

Graph Machine Learning

Lecture 8 : Graph Convolutional Networks Spectral and Spatial Techniques

Sept 30 2026

Xavier Bresson

<https://x.com/xbresson>

Department of Computer Science
National University of Singapore (NUS)



Course lectures

- Introduction to Graph Machine Learning
- Part 1: GML without feature learning (before 2014)
 - Introduction to Graph Science
 - Graph Analysis Techniques without Feature Learning
 - Graph clustering
 - Graph SVM
 - Recommendation on graphs
 - Graph-based visualization
- Part 2 : GML with shallow feature learning (2014-2016)
 - Shallow graph learning
- Part 3 : GML with deep feature learning, a.k.a. GNNs (after 2016)
 - • Graph Convolutional Networks (spectral and spatial)
 - Graph Transformer
 - Graph generation & molecular science
 - Material design
 - Integrating GNNs and LLMs
 - Deep Learning for Combinatorial optimization

Outline

- ConvNets on Euclidean Grids
 - Architecture
 - From grids to graphs
 - Convolution on grids
- Spectral Graph ConvNets
 - Spectral convolution
 - Spectral graph networks
- Spatial Graph ConvNets
 - Template matching
 - Isotropic graph networks
 - Anisotropic graph networks
- Conclusion

Outline

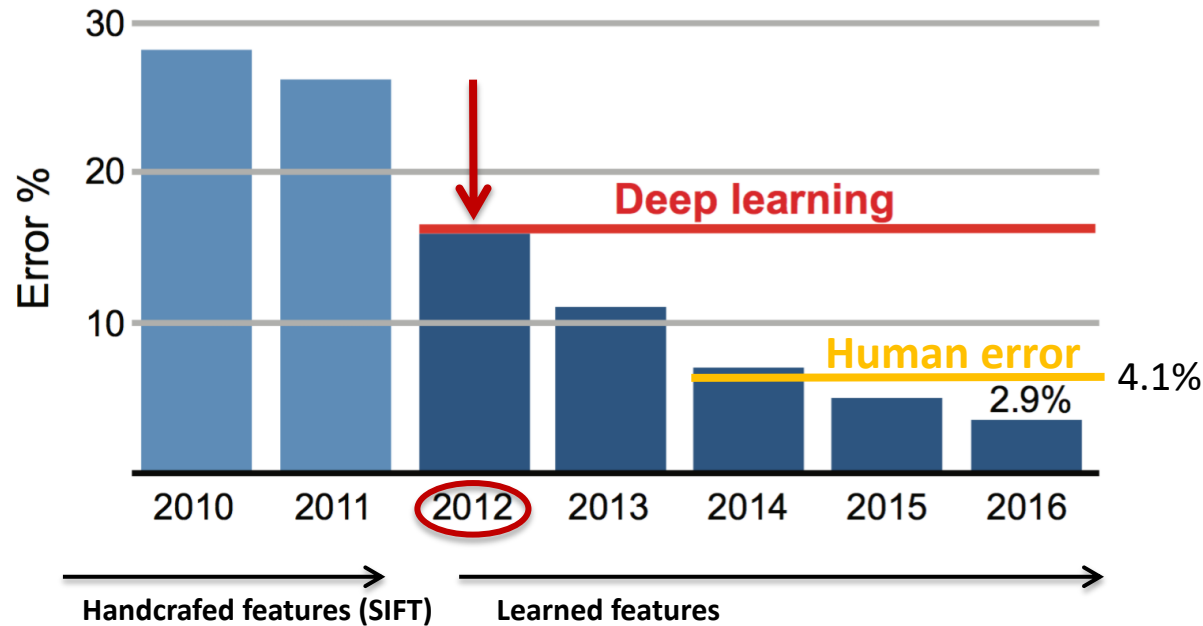
- **ConvNets on Euclidean Grids**
 - **Architecture**
 - From grids to graphs
 - Convolution on grids
- Spectral Graph ConvNets
 - Spectral convolution
 - Spectral graph networks
- Spatial Graph ConvNets
 - Template matching
 - Isotropic graph networks
 - Anisotropic graph networks
- Conclusion

The ConvNet revolution

- CNN / ConvNet transformed computer vision.

LeCun-Bengio et-al 1998^[1] : Foundations of modern CNNs.

Krizhevsky-Sutskever-Hinton^[2], 2012 : The ImageNet breakthrough.

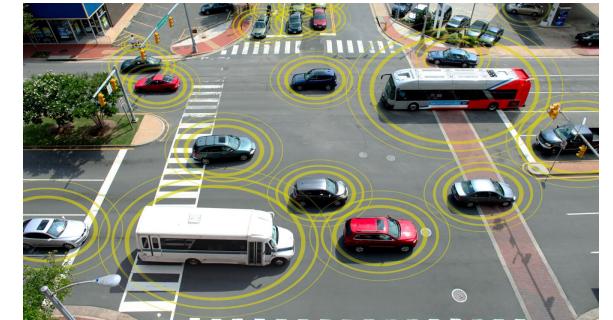


- ConvNet was also successfully applied to speech^[3] and NLP^[4].

[1] LeCun et-al, Gradient-Based Learning Applied to Document Recognition, 1998
[2] Krizhevsky et-al, ImageNet Classification with Deep Convolutional Neural Networks, 2012
[3] Abdel-Hamid et-al, Convolutional Neural Networks for Speech Recognition, 2014
[4] Kim, Convolutional Neural Networks for Sentence Classification, 2014

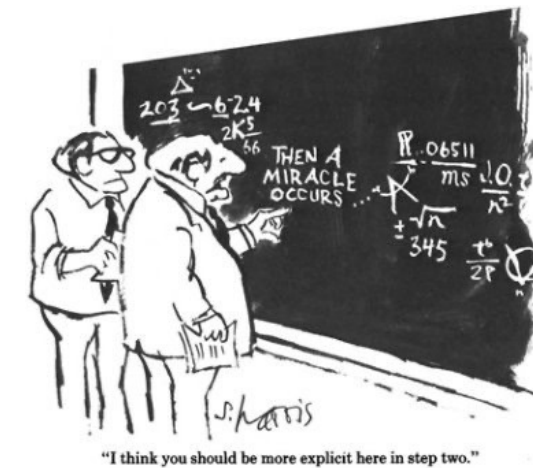
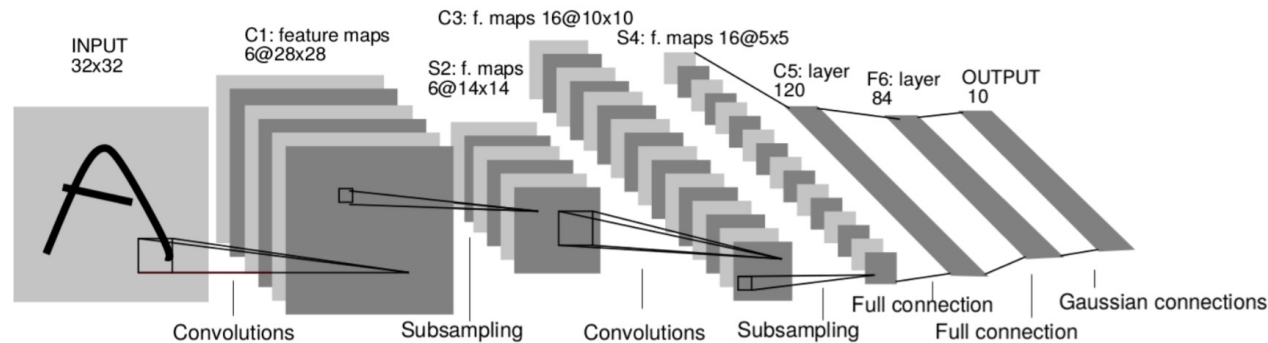


IMAGENET



Curse of dimensionality

- Images live in extremely high-dimensional spaces.
- Curse of dimensionality^[1] :
 $\text{dim}(\text{image}) = 1024 \times 1024 \approx 10^6$
For $N = 10$ samples/dim $\Rightarrow 10^{1,000,000}$ points/images !
- ConvNet mitigates the curse of dimensionality^[2] by exploiting local patterns, hierarchical structure and shared weights.

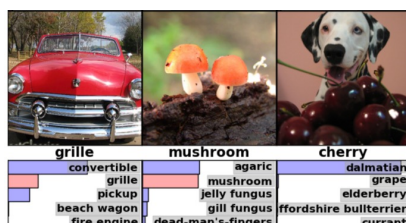
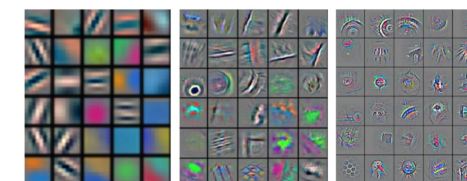
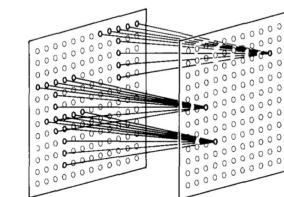


[1] Bellman et-al, Dynamic Programming, 1957

[2] Poggio et-al, Why and When Can Deep-but Not Shallow-networks Avoid the Curse of Dimensionality: A Review, 2017

Compositional structure

- Main assumption / inductive bias :
 - Data (image, video, speech) is compositional, i.e. formed of patterns that are :
 - Local : Nearby variables form meaningful patterns (Hubel-Wiesel^[1]).
 - Stationary : Translation-shared patterns^[2].
 - Hierarchical / multi-scale : Simple patterns compose into more complex ones^[2].
- ConvNets exploit this structure :
 - They learn compositional features across layers, and train end-to-end a classifier, regressor, recommender, etc, for downstream tasks.



Computer Vision

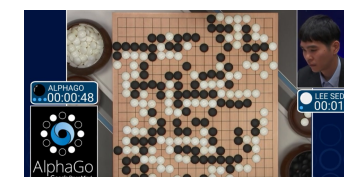
```

1 // Duplicate LSN field information. The LSN rule is known.
2 // Rule initialized.
3 static inline int audit_rule_lsn(struct audit_field *df,
4                               struct audit_rule *r)
5 {
6     int ret = 0;
7     char *lsn_str;
8     // Own copy of LSN string.
9     lsn_str = strdup(df->lsn_str, GFP_KERNEL);
10    if (unlikely(!lsn_str))
11        return -ENOMEM;
12    df->lsn_str = lsn_str;
13    // Security audit rule init (df->type, df->exp, df->lsn_str)
14    ret = security_audit_rule_init(df->type, df->exp, df->lsn_str);
15    if (ret)
16        return ret;
17    // Keep currently loaded rules around in case later
18    // become valid after a policy reload.
19    if (ret == -EINVAL)
20        return 0;
21    return ret;
22 }
  
```

NLP



Speech



Game of Go

[1] Hubel-Wiesel, Receptive Fields, Binocular Interaction and Functional Architecture in the Cat's Visual Cortex, 1962

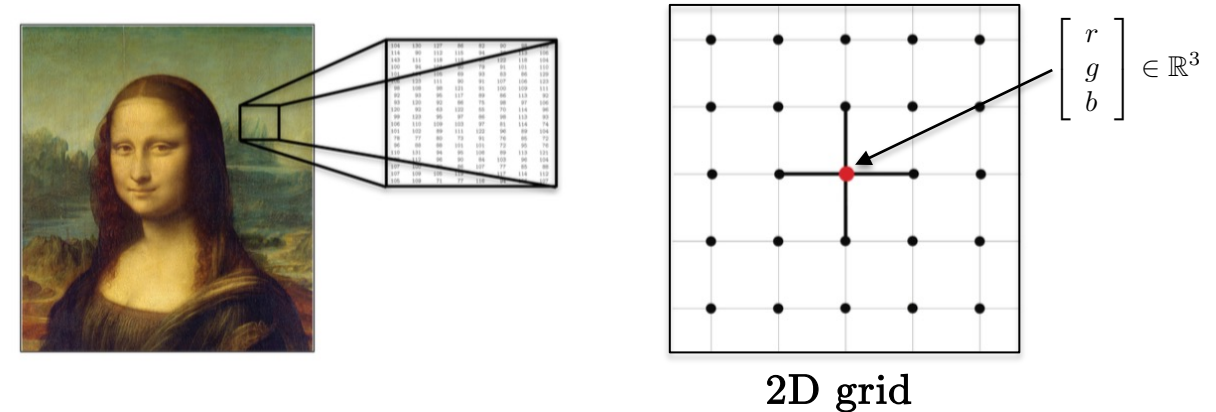
[2] LeCun-Bengio et-al, Gradient-Based Learning Applied to Document Recognition, 1998

Outline

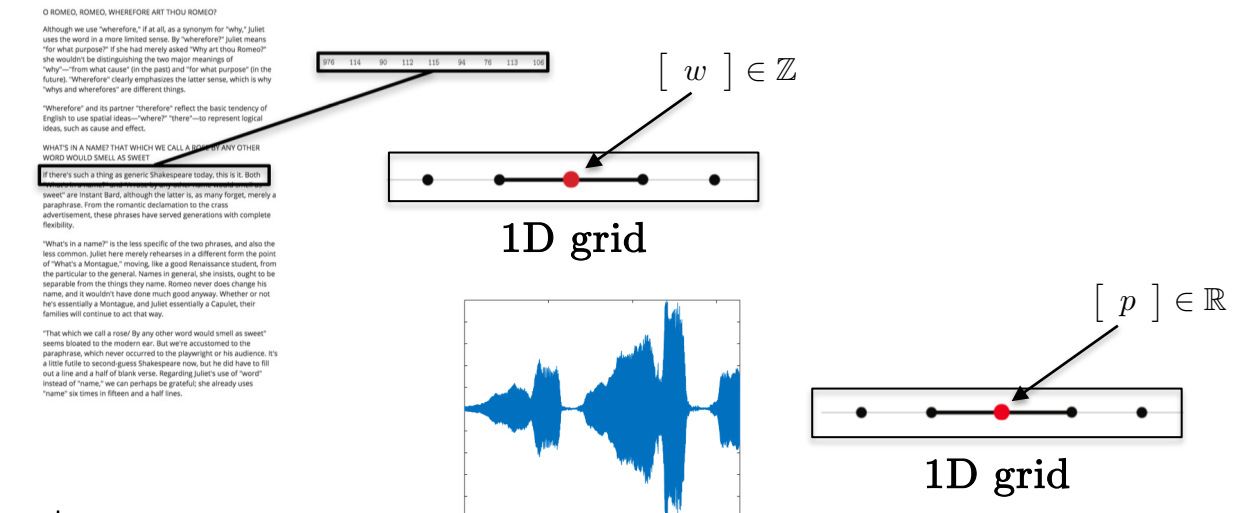
- ConvNets on Euclidean Grids
 - Architecture
 - From grids to graphs
 - Convolution on grids
- Spectral Graph ConvNets
 - Spectral convolution
 - Spectral graph networks
- Spatial Graph ConvNets
 - Template matching
 - Isotropic graph networks
 - Anisotropic graph networks
- Conclusion

Data on regular grids / sequences

- Images, volumes, and videos are signals sampled on
2D, 3D, 2D+time Euclidean grids



- Text and speech can be represented as sequences on
1D regular grids



- These domains have strong regular spatial structures.
 - ConvNet operations, i.e. convolution and pooling, are therefore well defined and efficient^[1].

