

Aussois, 2018-09-11



GRAPH SIGNAL PROCESSING WITH APPLICATIONS
TO 3D CLOUDS OF POINTS AND NEUROSCIENCE

DEEP LEARNING ON GRAPHS

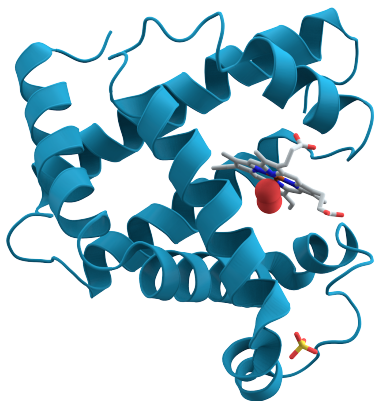
Michaël DEFFERRARD



ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Motivation

$x =$



$$y = f(x) = \begin{cases} \text{"toxic"} \\ \text{"non-toxic"} \end{cases}$$

Goal: learn the **unknown** function f , using both **structure** and **features**.

Learning 101

- ▶ ideal unknown function: $y = f(x)$
- ▶ parameterized approximation: $y \approx f_{\theta}(x)$, where θ are the parameters to be learned
- ▶ learning a function: $\min_{\theta} E(y, f_{\theta}(x))$
- ▶ example of energy/loss/objective: $E(y, \hat{y}) = \|y - \hat{y}\|_2^2 + \|\hat{y}\|_1$
- ▶ in our case, f is a graph neural network
- ▶ learning by gradient descent: $\theta^{t+1} = \theta^t - \frac{\partial E}{\partial \theta}$

Structure and features

Structure: graph (or network)

- ▶ Graph: a set of nodes (vertices) and a set of pairwise relations (edges)
- ▶ Relations: interactions, similarity, geometry
- ▶ Structure from features: similarity between features
- ▶ Structure only \Rightarrow graph embeddings (e.g., word2vec, node2vec)

Features: data on the graph (or signal)

- ▶ Features: set of characteristics (or properties) about each node
- ▶ Features from structure: degree, centrality, modularity, etc.
- ▶ Features only \Rightarrow traditional ML

Using the structure

Extrinsic: embed the graph in an Euclidean space.

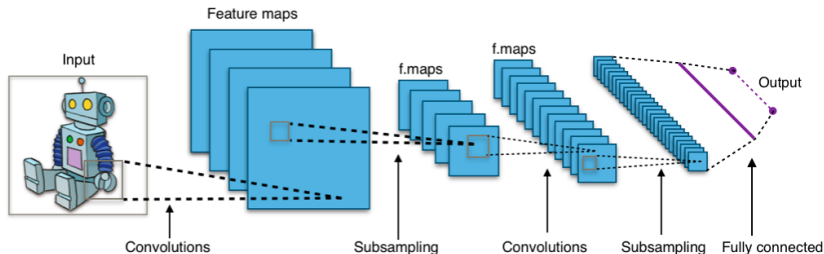
- ▶ Learn a vector representation of each node.
- ▶ Use that embedding as additional features for a classifier.

Intrinsic: a Neural Net defined on graphically structured data.

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

Convolutional Neural Networks

Main benefit (over MLPs): they **exploit the structure** of the data.



Key properties:

- ▶ **Convolutional**: translation equivariance (stationarity).
- ▶ **Localized**: deformation stability & compact filters (independent of input size n).
- ▶ **Multi-scale**: hierarchical features extracted by multiple layers (compositionality).
- ▶ $\mathcal{O}(n)$ computational complexity.

ConvNets on graphs

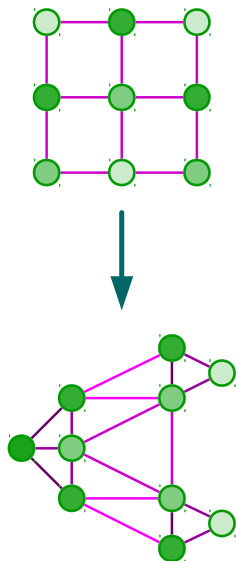
Graphs vs Euclidean grids:

- ▶ Irregular sampling.
- ▶ Weighted edges.
- ▶ No orientation or ordering (in general)
→ permutation invariance.

Ingredients:

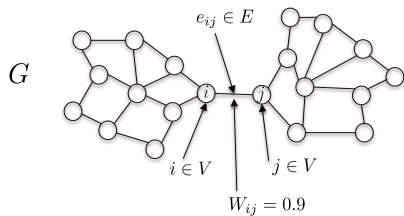
- ▶ Convolution (local)
- ▶ Non-linearity (point-wise)
- ▶ Down-sampling (global / local)
- ▶ Pooling (local)

Challenge: efficient formulation of convolution and down-sampling on graphs.



Notation

$\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$: 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):

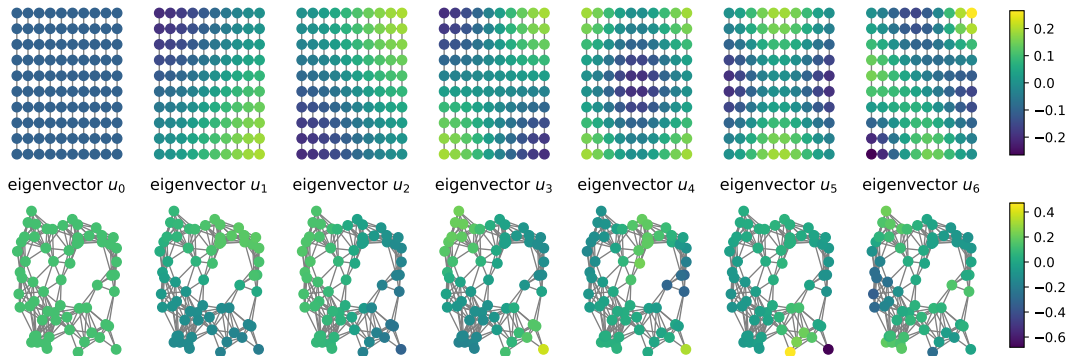
- ▶ combinatorial Laplacian $L = D - W \in \mathbb{R}^{n \times n}$
- ▶ normalized Laplacian $L = I_n - D^{-1/2} W D^{-1/2} \in \mathbb{R}^n$

