Convolutional Neural Networks on Graphs

Xavier Bresson

School of Computer Science and Engineering
NTU, Singapore

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

NATIONAL RESEARCH FOUNDATION
PRIME MINISTER'S OFFICE
SINGAPORE

M. Defferrard
EPFL

P. Vandergheynst
EPFL

M. Bronstein
USI

F. Monti
USI

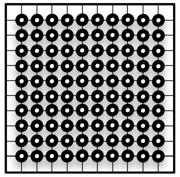
R. Levie
TAU

T. Laurent
LMU

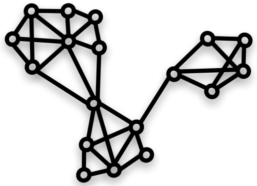
IPAM Workshop "New Deep Learning Techniques"
Feb 7th 2017

Outline

- **Part 1: Euclidean ConvNets**
  - *Architecture*
  - *Non-Euclidean data*

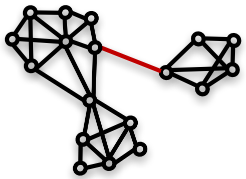


- **Part 2: Spectral ConvNets for Fixed Graphs**



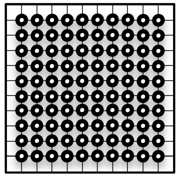
  - **Spectral Graph Theory**
    - *Graph convolution*
    - *Graph coarsening*
  - **Spectral ConvNets**
    - *SplineNets*
    - *ChebNets* [NIPS'16]*
    - *GraphConvNets*
    - *CayleyNets**
    - *Multiple fixed graphs* [NIPS'17]*

- **Part 3: ConvNets for Variable Graphs**
  - *Graph learning problems**

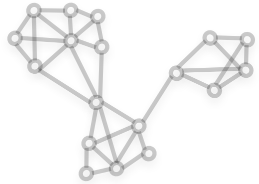


- **Conclusion**

Outline

➢ **Part 1: Euclidean ConvNets** 

- *Architecture*
- *Non-Euclidean data*

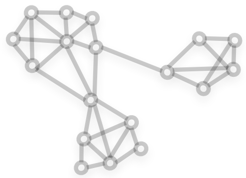
➢ Part 2: Spectral ConvNets for Fixed Graphs 

➢ Spectral Graph Theory

- *Graph convolution*
- *Graph coarsening*

➢ Spectral ConvNets

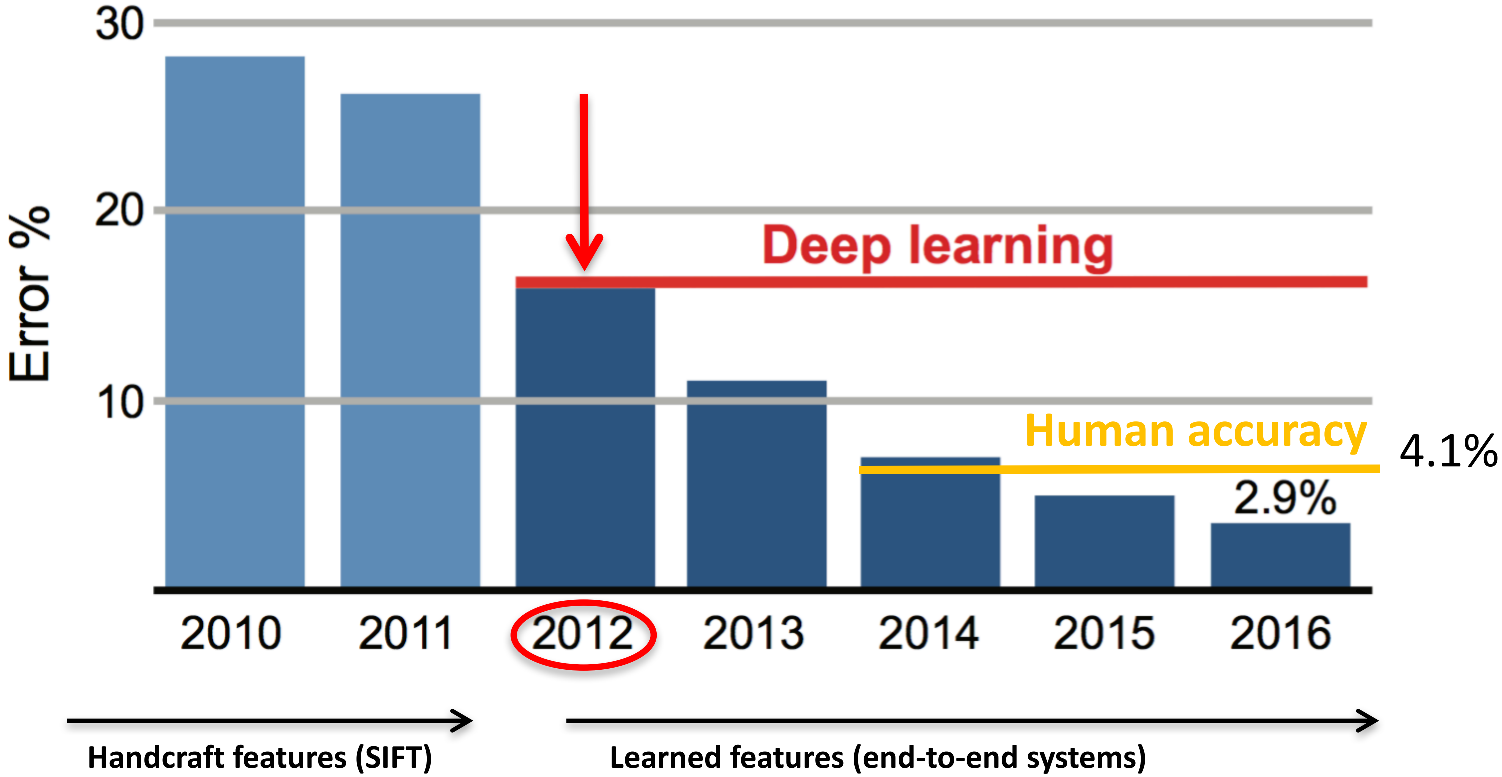
- *SplineNets*
- *ChebNets* [NIPS'16]*
- *GraphConvNets*
- *CayleyNets**
- *Multiple fixed graphs* [NIPS'17]*

➢ Part 3: ConvNets for Variable Graphs 

- *Graph learning problems**

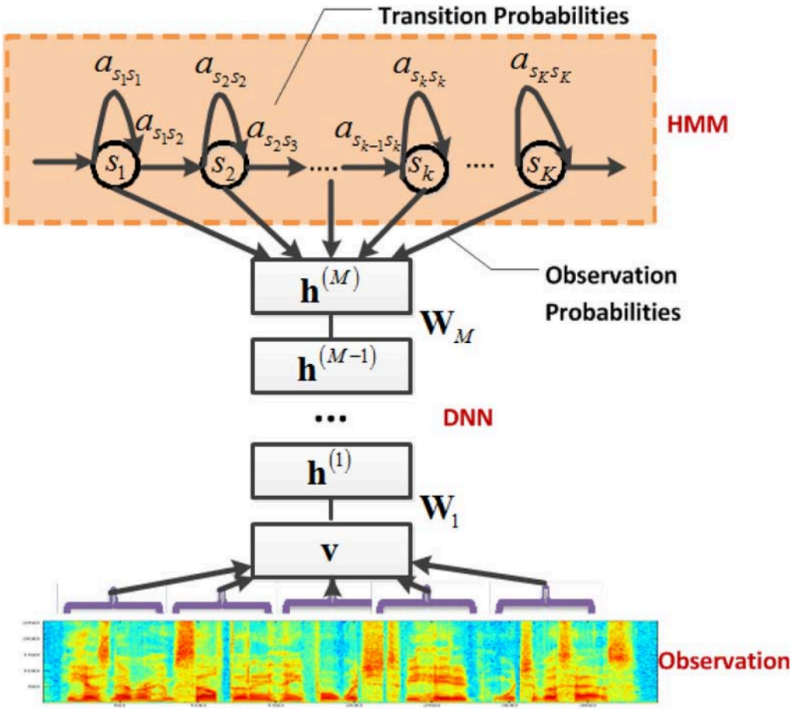
➢ Conclusion

ConvNets: A breakthrough in image recognition...



[1] LeCun, Bottou, Bengio, Haffner 1998
[2] Krizhevsky, Sutskever Hinton, 2012

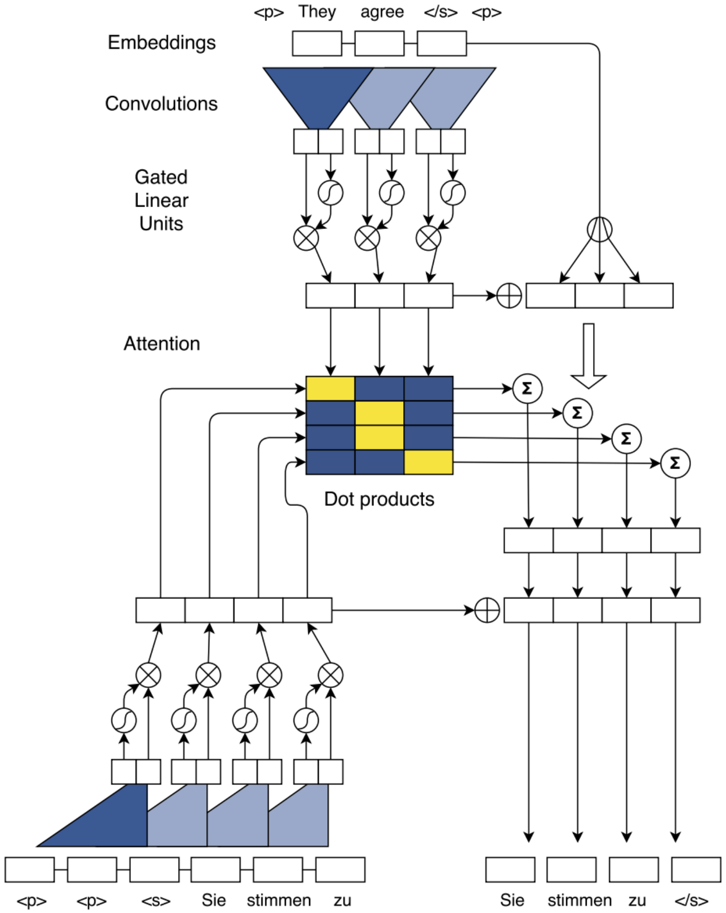
in speech recognition...



Acoustic model	Recog \ WER	RT03S FSH	Hub5 SWB
Traditional features	1-pass −adapt	27.4	23.6
Deep Learning	1-pass −adapt	18.5	16.1

[3] Dahl, Yu, Deng, Acero, 2010
[4] Hinton, Deng, Yu, Dahl et al. 2012

in language translation...



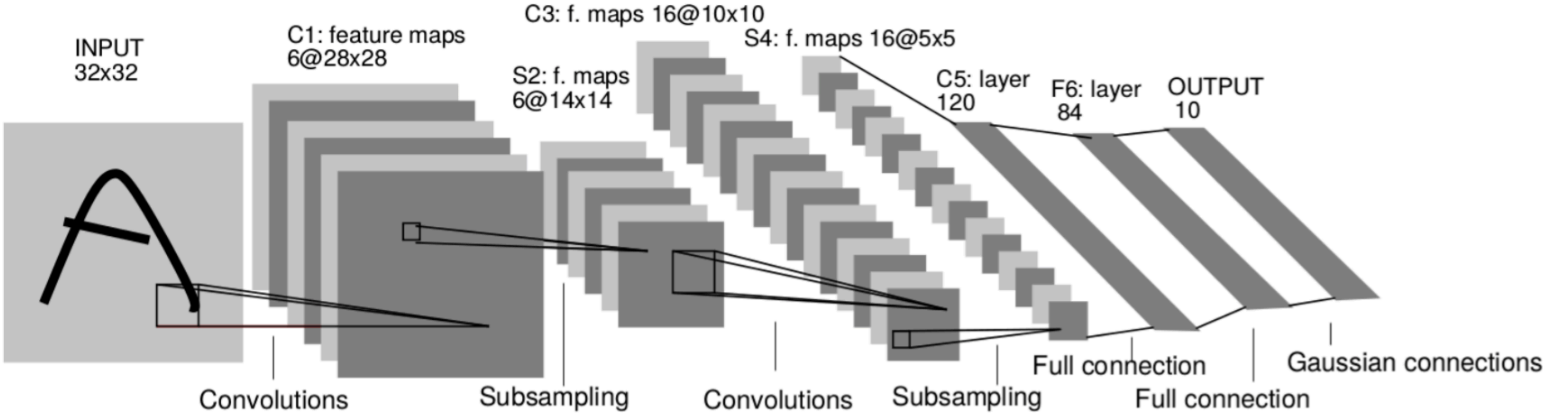
WMT'14 English-German	BLEU
Wu et al. (2016) GNMT	26.20
Wu et al. (2016) GNMT + RL	26.30
ConvS2S	26.43

WMT'14 English-French	BLEU
Zhou et al. (2016)	40.4
Wu et al. (2016) GNMT	40.35
Wu et al. (2016) GNMT + RL	41.16
ConvS2S	41.44
ConvS2S (10 models)	41.62

	BLEU	Time (s)
GNMT GPU (K80)	31.20	3,028
GNMT CPU 88 cores	31.20	1,322
GNMT TPU	31.21	384
ConvS2S GPU (K40) $b = 1$	33.45	327
ConvS2S GPU (M40) $b = 1$	33.45	221
ConvS2S GPU (GTX-1080ti) $b = 1$	33.45	142
ConvS2S CPU 48 cores $b = 1$	33.45	142
ConvS2S GPU (K40) $b = 5$	34.10	587
ConvS2S CPU 48 cores $b = 5$	34.10	482
ConvS2S GPU (M40) $b = 5$	34.10	406
ConvS2S GPU (GTX-1080ti) $b = 5$	34.10	256

[5] Gehring, Auli, Grangier, Yarats, Dauphin, 2017

An architecture for high-dimensional learning

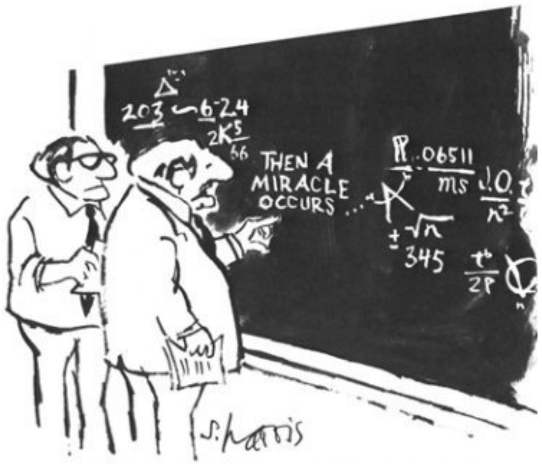


- **Curse of dimensionality:**

  dim(image) = 512 x 512 ≈ 10^6

  For N=10 samples/dim ⇒ $10^{1,000,000}$ points

- **ConvNets are powerful to solve high-dimensional learning problems.**

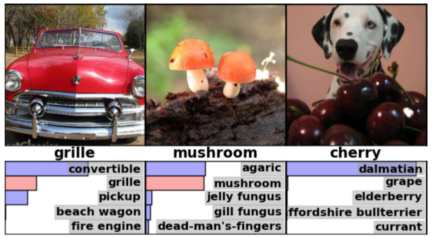


"I think you should be more explicit here in step two."

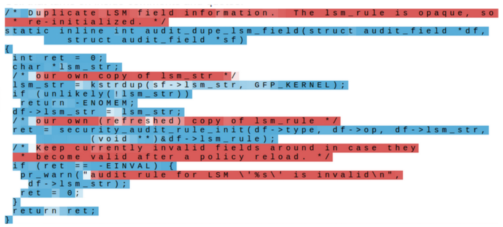
ConvNets

- Main assumption: Data (images, videos, sounds) are compositional, they are formed of patterns that are:
  - Local
  - Stationary
  - Multi-scale (hierarchical)

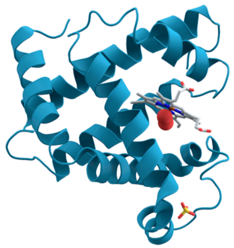
- ConvNets leverage the compositionality structure: They extract compositional features and feed them to classifier, recommender, etc (end-to-end).