[1] LeCun-Bengio et-al, Gradient-Based Learning Applied to Document Recognition, 1998

Data on graphs



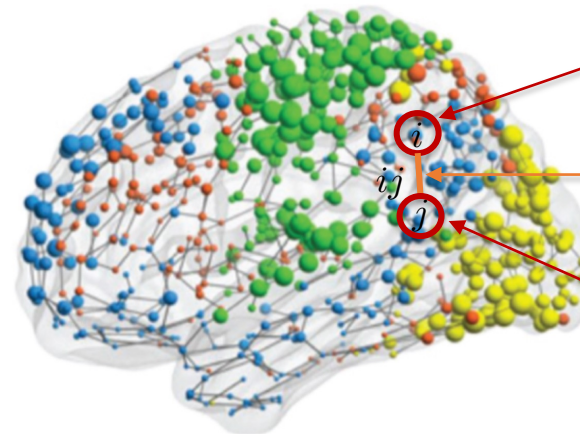
Social networks
(Advertisement/
recommendation)

$$\text{User}_i \begin{bmatrix} \text{messages} \\ \text{images} \\ \text{videos} \end{bmatrix}_i \in \mathbb{R}^d$$

$$\text{User connection}_{ij} \quad A_{ij} = \begin{cases} 1 & \text{if } ij \text{ friends} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{User}_j \begin{bmatrix} \text{messages} \\ \text{images} \\ \text{videos} \end{bmatrix}_j \in \mathbb{R}^d$$

Brain
connectivity
(sMRI)



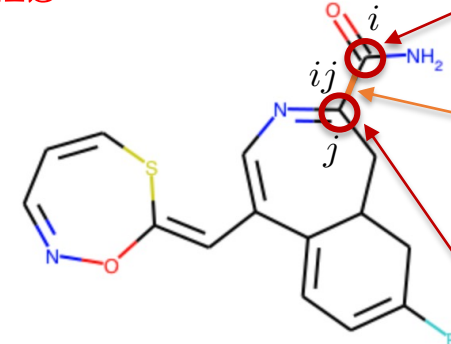
Brain analysis
(Neuroscience/neuro-diseases)

$$\text{ROI}_i \begin{bmatrix} a_1 \\ \vdots \\ a_T \end{bmatrix}_i \in \mathbb{R}^T$$

$$\text{Cerebral connection}_{ij} \quad A_{ij} \in \mathbb{R}_+$$

$$\text{ROI}_j \begin{bmatrix} a_1 \\ \vdots \\ a_T \end{bmatrix}_j \in \mathbb{R}^T$$

Functional
activations (fMRI)



Quantum Chemistry
(novel molecules for drugs
and materials)

$$\text{Atom}_i \begin{bmatrix} \text{type} \\ \text{coordinates} \\ \text{charge} \end{bmatrix}_i \in \mathbb{R}^{d_v}$$

$$\text{Bond}_{ij} \quad A_{ij} = \begin{cases} 1 & \text{if } ij \text{ bond} \\ 0 & \text{otherwise} \end{cases}$$

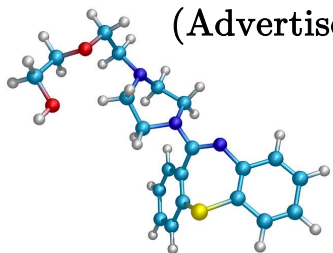
$$\begin{bmatrix} \text{type} \\ \text{energy} \end{bmatrix}_{ij} \in \mathbb{R}^{d_e}$$

$$\text{Atom}_j \begin{bmatrix} \text{type} \\ \text{coordinates} \\ \text{charge} \end{bmatrix}_j \in \mathbb{R}^{d_v}$$

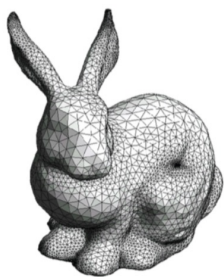
Graph-structured data across domains



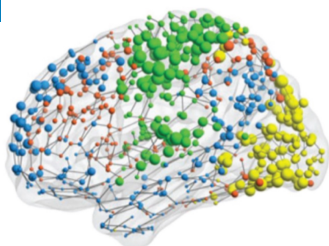
Social networks
(Advertisement)



Drug/Material
molecules
(Chemistry)



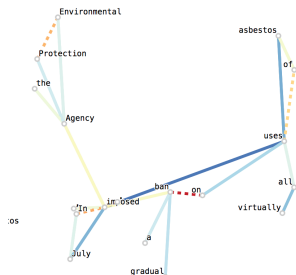
3D Meshes
(Computer
Graphics)



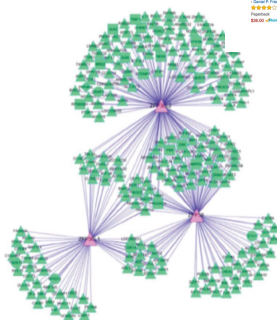
Brain
connectivity
(Neuroscience)



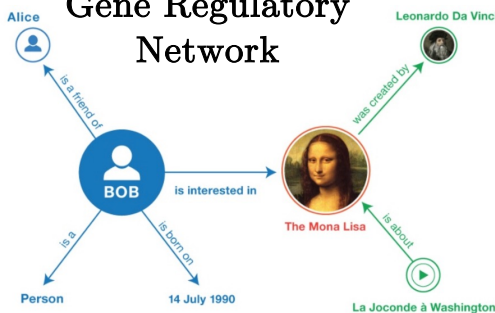
Transportation
networks



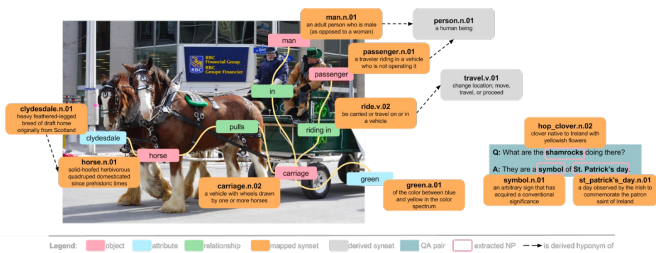
Words relationships
(NLP)



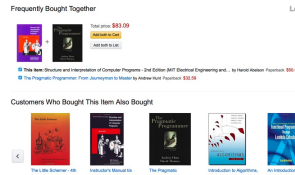
Gene Regulatory
Network



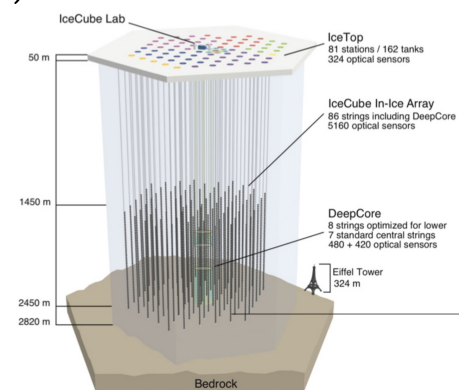
Knowledge graphs



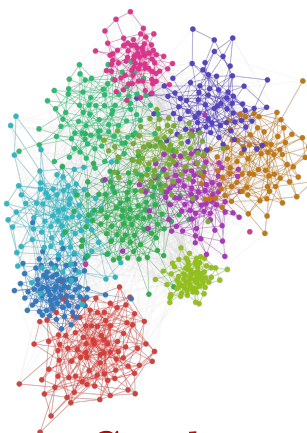
Scene understanding



Recommender
systems (Amazon,
Netflix)



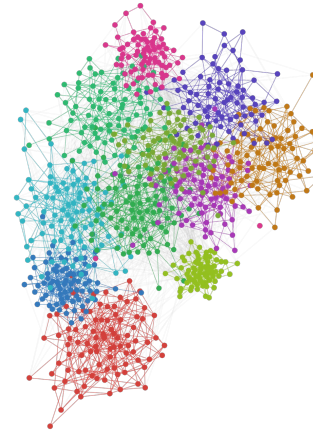
Neutrino
detection (High-
energy Physics)



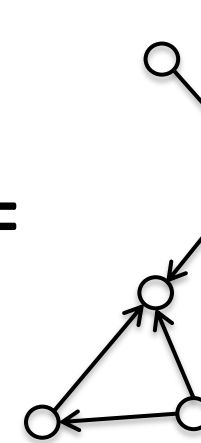
Graph-
structured
data

Graph-structured data

- A graph G is defined by :
 - Vertices $V = \{1, \dots, n\}$
 - Edges $E \subseteq V \times V$
 - Adjacency matrix A encoding edge strength



=



$$h_j = \begin{bmatrix} \vdots \end{bmatrix} \in \mathbb{R}^{d_v}$$
$$e_{ij} = \begin{bmatrix} \vdots \end{bmatrix} \in \mathbb{R}^{d_e}$$
$$h_i = \begin{bmatrix} \vdots \end{bmatrix} \in \mathbb{R}^{d_v}$$

- Graph features :
 - Node features : h_i, h_j (atom type)
 - Edge features : e_{ij} (bond type)
 - Graph features : g (molecule energy)

$$\underbrace{\mathcal{G} = (V, E, A)}$$

$$g = \begin{bmatrix} \vdots \end{bmatrix} \in \mathbb{R}^{d_g}$$

Outline

- **ConvNets on Euclidean Grids**
 - Architecture
 - From grids to graphs
 - **Convolution on grids**
- Spectral Graph ConvNets
 - Spectral convolution
 - Spectral graph networks
- Spatial Graph ConvNets
 - Template matching
 - Isotropic graph networks
 - Anisotropic graph networks
- Conclusion

Convolution on regular grids

- Convolutional layer^[1] on a regular grid :

$$h^{\ell+1} = w^{\ell} * h^{\ell}, \text{ where } * \text{ is the convolution operator}$$

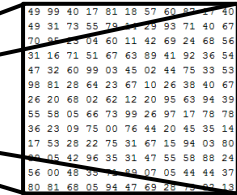
$$\text{and } h^{\ell+1} \in \mathbb{R}^{n_1 \times n_2 \times d_{out}}, w^{\ell} \in \mathbb{R}^{3 \times 3 \times d_{in}}, h^{\ell} \in \mathbb{R}^{n_1 \times n_2 \times d_{in}}.$$

For this lecture, we choose $d = d_{in} = d_{out}$.



h^{ℓ}

Image/ feature
at layer ℓ



49	99	40	17	81	18	57	60	87	40
49	31	73	55	75	77	93	71	40	67
70	95	77	04	60	11	42	69	24	68
51	16	71	51	67	63	89	41	92	36
47	32	60	99	03	45	02	44	75	33
98	81	28	64	23	67	10	26	38	40
26	20	68	02	62	12	20	95	63	94
55	58	05	66	73	99	26	97	17	78
36	23	09	75	00	76	44	20	45	35
17	53	28	22	75	31	67	15	94	03
77	05	42	96	35	31	47	55	58	88
56	00	48	35	71	44	57	05	44	44
89	51	68	05	94	47	69	26	77	13

*



=



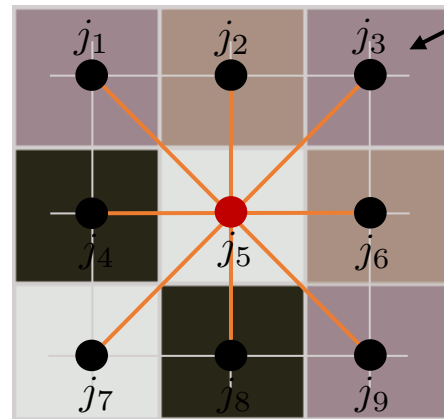
$h^{\ell+1}$

Activation at
layer $\ell+1$

[1] Folland, Real Analysis: Modern Techniques and Their Applications, 1999

Template matching

- How to define convolution ?
 - Definition #1: Convolution as local template matching.
 - $O(n \cdot 3^2 \cdot d) = O(n)$ time complexity for fixed-size local kernels $\mathcal{N}_i$ and highly parallelizable on GPUs.



Node j_3 always denotes the top-right corner of the pattern.

$$\begin{bmatrix} w_{j_3}^\ell \end{bmatrix} \in \mathbb{R}^d$$

Template features at j_3

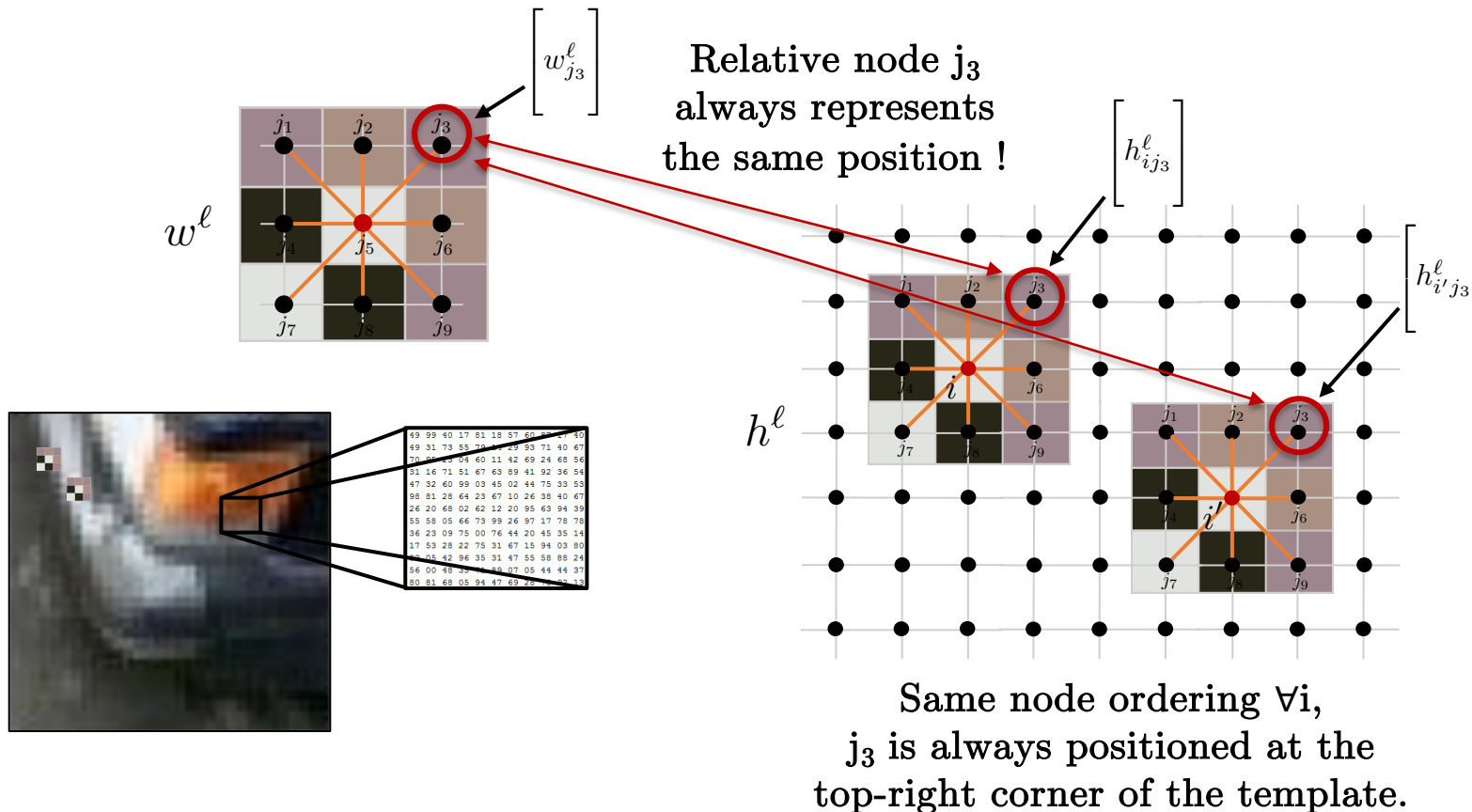
The local node neighborhood of template w^ℓ has the same canonical ordering.