Graph Fourier basis

Shuman, Narang, Frossard, Ortega, and Vandergheynst 2013

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

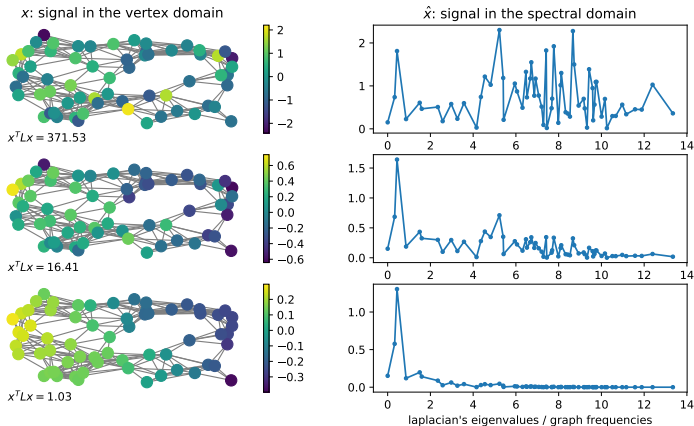
- ▶ Graph Fourier basis $U = [u_1, \dots, u_n] \in \mathbb{R}^{n \times n}$
- ▶ Graph “frequencies” $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n) \in \mathbb{R}^{n \times n}$



Graph Fourier Transform

Shuman, Narang, Frossard, Ortega, and Vandergheynst 2013

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

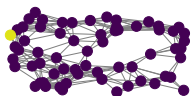


Filtering with convolution on graphs

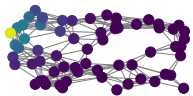
Shuman, Narang, Frossard, Ortega, and Vandergheynst 2013

$$y = x *_G g = U(\hat{g} \odot U^T x) = U \begin{bmatrix} \hat{g}(\lambda_1) & & 0 \\ & \ddots & \\ 0 & & \hat{g}(\lambda_n) \end{bmatrix} U^T x = U \hat{g}(\Lambda) U^T x = \hat{g}(L)x$$

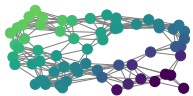
$y = \hat{g}(L)\delta_{10}$: localized on sensor



$y^T L y = 367.18$

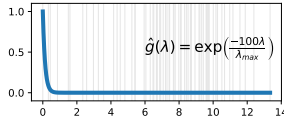
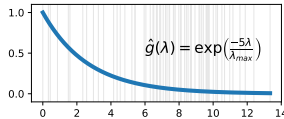
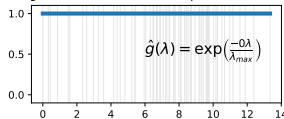


$y^T L y = 6.83$



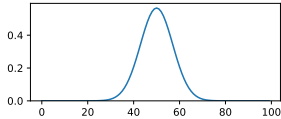
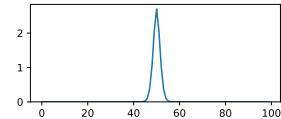
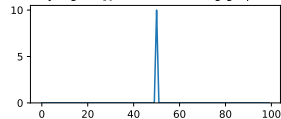
$y^T L y = 0.00$

$\hat{g}(\lambda)$: filter defined in the spectral domain



λ : laplacian's eigenvalues / graph frequencies

$y = \hat{g}(L)\delta_{50}$: localized on ring graph



No translation \Rightarrow no *real* convolution

Perraudin and Vandergheynst 2017

1D Euclidean convolution:

$$(x * g)[i] = \sum_{j=-\infty}^{\infty} x[j]g[i-j] = \langle T_i g, x \rangle,$$

where $T_i g$ is a **translation** of the signal g by i steps.

Graph convolution:

$$(x *_G g)[i] = (\hat{g}(L)x)[i] = \langle \mathcal{T}_i \hat{g}(L), x \rangle = \langle \hat{g}(L) \delta^i, x \rangle,$$

where $\mathcal{T}_i \hat{g}$ is the **localization** of the kernel \hat{g} at node v_i .

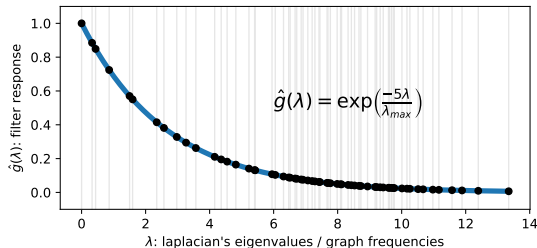
We convolve x with a kernel \hat{g} , not a graph signal.

Spectral filtering of graph signals

Defferrard, Bresson, and Vandergheynst 2016

Non-parametric filter, can learn any filter (n degrees of freedom):

$$\hat{g}_\theta(\Lambda) = \text{diag}(\theta), \quad \theta \in \mathbb{R}^n \Rightarrow y = U \text{diag}(\theta) U^\top x$$



