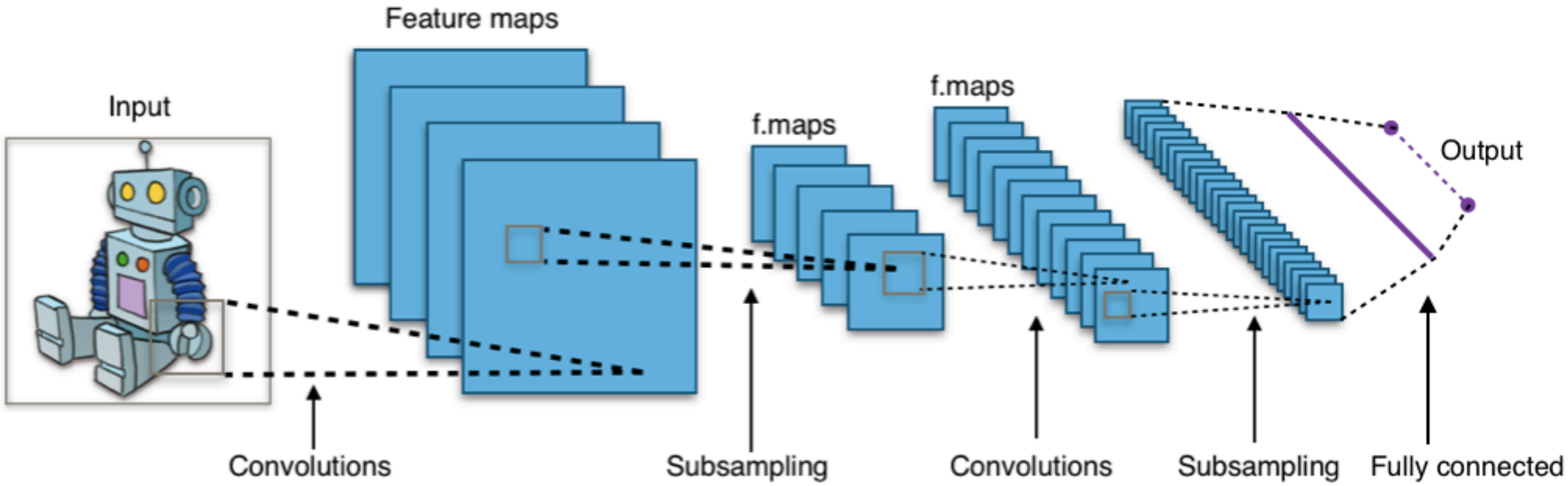
- Example: SortPool (Zhang et al. 2018)
  - sorts embeddings for nodes according to the structural roles of a graph

Hierarchical pooling

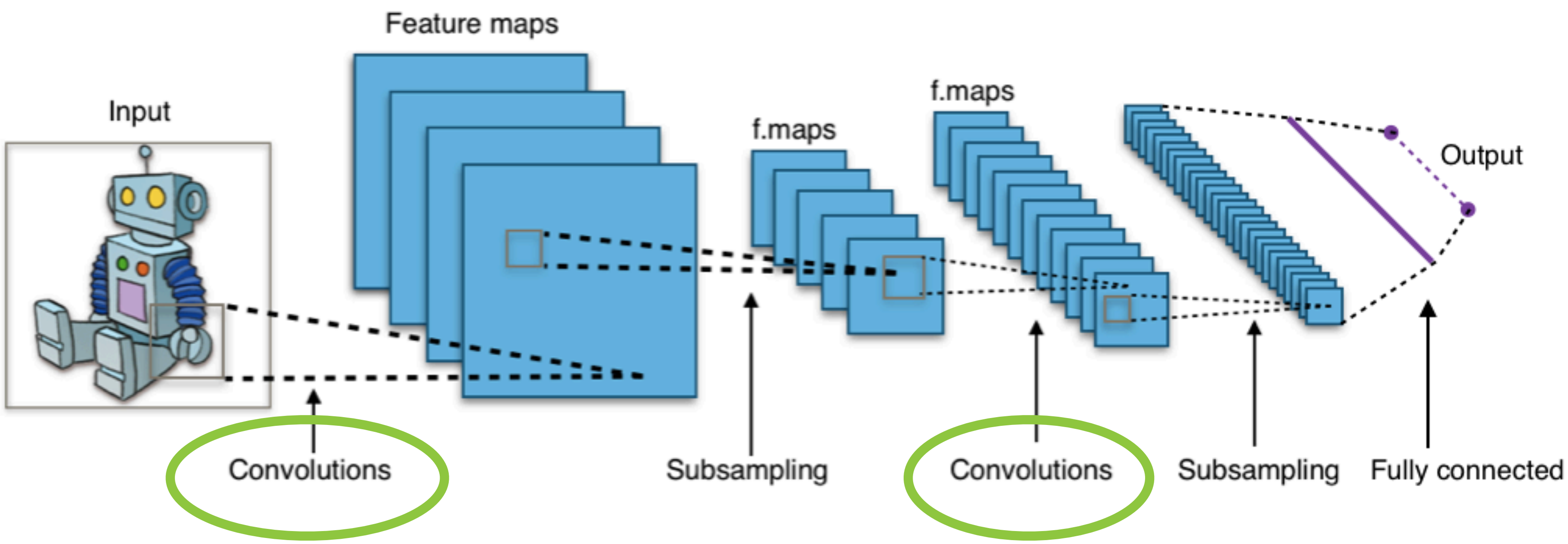
- Involved nodes features
- Aggregate information in a hierarchical way that respects the graph structure
- Results in cluster selection
- Example: DiffPool [Ying et al. 2019]



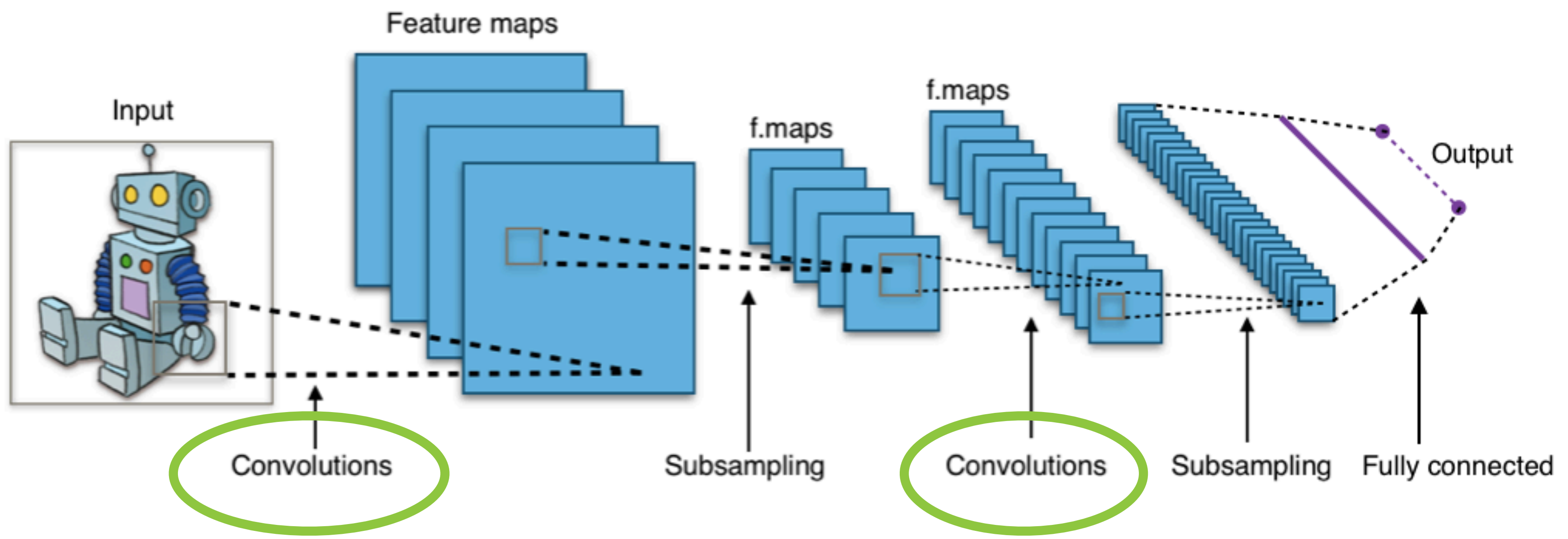
Summary so far



Summary so far

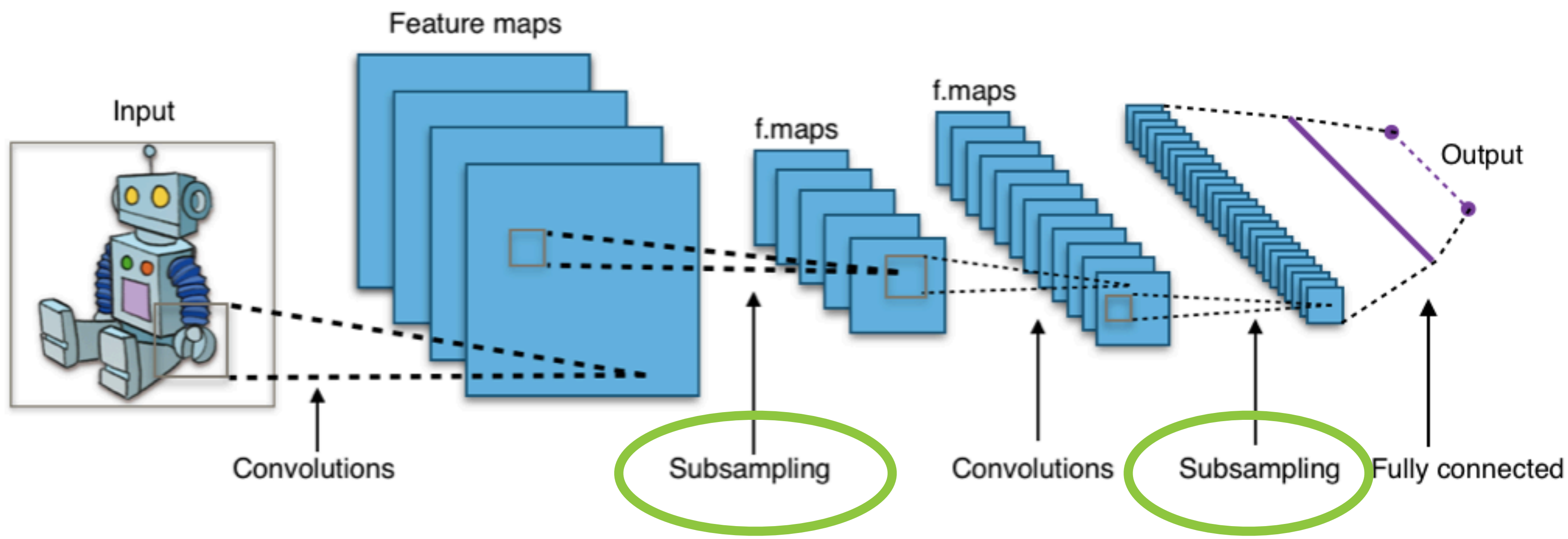


Summary so far



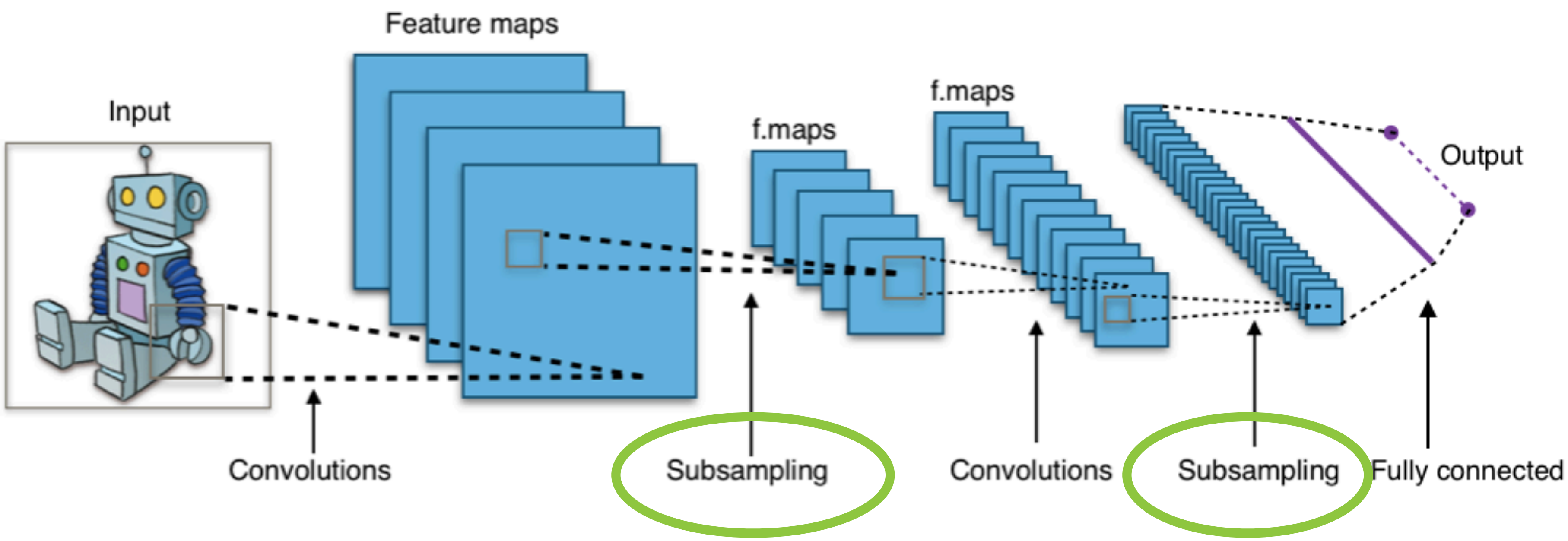
- Graph convolution: spectral or spatial

Summary so far



- Graph convolution: spectral or spatial

Summary so far

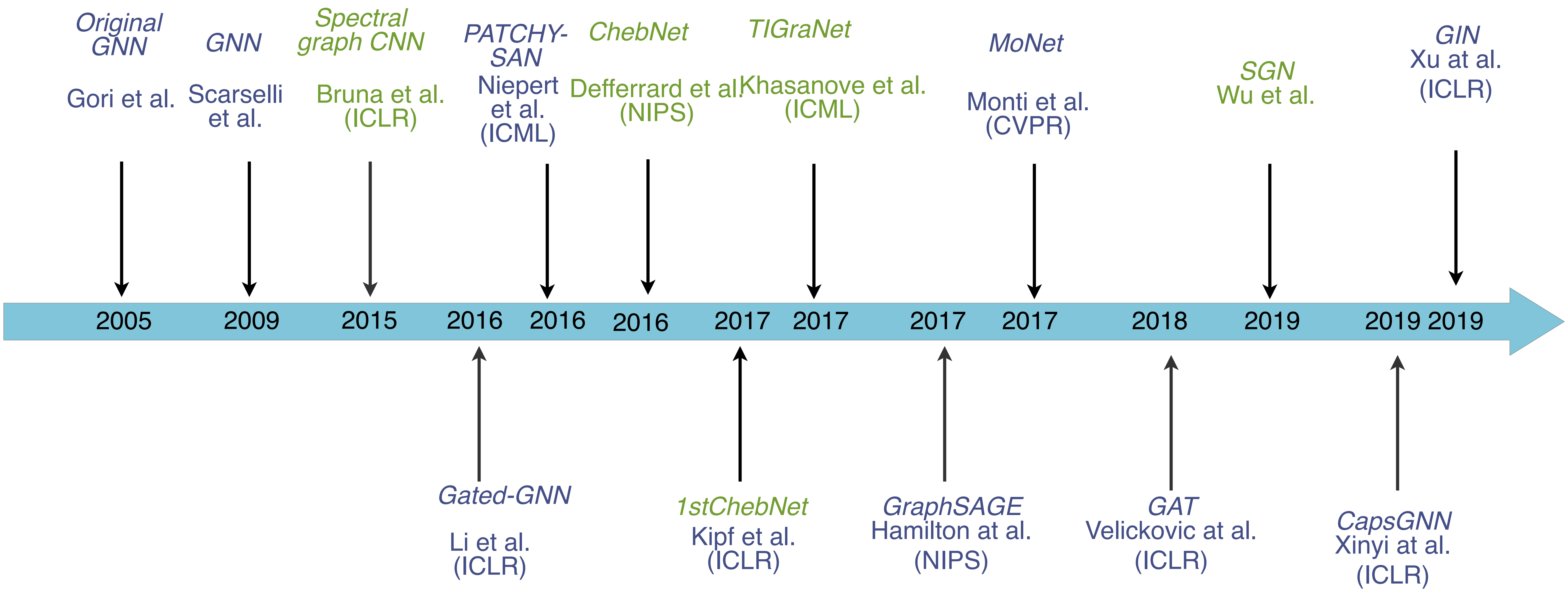


- Graph convolution: spectral or spatial
- Graph pooling: structural, global, or hierarchical

Outline

- Motivation: Why Graph Neural Networks?
- Basic definitions on graphs
- Partial historical overview
  - **Graph CNN (main focus)**
  - Graph autoencoders (briefly)
- Applications
  - Naturally graph-structured data
  - Images