$$\begin{aligned} h_i^{\ell+1} &= w^\ell * h_i^\ell \in \mathbb{R}^{n_1 \times n_2 \times d}, \\ &\text{with } w^\ell \in \mathbb{R}^{3 \times 3 \times d} \\ &= \sum_{\substack{j \in \Omega \\ j \in \mathcal{N}_i}} \langle w_j^\ell, \underbrace{h_{i-j}^\ell}_{h_{ij}^\ell} \rangle \\ &= \sum_{j \in \mathcal{N}_i} \langle w_j^\ell, h_{ij}^\ell \rangle \\ &= \sum_{j \in \mathcal{N}_i} \left\langle \begin{bmatrix} w_j^\ell \end{bmatrix}, \begin{bmatrix} h_{ij}^\ell \end{bmatrix} \right\rangle \end{aligned}$$

Convolution as shared template matching

- How to define convolution ?
 - Definition #1: Convolution as local template matching.

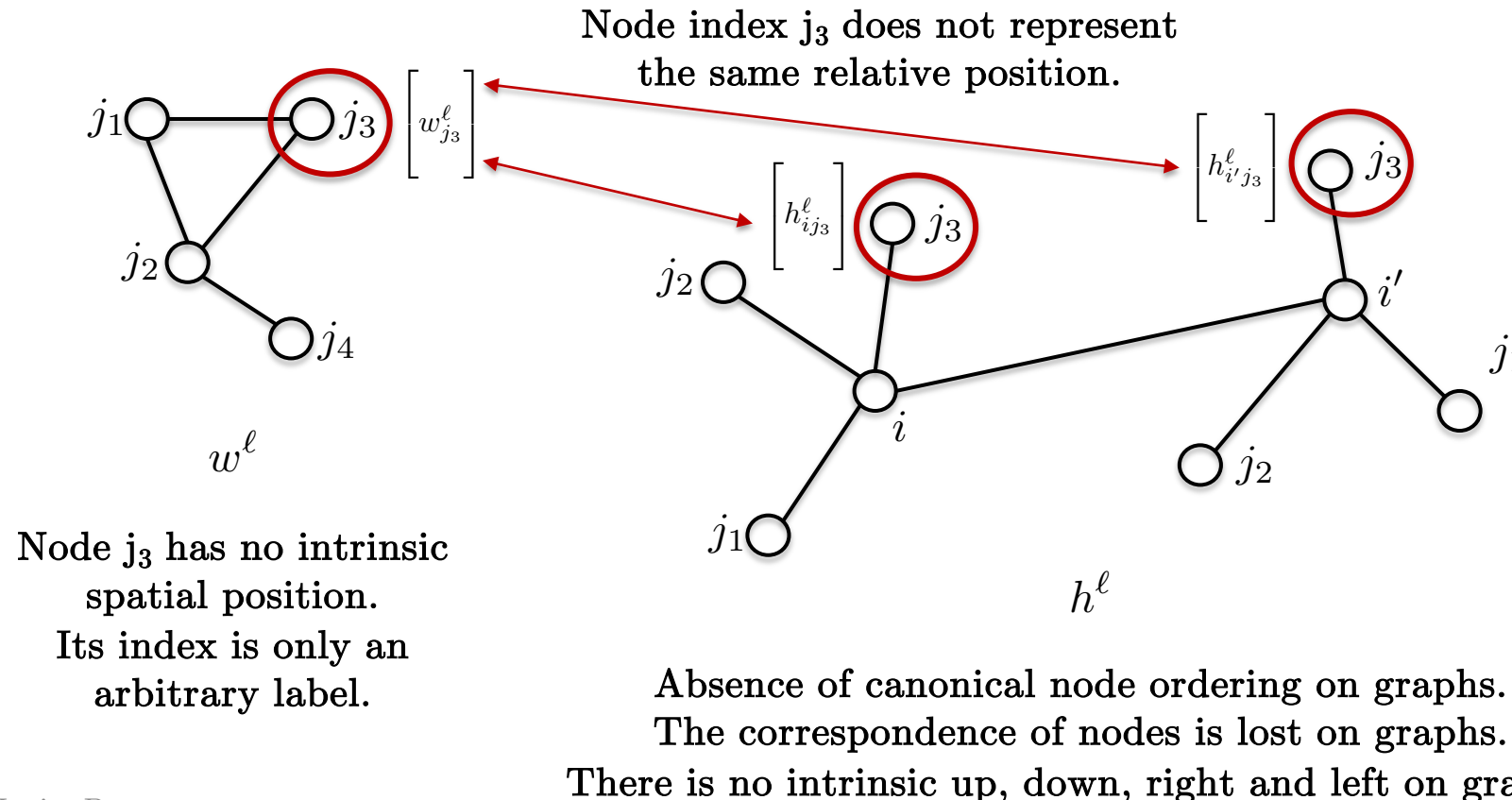
$$\begin{aligned}
 h_i^{\ell+1} &= w^\ell * h_i^\ell \\
 &= \sum_{j \in \mathcal{N}_i} \langle w_j^\ell, h_{ij}^\ell \rangle \\
 &= \sum_{j \in \mathcal{N}_i} \left\langle \begin{bmatrix} w_j^\ell \end{bmatrix}, \begin{bmatrix} h_{ij}^\ell \end{bmatrix} \right\rangle \\
 &\quad \left\langle \begin{bmatrix} w_{j_3}^\ell \end{bmatrix}, \begin{bmatrix} h_{ij_3}^\ell \end{bmatrix} \right\rangle \\
 &\quad \left\langle \begin{bmatrix} w_{j_3}^\ell \end{bmatrix}, \begin{bmatrix} h_{i'j_3}^\ell \end{bmatrix} \right\rangle
 \end{aligned}$$



The same weight $w_{j_3}^\ell$ matches the top-right feature at every image location.

Challenges of convolution on graphs

- Can we extend template matching from grids to graphs ?
 - Main challenges :
 - No canonical ordering of nodes : Then, how to match template features with data features when template positions cannot be matched consistently across nodes ?



$$\left\langle \begin{bmatrix} w_{j_3}^\ell \end{bmatrix}, \begin{bmatrix} h_{i j_3}^\ell \end{bmatrix} \right\rangle$$

$$\left\langle \begin{bmatrix} w_{j_3}^\ell \end{bmatrix}, \begin{bmatrix} h_{i' j_3}^\ell \end{bmatrix} \right\rangle$$

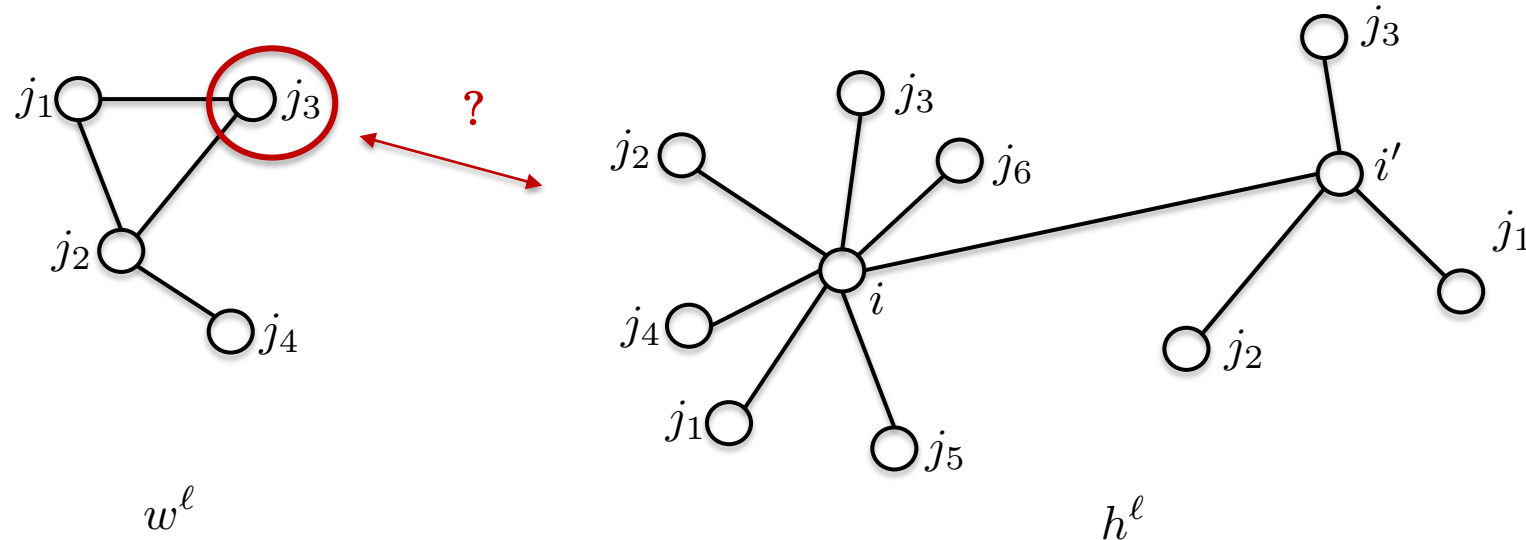
These matching scores have no meaning as they do not compare the same information.

$$h_i^{\ell+1} = w^\ell *_{\mathcal{G}} h_i^\ell \in \mathbb{R}^{n \times d}$$

$$= \sum_{j \in \mathcal{N}_i} \cancel{\langle w_{j_3}^\ell, h_{i j}^\ell \rangle}$$

Challenges of convolution on graphs

- Can we extend template matching from grids to graphs ?
 - Main challenges :
 - No canonical ordering of nodes : Then, how to match template features with data features when template positions cannot be matched consistently across nodes ?
 - Heterogeneous neighborhood : How to deal with different neighborhood sizes ?



Spectral graph convolution

- How to define convolution ?

- Definition #1 : Local template matching

- Definition #2 : Spectral convolution theorem^[1]

- Convolution becomes point-wise multiplication in the Fourier domain :

$$\mathcal{F}(w * h) = \mathcal{F}(w) \odot \mathcal{F}(h) \quad \Rightarrow \quad w * h = \mathcal{F}^{-1}(\mathcal{F}(w) \odot \mathcal{F}(h))$$

- The direct Discrete Fourier Transform (DFT) costs $O(n^2)$, but reduces to $O(n \log n)$ on regular grids with Fast Fourier Transform^[2] (FFT).

- Can we define an analogous spectral convolution on graphs ?

Yes, the convolution theorem remains the same. However,

- How do we define Fourier transform on graphs ?

- How do we compute fast spectral convolutions in $O(n)$ time for compact kernels ?

$$w *_G h \stackrel{?}{=} \mathcal{F}_G^{-1}(\mathcal{F}_G(w) \odot \mathcal{F}_G(h))$$

[1] Fourier, Théorie analytique de la chaleur, 1822

[2] Cooley-Tukey, An algorithm for the machine calculation of complex Fourier series, 1965

Outline

- ConvNets on Euclidean Grids
 - Architecture
 - From grids to graphs
 - Convolution on grids
- Spectral Graph ConvNets
 - Spectral convolution
 - Spectral graph networks
- Spatial Graph ConvNets
 - Template matching
 - Isotropic graph networks
 - Anisotropic graph networks
- Conclusion

Spectral Convolution

- How to perform spectral convolution?
- We apply spectral graph theory^[1] following these steps :
 - Compute the graph Laplacian
 - Fourier functions
 - Fourier transform
 - Apply the convolution theorem

[1] Chung, Spectral graph theory, 1997

Graph Laplacian

- Core operator in spectral graph theory :

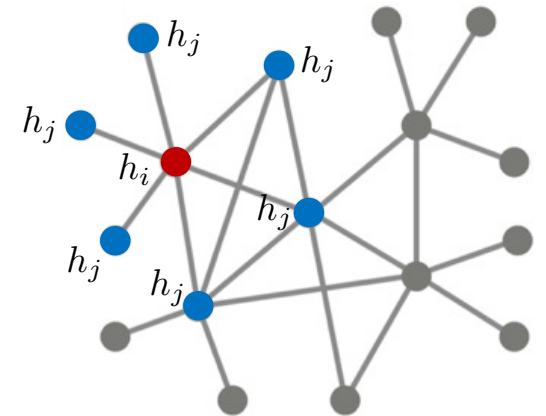
$$\mathcal{G} = (V, E, A) \rightarrow L = I - D^{-1/2} A D^{-1/2} \in \mathbb{R}^{n \times n} \quad \text{Normalized Laplacian}$$

$$\text{where } D = \text{diag}(d_1, \dots, d_n) \in \mathbb{R}^{n \times n}, \quad d_i = \sum_{j \neq i} A_{ij} \quad \text{Degree matrix}$$

- Interpretation of the graph Laplacian :

- Suppose h is a scalar, then value $(Lh)_i$ measures local smoothness / local variation of a signal at node i by computing the difference between local value h_i and its neighborhood average values h_j .
- The value $h^T L h$ measures the global signal smoothness on graph.

$$(Lh)_i = h_i - \sum_{j \in \mathcal{N}_i} \frac{A_{ij}}{\sqrt{d_i d_j}} h_j \in \mathbb{R}^d$$
$$h^T L h = \frac{1}{2} \sum_{i,j} A_{ij} \left(\frac{h_i}{\sqrt{d_i}} - \frac{h_j}{\sqrt{d_j}} \right)^2 \in \mathbb{R}_+$$



