Graph Signal Processing Workshop

Carnegie Mellon University June 2, 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

Majority of data is naturally unstructured.
But can be structured by graphs.

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.

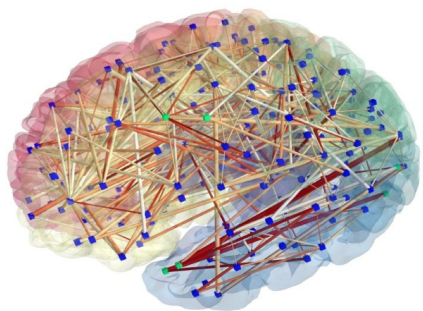
Non-Euclidean data: natural graphs

Modeling versatility: graphs model heterogeneous pairwise relationships.

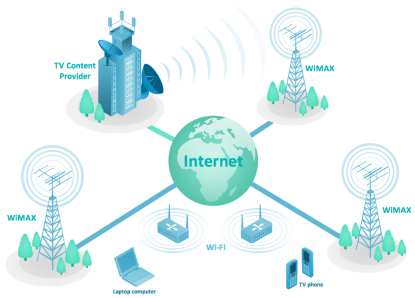
Examples of irregular / graph-structured data:

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

Social network



Brain structure



Telecommunication

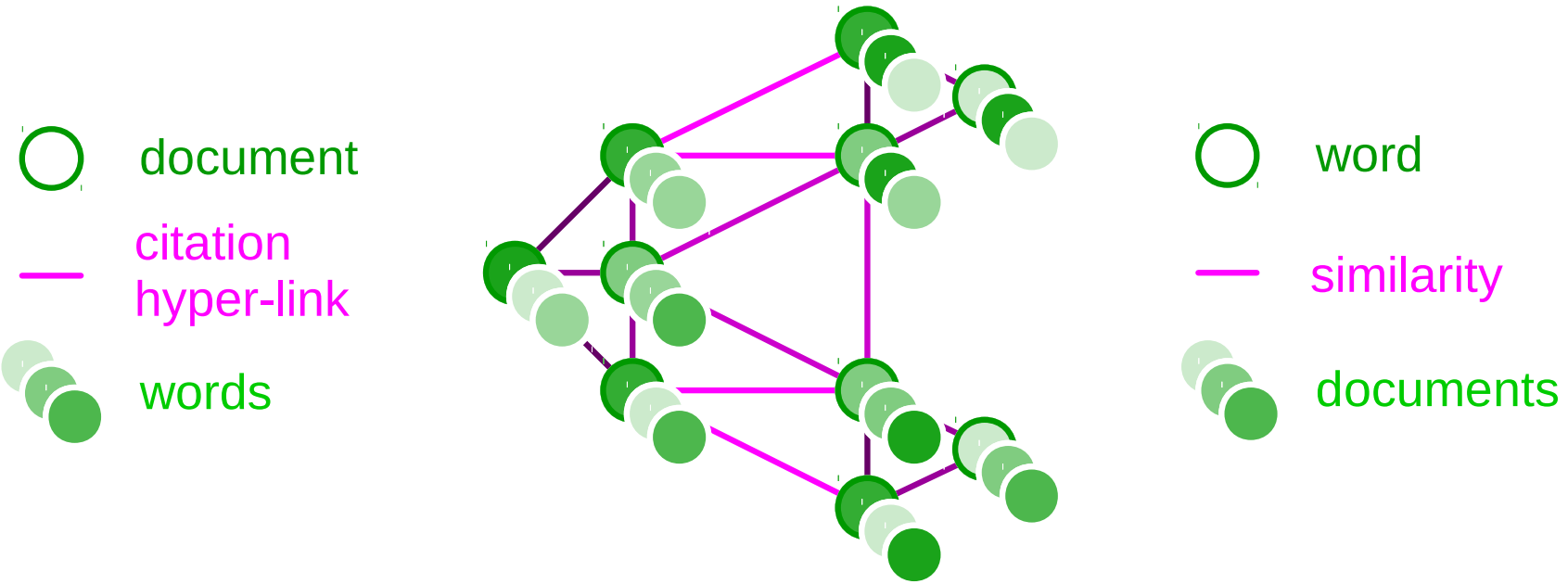
Non-Euclidean data: constructed graphs

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.

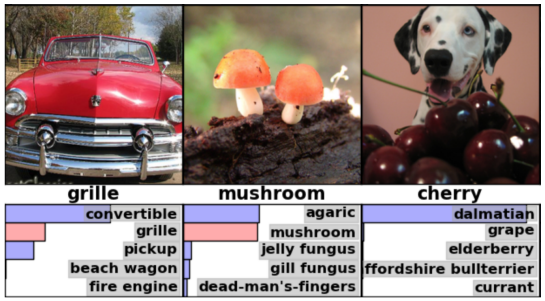
- Exploit geometric structure for computational efficiency.
- Starting point: ConvNet, an intrinsic formulation for Euclidean grids.

ConvNets are ubiquitous

LeCun, Bengio, and Hinton 2015

First developed for Computer Vision

- Object recognition
- Image captioning
- Image inpainting



Spreading outside CV

- Natural language processing
- Audio: sound & voice
- Autonomous agents (playing Atari or Go)

Why are ConvNets good ?

