Modified from the course material of:
Nara Institute of Science and Technology
Data Science Special Lecture

An Introduction to Graph Neural Networks: basics and applications

Katsuhiko ISHIGURO, Ph. D (Preferred Networks, Inc.)

Oct. 23, 2020

Take home message

- Graph Neural Networks (GNNs): Neural Networks (NNs) to compute nodes' representation of graph-structured data
- Practical applications in industry, hard competitions in academia
- Model: The fundamental is approximated Graph Convolution
- Applications: applicable in several tasks in different domains

Table of Contents (1/2)

Table of Contents (2/2)

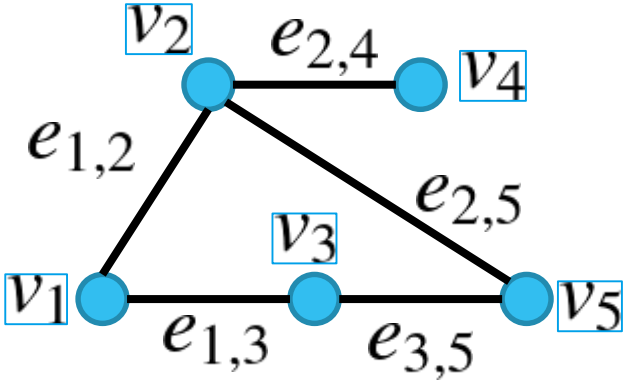
Acknowledgements

Many thanks for helpful comments and provided materials!!

Daisuke Okanohara, Shin-ichi Maeda, Kentaro Minami, Yuta Tsuboi, Sousuke Kobayashi, Kenta Oono, Hiroshi Maruyama (Preferred Networks)

Graph: for relational data

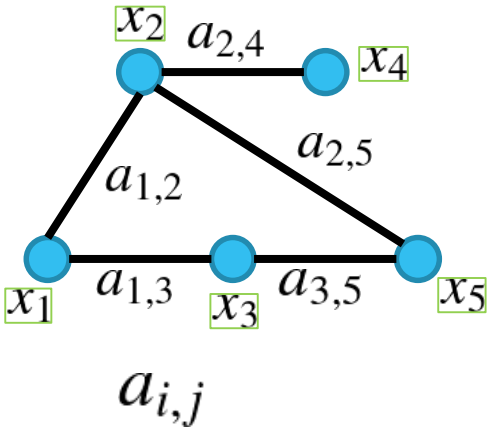
- Graph $G = (\mathcal{V}, \mathcal{E})$
- Set of nodes (vertices): $\mathcal{V} = \{v_i\}$
- Set of edges: $\mathcal{E} = \{e_{i,j}\}$
  - directional/non-directional



- Especially interested in Attributed graph
  - Nodes and/or edges have some features (label, numbers, vectors)

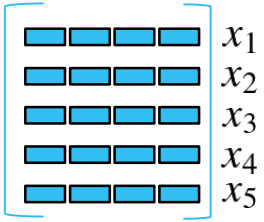
Matrix representation of Attributed graph $G = (X, A)$

- Suitable for formulations/implementations of GNNs



Feature Matrix $X = (x_{i,d}) = (x_1\ x_2\ \ldots)^T$

$X \in \mathbb{R}^{N \times D}$ N nodes, D dim. features



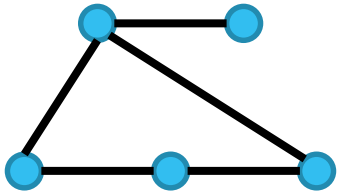
Adjacency Matrix $A = (a_{i,j})_{i,j=1,2,\ldots}$

$A \in \{0,1\}^{N \times N}$ Edge existence between N X N node pairs

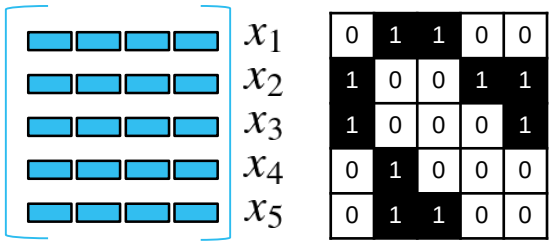
0	1	1	0	0
1	0	0	1	1
1	0	0	0	1
0	1	0	0	0
0	1	1	0	0

Symmetric A <-- non-directional edges

GNN Functionality: compute latent representation H



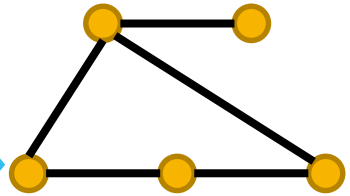
$G = (X, A)$

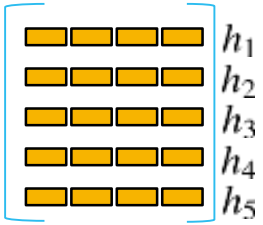


$X \in \mathbb{R}^{N \times D}$ $A \in \{0,1\}^{N \times N}$

L-layered GNN

Latent repr. Matrix H of nodes





$H \in \mathbb{R}^{N \times D'}$

Node Prediction tasks

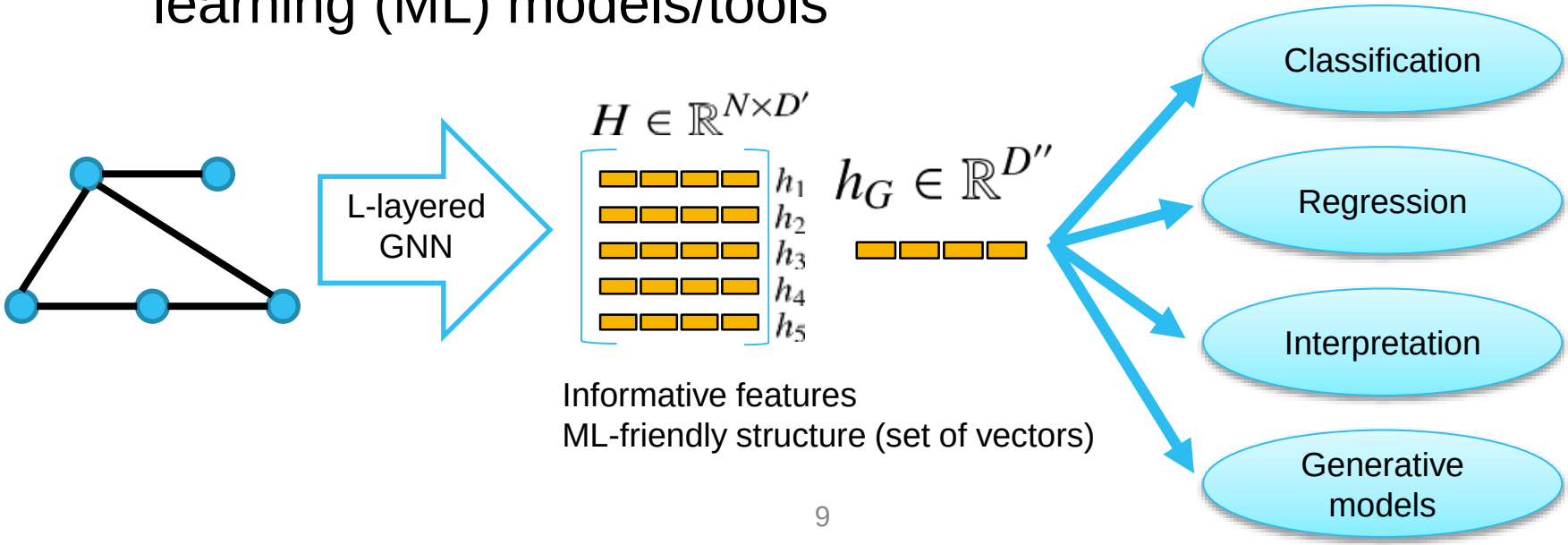
Latent repr. Vector h_G of a graph

$h_G \in \mathbb{R}^{D''}$

Graph Prediction tasks

Interface for intelligent processes on graphs

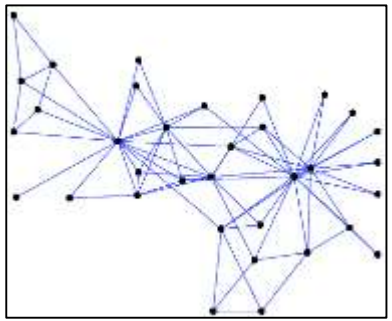
- H is informative and easy-to-handle “data” for machine-learning (ML) models/tools



Informative features
ML-friendly structure (set of vectors)

Application Domains of GNNs: concrete applications

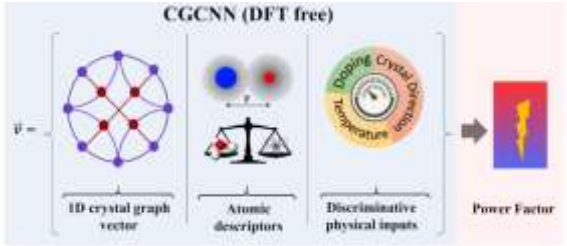
Social Network

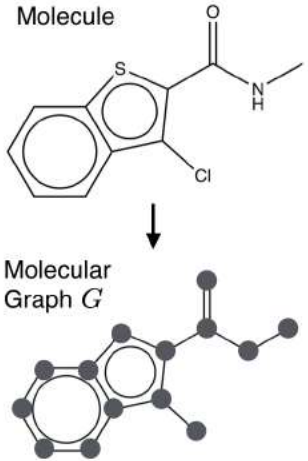


[http://konect.uni-koblenz.de/networks/ucidata-zachary]

Chemical Molecular Graph

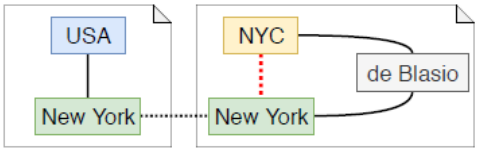
[Jin18JTVAE, Laugier18CGCNN]





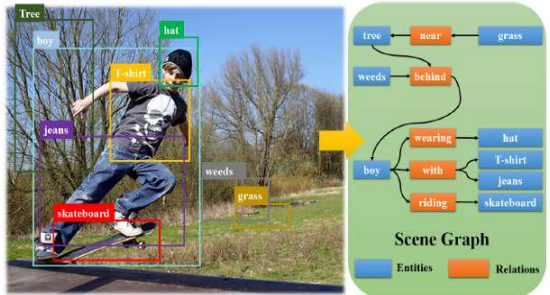
Knowledge Graph

[DeCao19QA2]



Scene Graph

[Qi19AttentiveRN]