Graph Fourier basis

- Eigendecomposition of the normalized graph Laplacian^[1] :

$$L = \Phi \Lambda \Phi^T \in \mathbb{R}^{n \times n}$$

$$\text{where } \Phi = [\phi_1, \dots, \phi_n] = \begin{bmatrix} | & & | \\ \phi_1 & \dots & \phi_n \\ | & & | \end{bmatrix} \in \mathbb{R}^{n \times n},$$

$$\Phi^T \Phi = \mathbf{I}, \quad \langle \phi_k, \phi_{k'} \rangle = \delta_{kk'},$$

$$\text{where } \Lambda = \text{diag}(\lambda_1, \dots, \lambda_n) = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix} \in \mathbb{R}^{n \times n},$$

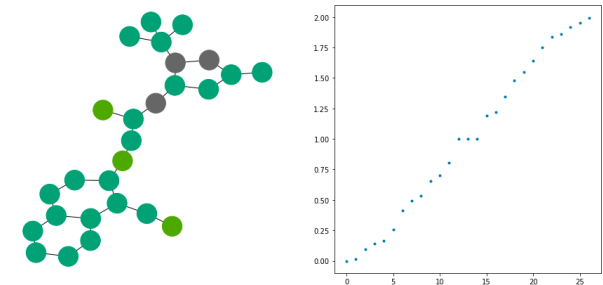
$$0 \leq \lambda_1 \leq \dots \leq \lambda_n = \lambda_{\max} \leq 2,$$

$$\text{and } L\phi_k = \lambda_k \phi_k \in \mathbb{R}^{n \times 1}, \quad k = 1, \dots, n.$$

Laplacian eigenvectors /
graph Fourier functions / basis / modes

Fourier functions form
an orthonormal basis

Laplacian eigenvalues /
graph spectrum / frequencies



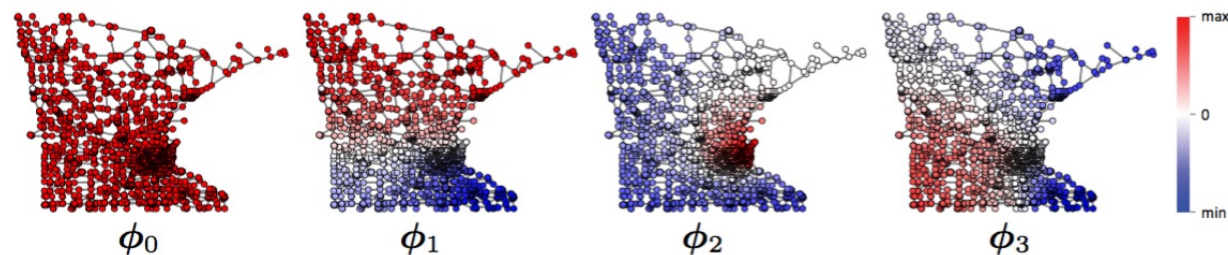
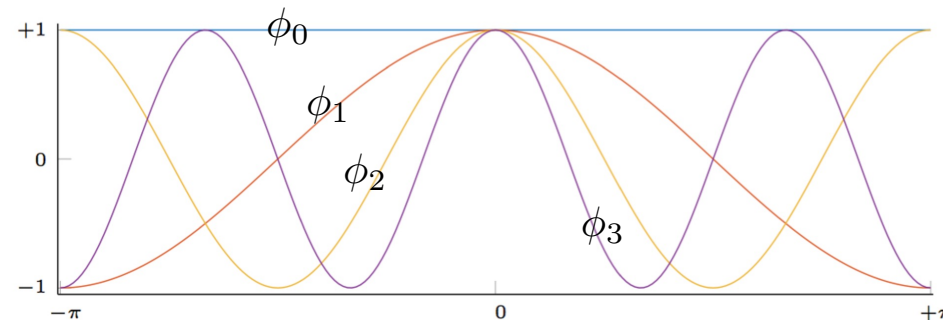
Graph spectrum

Low frequency / smooth
High frequency / oscillatory

[1] Chung, Spectral graph theory, 1997

Fourier functions

- Grid / Euclidean domain : Laplacian eigenfunctions recover the standard Fourier modes, i.e. cosine/sin functions.
- Graph domain : Laplacian eigenvectors define Fourier modes adapted to graph connectivity^[1].



First Laplacian eigenvectors of a graph

Low-frequency modes reveal
smooth structure and communities
Spectral graph clustering^[2]

[1] Hammond-et al, Wavelets on graphs via spectral graph theory, 2011

[2] Von Luxburg, A tutorial on spectral clustering, 2007

Graph Fourier transform

- Fourier transform : Decompose a signal h into the Fourier basis^[1].
- Both the Fourier transform and its inverse are linear operators and can each be implemented in a single line of code, and in parallel.

$$h = \sum_{k=1}^n \langle \phi_k, h \rangle \phi_k = \sum_{k=1}^n \hat{h}_k \phi_k = \Phi \hat{h} \in \mathbb{R}^{n \times 1}, \quad \Phi \in \mathbb{R}^{n \times n}, \quad \hat{h} \in \mathbb{R}^{n \times 1}$$

with

$$\text{Fourier transform :} \quad \hat{h} = \mathcal{F}_G(h) = \Phi^\top h \in \mathbb{R}^{n \times 1}$$

$$\text{Fourier coefficients :} \quad \hat{h}_k = \langle \phi_k, h \rangle \in \mathbb{R}$$

$$\text{Inverse Fourier transform :} \quad h = \mathcal{F}_G^{-1}(\hat{h}) = \Phi \hat{h} \in \mathbb{R}^{n \times 1}$$

$$\text{Orthonormal basis :} \quad \Phi^\top \Phi = \Phi \Phi^\top = I_n \in \mathbb{R}^{n \times n}$$

Scalar function $h \in \mathbb{R}^{n \times 1}$ can be extended to vector-valued $h \in \mathbb{R}^{n \times d}$

[1] Hammond-et-al, Wavelets on graphs via spectral graph theory, 2011

Spectral graph convolution

- Spectral graph convolution is defined by analogy with the convolution theorem :

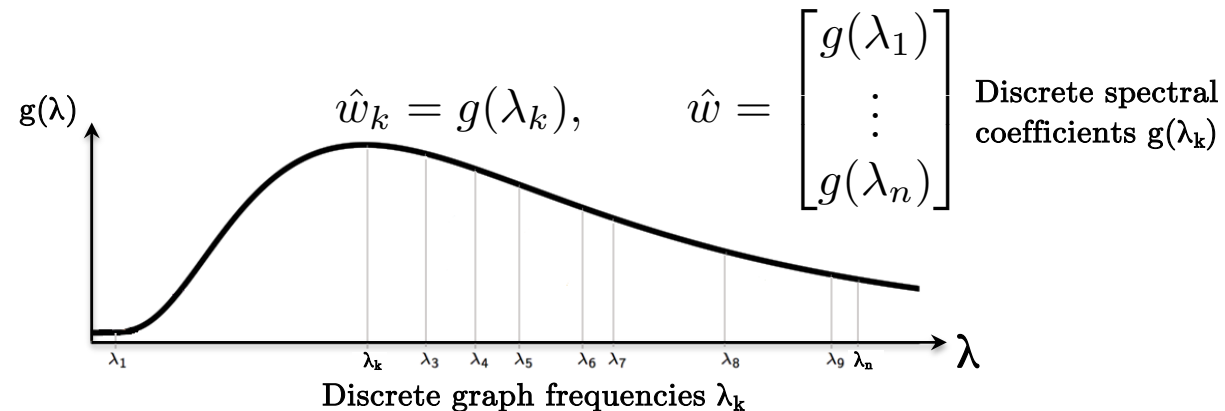
Point-wise multiplication of graph Fourier coefficients.

$$\begin{aligned} w *_G h &= \underbrace{\mathcal{F}_G^{-1}}_{\Phi} \left(\underbrace{\mathcal{F}_G(w)}_{\Phi^T w = \hat{w}} \odot \underbrace{\mathcal{F}_G(h)}_{\Phi^T h} \right), \quad \Phi \in \mathbb{R}^{n \times n}, \quad w, h \in \mathbb{R}^n \\ &= \Phi (\hat{w} \odot \Phi^T h), \quad \hat{w}, \Phi^T h \in \mathbb{R}^n, \quad \odot \text{ is the Hadamard product} \\ &= \Phi (\text{diag}(\hat{w}) \Phi^T h), \quad \hat{w}_k = g(\lambda_k) \in \mathbb{R}, \quad \text{diag}(\hat{w}) \in \mathbb{R}^{n \times n} \\ &= \Phi g(\Lambda) \Phi^T h, \quad \Lambda, g(\Lambda) \in \mathbb{R}^{n \times n} \\ (\text{notation abuse}) &= g(\Phi \Lambda \Phi^T) h, \quad L = \Phi \Lambda \Phi^T \in \mathbb{R}^{n \times n} \\ &= g(L) h \in \mathbb{R}^n \end{aligned}$$

- General graph Fourier transform is expensive with $O(n^2)$ complexity.
- No FFT exist on arbitrary graphs.

Spectral filter

- A spectral filter is a function $g(\lambda)$ defined over graph frequencies λ_k :
 - Each Laplacian eigenvalue λ_k represents a graph frequency.
 - $g(\lambda_k)$ controls how much frequency k is amplified or attenuated.
- Spectral filter g is a continuous function $g(\lambda) \in \mathbb{R}$, for $\lambda \in [0, 2]$.
- Learning independent coefficients $g(\lambda_k)$ requires $O(n)$ graph-dependent parameters.
- Spectral filtering is point-wise multiplication in the Fourier domain : $\hat{h}_k^{\text{out}} = g(\lambda_k) \hat{h}_k^{\text{in}}$.



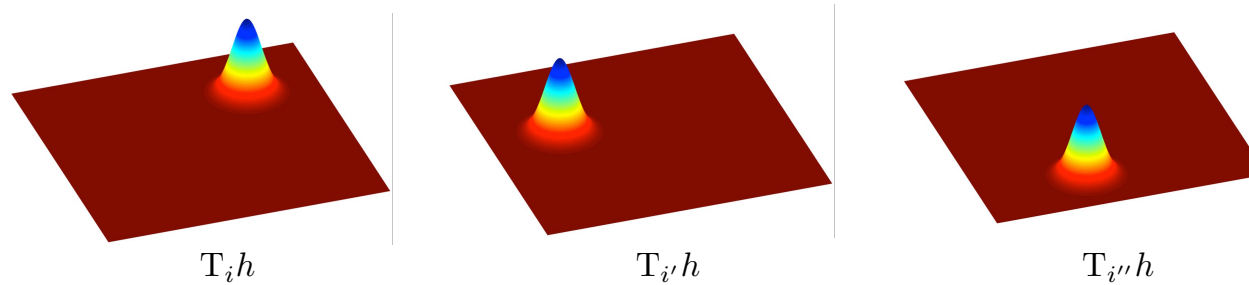
Parameterized continuous spectral function / filter $g(\lambda)$

No shift invariance on graphs

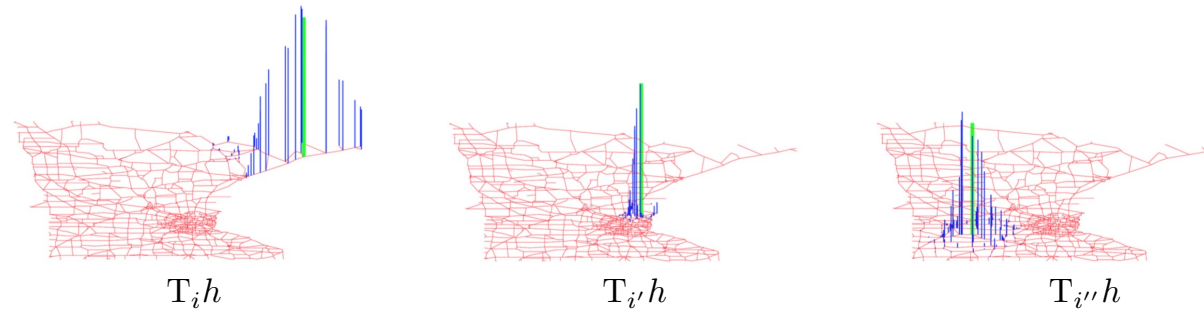
- A generalized translation of a graph signal h to vertex i can be defined by :

$$T_i h = \delta_i *_G h, \quad \delta_i \text{ is the Dirac function located at node } i$$

- Grid / Euclidean domain: Translations preserve the shape of the signal h .



- Graph domain : Translations depend on local graph structure and generally deform the signal^[1].



[1] Hammond-etal, Wavelets on graphs via spectral graph theory, 2011

Outline

- ConvNets on Euclidean Grids
 - Architecture
 - From grids to graphs
 - Convolution on grids
- Spectral Graph ConvNets
 - Spectral convolution
 - Spectral graph networks
- Spatial Graph ConvNets
 - Template matching
 - Isotropic graph networks
 - Anisotropic graph networks
- Conclusion

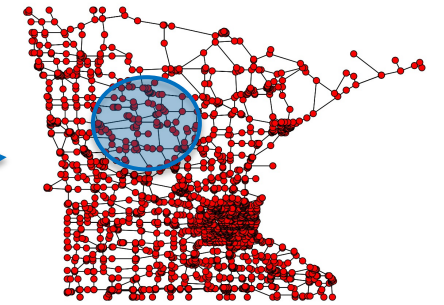
Early spectral graph networks

- The first graph spectral graph convolutional layer^[1] :

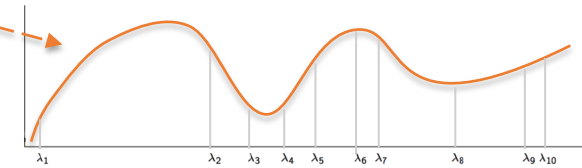
$$h^{\ell+1} = \eta(w^\ell *_G h^\ell) \in \mathbb{R}^{n \times d}, \quad \eta \text{ is e.g. ReLU, } w^\ell \text{ is the spatial filter at layer } \ell$$

$$= \eta(\Phi g^\ell(\Lambda) \Phi^T h^\ell) \in \mathbb{R}^{n \times d}, \quad g^\ell \text{ is the spectral filter at layer } \ell$$

$$\text{and } g^\ell(\Lambda) = \text{diag}(g^\ell(\lambda_1), \dots, g^\ell(\lambda_n)) \in \mathbb{R}^{n \times n}$$



Spatial filter w^ℓ



Spectral filter g^ℓ

- First spectral approach to graph ConvNets.
- Limitations :
 - Spectral filters are not guaranteed to be spatially localized.
 - $O(n)$ learnable parameters per filter.
 - $O(n^2)$ learning cost due to dense graph Fourier transforms, i.e. full matrix $\Phi \in \mathbb{R}^{n \times n}$.

