Deep Learning on Irregular Domains (DLID)

BMVC, ICL, London September 7, 2017

Deep Learning on Graphs

Learning beyond Euclidean Data

Michaël Defferrard

École Polytechnique Fédérale de Lausanne (EPFL)

Joint work with Youngjoo Seo (EPFL)
Xavier Bresson (NTU)
Pierre Vandergheynst (EPFL)

Structured data

Why structure data ?

- To incorporate additional information.
- To exploit spatial correlations.
- To decrease learning complexity by making geometric assumptions.

Data structured by Euclidean grids.

- 1D: sound, time-series.
- 2D: images.
- 3D: video, hyper-spectral images.

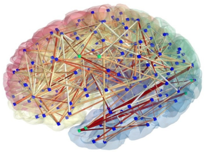
Naturally graph-structured data

Modeling versatility: graphs model heterogeneous pairwise relationships.

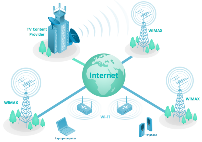
Examples of irregular / graph-structured data:

- Social networks: Facebook, Twitter.
- Biological networks: genes, molecules, brain connectivity.
- Infrastructure networks: energy, transportation, Internet, telephony.

Social network



Brain structure



Telecommunication

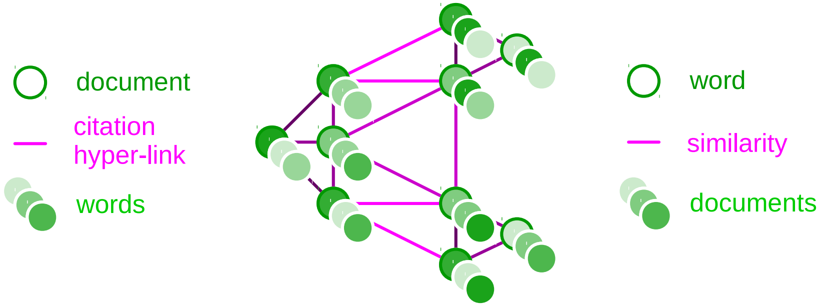
Types of graphs

Sample graph

- Semi-supervised learning.
- Incorporate external information.

Feature graph

- Reduce computations.
- Incorporate external information.



Problems: signals, nodes or graphs classification (regression).

Using the structure

Extrinsic: embed the graph in an Euclidean space.

- Each node is represented by a vector.
- Use that embedding as additional features for a fully connected NN.
- Use a convolutional NN in the embedding space.
  Possibly very high-dimensional!

Intrinsic: a Neural Net defined on graphically structured data.

- Exploit geometric structure for computational efficiency.
- Starting point: ConvNet, an intrinsic formulation for Euclidean grids.

Why ConvNets?

ConvNets are extremely efficient at extracting meaningful statistical patterns in large-scale and high-dimensional datasets.

They exploit the geometry.

Statistical assumptions

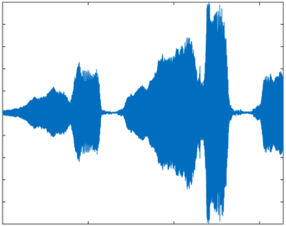
- Localization: compact filters for low complexity.
- Stationarity: translation invariance.
- Compositionality: analysis with a filterbank.
- Multi-scale: hierarchical features extracted by multiple layers.

Developed for data lying on Euclidean grids

All operations are well defined and computationally efficient:

1. Convolution $\rightarrow$ filter translation or fast Fourier transform (FFT).
2. Down-sampling $\rightarrow$ pick one pixel out of n.
3. Non-linearity $\rightarrow$ point-wise operation.
4. Pooling $\rightarrow$ summarize the receptive field.