Partial historical overview

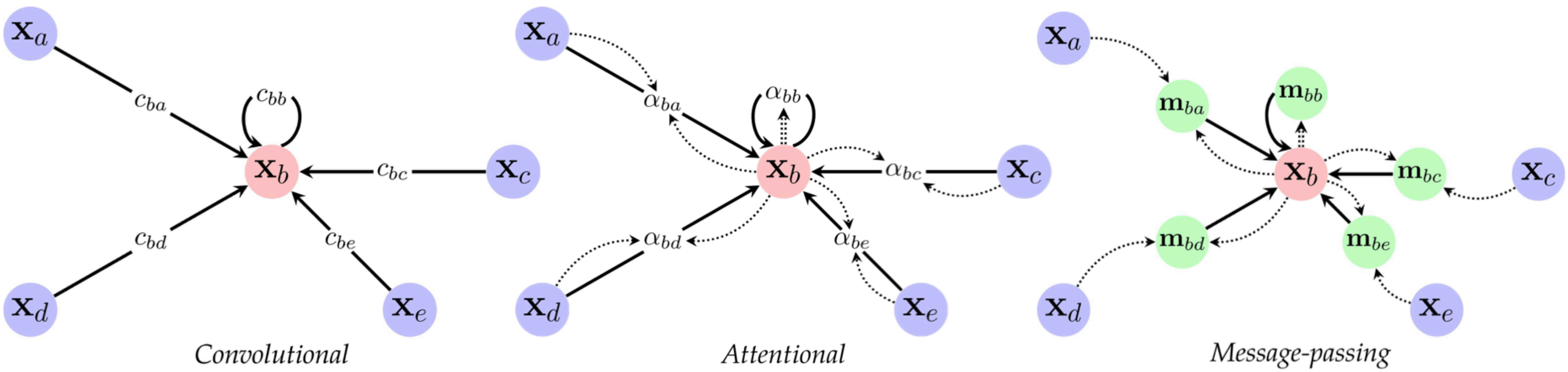


Spatial-based methods Spectral-based methods

- Recent trends in introducing GSP-inspired architectures
  - GraphHeat (Xu'19), GWNN (Xu'19), SIGN (Frasca'20), DGN (Beaini'20), Spectral GNs (Stachenfeld'20), Framelets (Zheng'21), FAGCN (Bo'21) …

Balcilar et al., "Analyzing the expressive power of graph neural networks in a spectral perspective," ICLR, 2021
Bronstein et al., Geometric deep learning: Grids, Groups, Graphs, Geodesics, and Gauges, arXiv, 2021

GNNs in one slide



$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij}\psi(\mathbf{x}_j)\right)$$

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j)\psi(\mathbf{x}_j)\right)$$

$$\mathbf{h}_i = \phi\left(\mathbf{x}_i, \bigoplus_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

ChebyNet
GCN
SGC

MoNet
GAT

MPNN
GraphNets

Slide taken from P. Veličković

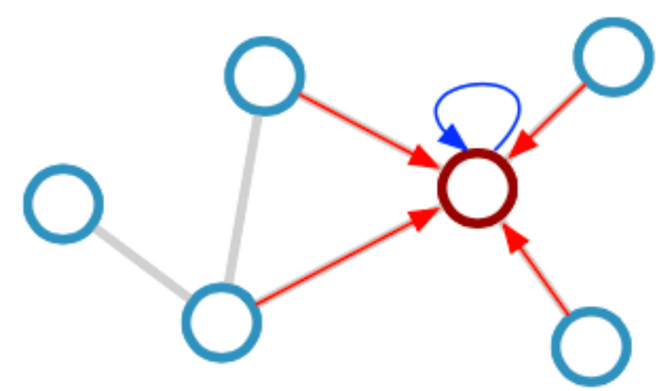
Spatial approaches in one slide

- Generate node embeddings based on local neighborhoods
  - nodes aggregate information from their neighbors using neural networks

- Feed the embeddings into a loss function

- **Key difference:** how nodes aggregate information across layers

$$h_i^{(k+1)} = \sigma(W^{(k)} h_i^{(k)} + Q^{(k)} \sum_{n \in \mathcal{N}_i} h_n^{(k)})$$



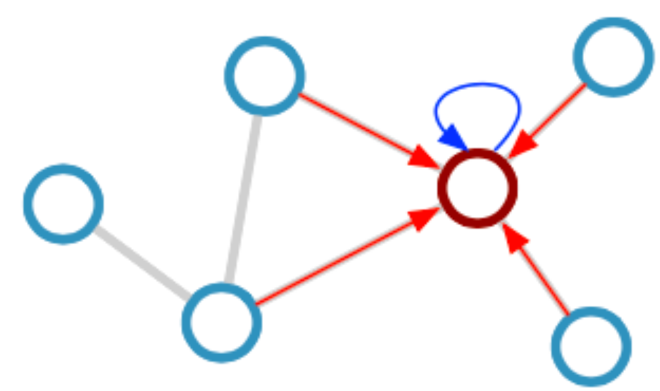
Spatial approaches in one slide

- Generate node embeddings based on local neighborhoods
  - nodes aggregate information from their neighbors using neural networks

- Feed the embeddings into a loss function

- **Key difference:** how nodes aggregate information across layers

$$h_i^{(k+1)} = \sigma\left(W^{(k)} h_i^{(k)} + Q^{(k)} \sum_{n \in \mathcal{N}_i} h_n^{(k)}\right)$$



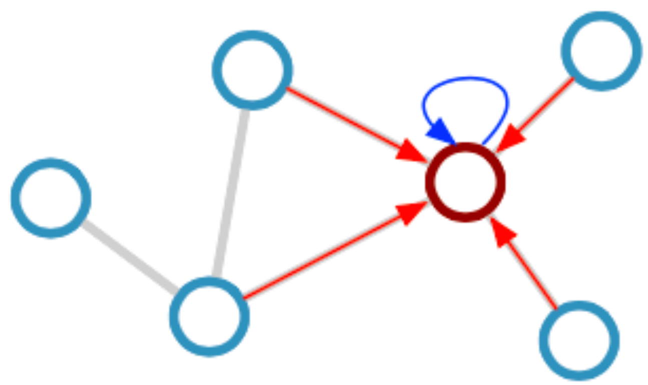
Spatial approaches in one slide

- Generate node embeddings based on local neighborhoods
  - nodes aggregate information from their neighbors using neural networks

- Feed the embeddings into a loss function

- **Key difference:** how nodes aggregate information across layers

$$h_i^{(k+1)} = \sigma\left(W^{(k)} h_i^{(k)} + Q^{(k)} \sum_{n \in \mathcal{N}_i} h_n^{(k)}\right)$$



Spatial approaches in one slide

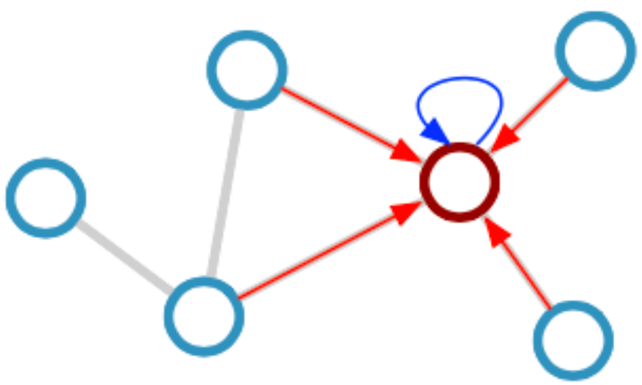
- Generate node embeddings based on local neighborhoods
  - nodes aggregate information from their neighbors using neural networks

- Feed the embeddings into a loss function

- **Key difference:** how nodes aggregate information across layers

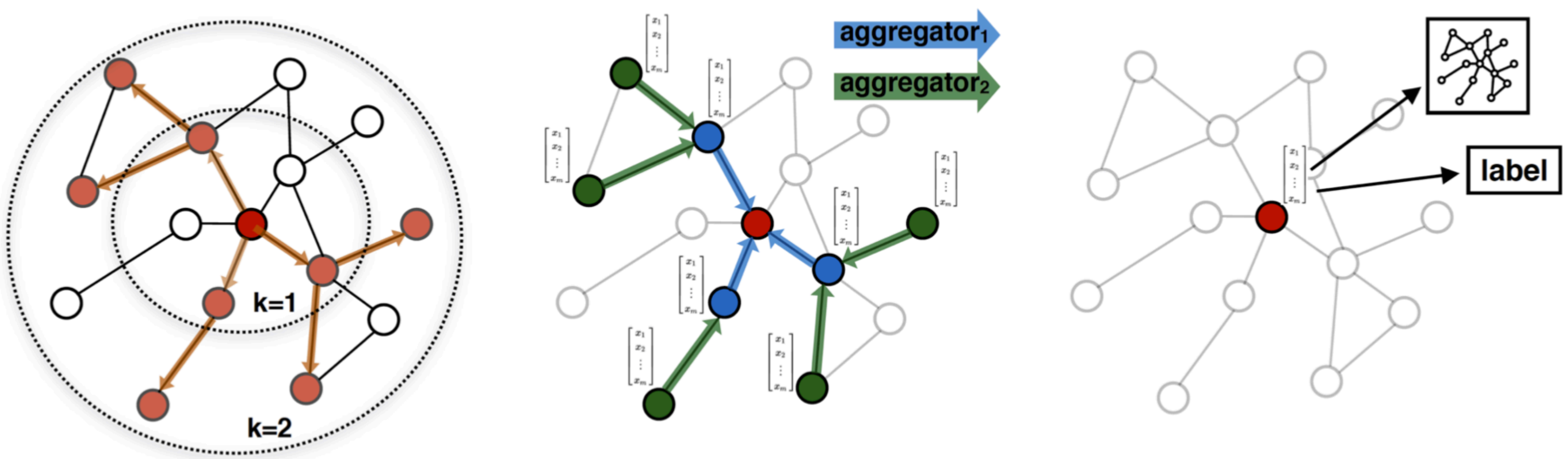
$$h_i^{(k+1)} = \sigma\left(W^{(k)} h_i^{(k)} + Q^{(k)} \sum_{n \in \mathcal{N}_i} h_n^{(k)}\right)$$

Trainable parameters



GraphSAGE

- Node's neighborhood defines a computational graph
- Each edge in this graph is a transformation/aggregator function
- Cross-entropy loss
- 2-3 layers deep

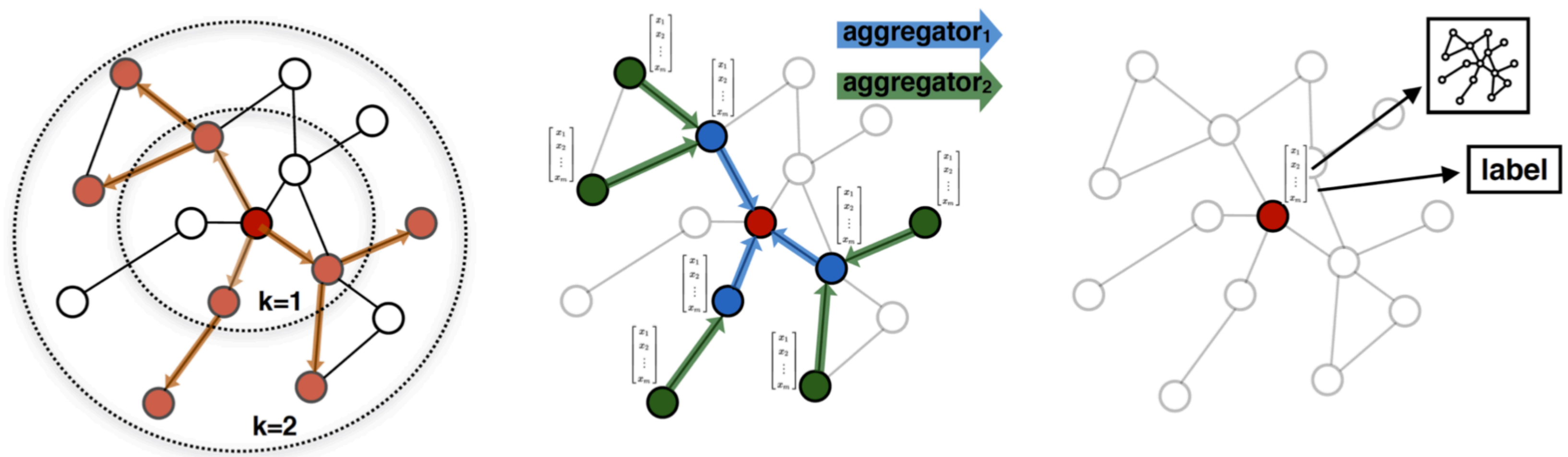


1. Sample neighborhood
2. Aggregate feature information from neighbors
3. Predict graph context and label using aggregated information

$$h_i^{(k+1)} = Relu(W^{(k)} h_i^{(k)}, \sum_{n \in \mathcal{N}_i} (Relu(Q^{(k)} h_n^{(k)})))$$

GraphSAGE

- Node's neighborhood defines a computational graph
- Each edge in this graph is a transformation/aggregator function
- Cross-entropy loss
- 2-3 layers deep

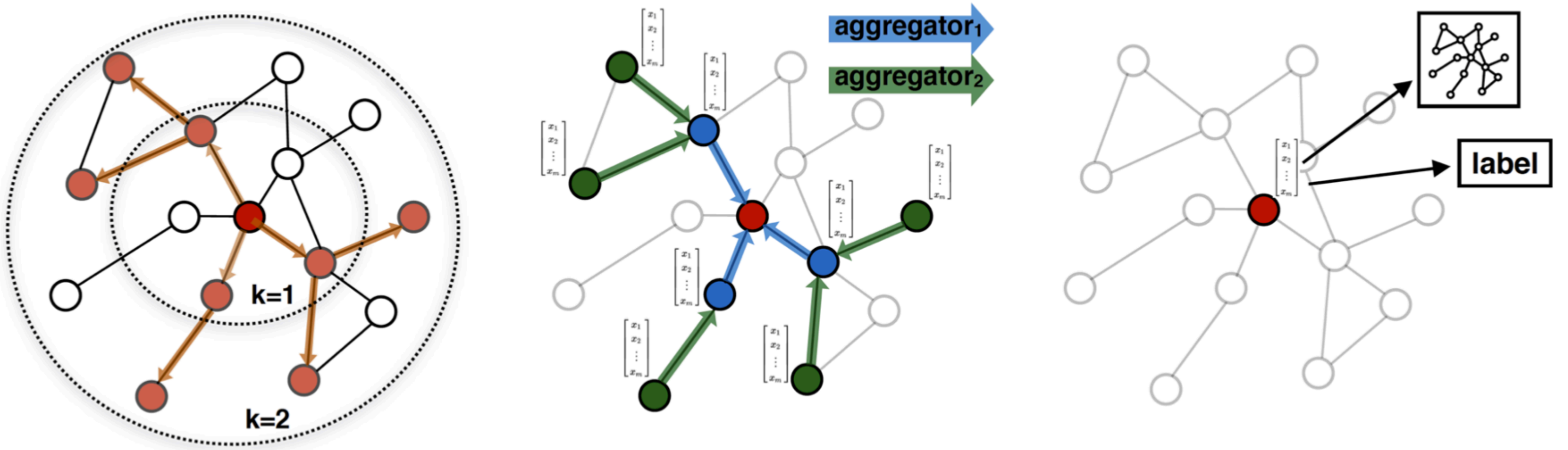


1. Sample neighborhood
2. Aggregate feature information from neighbors
3. Predict graph context and label using aggregated information

$$h_i^{(k+1)} = Relu(W^{(k)} h_i^{(k)}, \sum_{n \in \mathcal{N}_i} (Relu(Q^{(k)} h_n^{(k)})))$$

GraphSAGE

- Node's neighborhood defines a computational graph
- Each edge in this graph is a transformation/aggregator function
- Cross-entropy loss
- 2-3 layers deep



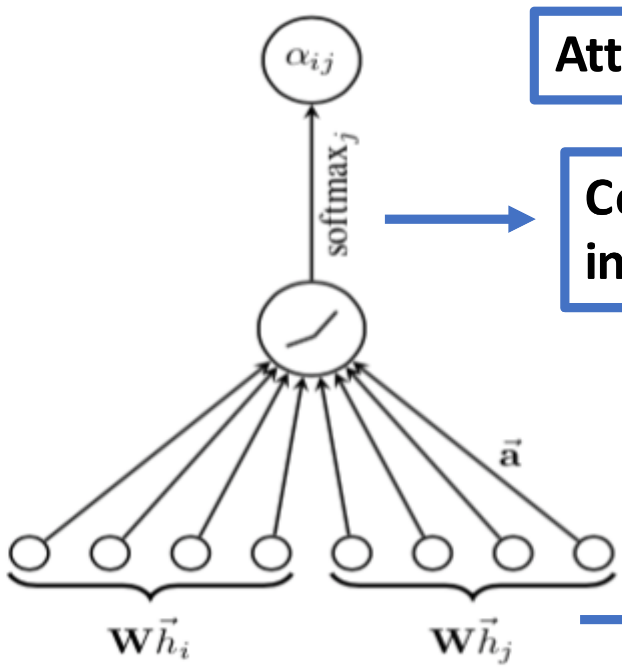
1. Sample neighborhood
2. Aggregate feature information from neighbors
3. Predict graph context and label using aggregated information

$$h_i^{(k+1)} = Relu(W^{(k)} h_i^{(k)}, \sum_{n \in \mathcal{N}_i} (Relu(Q^{(k)} h_n^{(k)})))$$

Averaging/max pooling/LSTM

Graph attention networks (GAT)