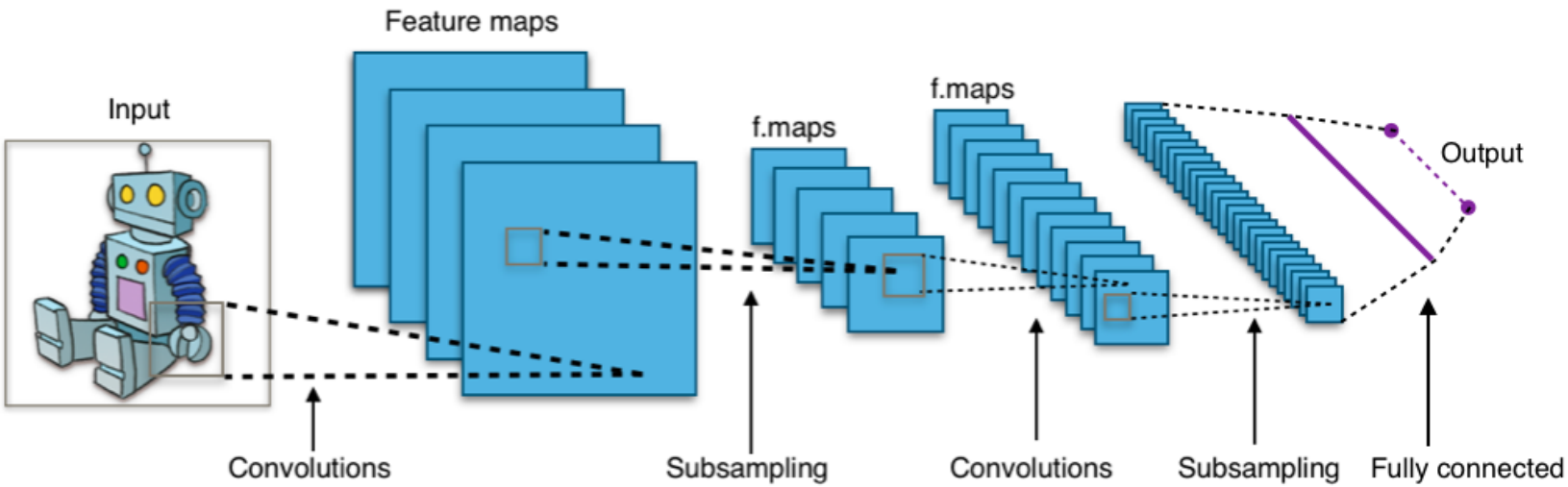
ConvNets are extremely efficient at extracting meaningful statistical patterns in large-scale and high-dimensional datasets.

They exploit the underlying geometric structure in the data.

Statistical assumptions

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

ConvNets: architecture



Ingredients

1. Convolution (local)
2. Non-linearity (point-wise)
3. Down-sampling (global / local)
4. Pooling (local)

ConvNets: feature extraction

Zeiler and Fergus 2014

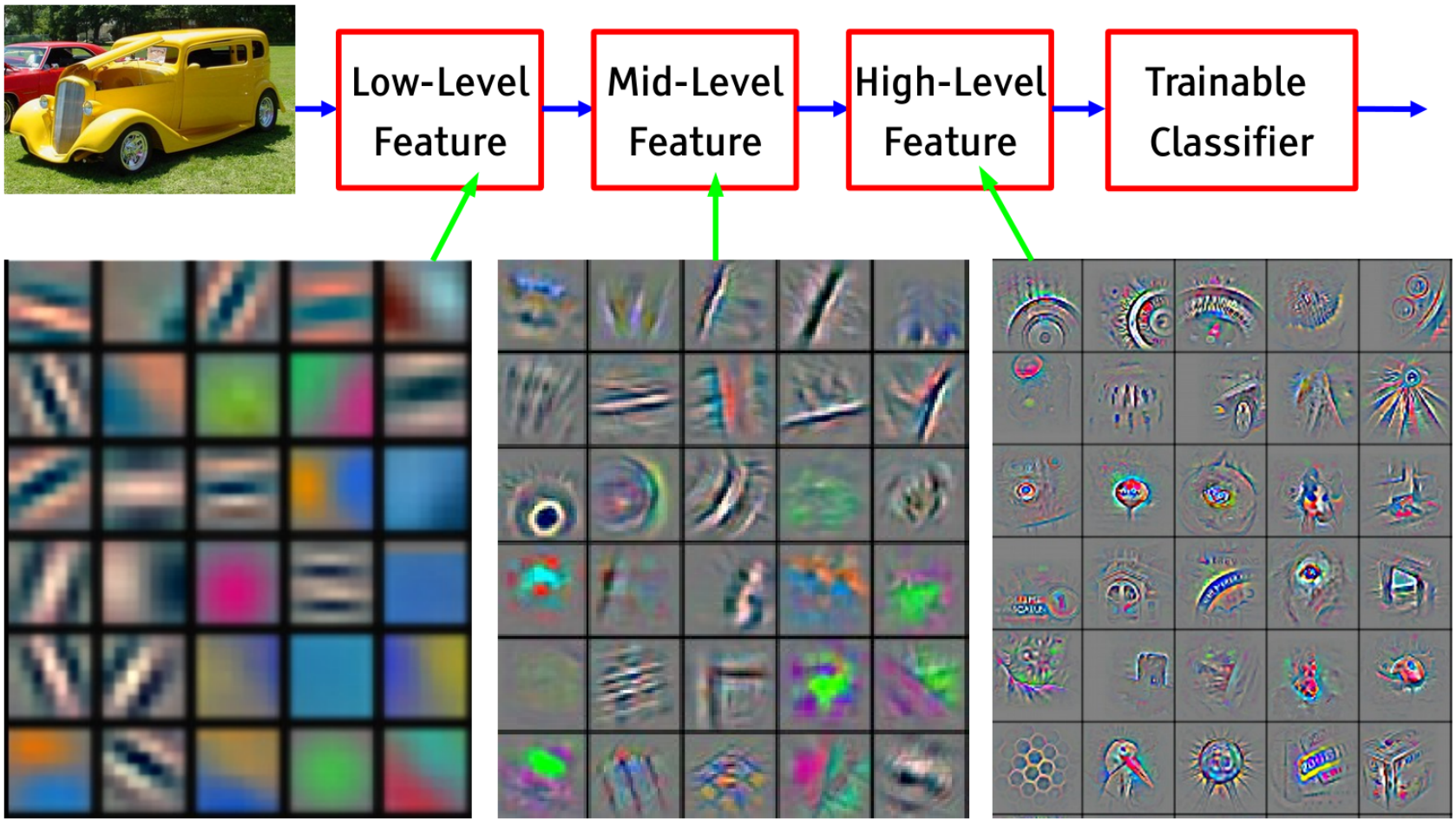


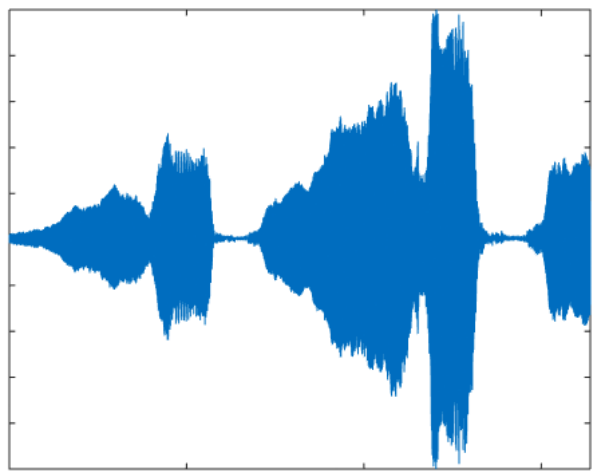
Figure: Features extracted from ImageNet.