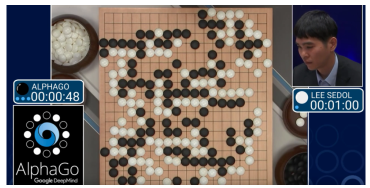
Computer Vision



NLP



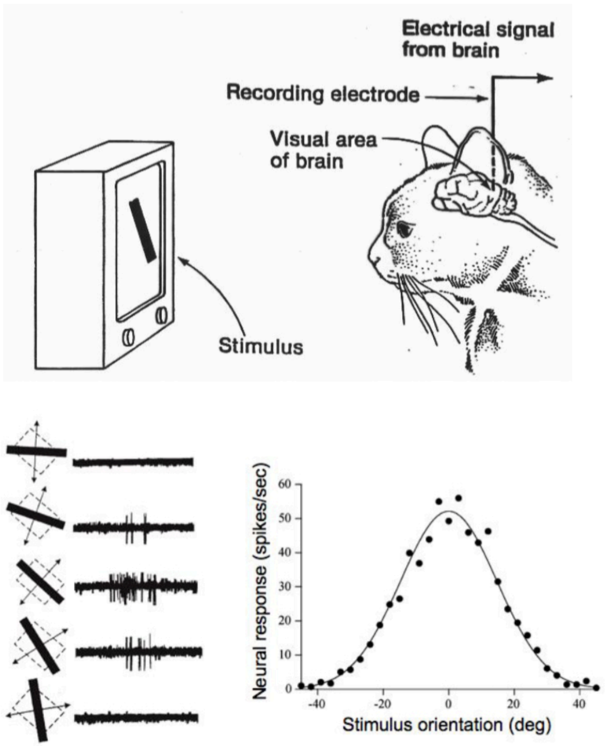
Drug discovery

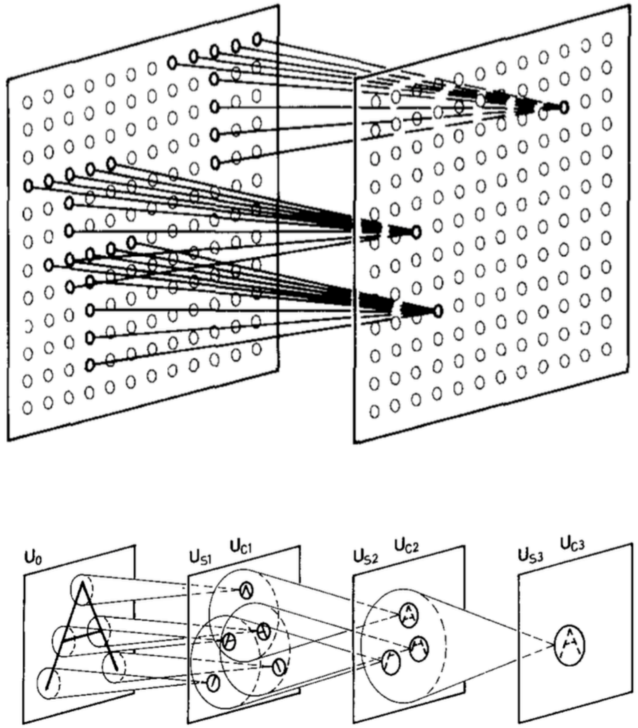


Games

Key property

- **Locality:** Property inspired by visual cortex neurons.
- **Local receptive fields**[6] activate in the presence of local features.





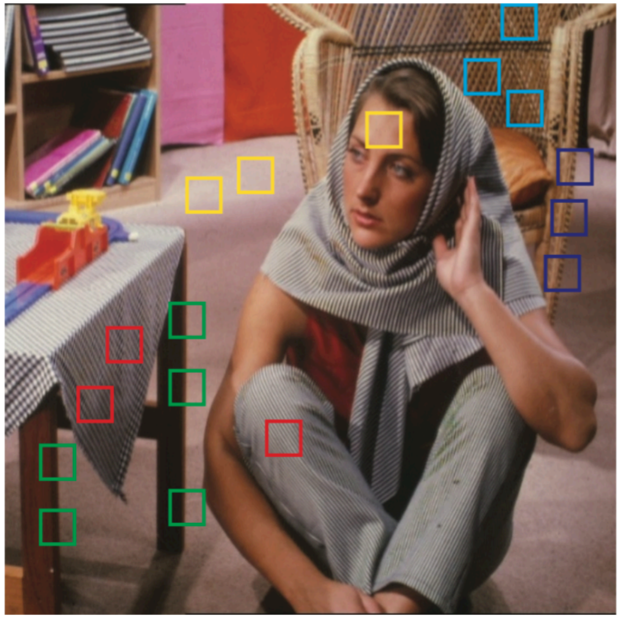
Neocognitron[7]

[6] Hubel, Wiesel 1962
[7] Fukushima 1980

Key property

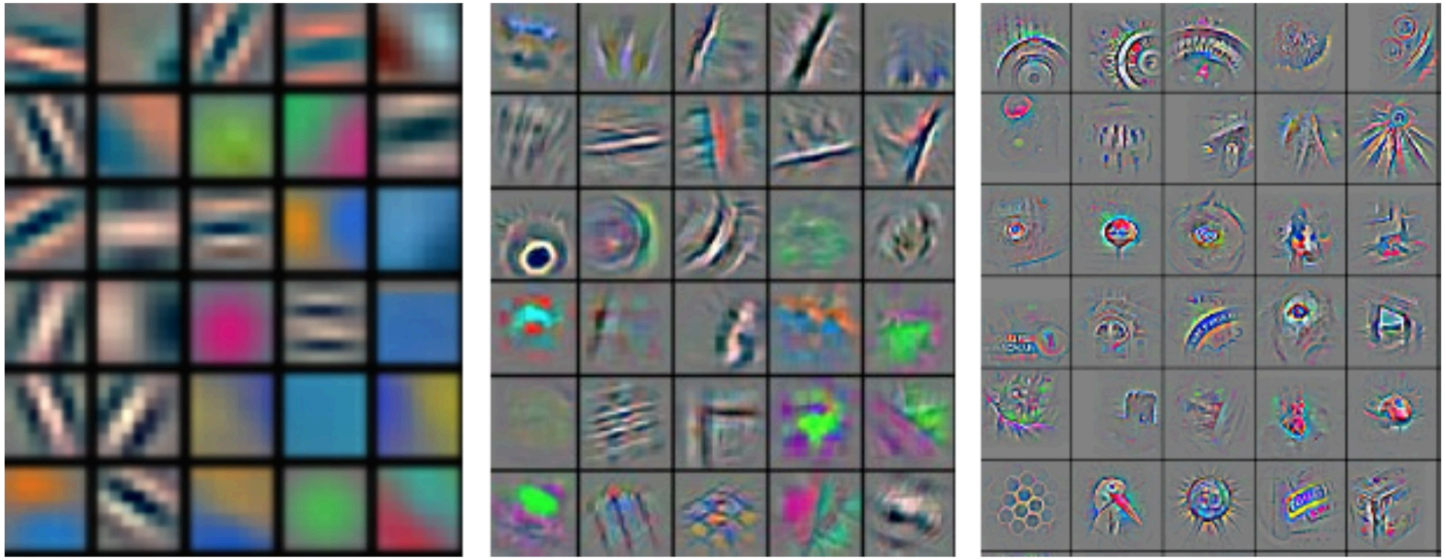
- Stationarity ⇔ Translation invariance (global invariance)

- Local stationarity ⇔ Similar patches are shared across the data domain (local invariance, good for intra-class variations)



Key property

- Multi-scale: Simple structures combine to compose slightly more abstract structures, and so on, in a hierarchical way.
- Inspired by brain visual primary cortex (V1 and V2 neurons).

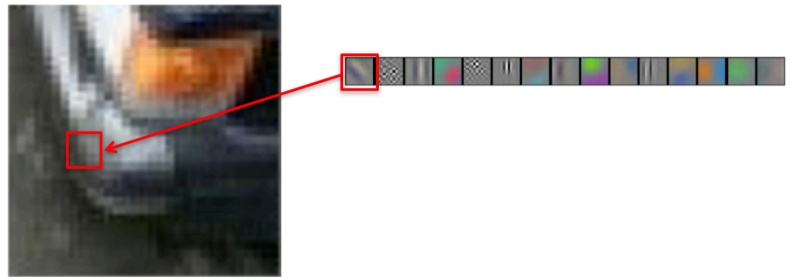


Features learned by ConvNet become increasingly more complex at deeper layers[8].

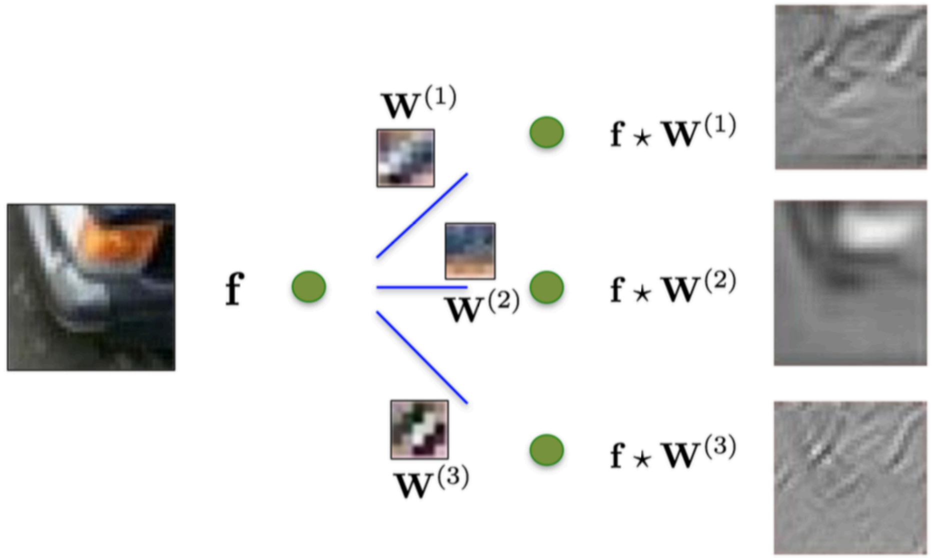
[8] Zeiler, Fergus 2013

Implementation complexity

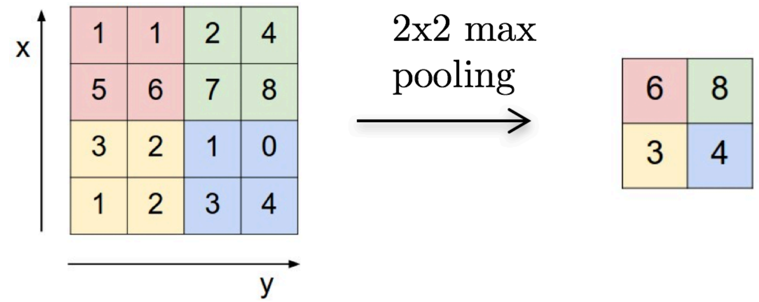
- **Locality:** Compact support kernels $\Rightarrow O(1)$ parameters per filter.



- **Stationarity:** Convolutional operators $\Rightarrow O(n\log n)$ in general (FFT) and $O(n)$ for compact kernels.

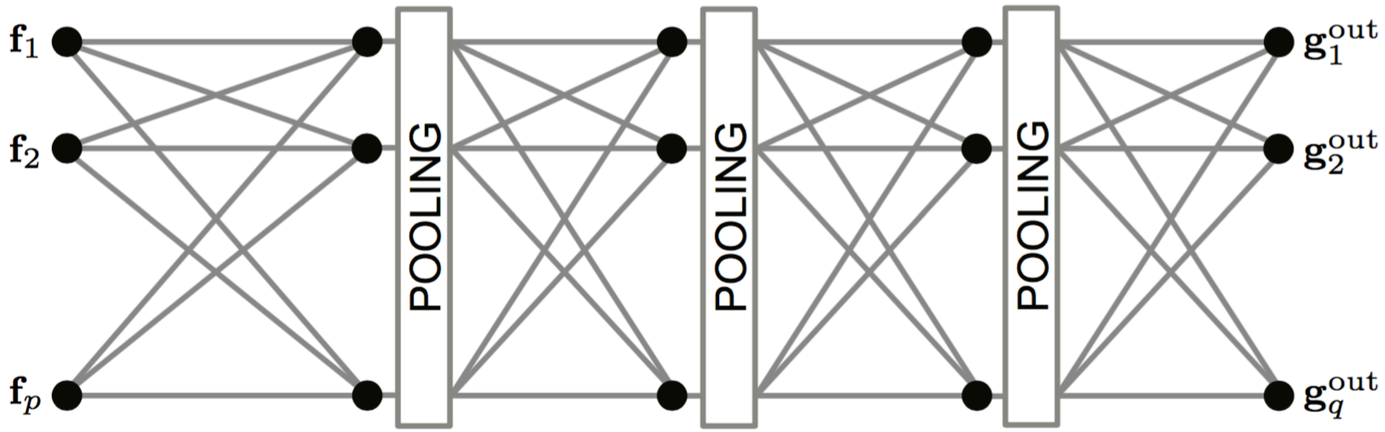


- **Multi-scale:** Downsampling + pooling $\Rightarrow O(n)$



Compositional layers

$$\begin{aligned}\mathbf{f}_l &= l\text{-th image feature (R,G,B channels), } \dim(\mathbf{f}_l) = n \times 1 \\ \mathbf{g}_l^{(k)} &= l\text{-th feature map, } \dim(\mathbf{g}_l^{(k)}) = n_l^{(k)} \times 1\end{aligned}$$



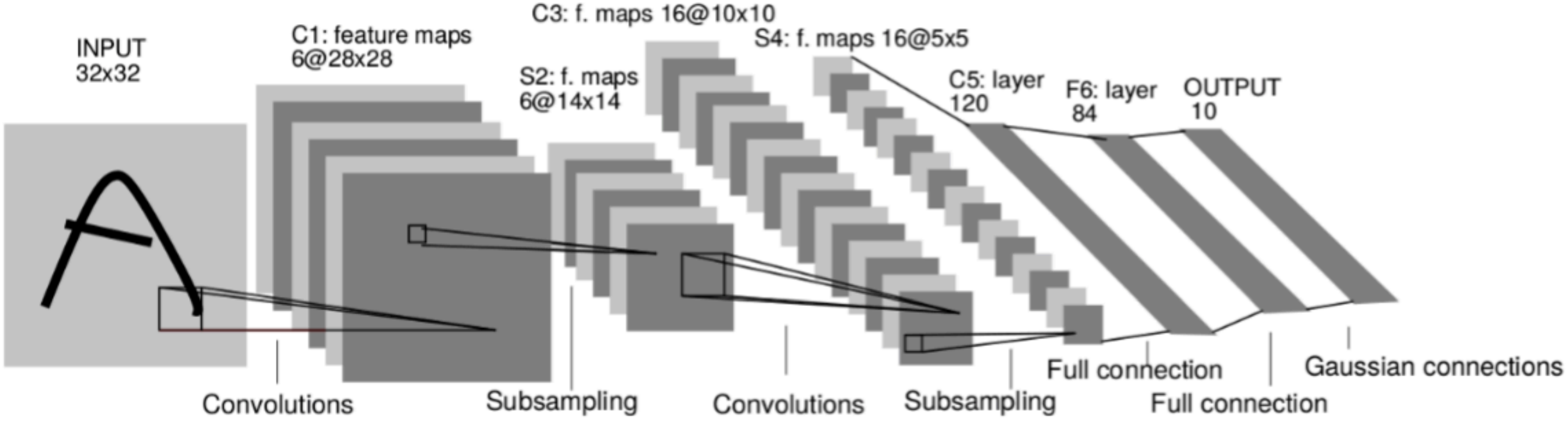
Compositional features consist of multiple convolutional + pooling layers.