Image (2D) Video (3D)

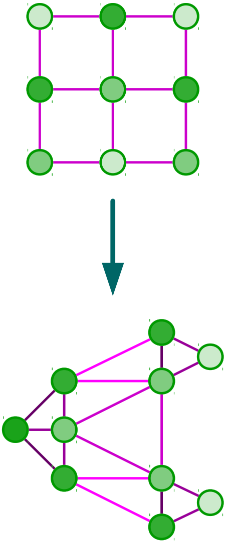


Sound (1D)

ConvNets on graphs

Graphs vs Euclidean grids

- Irregular sampling.
- Weighted edges.
- No orientation (in general).



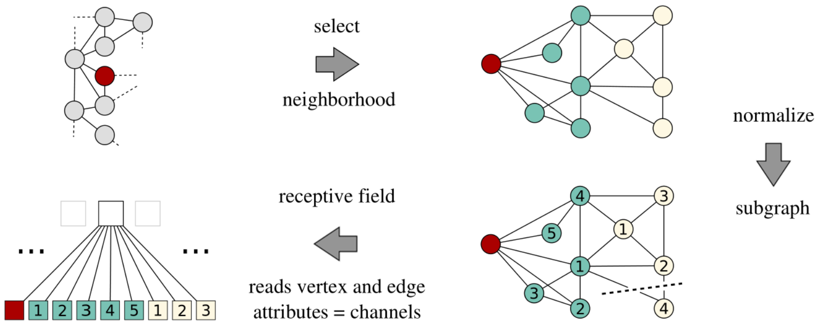
Challenges

1. Formulate convolution and down-sampling on graphs.
2. Make them efficient!

ConvNets on graphs: spatial approach

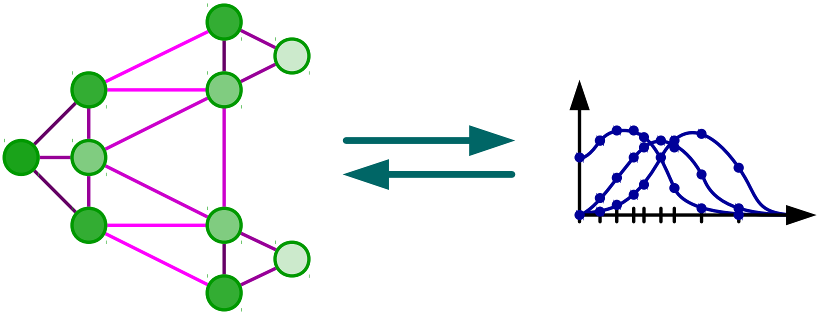
Niepert, Ahmed, and Kutzkov 2016

1. Define receptive field / neighborhood.
2. Order nodes, i.e. give orientations to a node's edges.



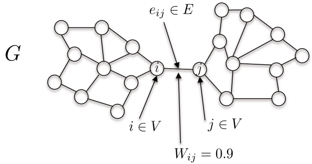
ConvNets on graphs: spectral approach

Bruna, Zaremba, Szlam, and LeCun 2014; Henaff, Bruna, and LeCun 2015



Graphs

$\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$: undirected and connected graph



- $\mathcal{V}$: set of $|\mathcal{V}| = n$ vertices
- $\mathcal{E}$: set of edges
- $W \in \mathbb{R}^{n \times n}$: weighted adjacency matrix
- $D_{ii} = \sum_j W_{ij}$: diagonal degree matrix

Graph Laplacians (core operator to spectral graph theory):

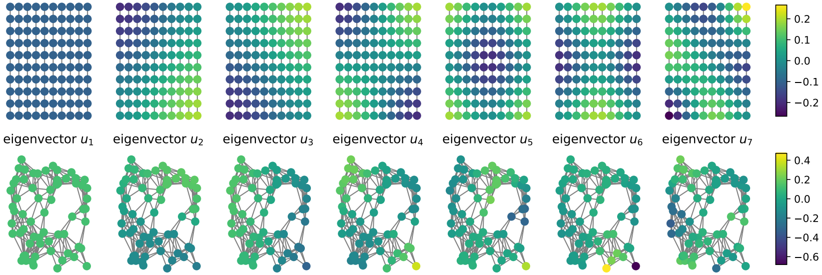
- $L = D - W \in \mathbb{R}^{n \times n}$: combinatorial
- $L = I_n - D^{-1/2} W D^{-1/2}$: normalized

