Swiss Machine Learning Day (SMLD)

November 22, 2016

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)

Introduction

Data Science objective

Analyze and extract information for decision-making from large-scale and high-dimensional datasets.

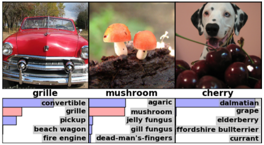
Machine Learning objective

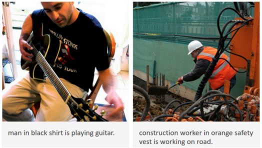
Extand convolutional neural networks to graph-structured data.

ConvNets are ubiquitous

First developed for Computer Vision [LeCun et al 98]

- Object recognition [Krizhevsky & Sutskever & Hinton 12]
- Image captioning [Karpathy & FeiFei 15]
- Image inpainting [Pathak & Efros et al 16]





Spreading outside CV

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

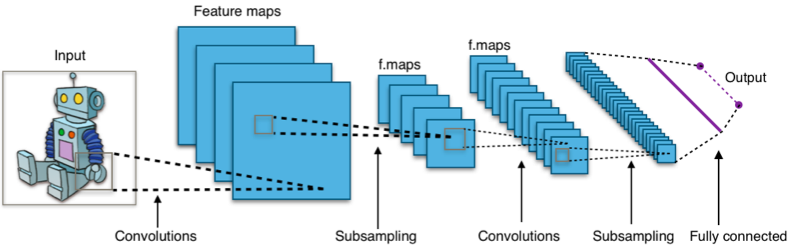
Why are they good ?

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

Statistical assumptions

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

Architecture



Ingredients

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

Feature extraction

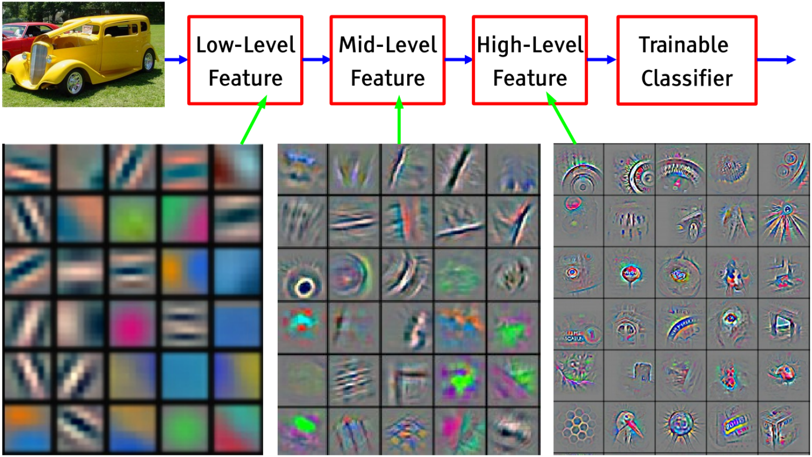


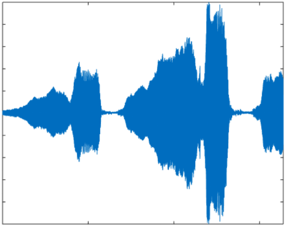
Figure: Features extracted from ImageNet [Zeiler & Fergus 2013]

Developed for data lying on Euclidean grids

All operations are well defined and computationally efficient:

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

Image (2D) Video (3D)



Sound (1D)

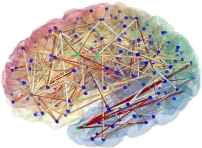
Non-Euclidean Data

Modeling versatility: graphs model heterogeneous pairwise relationships

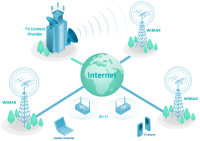
Examples of irregular / graph-structured data:

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

Social network



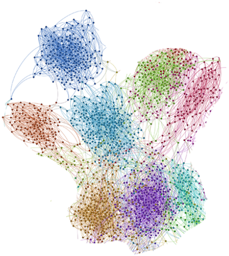
Brain structure



Telecommunication

Constructed graphs:

- Sample graph for e.g. semi-supervised learning.
- Feature graph to reduce computational complexity.