Convolutional layer $\quad \mathbf{g}_l^{(k)} = \xi\left(\sum_{l'=1}^{q_{k-1}} \mathbf{W}_{l,l'}^{(k)} \star \xi\left(\sum_{l'=1}^{q_{k-2}} \mathbf{W}_{l,l'}^{(k-1)} \star \xi\left(\cdots \mathbf{f}_{l'}\right)\right)\right)$

Activation, e.g. $\quad \xi(x) = \max\{x, 0\}$ rectified linear unit (ReLU)

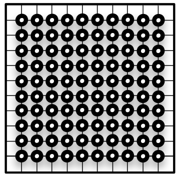
Pooling $\quad \mathbf{g}_l^{(k)}(x) = \|\mathbf{g}_l^{(k-1)}(x') : x' \in \mathcal{N}(x)\|_p \quad p = 1, 2, \text{ or } \infty$

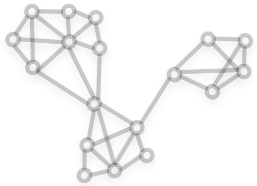
ConvNets



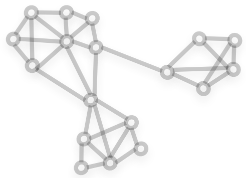
- ☺ Filters localized in space (Locality)
- ☺ Convolutional filters (Stationarity)
- ☺ Multiple layers (Multi-scale)
- ☺ O(1) parameters per filter (independent of input image size n)
- ☺ O(n) complexity per layer (filtering done in the spatial domain)

Outline

- **Part 1: Euclidean ConvNets** 
  - *Architecture*
  - *Non-Euclidean data*



- Part 2: Spectral ConvNets for Fixed Graphs
  - Spectral Graph Theory
    - *Graph convolution*
    - *Graph coarsening*
  - Spectral ConvNets
    - *SplineNets*
    - *ChebNets* [NIPS'16]*
    - *GraphConvNets*
    - *CayleyNets**
    - *Multiple fixed graphs* [NIPS'17]*

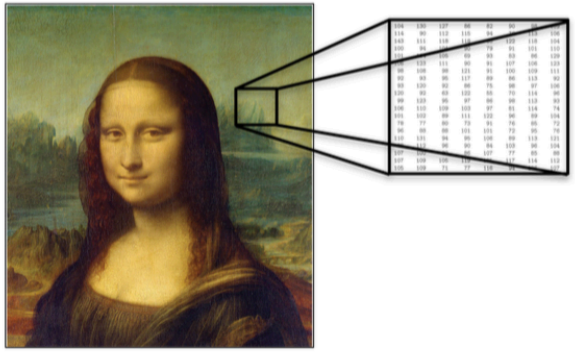


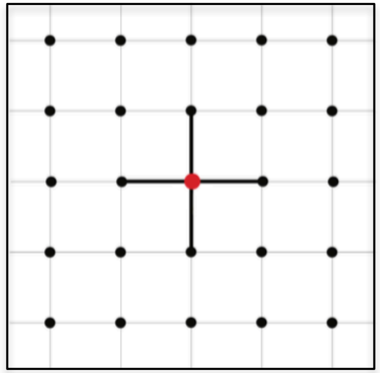
- Part 3: ConvNets for Variable Graphs
  - *Graph learning problems**

- Conclusion

Data domain for ConvNets

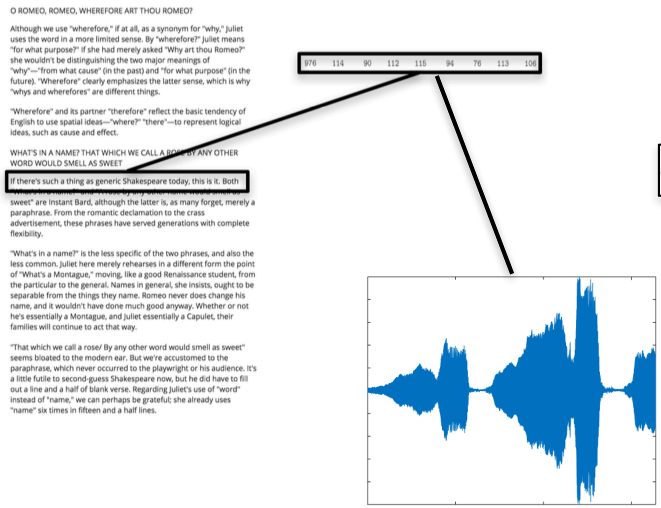
- Image, volume, video: 2D, 3D, 2D+1 Euclidean domains

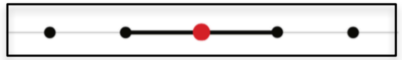




2D grids

- Sentence, word, sound: 1D Euclidean domain



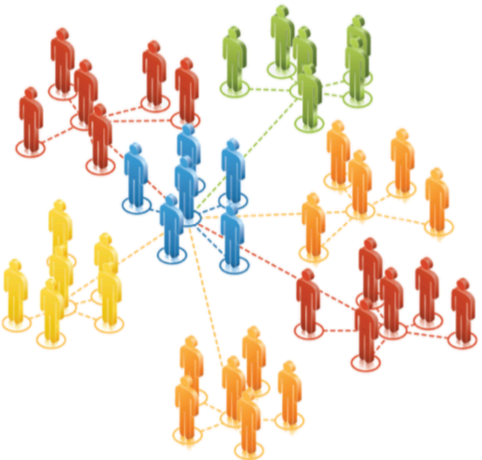


1D grid

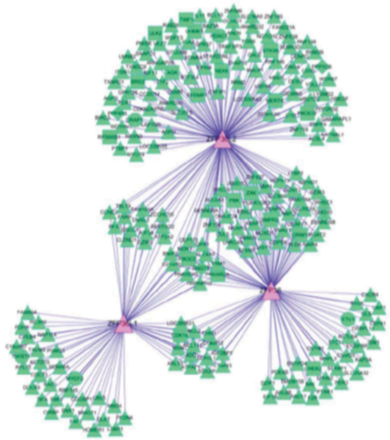
- These domains have nice regular spatial structures.

  $\Rightarrow$ All ConvNet operations are math well defined and fast (convolution, pooling).

Non-Euclidean data

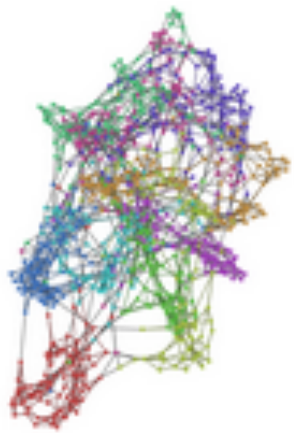


Social networks

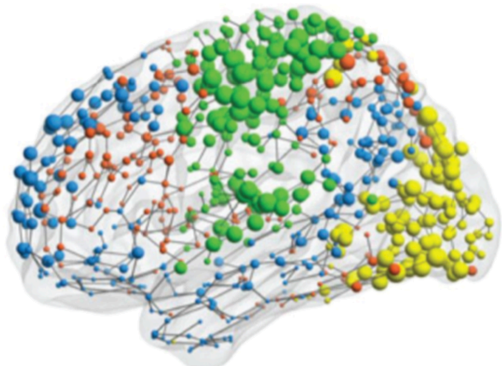


Regulatory networks

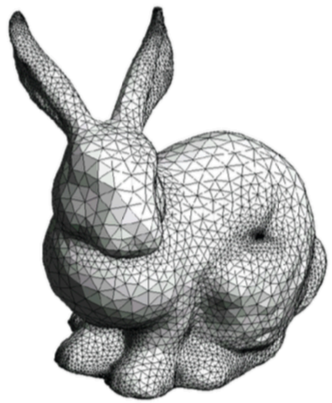
=



Graphs/
Networks



Functional networks



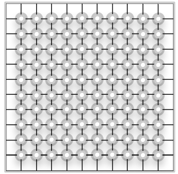
3D shapes

- Also chemistry, NLP, physics, social science, communication networks, etc.

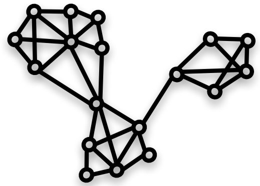
Challenges

- How to extend ConvNets to graph-structured data?
- Assumption: Non-Euclidean data are locally stationary and manifest hierarchical structures.
- How to define compositionality on graphs? (convolution and pooling on graphs)
- How to make them fast? (linear complexity)

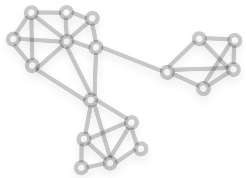
Outline



- Part 1: Euclidean ConvNets
  - *Architecture*
  - *Non-Euclidean data*



- **Part 2: Spectral ConvNets for Fixed Graphs**
  - Spectral Graph Theory
    - *Graph convolution*
    - *Graph coarsening*
  - Spectral ConvNets
    - *SplineNets*
    - *ChebNets* [NIPS'16]*
    - *GraphConvNets*
    - *CayleyNets**
    - *Multiple fixed graphs* [NIPS'17]*

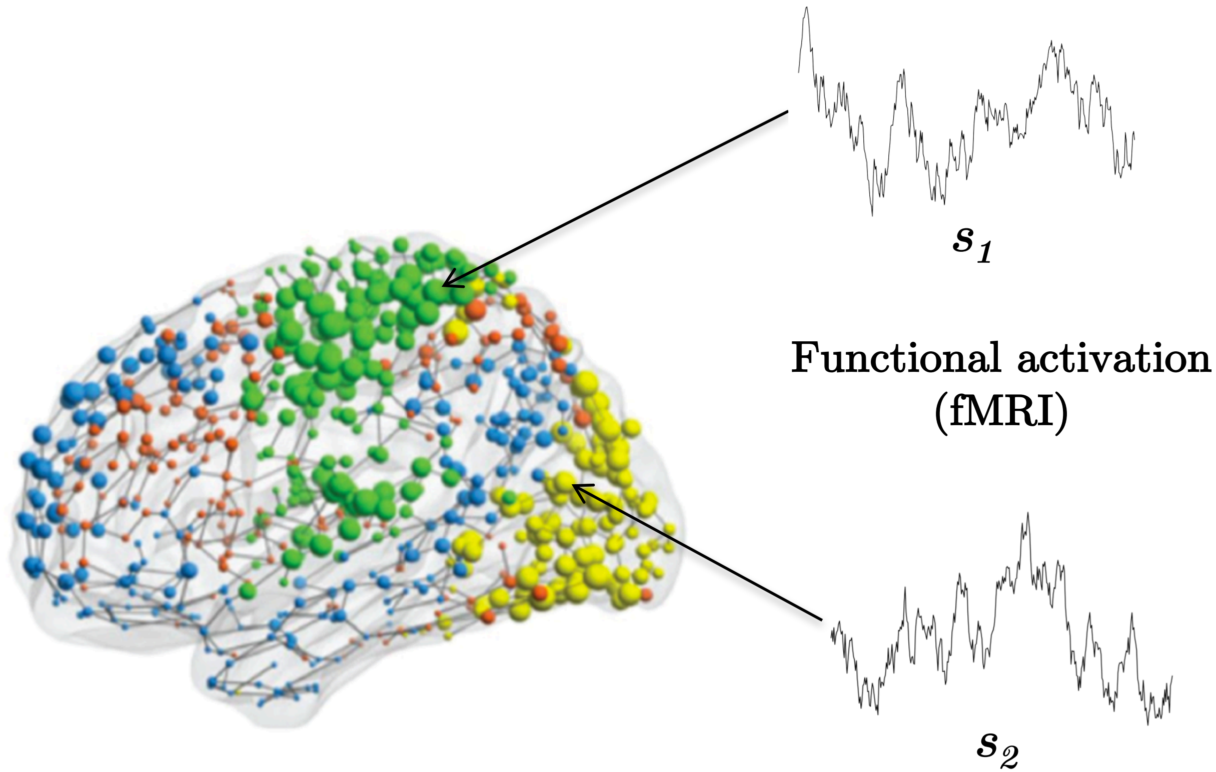


- Part 3: ConvNets for Variable Graphs
  - *Graph learning problems**

- Conclusion

Problem setting

- Given fixed graph(s) G, and a set of signals s_k on G to be analyzed with ConvNets:

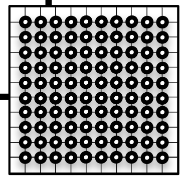


Brain connectivity network (sMRI)
Fixed graph G

Spectral ConvNets

Signal s_k: Image

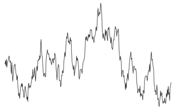
Part 1:
(Euclidean)
ConvNets



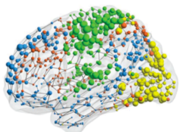
Classification

New data domain

Signal s_k: fMRI



Part 2:
Non-Euclidean
ConvNets



Fixed graph G

Classification

What do we need to generalize?

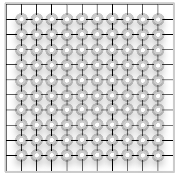
- Convolution and downsampling must be generalized from Euclidean grid domains to graphs. How?

  - Spectral graph theory allows to redefine convolution in the context of graphs with Fourier analysis.

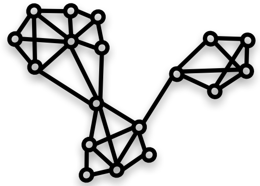
  - Graph theory provide graph clustering techniques to reformulate downsampling for graphs.

Outline

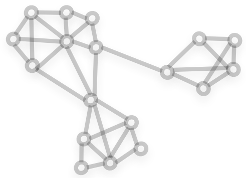
- Part 1: Euclidean ConvNets
  - *Architecture*
  - *Non-Euclidean data*



- **Part 2: Spectral ConvNets for Fixed Graphs**



  - **Spectral Graph Theory**
    - *Graph convolution: Graphs, Fourier modes, Fourier analysis, convolution*
    - *Graph coarsening: Graph clustering, HEM, binary tree indexing*
  - Spectral ConvNets
    - *SplineNets*
    - *ChebNets* [NIPS'16]*
    - *GraphConvNets*
    - *CayleyNets**
    - *Multiple fixed graphs* [NIPS'17]*
- Part 3: ConvNets for Variable Graphs
  - *Graph learning problems**



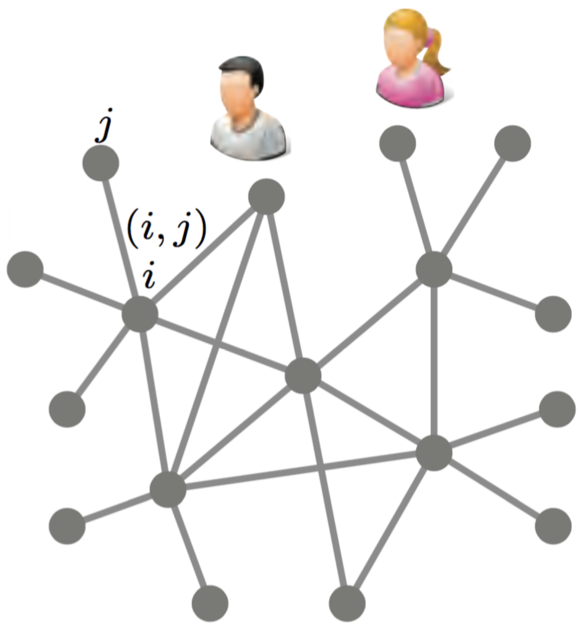
- Conclusion

Graphs

- **Graph** $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- **Vertices** $\mathcal{V} = \{1, \ldots, n\}$
- **Edges** $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$
- **Vertex weights** $b_i > 0$ for $i \in \mathcal{V}$
- **Edge weights** $a_{ij} \geq 0$ for $(i,j) \in \mathcal{E}$
- **Vertex fields** $L^2(\mathcal{V}) = \{f : \mathcal{V} \to \mathbb{R}^h\}$

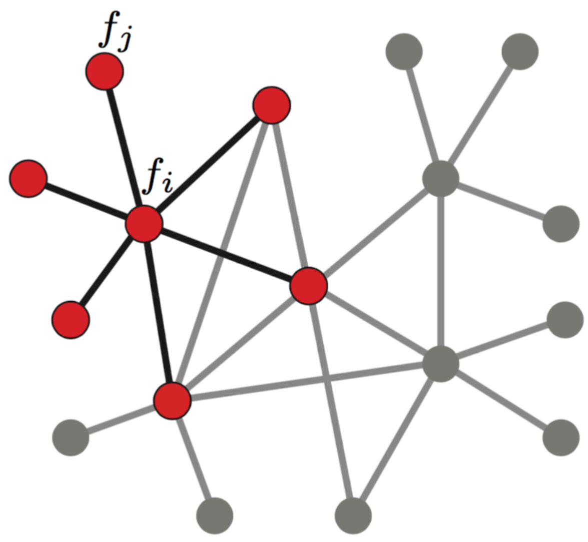
  Represented as $\mathbf{f} = (f_1, \ldots, f_n)$
- **Hilbert space with inner product**

$$\langle f, g \rangle_{L^2(\mathcal{V})} = \textstyle\sum_{i \in \mathcal{V}} a_i f_i g_i$$



[9] Chung 1994

Graph Laplacian



- Laplacian operator $\Delta : L^2(\mathcal{V}) \to L^2(\mathcal{V})$

$$(\Delta f)_i = \frac{1}{b_i} \sum_{j:(i,j)\in\mathcal{E}} a_{ij}(f_i - f_j)$$

difference between f and its local average (2$^{\text{nd}}$ derivative on graphs)

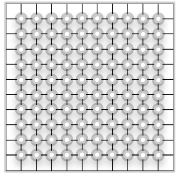
- Core operator in spectral graph theory.

- Represented as a positive semi-definite $n \times n$ matrix
  - Unnormalized Laplacian $\quad \boldsymbol{\Delta} = \mathbf{D} - \mathbf{A}$
  - Normalized Laplacian $\quad \boldsymbol{\Delta} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$
  - Random walk Laplacian $\quad \boldsymbol{\Delta} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$

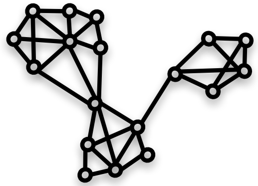
  where $\mathbf{A} = (a_{ij})$ and $\mathbf{D} = \text{diag}(\sum_{j\neq i} a_{ij})$

Outline

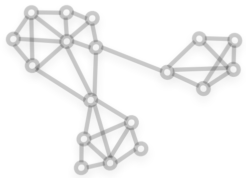
- Part 1: Euclidean ConvNets
  - *Architecture*
  - *Non-Euclidean data*



- **Part 2: Spectral ConvNets for Fixed Graphs**
  - **Spectral Graph Theory**
    - *Graph convolution: Graphs, Fourier modes, Fourier analysis, convolution*
    - *Graph coarsening: Graph clustering, HEM, binary tree indexing*
  - Spectral ConvNets
    - *SplineNets*
    - *ChebNets* [NIPS'16]*
    - *GraphConvNets*
    - *CayleyNets**
    - *Multiple fixed graphs* [NIPS'17]*



- Part 3: ConvNets for Variable Graphs
  - *Graph learning problems**



- Conclusion

Spectral decomposition

- A Laplacian of a graph of n vertices admits n eigenvectors:

$$\mathbf{\Delta}\phi_k = \lambda_k \phi_k, \qquad k = 1, 2, \ldots$$

  - Eigenvectors are real and orthonormal (self-adjointness) $\langle \phi_k, \phi_{k'} \rangle_{L^2(\mathcal{V})} = \delta_{kk'}$
  - Eigenvalues are non-negative (positive-semidefiniteness) $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$
  - Laplacian eignenvectors are also called Fourier basis functions/modes.

