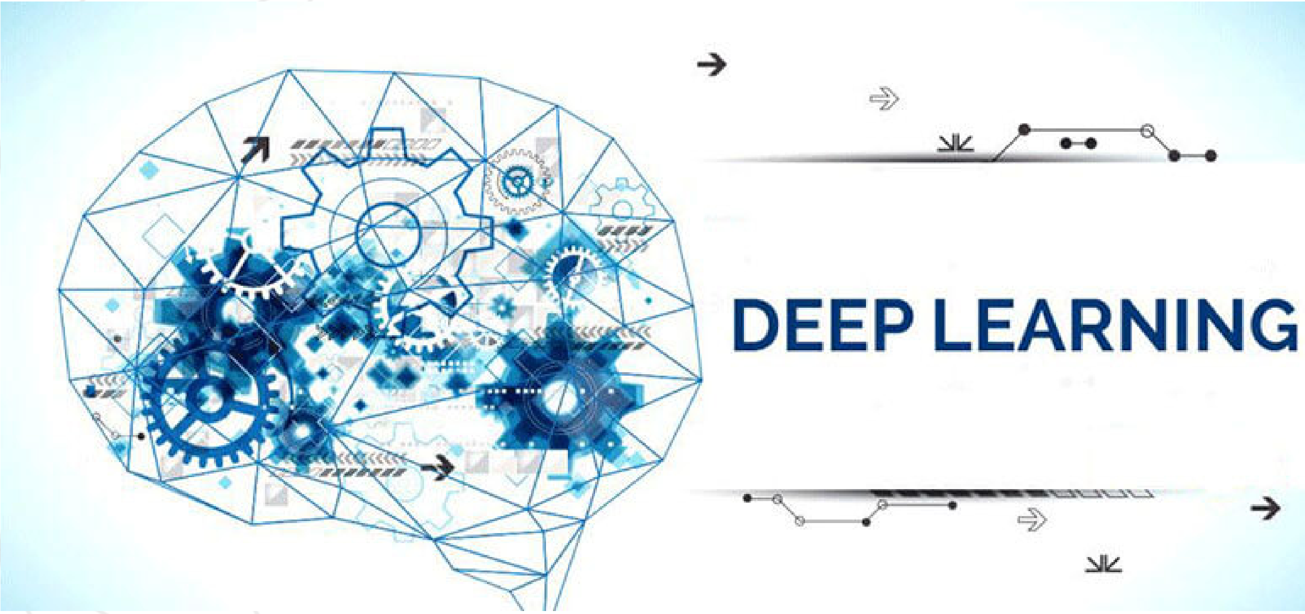
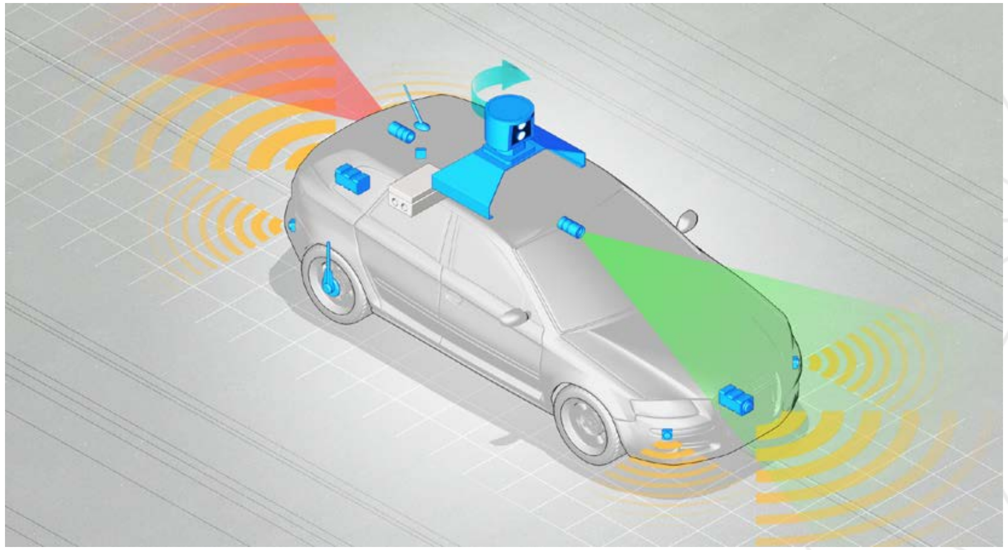
Graph-convolutional neural networks

Diego Valsesia

Télécom ParisTech
7/3/2019

Obligatory Deep Learning Preamble





Forbes

3 Reasons To Believe The Singularity Is Near

Obligatory Deep Learning Preamble

POLITECNICO DI TORINO

SmartData

- 1989: training a convolutional neural network with backprop

Handwritten Digit Recognition with a Back-Propagation Network

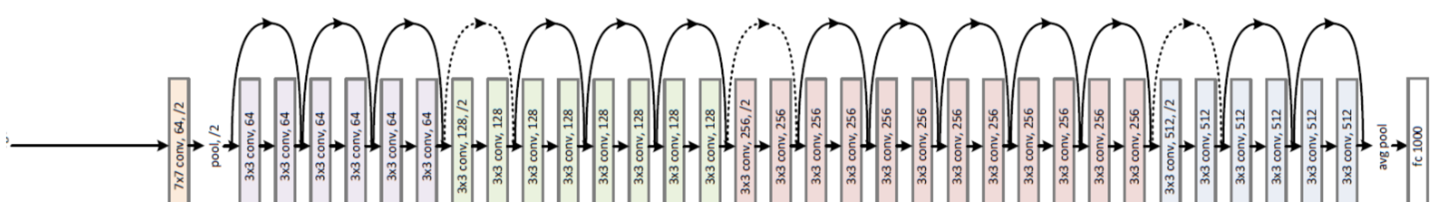
Y. Le Cun, B. Boser, J. S. Denker, D. Henderson,
R. E. Howard, W. Hubbard, and L. D. Jackel
AT&T Bell Laboratories, Holmdel, N. J. 07733

Obligatory Deep Learning Preamble

- **2010s**:
  - More data

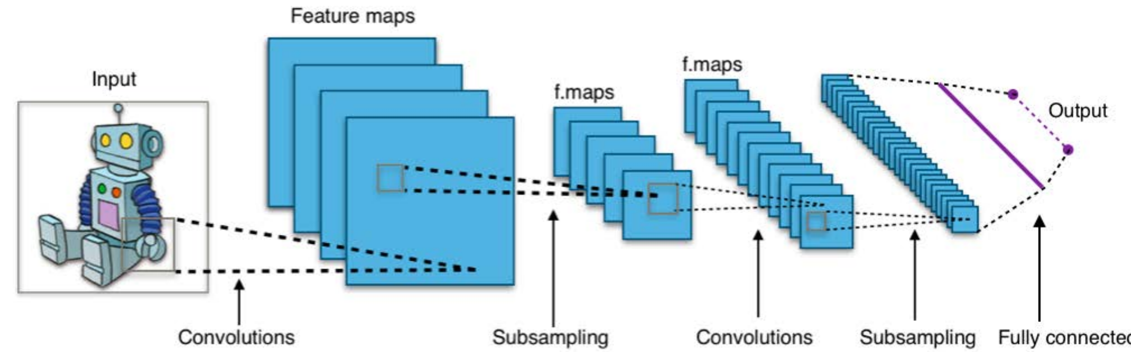
  - More computational power

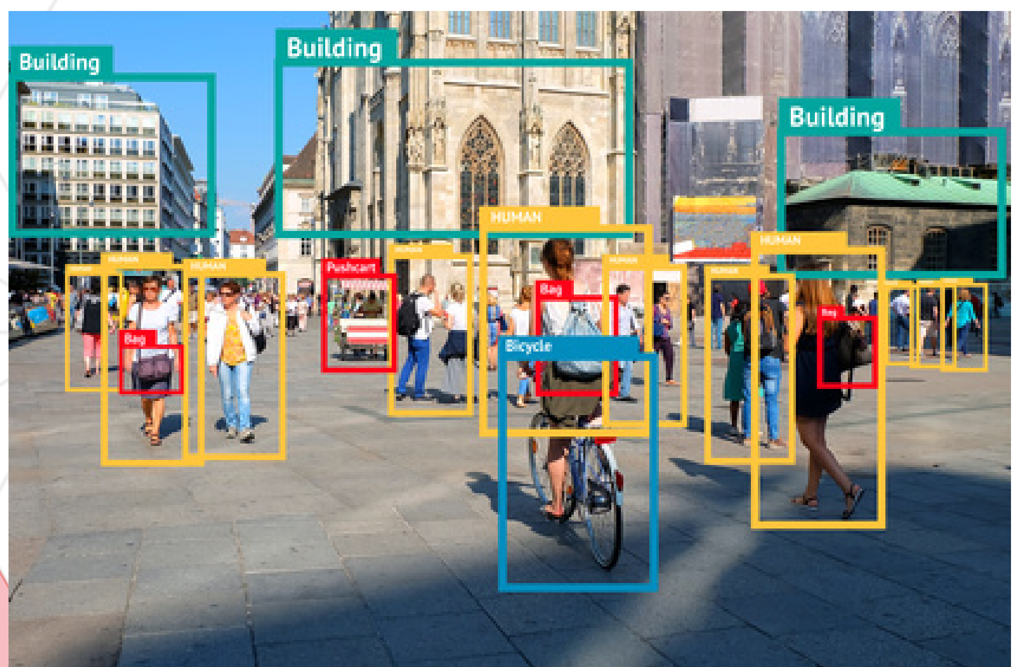
  - New techniques to train deeper networks