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)

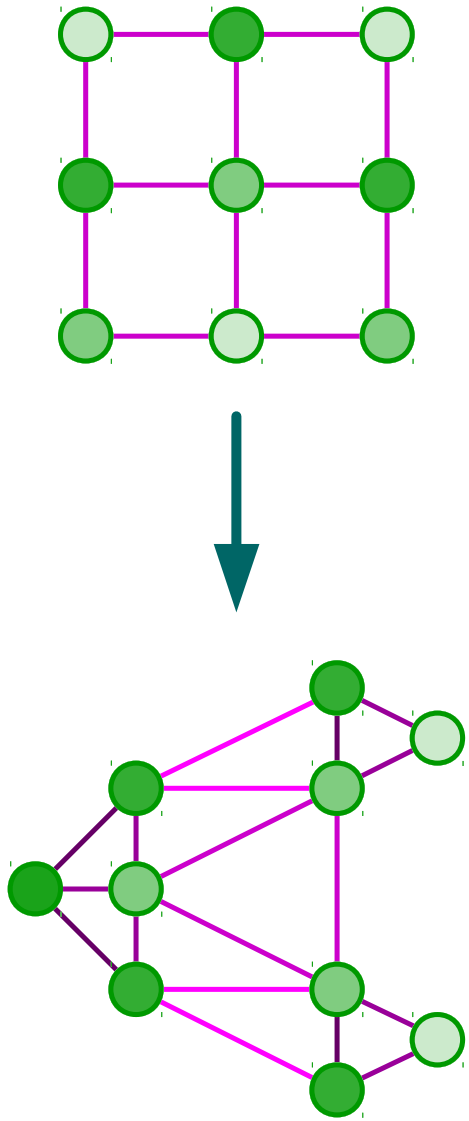


Sound (1D)

