Geometric Deep Learning Seminar

Tel Aviv University January 17, 2017

# Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering

Michaël Defferrard

Swiss Federal Institute of Technology (EPFL)

Joint work with Xavier Bresson (EPFL) and
Pierre Vandergheynst (EPFL)

# Structured data

Majority of data is naturally unstructured, but can be structured.

## Why structure data ?

- To incorporate additional information.
- To regularize the learning process.
- To decrease learning complexity by making geometric assumptions.

## Data structured by Euclidean grids.

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

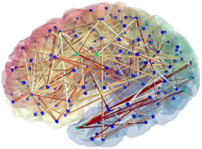
# Non-Euclidean data: natural graphs

Modeling versatility: graphs model heterogeneous pairwise relationships

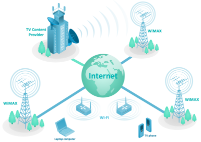
Examples of irregular / graph-structured data:

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

Social network



Brain structure




Telecommunication

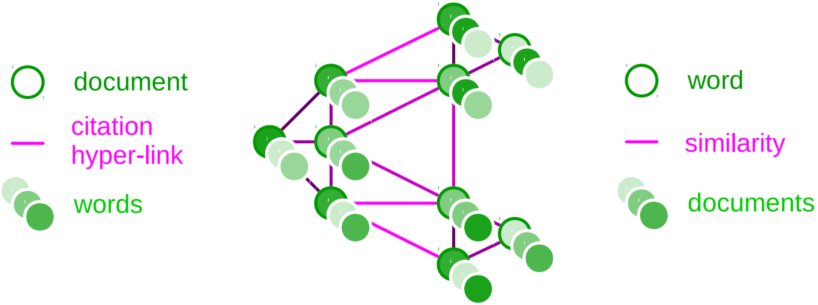
# Non-Euclidean data: constructed graphs

## Sample graph

- Semi-supervised learning.
- Incorporate external information.

## Feature graph

- Reduce computations.
- Incorporate external information.



# 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 working on graph-structured data.

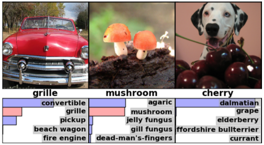
- Exploit geometric structure for computational efficiency.
- Starting point: ConvNets, an intrinsic formulation for Euclidean grids.

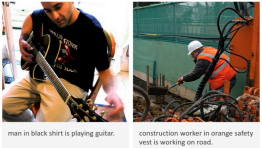
# ConvNets are ubiquitous

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 ?

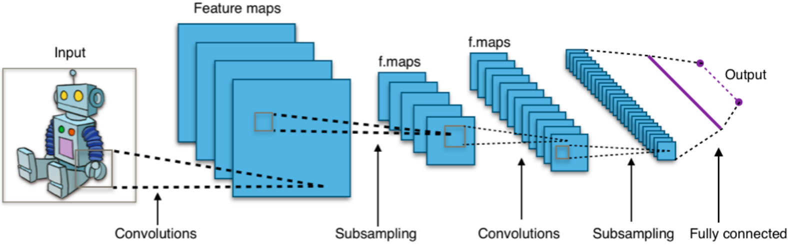
ConvNets are extremely efficient at extracting meaningful statistical patterns in large-scale and high-dimensional datasets.

Because they make use of the underlying structure in the data.

## Statistical assumptions

- Localization: compact filters for low complexity
- Stationarity: translation invariance
- Compositionality: analysis with a filterbank