Graph Fourier basis

Shuman, Narang, Frossard, Ortega, and Vandergheynst 2013

L is symmetric and positive semidefinite $\rightarrow L = U \Lambda U^T$ (EVD)

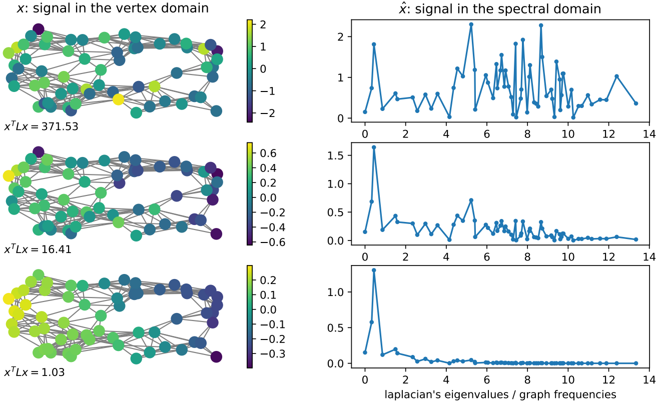
- Graph Fourier basis $U = [u_1, \ldots, u_n] \in \mathbb{R}^{n \times n}$
- Graph "frequencies" $\Lambda = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix} \in \mathbb{R}^{n \times n}$



Graph Fourier Transform

Shuman, Narang, Frossard, Ortega, and Vandergheynst 2013

- Graph signal $x : \mathcal{V} \to \mathbb{R}$ seen as $x \in \mathbb{R}^n$
- Transform: $\hat{x} = \mathcal{F}_{\mathcal{G}}\{x\} = U^T x \in \mathbb{R}^n$
- Inverse: $x = \mathcal{F}_{\mathcal{G}}^{-1}\{x\} = U\hat{x} = UU^T x = x$

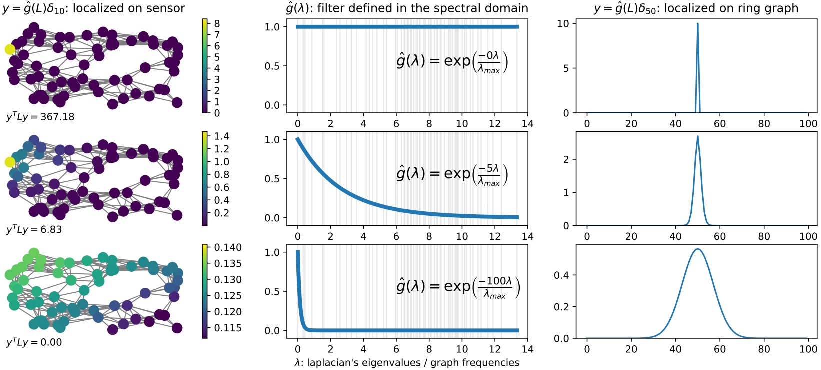


Filtering with convolution on graphs

Shuman, Narang, Frossard, Ortega, and Vandergheynst 2013

Convolution theorem: $y = x *_{\mathcal{G}} g = U\left(U^T g \odot U^T x\right) = U\left(\hat{g} \odot U^T x\right)$

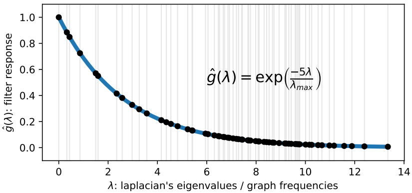
$$y = x *_{\mathcal{G}} g = U \begin{bmatrix} \hat{g}(\lambda_0) & & 0 \\ & \ddots & \\ 0 & & \hat{g}(\lambda_{n-1}) \end{bmatrix} U^T x = U \hat{g}(\Lambda) U^T x = \hat{g}(L) x$$



Spectral filtering of graph signals

Non-parametric filter, can learn all possible filters:

$$\hat{g}_\theta(\Lambda) = \mathrm{diag}(\theta),\ \theta \in \mathbb{R}^n$$



- Non-localized in vertex domain
- Learning complexity in $\mathcal{O}(n)$
- Computational complexity in $\mathcal{O}(n^2)$ (& memory)

Variation: a smooth function such as $\hat{g}_\theta(\Lambda) = B\theta$ where B is the cubic spline basis (Bruna, Zaremba, Szlam, and LeCun 2014).