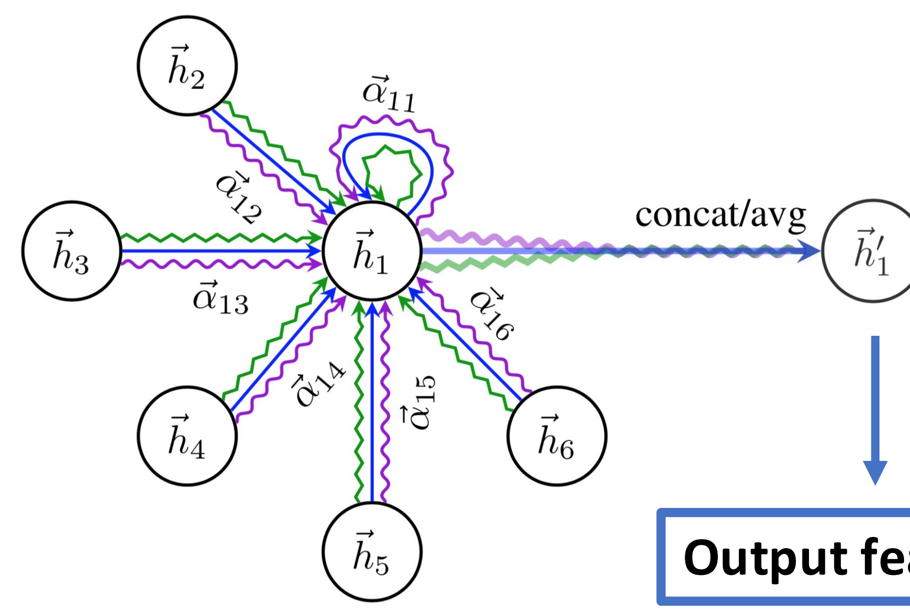
- Nodes attend over their neighborhood's features
  - Different weights to different nodes in a neighbourhood
  - Remove dependence on the global graph structure



Attention weights

Comparison of input features

Input features of nodes i, j



Output features

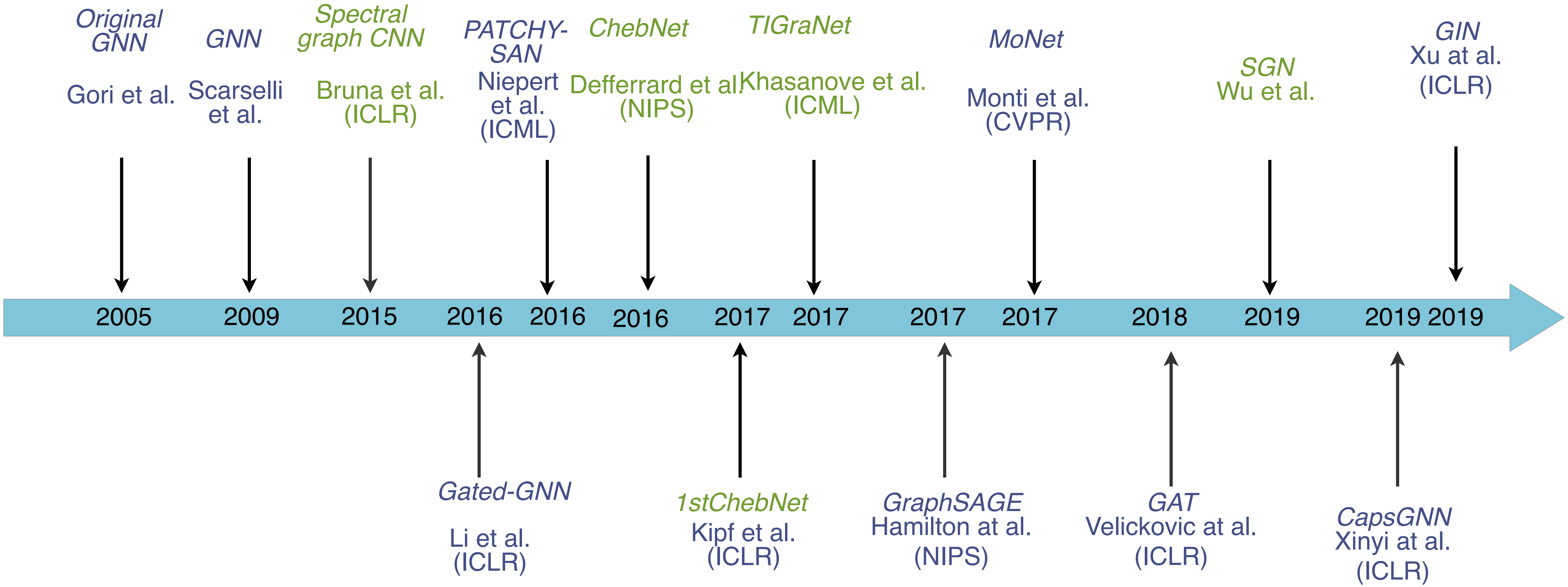
$$\vec{h}'_i = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}^k\mathbf{W}^k\vec{h}_j\right)$$

Summary of spatial methods

- **Pros:**
  - Intuitive
  - Easy to implement
  - Generalized to inductive settings

- **Cons:**
  - Lack of interpretation in the spectral domain
  - Requires many message passing iterations is the size of the graph is large

Partial historical overview



Spatial-based methods

Spectral-based methods

Spectral approaches in one slide

- Convolution is defined on the graph Fourier domain

$$x *_\theta g = \chi \hat{g}(\theta) \chi^T x$$

- **Spectral GCNN:** $\hat{g}(\theta) = \Theta$ ➡ $x *_\theta g = \chi \Theta \chi^T x$

- **ChebNet:** $\hat{g}(\theta) = \sum_{i=1}^{K} \theta_i T_i(\Lambda)$ ➡ $x *_\theta g = \sum_{i=1}^{K} \theta_i T_i(L) x$

- **GCN:** $K = 1$ ➡ $x *_\theta g = (\theta_0 I - \theta_1 D^{-1/2} A D^{-1/2}) x$

- Parameters Θ are learned through the network

Spectral approaches in one slide

- Convolution is defined on the graph Fourier domain

$$x *_\theta g = \chi \hat{g}(\theta) \chi^T x$$

- **Spectral GCNN:** $\hat{g}(\theta) = \Theta$ → $x *_\theta g = \chi \Theta \chi^T x$
- **ChebNet:** $\hat{g}(\theta) = \sum_{i=1}^{K} \theta_i T_i(\Lambda)$ → $x *_\theta g = \sum_{i=1}^{K} \theta_i T_i(L) x$
- **GCN:** $K = 1$ → $x *_\theta g = (\theta_0 I - \theta_1 D^{-1/2} A D^{-1/2}) x$

- Parameters Θ are learned through the network

Spectral approaches in one slide

- Convolution is defined on the graph Fourier domain

$$x *_\theta g = \chi \hat{g}(\theta) \chi^T x$$

- **Spectral GCNN:** $\hat{g}(\theta) = \Theta$ ⇒ $x *_\theta g = \chi \Theta \chi^T x$
- **ChebNet:** $\hat{g}(\theta) = \sum_{i=1}^{K} \theta_i T_i(\Lambda)$ ⇒ $x *_\theta g = \sum_{i=1}^{K} \theta_i T_i(L) x$
- **GCN:** $K = 1$ ⇒ $x *_\theta g = (\theta_0 I - \theta_1 D^{-1/2} A D^{-1/2}) x$

- Parameters Θ are learned through the network

Spectral approaches in one slide

- Convolution is defined on the graph Fourier domain

$$x *_\theta g = \chi \hat{g}(\theta) \chi^T x$$

- **Spectral GCNN:** $\hat{g}(\theta) = \Theta \Rightarrow x *_\theta g = \chi \Theta \chi^T x$
- **ChebNet:** $\hat{g}(\theta) = \sum_{i=1}^{K} \theta_i T_i(\Lambda) \Rightarrow x *_\theta g = \sum_{i=1}^{K} \theta_i T_i(L) x$
- **GCN:** $K = 1 \Rightarrow x *_\theta g = (\theta_0 I - \theta_1 D^{-1/2} A D^{-1/2}) x$
- Parameters Θ are learned through the network

Spectral approaches in one slide

- Convolution is defined on the graph Fourier domain

$$x *_\theta g = \chi \hat{g}(\theta) \chi^T x$$

- **Spectral GCNN:** $\hat{g}(\theta) = \Theta$ ➡ $x *_\theta g = \chi \Theta \chi^T x$

- **ChebNet:** $\hat{g}(\theta) = \sum_{i=1}^{K} \theta_i T_i(\Lambda)$ ➡ $x *_\theta g = \sum_{i=1}^{K} \theta_i T_i(L) x$

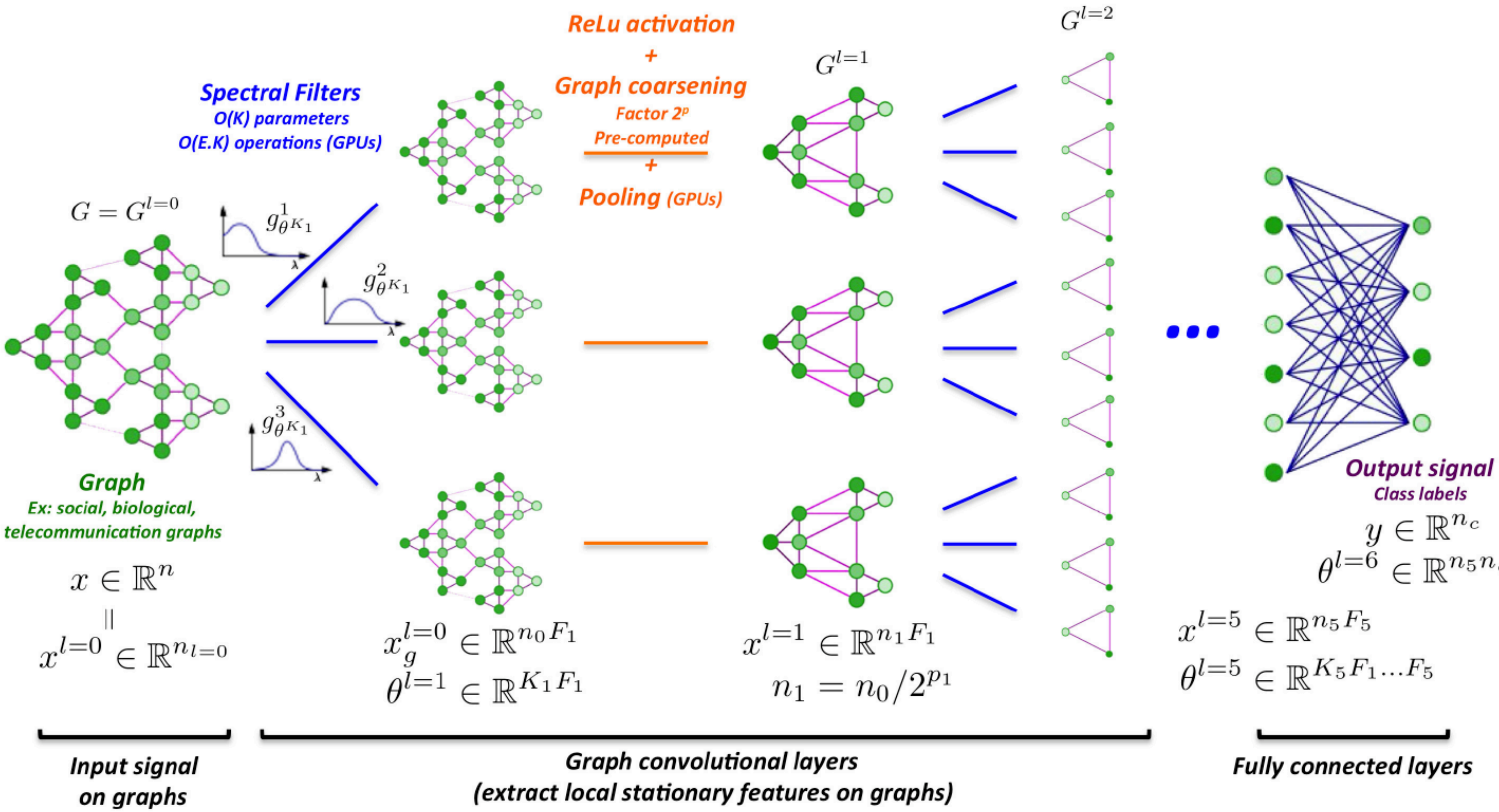
- **GCN:** $K = 1$ ➡ $x *_\theta g = (\theta_0 I - \theta_1 D^{-1/2} A D^{-1/2}) x$

Neighborhood aggregation

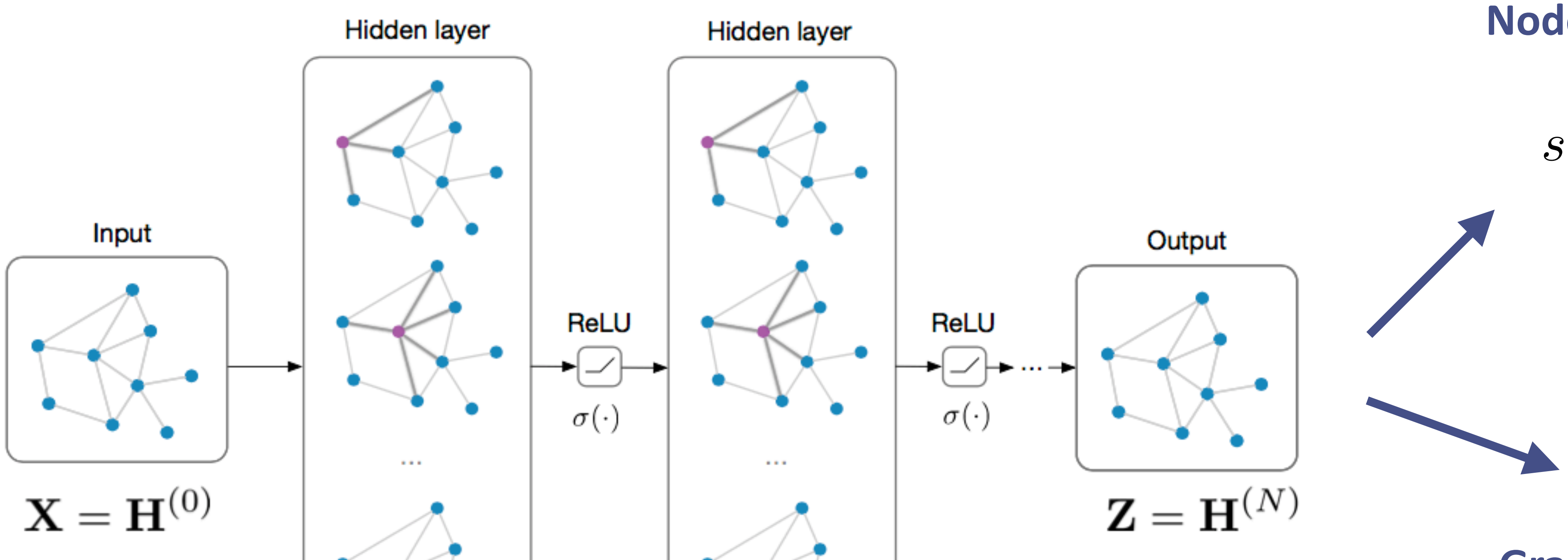
- Parameters Θ are learned through the network

ChebNet

- Graph filters: Chebyshev polynomials



GCN



$$\mathbf{H}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

Node classification

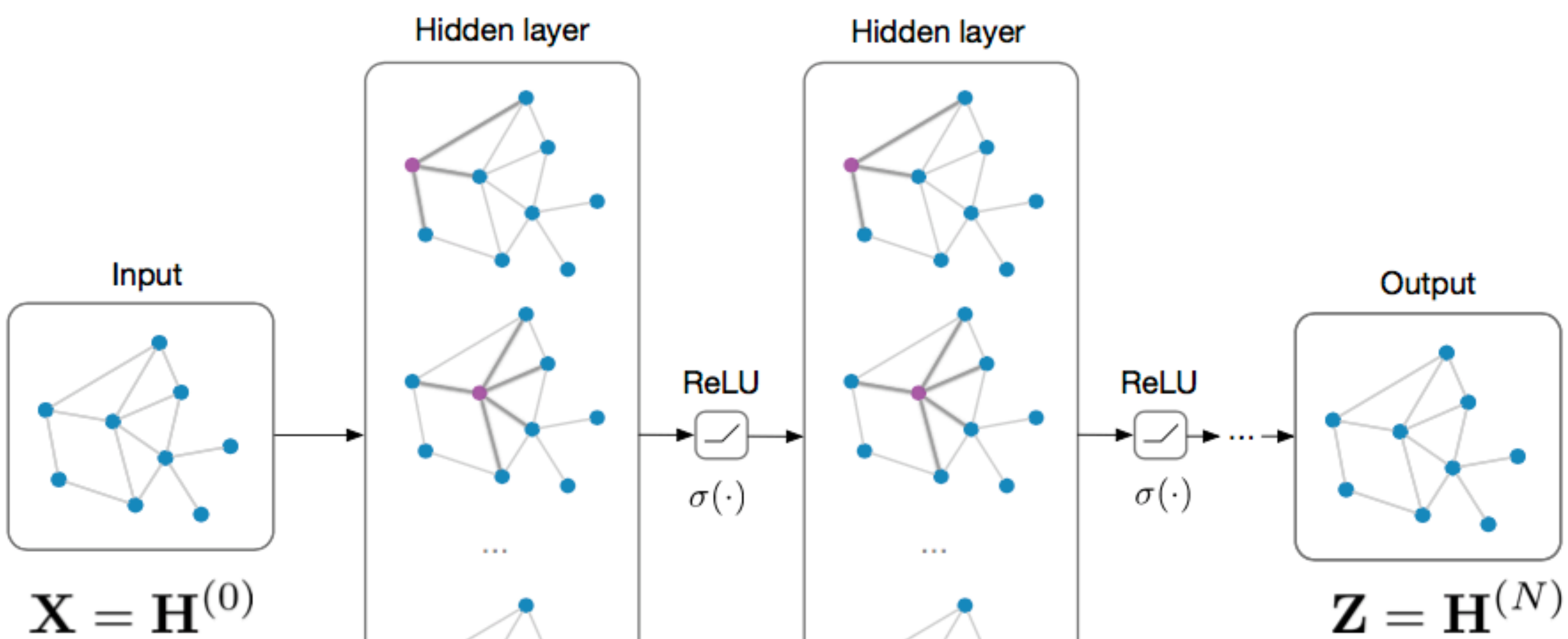
$$softmax(h_n^{(l)})$$

Graph classification

$$softmax(\sum_{n=1}^{N} h_n^{(l)})$$

Fig. from T. Kirf

GCN



$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

Node classification

$$softmax(h_n^{(l)})$$

Graph classification

$$softmax(\sum_{n=1}^{N} h_n^{(l)})$$

Fig. from T. Kirf

Simple graph convolution (SGC)