Alternative approach:

1. Embed nodes in an Euclidean space.
2. Use that embedding as features.

Reproduce the breakthrough of ConvNets beyond Computer Vision!

ConvNets on Graphs

Challenges

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

Contributions

- Generalizing ConvNets to general graph-structured data.
- Same computational complexity as classical ConvNets!

Tools

- Spectral graph theory for convolution on graphs.
- Balanced cut model for graph coarsening (sub-sampling).
- Graph pooling with binary tree structured coarsened graphs.

Related Works 1/2

- Local Receptive Fields (Gregor and LeCun 2010; Coates and Ng 2011)
  - Group features based upon similarity
  - No weight-sharing / convolution / stationarity
- Spatial approaches (Niepert, Ahmed, and Kutzkov 2016; Vialatte, Gripon, and Mercier 2016)
  - Define receptive field / neighborhood
  - Order nodes
- Geodesic CNNs on Riemannian manifolds (Masci et al. 2015)
  - Generalization of CNNs to non-Euclidean manifolds
  - Local geodesic system of polar coordinates to extract patches
  - Tailored for geometry analysis and processing

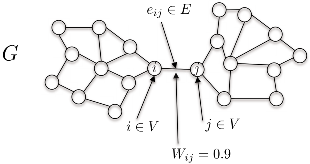
Related Works 2/2

- Graph Neural Networks (GNNs) (Scarselli et al. 2009)
  - Propagate node representations until convergence (RNN on graphs)
  - Representations used as features
- Diffusion-Convolutional Neural Networks (DCNNs) (Atwood and Towsley 2015)
  - Multiplication with powers (0 to H) of transition matrix
  - Diffused features multiplied by weight vector of support H
  - No pooling, followed by a fully connected layer
- Spectral Networks on Graphs (Bruna et al. 2013)
  - First spectral definition
  - Spline filter parametrization
  - Agglomerative method for coarsening

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

Hammond, Vandergheynst, and Gribonval 2011

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 Graph

Hammond, Vandergheynst, and Gribonval 2011

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

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

- 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

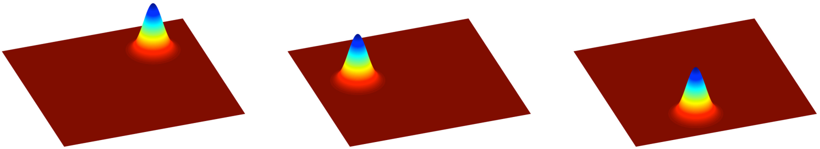


Figure: Localization on regular Euclidean grid.

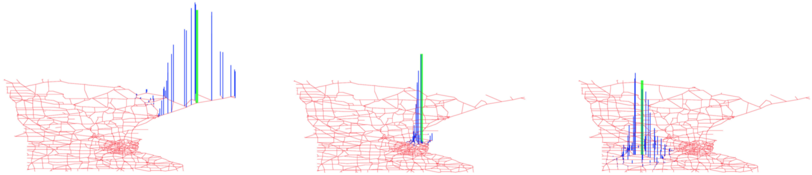


Figure: Localization on graph with $(\hat{g}_\theta(L)\delta_i)_j = (\hat{g}_\theta(L))_{i,j}$.