Polynomial parametrization

Shuman, Ricaud, and Vandergheynst 2016

$$\hat{g}_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k, \ \theta \in \mathbb{R}^K$$

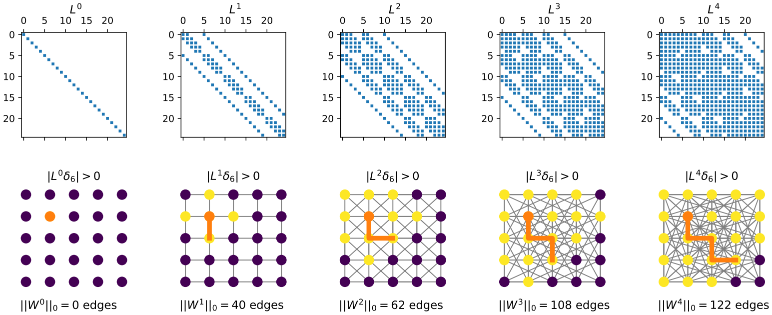
- Can learn all K-localized filters.
- Distributed computing: only need access to the K-neighborhood.

- K-localized
- Learning complexity in $\mathcal{O}(K)$
- Computational complexity in $\mathcal{O}(n^2)$

Filter localization

Hammond, Vandergheynst, and Gribonval 2011, Lemma 5.2

- Value at j of g_θ centered at i: $(\hat{g}_\theta(L)\delta_i)_j = (\hat{g}_\theta(L))_{i,j} = \sum_k \theta_k (L^k)_{i,j}$
- $d_\mathcal{G}(i,j) > K$ implies $(L^K)_{i,j} = 0$



Filter localization

Shuman, Ricaud, and Vandergheynst 2016

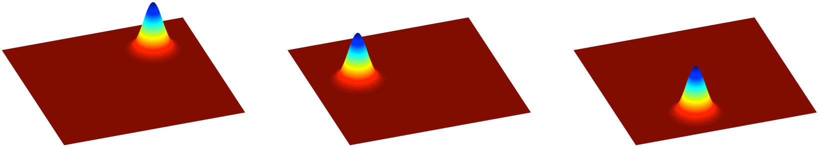


Figure: Localization on regular Euclidean grid.

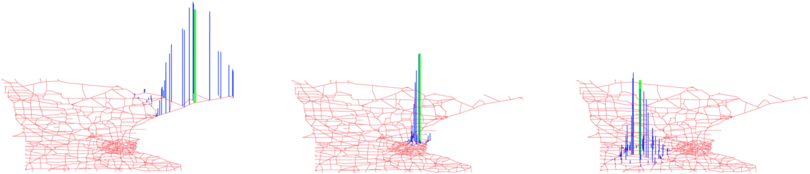


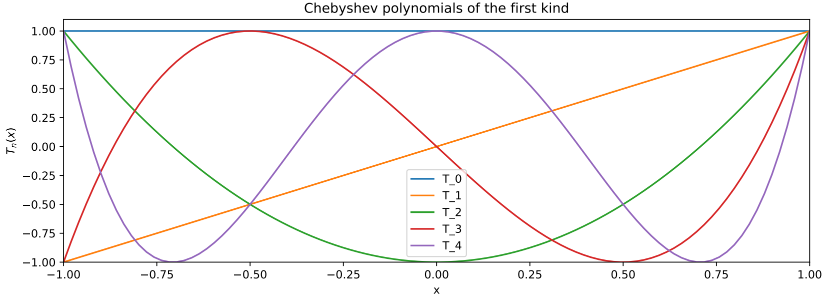
Figure: Localization on graph with $(\hat{g}_\theta(L)\delta_i)_j = (\hat{g}_\theta(L))_{i,j}$.