[1] Bruna-LeCun-et al, Spectral Networks and Locally Connected Networks on Graphs, 2014

Spectral smoothness and spatial localization

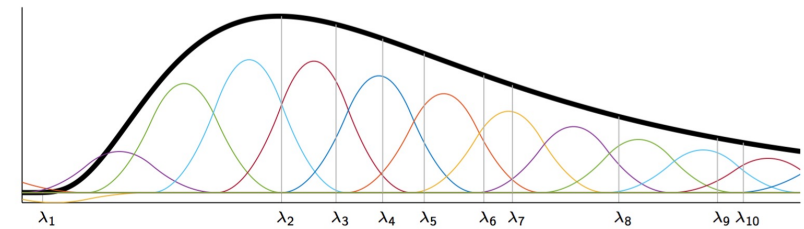
- By Parseval / Plancherel^[1] and the Fourier differentiation property, spectral smoothness is related to spatial localization.
- For $k = 1$, a small spectral derivative implies a small spatial second moment :

Smoother spectral filters produce more concentrated spatial responses.



Localized spatial filter w

$$\int |x|^{2k} |w(x)|^2 dx = \int \left| \frac{\partial^k g(\lambda)}{\partial \lambda^k} \right|^2 d\lambda$$
$$\Leftrightarrow$$



Smooth spectral filter g

[1] Mallat, A Wavelet Tour of Signal Processing, 2008

Spline-parameterized spectral filters

- Spectral graph networks with K splines, representing smooth basis functions^[1,2] :

$$\begin{aligned} h^{\ell+1} &= \eta(w^\ell *_G h^\ell) \in \mathbb{R}^{n \times d} \\ &= \eta(\Phi g^\ell(\Lambda) \Phi^T h^\ell) \in \mathbb{R}^{n \times d} \end{aligned}$$

with

$$g_{\theta^\ell}(\Lambda) = \text{diag}(B\theta^\ell) \in \mathbb{R}^{n \times n}, \quad B \in \mathbb{R}^{n \times K}, \quad \theta^\ell \in \mathbb{R}^K$$

- Two major advantages :
 - Smooth spectral filters yield better spatial localization / fast decay.
 - They reduce the number of learnable parameters from $O(n)$ to $O(K)$, i.e. independent of the graph size n and with $K \ll n$.
- Spectral coefficients, i.e. K spline coefficients θ^ℓ , are learned by backpropagation.
- However, the learning cost is still quadratic $O(n^2)$ because of dense graph Fourier transforms.

[1] Bruna-LeCun-etal, Spectral Networks and Locally Connected Networks on Graphs, 2014

[2] Henaff-LeCun-etal, Deep Convolutional Networks on Graph-Structured Data, 2015

Localized graph convolution

- Laplacian-polynomial filters are exactly localized in a K-hop neighborhood support.
- Each multiplication by Laplacian L increases the function support by one graph-hop.

$$h^{t=1} = \delta_i \in \mathbb{R}^n$$

For $t=0,2,\dots,K-1$

$$h^{t+1} = Lh^t \in \mathbb{R}^n$$

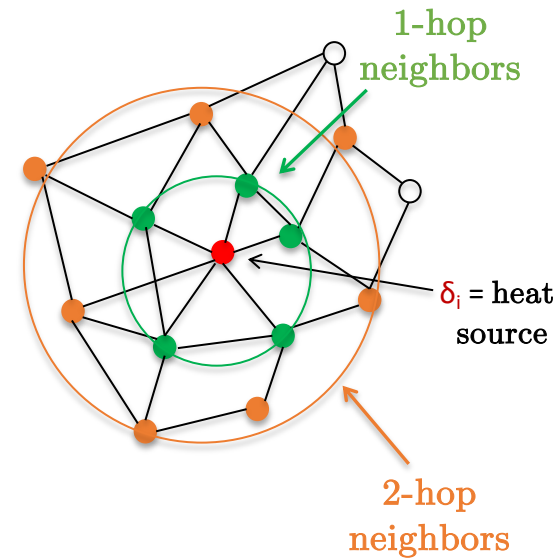
This implies :

$$h^1 = L^1 h^0 = L\delta_i$$

$$h^2 = Lh^1 = L.Lh^0 = L^2\delta_i$$

$\vdots$

$$h^K = L^K h^0 = L^K \delta_i$$



Graph convolution with Laplacian polynomials

- How to obtain scalable graph convolution without eigen-decomposition, i.e. linear training complexity w.r.t. the number of graph nodes n ?

- Direct spectral filtering with eigenvectors costs $O(n^2)$:

$$O(w *_G h) = O(\Phi g^\ell(\Lambda) \Phi^T h) = O(n^2), \quad \Phi \in \mathbb{R}^{n \times n}, \quad O \text{ is complexity}$$

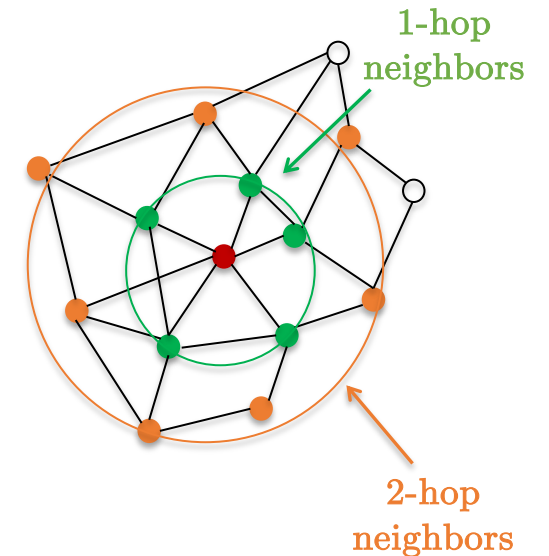
- How to avoid the eigendecomposition ?

- Parametrize the spectral filter with polynomials of Laplacian^[1] :

$$w *_G h = g(L)h \in \mathbb{R}^{n \times d}, \quad \text{with } g(L) = \sum_{k=0}^{K-1} \theta_k L^k \in \mathbb{R}^{n \times n}$$

$$= \sum_{k=0}^{K-1} \theta_k L^k h, \quad \text{computed without eigenvectors !}$$

- Degree-K filters are exactly localized within K-hop neighborhoods.



[1] Defferrard-Bresson-Vanderghenst, Convolutional neural networks on graphs with fast localized spectral filtering, 2016

Fast Laplacian graph convolution

- What is the learning complexity of this spectral convolution layer ?
- We introduce the sequence $\{X_k\}$ as follows :

$$w *_G h = g_\theta(L)h = \sum_{k=0}^{K-1} \theta_k L^k h = \sum_{k=0}^{K-1} \theta_k X_k \in \mathbb{R}^{n \times d}, \quad \theta_k \in \mathbb{R}$$

with recursive equation : $X_k = LX_{k-1} \in \mathbb{R}^{n \times d}, \quad k = 1, \dots, K-1, \quad X_0 = h.$

- The sequence $\{X_k\}$ uses sparse matrix-vector products, which cost is $O(E.K.d)$, i.e. $O(n)$ for sparse graphs (real-world graphs are sparse).
- No eigendecomposition (Φ, Λ) of the Laplacian is required.
- Polynomial Laplacian filters are spectral in formulation, but computed directly in the vertex / spatial domain (the name “spectral” has been “scary” for the community).
- Sparse graph convolutions are GPU-compatible, although generally much less efficient than convolutions on regular grids.

Implementing Laplacian graph filters

- Implementation of polynomial Laplacian filtering :

$$\begin{aligned} h^{\ell+1} &= \eta \left(\sum_{k=0}^{K-1} \theta_k^\ell L^k X_k \right) \in \mathbb{R}^{n \times d}, \quad X_k = L^k h^\ell \in \mathbb{R}^{n \times d} \\ &= \eta(\theta^\ell \bar{X}^\ell), \quad \text{with } \theta^\ell = [\theta_0^\ell, \dots, \theta_{K-1}^\ell] \in \mathbb{R}^{1 \times K}, \\ &\quad \bar{X}^\ell = [\bar{X}_0, \dots, \bar{X}_{K-1}]^T \in \mathbb{R}^{K \times nd} \\ &\quad \text{and } \bar{X}_k = \text{reshape}(X_k) \in \mathbb{R}^{nd \times 1} \end{aligned}$$

- Degree-K filters are exactly localized within K-hop neighborhoods.
- K learnable parameters per scalar filter, independent of graph size n.
- Cost is O(E), i.e. O(n) for sparse graphs.
- However, high-degree monomial bases, $\{1, x, x^2, x^3, \dots, x^{K-1}\}$, can be unstable under coefficients perturbation (poorly conditioned, hard to optimize).

ChebNet : Chebyshev graph convolution

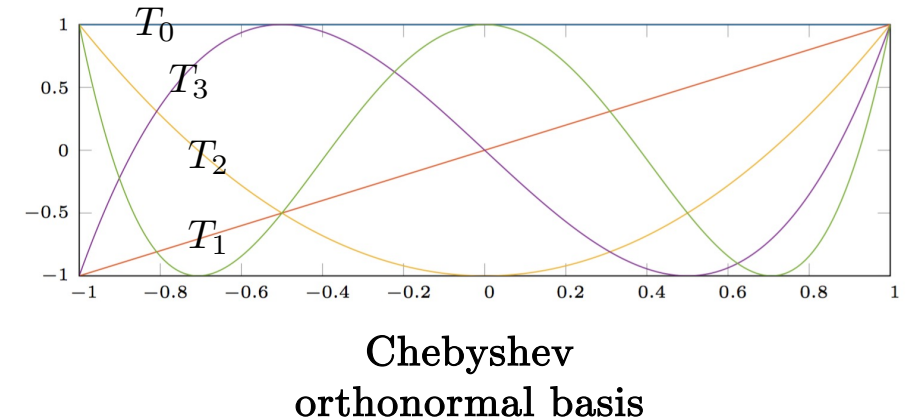
- Graph spectral convolution using Chebyshev polynomials $T_k^{[1,2]}$:

$$h^{\ell+1} = \eta(g_{\theta^\ell}(L)h^\ell) \in \mathbb{R}^{n \times d}, \quad \text{with } g_{\theta^\ell}(L) = \sum_{k=0}^{K-1} \theta_k^\ell T_k(L) \in \mathbb{R}^{n \times n}$$

$$= \eta\left(\sum_{k=0}^{K-1} \theta_k^\ell X_k^\ell\right) \in \mathbb{R}^{n \times d}, \quad X_k^\ell = T_k(L)h^\ell \in \mathbb{R}^{n \times d}$$

and new recursive equation : $X_k = 2\tilde{L}X_{k-1} - X_{k-2}, X_0, k = 1, \dots, K - 1,$
with $X_1 = \tilde{L}h, X_0 = h, \tilde{L} = 2\lambda_n^{-1}L - I_n.$

- Degree-K filters are exactly localized within K-hop neighborhoods.
- $O(K)$ learnable parameters per filter, independent of graph size n .
- Chebyshev basis is numerically stable / better conditioned than the monomial basis.

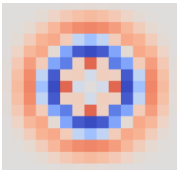


[1] Rivlin, Chebyshev Polynomials: From Approximation Theory to Algebra and Number Theory, 1990

[2] Defferrard-Bresson-Vanderghenst, Convolutional neural networks on graphs with fast localized spectral filtering, 2016

ChebNet : Fast but isotropic

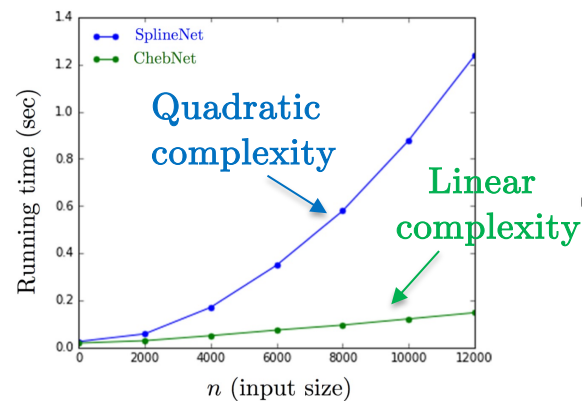
- ChebNet :
 - Generalized ConvNet for arbitrary graph domains.
 - Same $O(n)$ / linear learning complexity (but with larger complexity constant).
 - Limitation : Isotropic filtering.
- Isotropy vs. anisotropy :
 - Standard / grid ConvNet produce anisotropic filters because Euclidean grids have directional structures (up, down, left, right).
 - Spectral ConvNets like ChebNet compute isotropic filters around each node because there is no notion of directions on arbitrary graphs.



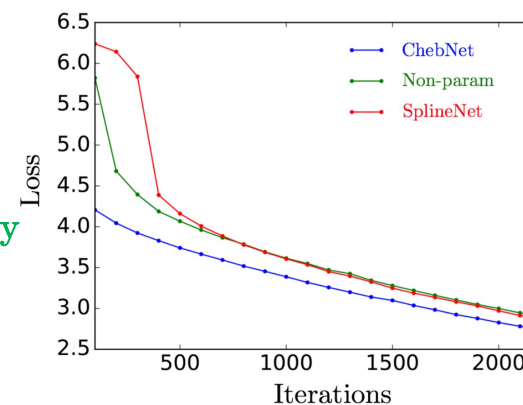
MNIST
experiment

Model	Order	Accuracy
LeNet5	-	99.33%
SplineNet	25	97.75%
ChebNet	25	99.14%

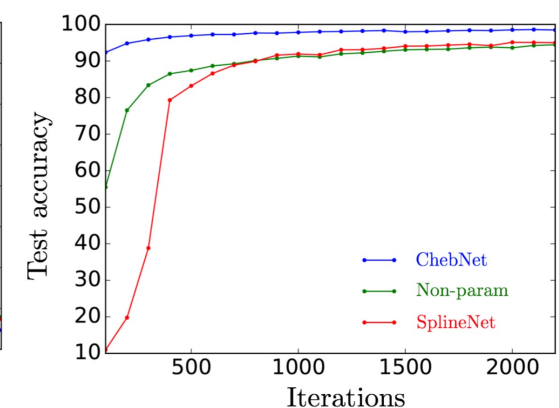
Accuracy



Running time



Optimization



Lab 1 : ChebNet

- Implement ChebNet^[1] (10th most cited paper at NeurIPS)

Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering

Michaël Defferrard

Xavier Bresson

Pierre Vandergheynst

EPFL, Lausanne, Switzerland

{michael.defferrard,xavier.bresson,pierre.vandergheynst}@epfl.ch

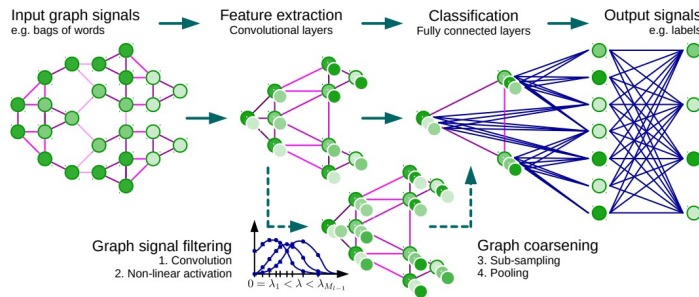


Figure 1: Architecture of a CNN on graphs and the four ingredients of a (graph) convolutional layer.