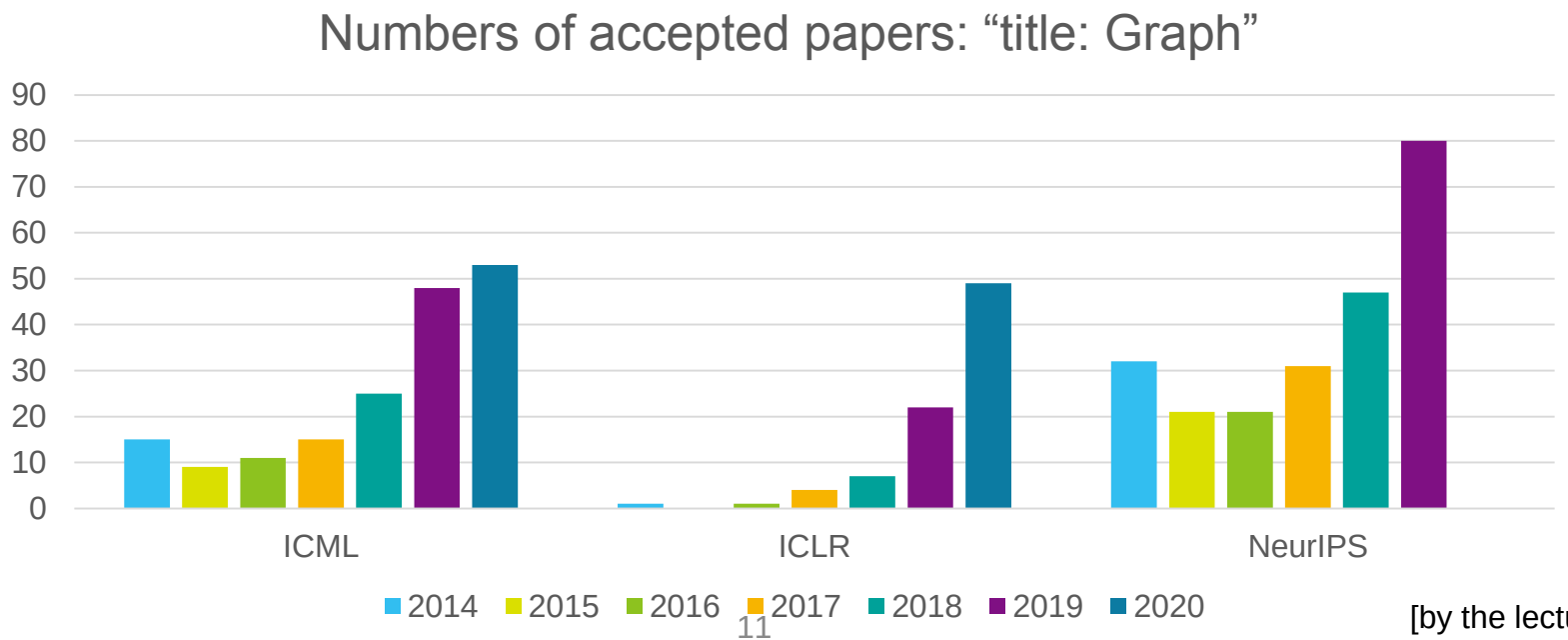


Pointcloud [Landrieu_Boussaha19PCS]

Physical Simulator [Sanchez-Gonzalez20GNS]

GNN in Machine Learning Researches

~2016: “graphical models”, “graph kernels”, and pure graph theory works
2017~: GNN works add up the counts



[by the lecturer]

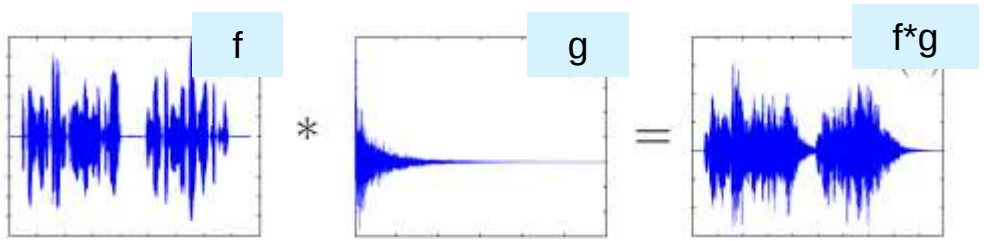
The basic and the de-facto standard: GCN

Convolution

- Modify a signal f by a filter g, accumulate over ``shift'' t on the coordinate (axes) x

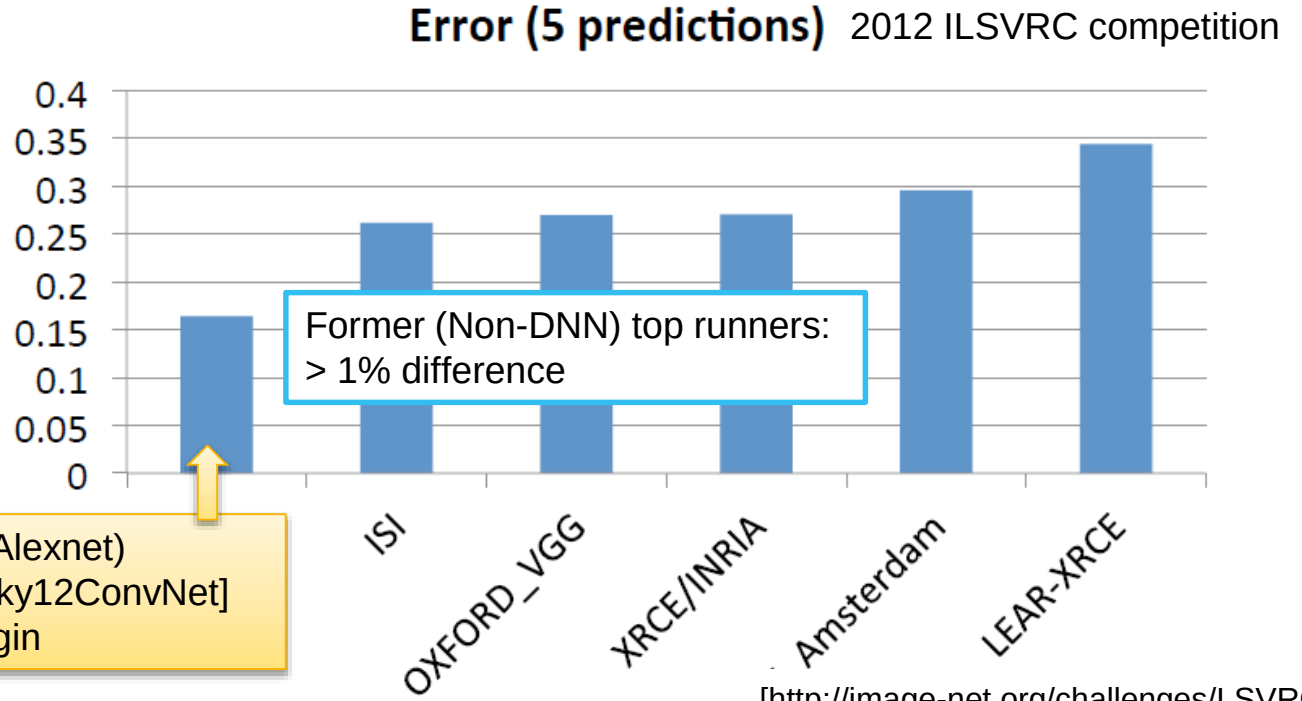
$$(f * g)(x) = \int f(x)g(x-t)dt \qquad (f * g)(x) = \sum_t f(x)g(x-t)$$

1D signal example
f: Sound signal
g: Impulse response
→ Hall resonated sound (reverb)



abcpedia.acoustics.jp

E.g. ConvNet in computer vision [Krizhevsky12ConvNet]



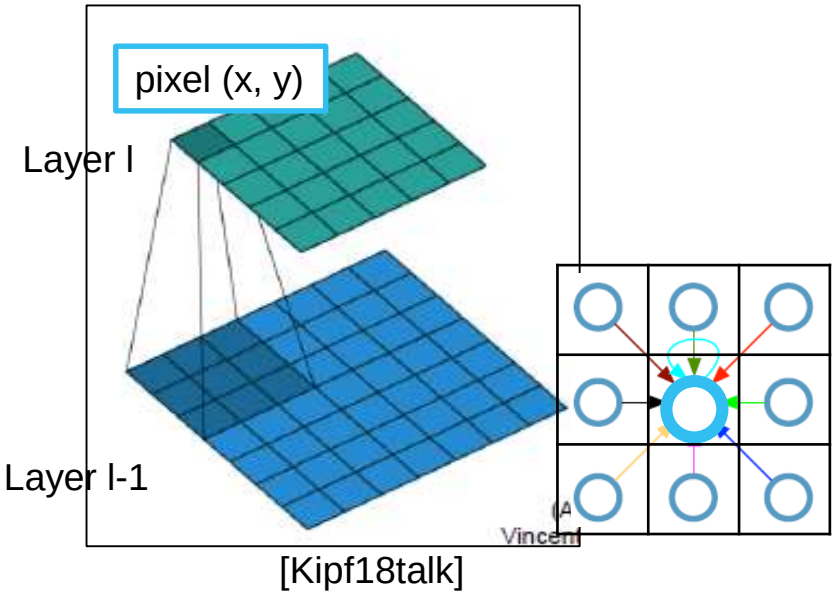
[http://image-net.org/challenges/LSVRC/2012/ilsvrc2012.pdf]

$$(f * g)(x) = \sum_t f(x)g(x-t)$$

E.g. ConvNet in computer vision

[Krizhevsky12ConvNet]

- L- iterative conv. Operations for image pixels



[Kipf18talk]

Image signal (feature) at pixel (x,y) of layer l

$$h^{\ell}_{x,y} = \left(h^{\ell-1} * W^{\ell}\right)_{x,y}$$

$$= \sigma\left(\sum_{i,j\in\{-1,0,1\}} h^{\ell-1}_{x+i,y+j} \times W^{\ell}_{i,j}\right)$$

shift signal filter

How can we define Conv. in graphs?

- No “axes” in graphs unlike sound signals (time) and image signals (x, y)
- What is “coordinate”? What is “shift”?

[いらすとや]

Spectral approach on Graphs

Fourier Transform and Convolution

Convolutions in the signal (spatial) domain = Multiplication in the spectral (frequency) domain

$$\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(g)$$

1D case:

$$\mathcal{F}(f(x)) = \int f(x) e^{(-2\pi i x k)} dx$$

K-th frequency'sFourier base function

For Graphs, what is the signal? How the FT defined??

Def: Graph signals

- Simply concatenate numbers
- 1D node attribute case: N-array

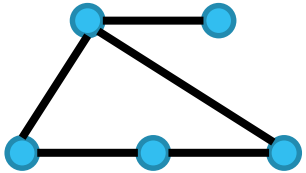
$$x = (x_1, x_2, \ldots, x_N)^T \in \mathbb{R}^N$$

- D-dim attribute case → Feature matrix

$$X = (x_{i,d}) = (x_1\ x_2\ \ldots)^T \quad X \in \mathbb{R}^{N \times D}$$

[Ma20tutorial]

Def: Graph Laplacian and Normalized Laplacian



$$G = (X, A)$$

0	1	1	0	0
1	0	0	1	1
1	0	0	0	1
0	1	0	0	0
0	1	1	0	0

Adjacency Matrix

$$A \in \{0, 1\}^{N \times N}$$

2	0	0	0	0
0	3	0	0	0
0	0	2	0	0
0	0	0	1	0
0	0	0	0	2

Degree Matrix

$$D = \mathrm{Diag}\left(\sum_i a_{1,i}, , \sum_i a_{2,i}, \dots\right)$$

Graph Laplacian Matrix

$$L = D - A$$

2	-1	-1	0	0
-1	3	0	-1	-1
-1	0	2	0	-1
0	-1	0	1	0
0	-1	-1	0	2

Normalized Graph Laplacian

$$\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$$

1	−1/2	−1/2	0	0
−1/3	1	0	−1/3	−1/3
−1/2	0	1	0	−1/2
0	-1	0	1	0
0	−1/2	−1/2	0	1

Graph Fourier Transform = Multiply Eigenvectors of (Normalized) Graph Laplacians

i-th eigenvevtor = i-th graph Fourier base

$$u_i^T \mathcal{L} u_i = u_i^T \lambda_i u_i = \lambda_i$$

i-th eigenvalue = frequency of i-th base

Graph Fourier Transform(GFT)

Graph signal in spectral (freq.) space

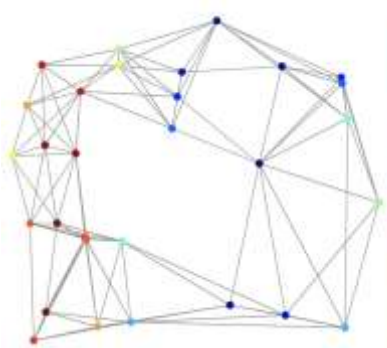
$$\tilde{x} = \sum_i u_i^T x = U^T x$$

Inner product between the graph signal (x) and Fourier bases (U)

$$U = [u_1 u_2 \dots u_N] \qquad \Lambda = \mathrm{Diag}(\lambda_1, \lambda_2, \dots, \lambda_N)$$

$$0 \le \lambda_1 \le \lambda_2 \le \cdots \le \lambda_N$$

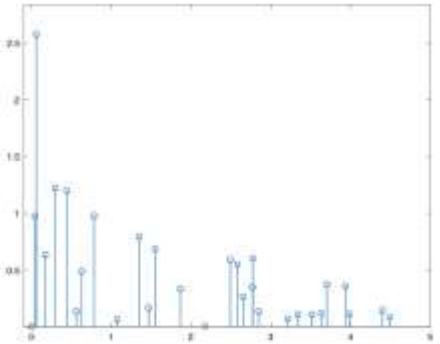
Graph Fourier Transform: just multiply U^T or U



Spatial domain: x

GFT $\tilde{x} = U^T x$

IGFT $x = U\tilde{x}$



Spectral domain:$\hat{x}$

[Ma20tutorial,Shuman13Graph]

Graph Convolution via GFT

- Convolution is a multiplication in Fourier space
  and GFT as well $\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(g)$

GFT of the graph signal $\tilde{x} = \mathrm{GFT}(x) = U^T x$

Convolution = simple multiplication in spectral $W_\Lambda \cdot \tilde{x} = W_\Lambda U^T x$

IGFT to get conv-filtered graph signal $\mathrm{conv}(x) = \mathrm{IGFT}(W_\Lambda \tilde{x}) = U W_\Lambda U^T x$

$$(f * g)(x) = \sum_t f(x) g(x - t)$$

Naïve Graph Convolution

$$\text{conv}(x) = U W_\Lambda U^T x$$

Two parameters: Fixed U (eigenvectors of the Laplacian = Fourier bases over freq.)