ConvNets on graphs

Graphs vs Euclidean grids

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



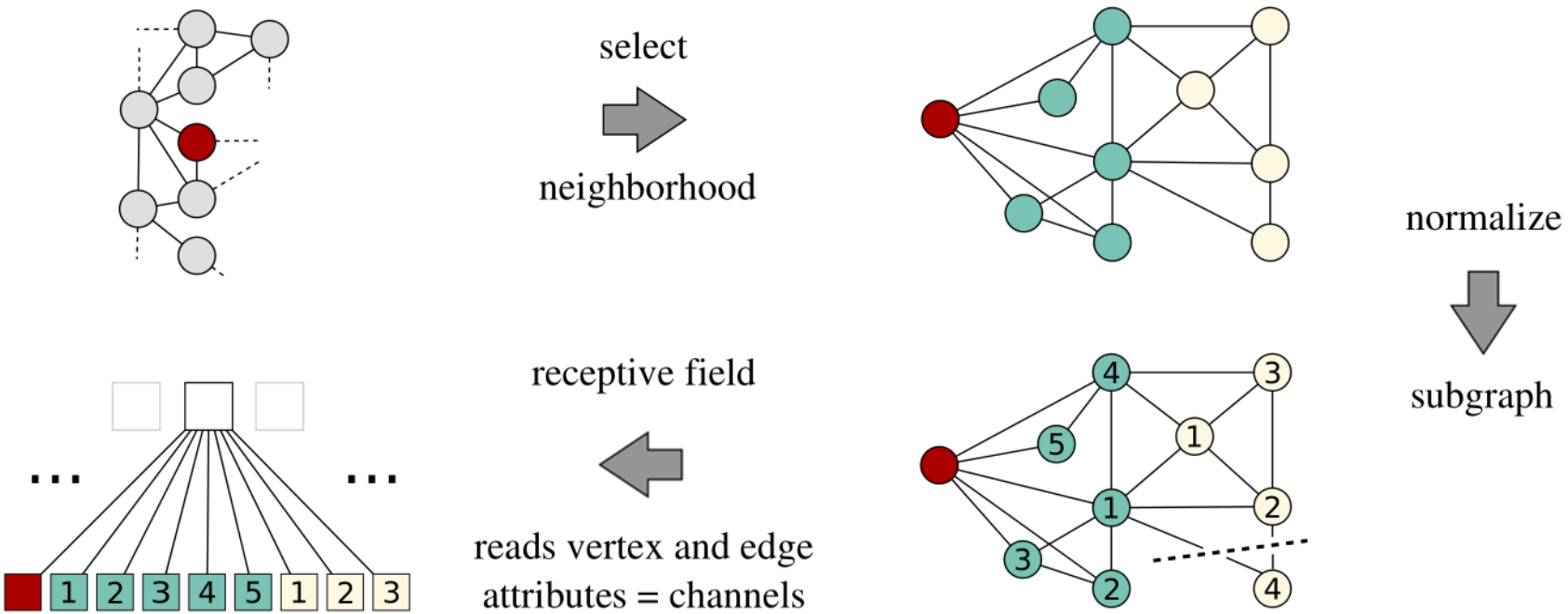
Challenges

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

ConvNets on graphs: spatial approach

Niepert, Ahmed, and Kutzkov 2016

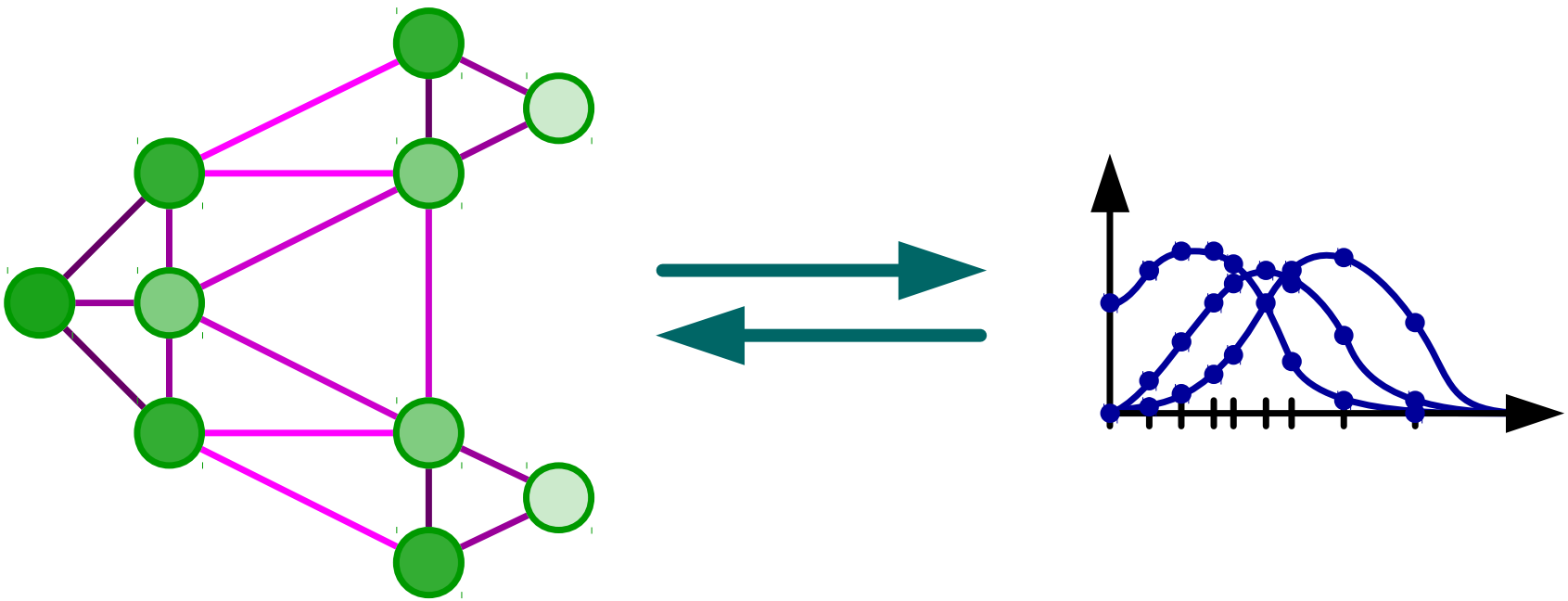
1. Define receptive field / neighborhood.
2. Order nodes, i.e. give an orientation.



ConvNets on graphs: spectral approach

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

- Spectral graph theory for convolution on graphs.
- Balanced cut model for graph coarsening (sub-sampling).

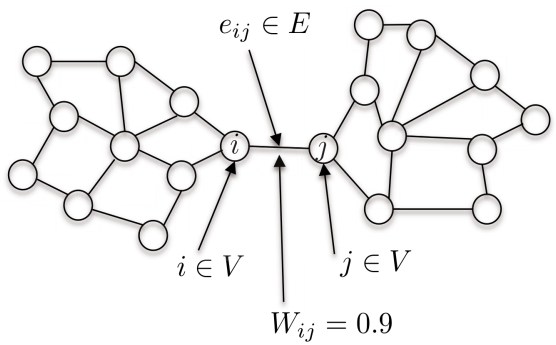


Definitions: graph

Chung 1997

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

G



- $\mathcal{V}$: 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

Definitions: graph Fourier transform

Shuman, Narang, Frossard, Ortega, and Vandergheynst 2013

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

- Graph Fourier basis $U = [u_0, \ldots, u_{n-1}] \in \mathbb{R}^{n \times n}$
- Graph "frequencies" $\Lambda = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix} \in \mathbb{R}^{n \times n}$

Graph Fourier Transform

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

Definitions: convolution on graphs

Shuman, Narang, Frossard, Ortega, and Vandergheynst 2013

Convolution theorem:

$$\begin{aligned} x *_{\mathcal{G}} g &= U\left(U^T g \odot U^T x\right) \\ &= U\left(\hat{g} \odot U^T x\right) \end{aligned}$$

Conveniently written as:

$$\begin{aligned} x *_{\mathcal{G}} g &= U \begin{bmatrix} \hat{g}(\lambda_1) & & 0 \\ & \ddots & \\ 0 & & \hat{g}(\lambda_n) \end{bmatrix} U^T x \\ &= U \hat{g}(\Lambda) U^T x \\ &= \hat{g}(L) x \end{aligned}$$

Spectral filtering of graph signals

$$y = \hat{g}_\theta(L)x = U\hat{g}_\theta(\Lambda)U^T x$$

Non-parametric filter:

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

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

Polynomial parametrization for localized filters

Shuman, Ricaud, and Vandergheynst 2016

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

- 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 $(L^K)_{i,j} = 0$
  (Hammond, Vandergheynst, and Gribonval 2011, Lemma 5.2)

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

Filter localization

Shuman, Ricaud, and Vandergheynst 2016

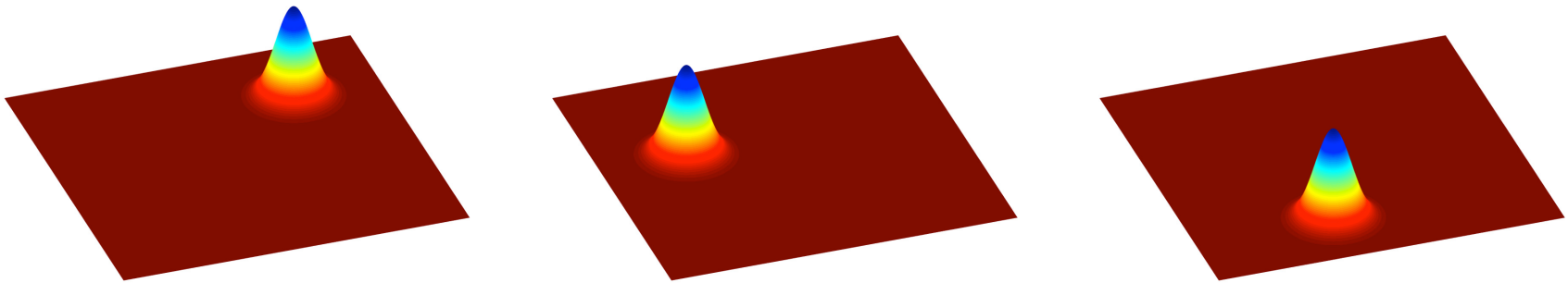


Figure: Localization on regular Euclidean grid.