- Eigendecomposition of graph Laplacian:

$$\mathbf{\Delta} = \mathbf{\Phi}^T \mathbf{\Lambda} \mathbf{\Phi}$$

where $\mathbf{\Phi} = (\phi_1, \ldots, \phi_n)$ and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$

Interpretation

- Find the smoothest orthonormal basis $\boldsymbol{\Phi} = (\phi_1, \ldots, \phi_n)$ on a graph

$$\min_{\boldsymbol{\phi}_k} E_{\mathrm{Dir}}(\boldsymbol{\phi}_k) \quad \text{s.t.} \quad \|\boldsymbol{\phi}_k\| = 1, \quad k = 2, 3, \ldots n$$
$$\boldsymbol{\phi}_k \perp \mathrm{span}\{\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_{k-1}\}$$

where E_{Dir} is the Dirichlet energy = measure of smoothness of a function

$$E_{\mathrm{Dir}}(\mathbf{f}) = \mathbf{f}^\top \boldsymbol{\Delta} \mathbf{f}$$

- Solution: first n Laplacian eigenvectors

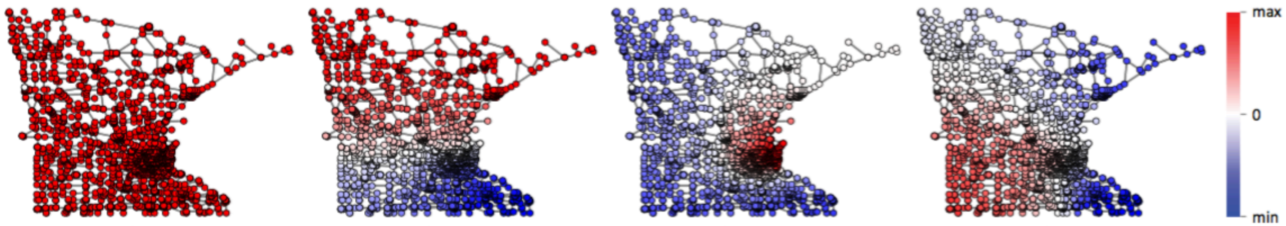
$$\min_{\boldsymbol{\Phi} \in \mathbb{R}^{n \times n}} \underbrace{\mathrm{trace}(\boldsymbol{\Phi}^\top \boldsymbol{\Delta} \boldsymbol{\Phi})}_{\|\boldsymbol{\Phi}\|_{\mathcal{G}} \text{ Dirichlet norm}} \quad \text{s.t.} \quad \boldsymbol{\Phi}^\top \boldsymbol{\Phi} = \mathbf{I}$$

Fourier modes

- Euclidean domain:

First eigenvectors of 1D Euclidean Laplacian = standard Fourier basis

- Graph domain:



ϕ_0 ϕ_1 ϕ_2 ϕ_3

First Laplacian eigenvectors of a graph

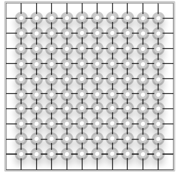
Lap eigenvectors related to graph geometry
(s.a. communities, hubs, etc), spectral clustering[10]

[10] Von Luxburg 2007

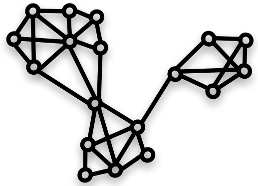
Outline

➢ Part 1: Euclidean ConvNets

- *Architecture*
- *Non-Euclidean data*



➢ **Part 2: Spectral ConvNets for Fixed Graphs**



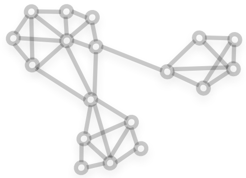
➢ **Spectral Graph Theory**

- *Graph convolution: Graphs, Fourier modes, Fourier analysis, convolution*
- *Graph coarsening: Graph clustering, HEM, binary tree indexing*

➢ Spectral ConvNets

- *SplineNets*
- *ChebNets* [NIPS'16]*
- *GraphConvNets*
- *CayleyNets**
- *Multiple fixed graphs* [NIPS'17]*

➢ Part 3: ConvNets for Variable Graphs



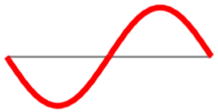
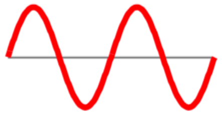
- *Graph learning problems**

➢ Conclusion

Euclidean Fourier analysis

- **A function** $f : [-\pi, \pi] \to \mathbb{R}$ **can be written as Fourier series:**

$$f(x) = \sum_{k \geq 0} \underbrace{\frac{1}{2\pi} \int_{-\pi}^{\pi} f(x') e^{-ikx'} dx'}_{\hat{f}_k = \langle f, e^{-ikx} \rangle_{L^2([-\pi,\pi])}} e^{-ikx}$$

f  $= \hat{f}_1$  $+ \hat{f}_2$  $+ \hat{f}_3$  $+ \ldots$

- **Fourier basis** e^{-ikx} $=$ **Laplace-Beltrami eigenfunctions:**

$$-\Delta \phi_k = k^2 \phi_k$$

$$\begin{cases} \phi_k & = & \text{Fourier mode} \\ k & = & \text{frequency of Fourier mode} \end{cases}$$

Fourier analysis on graphs

- A function $f : \mathcal{V} \to \mathbb{R}$ can be written as Fourier series[11]:

$$f_i = \sum_{k=1}^{n} \underbrace{\langle f, \phi_k \rangle_{L^2(\mathcal{V})}}_{\hat{f}_k} \phi_{k,i}$$

- $\hat{f}_k$ is the k-th graph Fourier coefficient.

- In matrix-vector notation, with the n x n Fourier matrix $\mathbf{\Phi} = [\phi_1, ..., \phi_n]$

$$\hat{\mathbf{f}} = \mathbf{\Phi}^\top \mathbf{f} \text{ and } \mathbf{f} = \mathbf{\Phi}\hat{\mathbf{f}}$$

Fourier transform

Inverse Fourier transform

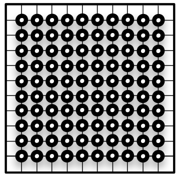
- Graph Fourier basis ϕ_k = Laplacian eigenvectors :

$$\mathbf{\Delta}\phi_k = \lambda_k \phi_k \quad \text{with} \quad \begin{cases} \phi_k & = & \text{graph Fourier mode} \\ \lambda_k & = & \text{(square) frequency} \end{cases}$$

[11] Hammond, Vandergheynst, Gribonval, 2011

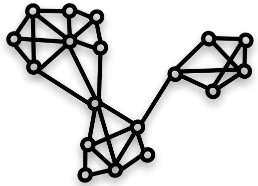
Outline

- Part 1: Euclidean ConvNets



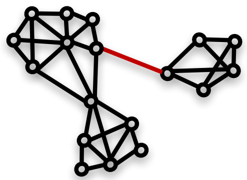
  - *Architecture*
  - *Non-Euclidean data*

- **Part 2: Spectral ConvNets for Fixed Graphs**



  - **Spectral Graph Theory**
    - *Graph convolution: Graphs, Fourier modes, Fourier analysis, convolution*
    - *Graph coarsening: Graph clustering, HEM, binary tree indexing*
  - Spectral ConvNets
    - *SplineNets*
    - *ChebNets* [NIPS'16]*
    - *GraphConvNets*
    - *CayleyNets**
    - *Multiple fixed graphs* [NIPS'17]*

- Part 3: ConvNets for Variable Graphs



  - *Graph learning problems**

- Conclusion

Convolution in Euclidean space

- Given two functions $f, g : [-\pi, \pi] \to \mathbb{R}$ their convolution is a function

$$(f \star g)(x) = \int_{-\pi}^{\pi} f(x')g(x - x')dx'$$

  - Shift-invariance: $f(x - x_0) \star g(x) = (f \star g)(x - x_0)$
  - Convolution theorem: Convolution can be computed in the Fourier domain as

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

  - Efficient computation using FFT: O($n$log$n$)

Convolution in discrete Euclidean space

- **Convolution of two vectors** $\mathbf{f} = (f_1, \dots, f_n)^\top$ and $\mathbf{g} = (g_1, \dots, g_n)^\top$

$$(\mathbf{f} \star \mathbf{g})_i = \sum_m g_{(i-m) \bmod n} \cdot f_m$$

$$\mathbf{f} \star \mathbf{g} = \underbrace{\begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix}}_{\substack{\text{Circulant matrix} \\ \text{diagonalised by Fourier basis (Toeplitz)}}} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

$$= \mathbf{\Phi} \begin{bmatrix} \hat{g}_1 & & \\ & \ddots & \\ & & \hat{g}_n \end{bmatrix} \mathbf{\Phi}^\top \mathbf{f} = \mathbf{\Phi}(\mathbf{\Phi}^\top \mathbf{g} \circ \mathbf{\Phi}^\top \mathbf{f})$$

Convolution on graphs

- Spectral convolution of $f, g \in L^2(\mathcal{V})$ can be defined by analogy[11]

$$(f \star g)_i = \underbrace{\sum_{k \geq 1} \underbrace{\langle f, \phi_k \rangle_{L^2(\mathcal{V})} \langle g, \phi_k \rangle_{L^2(\mathcal{V})}}_{\text{product in the Fourier domain}} \phi_{k,i}}_{\text{inverse Fourier transform}}$$

- In matrix-vector notation

$$\begin{aligned} \mathbf{f} \star \mathbf{g} &= \boldsymbol{\Phi} \left(\boldsymbol{\Phi}^\top \mathbf{g} \circ \boldsymbol{\Phi}^\top \mathbf{f} \right) \\ &= \underbrace{\boldsymbol{\Phi} \, \mathrm{diag}(\hat{g}_1, \ldots, \hat{g}_n) \boldsymbol{\Phi}^\top}_{\mathbf{G}} \mathbf{f} \\ &= \boldsymbol{\Phi} \hat{g}(\boldsymbol{\Lambda}) \boldsymbol{\Phi}^\top \mathbf{f} = \hat{g}(\boldsymbol{\Phi} \boldsymbol{\Lambda} \boldsymbol{\Phi}^\top) \mathbf{f} = \hat{g}(\boldsymbol{\Delta}) \mathbf{f} \end{aligned}$$

  - Not shift-invariant (G has no circulant structure)
  - Filter coefficients depend on basis $\phi_1, \ldots, \phi_n$
  - Expensive computation (no FFT): $O(n^2)$

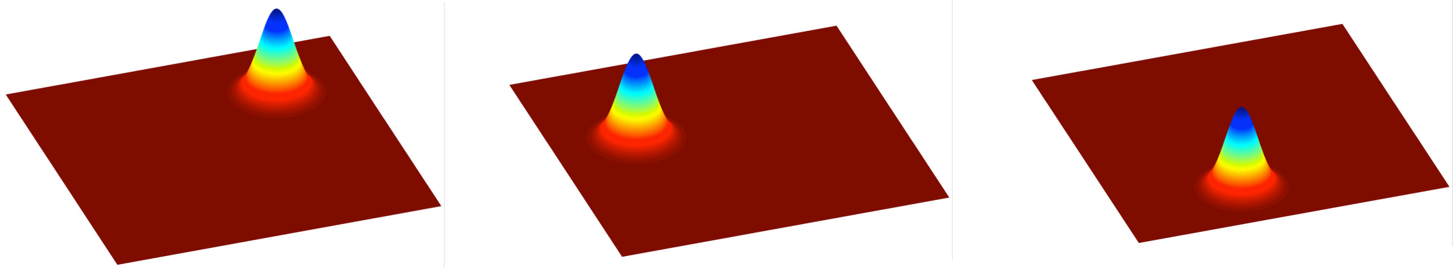
[11] Hammond, Vandergheynst, Gribonval, 2011

No shift invariance on graphs

- A signal f on graph can be translated to vertex i:

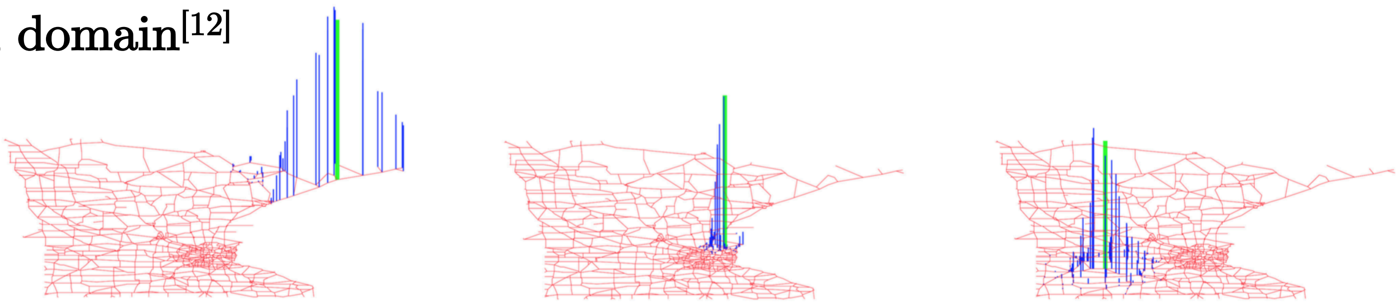
$$T_i \mathbf{f} = \mathbf{f} \star \boldsymbol{\delta}_i$$

- Euclidean domain



(a) $T_s f$ (b) $T_{s'} f$ (c) $T_{s''} f$

- Graph domain[12]

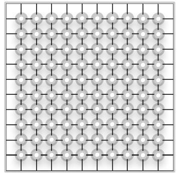


(d) $T_i f$ (e) $T_{i'} f$ (f) $T_{i''} f$

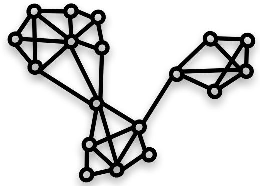
[12] Shuman et al. 2016

Outline

- Part 1: Euclidean ConvNets
  - Architecture
  - Non-Euclidean data

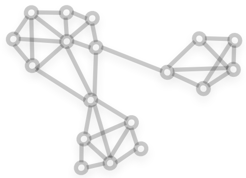


- **Part 2: Spectral ConvNets for Fixed Graphs**



  - **Spectral Graph Theory**
    - *Graph convolution: Graphs, Fourier modes, Fourier analysis, convolution*
    - *Graph coarsening: Graph clustering, HEM, binary tree indexing*
  - Spectral ConvNets
    - *SplineNets*
    - *ChebNets* [NIPS'16]*
    - *GraphConvNets*
    - *CayleyNets**
    - *Multiple fixed graphs* [NIPS'17]*

- Part 3: ConvNets for Variable Graphs
  - *Graph learning problems**



- Conclusion

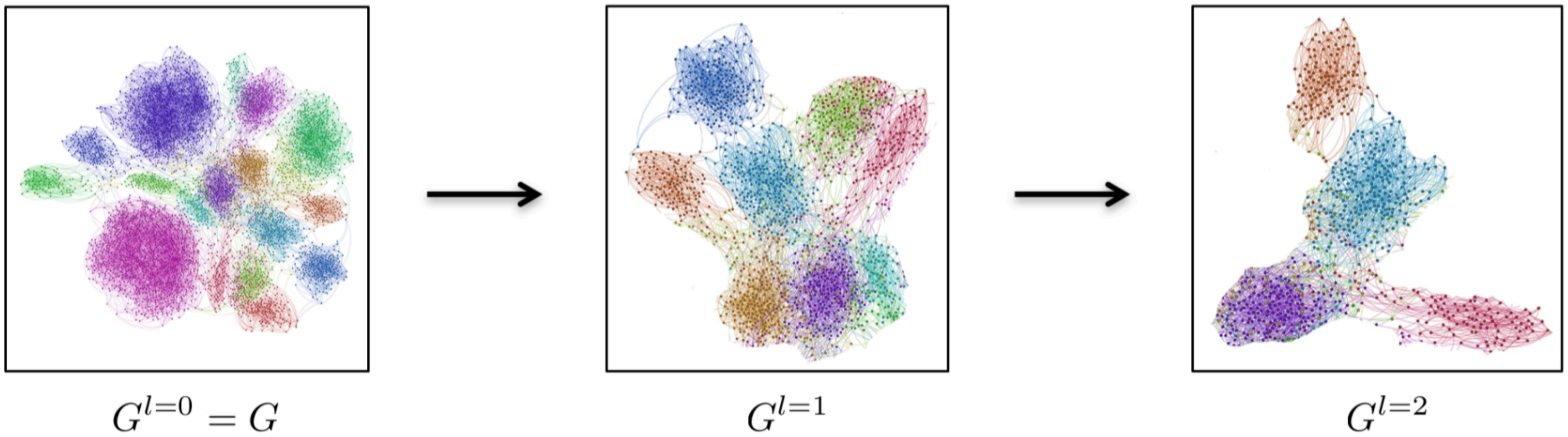
Graph dowsampling

- Goals:
  - Pool similar local features (max/average pooling).
  - Series of pooling layers create invariance to global geometric deformations.

- Challenges:
  - Design a multi-scale coarsening algorithm that preserves non-linear graph structures.
  - How to make graph pooling fast?