Trainable $W_\Lambda = \text{Diag}(\theta_{\lambda_1}, \theta_{\lambda_2}, \ldots, \theta_{\lambda_N})$

(amplify each spectral coefficients)

☺ Simple linear algebra for a generic Graph Conv.

☹ Eig.Decompose of (normalized) graph Laplacians O(N^3) for **each graph**

Approximation: ChebyNet [Defferrard16ChebNet]

- Model N trainable param. with K (<<N) order Chebychev Polynomials

$$W_\Lambda = \mathrm{Diag}\left(\theta_{\lambda_1}, \theta_{\lambda_2}, \ldots, \theta_{\lambda_N}\right)$$

$$\Rightarrow \sum_{k=0}^{K} w_k \mathcal{T}_k\left(\tilde{\Lambda}\right) \quad \tilde{\Lambda} = \frac{2}{\lambda_N}\Lambda - I$$

$$\Lambda = \mathrm{Diag}\left(\lambda_1, \lambda_2, \ldots, \lambda_N\right)$$

Chebychev Polynomial

$$\mathcal{T}_k(x) = 2x\mathcal{T}_{k-1}(x) - \mathcal{T}_{k-2}$$
$$\mathcal{T}_1(x) = x, \ \mathcal{T}_0(x) = 1$$

No U-s = No Eig. Decomp.

$$\text{conv}\,(x) = U W_\Lambda U^T x$$

$$= U \left[\sum_{k=0}^{K} w_k \mathcal{T}_k \left(\tilde{\Lambda} \right) \right] U^T x \; = \left[\sum_{k=0}^{K} w_k \mathcal{T}_k \left(\frac{2}{\lambda_N} \mathcal{L} - I \right) \right] x$$

☺ Graph Laplacian is sparse in nature
→ Polynomial L is lightweight << O(N^3)

GCN [Kipf_Welling17GCN]: much simpler Convolution

Set K = 1, and replace λ_N with 2 (the theoretical maximum value)

$$\text{conv}(x) = \left[\sum_{k=0}^{K} w_k \mathcal{T}_k \left(\frac{2}{\lambda_N}\mathcal{L} - I\right)\right] x$$

$$= \left[w_0 + w_1 (\mathcal{L} - I)\right] x$$

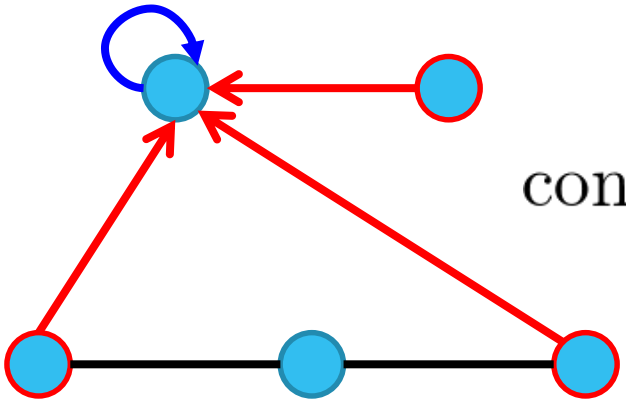
Assume w = -w1 = (w0-w1)

$$= w \left[I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right] x$$

Very simple GraphConv
- No Eig. Decomp.
- No Matrix Polynomials
- single tunable parameter (w)

spatial interpretation of GCN's convolution

- Weighted sum of neighbor nodes (shift) + self-link
- Number of neighbor nodes vary unlike image Cnvnet



$$\mathrm{conv}_{\mathrm{GCN}}(x) = w\left[I + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}\right]x$$

Self-link

Neighboring nodes

GCN formulation: multi-dim simpler conv + activation

Update rule of GCN ℓ-th layer

Non-linear activation (e.g. sigmoid, ReLU)

$$H^{\ell+1} = \mathrm{GCN}\left(H^{\ell}\right) = \sigma\left[\tilde{A}H^{\ell}W^{\ell}\right]$$

output: updated node vectors

Input: node vectors @ layer ℓ

Trainable weight parameter for node vector updates

$$W^{\ell} \in \mathbb{R}^{d_{\mathrm{in}} \times d_{\mathrm{out}}}$$

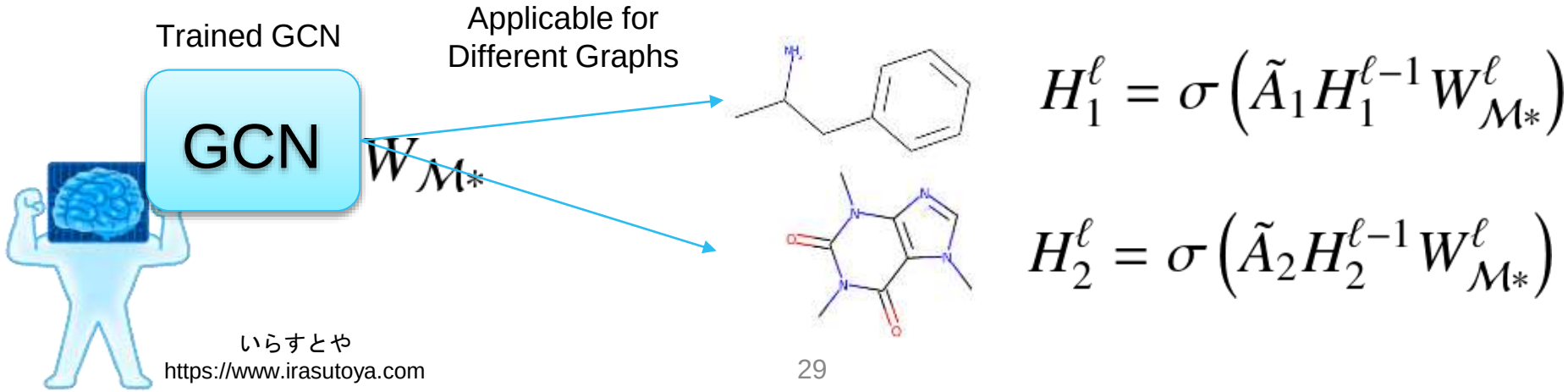
Fixed Adjacency parameter for node conv (mixing)

$$\tilde{A} = (D+I)^{-\frac{1}{2}}(A+I)(D+I)^{-\frac{1}{2}}$$

Once trained, run anywhere (on different topologies)

- The sole trainable parameter W does not care the adjacency

$$H^{\ell} = \sigma\left(\tilde{A} H^{\ell-1} W^{\ell}\right)$$



The simplest GCN… it works quite well

Table 3: Comparison of propagation models.

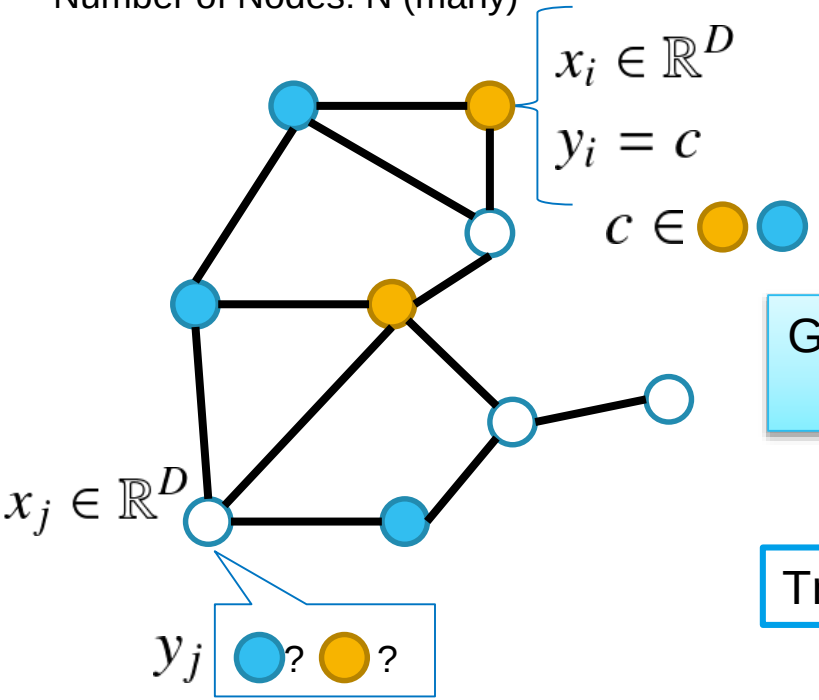
Description		Propagation model	Citeseer	Cora	Pubmed
Chebyshev filter (Eq. 5)	$K = 3$	$\sum_{k=0}^{K} T_k(\tilde{L})X\Theta_k$	69.8	79.5	74.4
	$K = 2$		69.6	81.2	73.8
1st-order model (Eq. 6)		$X\Theta_0 + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta_1$	68.3	80.0	77.5
Single parameter (Eq. 7)		$(I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})X\Theta$	69.3	79.2	77.4
Renormalization trick (Eq. 8)		$\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta$	**70.3**	**81.5**	**79.0**
1st-order term only		$D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta$	68.7	80.5	77.8
Multi-layer perceptron		$X\Theta$	46.5	55	71.4

[Kipf_Welling17GCN]

Perform better than the original ChebNet

Exemplar task: semi-supervised node classification with GCN

Number of Nodes: N (many)



Set of labeled nodes (supervisions)

$\mathcal{D}_{\text{labeled}} = \{x_i, y_i\}$

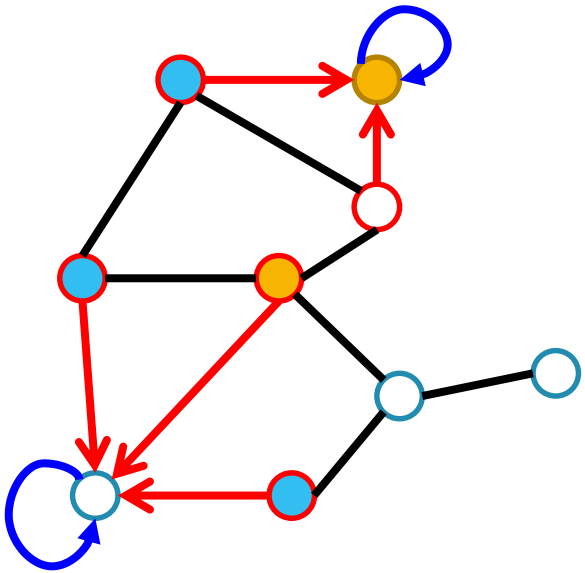
Set of unlabeled nodes

$\mathcal{D}_{\text{unlabeled}} = \{x_j\}$

Goal: predict the labels of unlabeled nodes y_j using a trained model (GCN and classifier)