Recursive Formulation for Fast Filtering

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

- 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 $\mathcal{O}(K|\mathcal{E}|)$

Learning Filters

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

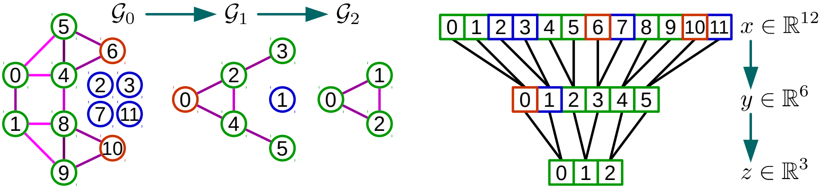
- $x_{s,i}$: feature map i of sample s
- $\theta_{i,j}$: trainable parameters
  ($F_{in} \times F_{out}$ vectors of Chebyshev coefficients)

Gradients for backpropagation:

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

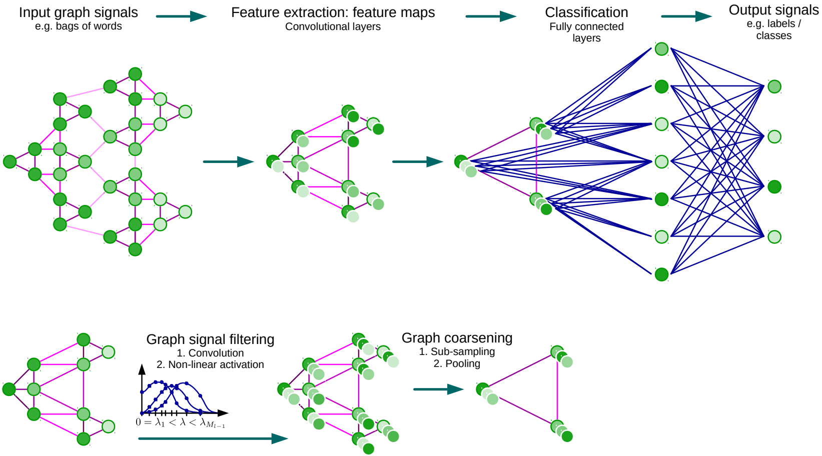
Overall cost of $\mathcal{O}(K|\mathcal{E}|F_{in}F_{out}S)$ operations

Coarsening & Pooling

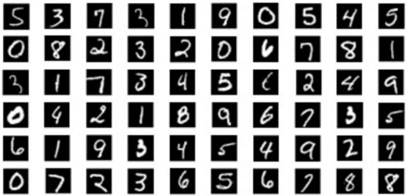


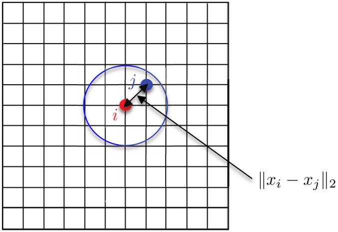
- Coarsening: Graclus / Metis
  - Normalized cut minimization
- Pooling: as regular 1D signals
  - Satisfies parallel architectures like GPUs
- Activation: ReLU (or tanh, sigmoid)

Architecture



Revisiting Euclidean ConvNets





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

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 CNNs.

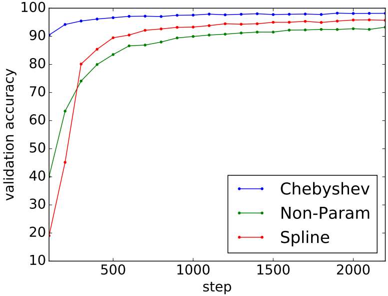
Comparable to classical CNNs and better than other parametrizations !

Isotropic filters $\rightarrow$ rotation invariance

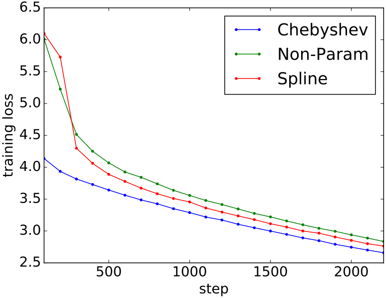
	Accuracy		
Architecture	Non-Param	Spline	Chebyshev
GC10	95.75	97.26	97.48
GC32-P4-GC64-P4-FC512	96.28	97.15	99.14