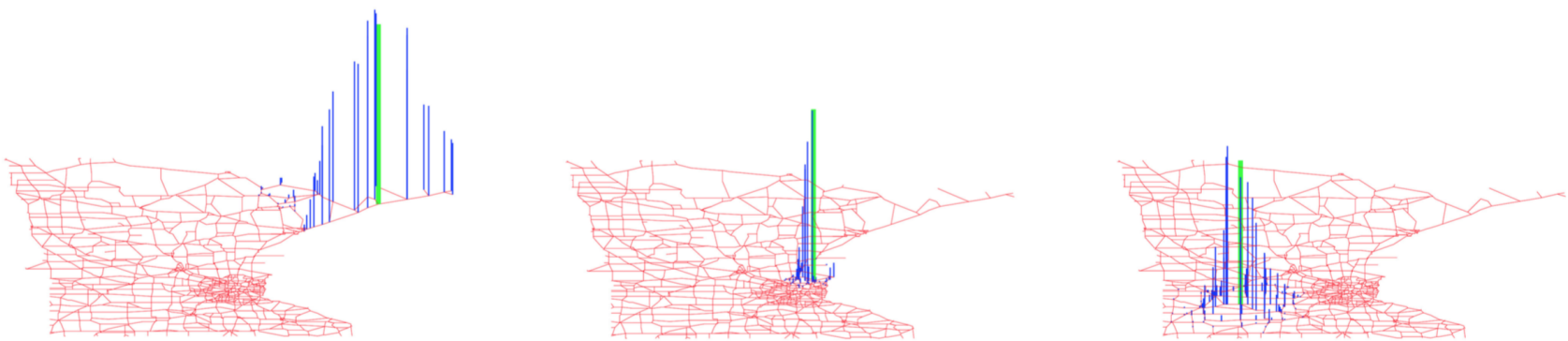


Figure: Localization on graph with $(\hat{g}_\theta(L)\delta_i)_j = (\hat{g}_\theta(L))_{i,j}$.

Recursive formulation for fast filtering

Hammond, Vandergheynst, and Gribonval 2011

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

- Chebyshev polynomials: $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0 = 1$ and $T_1 = x$
- Filtering: $y = \hat{g}_\theta(L)x = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})x$
- Recurrence: $y = \hat{g}_\theta(L)x = [\bar{x}_0, \ldots, \bar{x}_{K-1}]\theta$, $\bar{x}_k = T_k(\tilde{L})x = 2\tilde{L}\bar{x}_{k-1} - \bar{x}_{k-2}$ with $\bar{x}_0 = x$ and $\bar{x}_1 = \tilde{L}x$

- 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
  ($F_{in} \times F_{out}$ vectors of Chebyshev coefficients)

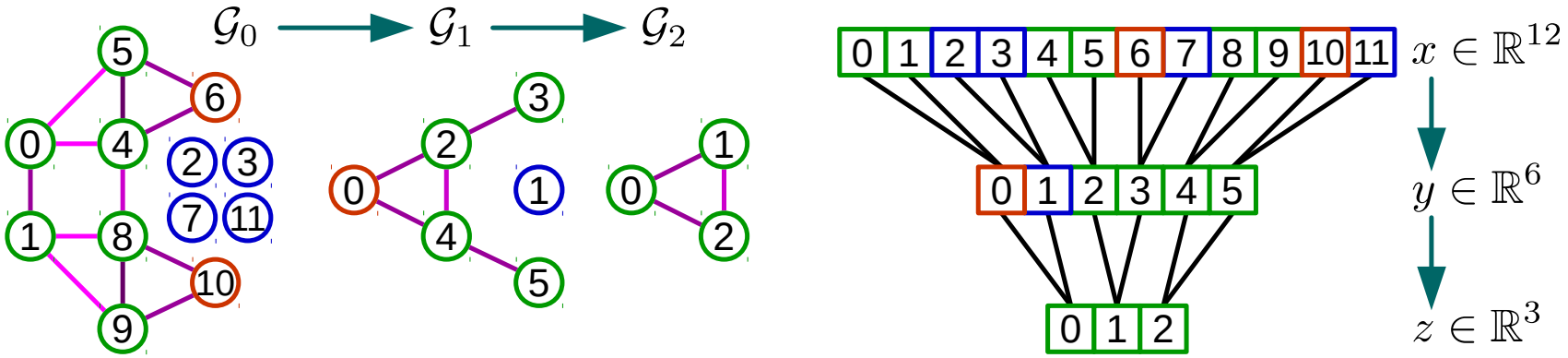
Gradients for backpropagation:

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

Overall cost of $\mathcal{O}(K|\mathcal{E}|F_{in}F_{out}S)$ operations

Coarsening & Pooling

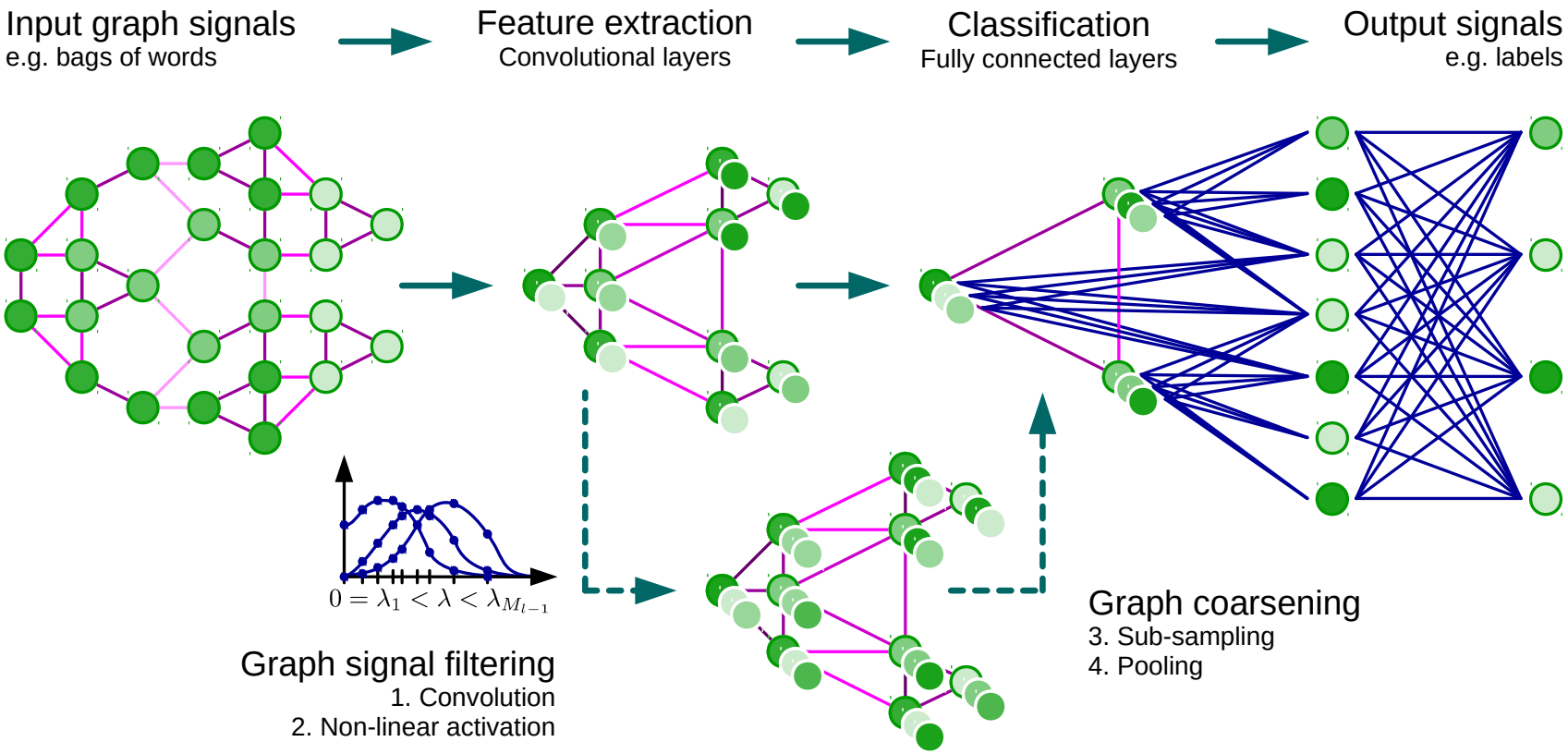
Defferrard, Bresson, and Vandergheynst 2016



- Coarsening: Graclus / Metis
  - Greedy node merging.
  - Very fast!
- Pooling: as regular 1D signals
  - Binary tree structured coarsened graphs.
  - Satisfies parallel architectures like GPUs.
- Activation: ReLU, LeakyReLU, maxout, tanh, sigmoid.

Graph ConvNet architecture

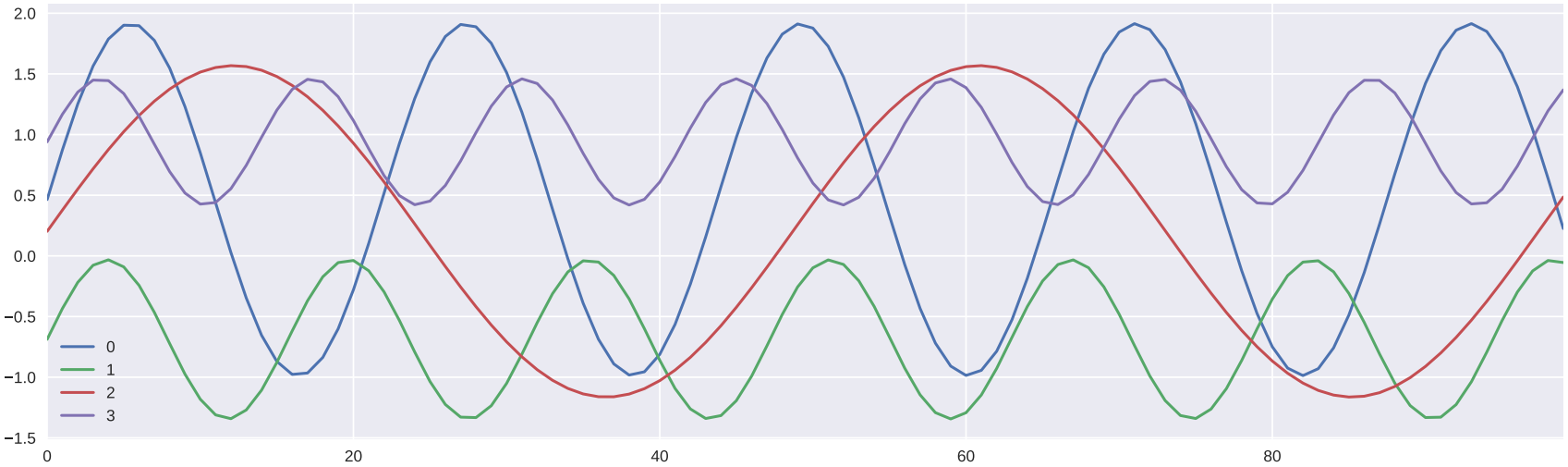
Defferrard, Bresson, and Vandergheynst 2016



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]