Train: tune parameters of the model to recover y_i

Forward path



Apply L-layered GCN to update the latent node representations, H

$$H^{\ell} = \sigma\left(\tilde{A}H^{\ell-1}W^{\ell}\right) \quad \ell = 1, 2, \ldots, L$$

$$H^{\ell} = \left[h_1^{\ell} \cdots h_N^{\ell}\right]^T \quad h_i^0 = x_i$$

The initial node vector = node attributes

Classifier computes the label prob. distribution based on the final H

$$H = H^L \qquad H = [h_1 \cdots h_N]^T$$

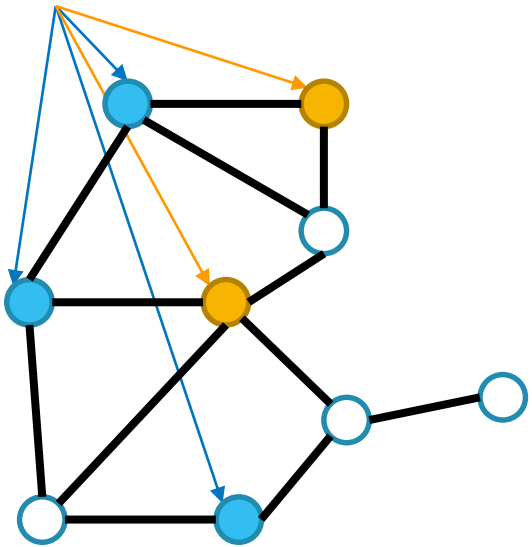
$$P(c|h) \propto \exp\left(\theta_c^T h\right)$$

C = {●, ●}

θ: trainable param of a classifier

Objective Function and Training

``Labeled''



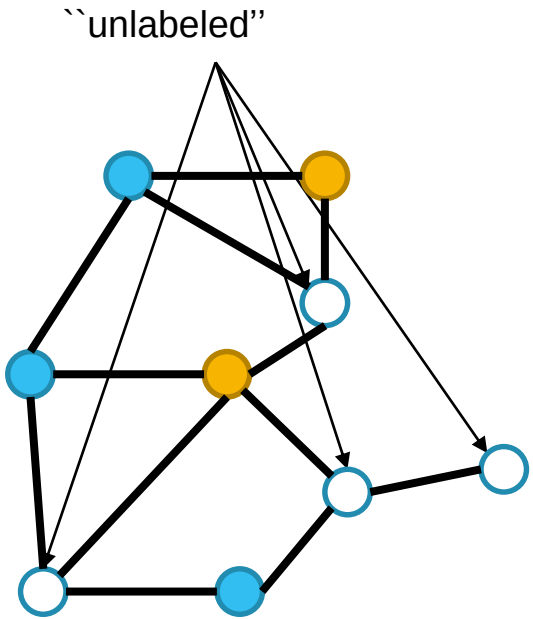
Objective function: cross entropy
≒ accuracy of label prediction on Labeled Nodes

$$\mathcal{L} = \sum_{(x_i, y_i) \in \mathcal{D}_{\text{labeled}}} \sum_{c} \delta(y_i = c) \log P(c|h_i)$$

Training: find the best parameters W(GCN) and θ(classifier) to minimize the negative of the objective function L

$$\hat{W}, \hat{\theta} = \arg\min_{W,\theta} -\mathcal{L}$$

Prediction using the Trained Model



Predict y_j (a label of node j) in unlabeled set, perform forward path from the observed x_j

$$x_j \in \mathcal{D}_{\text{unlabeled}}$$

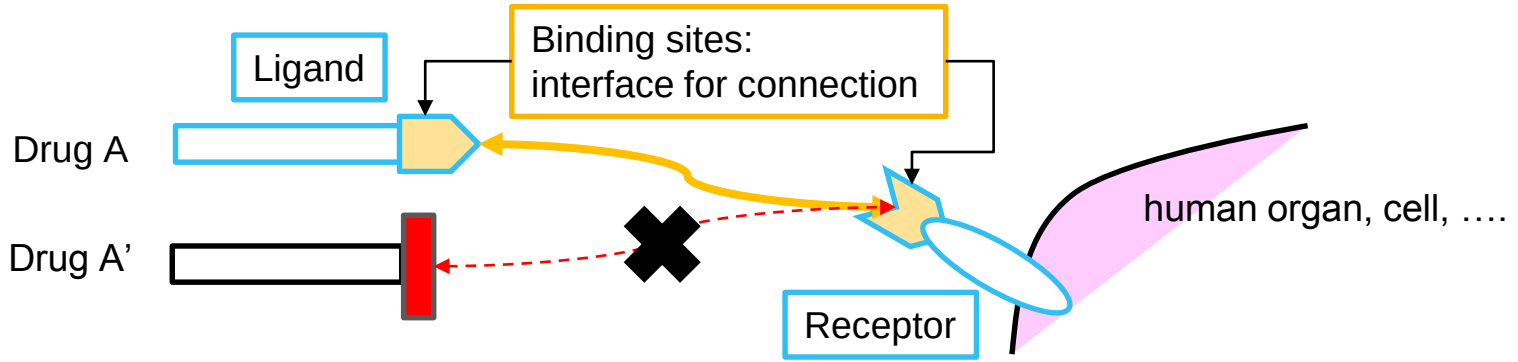
$$h_j \leftarrow H^{L}_{\hat{W}^L}\left(H^{L-1}_{\hat{W}^{L-1}}\left(\ldots\left(H^{1}_{\hat{W}^1}\left(x_j\right)\right)\right)\right)$$

$$y_j = \hat{c} = \arg\max_c \exp\left(\hat{\theta}_c^T h_j\right)$$

Application in Chemo-Informatics: Protein Interface Prediction

Interacting proteins

- Where is a binding site between two proteins?
  - Biding site: where a **ligand** and a **receptor** interacts



Predict binding site nodes by GNN [Fout17Interface]

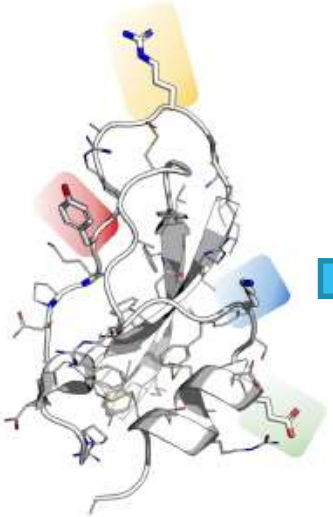
- Predict the binding site regions of ligand-receptor proteins
  - Possible binding = possible drug discovery and improvement

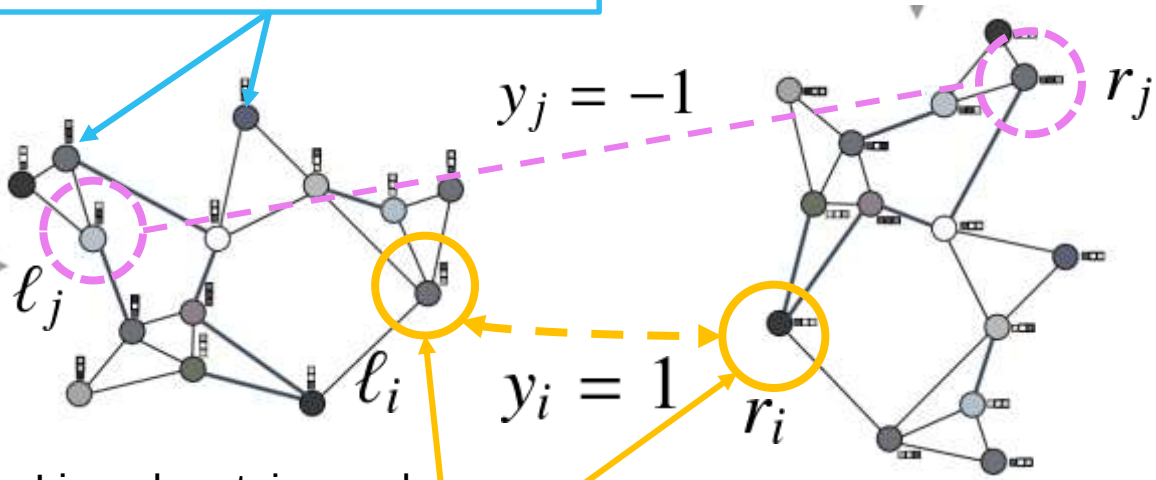
- Protein: a graph of inter-connected amino acids
- Use GNN for the binding prediction

[Sanyal20PGCN]

Attributed Protein Graphs

Node **(residue)**: a sub-structure consists of amino acids





Binding sites: specific nodes **interact between ligand and receptor**

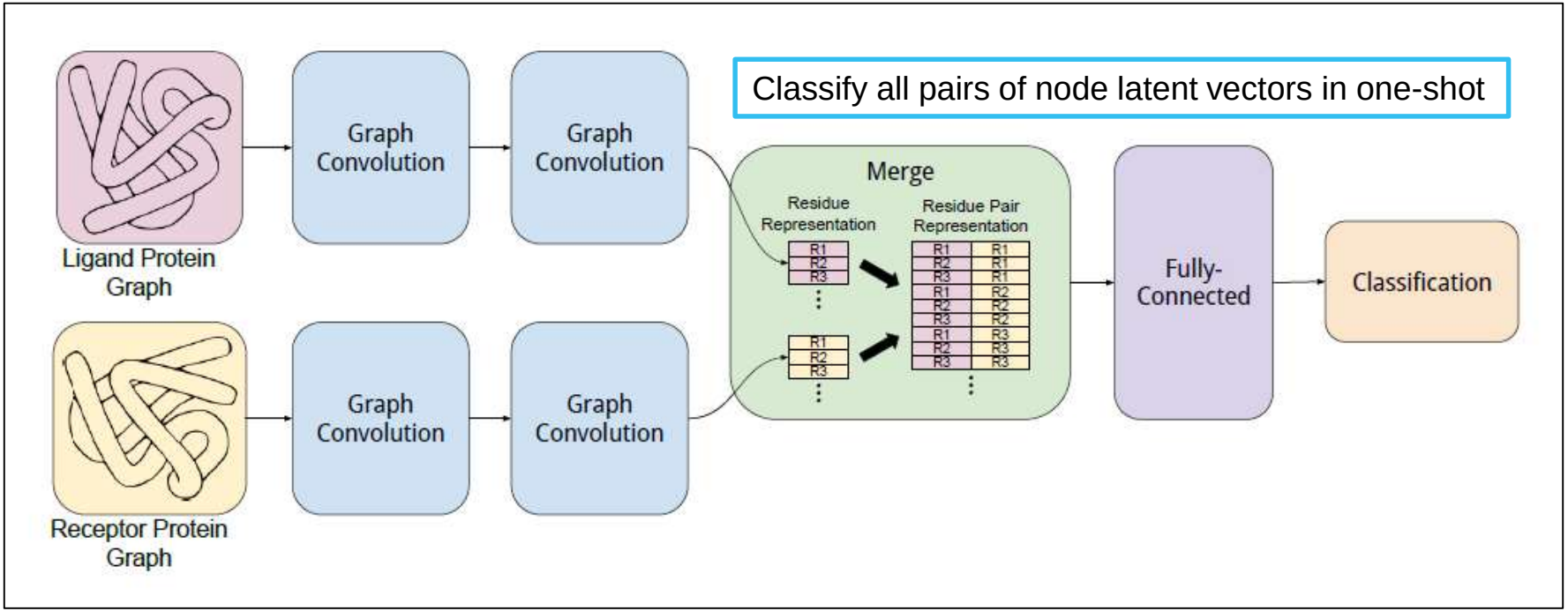
Training/Test of Interaction Prediction [Fout17Interface]

- Training: optimize parameters to predict the interaction label y_i between a node pair (ℓ_i , r_i) (for all training sampls)

$$y_i \in \{-1, 1\} \quad \overset{\text{minimize}}{\Longleftrightarrow} \quad \hat{y}_i = F_{\theta_F}\left[\text{GNN}_{\theta_L}(\ell_i), \text{GNN}_{\theta_R}(r_i)\right]$$