Lecture : Graph Convolutional Networks

Lab 01 : ChebNets

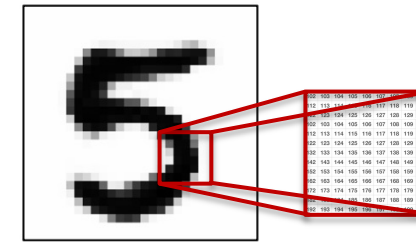
Xavier Bresson

Defferrard, Bresson, Vandergheynst, Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, 2016

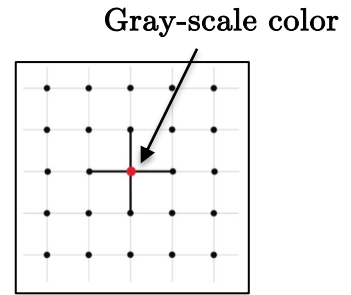
<https://arxiv.org/pdf/1606.09375>

```
# For Google Colaboratory
import sys, os
if 'google.colab' in sys.modules:
    # mount google drive
    from google.colab import drive
    drive.mount('/content/gdrive')
    path_to_file = '/content/gdrive/My Drive/GML_May23_codes/codes/08_Graph_Convnets'
    print(path_to_file)
    # change current path to the folder containing "path_to_file"
    os.chdir(path_to_file)
    !pwd
```

```
import torch
from torch.autograd import Variable
import torch.nn as nn
import collections
import time
import numpy as np
import sys
sys.path.insert(0, 'lib/')
%load_ext autoreload
%autoreload 2
```



28 x 28 image



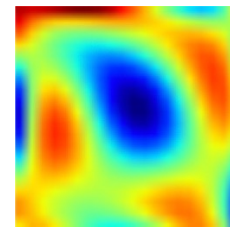
Grid

[1] Defferrard-Bresson-Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, 2016

ChebNet for multiple graphs

- Chebyshev convolution on two graph domains^[1] :

$$\begin{aligned}
 h^{\ell+1} &= \eta(g_\theta(L_1, L_2) *_{\mathcal{G}} h^\ell) \in \mathbb{R}^{n_1 \times n_2 \times d}, \quad L_1 \in \mathbb{R}^{n_1 \times n_1}, \quad L_2 \in \mathbb{R}^{n_2 \times n_2} \\
 &= \eta\left(\sum_{k_1=0}^{K_1-1} \sum_{k_2=0}^{K_2-1} g_{\theta_{k_1, k_2}^\ell} T_{k_1}(\tilde{L}_1) h^\ell T_{k_2}(\tilde{L}_2)\right), \quad \tilde{L} = 2\lambda_n^{-1} L - I_n \\
 g_\theta(\lambda_1, \lambda_2) &= \sum_{k_1=0}^{K_1-1} \sum_{k_2=0}^{K_2-1} \theta_{k_1, k_2}^\ell T_{k_1}(\Lambda_1) T_{k_2}(\Lambda_2), \quad \theta_{k_1, k_2}^\ell \in \mathbb{R}^{K_1 \times K_2}
 \end{aligned}$$

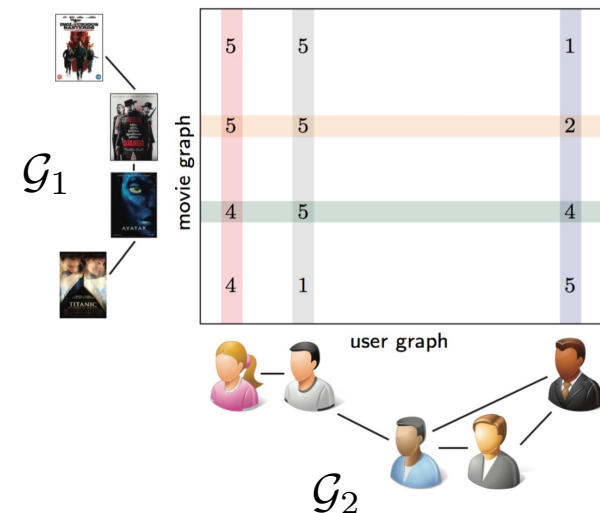


2D spectral filter
 $g_\theta(\lambda_1, \lambda_2)$

- Recommendation task :

- Ratings lie on the product of user and item graphs.
- Matrix completion regularized by both graph structures^[2].

$$\min_{h \in \mathbb{R}^{n_1 \times n_2}} \|h\|_* + \mu \|\Omega \circ (h - r)\|_F^2 + \underbrace{\mu_1 \text{tr}(h \Delta_1 h^\top)}_{\|h\|_{\mathcal{G}_1}^2} + \underbrace{\mu_2 \text{tr}(h^\top \Delta_2 h)}_{\|h\|_{\mathcal{G}_2}^2}$$

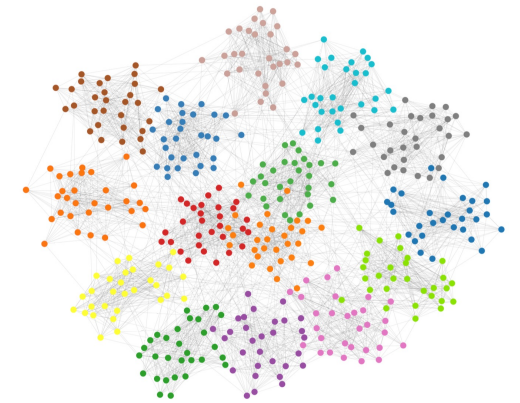


[1] Monti-Bronstein-Bresson, Geometric matrix completion with recurrent multi-graph neural networks, 2017

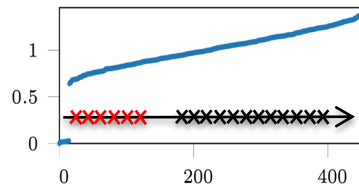
[2] Kalofolias-Bresson-Bronstein-Vanderghelynst, Matrix completion on graphs, 2014

CayleyNet : Rational spectral filters

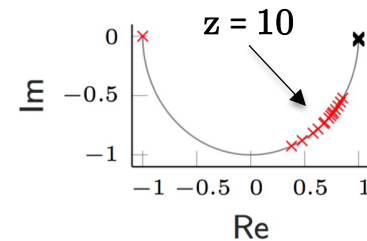
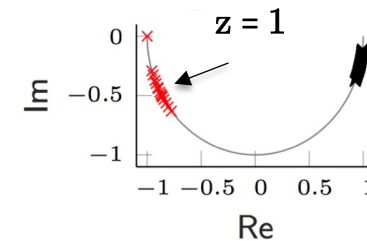
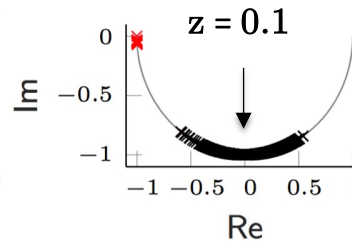
- ChebNets are unstable to produce filters with narrow spectral bands, e.g. graph communities.
- Cayley rational^[1] filters introduce a spectral zoom parameter $z > 0$.
- CayleyNets^[2] :
 - Focus on selected frequency bands through spectral zoom.
 - Spatially localized (exponential decay) and scalable on sparse graphs.
 - Richer spectral responses than polynomial filters of comparable order.
 - Laplacian-based/isotropic : No explicit directional parameterization.



$$g(L) = \theta_0 + 2\text{Re}\left\{ \sum_{k=0}^{K-1} \theta_k \frac{(z\Delta - i)^k}{(z\Delta + i)^k} \right\} \in \mathbb{R}^{n \times n}$$



Laplacian spectrum



Spectral zoom with distinct z

[1] Cayley, Sur quelques propriétés des déterminants gauches, 1846

[2] Levi-Monti-Bresson-Bronstein, CayleyNets: Graph convolutional neural networks with complex rational spectral filters, 2018

Outline

- ConvNets on Euclidean Grids
 - Architecture
 - From grids to graphs
 - Convolution on grids
- Spectral Graph ConvNets
 - Spectral convolution
 - Spectral graph networks
- **Spatial Graph ConvNets**
 - **Template matching**
 - Isotropic graph networks
 - Anisotropic graph networks
- Conclusion

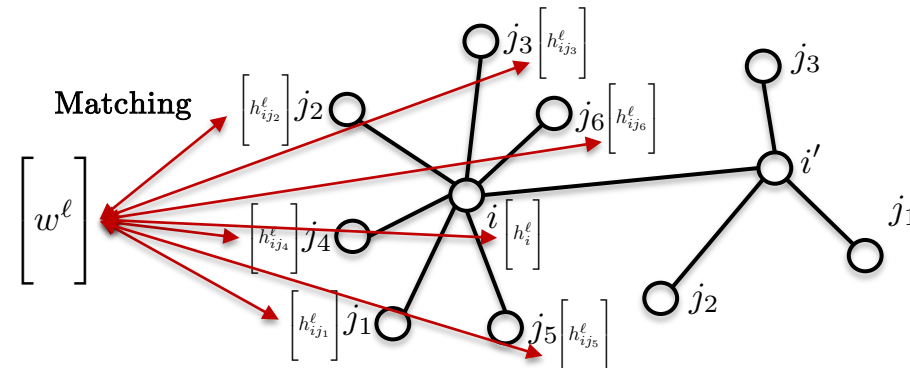
Permutation-invariant graph convolution

- How to define template matching on graphs ?
- Main issue : Neighbors have no canonical ordering or relative position.
- Node indices are arbitrary labels and carry no positional meaning.
- How to make graph convolution invariant to node label re-parametrization ?
- Apply the same learnable transformation to every neighbor and aggregate symmetrically.

$$h_i^{\ell+1} = \eta\left(\sum_{j \in \mathcal{N}_i} \langle w^\ell, h_j^\ell \rangle\right) \in \mathbb{R}, \quad w^\ell, h_j^\ell \in \mathbb{R}^d, \quad \text{one output feature}$$

$$h_i^{\ell+1} = \eta\left(\sum_{j \in \mathcal{N}_i} W^\ell h_j^\ell\right) \in \mathbb{R}^d, \quad W^\ell \in \mathbb{R}^{d \times d}, \quad \text{multiple output features}$$

$$h^{\ell+1} = \eta(Ah^\ell W^\ell) \in \mathbb{R}^{n \times d}, \quad A \in \mathbb{R}^{n \times n} \text{ is the adjacency matrix}$$



Outline

- ConvNets on Euclidean Grids
 - Architecture
 - From grids to graphs
 - Convolution on grids
- Spectral Graph ConvNets
 - Spectral convolution
 - Spectral graph networks
- **Spatial Graph ConvNets**
 - Template matching
 - **Isotropic graph networks**
 - Anisotropic graph networks
- Conclusion

Graph convolutional network (GCN)

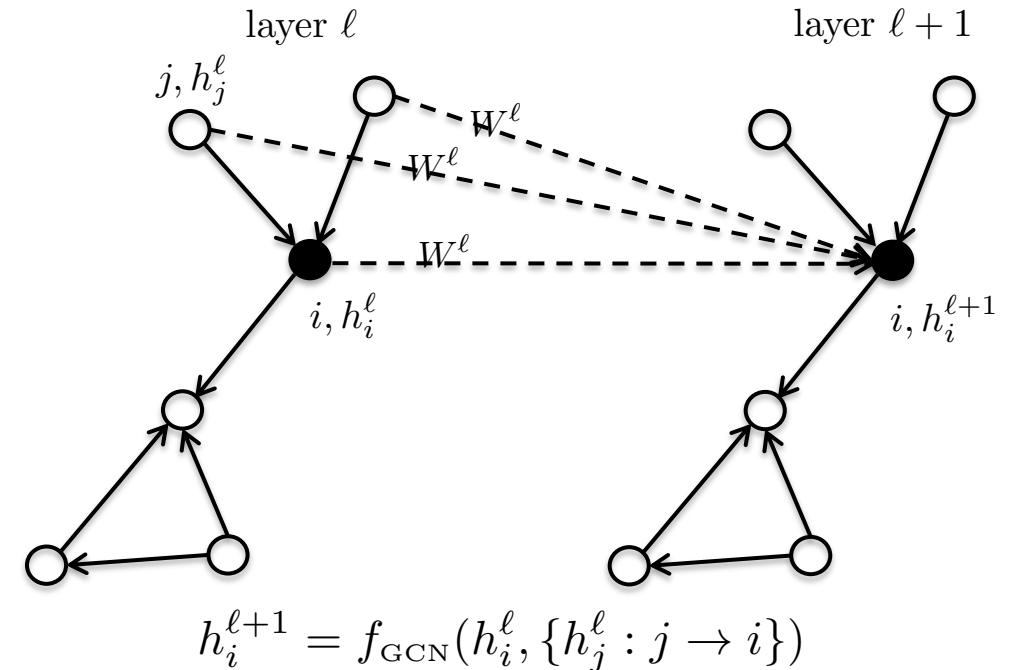
- Mathematical formulation^[1,2] :

$$h^{\ell+1} = \eta(D^{-1}Ah^{\ell}W^{\ell}) \in \mathbb{R}^{n \times d} \quad \text{mean-neighborhood aggregation}$$

$$h_i^{\ell+1} = \eta\left(\frac{1}{d_i} \sum_{j \in \mathcal{N}_i} W^{\ell} h_j^{\ell}\right) \in \mathbb{R}^d \quad \text{node-level}$$

$$h^{\ell+1} = \eta(D^{-1/2}AD^{-1/2}h^{\ell}W^{\ell}) \in \mathbb{R}^{n \times d} \quad \text{Kipf-Welling version with } A \leftarrow A + I$$

$$h_i^{\ell+1} = \eta\left(\frac{1}{\sqrt{d_i}} \sum_{j \in \mathcal{N}_i \cup \{i\}} \frac{1}{\sqrt{d_j}} W^{\ell} h_j^{\ell}\right) \in \mathbb{R}^d$$



[1] Kipf-Welling, Semi-supervised classification with graph convolutional networks, 2016

[2] Sukhbaatar-Szlam-Fergus, Learning multiagent communication with backpropagation, 2016

Basic spatial graph convolution network

- A minimal spatial GCN layer / function :
 - Handles the absence of a canonical ordering of nodes.
 - Is invariant to permutations of neighboring nodes and equivariant to global node re-labeling.
 - Uses normalization to account for varying neighborhood sizes.
 - Has a local receptive field defined by the graph neighborhood.
 - Shares weights across all nodes and edges, analogous to the weight-sharing property of convolution.
 - Has a number of learnable parameters independent of the graph size.
 - Is, however, limited to isotropic aggregation, treating all neighboring directions / features uniformly.