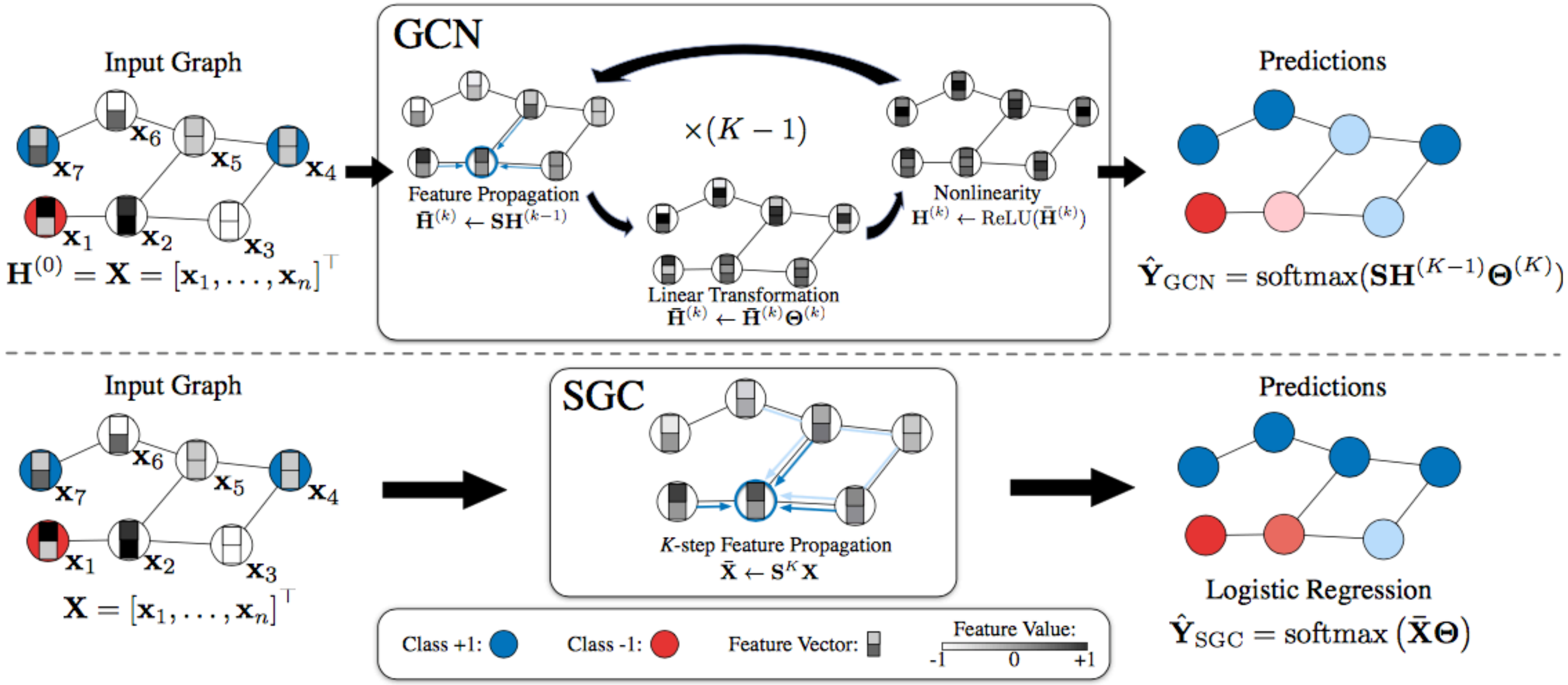


Fig. from Wu et al.

- Reduces the procedure to a simple feature propagation step followed by standard logistic regression

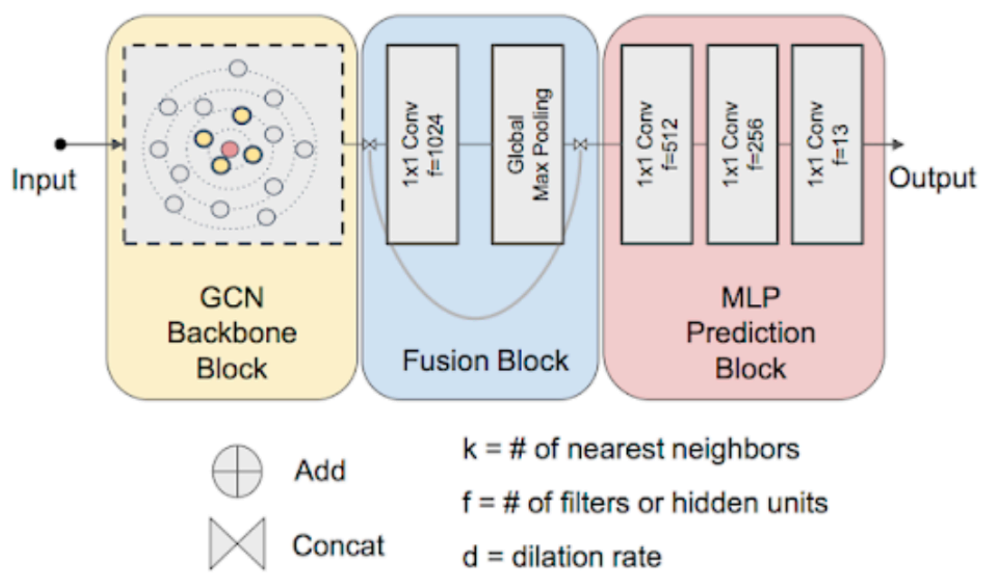
Summary of spectral methods

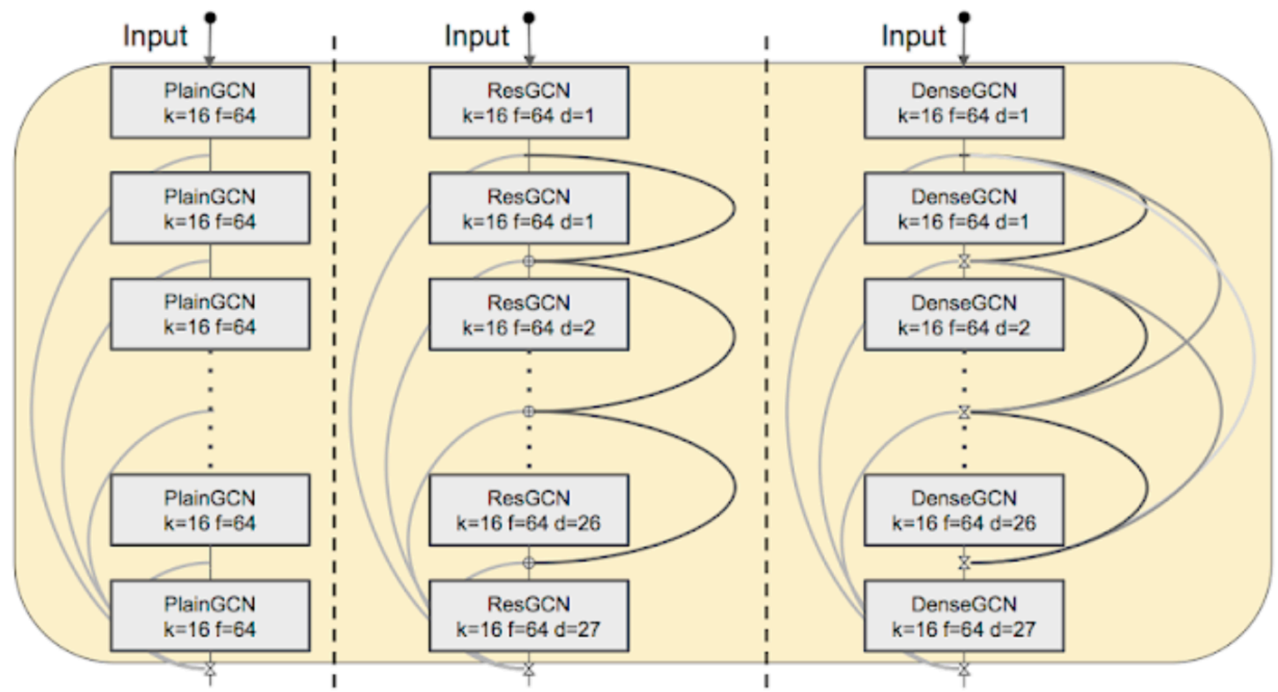
- **Pros:**
  - More interpretable
  - Takes into account the global structure of the graph

- **Cons:**
  - Generalization is an open research question

DeepGCNs

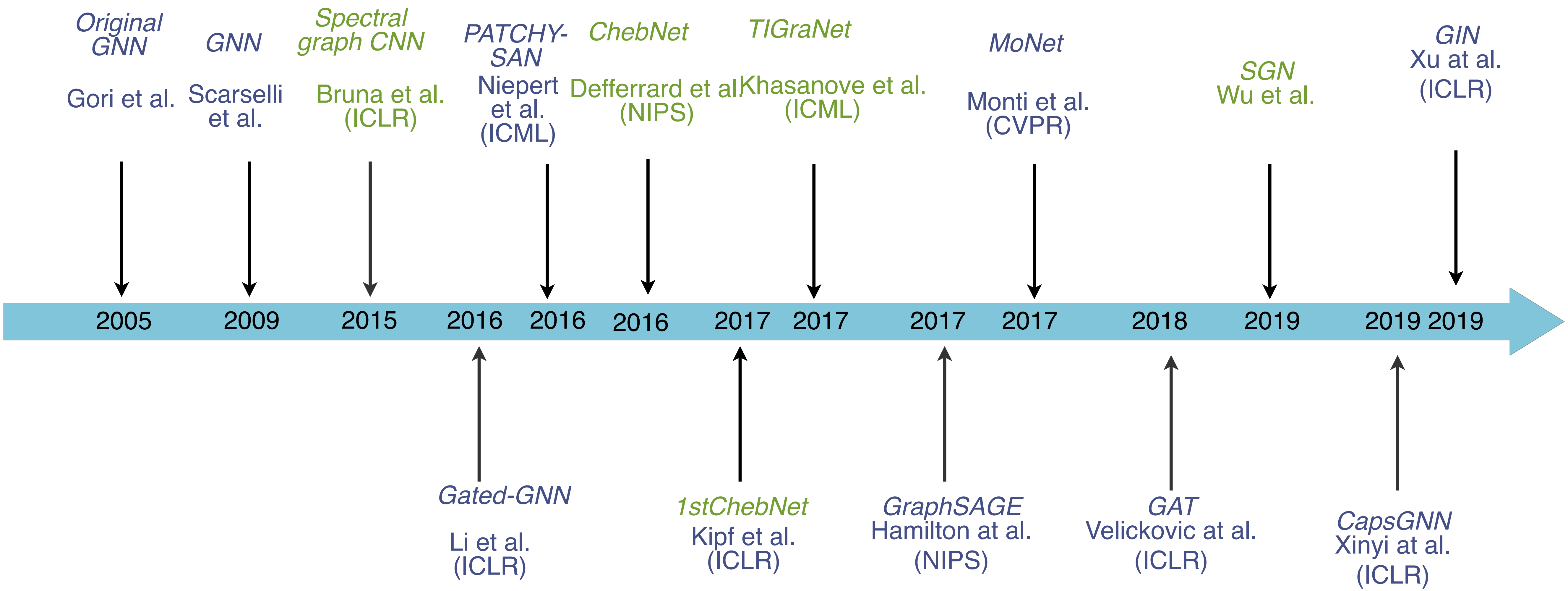
- Design deeper networks by using intuitions from ResNet, DenseNet