Recursive formulation with Chebyshev polynomials

Hammond, Vandergheynst, and Gribonval 2011

$$\hat{g}_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}), \quad \tilde{\Lambda} = 2\lambda_n^{-1}\Lambda - I_n$$

Chebyshev polynomials: $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0 = 1$ and $T_1 = x$



Recursive formulation with Chebyshev polynomials

$$y = \hat{g}_\theta(L)x = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})x, \quad \tilde{L} = 2\lambda_n^{-1} L - I_n$$

Recurrence:

$$\begin{aligned} y &= \hat{g}_\theta(L)x = [\bar{x}_0, \ldots, \bar{x}_{K-1}]\theta \\ \bar{x}_k &= T_k(\tilde{L})x = 2\tilde{L}\bar{x}_{k-1} - \bar{x}_{k-2} \\ \bar{x}_0 &= x \\ \bar{x}_1 &= \tilde{L}x \end{aligned}$$

- K-localized
- Learning complexity in $\mathcal{O}(K)$
- Computational complexity in $\mathcal{O}(K|\mathcal{E}|)$ (same as classical ConvNets!)

Learning filters

Defferrard, Bresson, and Vandergheynst 2016

$$y_{s,j} = \sum_{i=1}^{F_{in}} \hat{g}_{\theta_{i,j}}(L) x_{s,i} \in \mathbb{R}^n$$

- $x_{s,i}$: feature map i of sample s
- $\theta_{i,j}$: trainable parameters
  ($F_{in} \cdot F_{out}$ vectors of K Chebyshev coefficients)

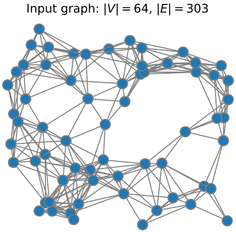
Gradients for backpropagation:

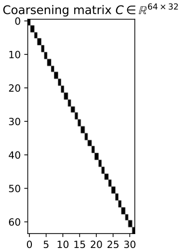
- $\frac{\partial E}{\partial \theta_{i,j}} = \sum_{s=1}^{S} [\bar{x}_{s,i,0}, \ldots, \bar{x}_{s,i,K-1}]^T \frac{\partial E}{\partial y_{s,j}}$
- $\frac{\partial E}{\partial x_{s,i}} = \sum_{j=1}^{F_{out}} g_{\theta_{i,j}}(L) \frac{\partial E}{\partial y_{s,j}}$

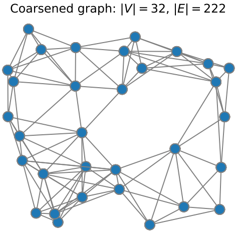
Overall cost of $\mathcal{O}(K|\mathcal{E}|F_{in}F_{out}S)$ operations, $|\mathcal{E}| \propto n$ for sparse graphs

Coarsening

Defferrard, Bresson, and Vandergheynst 2016



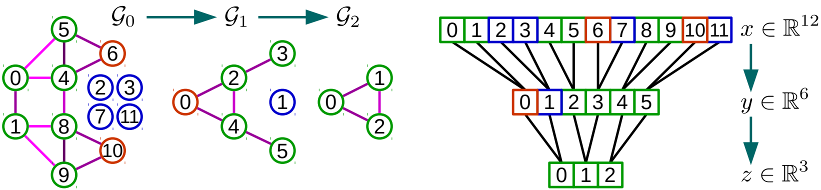




- Inherently combinatorial problem.
- Can be done as pre-processing.
- Greedy node merging with Graclus / Metis (very fast).