- Test: predict interactions of node pairs in unseen Ligand-Receptor pairs
  - "Train once, run any graphs (with different topology)"

architecture



[Fout17Interface]

GCN formulation choices [Fout17Interface]

Considering nodes features only

$$z_i = \sigma\left(W^{\mathrm{C}} x_i + \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} W^{\mathrm{N}} x_j + b\right), \quad (1)$$

Node features plus Edge features

$$z_i = \sigma\left(W^{\mathrm{C}} x_i + \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} W^{\mathrm{N}} x_j + \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} W^{\mathrm{E}} A_{ij} + b\right), \quad (2)$$

With ordering of nearest node orders

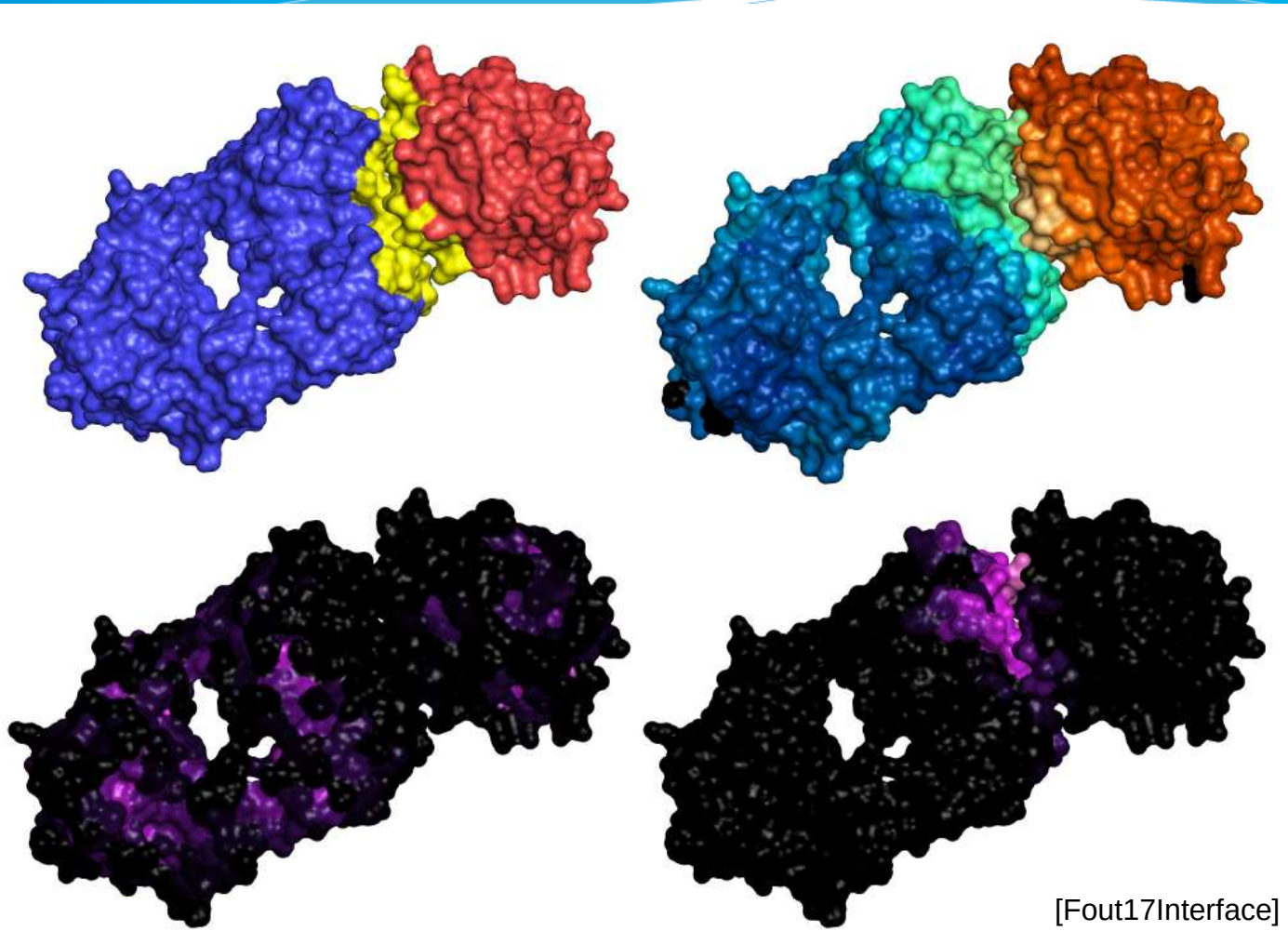
$$z_i = \sigma\left(W^{\mathrm{C}} x_i + \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} W_j^{\mathrm{N}} x_j + \frac{1}{|\mathcal{N}_i|}\sum_{j\in\mathcal{N}_i} W_j^{\mathrm{E}} A_{ij} + b\right). \quad (3)$$

Neighbor nodes j are ordered by distances from the node i

Results

Method	Convolutional Layers			
	1	2	3	4
No Convolution	**0.812 (0.007)**	0.810 (0.006)	0.808 (0.006)	0.796 (0.006)
Diffusion (DCNN) (2 hops) [5]	**0.790 (0.014)**	–	–	–
Diffusion (DCNN) (5 hops) [5])	**0.828 (0.018)**	–	–	–
Single Weight Matrix (MFN [9])	0.865 (0.007)	0.871 (0.013)	**0.873 (0.017)**	0.869 (0.017)
Node Average (Equation 1)	0.864 (0.007)	0.882 (0.007)	**0.891 (0.005)**	0.889 (0.005)
Node and Edge Average (Equation 2)	0.876 (0.005)	**0.898 (0.005)**	0.895 (0.006)	0.889 (0.007)
DTNN [21]	0.867 (0.007)	0.880 (0.007)	**0.882 (0.008)**	0.873 (0.012)
Order Dependent (Equation 3)	0.854 (0.004)	0.873 (0.005)	**0.891 (0.004)**	0.889 (0.008)

[Fout17Interface]

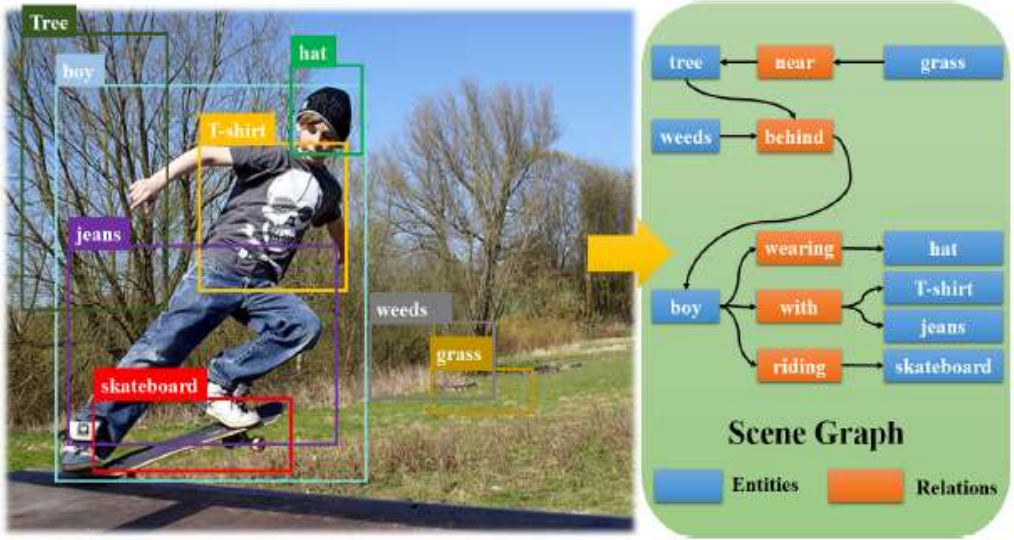


[Fout17Interface]

Computer Vision Application Example: Scene Graph generation

Scene Graph: summary graph of image contents

- Useful representation of images' understanding
- Want to generate a scene graph from image input

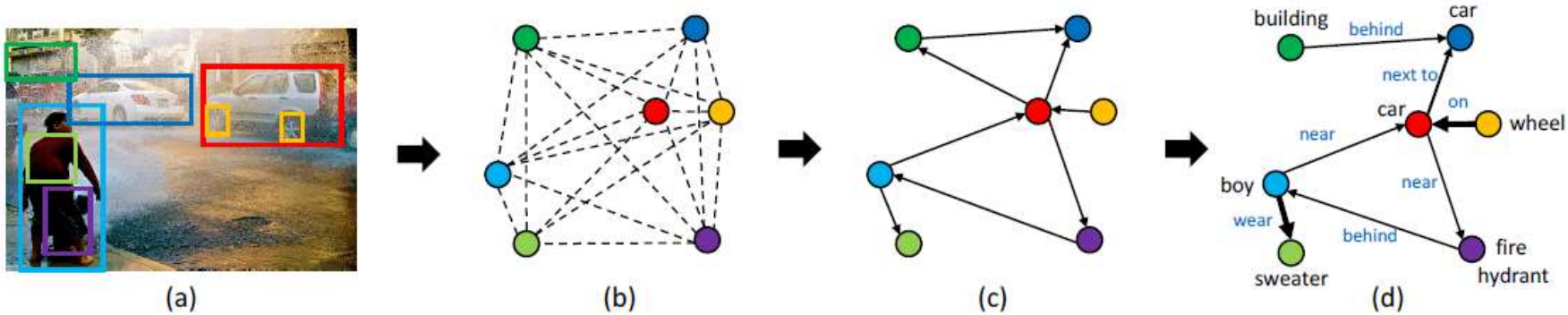


In a scene graph of an image,
node: object region in an image
edge: relation between object regions

[Qi19AttentiveRN]

Graph R-CNN for scene-graph generation
[Yang18GRCNN]

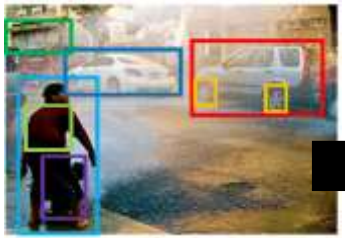
- Consists of three inner components
  - (a) → (b) Off-the-shelf object detector [Ren15RCNN] (omit)
  - (b) → (c) RePN to prune unnecessary branches (omit)
  - (c) → (d) Attentional GCN to predict labels of nodes and edges



[Yang18GRCNN]

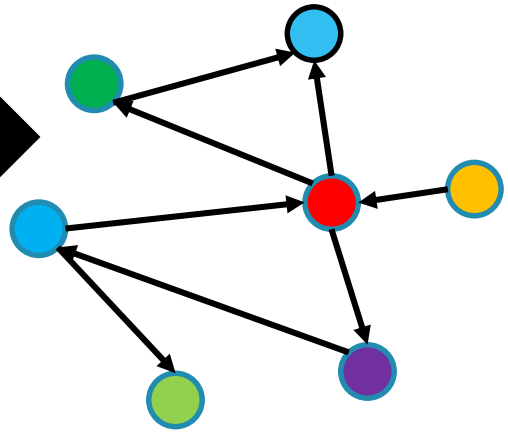
Expand Graph with Relation Nodes

One GNN can infer all objects and relations



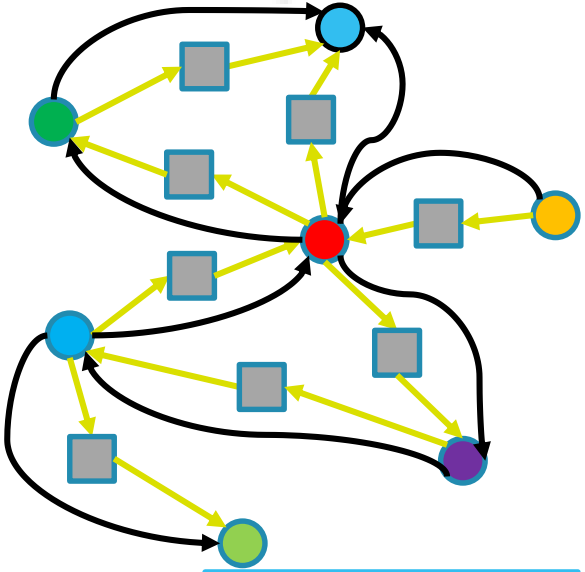
Objects detected

[Yang18GRCNN]