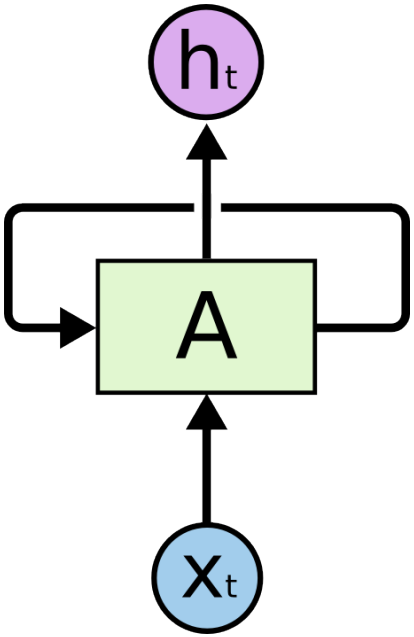
Time Series

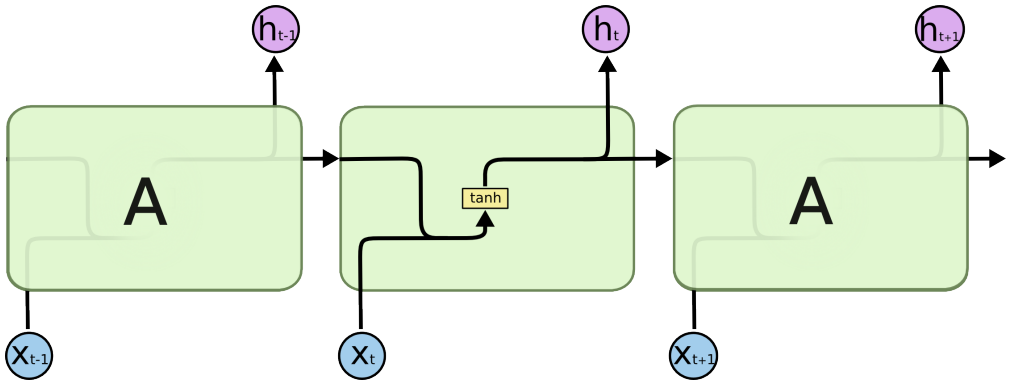


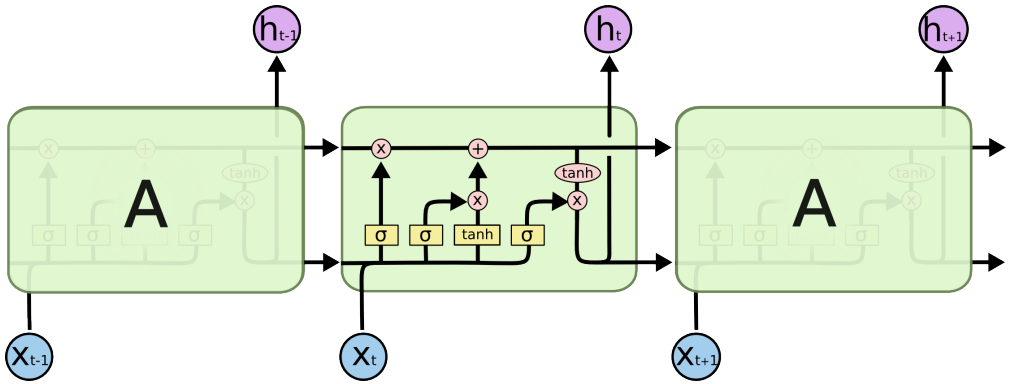
- Sensors: temperature, wind, pressure, body signals, etc.
- Stock market
- Text (series of discrete symbols, i.e. words)
- Network activity: energy, transportation, communication, brain

Recurrent Neural Networks & LSTM

Figures by Colah, 2015







Recurrent Graph Convolutional Network

Seo, Defferrard, Bresson, and Vandergheynst 2016

1D signals

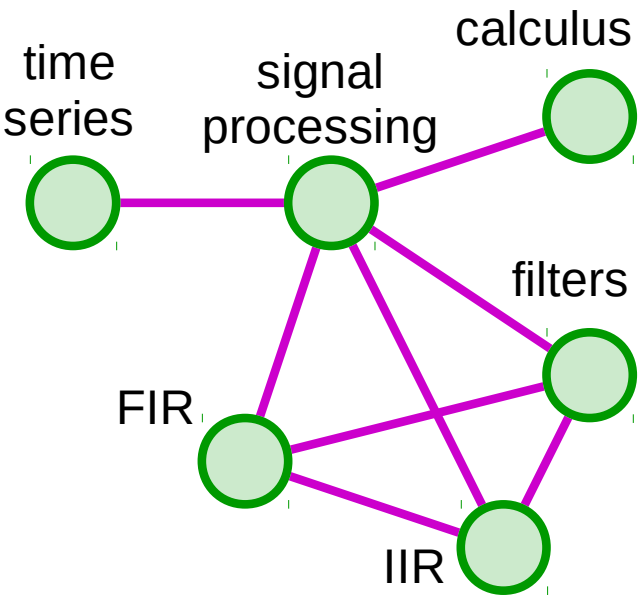
- $h_t = \tanh(W_x x_t + W_h h_{t-1})$
- $y_t = W h_t$
- State stored in hidden units

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.