# ConvNets: architecture




## Ingredients

1. Convolution
2. Non-linearity (ReLU)
3. Down-sampling
4. Pooling

# ConvNets: feature extraction

## Zeiler and Fergus 2014

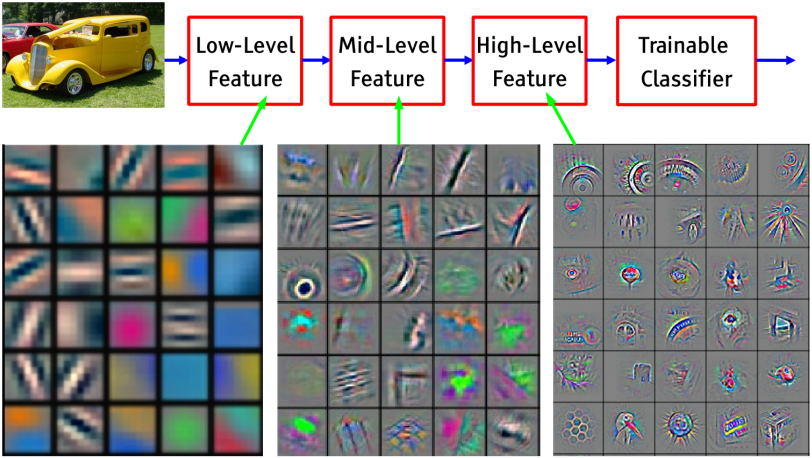


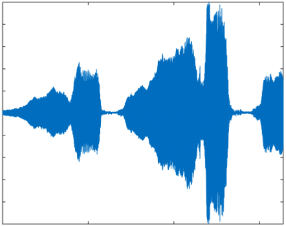

Figure: Features extracted from ImageNet

# Developed for data lying on Euclidean grids

All operations are well defined and computationally efficient:

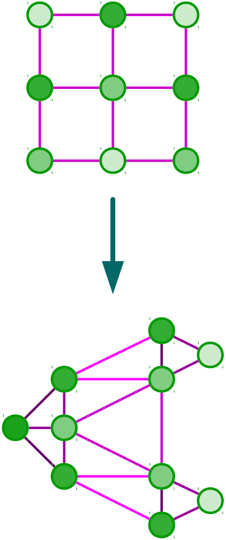
1. Convolution $\rightarrow$ filter translation or fast Fourier transform (FFT).
2. Down-sampling $\rightarrow$ pick one pixel out of $n$.

Image (2D) Video (3D)



Sound (1D)

# ConvNets on graphs



## Graphs vs Euclidean grids

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

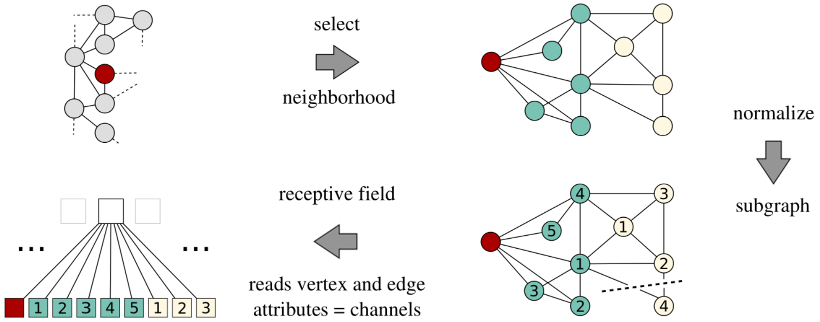
## Challenges

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

# ConvNets on graphs: spatial approach

Niepert, Ahmed, and Kutzkov 2016

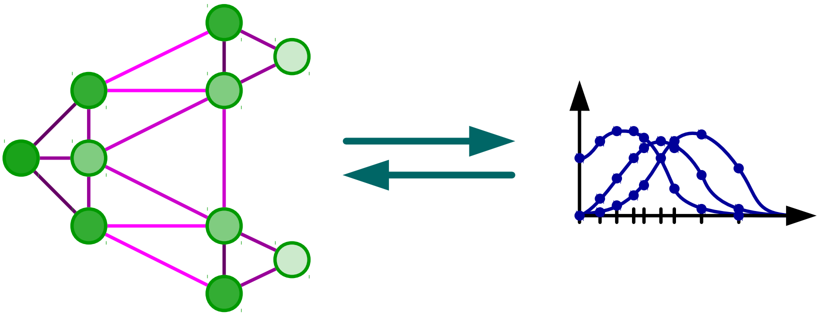
1. Define receptive field / neighborhood.
2. Order nodes.



# ConvNets on graphs: spectral approach

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

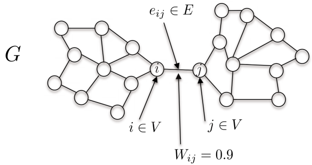
- Spectral graph theory for convolution on graphs.
- Balanced cut model for graph coarsening (sub-sampling).



# Definitions: graph

Chung 1997

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




- $\mathcal{V}$: 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:

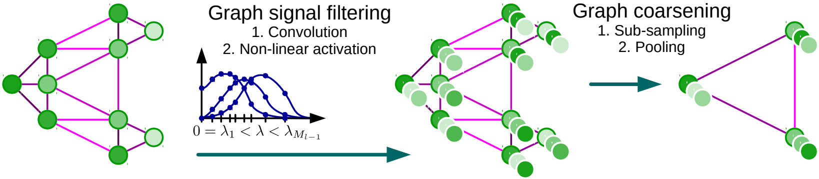
$$
\begin{aligned}
x *_{\mathcal{G}} g &= U \left( U^T g \odot U^T x \right) \\
&= U \left( \hat{g} \odot U^T x \right)
\end{aligned}
$$