Convolutional Neural Networks

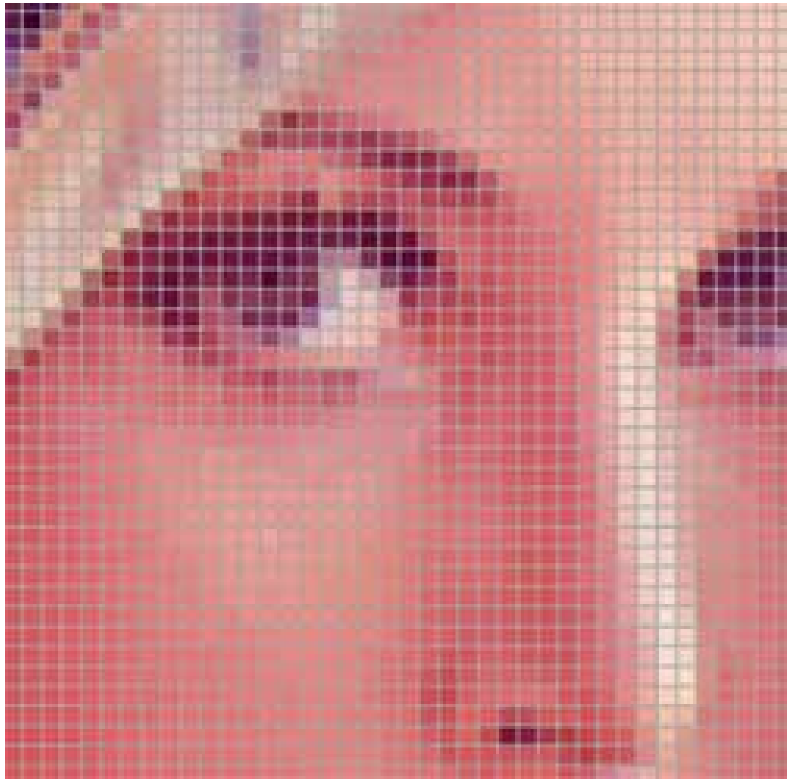
- Very successful
- Workhorse of vision problems



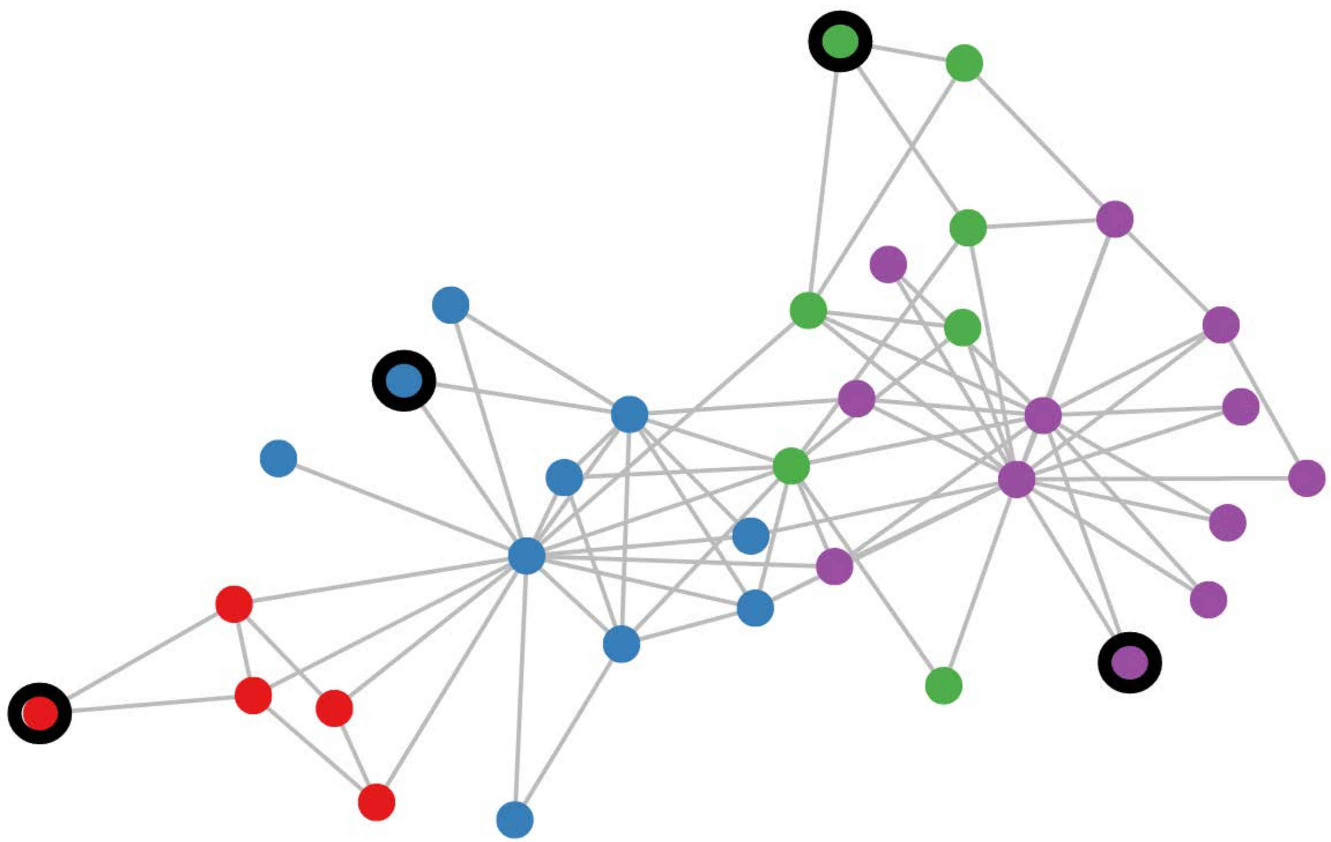


Convolutional Neural Networks

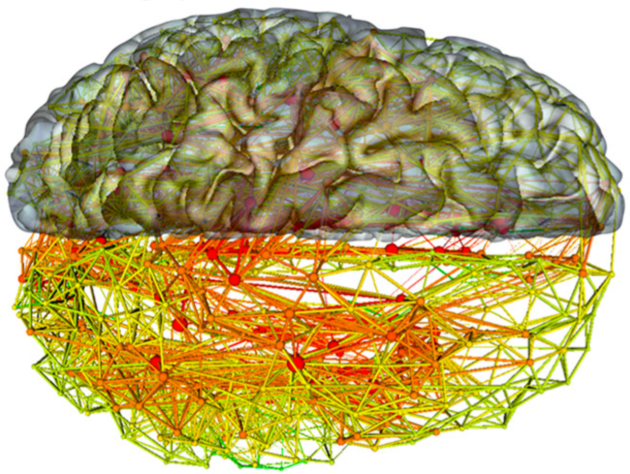
- Images have pixels nicely aligned on a grid



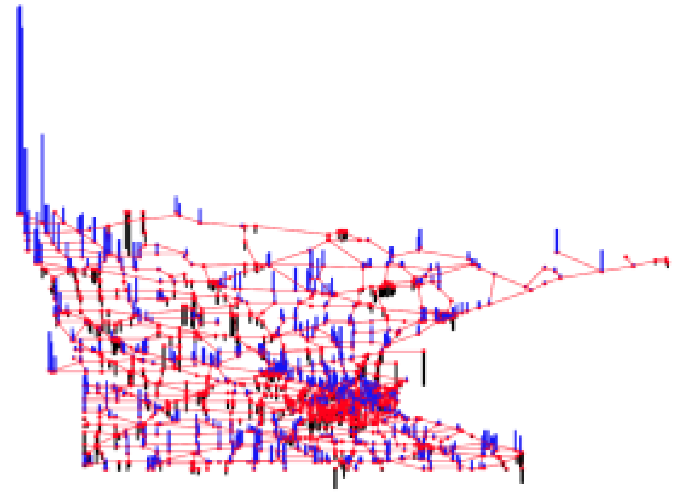
Graphs



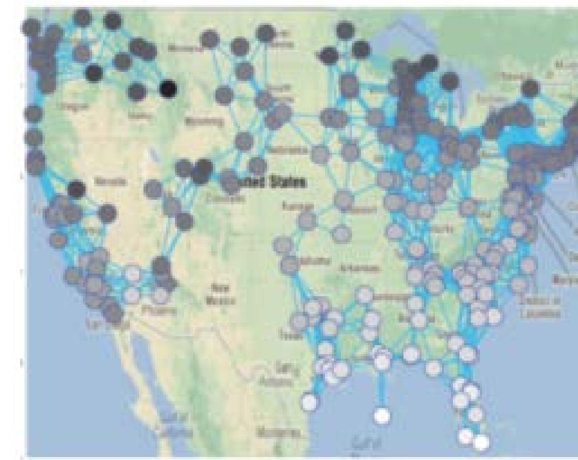
Graph Data



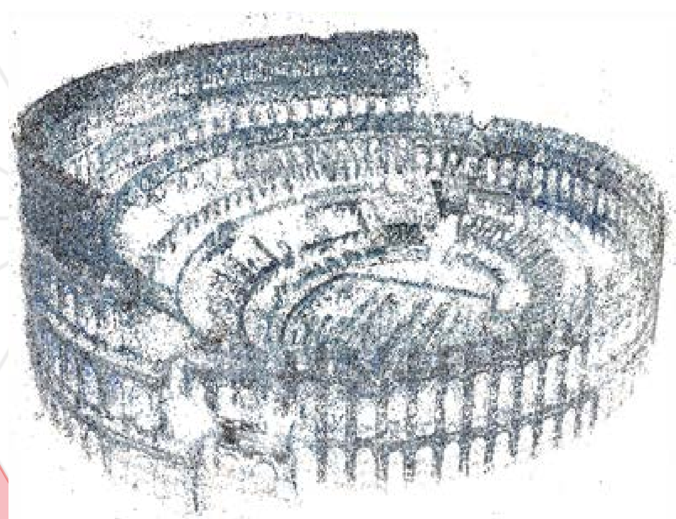
Brain functional networks



Traffic networks

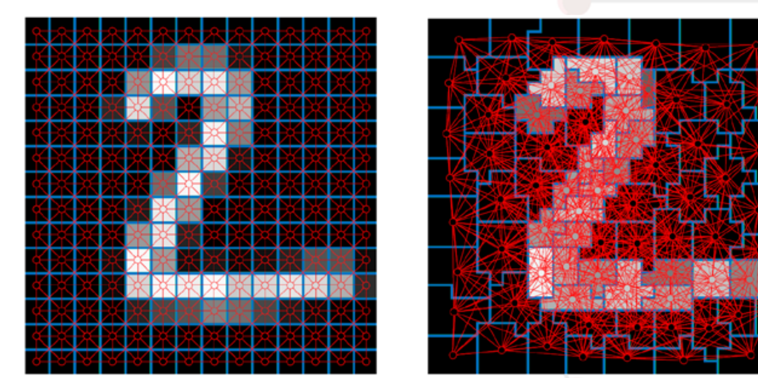


Sensor networks



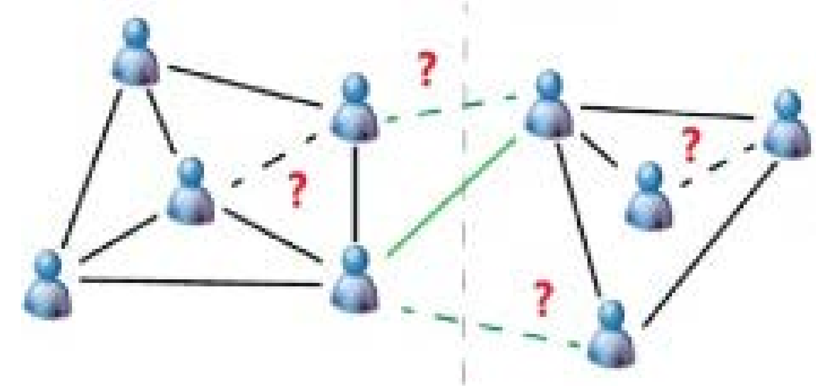
Point clouds

Social networks



Images!