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

Polynomial parametrization

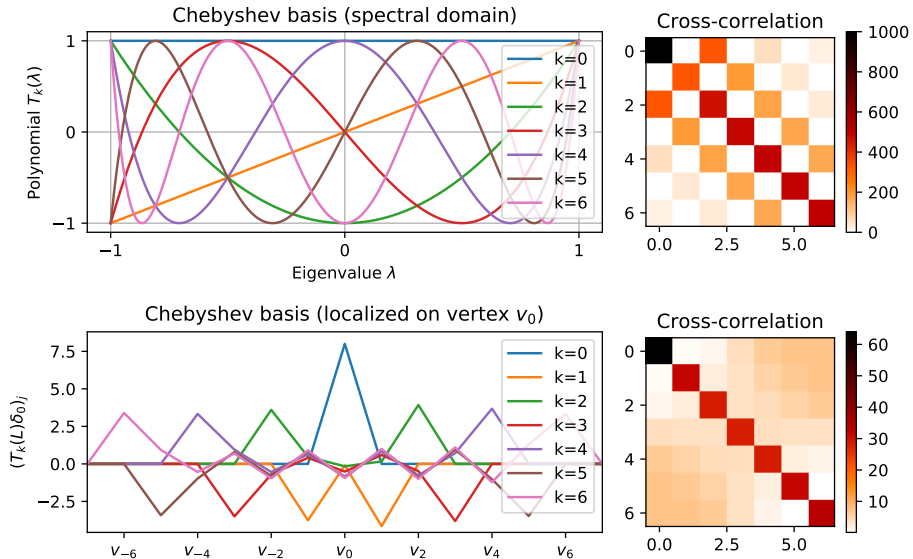
Defferrard, Bresson, and Vandergheynst 2016

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

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

- ▶ Can learn any K -localized filter.
- ▶ Allows a distributed implementation: only accesses the K -neighborhood.
- ▶ K -localized
- ▶ Learning complexity is $\mathcal{O}(K)$
- ▶ Computational complexity is $\mathcal{O}(K|\mathcal{E}|)$ (same as classical ConvNets!)

Chebyshev polynomials



Fast implementation by recursion

Defferrard, Bresson, and Vandergheynst 2016

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

Recurrence: $\bar{x}_0 = x$

$$\bar{x}_1 = \tilde{L}x$$

$$\bar{x}_k = T_k(\tilde{L})x = 2\tilde{L}\bar{x}_{k-1} - \bar{x}_{k-2}$$

- ▶ Any polynomial can be used. They all have the same representative power. Optimization difficulty might vary.
- ▶ Any matrix can be used instead of the Laplacian L , including the adjacency matrix, or even a non-symmetric adjacency or “Laplacian”.
- ▶ The learned filter parameters θ can be transferred across graphs, i.e., used with different L .

Spatial vs Spectral

Defferrard, Bresson, and Vandergheynst 2016

Convolution on graphs can be **spectrally motivated**.

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

In the absence of an $O(n \log n)$ Fast Fourier Transform (FFT), which only exists for specific domains, that is however too expensive. $O(n^3)$ operations for the EVD, plus $O(n^2)$ operations per forward and backward pass.

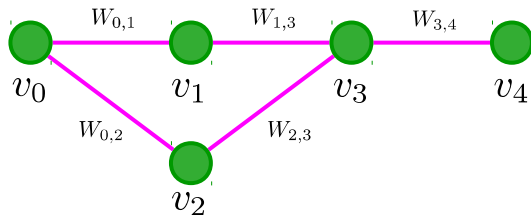
With polynomials, the convolution is however **spatially implemented**.

$$y = \hat{g}_\theta(L)x = \sum_k \theta_k L^k x = \sum_k \tilde{\theta}_k T_k(\tilde{L})x$$

Leading to many other interpretations: message-passing between nodes, local tangent planes, permutation invariant aggregation, etc.

Weights of paths

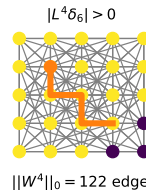
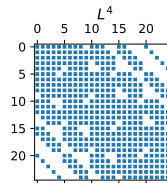
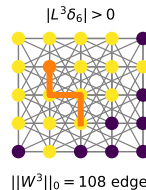
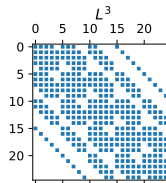
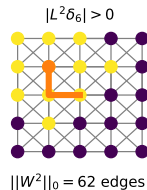
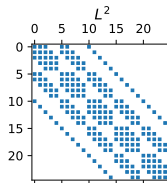
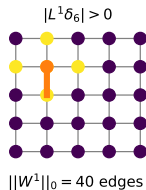
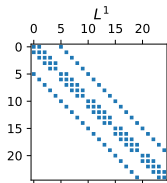
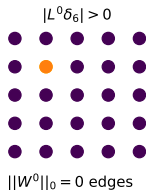
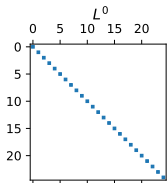
$(W^k)_{ij}$ is the sum of all weighted paths of length k between v_i and v_j .



- ▶ A path is an ordered set of nodes. Example: (v_2, v_3, v_4) .
- ▶ $p_{ij}^k = \{(v_i, \dots, v_j), \dots, (v_i, \dots, v_j)\}$ is the set of all paths of length k between v_i and v_j . Example: $p_{0,3}^2 = \{(v_0, v_1, v_3), (v_0, v_2, v_3)\}$.
- ▶ Path weight $(W^k)_{ij} = \text{weight}(p_{ij}^k) = \sum_{\text{paths}} \prod_{\text{edges } (v_k, v_l)} W_{kl}$.
Example: $(W^2)_{0,3} = (W_{0,1} \cdot W_{1,3}) + (W_{0,2} \cdot W_{2,3})$.

Neighborhoods

L^k defines the k -neighborhood



Localization: $d_G(v_i, v_j) > K$ implies $(L^K)_{ij} = 0$

Learned combination of neighboring values

$y = \sum_k \theta_k L^k x$ is a linear transformation, where the coefficients are:

- ▶ the learned parameter θ_k ,
- ▶ the k -neighborhood encoded by L^k .