Li et. al, 2019

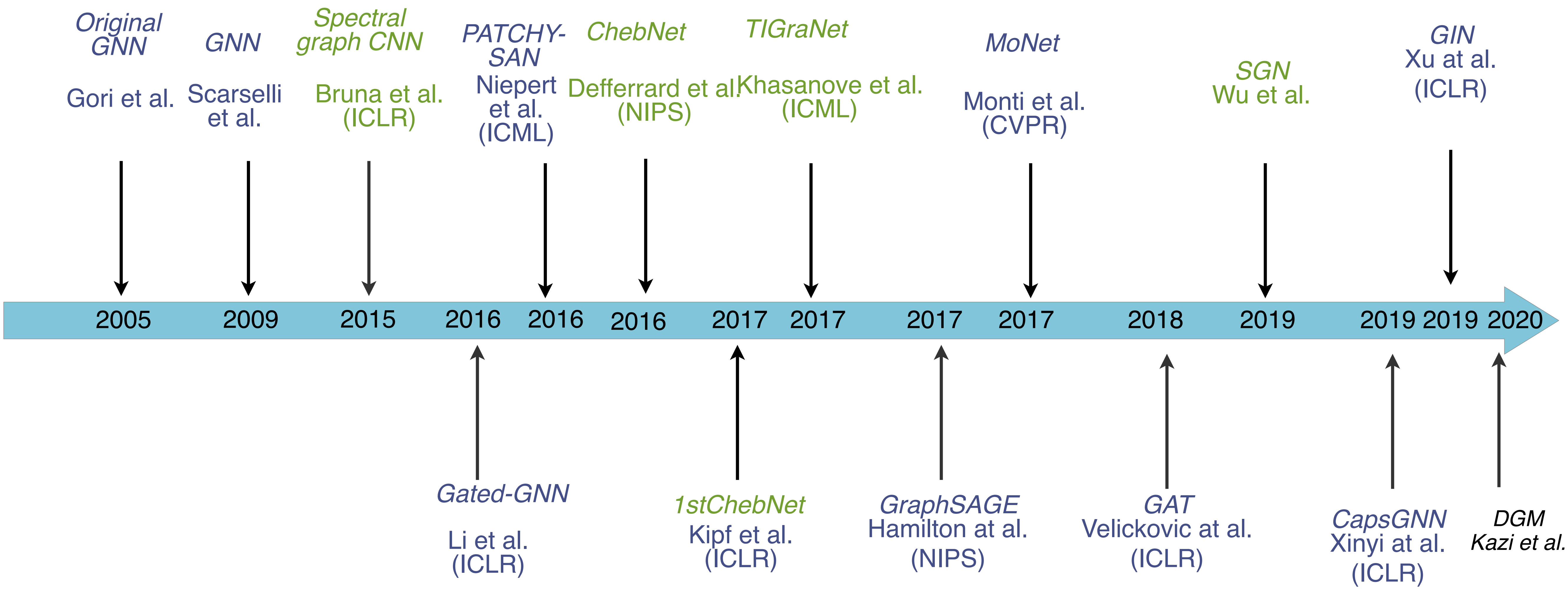
Partial historical overview



Spatial-based methods

Spectral-based methods

Partial historical overview

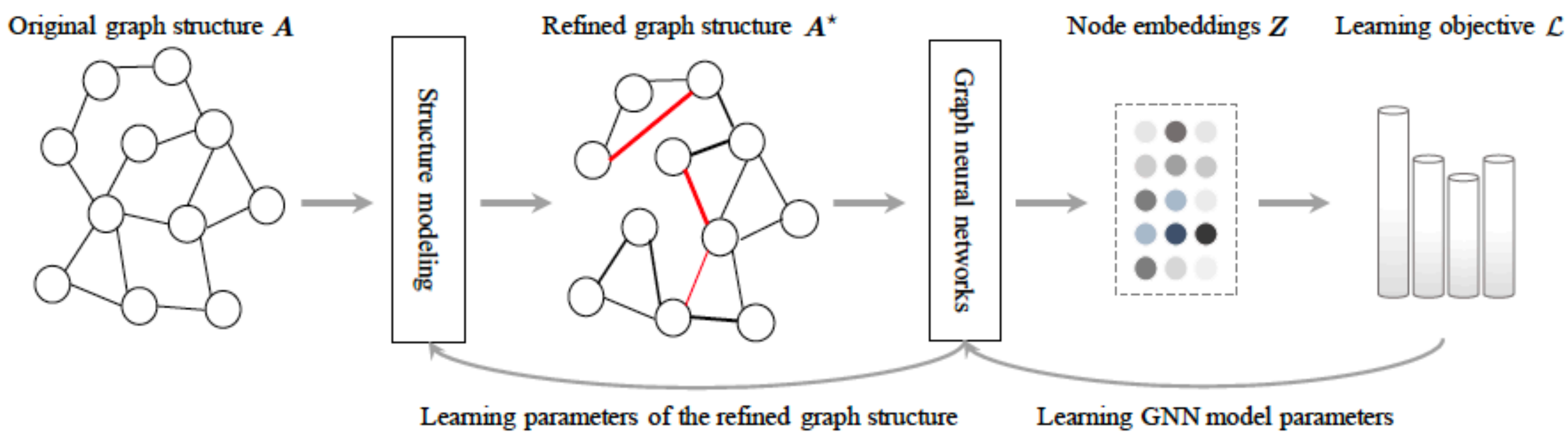


Spatial-based methods

Spectral-based methods

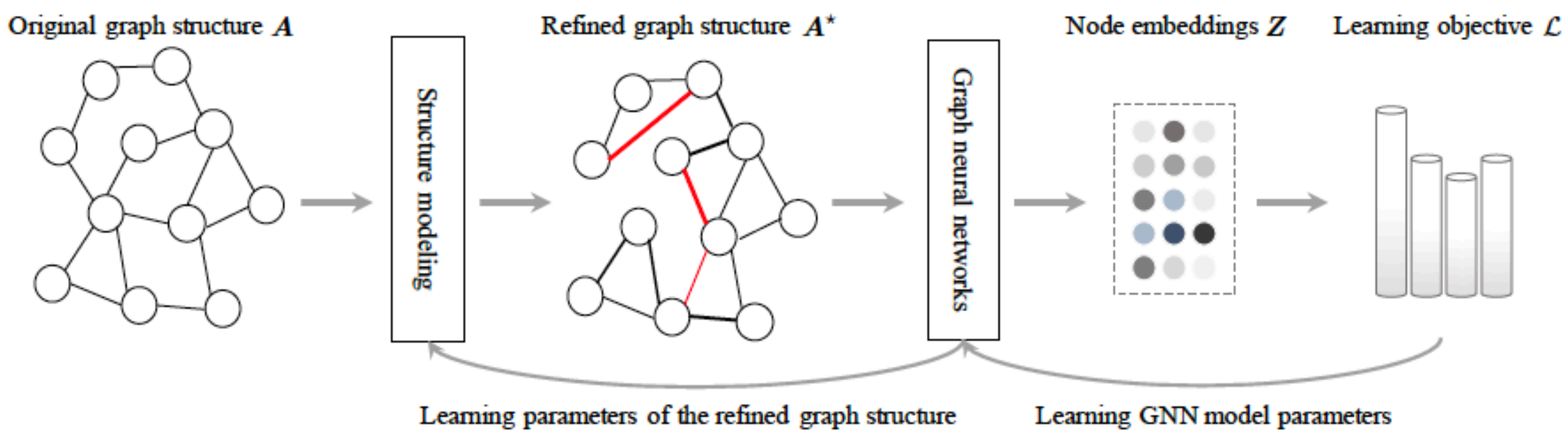
Joint graph learning

- Infers simultaneously the graph structure



Joint graph learning

- Infers simultaneously the graph structure



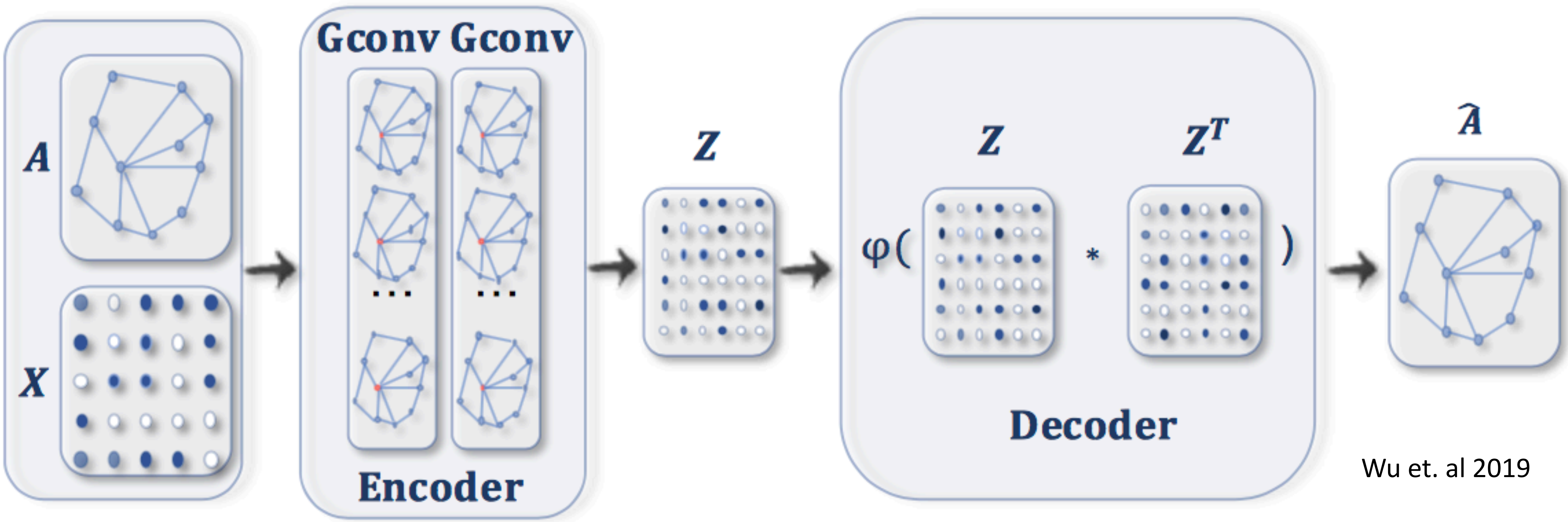
Zhu et al., Deep Graph Structure Learning for Robust Representations: A Survey, arXiv, 2021

Outline

- Motivation: Why Graph Neural Networks?
- Basic definitions on graphs
- Partial historical overview
  - Graph CNN (main focus)
  - **Graph autoencoders (briefly)**
- Applications
  - Naturally graph-structured data
  - Images

Graph autoencoders (GAEs)

- Unsupervised learning: encode graphs into a latent space and reconstruct the graph from that space
-

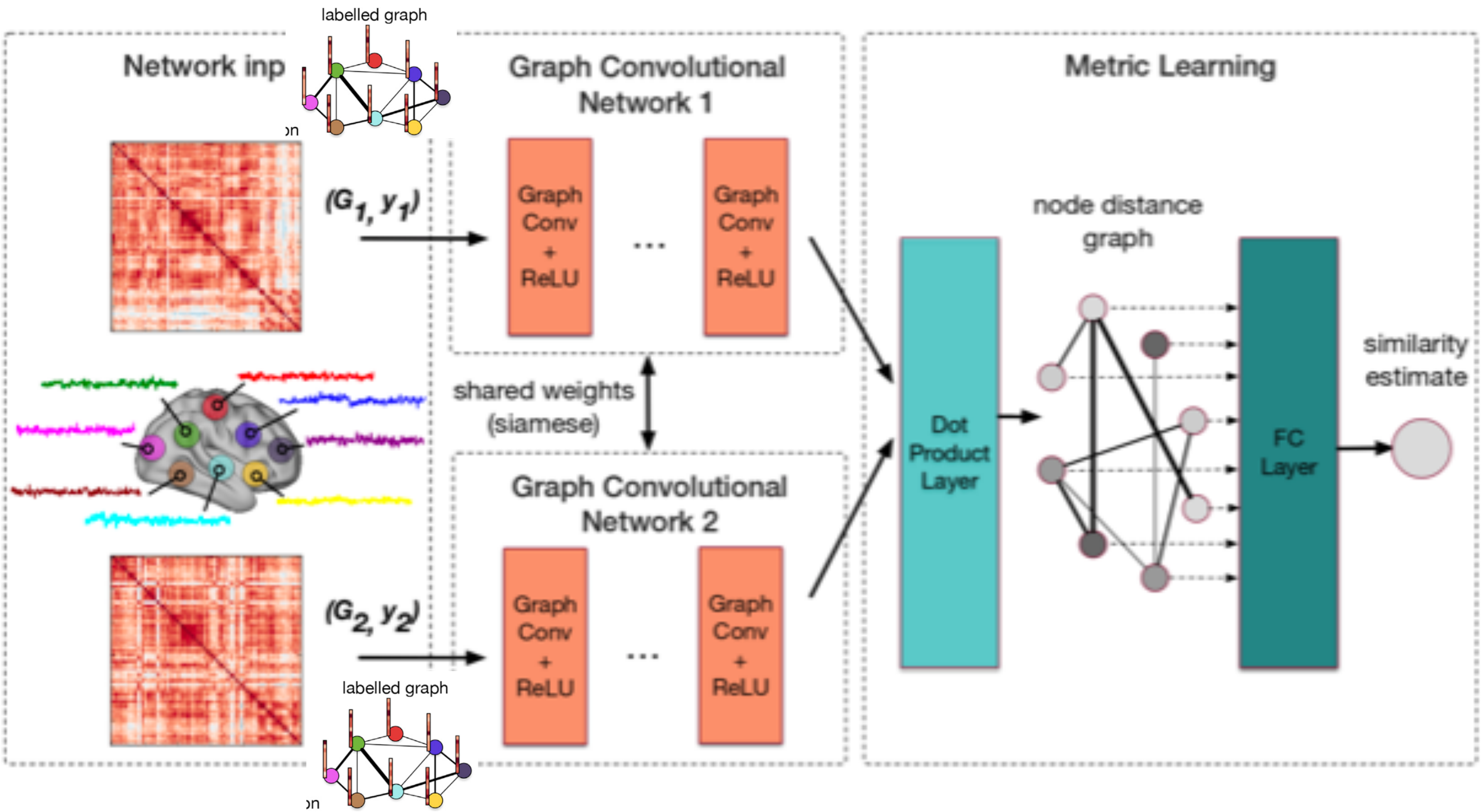


Wu et. al 2019

Outline

- Motivation: Why Graph Neural Networks?
- Basic definitions on graphs
- Partial historical overview
  - Graph CNN (main focus)
  - Graph autoencoders (briefly)
- **Applications**
  - Naturally graph-structured data
  - Images

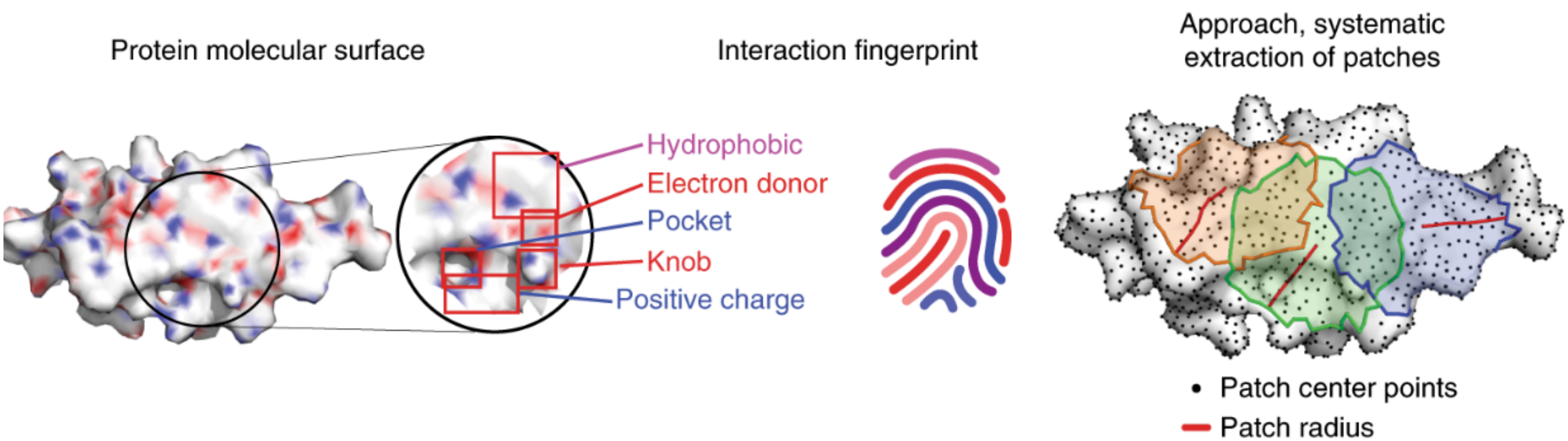
Neuroscience: learn to compare



Ktena et al., NeuroImage, 2018

Protein-protein interactions

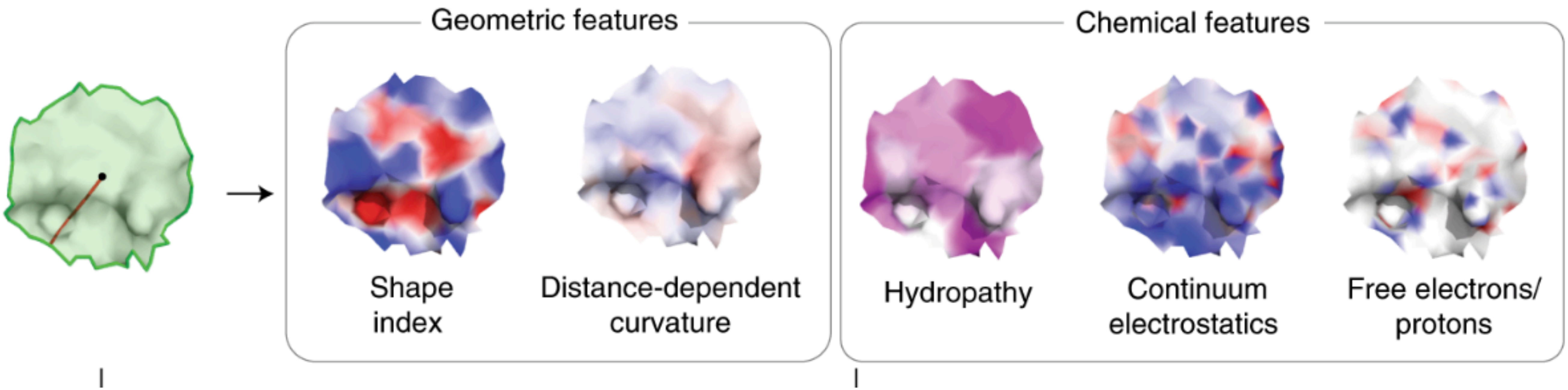
- Exploit GNNs to learn interaction fingerprints in protein molecular surfaces



Gainza et al, Nature methods, 2019

Protein-protein interactions

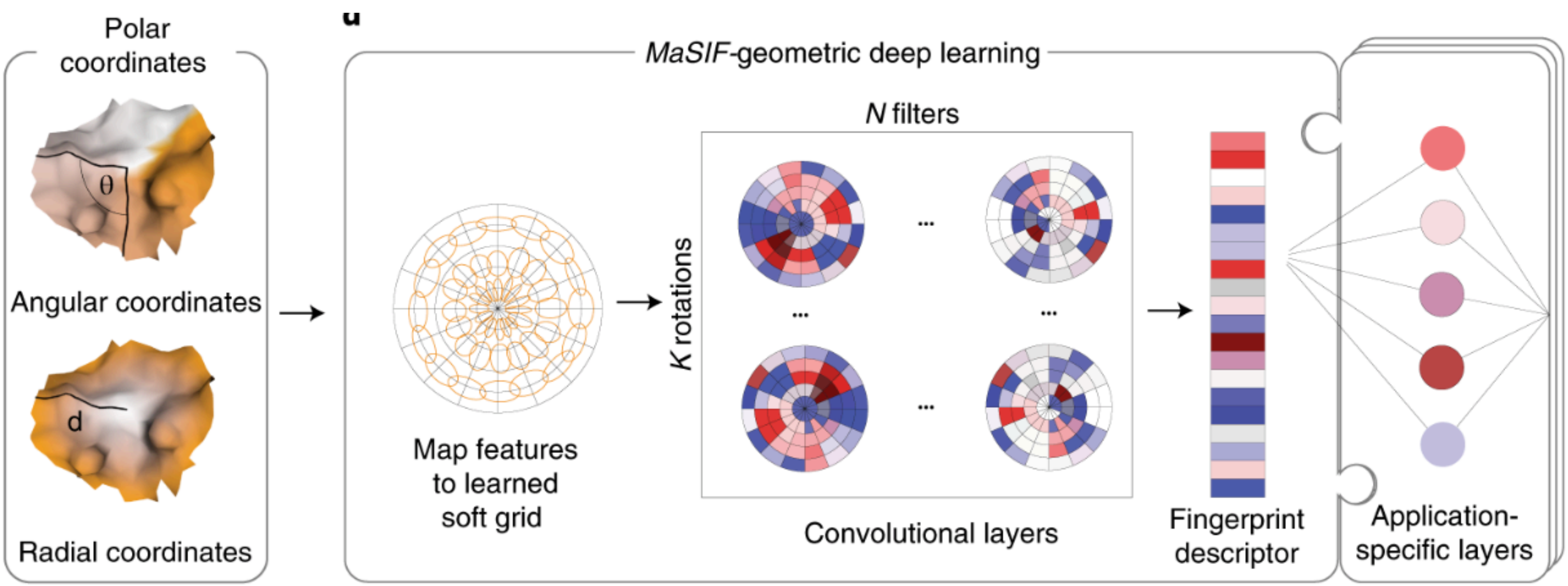
- Exploit GNNs to learn interaction fingerprints in protein molecular surfaces