Conveniently written as:

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

# Graph signal filtering

Defferrard, Bresson, and Vandergheynst 2016



# Spectral filtering of graph signals

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

Non-parametric filter:

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

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

# 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 $g_\theta$ 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

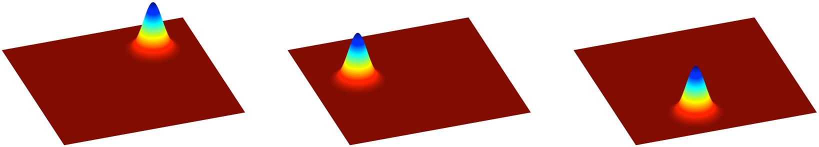


Figure: Localization on regular Euclidean grid.

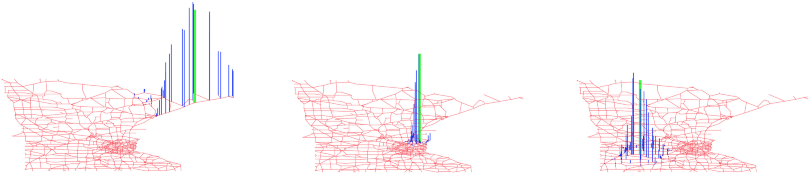


Figure: Localization on graph with $(\hat{g}_\theta(L)\delta_i)_j = (\hat{g}_\theta(L))_{i,j}$.

# 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: $y = \hat{g}_\theta(L)x = [\bar{x}_0, \dots, \bar{x}_{K-1}]\theta$, $\bar{x}_k = T_k(\tilde{L})x = 2\tilde{L}\bar{x}_{k-1} - \bar{x}_{k-2}$ 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)

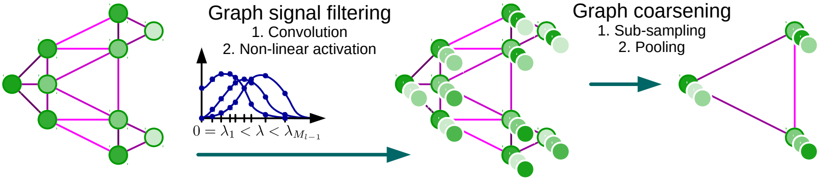
Gradients for backpropagation:

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

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

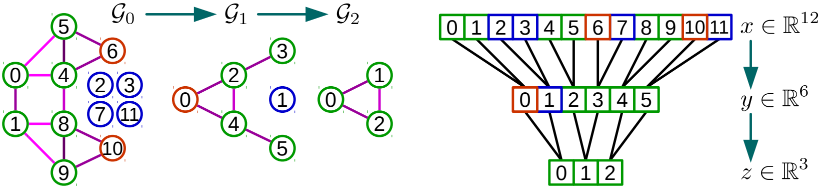
# Coarsening & Pooling

## Defferrard, Bresson, and Vandergheynst 2016



# Coarsening & Pooling

Defferrard, Bresson, and Vandergheynst 2016



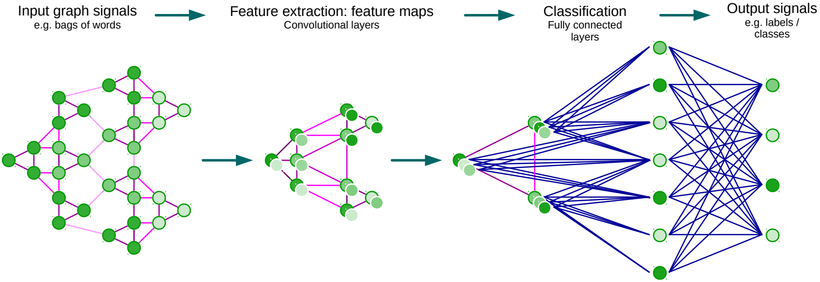

- Coarsening: Graclus / Metis
  - Approximate normalized cut minimization.

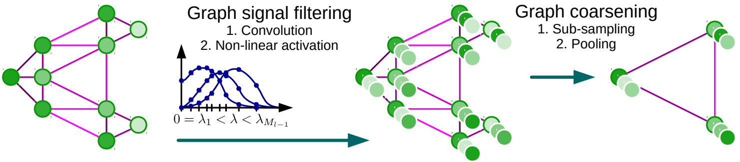
- Pooling: as regular 1D signals
  - Binary tree structured coarsened graphs.
  - Satisfies parallel architectures like GPUs.

- Activation: ReLU, LeakyReLU, maxout, tanh, sigmoid.