Weighted sum of neighborhoods:

$$y_i = \sum_k \theta_k \bar{x}_k = \underbrace{\theta_0 x}_{\text{own value}} + \underbrace{\theta_1 \bar{x}_1}_{\text{1-neighborhood}} + \underbrace{\theta_2 \bar{x}_2}_{\text{2-neighborhood}} + \cdots + \underbrace{\theta_K \bar{x}_K}_{\text{K-neighborhood}}$$

- ▶ Monomials in L : $\bar{x}_k = L^k x$
- ▶ Monomials in W : $\bar{x}_k = W^k x$
- ▶ Chebyshev polynomials in L : $\bar{x}_0 = x, \bar{x}_1 = \tilde{L}x, \bar{x}_k = T_k(\tilde{L})x = 2\tilde{L}\bar{x}_{k-1} - \bar{x}_{k-2}$

Aggregation function

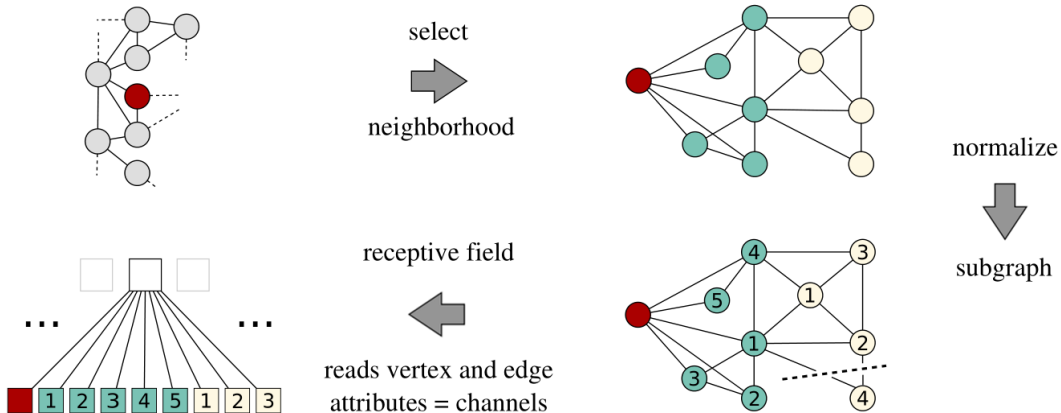
$y = f(x) = \sum_k \bar{x}_k$ is learning how to combine the values \bar{x}_k from the k -neighborhood.
The *basic unit* is the neighborhoods, not the nodes.

What else can be done? Any function f that is invariant to the number of neighbors and their permutation.

Goal: map a varying-length representation to a length K representation for $\mathcal{O}(K)$ learning complexity.

Spatial approach: node ordering

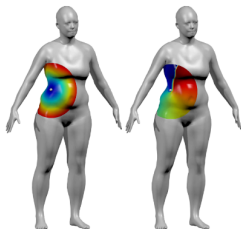
Niepert, Ahmed, and Kutzkov 2016



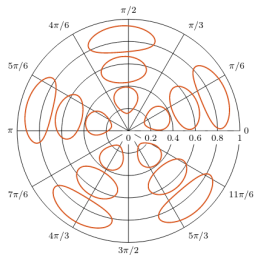
- ▶ anisotropic filters
- ▶ require an ordering of the nodes

Spatial approach: patches on the manifold's tangent plane

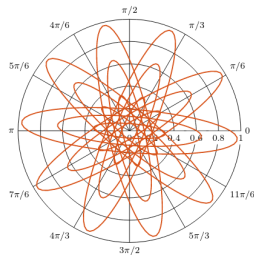
Monti, Boscaini, Masci, Rodola, Svoboda, and Bronstein 2017



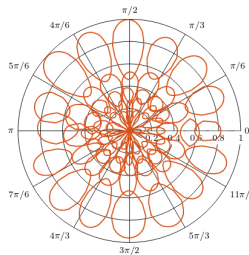
Polar coordinates ρ, θ



GCNN



ACNN



MoNet

- anisotropic filters
- manifolds only

The need to consider multiple scales

Most data on large graphs exhibit **patterns at multiple scales**.

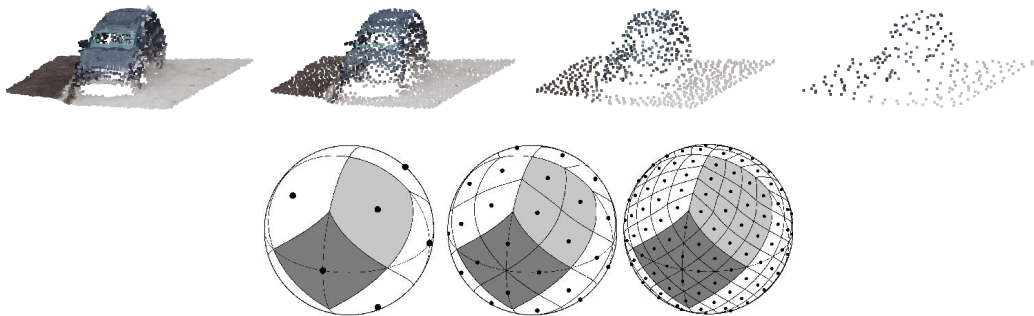
Some filters thus need to have larger receptive fields to capture longer-range dependencies. This can be achieved by:

1. increasing the size of the filters (the polynomial order),
2. increasing the number of layers,
3. down-sampling the domain (pooling).

While we can easily do (1) and (2), it can drastically increase the number of parameters to learn. For now, we don't yet have a generic and functional approach to (3).

Coarsening: hierarchical representation

Graph coarsening is certainly an answer to the down-sampling problem.

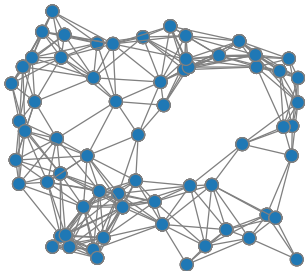


- Easy and well-defined when the domain has a hierarchical structure.