Gainza et al, Nature methods, 2019

Protein-protein interactions

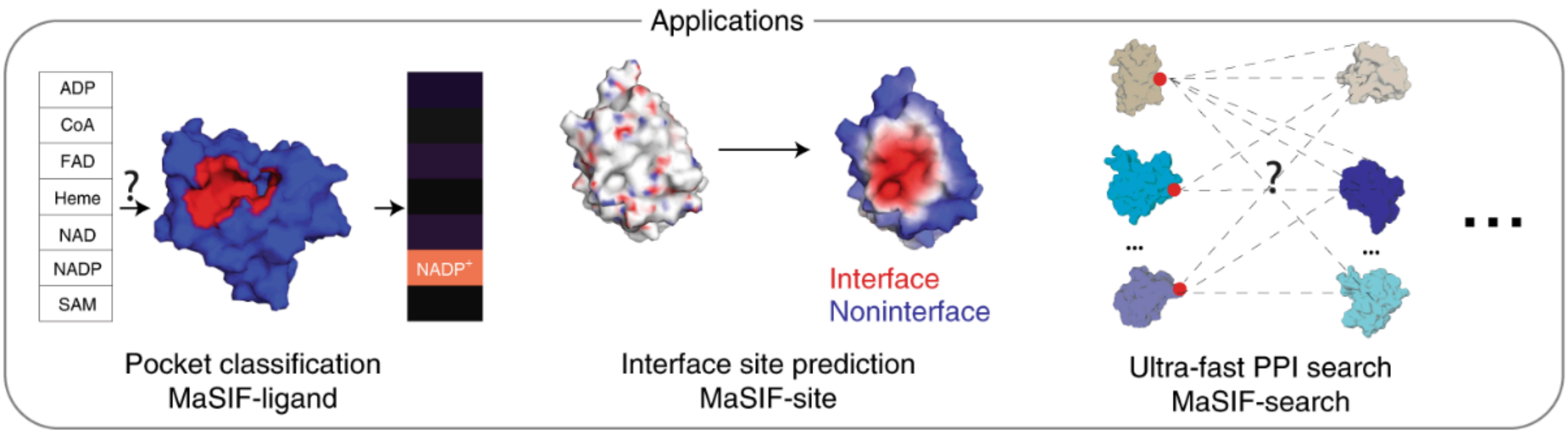
- Exploit GNNs to learn interaction fingerprints in protein molecular surfaces



Gainza et al, Nature methods, 2019

Protein-protein interactions

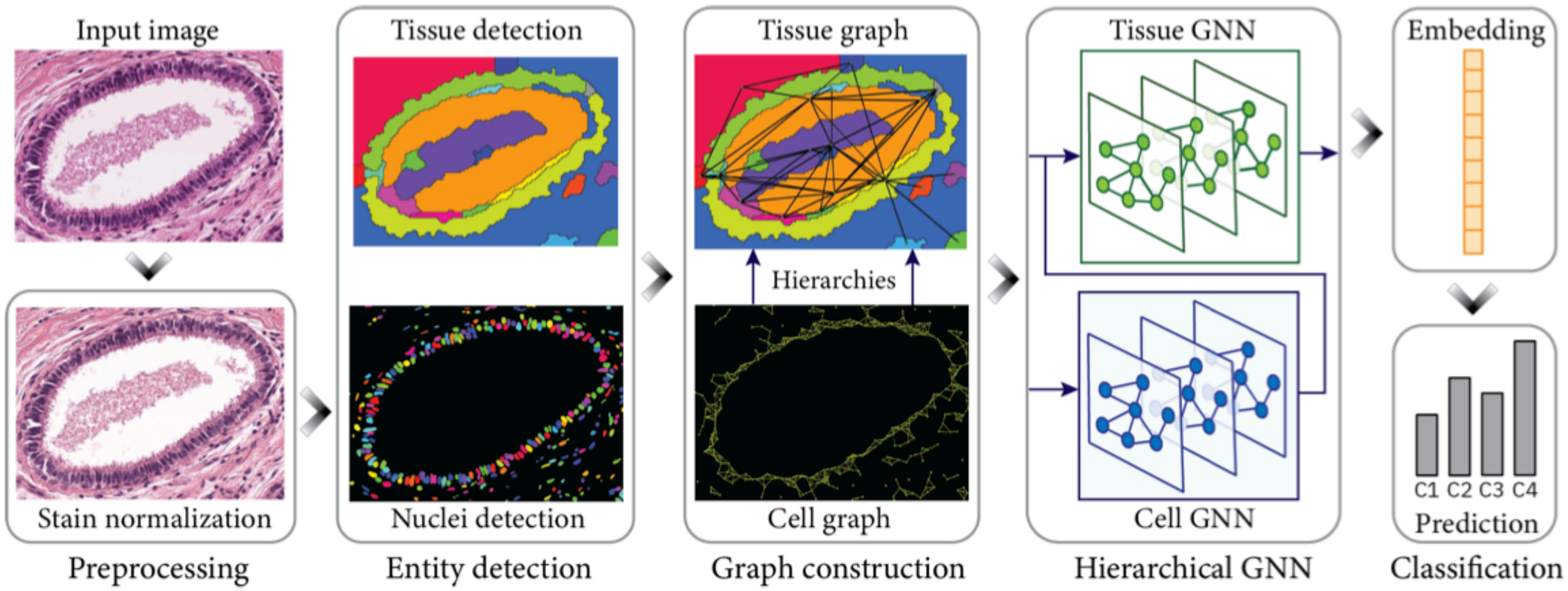
- Exploit GNNs to learn interaction fingerprints in protein molecular surfaces



Gainza et al, Nature methods, 2019

GNNs for medical imaging

- Digital pathology [1]



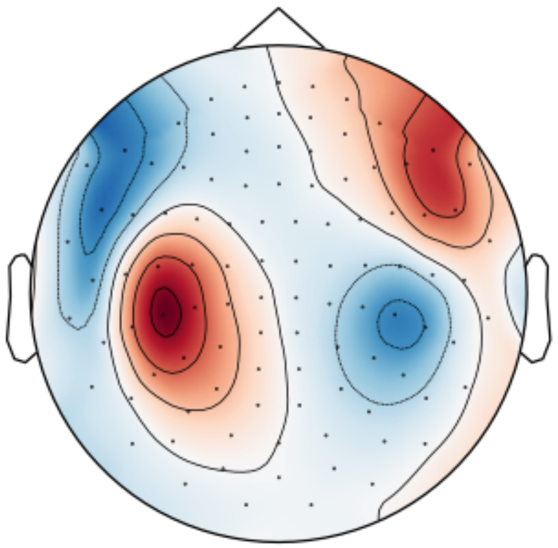
- Graph based representations provide a flexible tool for modelling complex dependencies at different levels of hierarchy (e.g., cells, tissues)

[1] Pati et al, "Hierarchical graph representation in digital pathology," arXiv, 2021

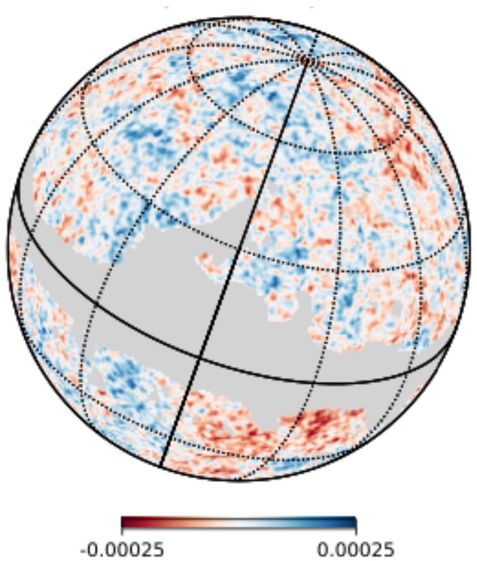
[2] Li et al, Representation learning for networks in biology and medicine: Advancements, challenges, and opportunities, arXiv, 2021

GNNs for spherical imaging

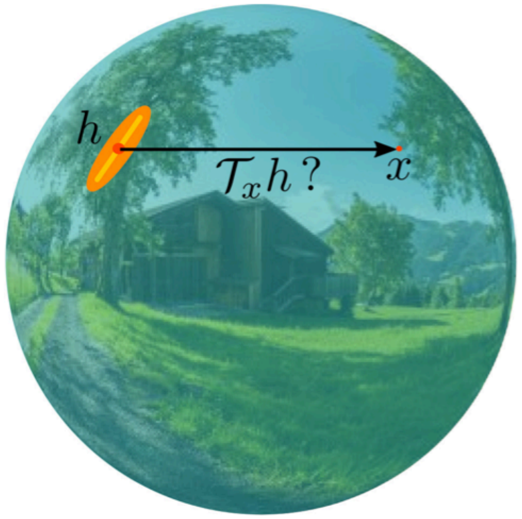
- Spherical data has specific spatial and statistical properties that cannot be captured by regular CNN models



Brain activity (MEG)



Cosmic microwave background temperature

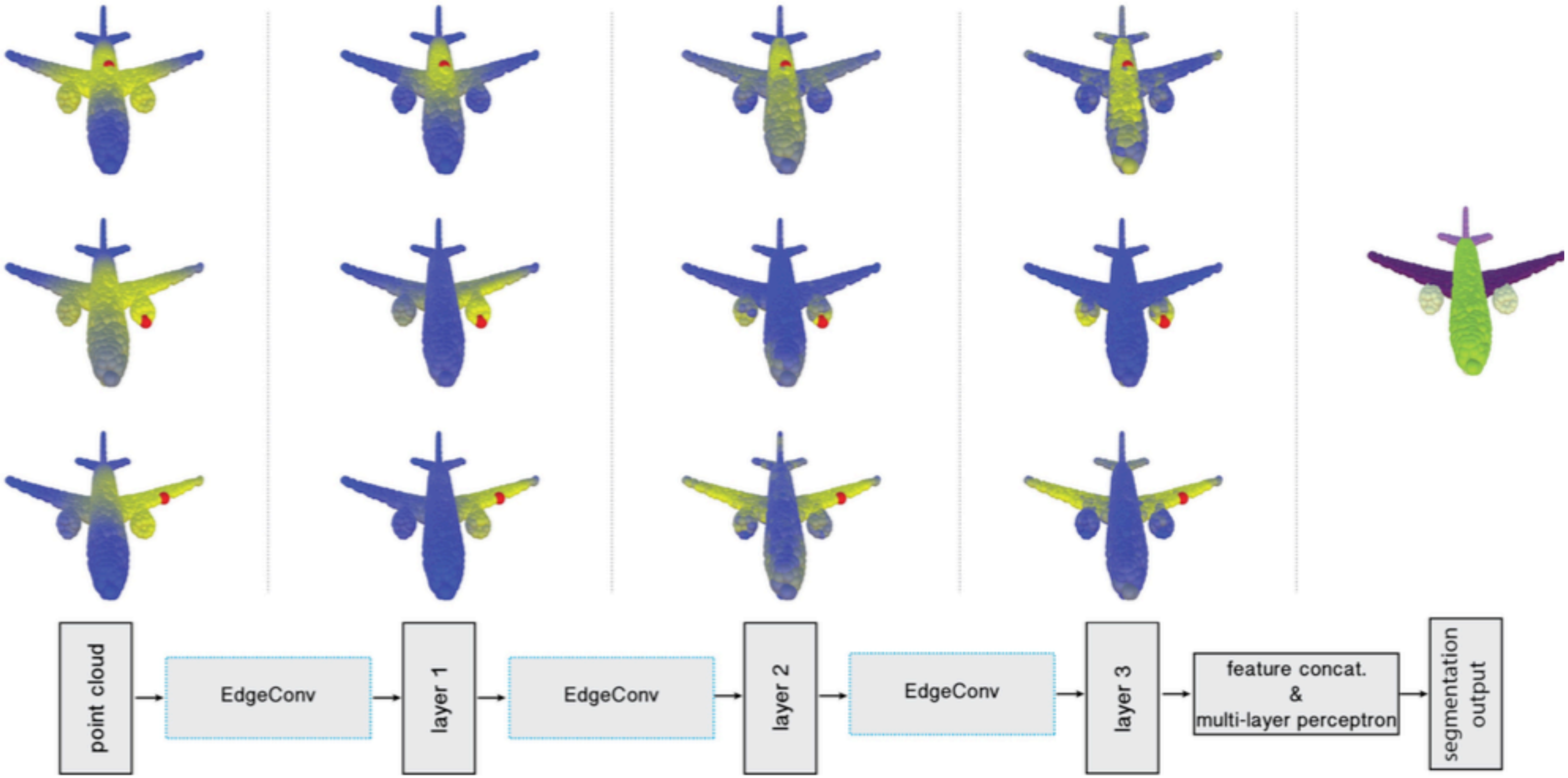


Omnidirectional images

- Sphere is modelled as a graph and classical operation (convolution, translation, pooling…) are performed on the graph

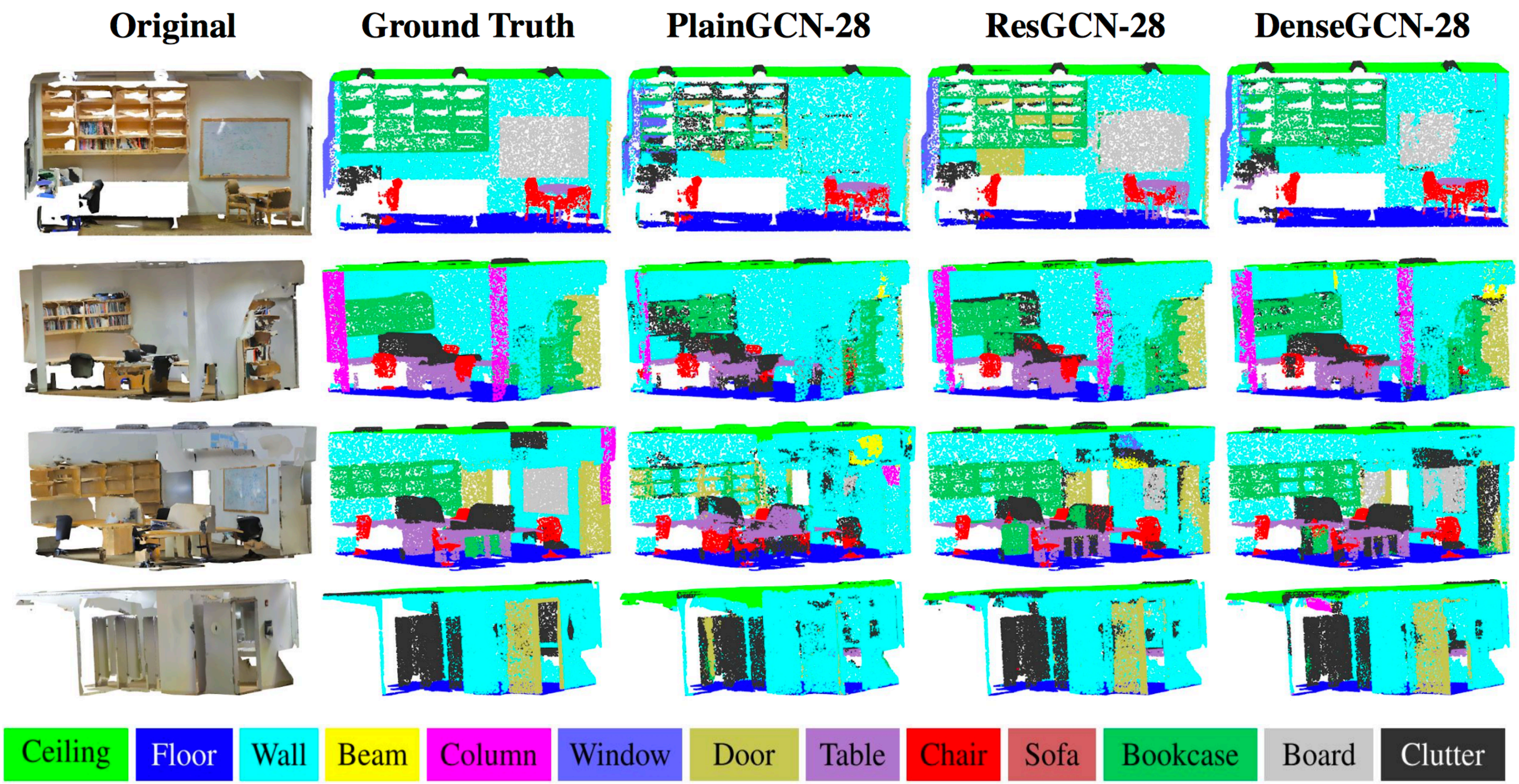
Perraudin et al., "DeepSphere", Astronomy and Computing, 2019
Bidgoli et al, OSLO: On-the-Sphere Learning for Omnidirectional images and its application to 360-degree image compression, arXiv, 2021