Example: link prediction

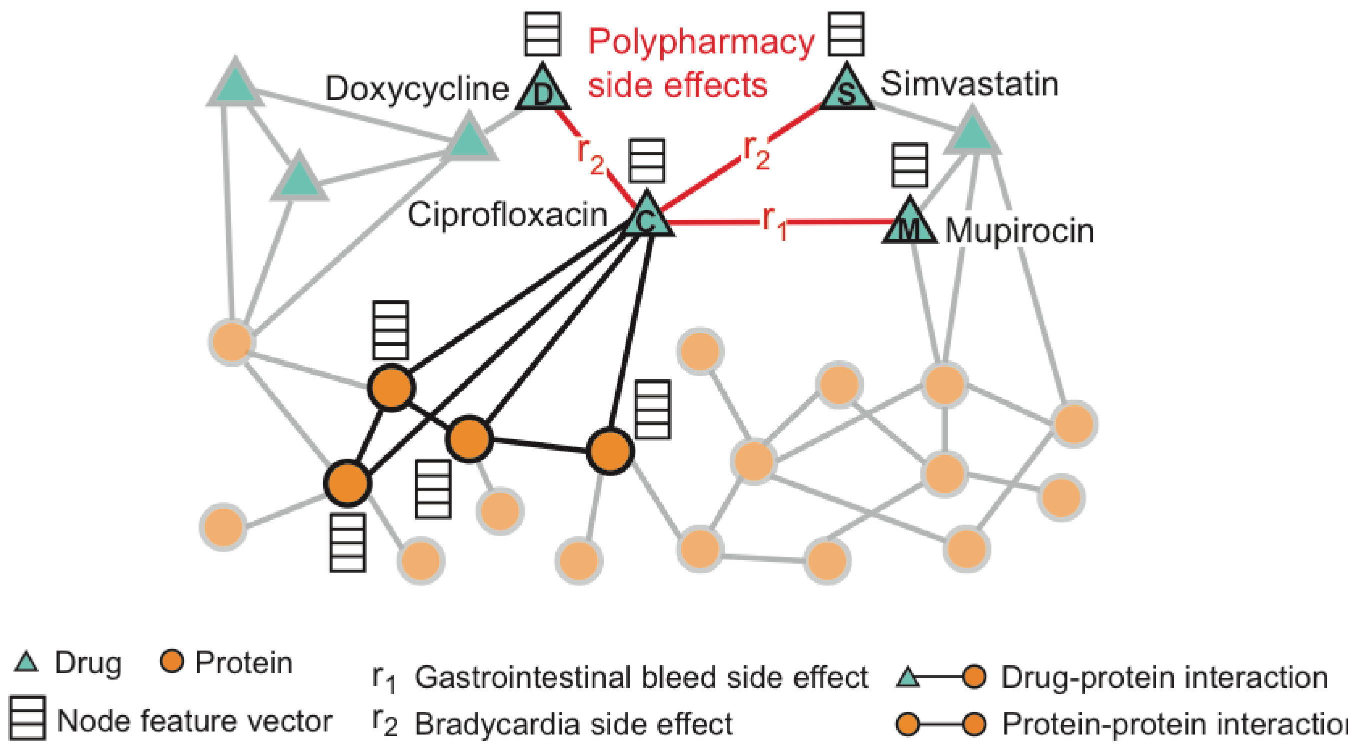


- Social network graph
- Node = Person
  - Feature vector on node describes the person
- Edge = Link between two persons

- Can we predict if two people should be linked even if they currently are not?
  - Recommending friends, movies, topics, ...

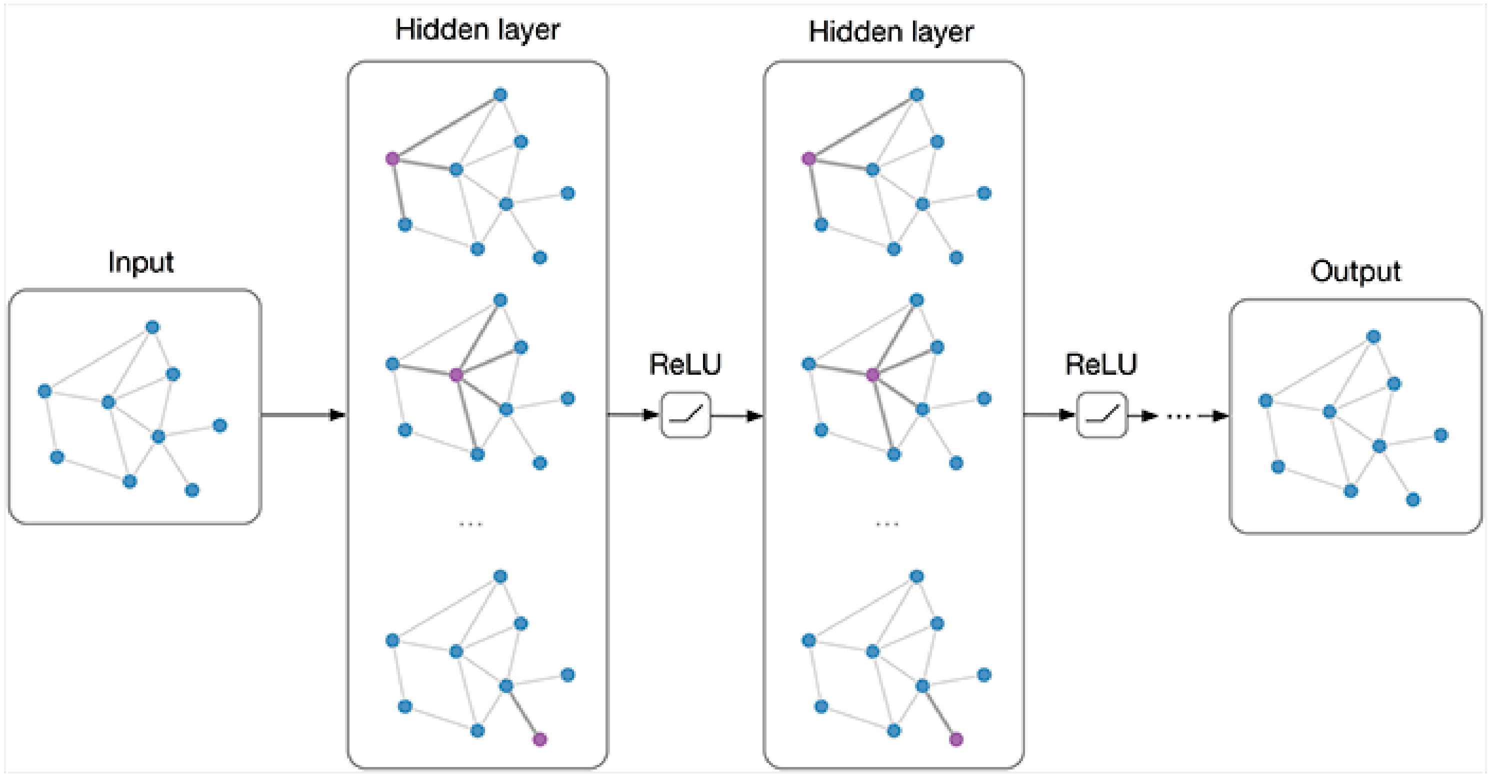
Example: link prediction

- Predict interactions among proteins and drugs [1]



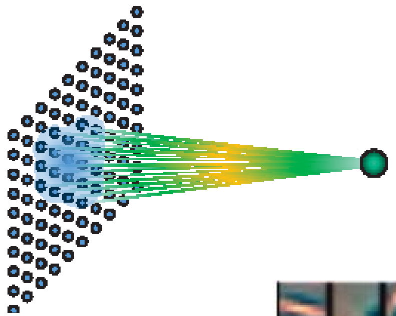
[1]: Marinka Zitnik, Monica Agrawal and Jure Leskovec, "*Modeling polypharmacy side effects with graph convolutional networks*", Bioinformatics, 2018

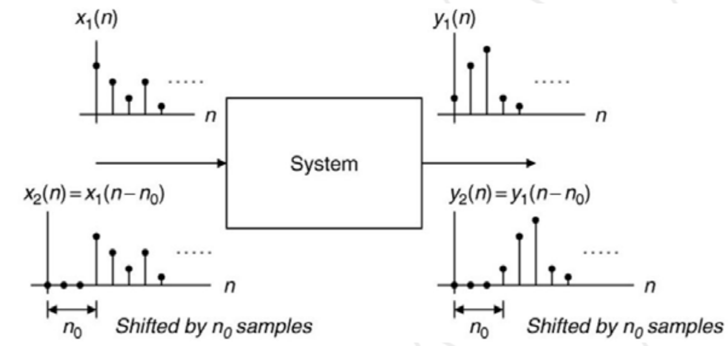
Graph Neural Networks?