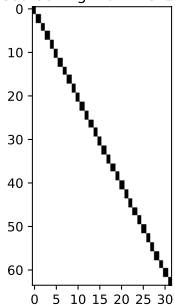
Coarsening: greedy local approach

Defferrard, Bresson, and Vandergheynst 2016

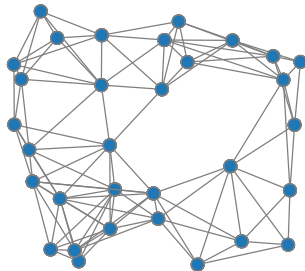
Input graph: $|V| = 64$, $|E| = 303$



Coarsening matrix $C \in \mathbb{R}^{64 \times 32}$



Coarsened graph: $|V| = 32$, $|E| = 222$

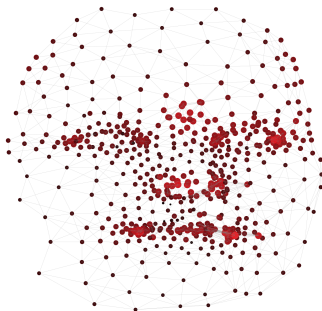


- ▶ Greedy node merging (e.g., Graclus, Metis) works well for regular graphs.
- ▶ Can be done as pre-processing.
- ▶ Conditioned on the structure only.
- ▶ Much harder on non-regular graphs.

Learned coarsening: an attention mechanism

Defferrard and Loukas 2018

hard combinatorial problem \Rightarrow learn a **continuous relaxation** of the operation



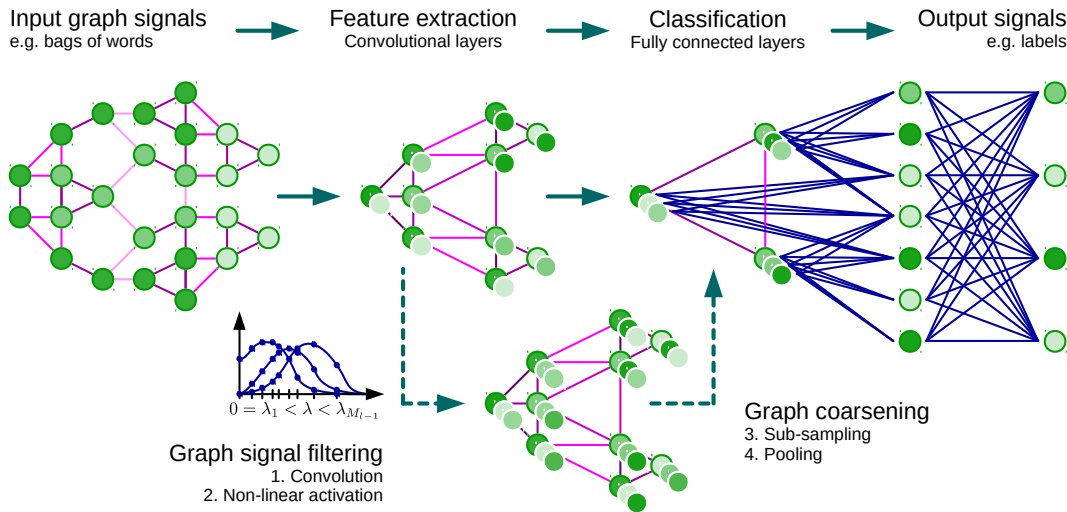
Conditioned on:

1. the structure
2. the features
3. the task

introspection!

Graph ConvNet architecture

Defferrard, Bresson, and Vandergheynst 2016



Multiple kinds of problems: combination of data and tasks

Graphs that model discrete relations

- ▶ Social networks
- ▶ Graph of citations or hyperlinks
- ▶ Molecules (proteins)
- ▶ Knowledge graphs

Graphs that represent sampled manifolds

- ▶ Meshes (shapes, surfaces)
- ▶ Point clouds
- ▶ Data on spheres (planets, sky)
- ▶ Traffic on roads

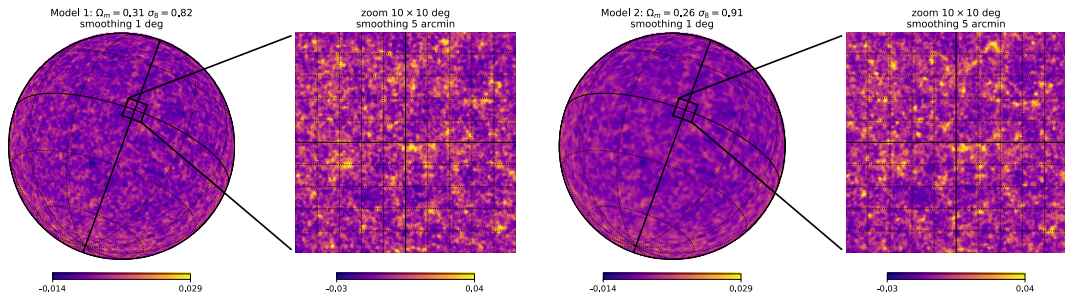
Tasks:

- ▶ Node classification or regression (semi-supervised learning)
- ▶ Graph classification or regression
- ▶ Signal classification or regression

Cosmology: data & problem

Perraudin, Defferrard, and Kacprzak 2018

- ▶ Cosmologists devise models of how the universe works.
- ▶ We only get to observe one real universe.
- ▶ Problem: which simulation is closest to the real thing? A signal classification task.

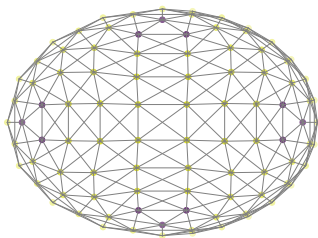


Two mass maps generated from different cosmological parameters.

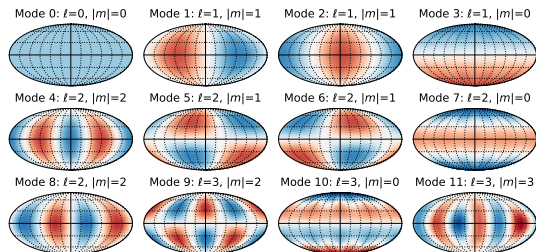
Cosmology: graph

Perraudin, Defferrard, and Kacprzak 2018

- ▶ Data lives on the sky, a sphere.
- ▶ The sphere is discretized, and can be represented by a graph.
- ▶ Numerous kind of spherical sky maps in cosmology and astrophysics.
Cosmic microwave background, galaxy clustering, gravitational lensing.