Point cloud semantic



Wang et al., 2019

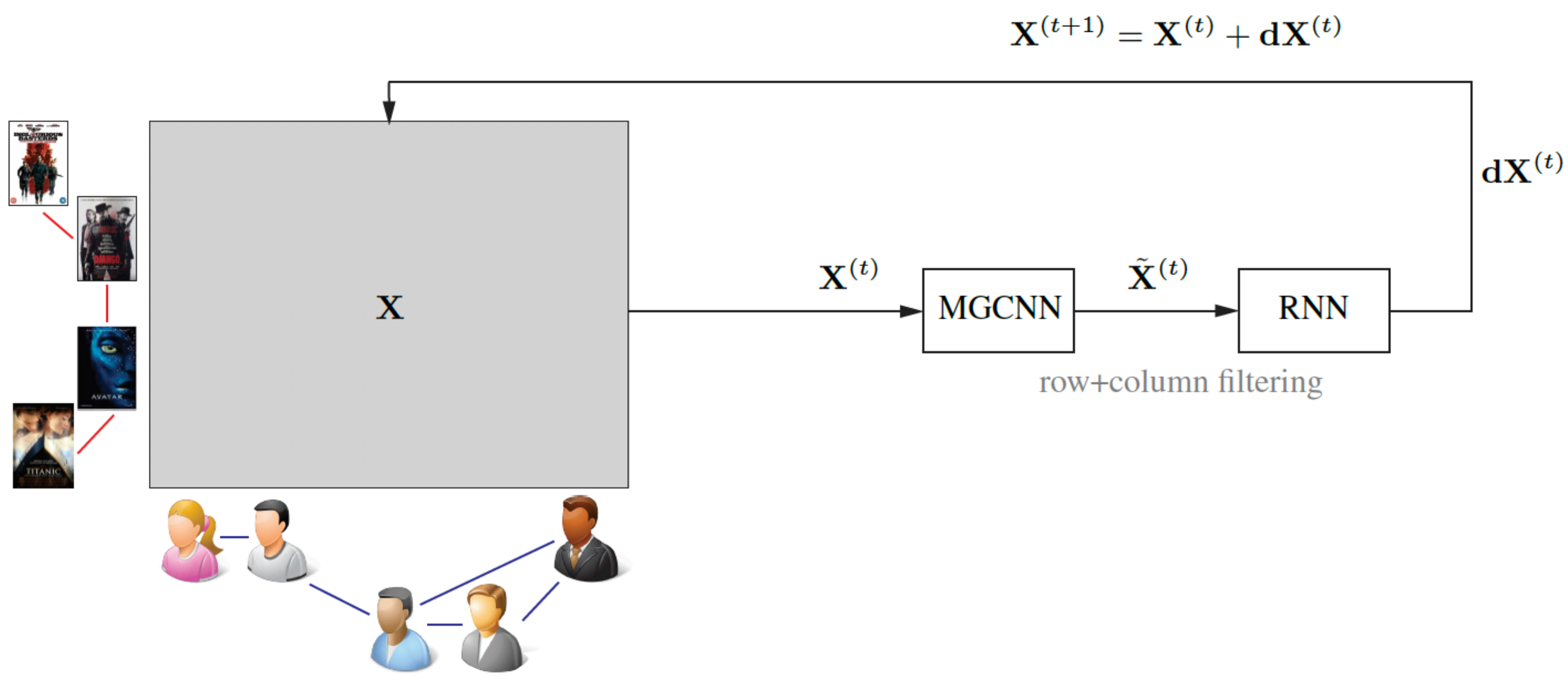
3D point clouds semantic



Li et al. 2019

Recommender systems

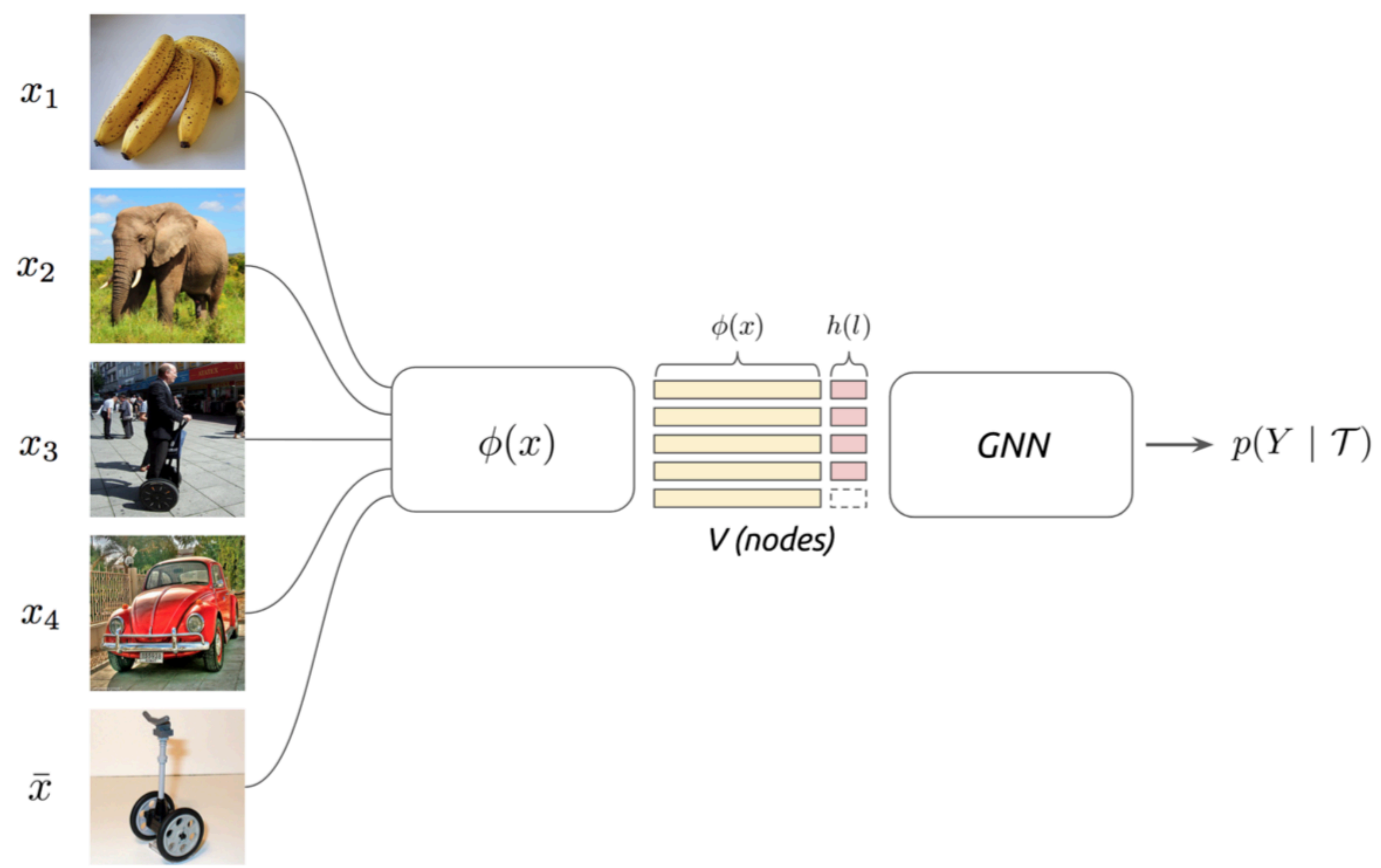
- Matrix completion: deep learning on user and item graphs
- Multi-graph convolution (spatial features), followed by LSTM (diffusion process)



Monti et al. NIPS, 2017

Few-shot learning

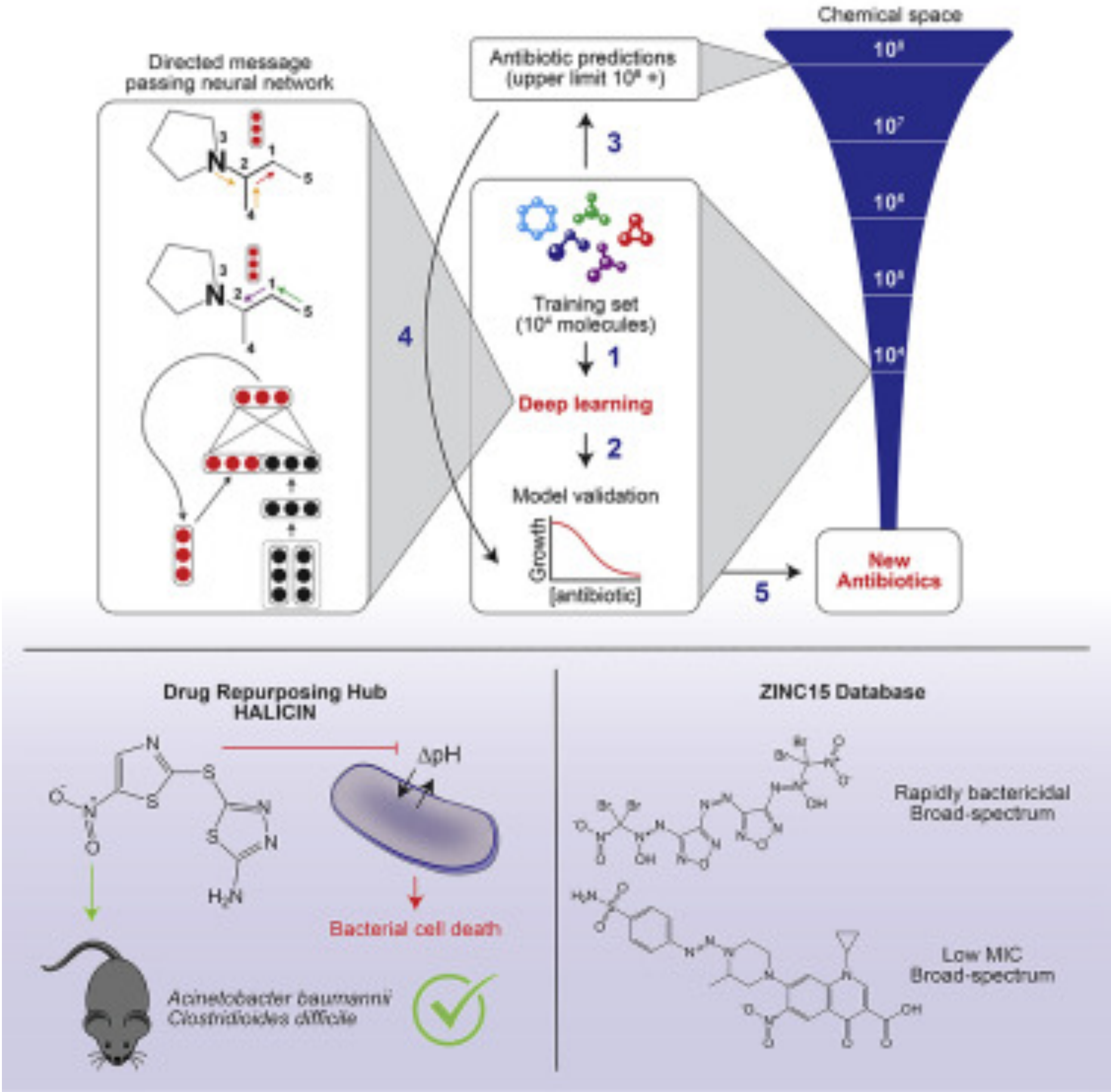
- Represent images as a fully connected graph
- Pose the problem as a supervised task using GNN

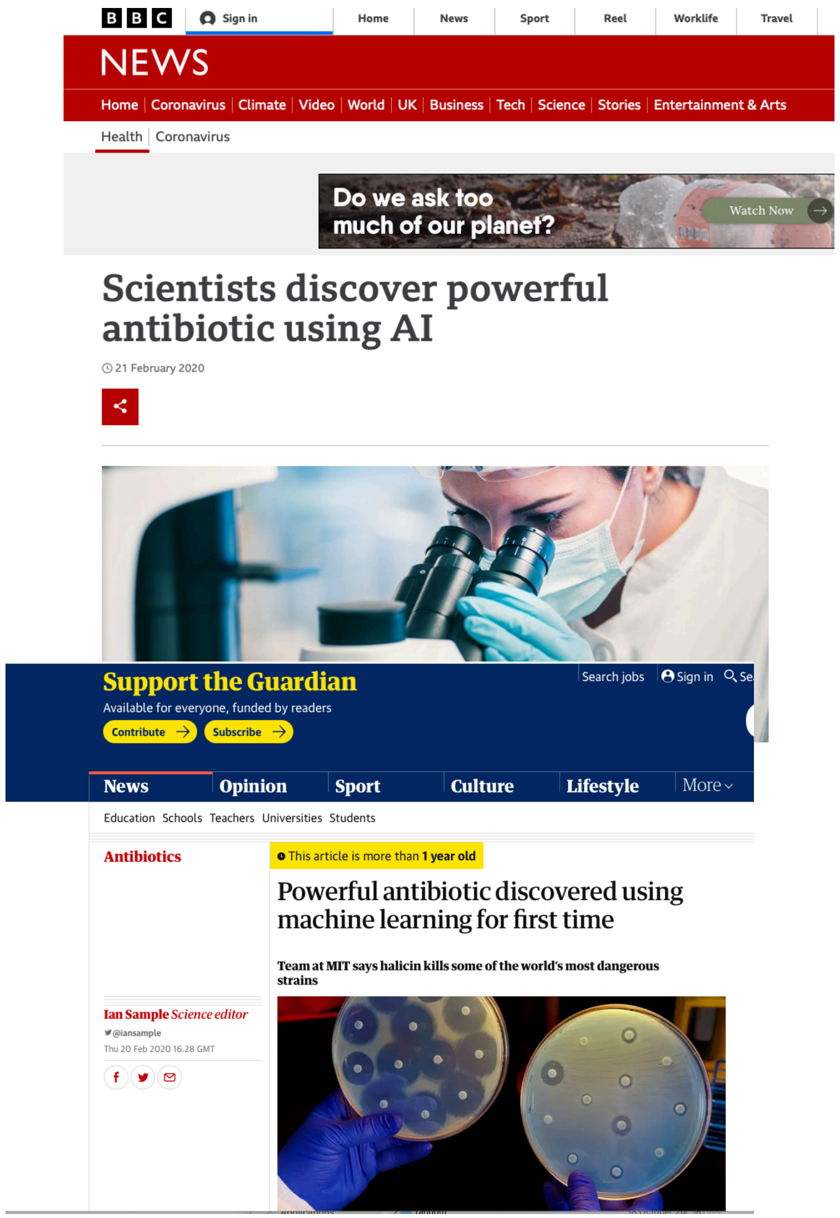


Garcia at et., ICLR, 2018

Molecular graph generation

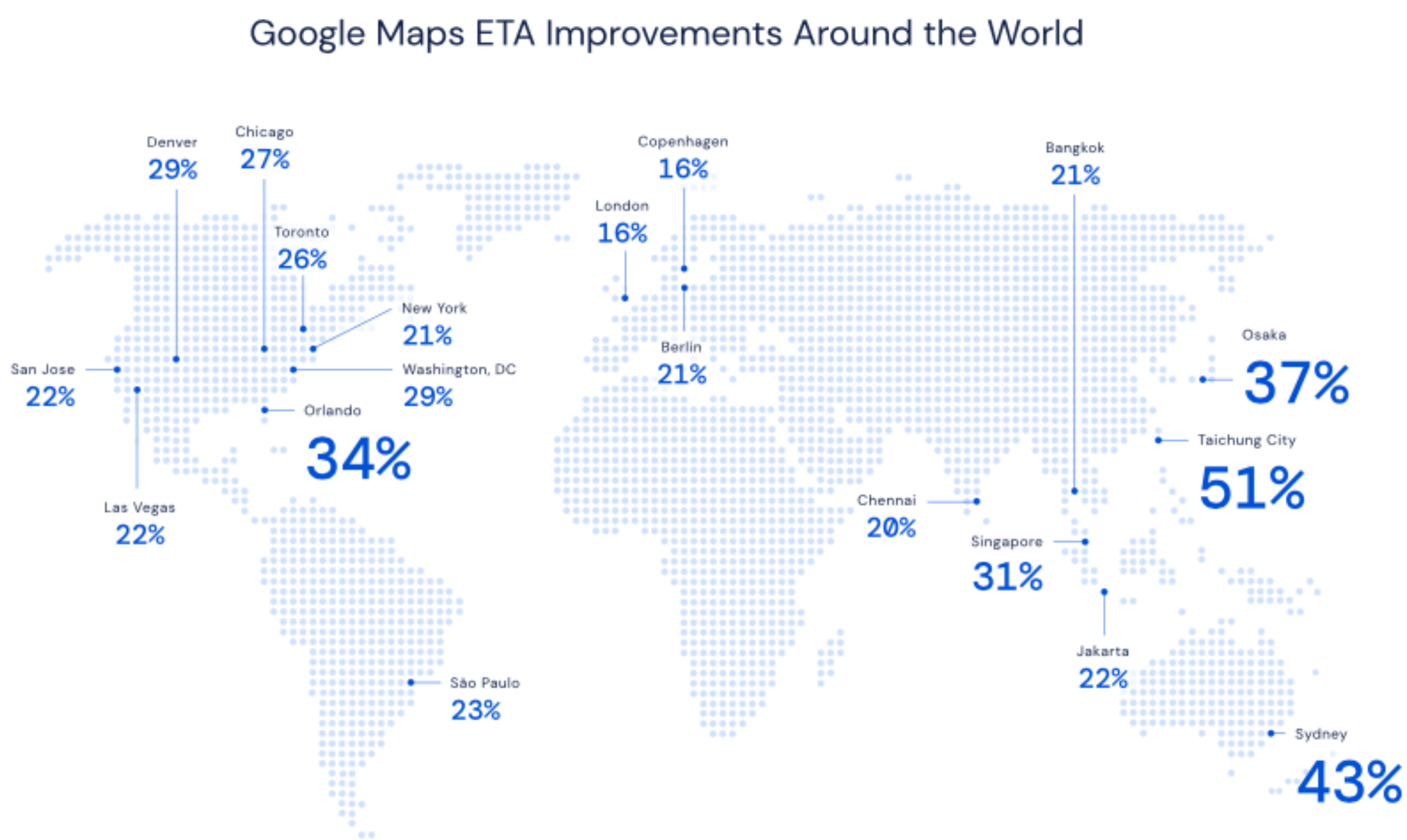
- Recent advances in antibiotic discovery …