Original input graph
(sparsified by RePN):
node = object
edge = relation

Graph Expansion



Object node

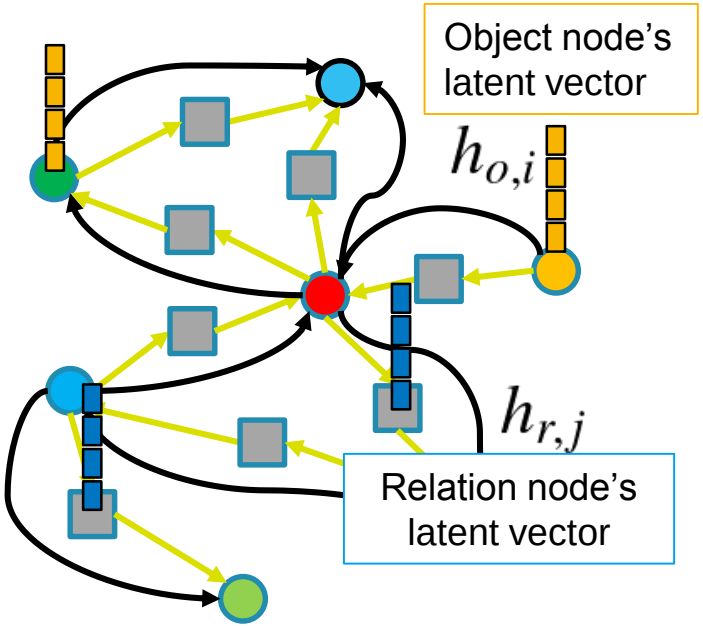
Obj-Obj edge

Relation node

Obj-Rel node

Attentional GCN [Yang18GRCNN]

- GCN update with Attention [Bahdanau15Attention] -based variable connection



GCN
Adjacency is fixed

$$H^{\ell} = \sigma\left(\tilde{A}H^{\ell-1}W^{\ell}\right)$$

Attentional GCN

$$H^{\ell} \propto \sigma\left(\hat{\mathrm{A}}H^{\ell-1}W^{\ell}\right)$$

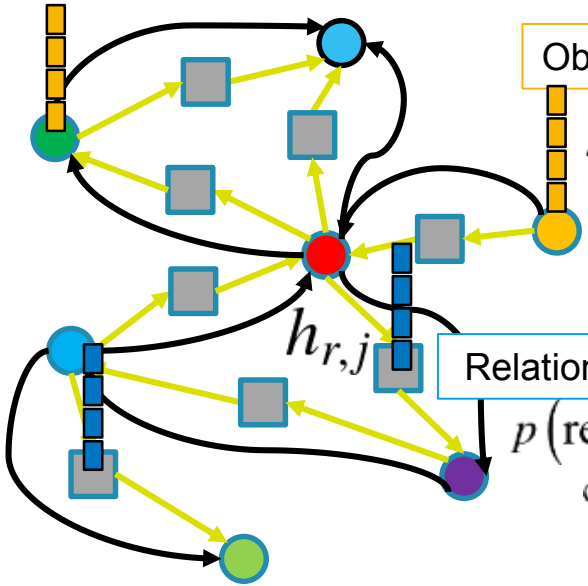
Attention conncection

$$\hat{\mathrm{A}}[k,l] = \begin{cases} \text{similarity}_{\phi}\left(h_k^{\ell}, h_l^{\ell}; \phi\right) & \text{nodes } (k,l) \text{ connected} \\ 0 & \text{disconnected} \end{cases}$$

Parameter ϕ can switch based on node types

Inferring labels of objects/relations

Expanded Graph



Object node's latent vector

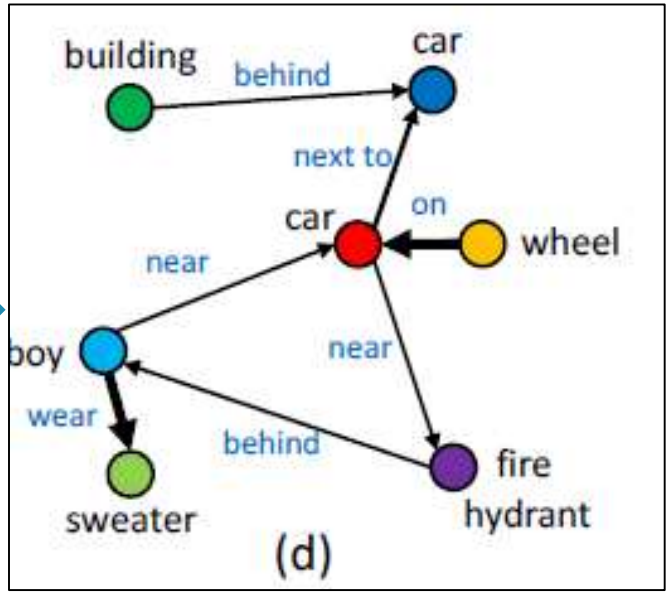
$h_{o,i}$ $p(\text{object}|h_{o,i}) \propto \exp\left(\theta_o^T h_{o,i}\right)$

$h_{r,j}$

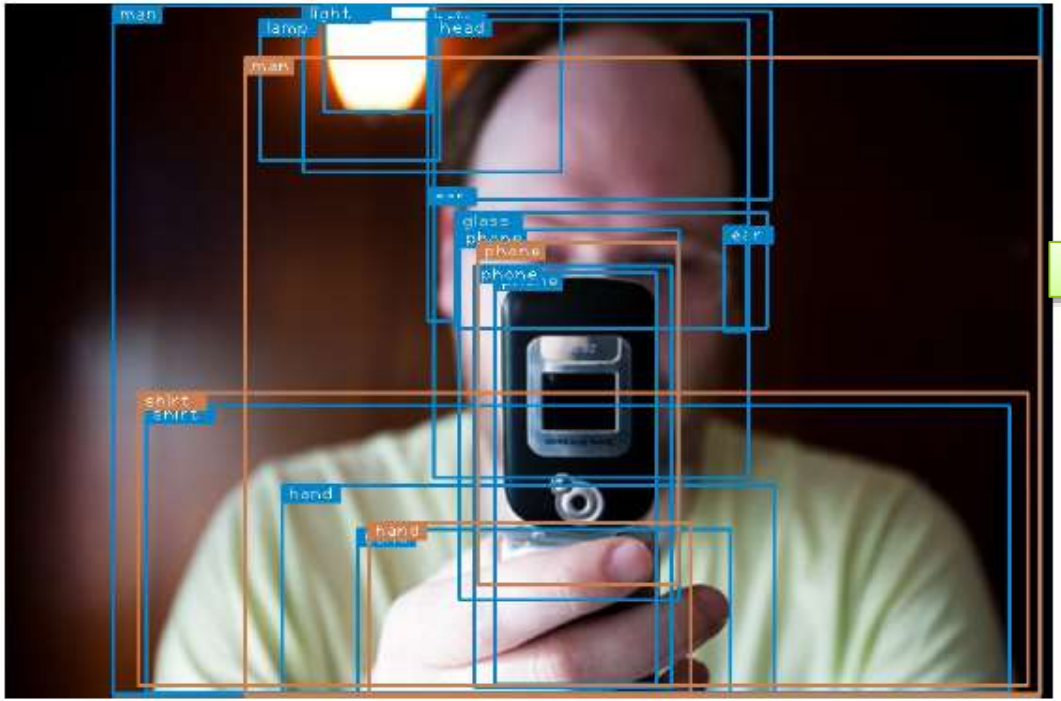
Relation node's latent vector

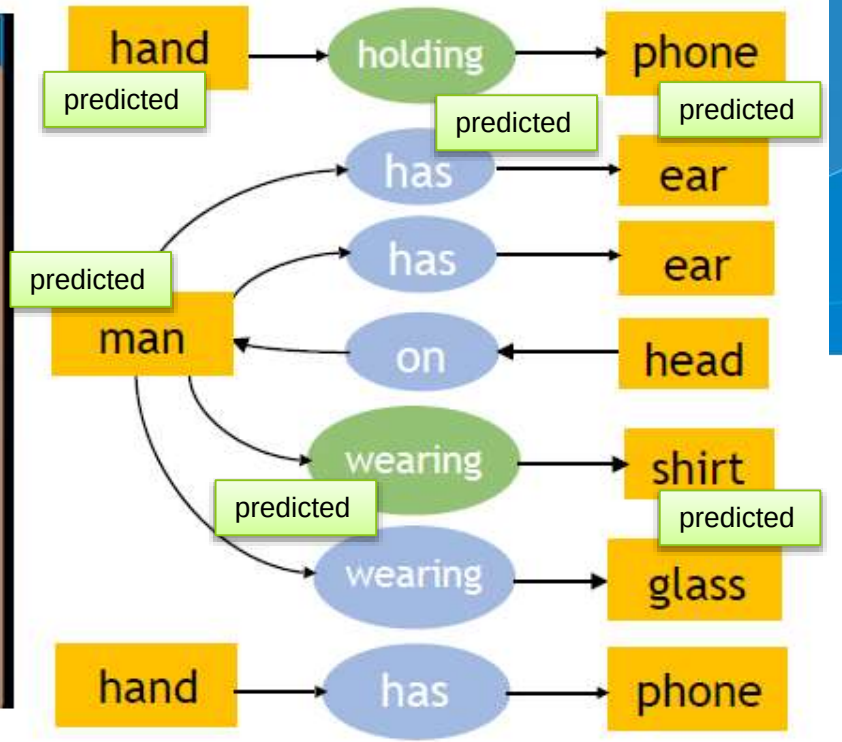
$p\left(\text{relation}|h_{r,j}\right) \propto \exp\left(\theta_r^T h_{r,j}\right)$

Ground Truth Scene Graph



[Yang18GRCNN]





[Yang18GRCNN]

Other Application of GNNs

- Segmentation of pointcloud w/ 10^5 samples [Hu20RandLaNet]
  - Decode laser rangefinder data into 3D obstacle map
- Particle-based Physical simulator [Sanchez-Gonzalez20GNS]
  - Simulate fluid dynamics via GNN
- Molecular Graph Generation [Jin18JTVAE, Madhawa19GVNP]
  - Generate a novel compound graph in computers

Theoretical issues

Theoretical Topics of GNN

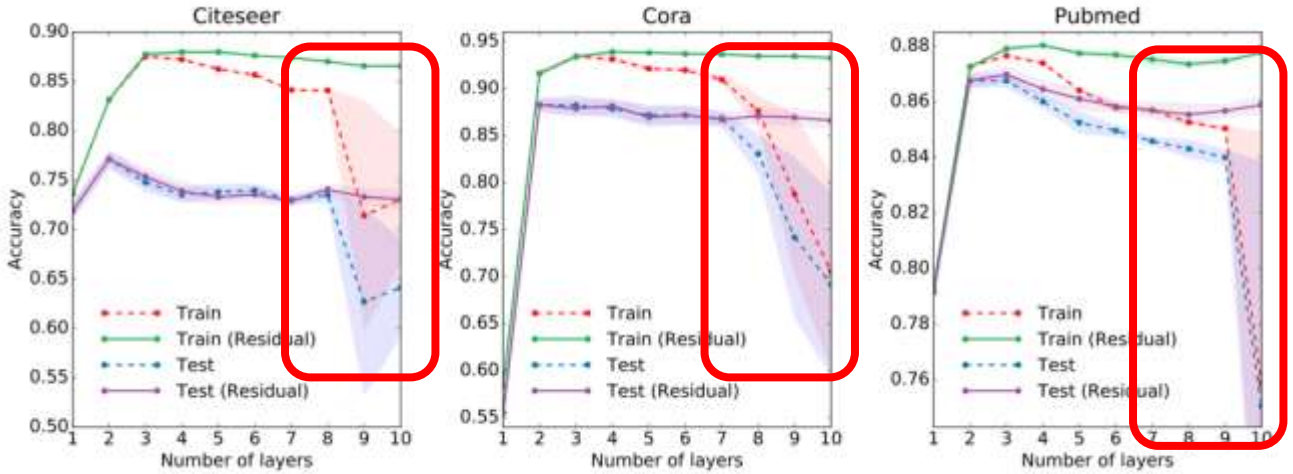
- “Deep” GNNs do no work well
  - “Oversmoothing”
  - Current solution: normalizer + residual connection