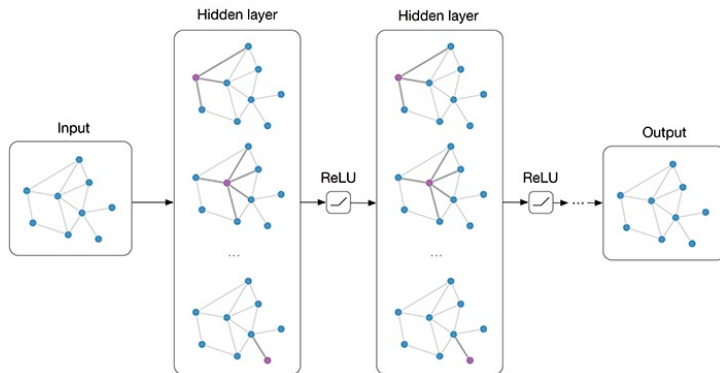
Lab 2 : GCN

- Implement GCN^[1] (Most cited paper in the GNN domain)

SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

Thomas N. Kipf
University of Amsterdam
T.N.Kipf@uva.nl

Max Welling
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
M.Welling@uva.nl



Lecture : Graph Convolutional Networks

Lab 02 : GCNs - Solution

Kipf, Welling, Semi-supervised classification with graph convolutional networks, 2016
<https://arxiv.org/pdf/1609.02907.pdf>

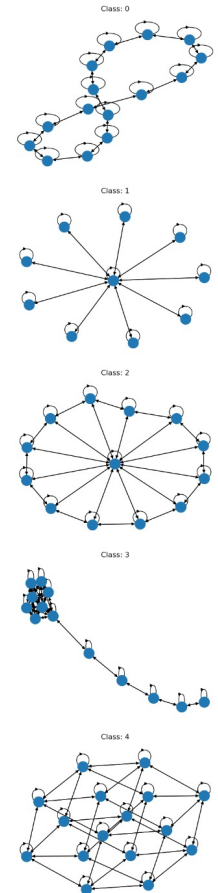
Xavier Bresson

Notebook goals :

- Add node features to DGL graphs
- Create a DGL batch of graphs
- Implement GCN message-passing function with DGL

```
# For Google Colaboratory
import sys, os
if 'google.colab' in sys.modules:
    # mount google drive
    from google.colab import drive
    drive.mount('/content/gdrive')
    path_to_file = '/content/gdrive/My Drive/GML_May23_codes/codes/05_Graph_Convnets'
    print(path_to_file)
    # change current path to the folder containing "path_to_file"
    os.chdir(path_to_file)
    !pwd
    !pip install dgl==1.0.0 # Install DGL

# Libraries
import dgl
from dgl.data import MiniGCDataset
import matplotlib.pyplot as plt
import networkx as nx
import torch
```



[1] Kipf-Welling, Semi-supervised classification with graph convolutional networks, 2016

From ChebNet to GCN

- Chebyshev spectral convolution^[1] :

$$h^{\ell+1} = \eta(g_{\theta^\ell}(L)h^\ell) = \eta\left(\sum_{k=0}^{K-1} \theta_k^\ell T_k(L)h^\ell\right) \in \mathbb{R}^{n \times d} \quad (1)$$

- GCN^[2] can be derived as a first-order approximation of ChebNet spectral filters^[2] :

Suppose $K = 2$, assume $\lambda_n = 2$, tie $\theta_0^\ell = -\theta_1^\ell = \theta^\ell$, then (1) reduces to

$h^{\ell+1} = \eta(\theta^\ell(\mathbf{I}_n + D^{-1/2}AD^{-1/2})h^\ell) \in \mathbb{R}^{n \times d}$, matrix formulation

$h_i^{\ell+1} = \eta\left(\frac{1}{\sqrt{d_i}} \sum_{j \in \mathcal{N}_i \cup \{i\}} \frac{1}{\sqrt{d_j}} W^\ell h_j^\ell\right) \in \mathbb{R}^d$, node-level formulation

given that $L = \mathbf{I}_n - D^{-1/2}AD^{-1/2}$, $T_0(L) = \mathbf{I}_n$,

$$T_1(L) = \tilde{L} = 2\lambda_n^{-1}L - \mathbf{I}_n = L - \mathbf{I} = D^{-1/2}AD^{-1/2}$$

- As multiple layers may cause numerical instabilities and exploding / vanishing gradients, GCN applies the renormalization trick, i.e. self-loop :

$$A \leftarrow A + \mathbf{I}_n \in \mathbb{R}^{n \times n}$$

[1] Defferrard-Bresson-Vanderghelynst, Convolutional neural networks on graphs with fast localized spectral filtering, 2016

[2] Kipf-Welling, Semi-supervised classification with graph convolutional networks, 2016

GraphSAGE : Sample and aggregate

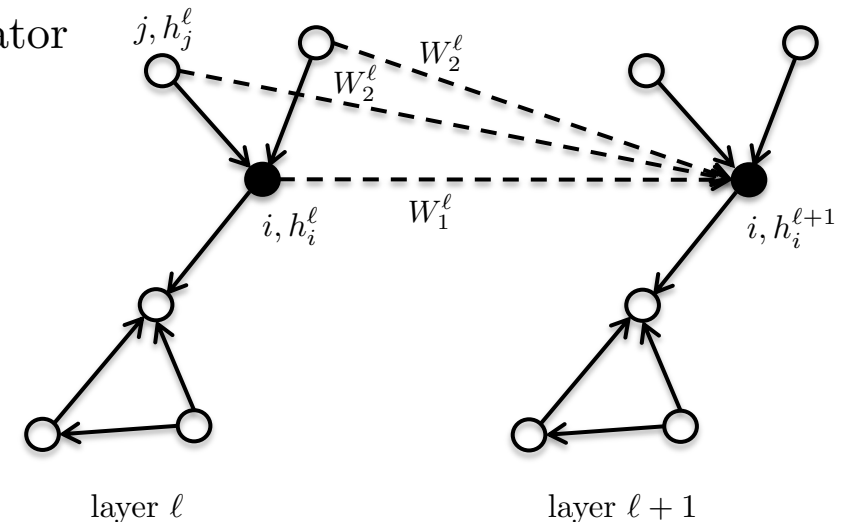
- GraphSAGE^[1] : Neighborhood aggregation with neighbor sampling.
- Combine the central-node representation with an aggregated neighborhood representation.
- Mean/max aggregators are permutation invariant, unlike the LSTM aggregator which is order sensitive.

$\tilde{\mathcal{N}}_i^\ell \sim \text{UniformSample}(\mathcal{N}_i, S_\ell) \subseteq \mathcal{N}_i$, sampled 1-hop neighbors

$m_i^\ell = \text{AGGREGATE}_\ell(\{h_j^\ell : j \in \tilde{\mathcal{N}}_i^\ell\}) \in \mathbb{R}^d$, AGGREGATE = {Mean, Max, LTSM}

$h_i^{\ell+1} = \sigma(W^\ell[h_i^\ell \| m_i^\ell]) = \sigma(W_1^\ell h_i^\ell + W_2^\ell m_i^\ell) \in \mathbb{R}^d$, message-passing layer

$= \sigma(W_1^\ell h_i^\ell + \frac{1}{S_\ell} \sum_{j \in \tilde{\mathcal{N}}_i^\ell} W_2^\ell h_j^\ell) \in \mathbb{R}^d$, using mean-aggregator



[1] Hamilton-Ying-Leskovec, Inductive representation learning on large graphs, 2017

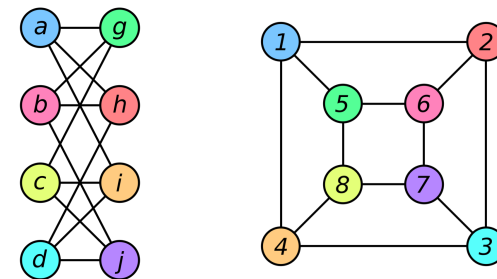
GIN : Expressive power of message passing

- Graph Isomorphism Network^[1] (GIN) has the same discriminative power as the 1-WL graph isomorphism test^[2].
- Two graphs are isomorphic if a bijection between their nodes preserves adjacency.
- The sum aggregation with an expressive MLP can distinguish multisets of neighbor features.

$$h_i^{\ell+1} = \text{MLP}_{\theta^\ell}((1 + \epsilon^\ell)h_i^\ell + \sum_{j \in \mathcal{N}_i} h_j^\ell) \in \mathbb{R}^{n \times d}$$

$$\text{with } \text{MLP}_{\theta^\ell}(h) = \text{ReLU} [\text{BN} (W_2^\ell \text{ReLU} [\text{BN} (W_1^\ell h)])]$$

- GIN is an isotropic message-passing GNN.



Two isomorphic graphs

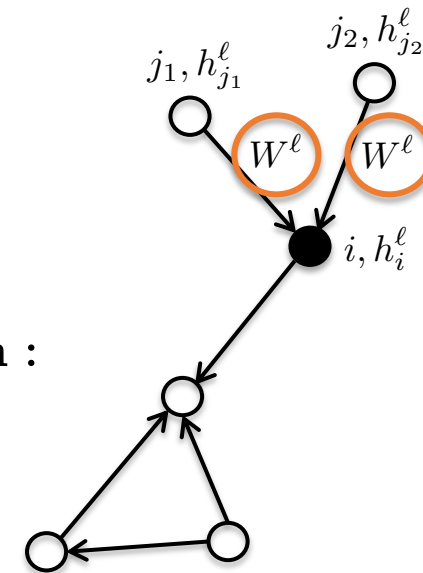
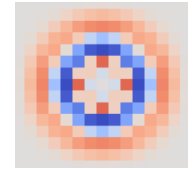
[1] Xu-Hu-Leskovec-Jegelka, How powerful are graph neural networks, 2018

Outline

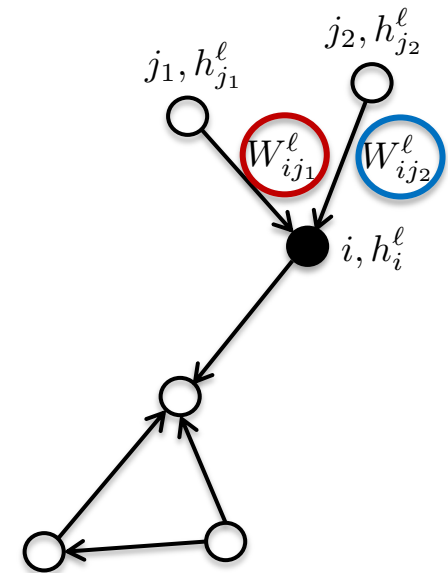
- ConvNets on Euclidean Grids
 - Architecture
 - From grids to graphs
 - Convolution on grids
- Spectral Graph ConvNets
 - Spectral convolution
 - Spectral graph networks
- **Spatial Graph ConvNets**
 - Template matching
 - Isotropic graph networks
 - **Anisotropic graph networks**
- Conclusion

From isotropic to anisotropic GNNs

- Reminder :
 - Grid ConvNets are anisotropic : Filter weights depend on relative positions (up, down, left, right).
 - Connectivity-only GNNs s.a. ChebNet, GCN, GraphSage, GIN are isotropic : Neighbors have no canonical directional labels.
- How to introduce anisotropy in GNNs ?
 - Use geometric or semantic edge features^[1,2] when available (e.g. bond type, relative position).
 - Learn edge-dependent weights while preserving equivariance to node re-parametrization/re-labeling.
 - Edge coordinates / edge gates / attention mechanism : MoNets^[3], GAT^[5], GatedGCNs^[4] treat neighbors differently.



Isotropic weights



Anisotropic weights

- [1] Gilmer-Schoenholz-Riley-Vinyals-Dahl, Neural message passing for quantum chemistry, 2017
[2] Bresson-Laurent, A Two-Step Graph Convolutional Decoder for Molecule Generation, 2019
[3] Monti-Bronstein-et-al, Geometric deep learning on graphs and manifolds using mixture model CNNs, 2016
[4] Bresson-Laurent, Residual gated graph convnets, 2017
[5] Velickovic-Bengio-et-al, Graph Attention Networks, 2018

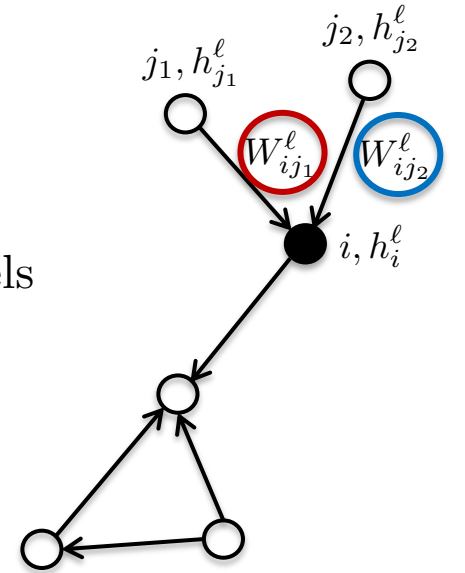
MoNet : Mixture-model graph convolution

- MoNet^[1] introduces anisotropy by learning neighbor-dependent Gaussian kernels^[2] over edge coordinates, allowing different neighbors to contribute differently.
- Unlike spectral methods, MoNet defines localized spatial filters directly in the graph domain, making it applicable to irregular graphs.

$$h_i^{\ell+1} = \text{ReLU}\left(\sum_{k=1}^K \sum_{j \in \mathcal{N}_i} e_{ij}^{k,\ell} W_k^\ell h_j^\ell\right) \in \mathbb{R}^d, \quad \text{edge-dependent weights}$$

$$e_{ij}^{k,\ell} = \exp\left(-\frac{1}{2} (u_{ij} - \mu_k^\ell)^\top (\Sigma_k^\ell)^{-1} (u_{ij} - \mu_k^\ell)\right) \in \mathbb{R}, \quad \text{Gaussian kernels}$$

$$u_{ij} = \begin{bmatrix} d_i^{-1/2} \\ d_j^{-1/2} \end{bmatrix} \in \mathbb{R}^2, \quad \text{edge coordinates}$$



[1] Monti-Boscaini-Masci-Rodolà-Svoboda-Bronstein, Geometric deep learning on graphs and manifolds using mixture model CNNs, 2016