Sphere discretized by graph.

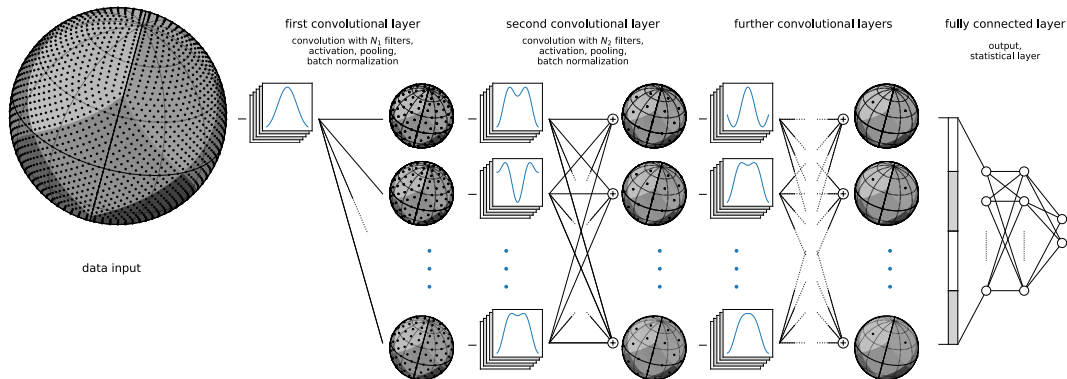


Fourier modes resemble spherical harmonics.

Cosmology: model

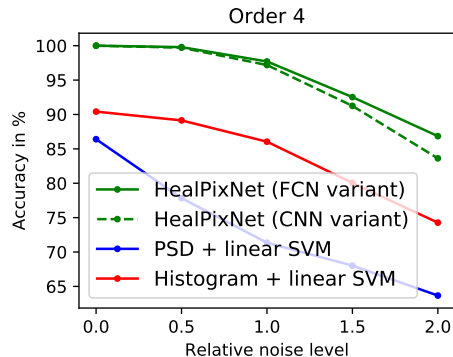
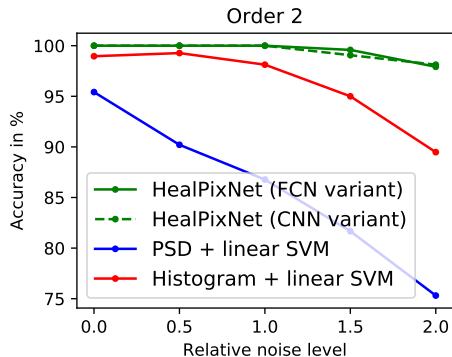
Perraudin, Defferrard, and Kacprzak 2018

A classical CNN or FCN architecture, but on the sphere, which is modeled by a graph.



Cosmology: results

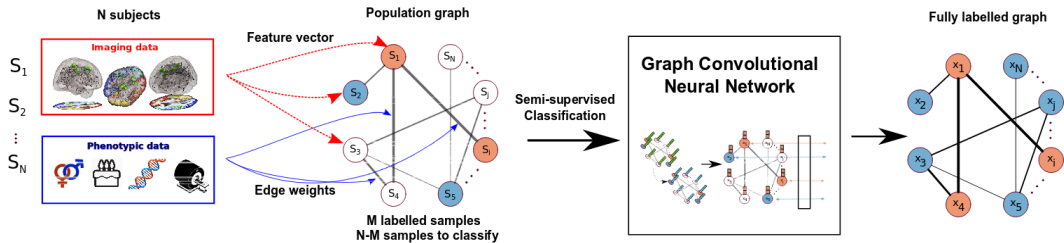
Perraudin, Defferrard, and Kacprzak 2018



Significantly better than two standard benchmarks used in cosmology.

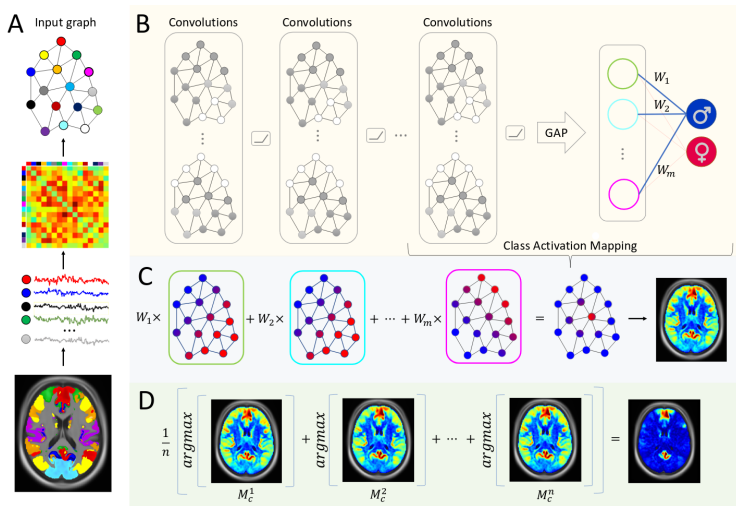
Neuroscience: predict diseases using a population graph