- The theoretical limit of representation powers of GNNs
  - Graph Isomorphism test
  - Invariance/Equivalence

``Deep" GNNs do not work well

Quite different from the successful deep Conv models in Computer vision

Better

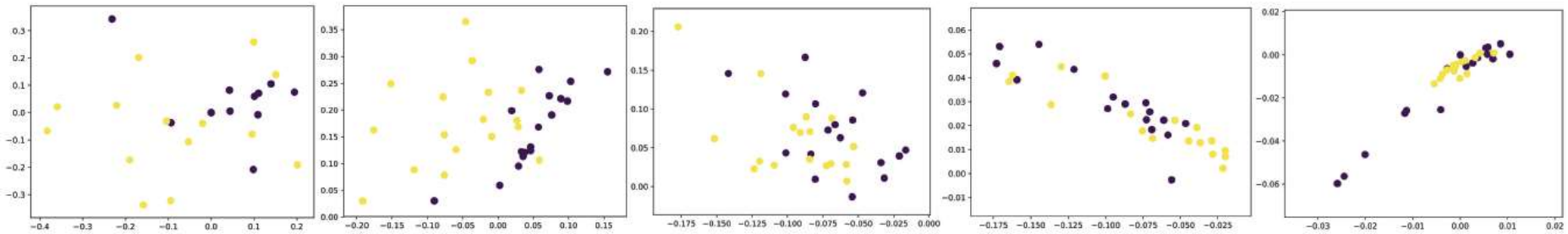


[Kipf_Welling17GCN]

Oversmoothing Problem [Li18Oversmoothing]

- Latent node vectors get closer to each other as the GCN layers go deeper (higher)
- Difficult to distinguish nodes in deeper GCNs

Latent node vectors of ``Karate club'' social network [Li18Oversmoothing]



(a) 1-layer (b) 2-layer (c) 3-layer (d) 4-layer (e) 5-layer

Oversmoothing is theoretically proven

- Deeper GCN converges to a solution where connected nodes will have similar latent vectors [Li18Oversmoothing, NT_Maehara19revisit]

- Such convergence in GCN proceeds very quickly (exponential to the depth), regardless of the initial node vectors [Oono_Suzuki20Exponential]
  - Similar conclusion also holds for a generic GNN

A good workaround [Zhou20Deep, Chen20SimpleDeep, Li20DeeperGCN]

- Combining a proper normalizer and a residual term
  - Normalizing the latent node vectors keep distant from each other [Zhao_Akoglu20PairNorm]
  - Residual terms keep the norm of loss gradients in a moderate scale [Kipf_Welling17GCN]

Residual term:
Add the current layer ‘’as it is’’

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) + H^{(l)}.$$

Not a workaround for “deeper layers, stronger GNNs” (like image recognition)

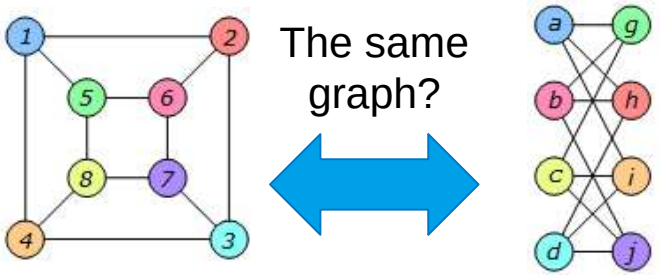
The theoretical limit of representation powers of GNNs

- Surprisingly, the limitation of GNNs is already known in the problem of “graph isomorphism” test

- The theory does NOT directly give limitations of GNN’s power for other tasks (i.e. node classification), but it is loosely related [Chen19RGNN]

Graph isomorphism problem (for theory of GNNs)

- Classify whether two given graphs have an edge-preserving bijective map of node sets
  - the same topology in terms of the edges?

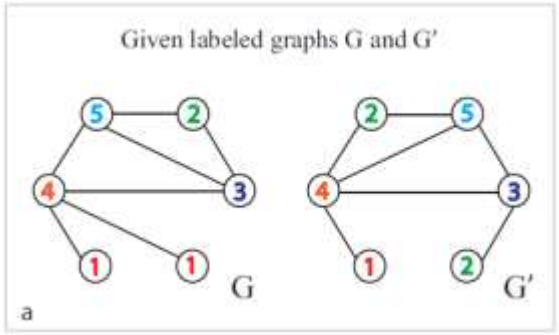


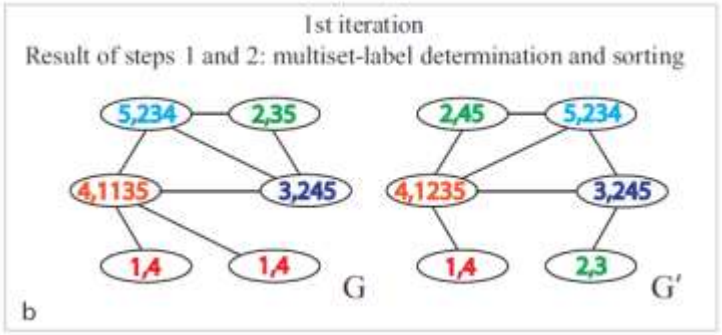
[https://ja.wikipedia.org/wiki/グラフ同型]

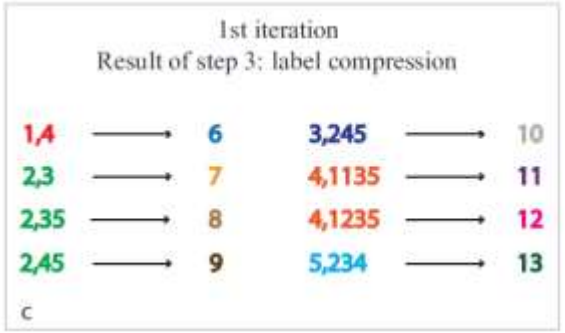
Weisfeiler-Lehman (WL) test algorithm [Weisfeiler_Lehman68WL]

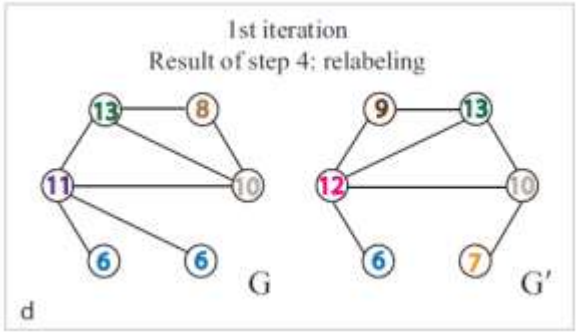
- A popular heuristics to test the isomorphism
- Idea: concatenate the neighbour nodes' labels to check the edge topology
  - Used in graph kernels [Shervashidze11WLKernel, Togninalli18WWL] and GNNs [Jin17WLorg, Morris19kGNN]

Weisfeiler-Lehman (WL) test algorithm [Weisfeiler_Lehman68WL]









[Shervashidze11WLKernel]

Upper limit of the GNN's representation power

- In terms of the Graph Isomorphism problem, a generic GNN $\leqq$ WL test algorithm [Xu19GIN, Morris19WL]
  - There could be a case where WL test can decide isomorphism but GNN can not.

Graph Isomorphism Network (GIN) [Xu19GIN]

- Proposed a specific GNN architecture that attain the same graph isomorphism detection power

One non-linear activation function

A layer update of typical GNNs

$$h_v^{(k)} = \mathrm{ReLU}\left(W \cdot \mathrm{MEAN}\left\{h_u^{(k-1)},\ \forall u \in \mathcal{N}(v) \cup \{v\}\right\}\right).$$

A layer update of the proposed GIN

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

Each layer update must be as powerful as MLP

Higher-order WL/GNN

- k-dimensional WL (k-WL) test
  - labels the k-tuple of nodes
  - powerful than the originalWL
  - K-GNN [Morris19kGNN] is as good as k-WL test

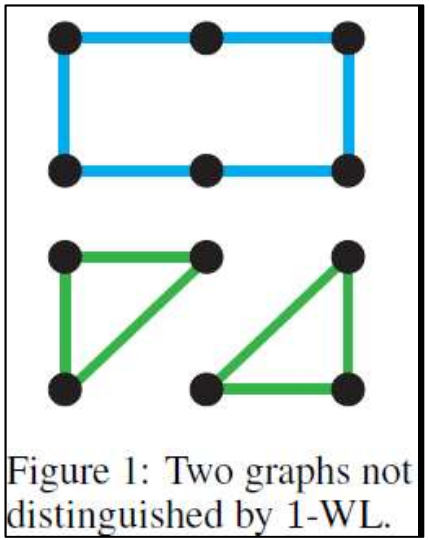


Figure 1: Two graphs not distinguished by 1-WL.

[Maron19NeurIPS]

More powerful GNNs [Sato20Survey]

- K-order graph network
  - consists of all linear functions that are invariant or equivariant to node permutations [Maron19ICLR]
  - as powerful as k-GNN, memory-efficient [Maron19NeurIPS]
- CPNGNN introduces the local node ordering, and is strictly powerful than GIN [Sato19CPNGNN]

Conclusion and Materials

Take home message (Revisited)

- Graph Neural Networks (GNNs): Neural Networks (NNs) to compute nodes' representation of graph-structured data
- Practical applications in industry, hard competitions in academia
- Model: The fundamental is approximated Graph Convolution
- Applications: applicable in several tasks in different domains

Surveys and documents for studying GNN

- Survey papers on GNN
  - [Wu19survey] [Zhou18survey]
  - For experts: [Battaglia18survey]

- Tutorials, slideshare, blogs
  - English: [Kipf18talk], [Ma20AAAI]
  - 日本語: [Honda19GNN]

The most famous dataset resources

- Prof. Leskovec (stanford)
  - Stanford Large Network Dataset collections
    http://snap.stanford.edu/data/index.html

- UC Santa Cruz LINQS group
  - https://linqs.soe.ucsc.edu/data
  - train/valid/test split:
    https://github.com/kimiyoung/planetoid

Finally: GNN for your use

- Say good-bye for “D = {vector x_i, label y_i}” framework with GNNs
  - Applicable for generic structured data = graph
  - The standard GCN is strong enough

- Many diverse application fields
  - chemo, pharmacy, materials, computer vision, social networks, …

Dataset for specific domains

- MoleculeNet [Wu18MoleculeNet]
  - Molecular graph datasets from several chemical field. TensorFlow implmenetaions attached
- Scene Graphs
  - Visual Genome[Krishna17VG]: 108K images
- Pointcloud
  - S3DIS [Armeni16S3DIS]: inside office

Programing Libraries

- **Recommended:** PyTorch Geometric https://github.com/rusty1s/pytorch_geometric
- Deep Graph Library https://www.dgl.ai/

- Chainer Chemistry https://github.com/pfnet-research/chainer-chemistry
  - Tailored for molecular graphs, but also applicable for other domains

References A-G

[Anderson19Cormorant] Anderson+, “Comorant: Covariant Molecular Neural Networks”, NeurIPS, 2019.

[Armeni16S3DIS] Armeni+, “3D Semantics Parsing of Large-scale Indoor Spaces”, CVPR, 2016.

[Bahdanau15Attention] Bahdanau+, “Neural Machine Translation by Jointly Learning to Align and Translate”, ICLR, 2015.

[Battaglia18survey] Battaglia+, “Relational Inductive Biases, Deep Learning, and Graph Networks”, arXiv: 1806.01261v3 [cs.LG], 2018.

[Bruna14Spectral] Bruna+, “Spectral Networks and Locally Connected Networks on Graphs”, ICLR, 2014.