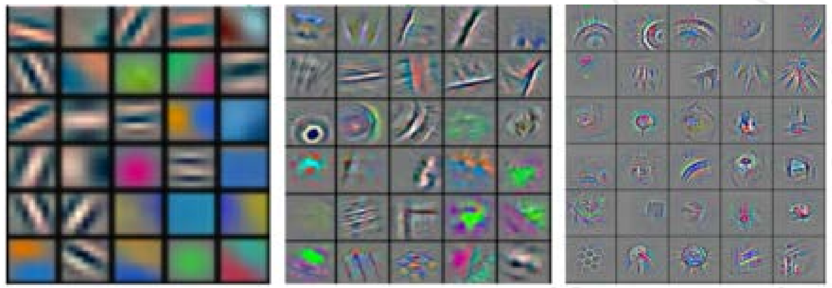


Convolution

- For signals on grids, convolutions impose prior knowledge of certain signal properties:
  - Stationarity: shift equivariance
  - Locality: short-range correlations
  - Compositionality: hierarchical structures







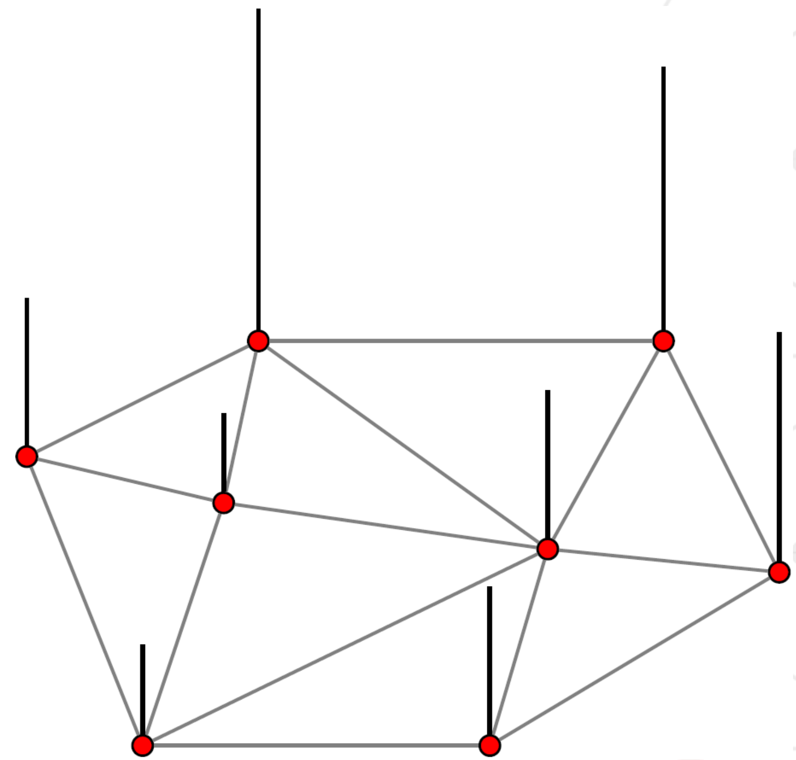
- Convolutional neural networks:
  - reduce the number of parameters (less overfitting)
  - encode prior knowledge in the model

Graph Signal Processing (GSP)

$$G = (V, E)$$

$$x : V \to \mathbb{R}^d$$

- Signals defined over an **irregular domain**
- GSP extends signal processing
- Issues:
  - Ordering is arbitrary
  - Translation?
  - Downsampling?
  - Upsampling?
  - Filtering?



Graph Signal Processing (GSP)

POLITECNICO DI TORINO

SmartData

How to define graph convolution?

(no single universally-accepted definition yet)

1. Spectral approach:
   - define a "Fourier" transform, work in frequency domain

2. Spatial approach:
   - define a way to aggregate data from neighboring nodes

Graph Fourier Transform

- How to define a *"frequency"* notion?
- **Graph Fourier Transform** as eigenvectors of graph Laplacian
  - Analogy with classical Fourier transform (eigenfunctions of ∇^2)

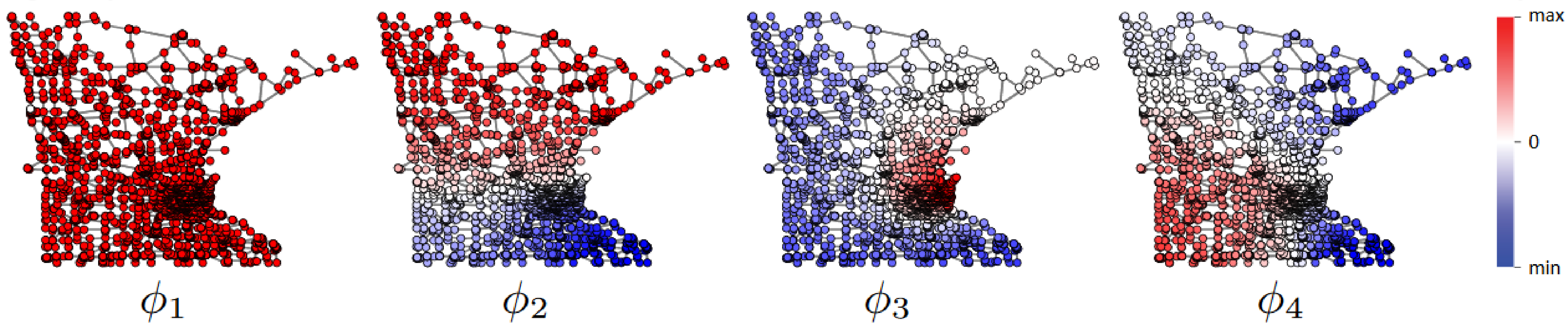
$$L = D - A$$
$$L = \Phi \Lambda \Phi^H$$

Graph Laplacian

$D = diag(\sum_{i \neq j} w_{ij})$: degree matrix
A: adjacency matrix

$$\hat{x} = \Phi^H x$$

Graph Fourier Transform



Spectral Graph Convolution?

Classic signal processing

Graph signal processing

$$x(t) * h(t) = \int x(t-\tau)h(\tau)d\tau$$

convolution in time domain

?

$$F[x(t) * h(t)] = F[x(t)] \cdot F[h(t)]$$

convolution in frequency domain

?

Spectral Graph Convolution

- Spectral approach
  - convolution as product in frequency domain

$$x_f = \Phi H \Phi^H x$$

Filtered signal ← x_f

x → Original signal

Φ → Inverse Graph Fourier Transform

H → Filter

Φ^H → Forward Graph Fourier Transform

Filter

$$\begin{bmatrix} h_1 & 0 & \cdots & 0 & 0 \\ 0 & h_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & h_{N-1} & 0 \\ 0 & 0 & \cdots & 0 & h_N \end{bmatrix}$$

Spatial Graph Convolution

- Spectral approach issues:
  - High computational cost: eigenvectors required
  - Not localized in vertex domain

- Approximation:
  - Fast Graph Filters[1]
  - Polynomials of graph Laplacian $g(L) = \Phi g(\Lambda)\Phi^{\mathrm{H}}$
  - Smoothness in frequency domain ⟷ Localization in vertex domain

$$x_f = g(L)x = \sum_{k=0}^{K-1} \theta_k L^k x$$

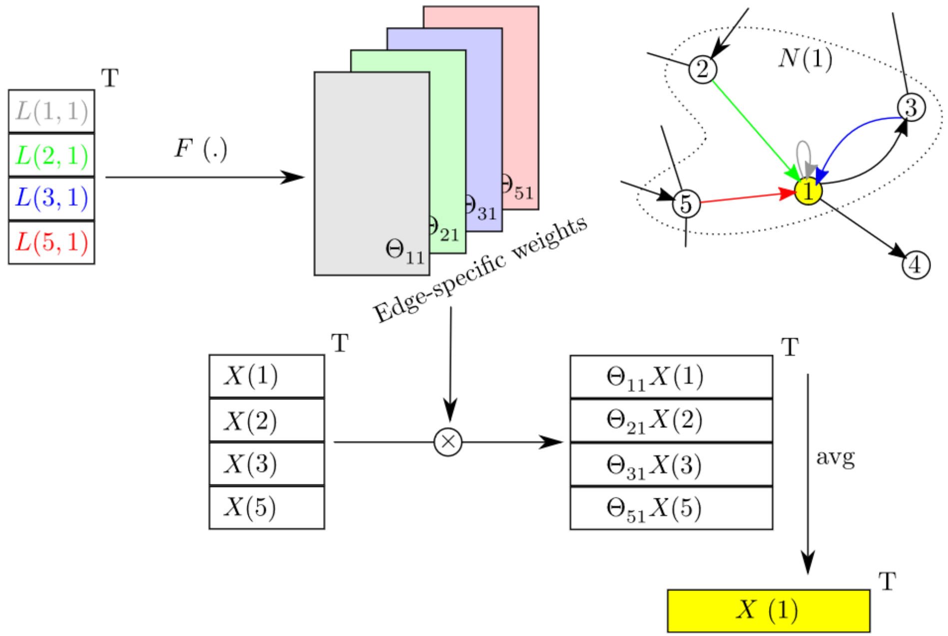
- Recursive implementations (Chebyshev polynomials, Lanczos method)
- $O(K|E|) \ll O(|V|^2)$

  - Graph-dependent: learned filter parameters do not generalize to different graphs

[1]: Defferrard M., Bresson X., Vandergheynst P., "*Convolutional neural networks on graphs with fast localized spectral filtering*", NIPS 2016

Spatial Graph Convolution

- Spatial local aggregations[2]
  - Weighted average of neighbourhood data
  - Weights are functions of edge labels
  - Localization and weight reuse by design



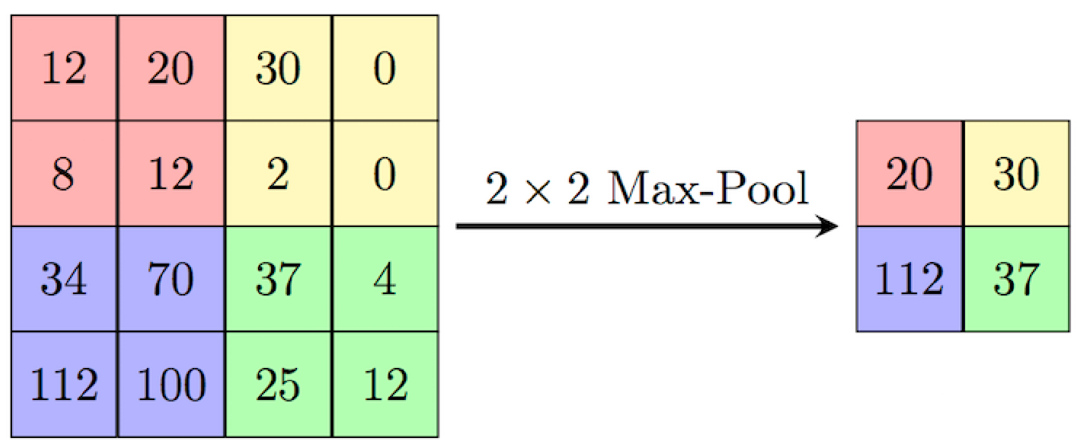
$$x_i = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \theta_{ji} x_j$$