Pariset, Ktena, Ferrante, Lee, Guerrero, Glocker, and Rueckert 2018



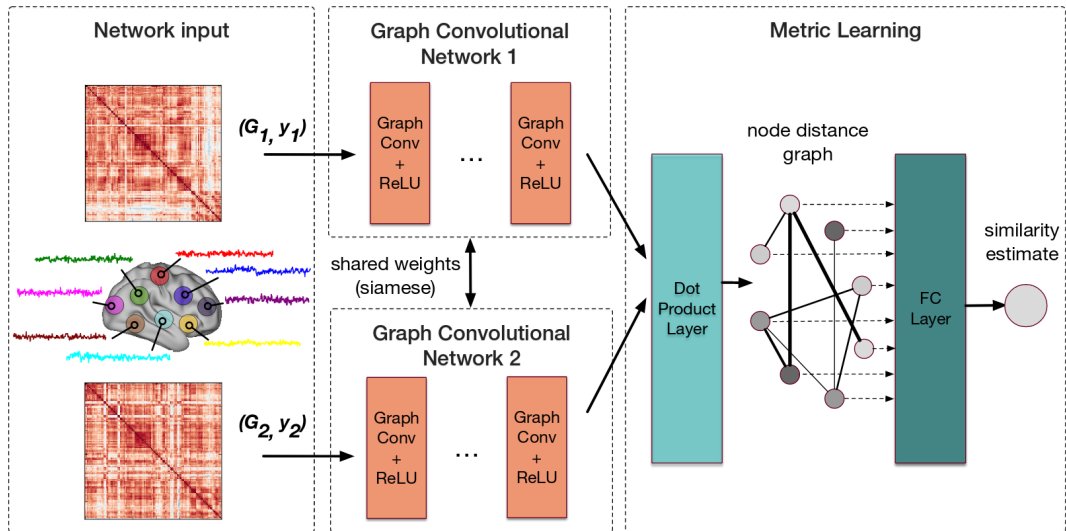
Neuroscience: find which regions are important (saliency map)

Arslan, Ktena, Glocker, and Rueckert 2018



Neuroscience: learn to compare functional brain networks

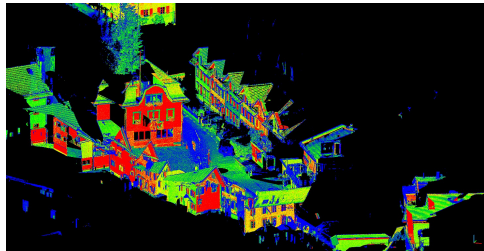
Ktena, Parisot, Ferrante, Rajchl, Lee, Glocker, and Rueckert 2017



Point clouds: semantic segmentation (teaser)



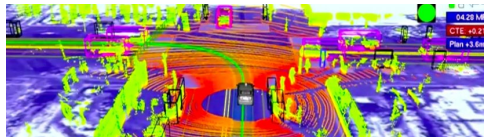
remote sensing / surveying



outdoor mapping



indoor mapping



autonomous driving

Conclusion

Successes:

- ▶ Convolution operation mostly solved (many formulations have been proposed for specific tasks) and understood (with multiple interpretations, including message-passing, local aggregation function, attention).
- ▶ Applications to many scientific and industrial problems

Challenges:

- ▶ Multiple scales, down-sampling, coarsening.
- ▶ Better understanding of the method – problem fit.

Aussois, 2018-09-12



GRAPH SIGNAL PROCESSING WITH APPLICATIONS
TO 3D CLOUDS OF POINTS AND NEUROSCIENCE

SEGMENTATION OF POINT CLOUDS



Michaël DEFFERRARD

Joint work with Alexandre CHERQUI and
Frank DE MORSIER.



ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Motivation: remote sensing



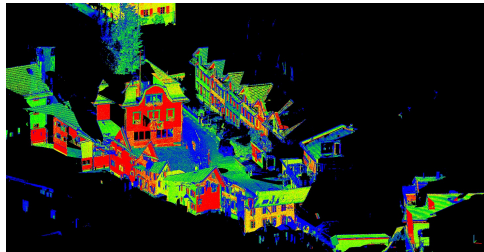
Motivation: remote sensing



Many applications



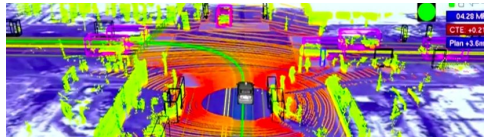
remote sensing / surveying



outdoor mapping



indoor mapping



autonomous driving

Different classification problems

Goal: assign class labels.

- ▶ granularity
- ▶ class vs instance



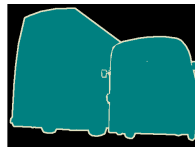
input¹



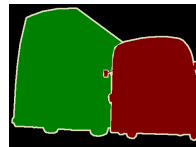
classification



object recognition



semantic seg.



instance seg.

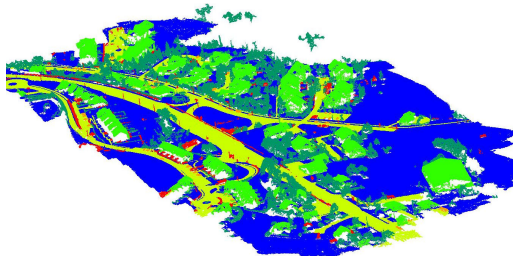
¹Image source: https://sthalles.github.io/assets/deep_segmentation_network/object_class_segmentation.png

Data

input a set of features associated to a set of points
output a label associated to each point

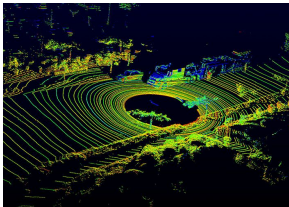


x,y,z coordinates with RGB colors

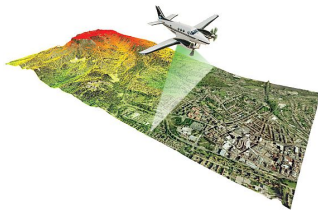


class labels

Data acquisition



ground LIDAR



aerial LIDAR



aerial images

Our case, aerial images:

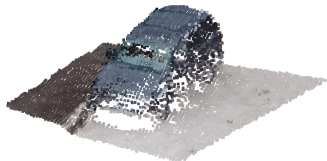
- ▶ Drones take aerial pictures of the ground.
- ▶ Each point is photographed multiple times from different point-of-views.
- ▶ Point cloud constructed by photogrammetry.