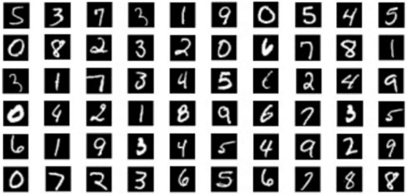
# Graph ConvNet architecture

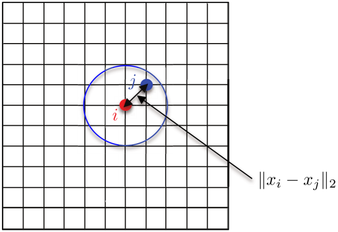
## Defferrard, Bresson, and Vandergheynst 2016



# MNIST: revisiting Euclidean ConvNets






$$W_{ij} = e^{-\|x_i - x_j\|_2^2/\sigma}$$

# MNIST: classification accuracy

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

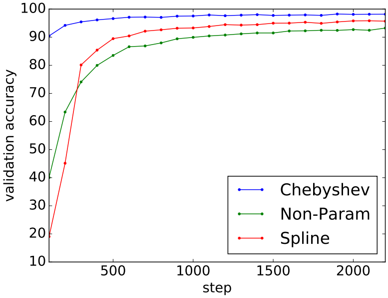
Table: Comparison to classical ConvNets.

Comparable to classical ConvNets,
and better than other parametrizations !

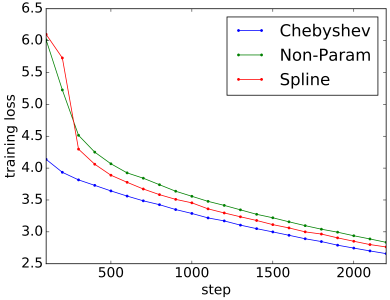
| | Accuracy | | |
|---|---|---|---|
| Architecture | Non-Param | Spline | Chebyshev |
| GC10 | 95.75 | 97.26 | 97.48 |
| GC32-P4-GC64-P4-FC512 | 96.28 | 97.15 | 99.14 |

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

# MNIST: convergence




Validation accuracy




Training loss

Faster convergence !

# Rotation invariance

Seo, Defferrard, Bresson, and Vandergheynst 2016

- Isotropic filters → rotation invariance
- Has been sought!
- Needs further investigations.

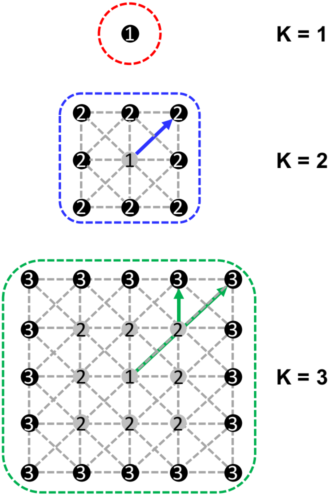
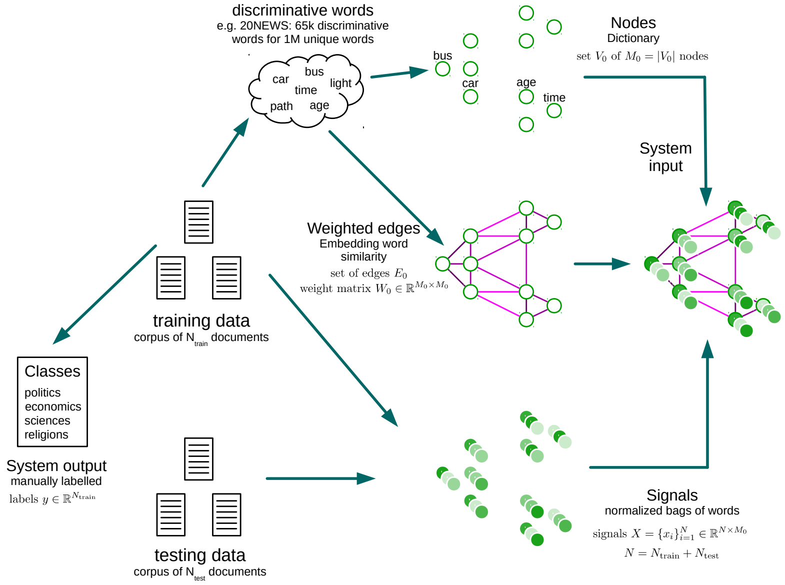



Figure: kNN = 8

# 20NEWS: structuring documents with a feature graph

Defferrard, Bresson, and Vandergheynst 2016



# 20NEWS: classification accuracies

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

Table: Accuracies of the proposed graph CNN and other methods on 20NEWS.