$\theta_{ji} = F(L(i,j))$, where $(i,j) \in E$

[2]: Simonovsky M., Komodakis N., *"Dynamic edge-conditioned filters in convolutional neural networks on graphs"*, CVPR 2017

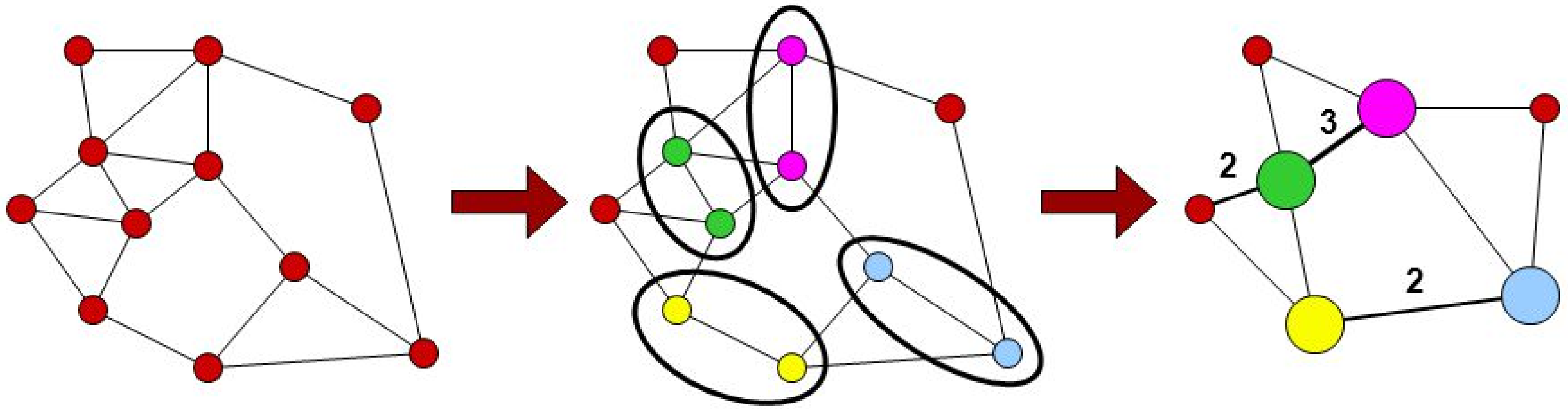
Pooling in CNNs

- Reduces spatial size of data
- Builds invariances into the model
  - Max pooling = local translation invariance



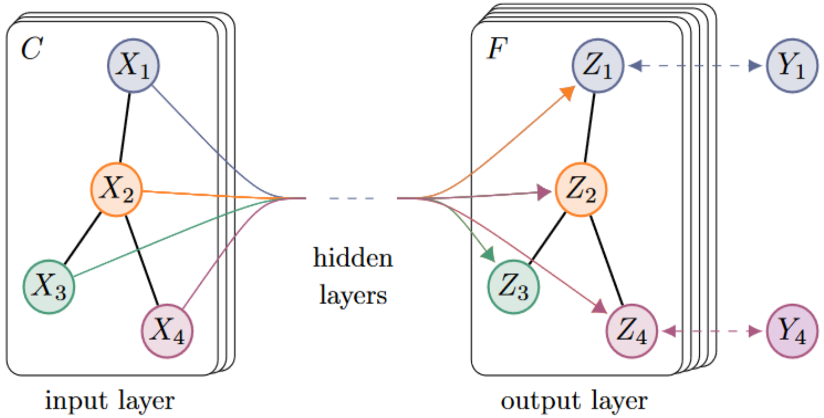
Pooling in Graph CNNs

- Graph coarsening:
  - algorithms to reduce the number of nodes and edges…
  - …while approximately preserving global structure
  - (no complete theory of what information these algorithms actually preserve/destroy)

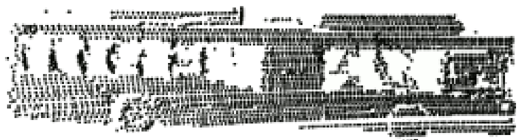


Graph-convolutional neural nets

- Use neural networks to process data defined on graph



- Graph signal processing operations as layers
  - Graph convolution
  - Graph coarsening
  - ...
- Applications:
  - Supervised: classification[3] of the entire graph signal
  - Semi-supervised:
    - Node classification[4]
    - Link prediction[5]
  - Unsupervised: generative models [6]

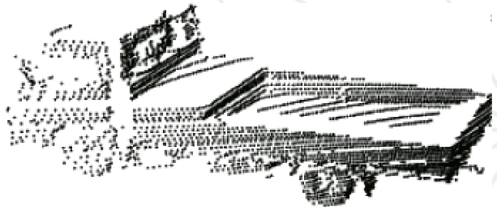


"bus"

[3]: Khasanova R., Frossard P., *"Graph-based isometry invariant representation learning"*, ICML 2017
[4]: Kipf T. N., Welling M., *"Semi-Supervised Classification with Graph Convolutional Networks"*, ICLR 2017
[5]: Schlichtkrull M., Kipf T., Bloem P., van den Berg R., Titov I., Welling M., *"Modeling relational data with graph convolutional networks"*, ESWC 2018
[6]: Valsesia D., Fracastoro G., Magli E., "Learning Localized Generative Models for 3D Point Clouds via Graph Convolution", ICLR 2019

Lidar scans classification

[2]: Simonovsky M., Komodakis N., *"Dynamic edge-conditioned filters in convolutional neural networks on graphs"*, CVPR 2017



- Point clouds with color/intensity value: predict the object class

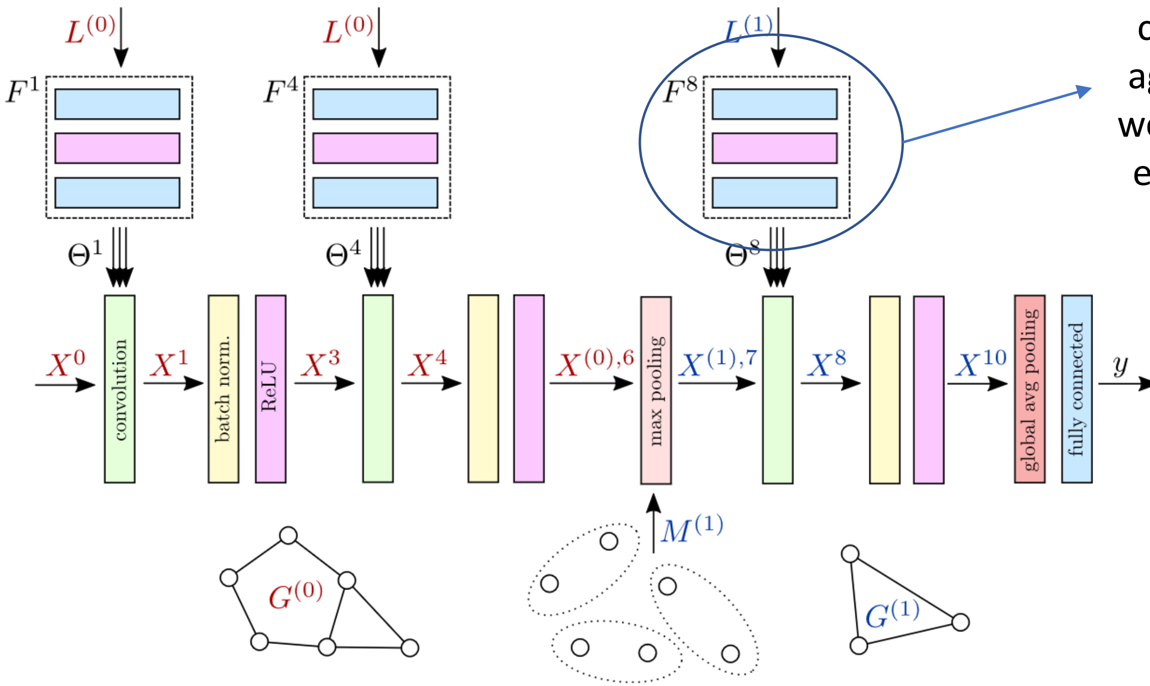
$$\{ (x_0, y_0, z_0, R_0, G_0, B_0), (x_1, y_1, z_1, R_1, G_1, B_1), \dots \}$$

- Before graph convolution: 3D space partition into voxels, quantization and 3D convolution

- Local aggregations

Edge labels
=
coordinate differences
between neighbours
(fixed radius)

$L_{ij} = (x_i - x_j, y_i - y_j, z_i - z_j)$



Dense network computing aggregation weights from edge labels

Lidar scans classification

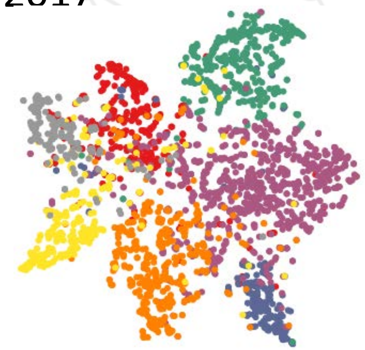
[2]: Simonovsky M., Komodakis N., *"Dynamic edge-conditioned filters in convolutional neural networks on graphs"*, CVPR 2017

Model	Mean F1
Triangle+SVM [9]	67.1
GFH+SVM [7]	71.0
VoxNet [26]	73.0
ORION [1]	77.8
ECC 2ρ	74.4
ECC 1.5ρ	76.9
ECC	78.4

Semi-supervised node classification

[4]: Kipf T. N., Welling M., *"Semi-Supervised Classification with Graph Convolutional Networks"*, ICLR 2017

- Large graph: some nodes are labelled
- Predict the missing labels
  (label distribution depends on graph topology)
- Convolution: 1st order approximation
  of fast graph filters, 2 hidden layers



Citation networks

$$Z = f(X, A) = \mathrm{softmax}\Big(\hat{A}\ \mathrm{ReLU}\Big(\hat{A}XW^{(0)}\Big)\ W^{(1)}\Big)$$