Pooling

Defferrard, Bresson, and Vandergheynst 2016

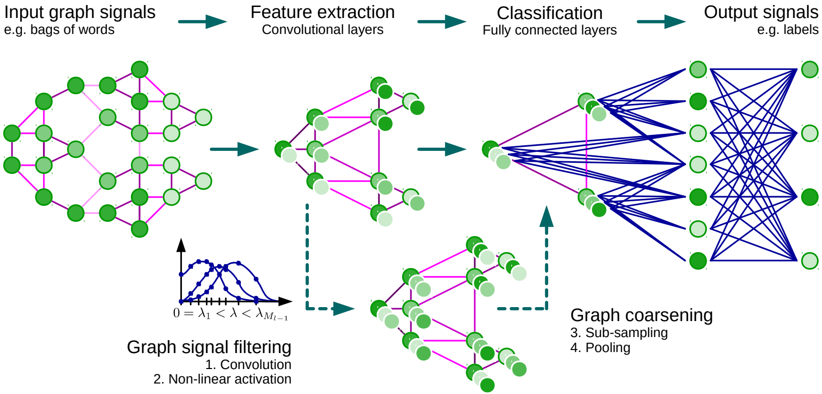


Pooling as any regular 1D signal

- Node order does not matter $\rightarrow$ arrange them for local access.
- Nodes at multiple levels are ordered as a tree.
- Satisfies parallel architectures like GPUs.

Graph ConvNet architecture

Defferrard, Bresson, and Vandergheynst 2016



Revisiting MNIST

Model	Architecture	Accuracy
Classical CNN	C32-P4-C64-P4-FC512	99.33
Proposed graph CNN	GC32-P4-GC64-P4-FC512	99.14

Table: Comparison to classical ConvNets.

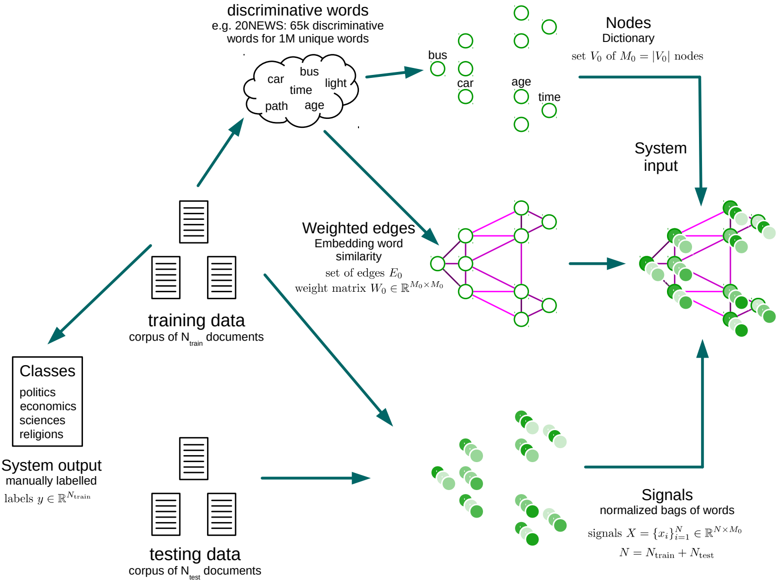
Comparable to classical ConvNets,
and better than other parametrizations!

	Accuracy		
Architecture	Non-Param	Spline	Chebyshev
GC10	95.75	97.26	97.48
GC32-P4-GC64-P4-FC512	96.28	97.15	99.14

Table: Comparison between spectral filters, $K = 25$.

20NEWS: structuring documents with a feature graph

Defferrard, Bresson, and Vandergheynst 2016



Classification accuracy & Graph quality

Model	Accuracy
Linear SVM	65.90
Multinomial Naive Bayes	68.51
Softmax	66.28
FC2500	64.64
FC2500-FC500	65.76
GC32	68.26

Proposed graph CNN vs. other methods on 20NEWS.

	word2vec			
bag-of-words	pre-learned	learned	approximate	random
67.50	66.98	68.26	67.86	67.75