[2] Dempster-Laird-Rubin, Maximum Likelihood from Incomplete Data Via the EM Algorithm, 1977

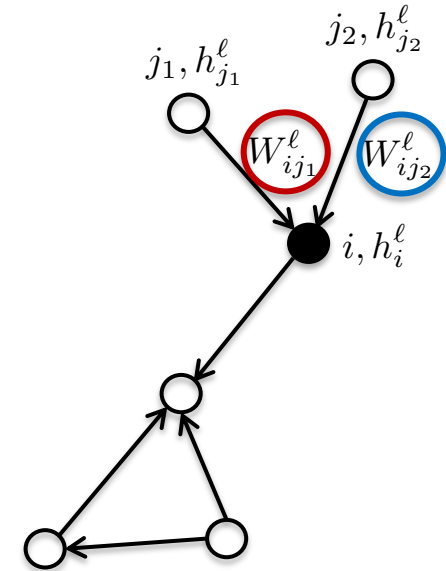
GAT : Attention-based anisotropic aggregation

- Graph Attention Network^[1] (GAT) introduces learnable attention coefficients α_{ij} that assign different importance to different neighbors, enabling anisotropic message passing while preserving permutation invariance for neighbors.
- Multi-head attention improves representation capacity by learning multiple aggregation patterns, similar to the Transformer attention mechanism^[2].

$$e_{ij}^{k,\ell} = \text{LeakyReLU} \left((a^{k,\ell})^\top [W^{k,\ell} h_i^\ell \| W^{k,\ell} h_j^\ell] \right) \in \mathbb{R}$$

$$\alpha_{ij}^{k,\ell} = \frac{\exp(e_{ij}^{k,\ell})}{\sum_{j' \in \mathcal{N}_i \cup \{i\}} \exp(e_{ij'}^{k,\ell})} \in \mathbb{R}_+$$

$$h_i^{\ell+1} = \parallel_{k=1}^K \eta \left(\sum_{j \in \mathcal{N}_i \cup \{i\}} \alpha_{ij}^{k,\ell} W^{k,\ell} h_j^\ell \right) \in \mathbb{R}^d$$



[1] Velickovic-Bengio-et al, Graph Attention Networks, 2018

[2] Vaswani-et al, Attention is all you need, 2017

Lab 3 : GAT

- Implement GAT^[1] (second most cited paper in the GNN domain)

GRAPH ATTENTION NETWORKS

Petar Veličković*

Department of Computer Science and Technology
University of Cambridge
petar.velickovic@cst.cam.ac.uk

Guillem Cucurull*

Centre de Visió per Computador, UAB
gcucurull@gmail.com

Arantxa Casanova*

Centre de Visió per Computador, UAB
ar.casanova.8@gmail.com

Adriana Romero

Montréal Institute for Learning Algorithms
adriana.romero.soriano@umontreal.ca

Pietro Liò

Department of Computer Science and Technology
University of Cambridge
pietro.liò@cst.cam.ac.uk

Yoshua Bengio

Montréal Institute for Learning Algorithms
yoshua.umontreal@gmail.com

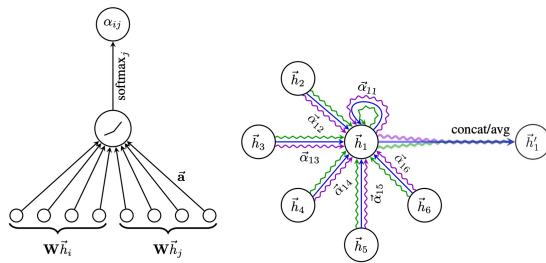


Figure 1: **Left:** The attention mechanism $a(W_{h_i}, W_{h_j})$ employed by our model, parametrized by a weight vector $\tilde{a} \in \mathbb{R}^{2F}$, applying a LeakyReLU activation. **Right:** An illustration of multi-head attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain $\tilde{h}'_1$.

Lecture : Graph Convolutional Networks

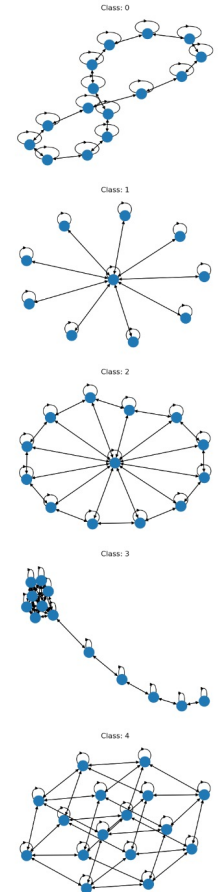
Lab 03 : GAT

Velickovic, Cucurull, Casanova, Romero, Lio, Bengio, Graph Attention Networks, 2017
<https://arxiv.org/pdf/1710.10903>

Xavier Bresson

```
# For Google Colaboratory
import sys, os
if 'google.colab' in sys.modules:
    # mount google drive
    from google.colab import drive
    drive.mount('/content/gdrive')
    path_to_file = '/content/gdrive/My Drive/GML_May23_codes/codes/05_Graph_Convnets'
    print(path_to_file)
    # change current path to the folder containing "path_to_file"
    os.chdir(path_to_file)
    !pwd
    !pip install dgl==1.0.0 # Install DGL
```

```
# Libraries
import dgl
from dgl.data import MiniGCDataSet
import matplotlib.pyplot as plt
import networkx as nx
import torch
from torch.utils.data import DataLoader
import torch.nn as nn
import time
```

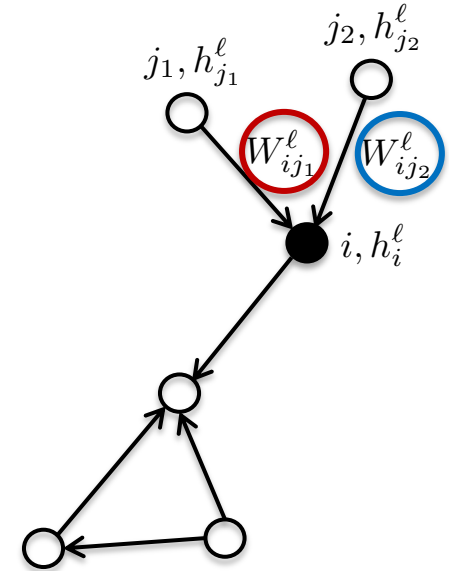


[1] Velickovic-Bengio-et-al, Graph Attention Networks, 2018

GatedGCN : Anisotropic edge gating

- GatedGCN^[1] use learnable edge gates for anisotropic neighborhood aggregation.
- The model is based on Perona-Malik's anisotropic PDE^[2] generalized on graphs.
- Edge gates are vectorial / feature-wise, unlike scalar attention coefficient^[3].
- Edge representations are explicitly updated, enabling edge-aware message passing, and critical for edge prediction tasks.
- Residual connections and batch normalization speed up optimization of multi-layer GNNs, and improve generalization.

$$e_{ij}^{\ell+1} = e_{ij}^{\ell} + \text{ReLU}\left(\text{BN}\left(V_1^{\ell}h_i^{\ell-1} + V_2^{\ell}h_j^{\ell-1} + V_3^{\ell}e_{ij}^{\ell}\right)\right) \in \mathbb{R}^d$$
$$w_{ij}^{\ell+1} = \frac{\sigma(e_{ij}^{\ell+1})}{\sum_{j' \in \mathcal{N}_i} \sigma(e_{ij'}^{\ell+1}) + \varepsilon} \in \mathbb{R}^d$$
$$h_i^{\ell+1} = h_i^{\ell} + \text{ReLU}\left(\text{BN}\left(W_1^{\ell}h_i^{\ell} + \sum_{j \in \mathcal{N}_i} w_{ij}^{\ell+1} \odot W_2^{\ell}h_j^{\ell}\right)\right) \in \mathbb{R}^d$$



[1] Bresson-Laurent, Residual gated graph convnets, 2017

[2] Perona-Malik, Scale-space and edge detection using anisotropic diffusion, 1987

[3] Vaswani-etal, Attention is all you need, 2017

Lab 4 : GatedGCN

- Implement GatedGCN^[1] (robust GNN architecture across the benchmark^[2])

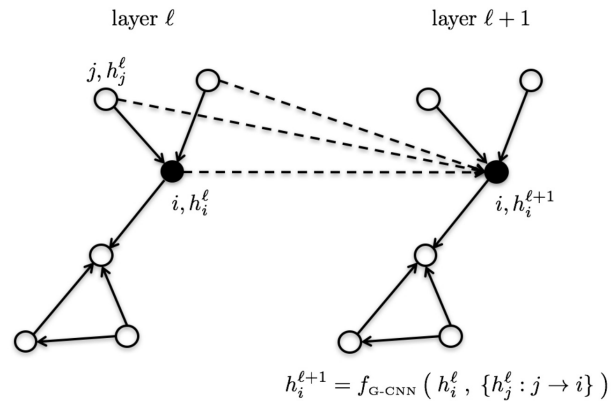
RESIDUAL GATED GRAPH CONVNETS

Xavier Bresson*

School of Computer Science and Engineering
Nanyang Technological University, Singapore
xbresson@ntu.edu.sg

Thomas Laurent

Department of Mathematics
Loyola Marymount University
tlaurent@lmu.edu



Lecture : Graph Convolutional Networks

Lab 04 : GatedGCNs

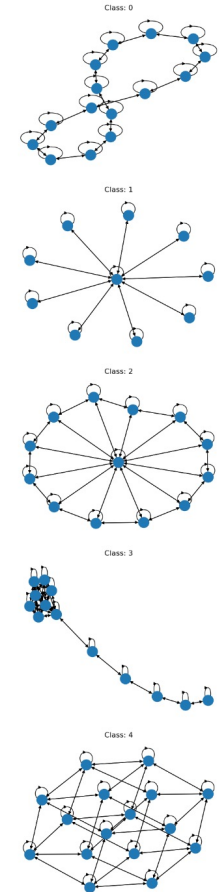
Bresson, Laurent, Residual Gated Graph ConvNets, 2017

<https://arxiv.org/pdf/1711.07553>

Xavier Bresson

```
# For Google Colaboratory
import sys, os
if 'google.colab' in sys.modules:
    # mount google drive
    from google.colab import drive
    drive.mount('/content/gdrive')
    path_to_file = '/content/gdrive/My Drive/GML_May23_codes/codes/05_Graph_Convnets'
    print(path_to_file)
    # change current path to the folder containing "path_to_file"
    os.chdir(path_to_file)
    !pwd
    !pip install dgl==1.0.0 # Install DGL
```

```
# Libraries
import dgl
from dgl.data import MiniGCDataset
import matplotlib.pyplot as plt
import networkx as nx
import torch
from torch.utils.data import DataLoader
import torch.nn as nn
import time
```



[1] Bresson-Laurent, Residual gated graph convnets, 2017

[2] Dwivedi-Joshi-Laurent-Bengio-Bresson, Benchmarking graph neural networks, 2020

Lab 5 : GatedGCN for molecular regression

- Implement GatedGCN^[1] for quantum chemistry^[2]

Lecture : Graph Convolutional Networks

Lab 05 : GatedGCNs for chemical regression - Solution

Bresson, Laurent, Residual Gated Graph ConvNets, 2017

<https://arxiv.org/pdf/1711.07553>

Bresson, Laurent, A two-step graph convolutional decoder for molecule generation, 2019

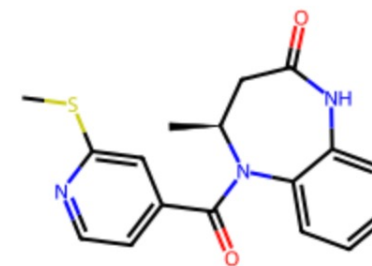
<https://arxiv.org/pdf/1906.03412>

Xavier Bresson

Notebook goals :

- Implement a two-layer MLP for regression
- Define the Mean Absolute Error (MAE) as regression loss

```
# For Google Colaboratory
import sys, os
if 'google.colab' in sys.modules:
    # mount google drive
    from google.colab import drive
    drive.mount('/content/gdrive')
    path_to_file = '/content/gdrive/My Drive/GML_May23_codes/codes/08_Graph_Convnets'
    print(path_to_file)
    # change current path to the folder containing "path_to_file"
    os.chdir(path_to_file)
    !pwd
    !pip install dgl==1.0.0 # Install DGL
```



[1] Bresson-Laurent, Residual gated graph convnets, 2017

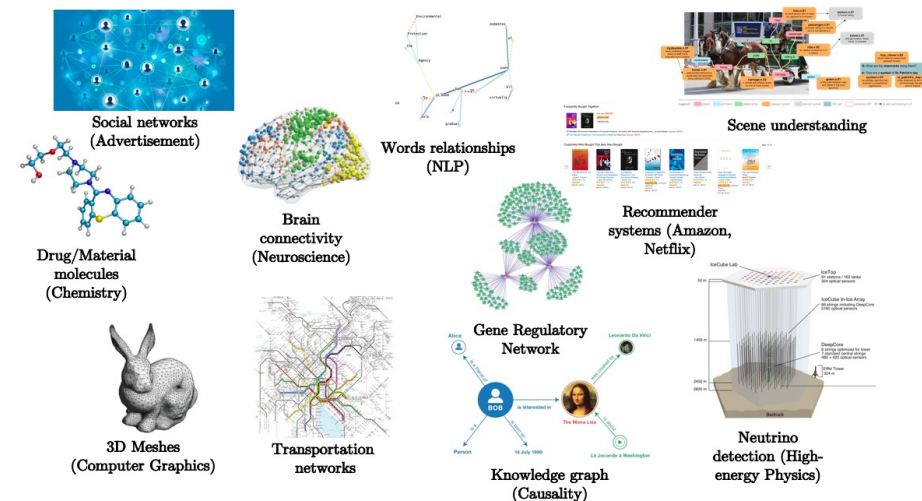
[2] Bresson-Laurent, A two-step graph convolutional decoder for molecule generation, 2019

Outline

- ConvNets on Euclidean Grids
 - Architecture
 - From grids to graphs
 - Convolution on grids
- Spectral Graph ConvNets
 - Spectral convolution
 - Spectral graph networks
- Spatial Graph ConvNets
 - Template matching
 - Isotropic graph networks
 - Anisotropic graph networks
- **Conclusion**

Conclusion

- Key contributions
 - Generalize convolution, i.e. local, shared-weight filters, from regular grids to graph data.
 - Define graph convolution through spectral filtering and spatial neighborhood aggregation / pattern matching.
 - Scalable computation : Linear complexity per layer for sparse graphs.
 - GPU-compatible sparse operations, but less hardware-efficient than dense grid convolutions.
 - Expressive permutation-equivariant models, with well-understood theoretical limitations.
 - Extensions to multiple, heterogeneous, and dynamic graphs.
 - Multiple applications across science, engineering and industry.





Questions?