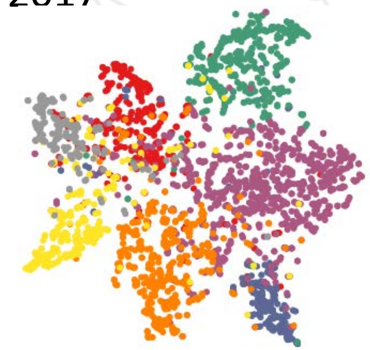
Method	**Citeseer**	**Cora**	**Pubmed**
ManiReg [3]	60.1	59.5	70.7
SemiEmb [28]	59.6	59.0	71.1
LP [32]	45.3	68.0	63.0
DeepWalk [22]	43.2	67.2	65.3
ICA [18]	69.1	75.1	73.9
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)
GCN (rand. splits)	67.9 ± 0.5	80.1 ± 0.5	78.9 ± 0.7

Classification accuracy:
Improves over methods not using convolutional neural nets

Semi-supervised node classification

[4]: Kipf T. N., Welling M., *"Semi-Supervised Classification with Graph Convolutional Networks"*, ICLR 2017

- Large graph: some nodes are labelled
- Predict the missing labels
  (label distribution depends on graph topology)
- Convolution: 1st order approximation
  of fast graph filters, 2 hidden layers



Citation networks

$$Z = f(X, A) = \mathrm{softmax}\Big(\hat{A}\ \mathrm{ReLU}\Big(\hat{A}XW^{(0)}\Big)\,W^{(1)}\Big)$$

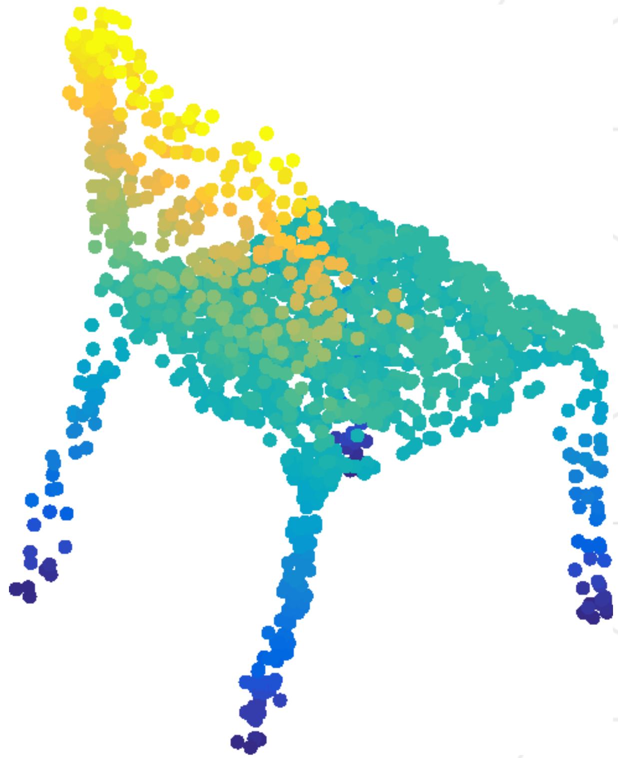
Method	**Citeseer**	**Cora**	**Pubmed**
ManiReg [3]	60.1	59.5	70.7
SemiEmb [28]	59.6	59.0	71.1
LP [32]	45.3	68.0	63.0
DeepWalk [22]	43.2	67.2	65.3
ICA [18]	69.1	75.1	73.9
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)
GCN (rand. splits)	67.9 ± 0.5	80.1 ± 0.5	78.9 ± 0.7

Classification accuracy:
Improves over methods not using convolutional neural nets

Point cloud generation

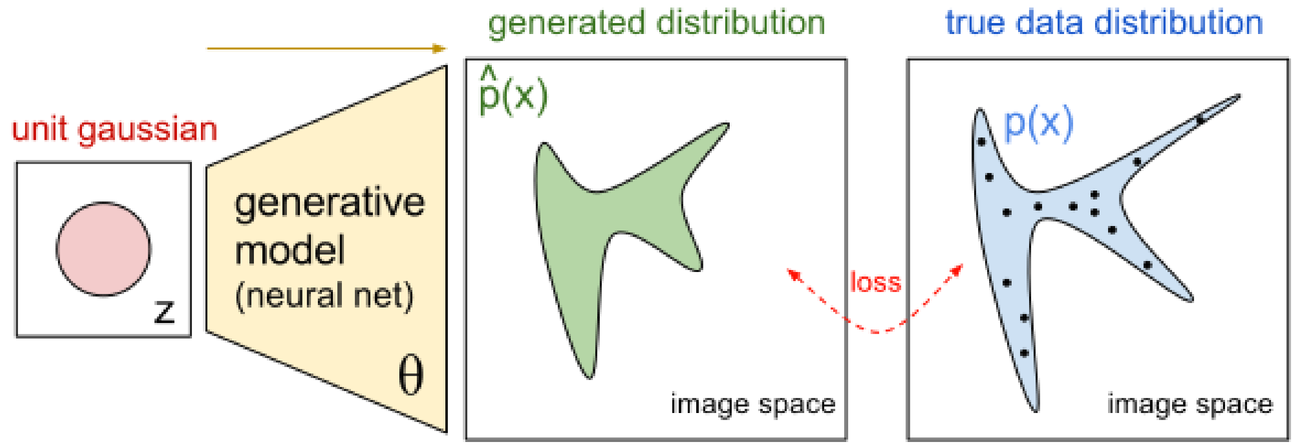
[6]: Valsesia D., Fracastoro G., Magli E., “Learning Localized Generative Models for 3D Point Clouds via Graph Convolution”, ICLR 2019
[7]: Valsesia D., Fracastoro G., Magli E., “Learning Localized Representations of Point Clouds with Graph-Convolutional Generative Adversarial Networks”, journal version, under review

- Generate point clouds
- Exploit graph-convolutional operations in a Generative Adversarial Network



GAN

- Generative model:
  - learns the (complicated) distribution of the data
- Why would I do that?
  - Generation of new samples (e.g. to train supervised models)
  - Regularization of inverse problems
  - Learning powerful representations of the data
  - ...

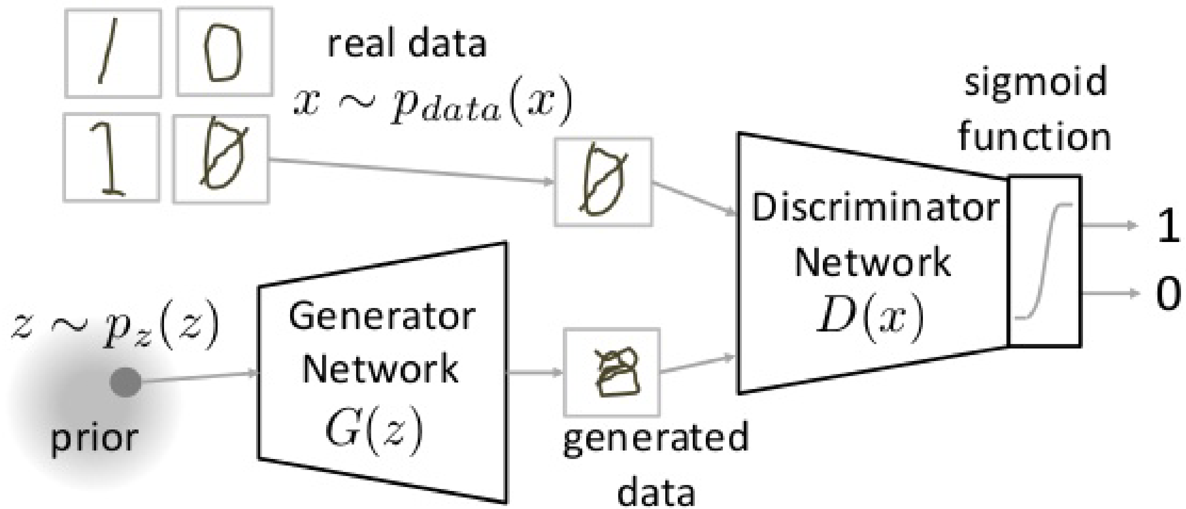


GAN

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "*Generative adversarial nets* ", NIPS 2014

- Two competing networks:
  - Generator: transforms a random latent vector into a «fake» sample
  - Discriminator: guesses if its input is «real» or «fake»

$$\min_{G} \max_{D} E_{x\sim P_r}\left[\log(D(x))\right] + E_{z\sim P_z}\left[\log\left(1 - D(G(z))\right)\right]$$



GAN

- Very impressive results for image generation[1]

- Hard to train, unstable loss
  - Recent improvements (e.g., WGANs[2], progressive growing[3])

[1]: Brock, A., Donahue, J., Simonyan K. , “*Large Scale GAN Training for High Fidelity Natural Image Synthesis*”, ICLR 2019
[2]: Arjovsky, M., Chintala, S., & Bottou, L., “*Wasserstein GAN*”. *arXiv preprint arXiv:1701.07875*.
[3]: Karras, T., Aila, T., Laine, S. and Lehtinen, J., “*Progressive growing of GANs for improved quality, stability, and variation*”. *arXiv preprint arXiv:1710.10196*.

Point cloud generation

Why generating points clouds is hard?

- Unordered sets of points
  - Any permutation is still the same point cloud
- How to exploit spatial correlation?
  - Fully-connected G, 1x1 conv D could generate PCs but no feature localization → no spatial similarity is exploited
  - Voxels encapsulating points: reuse classic 3D conv → approximation

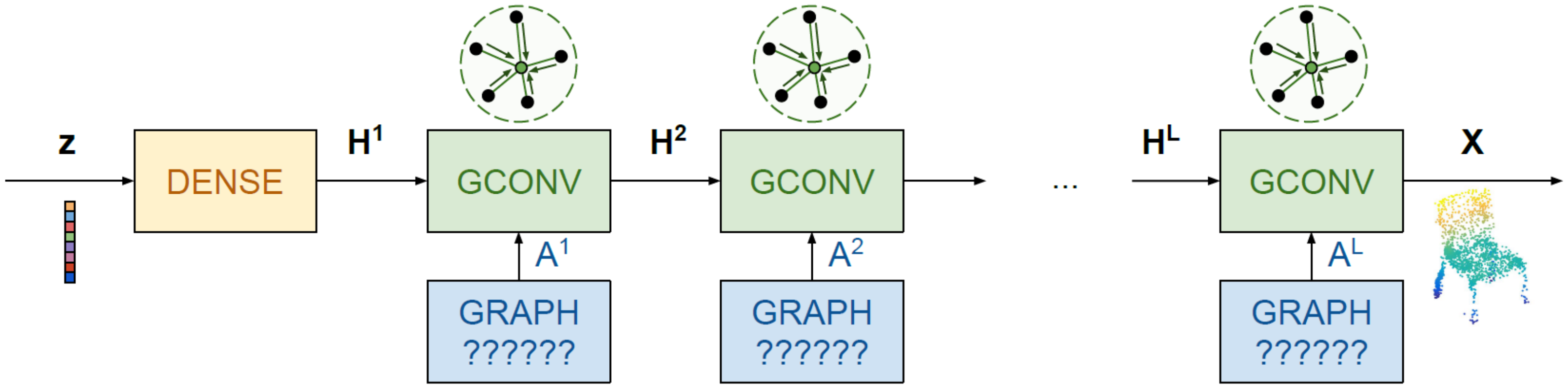
(x_0, y_0, z_0)
(x_1, y_1, z_1)
(x_2, y_2, z_2)
…

Point cloud generation

Why using graph convolution with GANs is hard?