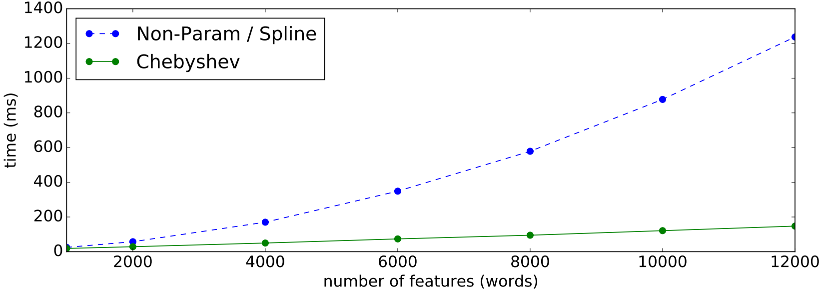
Different graphs on 20NEWS.

Architecture	8-NN on 2D Euclidean grid	random
GC32	97.40	96.88
GC32-P4-GC64-P4-FC512	99.14	95.39

Different graphs on MNIST.

20NEWS: training time

Defferrard, Bresson, and Vandergheynst 2016



Make CNNs practical for graph signals!

Spline: $\hat{g}_\theta(\Lambda) = B\theta$ where B is the cubic spline basis
(Bruna, Zaremba, Szlam, and LeCun 2014)

Recurrent Graph Convolutional Network

Seo, Defferrard, Bresson, and Vandergheynst 2016

1D signals

- $h_t = \tanh(W_x x_t + W_h h_{t-1})$
- $y_t = W h_t$
- State stored in hidden units

Graph signals

- $h_t = \tanh(W_x *_{\mathcal{G}} x_t + W_h *_{\mathcal{G}} h_{t-1})$
- $y_t = W *_{\mathcal{G}} h_t$
- State stored locally on the nodes

- Graph filtering x as $y = [\bar{x}_0, \ldots, \bar{x}_{K-1}]\theta$ is a weighted sum of diffused versions $\bar{x}$ of x.
- Data exchanged locally around the K-neighborhood.
- Reduces to independent signals if $K = 1$ or graph has no edge.

Various applications

- Semi-supervized learning
  [Kipf and Welling 2016; Manessi, Rozza, and Manzo 2017]
- Quantum Chemistry
  [Duvenaud et al. 2015; Gilmer, Schoenholz, Riley, Vinyals, and Dahl 2017]
- High Energy Physics
- Computer Graphics [Monti, Boscaini, et al. 2016; Yi, Su, Guo, and Guibas 2016; Wang, Gan, Zhang, and Shui 2017; Simonovsky and Komodakis 2017]
- Community detection [Bruna and Li 2017]
- Brain analysis
  [Ktena et al. 2017; Parisot et al. 2017; Anirudh and Thiagarajan 2017]
- Matrix completion for recommendation
  [Monti, Bronstein, and Bresson 2017]
- Neural machine translation
  [Bastings, Titov, Aziz, Marcheggiani, and Sima'an 2017]
- Link prediction and entity classification in knowledge bases
  [Schlichtkrull et al. 2017]

Conclusion

- Many problems can be defined as data structured by graphs.
- ConvNets are efficient because they exploit the geometry.
- Spatial approaches require nodes to be ordered.
- Convolution is well-defined in the Fourier domain, though the graph Fourier transform (GFT) is expensive.
- Polynomial formulations of order K are K-localized and do not require the GFT.
- Graph coarsening is a research problem in itself.
- Pooling made efficient by ordering nodes and inserting fake nodes.
- Loss of edge orientation is not killing performance on images.
- Having a good (relative to the task) graph is important.
- RNN extension can accommodate structured time series.
- Many proposed applications and many more to come.

Thanks Questions?

- Paper: Defferrard, Bresson and Vandergheynst, Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, NIPS, 2016.
- Code: `https://github.com/mdeff/cnn_graph`
- Paper: Seo, Defferrard, Bresson and Vandergheynst, Structured Sequence Modeling with Graph Convolutional Recurrent Networks, arXiv, 2017.
- Code: `https://github.com/youngjoo-epfl/gconvRNN`