Graph

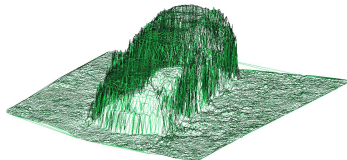
Cherqui, Morsier, and Defferrard 2018

A graph gives:

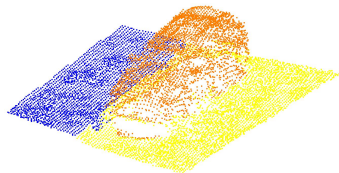
- ▶ Neighborhood information, needed for consistent labeling.
- ▶ A support, needed for efficient computation.



RGB features



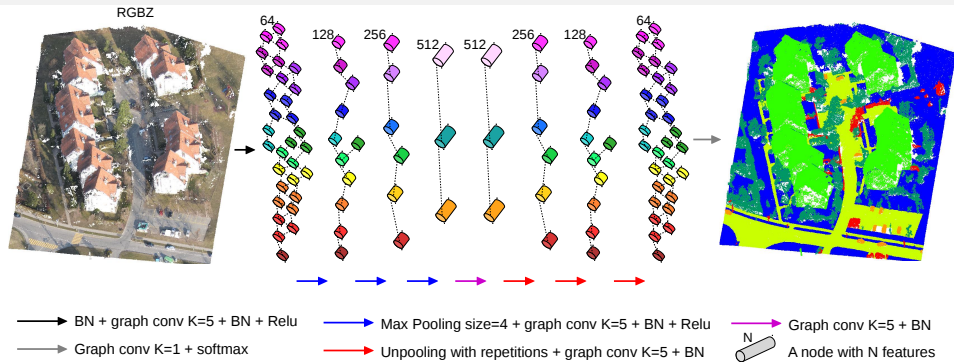
graph



labels

Model

Cherqui, Morsier, and Defferrard 2018



Characteristics:

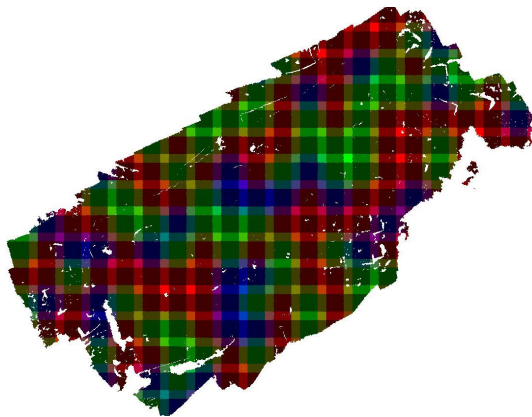
- ▶ Dense prediction.
- ▶ *Reason* at multiple scales.
- ▶ Local decisions.

Main difficulties:

- ▶ Large number of points.
- ▶ Training samples are of varying sizes.

Data preparation

Cherqui, Morsier, and Defferrard 2018

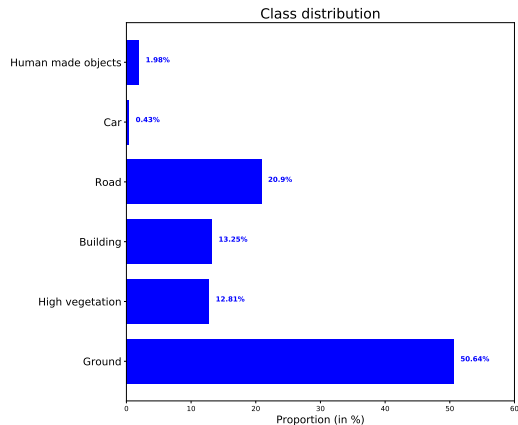


- ▶ tiling: $36m \times 36m$ ($48m \times 48m$ with context)
- ▶ split: 50% training tiles (green), 16% validation tiles (blue), 35% test tiles (red)

Results with RGBZ

Cherqui, Morsier, and Defferrard 2018

Model	Accuracy	
	Overall (micro)	Mean (macro)
Random Forest	75%	53%
Graph ConvNet	86%	68%



Results

Cherqui, Morsier, and Defferrard 2018

Random forest baseline

True label	Ground	807919	111645	20379	20821	620	4548
	High veg.	99011	134143	6184	2899	200	1159
	Building	19336	8995	198616	22489	1716	6586
	Road	42386	4866	75775	366655	2342	15912
	Car	1891	260	5030	2175	1412	636
	Human made obj.	12538	3360	11867	9200	436	3923
			Ground	High veg.	Building	Road	Car
		Predicted label					

Graph ConvNet

Ground	High veg.	Building	Road	Car	Human-made obj.
812954	90364	24305	28224	1146	8939
24180	210247	3780	2377	15	2997
8065	7025	226223	10405	310	5710
25005	7214	51894	418984	945	3894
220	291	2562	2191	3894	2246
6878	9103	8950	2528	1325	12540
Ground	High veg.	Building	Road	Car	Human-made obj.
Predicted label					

Conclusion

- ▶ graph signal processing fits well point clouds data
- ▶ graph neural networks can be used to solve semantic segmentation
- ▶ semantic segmentation of point clouds has many applications
- ▶ many tasks can potentially be learned: denoising, registration, etc.

Slides <https://doi.org/10.5281/zenodo.1414546>

Papers Defferrard, Bresson, Vandergheynst, Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, NIPS, 2016.
Seo, Defferrard, Bresson, Vandergheynst, Structured Sequence Modeling with Graph Convolutional Recurrent Networks, arXiv, 2017.
Defferrard, Loukas, Learning Where to Look on a Manifold with Geometric Attention Layers, in preparation, 2018.
Perraudin, Defferrard, Kacprzak, HealpixNet: Efficient spherical Convolutional Neural Network with HEALPix sampling for cosmological applications, in preparation, 2018.

Codes https://github.com/mdeff/cnn_graph
<https://github.com/youngjoo-epfl/gconvRNN>
<https://github.com/SwissDataScienceCenter/HealpixNet>

GSP in Python <https://github.com/epfl-lts2/pygsp>

Thanks Questions?