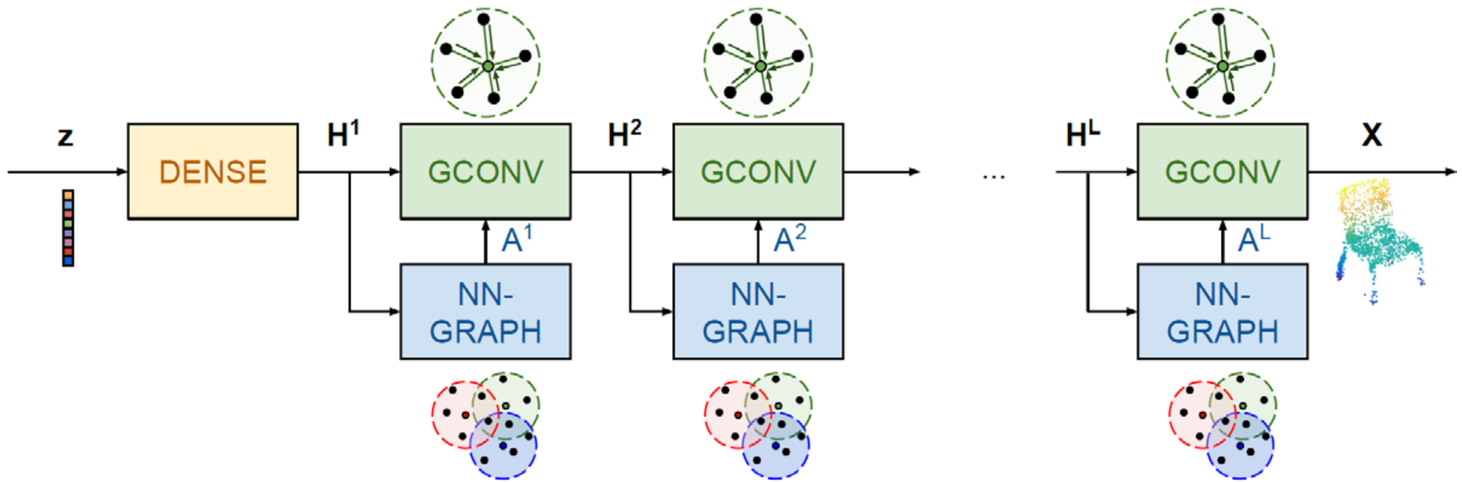
(at the generator)

- Graph convolution requires a graph to do the convolution
- How can I define a graph of neighboring points if the coordinates of the neighbors are the very output of the generator?



Point cloud generation

- Each hidden layers has a feature vector per point
- Build a nearest-neighbor graph from hidden feature vectors

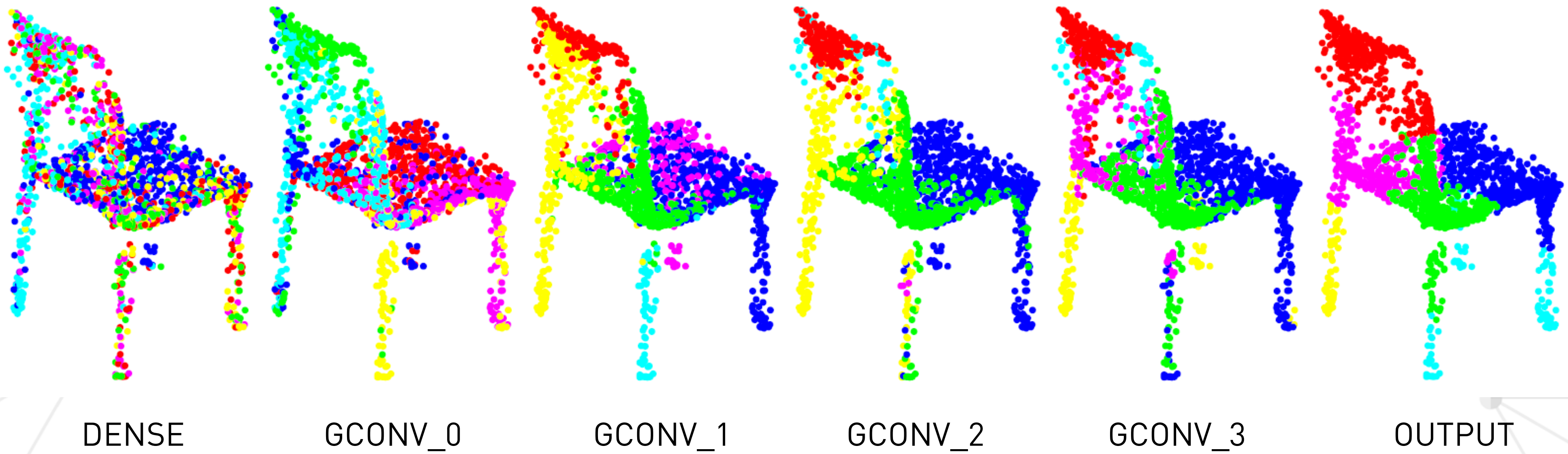


- Gconv is spatial-aggregation graph convolution by Simonovsky et al. : edge labels = differences between feature vectors

$$\mathbf{H}_i^{l+1} = \sigma\left(\sum_{j\in\mathcal{N}_i^l} \frac{F^l_{\mathbf{w}^l}\left(\mathbf{H}_j^l - \mathbf{H}_i^l\right)\mathbf{H}_j^l}{|\mathcal{N}_i^l|} + \mathbf{H}_i^l\mathbf{W}^l + \mathbf{b}^l\right)$$

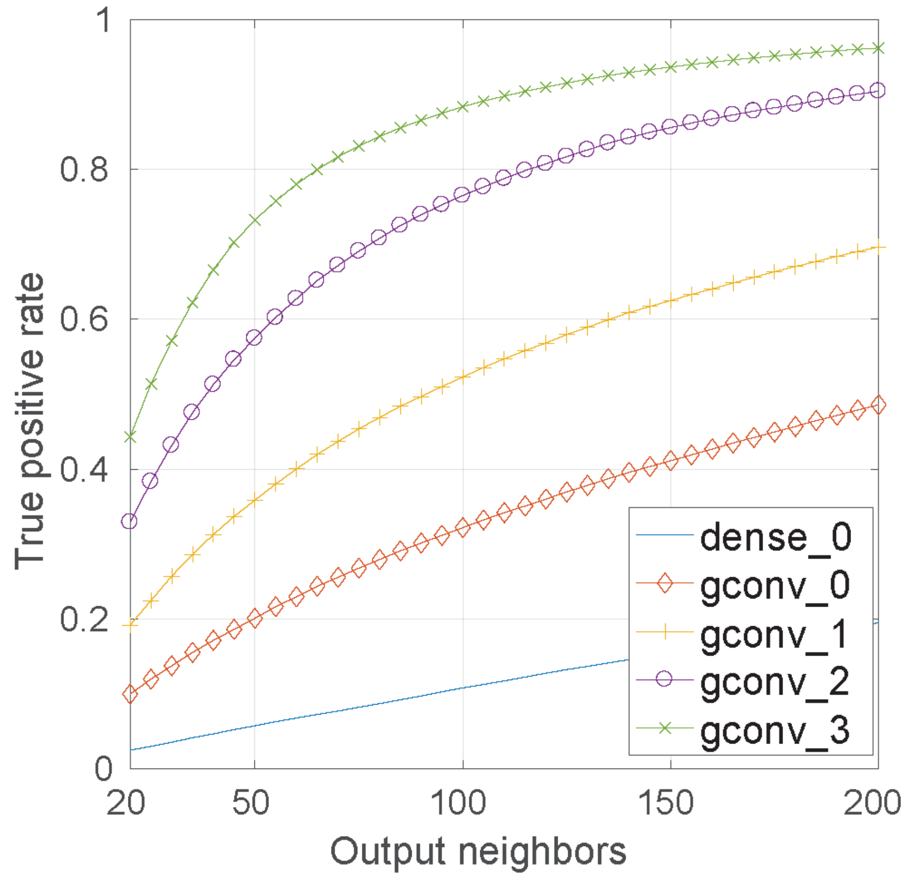
Point cloud generation - Features

- Hidden features are localized
  - They exploit local similarities



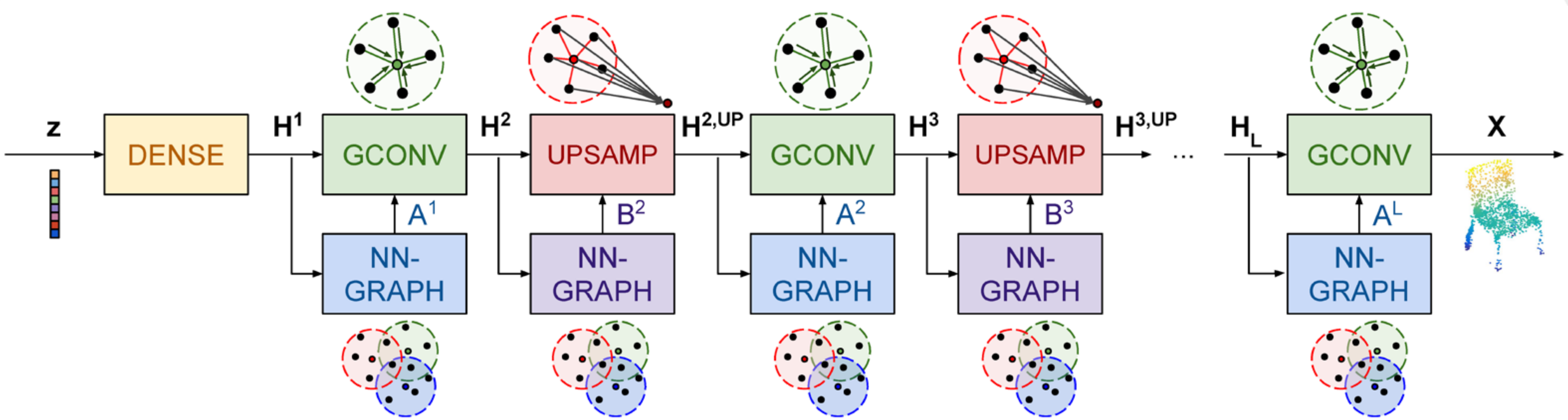
Point cloud generation - Features

- Hidden features are a graph embedding of the output
  - They can predict the output geometry