Graph dowsampling

- Graph downsampling $\Leftrightarrow$ graph coarsening $\Leftrightarrow$ graph partitioning: Decompose G into smaller meaningful clusters.



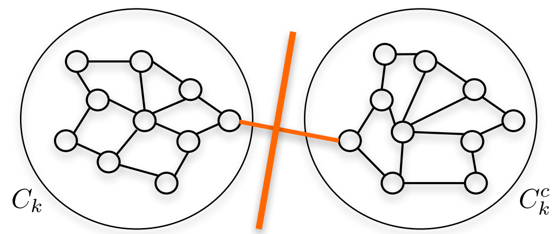
- Graph partitioning is NP-hard $\Rightarrow$ Approximation

Balanced cuts

- Powerful combinatorial graph partitioning models:

  - Normalized cut[13]

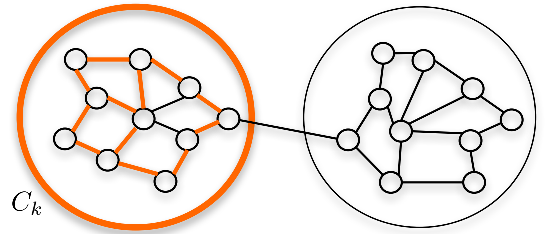
$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} \frac{\mathrm{Cut}(C_k, C_k^c)}{\mathrm{Vol}(C_k)}$$



Partitioning by min edge cuts.

  - Normalized association

$$\max_{C_1,\ldots,C_K} \sum_{k=1}^{K} \frac{\mathrm{Assoc}(C_k)}{\mathrm{Vol}(C_k)}$$



Partitioning by max vertex matching.

where $\mathrm{Cut}(A,B) := \sum_{i\in A, j\in B} a_{ij}$, $\mathrm{Assoc}(A) := \sum_{i\in A, i\in B} a_{ij}$, $\mathrm{Vol}(A) := \sum_{i\in A, j\in B} d_i$, and $d_i := \sum_{j\in V} a_{ij}$.

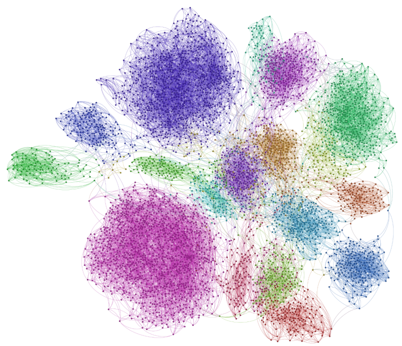
- Both models are equivalent, but lead to different algorithms.

[13] Shi, Malik, 2000

Balanced cuts

- Balanced cuts are NP-hard ⇒ most popular approximation techniques focus on linear spectral relaxation (eigenproblem with global solution).

- Graph geometry are generally not linear ⇒ Graclus[14] algorithm computes non-linear clusters that locally maximize the Normalized Association.

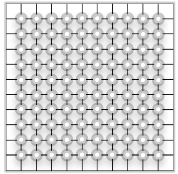


- Graclus algorithm offers a control of the coarsening ratio of ≈ 2 (like image grid) using heavy-edge matching[15].

[14] Dhillon, Guan, Kulis 2007
[15] Karypis, Kumar 1995

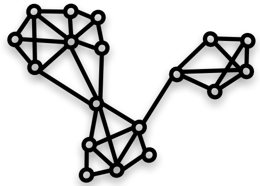
Outline

- Part 1: Euclidean ConvNets



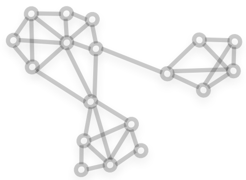
  - *Architecture*
  - *Non-Euclidean data*

- **Part 2: Spectral ConvNets for Fixed Graphs**



  - **Spectral Graph Theory**
    - *Graph convolution: Graphs, Fourier modes, Fourier analysis, convolution*
    - *Graph coarsening: Graph clustering, HEM, binary tree indexing*
  - Spectral ConvNets
    - *SplineNets*
    - *ChebNets* [NIPS'16]*
    - *GraphConvNets*
    - *CayleyNets**
    - *Multiple fixed graphs* [NIPS'17]*

- Part 3: ConvNets for Variable Graphs

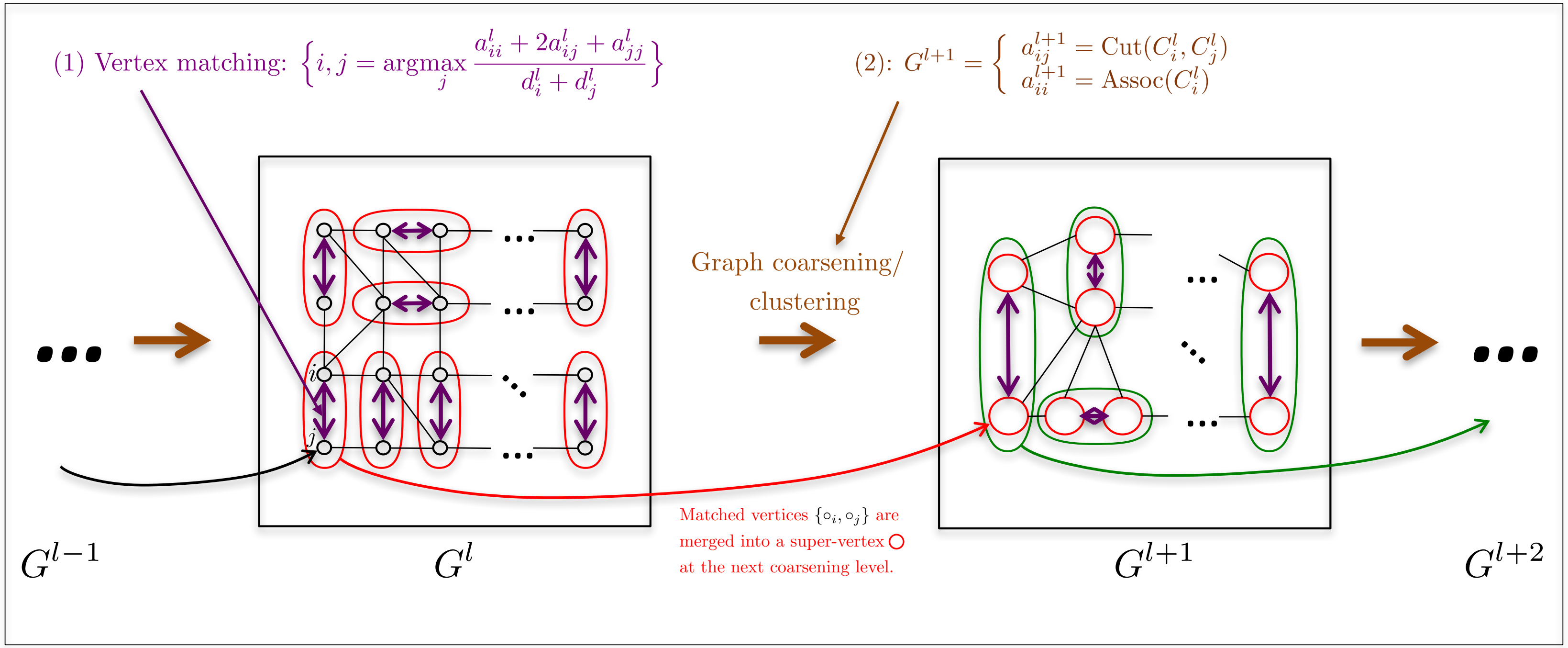


  - *Graph learning problems**

- Conclusion

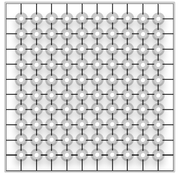
Heavy-Edge Matching (HEM)

- HEM proceeds by two successive steps, vertex matching and graph coarsening (that guarantees a local solution of Norm assoc):

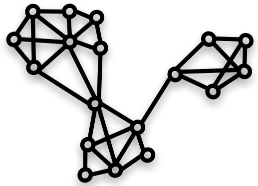


Outline

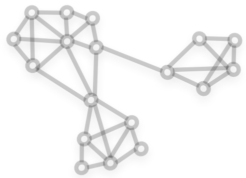
- Part 1: Euclidean ConvNets
  - *Architecture*
  - *Non-Euclidean data*



- **Part 2: Spectral ConvNets for Fixed Graphs**
  - **Spectral Graph Theory**
    - *Graph convolution: Graphs, Fourier modes, Fourier analysis, convolution*
    - *Graph coarsening: Graph clustering, HEM, binary tree indexing*
  - Spectral ConvNets
    - *SplineNets*
    - *ChebNets* [NIPS'16]*
    - *GraphConvNets*
    - *CayleyNets**
    - *Multiple fixed graphs* [NIPS'17]*



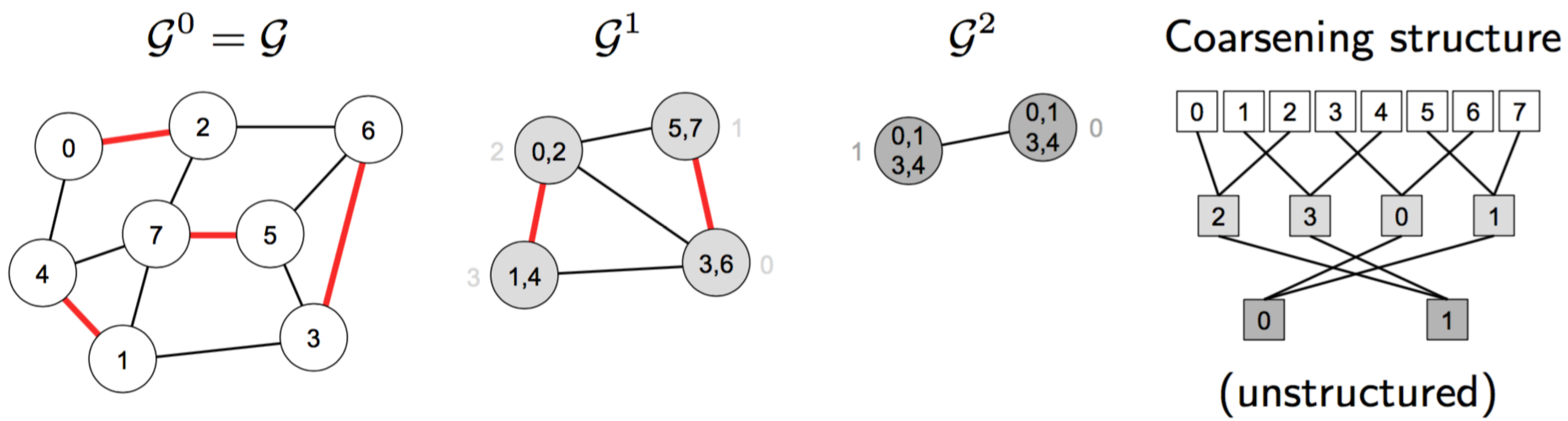
- Part 3: ConvNets for Variable Graphs
  - *Graph learning problems**



- Conclusion

Unstructured pooling

- Sequence of coarsened graphs produced by HEM:

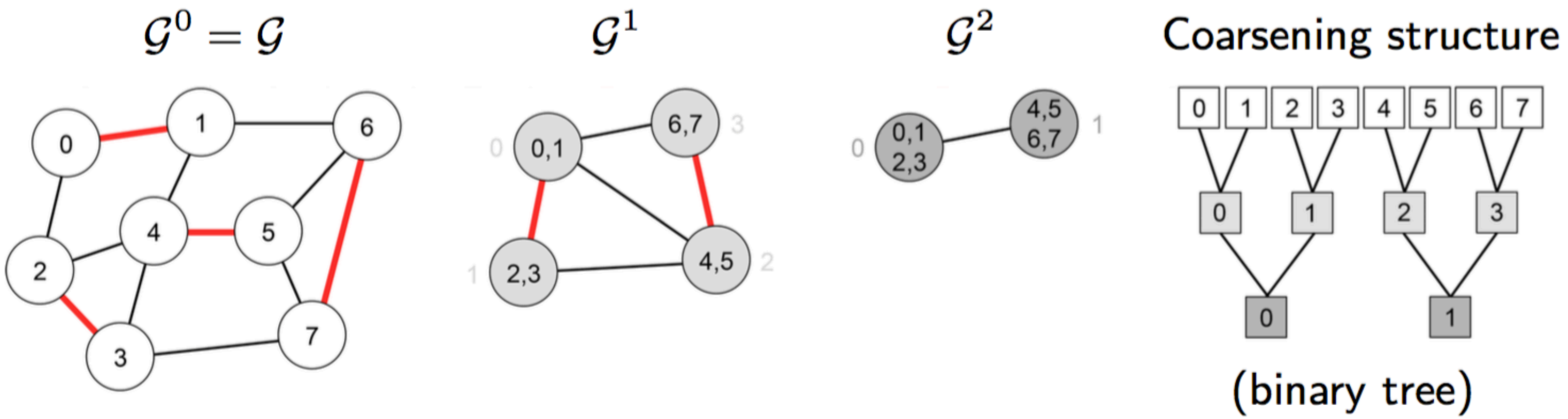


☹ Stores a table of indices for graph and all its coarsened versions

☹ Computationally inefficient

Fast graph pooling

- Structured pooling[18]: Arrangement of the node indexing such that adjacent nodes are hierarchically merged at the next coarser level.

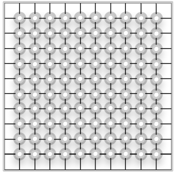


☺ As efficient as 1D-Euclidean grid pooling.

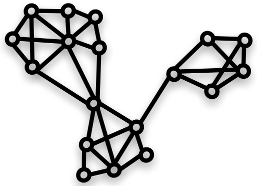
[18] Defferrard, Bresson, Vandergheynst 2016

Outline

- Part 1: Euclidean ConvNets
  - *Architecture*
  - *Non-Euclidean data*

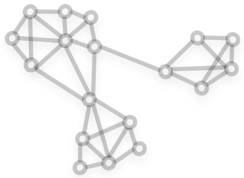


- **Part 2: Spectral ConvNets for Fixed Graphs**



  - Spectral Graph Theory
    - *Graph convolution*
    - *Graph coarsening*
  - **Spectral ConvNets**
    - *SplineNets*
    - *ChebNets* [NIPS'16]*
    - *GraphConvNets*
    - *CayleyNets**
    - *Multiple fixed graphs* [NIPS'17]*

- Part 3: ConvNets for Variable Graphs
  - *Graph learning problems**

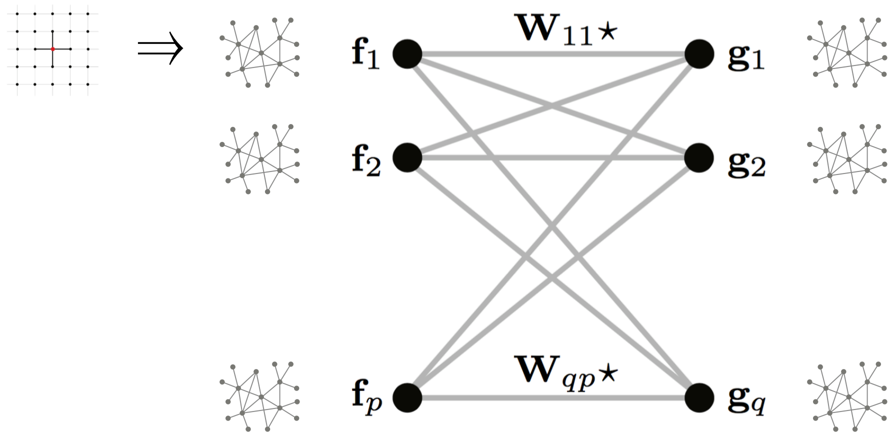


- Conclusion

Vanilla spectral graph ConvNets

- **Graph convolutional layer:**

$$\begin{aligned} \mathbf{f}_l &= l\text{-th data feature on graphs, } \dim(\mathbf{f}_l) = n \times 1 \\ \mathbf{g}_l &= l\text{-th feature map, } \dim(\mathbf{g}_l) = n \times 1 \end{aligned}$$



Conv. layer $\quad \mathbf{g}_l = \xi\left(\sum_{l'=1}^{p} \mathbf{W}_{l,l'} \star \mathbf{f}_{l'}\right)$

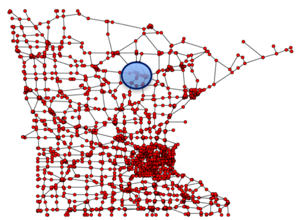
Activation, e.g. $\quad \xi(x) = \max\{x, 0\}$ $\quad$ rectified linear unit (ReLU)

[16] Bruna, Zaremba, Szlam, LeCun 2014

Spectral graph convolution

- Convolutional layer in the spatial domain:

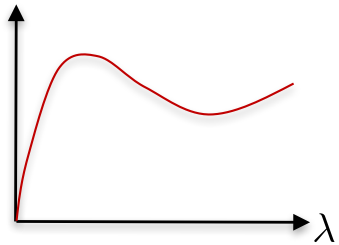
$$\mathbf{g}_l = \xi\left(\sum_{l'=1}^{p} \mathbf{W}_{l,l'} \star \mathbf{f}_{l'}\right),$$



where $\mathbf{W}_{l,l'}$ = matrix of graph spatial filter,

can also be expressed in the spectral domain (using $\mathbf{g} \star \mathbf{f} = \mathbf{\Phi}\,\hat{g}(\mathbf{\Lambda})\mathbf{\Phi}^\top \mathbf{f}$)

$$\mathbf{g}_l = \xi\left(\sum_{l'=1}^{p} \mathbf{\Phi}\hat{\mathbf{W}}_{l,l'}\mathbf{\Phi}^\top \mathbf{f}_{l'}\right),$$



where $\hat{\mathbf{W}}_{l,l'}$ = n x n diagonal matrix of graph spectral filter.