Real data: Wikipedia

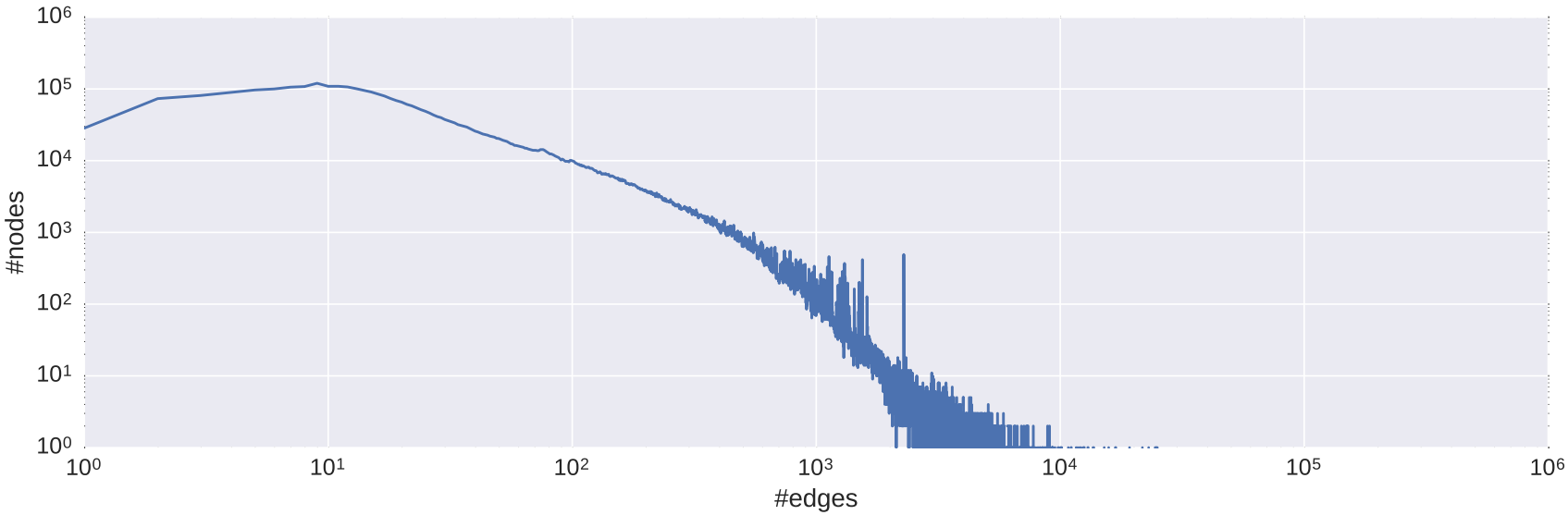


Goal: structured times series forecasting

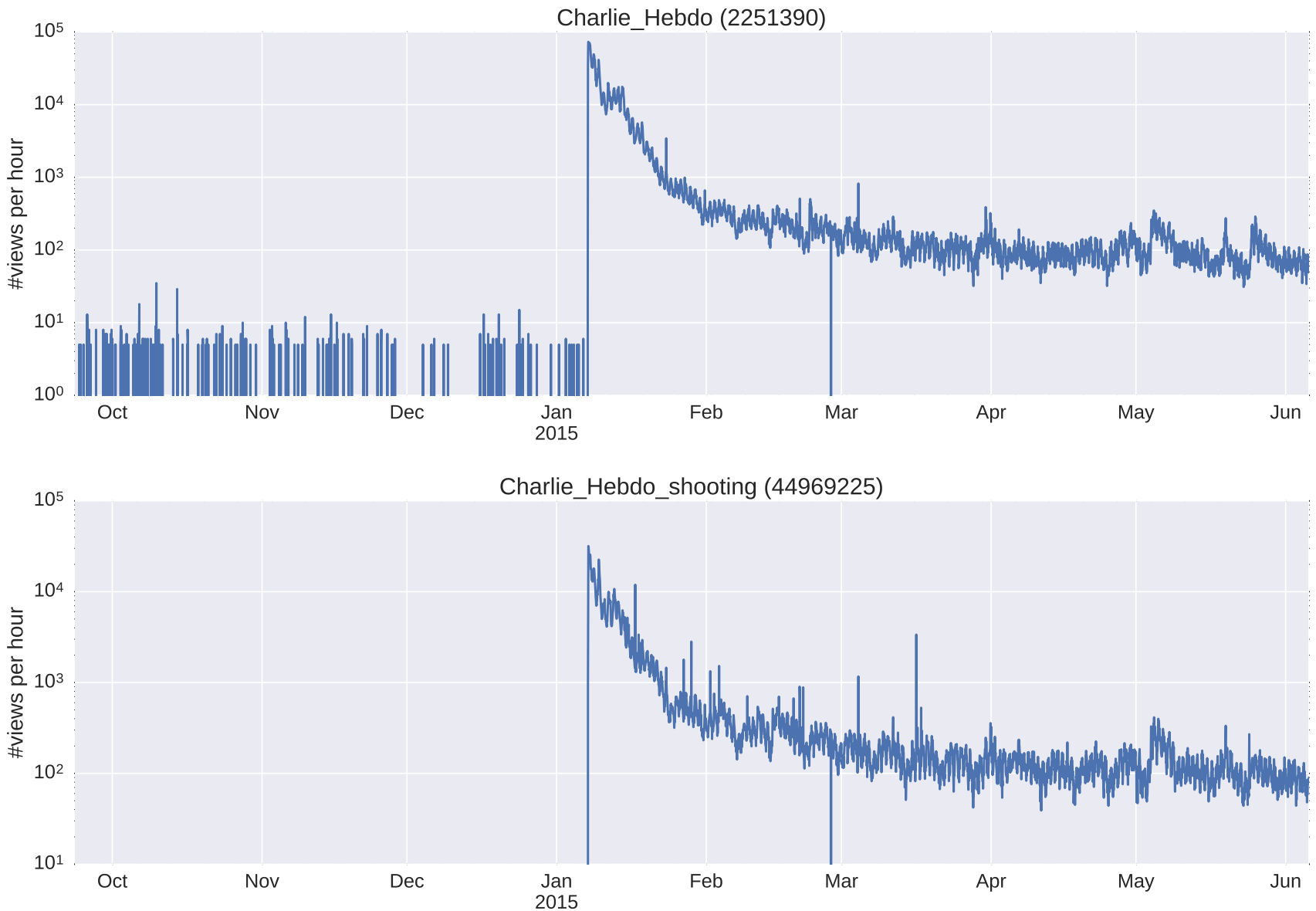
- Anomaly / event detection
- Regulation & Control
- Generative process understanding

Wikipedia network & signals

- Nodes: articles
- Edges: hyper-links
- Signals: number of hits per hour



Structured Time Series



Real data: Wikipedia

Real data (english wikipedia) is massive and hard

- 4.9M nodes
- 294M edges
- 6k time samples
- 760 GiB raw data
- dynamic: edges and nodes are added and removed

Example pruning: nodes with mean activity higher than 100 per hour

- 10k nodes
- 560k edges
- 6k time samples
- static: assume last hyper-link graph

Conclusion

- Graph are versatile tools to structure real data.
- Neural networks are the most effective ML algorithm today.

Deep Learning is coming to Graph Signal Processing

Further research

- Transfer between graphs / dynamic graphs
- Combine time & vertex domains with a joint transform
- Multi-scale approaches: both in time and vertex

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

Thanks Questions?

PS: next year, we're organizing the GSP workshop at EPFL!