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

Convergence



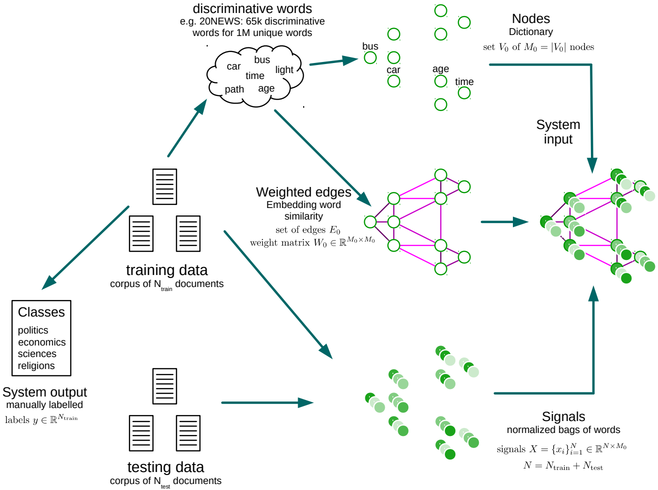
Validation accuracy



Training loss

Faster convergence !

Documents as graph signals



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

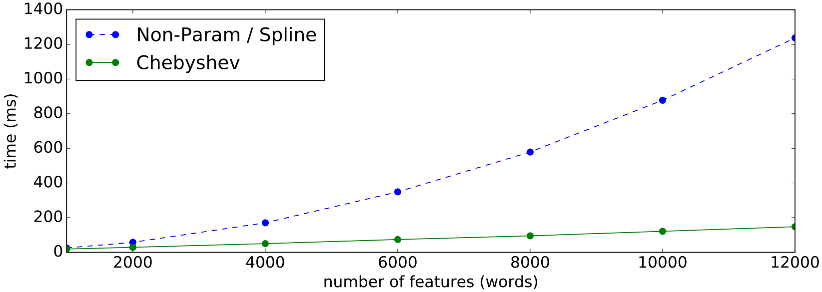
	word2vec			
bag-of-words	pre-learned	learned	approximate	random
67.50	66.98	68.26	67.86	67.75

Classification 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.

Training time



Make CNNs practical for graph signals !

Spline: $\hat{g}_\theta(\Lambda) = B\theta$ where B is the cubic spline basis (Bruna et al. 2013)

Semi-supervised learning

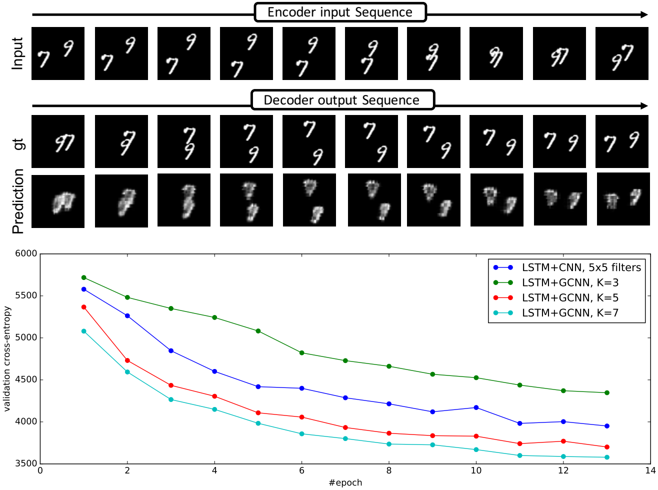
Kipf and Welling 2016

- Semi-supervised classification.
- Architecture: two graph convolutional layers
- First-order filters, 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)

Recurrent Neural Nets

Seo, Defferrard, Bresson and Vandergheynst 2016



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.

Further research

- Model definition
- Applications

Future applications

- Social networks (Facebook, Twitter)

- 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?