We will denote the spectral filter without the hat symbol, i.e. $\mathbf{W}_{l,l'}$

Vanilla spectral graph ConvNets

- Series of spectral convolutional layers:

$$\mathbf{g}_l^{(k)} = \xi\left(\sum_{l'=1}^{q^{(k-1)}} \mathbf{\Phi}\mathbf{W}_{l,l'}^{(k)}\mathbf{\Phi}^\top \mathbf{g}_{l'}^{(k-1)}\right),$$

with spectral coefficients $\mathbf{W}_{l,l'}^{(k)}$ to be learned at each layer.

☺ First spectral graph CNN architecture

☹ No guarantee of spatial localization of filters

☹ $O(n)$ parameters per layer

☹ $O(n^2)$ computation of forward and inverse Fourier transforms $\phi, \phi^\top$ (no FFT on graphs)

☹ Filters are basis-dependent ⇒ does not generalize across graphs

[16] Bruna, Zaremba, Szlam, LeCun 2014

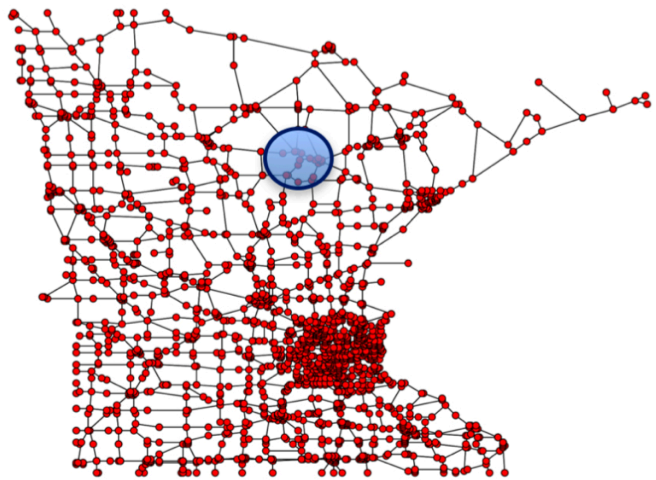
Spatial localization and spectral smoothness[17]

- **In the Euclidean setting (by Parseval's identity)**

$$\int_{-\infty}^{+\infty} |x|^{2k} |w(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{w}(\lambda)}{\partial \lambda^k} \right|^2 d\lambda$$

$\Rightarrow$ **Localization in space = smoothness in frequency domain**

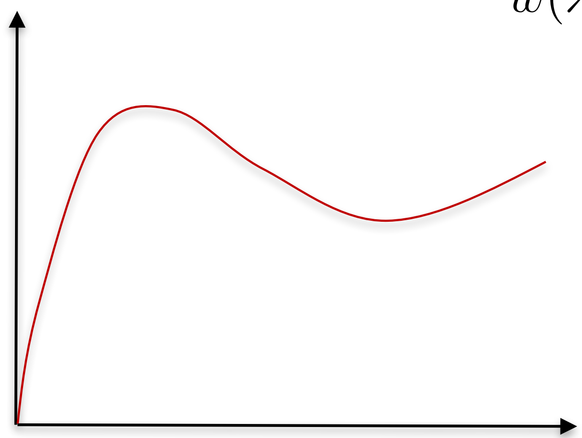
$w(x)$



Spatial localization

$\Leftrightarrow$

$\hat{w}(\lambda)$



λ

Smooth spectral filter function

[17] Henaff, Bruna, LeCun 2015

Smooth parametric spectral filter

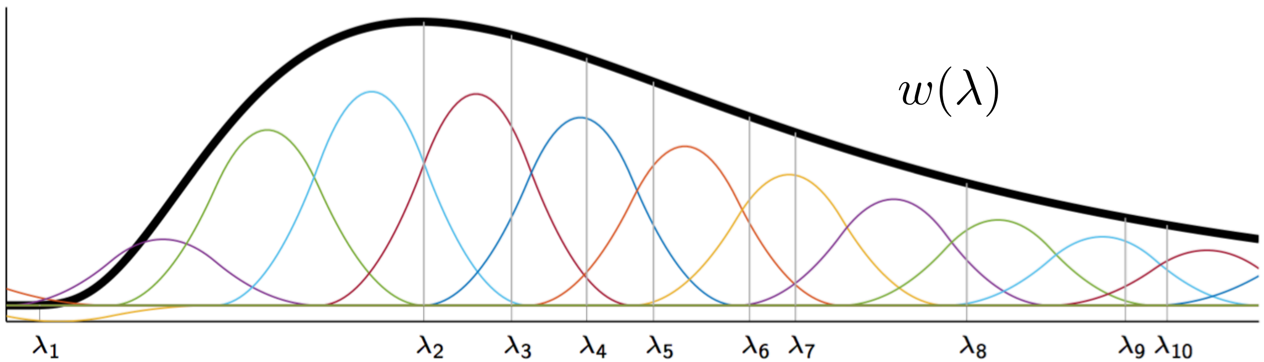
- **Parametrize the smooth spectral filter function** $w_(\lambda)$ **with a linear combination of smooth kernel functions** $\beta_1(\lambda), \ldots, \beta_r(\lambda)$ **, e.g. splines:**

$$w_{\boldsymbol{\alpha}}(\lambda) = \sum_{j=1}^{r} \alpha_j \beta_j(\lambda)$$

$$\Downarrow$$

$$w_{\boldsymbol{\alpha}}(\lambda_i) = \sum_{j=1}^{r} \alpha_j \beta_j(\lambda_i) = (\mathbf{B}\boldsymbol{\alpha})_i \qquad \Longrightarrow \qquad \mathbf{W} = \mathrm{Diag}(\mathbf{B}\boldsymbol{\alpha})$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r)^\top$ **is the vector of filter parameters**



[17] (Litman, Bronstein, 2014); Henaff, Bruna, LeCun 2015

SplineNets

- Series of parametric spectral convolutional layers:

$$\mathbf{g}_l^{(k)} = \xi \left(\sum_{l'=1}^{q^{(k-1)}} \mathbf{\Phi} \mathbf{W}_{l,l'}^{(k)} \mathbf{\Phi}^\top \mathbf{g}_{l'}^{(k-1)} \right),$$

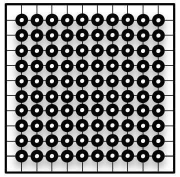
with smooth spectral parametric coefficients $\mathbf{W}_{l,l'}^{(k)}$ to be learned at each layer.

☺ Fast-decaying filters in space
☺ O(1) parameters per layer
☹ $O(n^2)$ computation of forward and inverse Fourier transforms ϕ, $\phi^\top$ (no FFT on graphs)
☹ Filters are basis-dependent ⇒ does not generalize across graphs

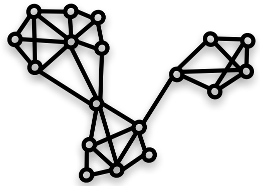
[17] Henaff, Bruna, LeCun 2015

Outline

➢ Part 1: Euclidean ConvNets
- *Architecture*
- *Non-Euclidean data*



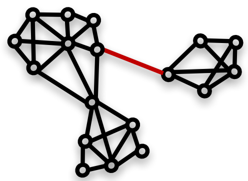
➢ **Part 2: Spectral ConvNets for Fixed Graphs**



➢ Spectral Graph Theory
- *Graph convolution*
- *Graph coarsening*

➢ **Spectral ConvNets**
- *SplineNets*
- *ChebNets* [NIPS'16]*
- *GraphConvNets*
- *CayleyNets**
- *Multiple fixed graphs* [NIPS'17]*

➢ Part 3: ConvNets for Variable Graphs
- *Graph learning problems**



➢ Conclusion

Spectral polynomial filters

- Represent smooth spectral functions with polynomials of Laplacian eigenvalues:

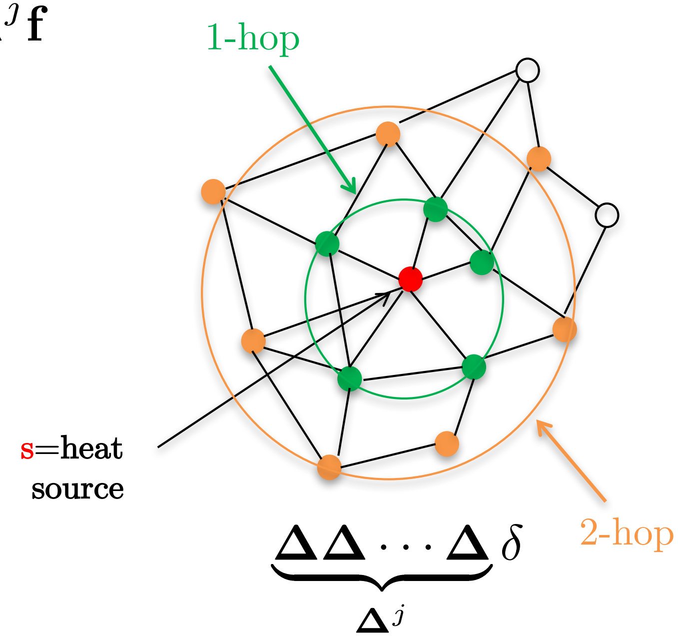
$$w_{\boldsymbol{\alpha}}(\lambda) = \sum_{j=0}^{r} \alpha_j \lambda^j$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r)^\top$ is the vector of filter parameters.

- Convolutional layer: Apply spectral filter to feature signal f

$$w_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} = \textstyle\sum_{j=0}^{r} \alpha_j \boldsymbol{\Delta}^j \mathbf{f}$$

- Key observation: Each Laplacian operation increases the support of a function by 1-hop $\Rightarrow$ Exact control the size of Laplacian-based filters.



[18] Defferrard, Bresson, Vandergheynst 2016

Linear complexity

- Application of the filter to a feature signal f

$$w_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} = \sum_{j=0}^{r} \alpha_j \boldsymbol{\Delta}^j \mathbf{f}$$

- Denote $\mathbf{X}_0 = \mathbf{f}$ and define $\mathbf{X}_1 = \boldsymbol{\Delta}\mathbf{X}_0 = \boldsymbol{\Delta}\mathbf{f}$ and the sequence $\mathbf{X}_j = \boldsymbol{\Delta}\mathbf{X}_{j-1}$

$$w_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} = \sum_{j=0}^{r} \alpha_j \mathbf{X}_j$$

- Two important observations:

  1. *No* need to compute the eigendecomposition of the Laplacian (ϕ,Λ).

  2. Observe that $\{X_j\}$ are generated by multiplication of a sparse matrix and a vector ⇒ Complexity is $O(Er)=O(n)$ for sparse graphs.

- Graph convolutional layers are GPU friendly.

Spectral graph ConvNets with polynomial filters

- Series of spectral convolutional layers

$$\mathbf{g}_l^{(k)} = \xi \left(\sum_{l'=1}^{q^{(k-1)}} \mathbf{\Phi} \mathbf{W}_{l,l'}^{(k)} \mathbf{\Phi}^\top \mathbf{g}_{l'}^{(k-1)} \right),$$

with spectral polynomial coefficients $\mathbf{W}_{l,l'}^{(k)}$ to be learned at each layer.

☺ Filters are exactly localized in r-hops support

☺ O(1) parameters per layer

☺ No computation of ϕ, ϕ$^\top$ ⇒ O(n) computational complexity (assuming sparsely-connected graphs)

☹ Unstable under coefficients perturbation (hard to optimize)

☹ Filters are basis-dependent ⇒ does not generalize across graphs

[18] Defferrard, Bresson, Vandergheynst 2016

Chebyshev polynomials

- **Graph convolution with (non-orthogonal) monomial basis** $1, x, x^2, x^3, \cdots$

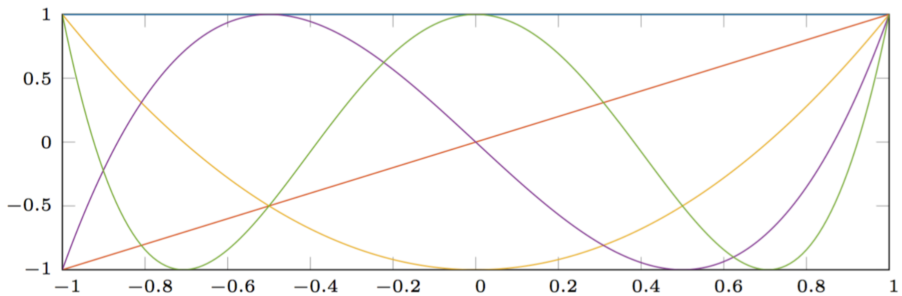
$$w_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} = \sum_{j=0}^{r} \alpha_j \boldsymbol{\Delta}^j \mathbf{f}$$

- **Graph convolution with (orthogonal) Chebyshev polynomials**

$$w_{\boldsymbol{\alpha}}(\tilde{\boldsymbol{\Delta}})\mathbf{f} = \sum_{j=0}^{r} \alpha_j T_j(\tilde{\boldsymbol{\Delta}})\mathbf{f}$$

- **Orthonormal on** $L^2([-1,+1])$ w.r.t. $\langle f, g\rangle = \int_{-1}^{+1} f(\tilde{\lambda}) g(\tilde{\lambda}) \frac{d\tilde{\lambda}}{\sqrt{1-\tilde{\lambda}^2}}$

- **Stable under perturbation of coefficients**



T_0, T_1, T_2, T_3

ChebNets

- Application of the filter with the scaled Laplacian $\tilde{\mathbf{\Delta}} = 2\lambda_n^{-1}\mathbf{\Delta} - \mathbf{I}$

$$w_{\boldsymbol{\alpha}}(\tilde{\mathbf{\Delta}})\mathbf{f} = \sum_{j=0}^{r} \alpha_j T_j(\tilde{\mathbf{\Delta}})\mathbf{f} = \sum_{j=0}^{r} \alpha_j \mathbf{X}^{(j)}$$

with

$$\begin{aligned} \mathbf{X}^{(j)} &= T_j(\tilde{\mathbf{\Delta}})\mathbf{f} \\ &= 2\tilde{\mathbf{\Delta}}\mathbf{X}^{(j-1)} - \mathbf{X}^{(j-2)}, \qquad \mathbf{X}^{(0)} = \mathbf{f}, \quad \mathbf{X}^{(1)} = \tilde{\mathbf{\Delta}}\mathbf{f} \end{aligned}$$

☺ Filters are exactly localized in r-hops support

☺ O(1) parameters per layer

☺ No computation of ϕ, $\phi^\top \Rightarrow O(n)$ computational complexity (assuming sparsely-connected graphs)

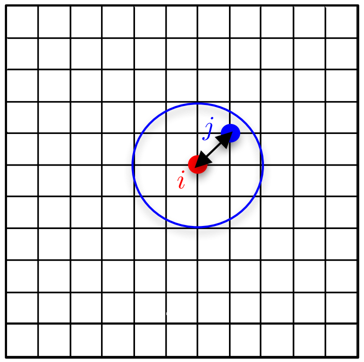
☺ Stable under coefficients perturbation

☹ Filters are basis-dependent ⇒ does not generalize across graphs

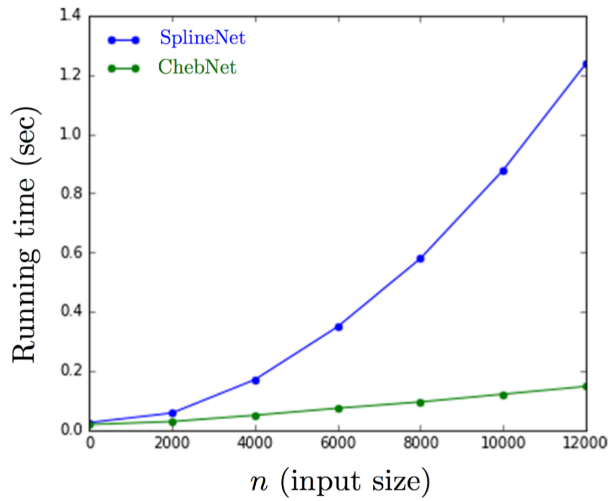
[18] Defferrard, Bresson, Vandergheynst 2016