# Graph quality

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

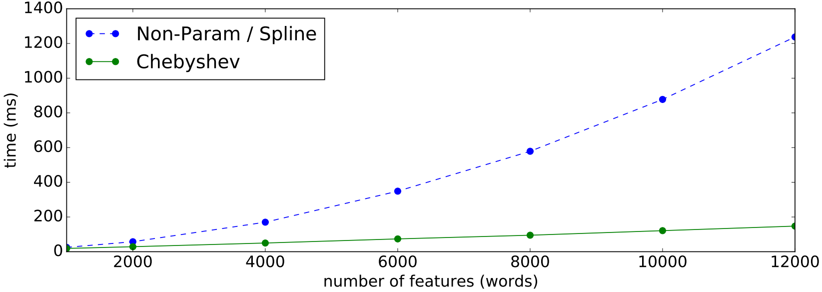
Accuracies of GC32 with different graph constructions on 20NEWS.

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

Classification accuracies with different graph constructions 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)

# Application: semi-supervised learning

Kipf and Welling 2016

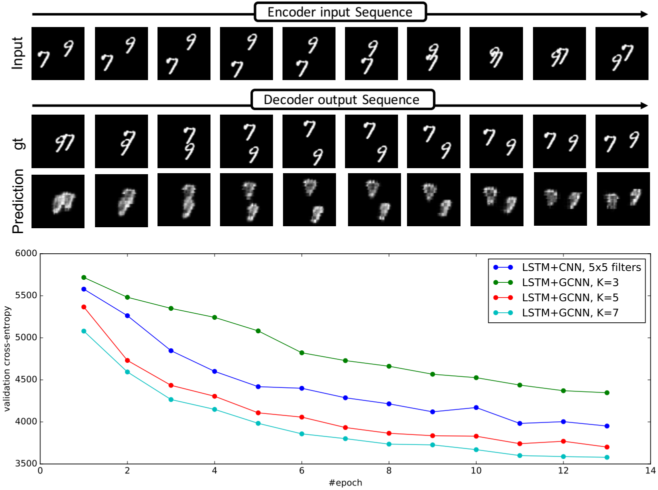
- Problem: Semi-supervised classification.
- Architecture: two graph convolutional layers.
- Filters: first-order approximation, i.e. $K = 1$.

| **Method** | **Citeseer** | **Cora** | **Pubmed** | **NELL** |
|---|---|---|---|---|
| ManiReg | 60.1 | 59.5 | 70.7 | 21.8 |
| SemiEmb | 59.6 | 59.0 | 71.1 | 26.7 |
| LP | 45.3 | 68.0 | 63.0 | 26.5 |
| DeepWalk | 43.2 | 67.2 | 65.3 | 58.1 |
| Planetoid | 64.7 (26s) | 75.7 (13s) | 77.2 (25s) | 61.9 (185s) |
| **GCN** (this paper) | **70.3** (7s) | **81.5** (4s) | **79.0** (38s) | **66.0** (48s) |

# Application: time-varying graph signals

Seo, Defferrard, Bresson, and Vandergheynst 2016

Stack a RNN on top of a graph ConvNet.



# Application: time-varying graph signals

Seo, Defferrard, Bresson, and Vandergheynst 2016

| Architecture | Representation | Parameters | Train Perplexity | Test Perplexity |
|---|---|---|---|---|
| Zaramba et al. code | embedding | 681,800 | 36.96 | 117.29 |
| Zaramba et al. code | one-hot | 34,011,600 | 53.89 | 118.82 |
| LSTM | embedding | 681,800 | 48.38 | 120.90 |
| LSTM | one-hot | 34,011,600 | 54.41 | 120.16 |
| LSTM, dropout | one-hot | 34,011,600 | 145.59 | 112.98 |
| GCRN-M1 | one-hot | 42,011,602 | 18.49 | 177.14 |
| GCRN-M1, dropout | one-hot | 42,011,602 | 114.29 | **98.67** |

# Conclusion

## Contributions

- Generalization of ConvNets to graph-structured data.
- Definition of fast and localized spectral filters on graphs.
- Same learning and computational complexities as classical ConvNets while being universal to any graph.

## Tools

- Spectral graph theory for convolution on graphs.
- Balanced cut model for graph coarsening (sub-sampling).
- Coarsened graphs organized as binary tree for fast pooling.

## Further research

- Model definition
- Applications

- Paper: Defferrard, Bresson and Vandergheynst, Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, NIPS, 2016.

- Code: `https://github.com/mdeff/cnn_graph`

# Thanks Questions?