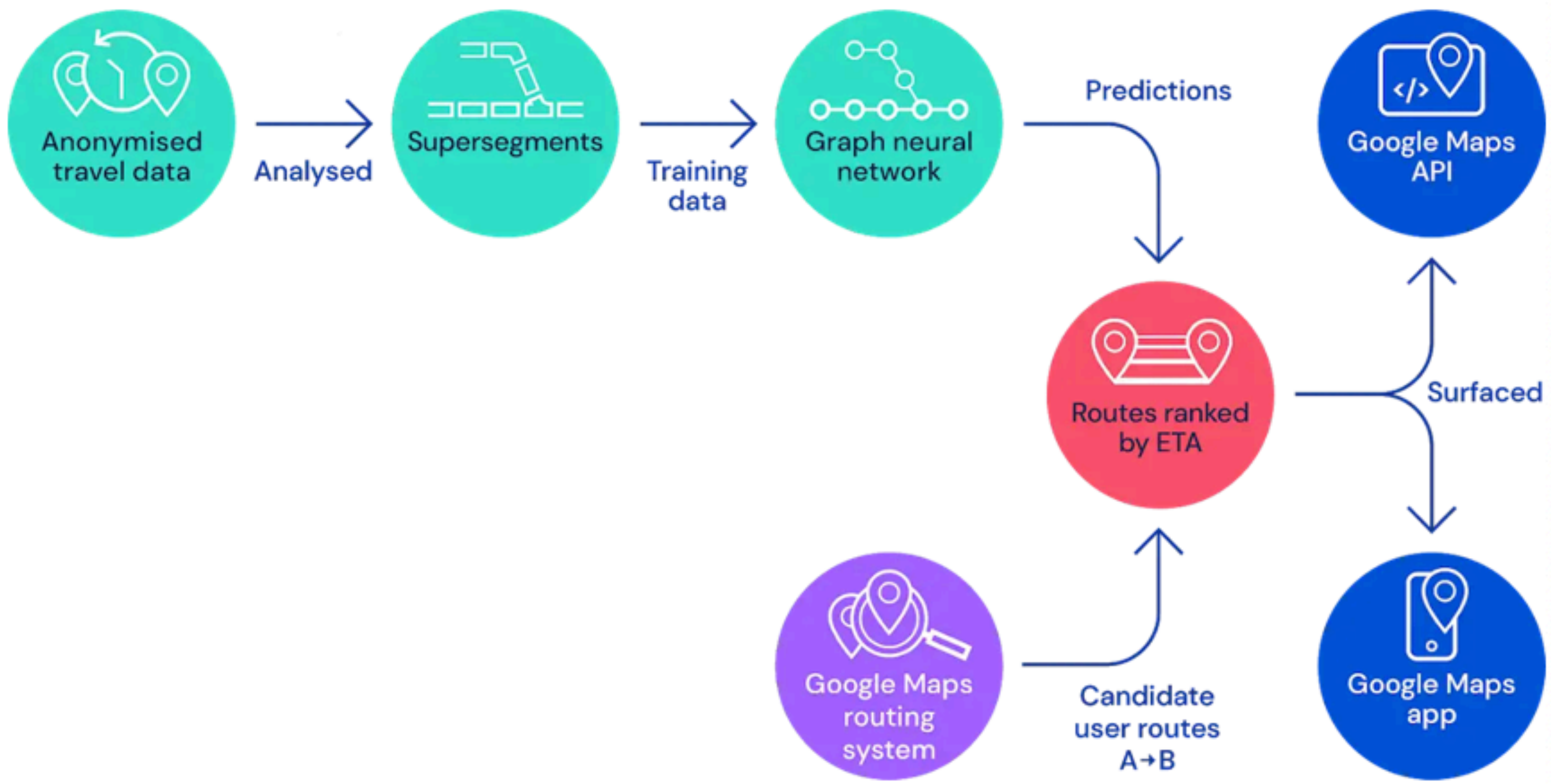
Simonovsky et al, 2017, De Cao et al 2018, Stokes et al 2020

Traffic prediction



[Derrow-Pinion et al., 2021]

Traffic prediction

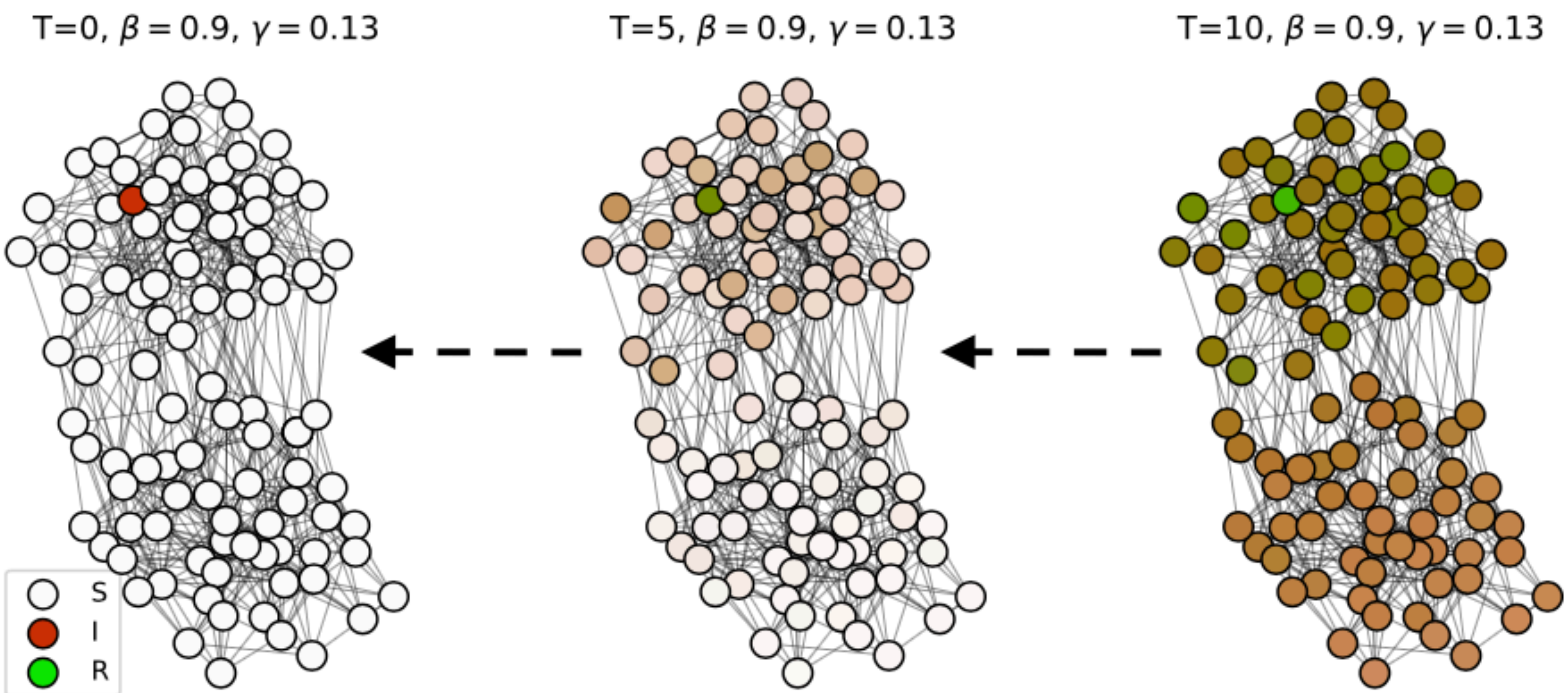


The model architecture for determining optimal routes and their travel time.

[Derrow-Pinion et al., 2021]

GNN for COVID-19

- Use GNN to locate the source of the epidemics [Shah et al. 2020]



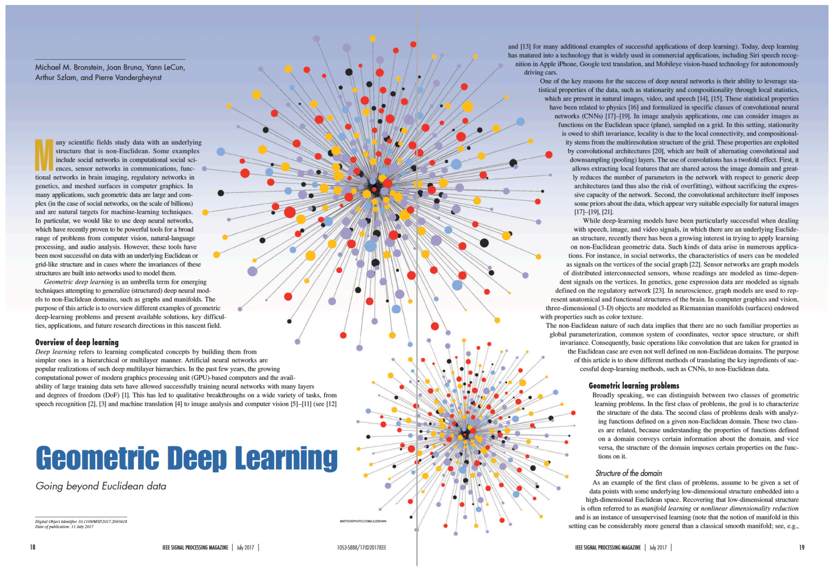
Concluding remarks

- Deep learning on graphs
  - Emerging field that extends data analysis to irregular domains
  - Highly disciplinary topic: machine learning, signal processing, graph theory, harmonic analysis, statistics
  - Spectral- and spatial-domain approaches: different frameworks, significant overlaps
  - More and more applications are emerging

Open issues/Future directions

- Scalability issues
- Limited theoretical understanding
- Lack of performance guarantees; vulnerability to adversarial attacks
- Lack of interpretability
- Dealing with dynamic graphs
- Incorporating higher-order structures into GNNs
- Lack of standardized benchmarks
- https://towardsdatascience.com/graph-deep-learning/home

References



Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst

Many scientific fields study data with an underlying structure that is non-Euclidean. Some examples include social networks in computational social sciences, sensor networks in communications, functional networks in brain imaging, regulatory networks in genetics, and meshed surfaces in computer graphics. In many applications, such geometric data are large and complex (in the case of social networks, on the scale of billions) and are natural targets for machine-learning techniques. In particular, we would like to use deep neural networks, which have recently proven to be powerful tools for a broad range of problems from computer vision, natural-language processing, and audio analysis. However, these tools have been most successful on data with an underlying Euclidean or grid-like structure and in cases where the invariances of these structures are built into networks used to model them.

Geometric deep learning is an umbrella term for emerging techniques attempting to generalize (structured) deep neural models to non-Euclidean domains, such as graphs and manifolds. The purpose of this article is to overview different examples of geometric deep-learning problems and present available solutions, key difficulties, applications, and future research directions in this nascent field.

Overview of deep learning

Deep learning refers to learning complicated concepts by building them from simpler ones in a hierarchical or multilayer manner. Artificial neural networks are popular realizations of such deep multilayer hierarchies. In the past few years, the growing computational power of modern graphics processing unit (GPU)-based computers and the availability of large training data sets have allowed successfully training neural networks with many layers and degrees of freedom (DoF) [1]. This has led to qualitative breakthroughs on a wide variety of tasks, from speech recognition [2], [3] and machine translation [4] to image analysis and computer vision [5]–[11] (see [12]

Geometric Deep Learning

Going beyond Euclidean data

Digital Object Identifier 10.1109/MSP.2017.2693418
Date of publication: 11 July 2017

©ISTOCKPHOTO.COM/LIUZISHAN

18 IEEE SIGNAL PROCESSING MAGAZINE | July 2017 | 1053-5888/17©2017IEEE

and [13] for many additional examples of successful applications of deep learning). Today, deep learning has matured into a technology that is widely used in commercial applications, including Siri speech recognition in Apple iPhone, Google text translation, and Mobileye vision-based technology for autonomously driving cars.

One of the key reasons for the success of deep neural networks is their ability to leverage statistical properties of the data, such as stationarity and compositionality through local statistics, which are present in natural images, video, and speech [14], [15]. These statistical properties have been related to physics [16] and formalized in specific classes of convolutional neural networks (CNNs) [17]–[19]. In image analysis applications, one can consider images as functions on the Euclidean space (plane), sampled on a grid. In this setting, stationarity is owed to shift invariance, locality is due to the local connectivity, and compositionality stems from the multiresolution structure of the grid. These properties are exploited by convolutional architectures [20], which are built of alternating convolutional and downsampling (pooling) layers. The use of convolutions has a twofold effect. First, it allows extracting local features that are shared across the image domain and greatly reduces the number of parameters in the network with respect to generic deep architectures (and thus also the risk of overfitting), without sacrificing the expressive capacity of the network. Second, the convolutional architecture itself imposes some priors about the data, which appear very suitable especially for natural images [17]–[19], [21].

While deep-learning models have been particularly successful when dealing with speech, image, and video signals, in which there are an underlying Euclidean structure, recently there has been a growing interest in trying to apply learning on non-Euclidean geometric data. Such kinds of data arise in numerous applications. For instance, in social networks, the characteristics of users can be modeled as signals on the vertices of the social graph [22]. Sensor networks are graph models of distributed interconnected sensors, whose readings are modeled as time-dependent signals on the vertices. In genetics, gene expression data are modeled as signals defined on the regulatory network [23]. In neuroscience, graph models are used to represent anatomical and functional structures of the brain. In computer graphics and vision, three-dimensional (3-D) objects are modeled as Riemannian manifolds (surfaces) endowed with properties such as color texture.

The non-Euclidean nature of such data implies that there are no such familiar properties as global parameterization, common system of coordinates, vector space structure, or shift invariance. Consequently, basic operations like convolution that are taken for granted in the Euclidean case are even not well defined on non-Euclidean domains. The purpose of this article is to show different methods of translating the key ingredients of successful deep-learning methods, such as CNNs, to non-Euclidean data.

Geometric learning problems

Broadly speaking, we can distinguish between two classes of geometric learning problems. In the first class of problems, the goal is to characterize the structure of the data. The second class of problems deals with analyzing functions defined on a given non-Euclidean domain. These two classes are related, because understanding the properties of functions defined on a domain conveys certain information about the domain, and vice versa, the structure of the domain imposes certain properties on the functions on it.

Structure of the domain

As an example of the first class of problems, assume to be given a set of data points with some underlying low-dimensional structure embedded into a high-dimensional Euclidean space. Recovering that low-dimensional structure is often referred to as *manifold learning* or *nonlinear dimensionality reduction* and is an instance of unsupervised learning (note that the notion of manifold in this setting can be considerably more general than a classical smooth manifold; see, e.g.,

IEEE SIGNAL PROCESSING MAGAZINE | July 2017 | 19

A Comprehensive Survey on Graph Neural Networks

Zonghan Wu, Shirui Pan, *Member, IEEE*, Fengwen Chen, Guodong Long, Chengqi Zhang, *Senior Member, IEEE*, Philip S. Yu, *Fellow, IEEE*

Abstract—Deep learning has revolutionized many machine learning tasks in recent years, ranging from image classification and video processing to speech recognition and natural language understanding. The data in these tasks are typically represented in the Euclidean space. However, there is an increasing number of applications where data are generated from non-Euclidean domains and are represented as graphs with complex relationships and interdependency between objects. The complexity of graph data has imposed significant challenges on existing machine learning algorithms. Recently, many studies on extending deep learning approaches for graph data have emerged. In this survey, we provide a comprehensive overview of graph neural networks (GNNs) in data mining and machine learning fields. We propose a new taxonomy to divide the state-of-the-art graph neural networks into different categories. With a focus on graph convolutional networks, we review alternative architectures that have recently been developed; these learning paradigms include graph attention networks, graph autoencoders, graph generative networks, and graph spatial-temporal networks. We further discuss the applications of graph neural networks across various domains and summarize the open source codes and benchmarks of the existing algorithms on different learning tasks. Finally, we propose potential research directions in this fast-growing field.

Index Terms—Deep Learning, graph neural networks, graph convolutional networks, graph representation learning, graph autoencoder, network embedding

Geometric Deep Learning
Grids, Groups, Graphs, Geodesics, and Gauges

Michael M. Bronstein[1], Joan Bruna[2], Taco Cohen[3], Petar Veličković[4]

May 4, 2021

https://www.cs.mcgill.ca/~wlh/grl_book/

https://github.com/DeepGraphLearning/LiteratureDL4Graph

https://github.com/thunlp/NRLPapers

https://github.com/thunlp/GNNPapers

Useful resources

- **Toolboxes**
  - https://github.com/rusty1s/pytorch_geometric
  - https://github.com/dmlc/dgl
- **Datasets**
  - https://chrsmrrs.github.io/datasets/
  - https://ogb.stanford.edu

Thank you for your attention!