Numerical experiments

Graph: a 8-NN graph of the Euclidean grid



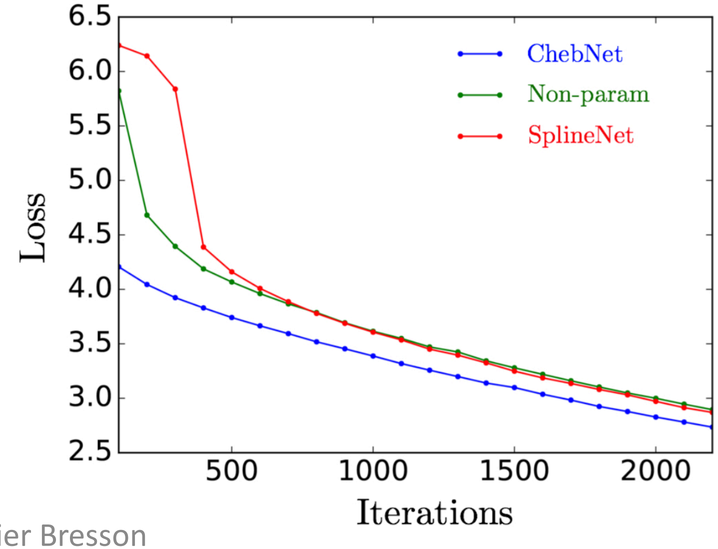
- **Running time**

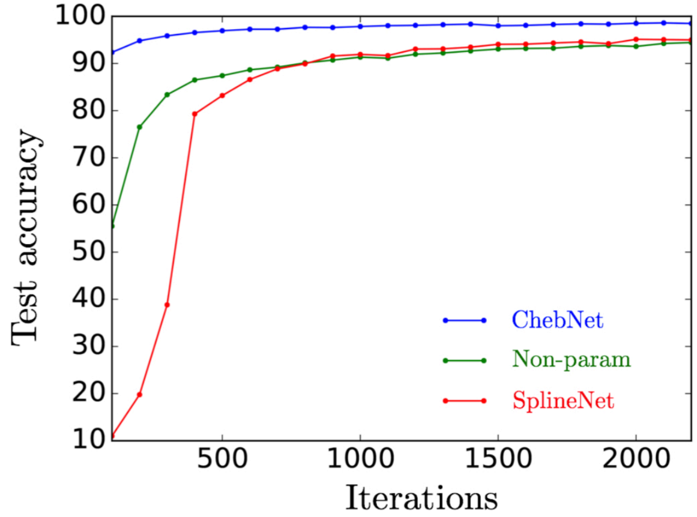


- **Accuracy**

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

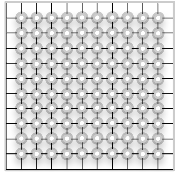
- **Optimization**



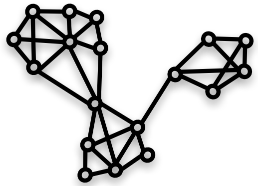


Outline

- Part 1: Euclidean ConvNets
  - *Architecture*
  - *Non-Euclidean data*

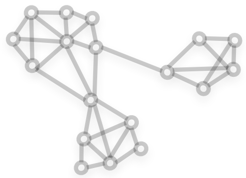


- **Part 2: Spectral ConvNets for Fixed Graphs**



  - Spectral Graph Theory
    - *Graph convolution*
    - *Graph coarsening*
  - **Spectral ConvNets**
    - *SplineNets*
    - *ChebNets* [NIPS'16]*
    - ***GraphConvNets***
    - *CayleyNets**
    - *Multiple fixed graphs* [NIPS'17]*

- Part 3: ConvNets for Variable Graphs
  - *Graph learning problems**



- Conclusion

Graph convolutional nets: simplified ChebNets

- Use Chebychev polynomials of degree *r=2* and assume $\lambda_n \approx 2$

$$\begin{aligned} w_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} &= \alpha_0\mathbf{f} + \alpha_1(\boldsymbol{\Delta} - \mathbf{I})\mathbf{f} \\ &= \alpha_0\mathbf{f} - \alpha_1\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}\mathbf{f} \end{aligned}$$

- Further constrain $\alpha = \alpha_0 = -\alpha_1$ to obtain a single-parameter filter

$$w_{\alpha}(\boldsymbol{\Delta})\mathbf{f} = \alpha(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2})\mathbf{f}$$

- Caveat: The eigenvalues of $\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ are now in [0,2]
  $\Rightarrow$ repeated application of the filter results in numerical instability

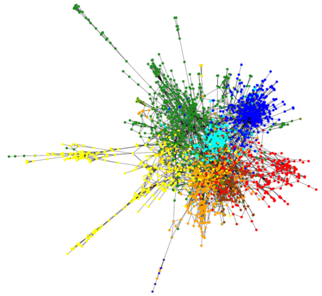
- Fix: Apply a renormalization

$$w_{\alpha}(\boldsymbol{\Delta})\mathbf{f} = \alpha\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{W}}\tilde{\mathbf{D}}^{-1/2}\mathbf{f}$$

with $\tilde{\mathbf{W}} = \mathbf{W} + \mathbf{I}$ and $\tilde{\mathbf{D}} = \mathrm{diag}(\sum_{j\neq i}\tilde{w}_{ij})$

[19] Kipf, Welling 2016

Example: Citation networks



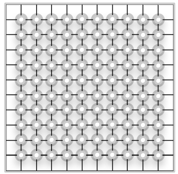
Method	Cora[1]	PubMed[2]
Manifold Regularization[3]	59.5%	70.7%
Semidefinite Embedding[4]	59.0%	71.1%
Label Propagation[5]	68.0%	63.0%
DeepWalk[6]	67.2%	65.3%
Planetoid[7]	75.7%	77.2%
Graph Convolutional Net[8]	**81.59%**	**78.72%**

Classification accuracy of different methods on citation network datasets

Monti et al. 2016; data: [1,2]Sen et al. 2008; methods: [3]Belkin et al. 2006; [4]Weston et al. 2012; [5]Zhu et al. 2003; [6]Perozzi et al. 2014; [7]Yang et al. 2016; [8]Kipf, Welling 2016

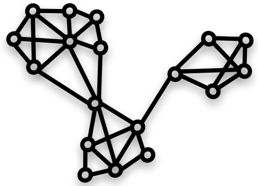
Outline

➢ Part 1: Euclidean ConvNets



- *Architecture*
- *Non-Euclidean data*

➢ **Part 2: Spectral ConvNets for Fixed Graphs**



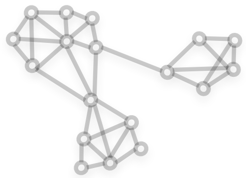
➢ Spectral Graph Theory

- *Graph convolution*
- *Graph coarsening*

➢ Spectral ConvNets

- *SplineNets*
- *ChebNets* [NIPS'16]*
- *GraphConvNets*
- *CayleyNets**
- *Multiple fixed graphs* [NIPS'17]*

➢ Part 3: ConvNets for Variable Graphs



- *Graph learning problems**

➢ Conclusion

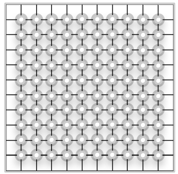
CayleyNets

Federico Monti (Universita della Svizzera Italiana)

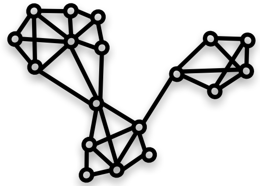
Deep Geometric Matrix Completion: a Geometric Deep Learning approach to Recommender Systems

Thursday, February 8, 2018, 10:10 - 10:50

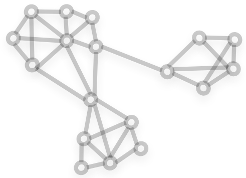
Outline



- Part 1: Euclidean ConvNets
  - *Architecture*
  - *Non-Euclidean data*



- **Part 2: Spectral ConvNets for Fixed Graphs**
  - Spectral Graph Theory
    - *Graph convolution*
    - *Graph coarsening*
  - Spectral ConvNets
    - *SplineNets*
    - *ChebNets* [NIPS'16]*
    - *GraphConvNets*
    - *CayleyNets**
    - *Multiple fixed graphs* [NIPS'17]*



- Part 3: ConvNets for Variable Graphs
  - *Graph learning problems**

- Conclusion

Spectral ConvNets for multiple fixed graphs

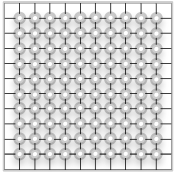
Federico Monti (Universita della Svizzera Italiana)

Deep Geometric Matrix Completion: a Geometric Deep Learning approach to Recommender Systems

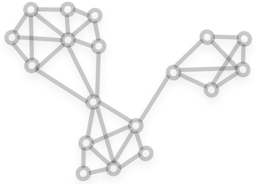
Thursday, February 8, 2018, 10:10 - 10:50

Outline

- Part 1: Euclidean ConvNets
  - *Architecture*
  - *Non-Euclidean data*

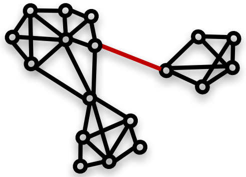


- Part 2: Spectral ConvNets for Fixed Graphs



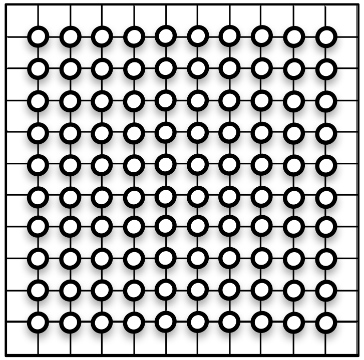
  - Spectral Graph Theory
    - *Graph convolution*
    - *Graph coarsening*
  - Spectral ConvNets
    - *SplineNets*
    - *ChebNets* [NIPS'16]*
    - *GraphConvNets*
    - *CayleyNets**
    - *Multiple fixed graphs* [NIPS'17]*

- **Part 3: ConvNets for Variable Graphs**
  - *Graph learning problems**

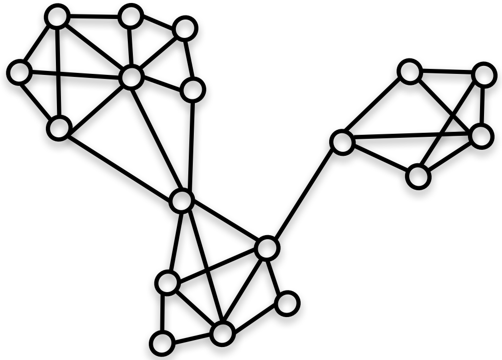


- Conclusion

Data domains



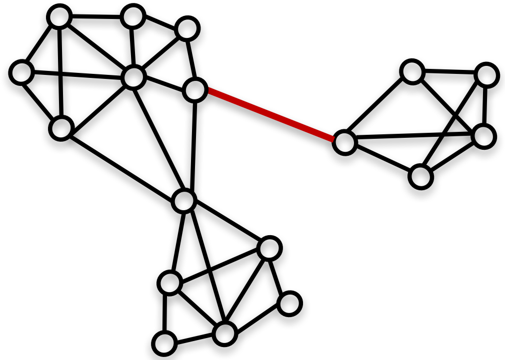
Euclidean
space/grid



Non-Euclidean
space/graph

Fixed domain

$\Rightarrow$



Change one single
edge

Variable domain

Standard
ConvNets

Spectral graph
ConvNets

Can we still use
spectral graph
ConvNets?

Spectral NNs offer rich
families of spectral filters

$\Rightarrow$

Are spectral filters
transferable?

Limitations of spectral NN techniques

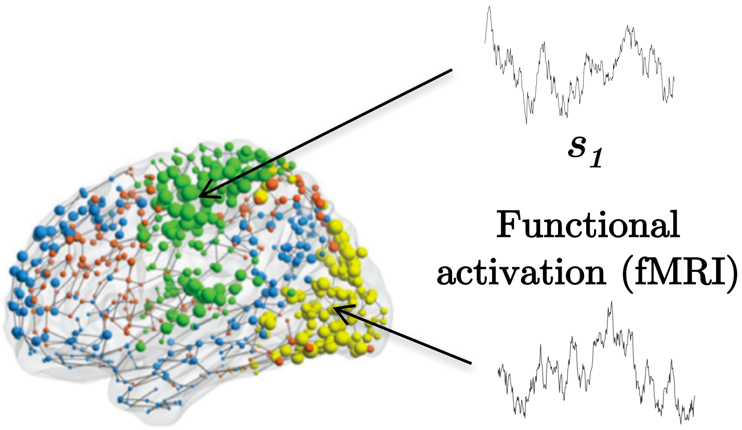
- Poor generalization to new/different graphs: Fourier modes are unstable under graph perturbations $\Rightarrow$ bad transfer/generalization of learned filters to new graphs:

$$\mathbf{\Phi}_{\mathcal{G}_1}\mathbf{W}\mathbf{\Phi}_{\mathcal{G}_1}^\top \mathbf{f} \neq \mathbf{\Phi}_{\mathcal{G}_2}\mathbf{W}\mathbf{\Phi}_{\mathcal{G}_2}^\top \mathbf{f}$$

- Aligning Fourier modes is hard, and does not guarantee good generalization.
- Directed graphs: Definition of directed graph Laplacian is unclear.
- Graphs with variable size: Spectral techniques work with fixed size graphs.

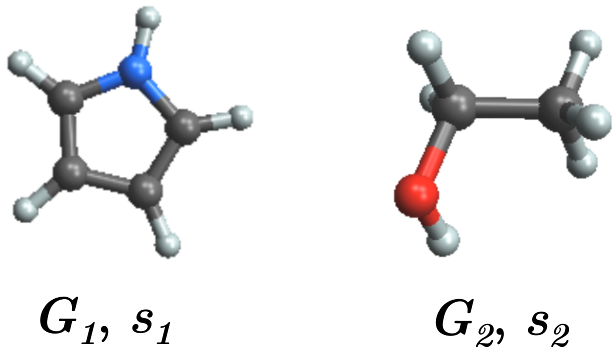
Problem setting

- **Spectral ConvNets:** Given fixed graph(s) G, and a set of signals s_k on G to be analyzed with ConvNets:



Brain connectivity network (sMRI)
Fixed graph G

- **ConvNets for arbitrary graphs:** Given a set of graphs G_k and signals s_k on G_k to be analyzed with ConvNets:

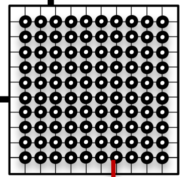


Molecules
Variable graphs G_k

Graph ConvNet architectures

Signal s_k: Image

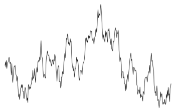
Part 1:
(Standard)
ConvNets



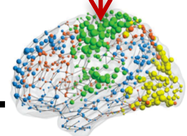
Classification

New data
domain

Signal s_k: fMRI



Part 2:
Spectral
ConvNets

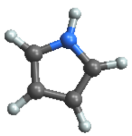


Classification

Fixed graph G

New data
domain

Signal s_k on graph G_k:
Molecule with atoms



Part 3:
Graph
ConvNets

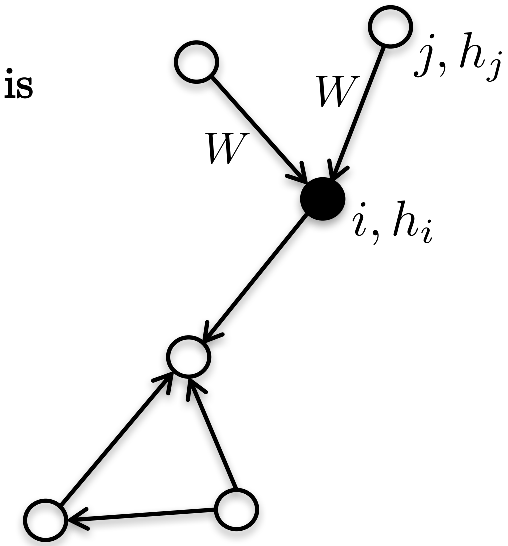
Classification

Variables graphs G_k

Graph neural networks

- Spatial NN technique to deal with arbitrary graphs[27].

- Minimal inner structures:
  - Invariant by vertex re-indexing (no graph matching is required)
  - Locality (only neighbors are considered)
  - Weight sharing (convolutional operations)
  - Independence w.r.t. graph size



$$h_i = f_{\text{GNN}}\left(\{h_j : j \to i\}\right)$$

- What instantiation of f ?