[Chem19RGNN] Chen+, “On the Equivalence between Graph Isofmorphism Testing and Function Approximaion with GNNs”, NeurIPS 2019.

[Chen20SimpleDeep] Chen+, “Simple and Deep Graph Convolutional Networks”, ICML 2020.

[DeCao19QA2] De Cao+, “Question Answering by Reasoning Across Documents with Graph Convolutional Networks“, ICML Workshop, 2019.

[Defferrard16ChebNet] Defferrard+, “Convoluional Neural Networks on Graphs with Fast Localized Spectral Filtering”, NIPS, 2016.

[Fout17Interface] Fout+, “Protein Interface Prediction using Graph Convolutional Networks”, NIPS, 2017.

[Gilmer17MPNN] Gilmer+, “Neural Message Passing for Quantum Chemistry”, ICML, 2017.

References H-K

[Hamilton17GraphSAGE] Hamilton+, “Inductive Representation Learning on Large Graphs”, NIPS 2017.

[Honda19GNN] Honda, “GNNまとめ(1-3)”, 2019. https://qiita.com/shionhonda/items/d27b8f13f7e9232a4ae5 https://qiita.com/shionhonda/items/0d747b00fe6ddaff26e2

https://qiita.com/shionhonda/items/e11a9cf4699878723844

[Hu20RandLaNet] Hu+, “RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds”, CVPR 2020.

[Jin17WLorg] Jin+, “Predicting Organic Reaction Outcomes with Weisfeiler-Lehman Network”, NIPS 2017.

[Jin18JTVAE] Jin+, “Junction Tree Variational Autoencoder for Molecular Graph Generation”, ICML 2018.

[Kipf18talk] Kipf, “Structured Deep Models: Deep learning on graphs and beyond”, 2018. http://tkipf.github.io/misc/SlidesCambridge.pdf

[Kipf_Welling17GCN] Kipf and Welling, “Semi-supervised Classification with Graph Convolutional Networks”, ICLR 2017.

[Klicepra20DimeNet] Klicepra+, “Directional Message Passing for Molecular Graphs”, ICLR 2020.

[Krizhevsky12ConvNet] Krizhevsky+, “ImageNet Classification with Deep Convolutional Neural Networks”, NIPS 2012.

[Krishna17VG] Krishna+, “Visual genome: Connecting language and vision using crowdsourced dense image annotations”, ICCV, 2017.

References L

[Landrieu_Boussaha19PCS] Landrieu and Boussaha, “Point Cloud Oversegmentation with Graph-Structured Deep Metric Learning”, CVPR, 2019.

[Laugier18CGCNN] Laugier+, “Predicting thermoelectric properties from crystal graphs and material descriptors - first application for functional materials”, NeurIPS Workshop, 2018.

[Li16GGNN] Li+, “Gated Graph Sequence Neural Networks”, ICLR, 2016.

[Li18Oversmoothing] Li+, “Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning”, arXiv: 1801.07606 [cs.LG], 2018.

[Li19DeepGCNs] Li+, “DeepGCNs: can GCNs go as deep as CNNs?”, ICCV 2019.

[Li20DeperGCN] Li+, “DeeperGCN: ALL You Need to Train Deeper GCNs”, arXiv: 2006.07739, 2020

[Lu16VRD] Lu+,”Visual Relationship Detection with Language Priors”, ECCV, 2016.

[Liu18CGVAE] Liu+, “Constrained Graph Variational Autoencoders for Molecule Design”, NeurIPS 2018.

References M-P

[Ma20tutorial] Ma+, “Graph Neural Networks: Models and Applications”, AAAI, 2020. http://cse.msu.edu/~mayao4/tutorials/aaai2020/

[Madhawa19GNVP] Madhawa+, “GraphNVP: An Invertible Flow Model for Generating Molecular Graphs”, arXiv: 1905.11600, 2019.

[Marrn19NeurIPS] Marron+, “Provably Powerful Graph Networks”, NeurIPS, 2019.

[Maron19ICLR] Marron+, “Invariant and Equivariant Graph Networks”, ICLR 2019.

[Morris19kGNN] Morris+, “Weisfeiler and Lehman Go Neural: Higher-order Graph Neural Networks”, AAAI, 2019.

[Ngueyn_Maehara20Homomorph] Nguyen and Maehara, “Graph Homomorphism Network”, ICML, 2020.

[NT_Maehara19revisit] NT and Maehara, “Revisiting Graph Neural Networks: All We Have is Low-pass Filters”, arXiv: 1905.09550, 2019.

[Oono_Suzuki20Exponential] Oono and Suzuki, “Graph Neural Networks Exponentially Lose Expressive Power for Node Classification”, ICLR 2020.

[Pei20GeomGCN] Pei+, “Geom-GCN: Geometric Graph Convolutional Networks”, ICLR, 2020.

References Q-V

[Qi19AttentiveRN] Qi+, “Attentive Relataional Networks for Mapping Images to Scene Graphs”, CVPR, 2019.

[Sanchez-Gonzalez20GNS] Sanchez-Gonzalez+, “Learning to Simulate Complex Physics with Graph Networks”, arXiv: 2002.09405. 2020.

[Sanyal20PGCN] Sanyal+, “Protein GCN: Protein model quality assessment using graph convolutional networks”, bioRxiv, 2020.

[Sato19CPNGNN] Sato+, “Approximation Ratios of Graph Neural Networks for Combinatorial Problems”, NeurIPS, 2019.

[Sato20Survey] Sato, “A Survey on the Expressive Power of Graph Neural Networks”, arXiv: 2003.04078.

[Scarselli08GNN] Scarselli+, “The Graph Neural Network Model”, IEEE Trans. Neural Networks, 20(1), pp. 61-80, 2008

[Schlichtkrull18RGCN] Modeling Relational Data with Graph Convolutional Networks, ESWC 2018.

[Shervashidze11WLKernel: Shervashidez+, “Weisfeiler-Lehman Graph Kernels”, JMLR, 12, pp.2539-2661, 2011.

[Shuman13Graph] Shuman+, “The emerging field of signal processing on graphs: extending high-dimensional data analysis to networks and other irregular domains”, IEEE Signal Processing Magazine, 30(3), pp.83-98, 2013

[Togninalli18WWL] Togninalli+, “Wasserstein Weisfeiler-Lehman Graph Kernels”, NeurIPS 2018.

[Veličković18GAT] Veličković+, “Graph Attention Networks”, ICLR 2018.

References W-Z

[Wang18NLNN] Wang+, “Non-local Neural Networks”, CVPR 2018.

[Weisfeiler_Lehman68WL] Weisfeiler and Lehman, “A Reduction of a Graph to a Canonical Form and an Algebra Arising during this Reduction”, Nauchno-Technicheskaya Informatsia, Ser.2(9), pp.12-16, 1968.

[Wu18MoleculeNet] Wu+, “MoleculeNet: a benchmark for molecular machine learning”, Chemical Science, 9(513), 2018.

[Wu19SGC] Wu+, “Simplifying Graph Convolutional Networks”, in Proc. ICML, 2019.

[Wu19survey] Wu+, “A Comprehensive Survey on Graph Neural Networks”, arXiv:1901.00596v1 [cs.LG], 2019.

[Xu19GIN] Xu+, “How powerful are Graph Neural Networks?”, in Proc. ICLR, 2019.

[Yang18GRCNN] Yang+, “Graph R-CNN for Scene Graph Generation”, ECCV, 2018.

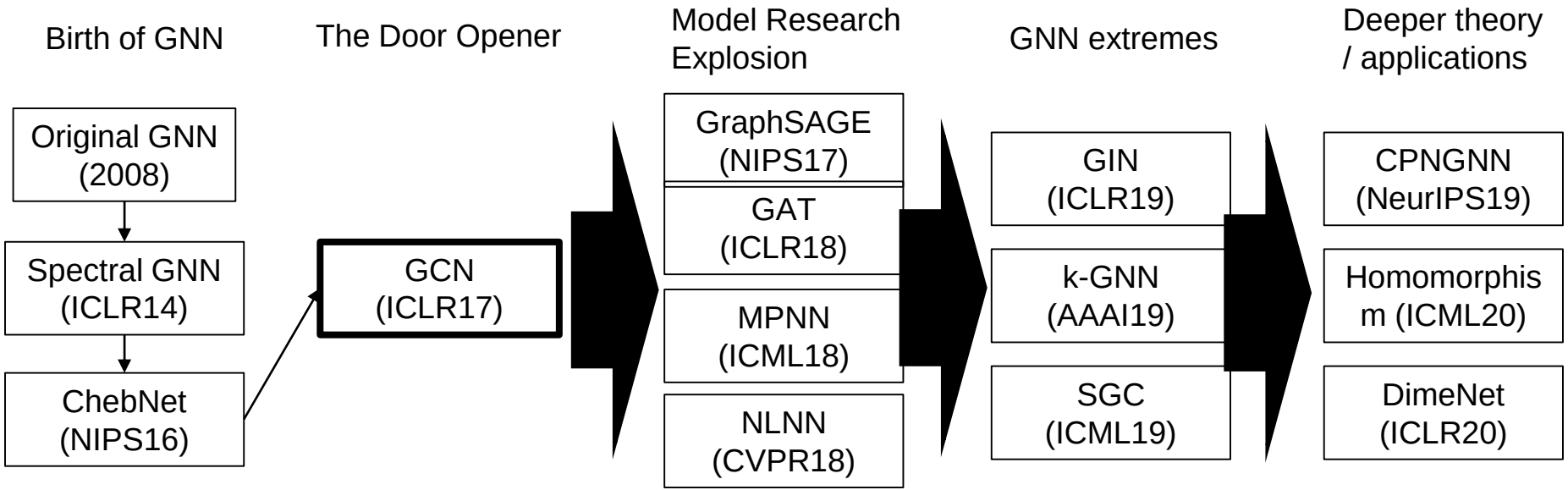
[You18GCPN] You+, “Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation”, NeurIPS 2018.

[Zhao_Akoglu20PairNorm] Zhao and Akoglu, “PairNorm: Tackling Oversmoothing in GNNs”, ICLR, 2020.

[Zhou20Deep] Zhou+, “Effective Training Strategies for Deep Graph Neural Networks”, arXiv: 2006.07107, 2020.

[Zhou18survey] Zhou+, “Graph Neural Networks: A Review of Methods and Applications”, arXiv: 1812.08434v2 [cs.LG], 2018.

GNN research history

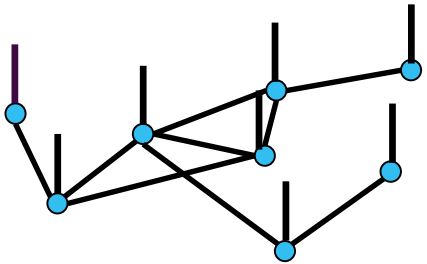


Original GNN [Scartesllli08GNN] Spectral GNN [Bruna14Spectral] ChebNet[Defferrard16Cheb]
GGNN[Li16GGNN] GCN [Kipf_Welling17GCN] GraphSAGE [Hamilton17GraphSAGE] GAT [Veličković18GAT]
MPNN [Gilmer17MPNN] NLNN [Wang18NLNN] GIN [Xu19GIN] K-GNN [Morris19kGNN] SGC [Wu19SGC]
CPNGNN [Sato19CPNGNN] Graph Homomorphism Convolution [Ngueyn_Maehara20Homomorph] DimeNet [Klcepra20DimeNet]

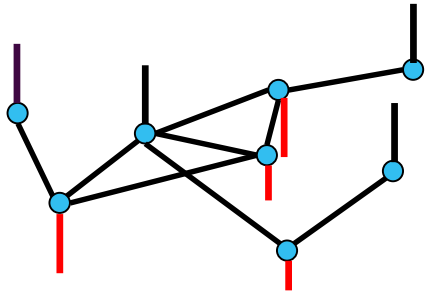
Def: Graph signals

- Suppose the 1D node attribute case: N-array

$$x = (x_1, x_2, \ldots, x_N)^T \in \mathbb{R}^N$$



Low frequency graph signal



High frequency graph signal

[Ma20tutorial]