Point cloud generation – Upsampling layer

- Upsampling: opposite of coarsening/pooling
- We want to increase the number of points through the layers
  - Computational efficiency
  - Exploit multi-resolution prior
- How to do that? In CNNs just place zeros on the grid then filter

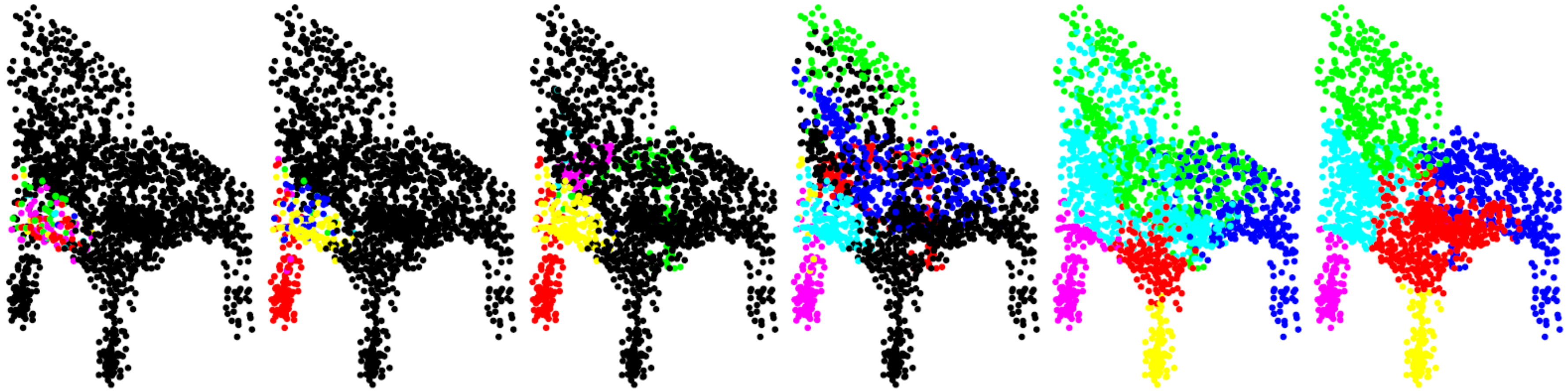


Point cloud generation – Upsampling layer

- Upsampling as aggregation

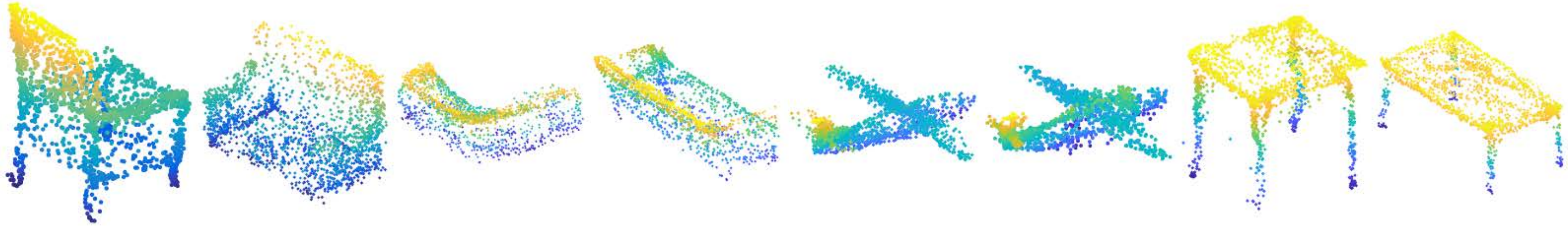
$$\tilde{\mathbf{H}}_i^l = \sigma\left(\sum_{j\in\mathcal{N}_i^l} \frac{\mathrm{diag}\left(F_{\tilde{\mathbf{w}}^l}^{up,l}\left(\mathbf{H}_j^l - \mathbf{H}_i^l\right)\right)\mathbf{H}_j^l}{|\mathcal{N}_i^l|} + \mathbf{H}_i^l\boldsymbol{\Gamma}^l + \mathbf{b}^l\right)$$

- Learns to exploit self-similarity (new neighborhood is similar to old neighborhood but somewhere else)

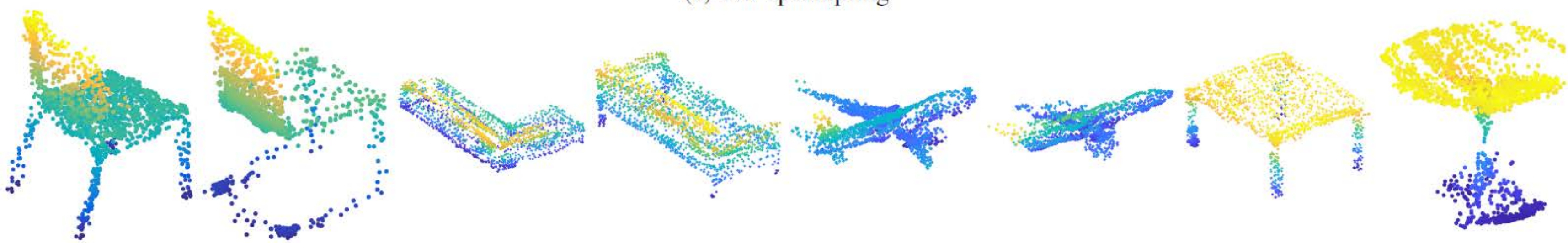


Point cloud generation – Results

- Generated point clouds

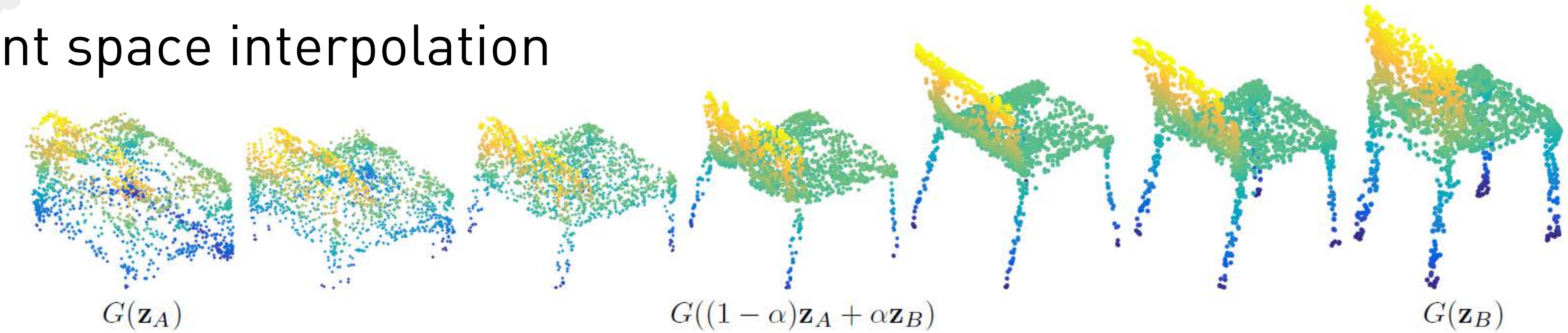


(a) No upsampling



(b) Upsampling - local aggregation approach

- Latent space interpolation



Point cloud generation – Results

Class	Model	JSD	MMD-CD	MMD-EMD	COV-CD	COV-EMD
Chair	**r-GAN-dense**	0.238	**0.0029**	0.136	33	13
	r-GAN-conv	0.517	0.0030	0.223	23	4
	Proposed (no up.)	0.119	0.0033	0.104	26	20
	Proposed (aggr. up.)	**0.100**	**0.0029**	**0.097**	30	26
	Proposed (prob. up.)	0.104	0.0034	0.106	**39**	**31**
Sofa	**r-GAN-dense**	0.221	**0.0020**	0.146	32	12
	r-GAN-conv	0.293	0.0025	0.110	21	12
	Proposed (no up.)	0.095	0.0024	0.094	25	19
	Proposed (aggr. up.)	**0.063**	**0.0020**	**0.083**	**39**	**24**
	Proposed (prob. up.)	0.119	0.0022	0.113	32	19
Airplane	**r-GAN-dense**	0.182	0.0009	0.094	31	9
	r-GAN-conv	0.350	**0.0008**	0.101	26	7
	Proposed (no up.)	0.164	0.0010	0.102	24	13
	Proposed (aggr. up.)	**0.083**	**0.0008**	**0.071**	31	14
	Proposed (prob. up.)	0.095	0.0010	0.075	**34**	**19**
Table	**r-GAN-dense**	0.217	**0.0031**	0.139	33	15
	r-GAN-conv	0.359	**0.0031**	0.247	29	4
	Proposed (no up.)	0.171	0.0045	**0.123**	24	18
	Proposed (aggr. up.)	**0.148**	0.0035	0.131	**36**	29
	Proposed (prob. up.)	0.167	0.0037	0.124	33	**34**

Graph-convolutional Image Denoising

- Denoising: classic problem in image processing
- Important in many applications (not just to get pretty pictures)

Graph-convolutional Image Denoising

- Model-based methods:
  - Non-local self-similarity is key to good performance
  - NLM, BM3D, …
- Deep Learning:
  - CNNs only create hierarchies of local features
  - Receptive field expands radially from local patch

Need to combine non-locality and neural networks!

Open Issues & Future

- What is graph convolution? Is there a better definition?
- More computationally-efficient definitions
- More widespread availability inside Tensorflow, PyTorch, ...

- Non-local models based on graphs are appearing in networks for many problems

Thank You!