[27] Scarselli et-al 2009

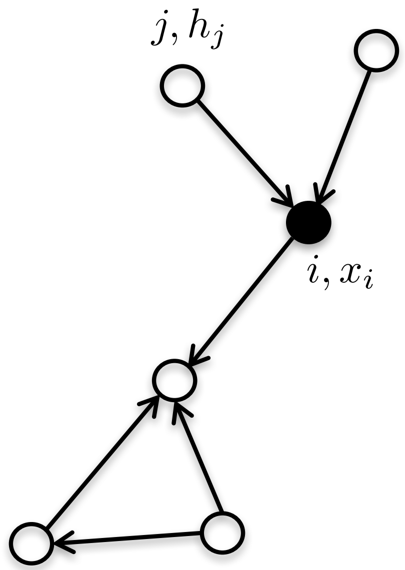
Graph RNNs

- **Graph RNN: Multilayer perceptron**[27]

$$h_i = \sum_{j \to i} \mathcal{C}_{\text{G-MLP}}(x_i, h_j) = \sum_{j \to i} A\sigma(B\sigma(Ux_i + Vh_j))$$

- **Graph GRU**[28,29] **(Gated Recurrent Unit)**

$$h_i = \mathcal{C}_{\text{G-GRU}}(x_i, \sum_{j \to i} h_j)$$



Fixed-point iterative scheme:

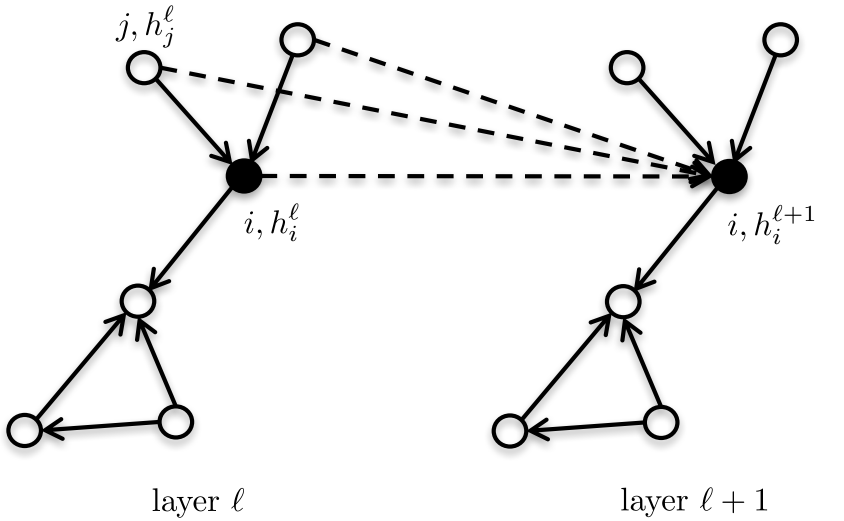
$$\begin{aligned}
\bar{h}_i^t &= \sum_{j \to i} h_j^t, \quad h_i^{t=0} = x_i \\
z_i^{t+1} &= \sigma(U_z h_i^t + V_z \bar{h}_i^t) \\
r_i^{t+1} &= \sigma(U_r h_i^t + V_r \bar{h}_i^t) \\
\tilde{h}_i^{t+1} &= \tanh\big(U_h(h_i^t \odot r_i^{t+1}) + V_h \bar{h}_i^t\big) \\
h_i^{t+1} &= (1 - z_i^{t+1}) \odot h_i^t + z_i^{t+1} \odot \tilde{h}_i^{t+1}
\end{aligned}$$

[27] Scarselli et-al 2009
[28] Li, Tarlow, Brockschmidt, Zemel, 2015
[29] Cho et-al, 2014

Graph ConvNets

- **Vanilla graph ConvNets[30]:**

$$
\begin{aligned}
h_i^{\ell+1} &= \mathcal{C}_{\text{G-VCN}}\Big(h_i^\ell, \sum_{j\to i} h_j^\ell\Big), \quad h_i^{\ell=0} = x_i \\
&= \mathrm{ReLU}\Big(U^\ell h_i^\ell + V^\ell \sum_{j\to i} h_j^\ell\Big), \quad h_i^{\ell=0} = x_i
\end{aligned}
$$



[30] Sukhbaatar, Szlam, Fergus 2016

Sainaa Sukhbaatar (New York University)

Deep Architecture for Sets and Its Application to
Multi-agent Communication

Friday, February 9, 2018, 9:00 - 9:40

Graph RNNs or ConvNets?

- Common trend: Most published papers use RNN architectures (GRU, LSTM) $\Rightarrow$ Are they superior to ConvNet architectures for arbitrary graphs?

- Numerical study to compare both graph architectures[31] for 2 basic and representative graph problems:
  - Subgraph matching[27]
  - Semi-supervised classification

[27] Scarselli et-al 2009
[31] Bresson, Laurent 2017

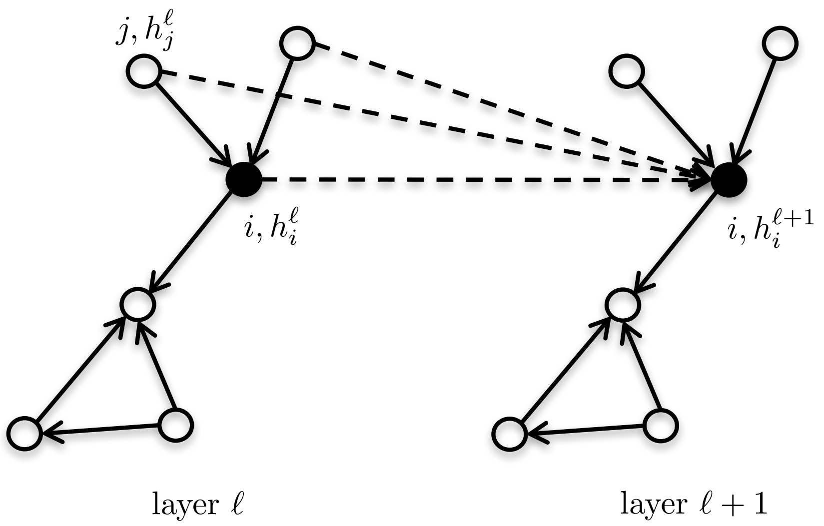
Gated graph ConvNets

- Graph ConvNets architecture with edge gating mechanism (leveraging[30,32,33]) and residuality[31]:

$$h_i^{\ell+1} = \mathrm{ReLU}\left(U^\ell h_i^\ell + \sum_{j \to i} \eta_{ij} \odot V^\ell h_j^\ell \right)$$

edge gates
(anisotropic property)

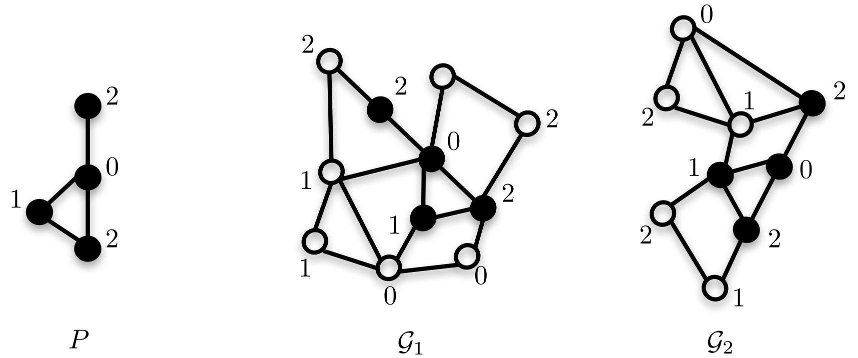
$$\eta_{ij} = \sigma\left(A^\ell h_i^\ell + B^\ell h_j^\ell \right)$$



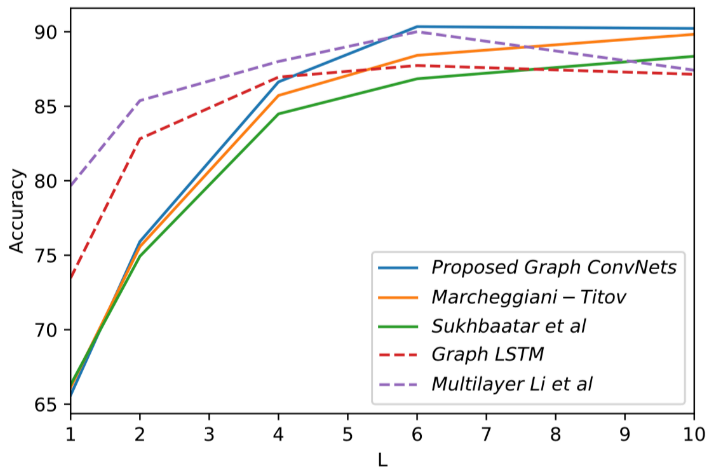
[31] Bresson, Laurent 2017
[30] Sukhbaatar, Szlam, Fergus 2016
[32] Tai, Socher, Manning, 2015
[33] Marcheggiani, Titov 2017

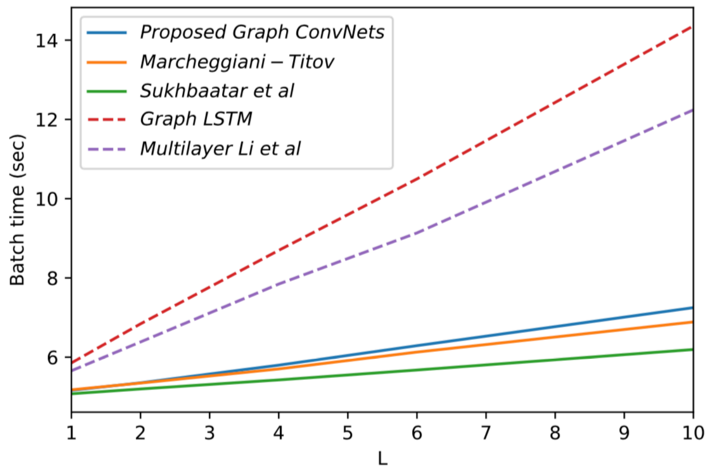
Graph learning problem 1

- Pattern matching



- Experimental results:



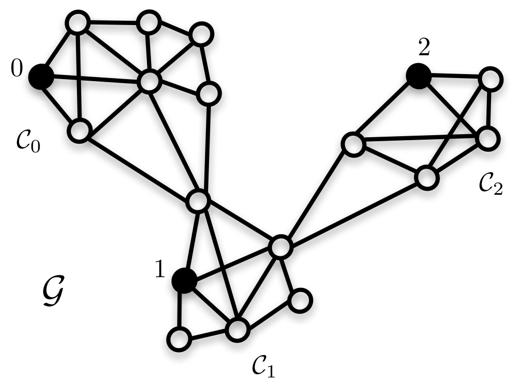


- All graph NNs are upgraded with residuality (offers 10% improvements).

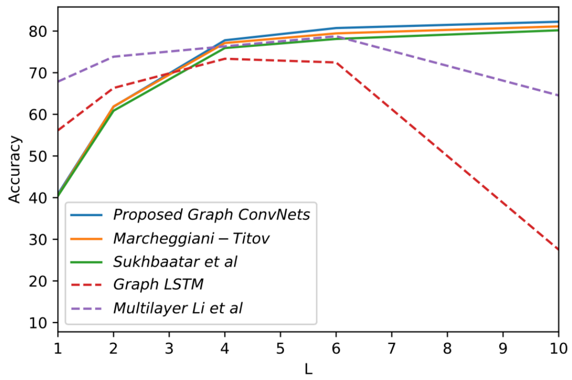
[31] Bresson, Laurent 2017

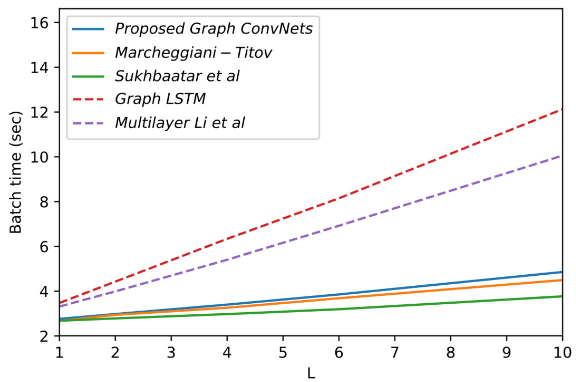
Graph learning problem 2

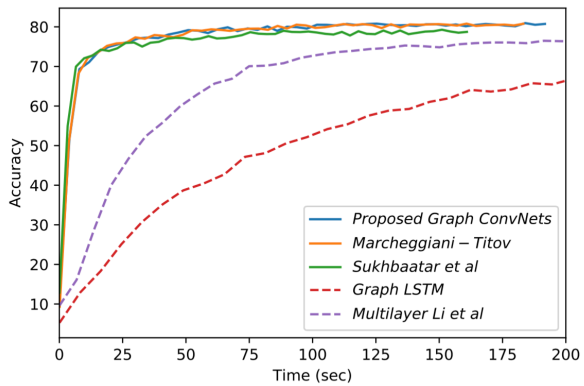
- **Semi-supervised clustering**



- **Experimental results:**



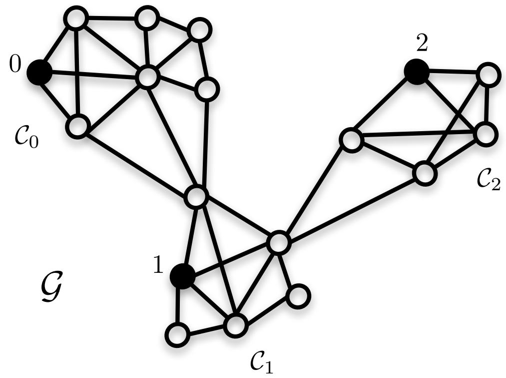




- **Conclusion:** Use ConvNets architecture for variable graphs.

Learning vs. non-learning techniques

- Semi-supervised clustering

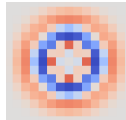


- Comparing learning vs non-learning (variational) techniques[34]: 82% vs 45% and test time is $O(E)$ vs $O(E^{3/2})$.

[34] Grady 2006

Remarks

- Anisotropy vs isotropy:
  - Standard ConvNets produce anisotropic filters because Euclidean grids have directional structure.
  - Graph ConvNets compute isotropic filters because there is no notion of directions on arbitrary graphs.
  - How to get anisotropy back for graphs?
    - Edge gates[31]/attention[36] information to treat neighbors differently.
    - Differentiate graph edges and graph vertices[35] (e.g. different atoms and atom connections)

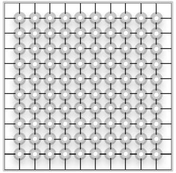


- Graph learning:
  - For social networks, brain connectivity, road network, the graph is fixed and given.
  - For citations network, image network, NLP, the graph must be constructed/learned.

[31] Bresson, Laurent 2017
[35] Gilmer et-al 2017
[36] Velickovic et-al 2017

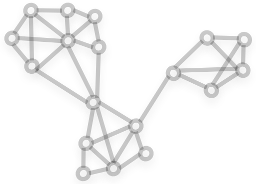
Outline

- Part 1: Euclidean ConvNets



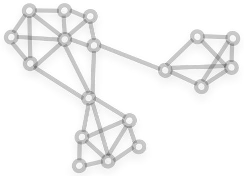
  - *Architecture*
  - *Non-Euclidean data*

- Part 2: Spectral ConvNets for Fixed Graphs



  - Spectral Graph Theory
    - *Graph convolution*
    - *Graph coarsening*
  - Spectral ConvNets
    - *SplineNets*
    - *ChebNets* [NIPS'16]*
    - *GraphConvNets*
    - *CayleyNets**
    - *Multiple fixed graphs* [NIPS'17]*

- Part 3: ConvNets for Variable Graphs



  - *Graph learning problems**

- **Conclusion**

Conclusion

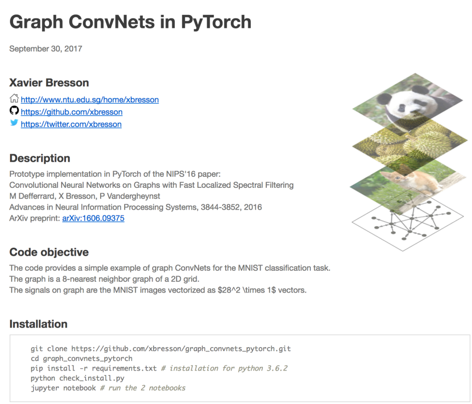
- Contributions:
  - Generalization of ConvNets to data on graphs
  - Exact/tight localized filters on graphs
  - Linear complexity for sparse graphs
  - GPU implementation
  - Rich expressivity
  - Multiple and arbitrary graphs

- Several potential applications in data and network sciences.

Papers and codes

- Papers
  - Convolutional neural networks on graphs with fast localized spectral filtering, M Defferrard, X Bresson, P Vandergheynst, NIPS, 2016, arXiv:1606.09375
  - Geometric matrix completion with recurrent multi-graph neural networks, F Monti, MM Bronstein, X Bresson, NIPS, 2017, arXiv:1704.06803
  - CayleyNets: Graph Convolutional Neural Networks with Complex Rational Spectral Filters, R Levie, F Monti, X Bresson, MM Bronstein, 2017, arXiv:1705.07664
  - Residual Gated Graph ConvNets, X Bresson, T Laurent, 2017, arXiv:1711.07553

- Codes
  - https://github.com/xbresson/graph_convnets_pytorch



Graph ConvNets in PyTorch

September 30, 2017

Xavier Bresson

http://www.ntu.edu.sg/home/xbresson
https://github.com/xbresson
https://twitter.com/xbresson

Description

Prototype implementation in PyTorch of the NIPS'16 paper:
Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering
M Defferrard, X Bresson, P Vandergheynst
Advances in Neural Information Processing Systems, 3844-3852, 2016
ArXiv preprint: arXiv:1606.09375

Code objective

The code provides a simple example of graph ConvNets for the MNIST classification task.
The graph is a 8-nearest neighbor graph of a 2D grid.
The signals on graph are the MNIST images vectorized as $28^2 \times 1$ vectors.

Installation

```
git clone https://github.com/xbresson/graph_convnets_pytorch.git
cd graph_convnets_pytorch
pip install -r requirements.txt # installation for python 3.6.2
python check_install.py
jupyter notebook # run the 2 notebooks
```

Thank you

http://www.ntu.edu.sg/home/xbresson
https://github.com/xbresson
https://twitter.com/xbresson