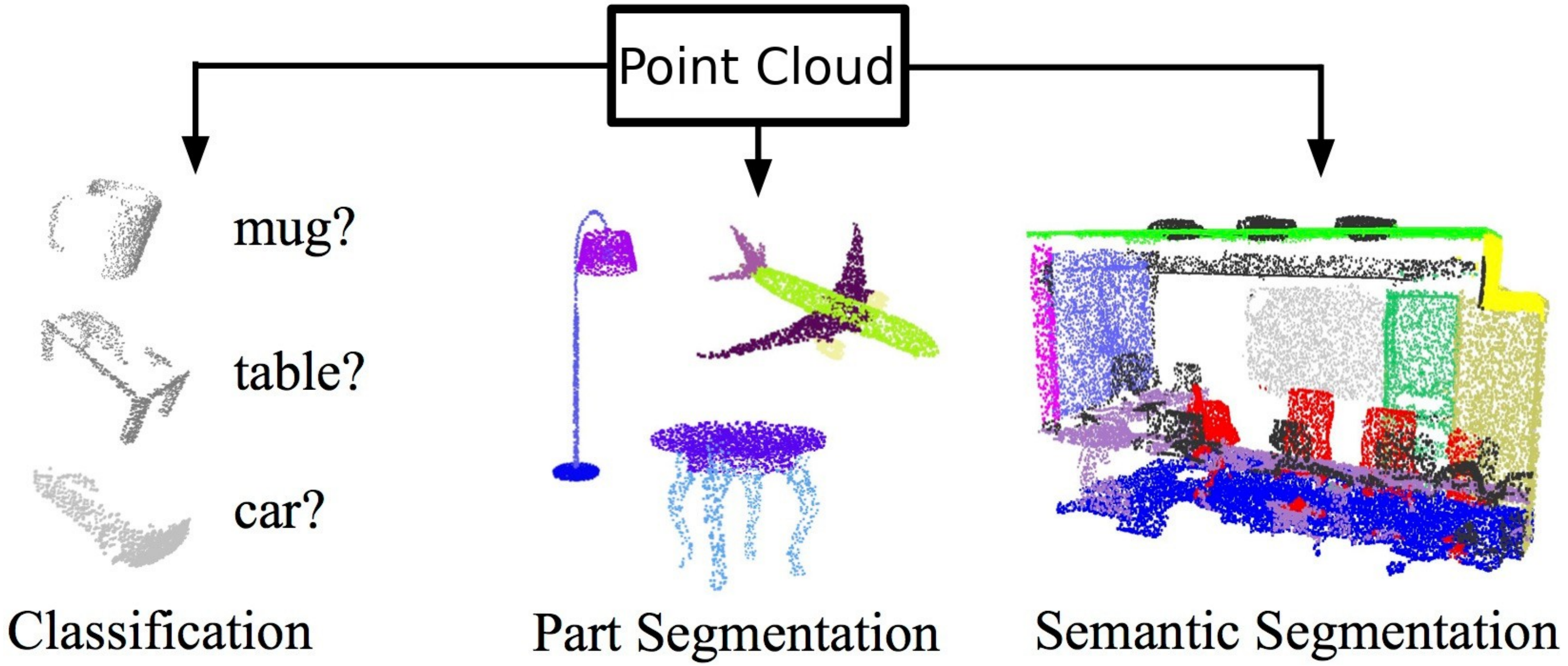
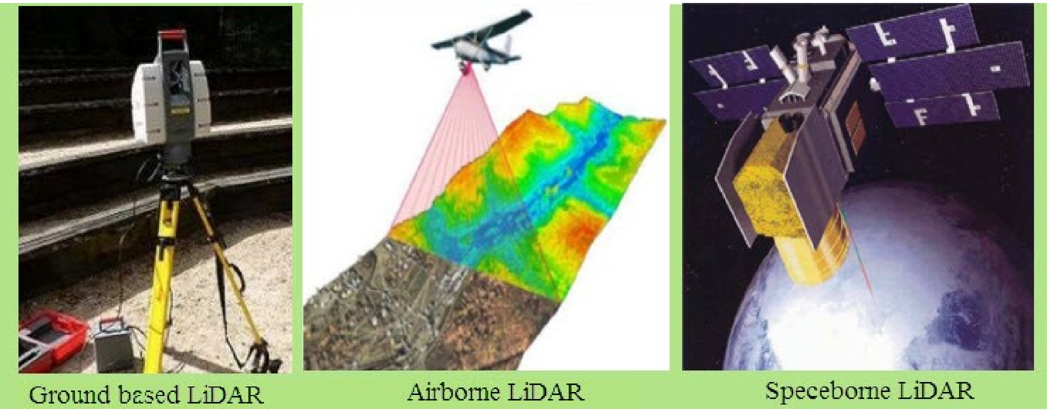
Point Cloud Neural Networks



Frédéric Magniette - LLR

Introduction

- A lot of application gets unordered 3D point cloud as input
- Main application : robotics, self-driving cars, monitoring (rivers level, volcanoes, glacier…) from drone or satellite
- Need dedicated algorithms
- Question : how to transfer convolution revolution to this kind of data



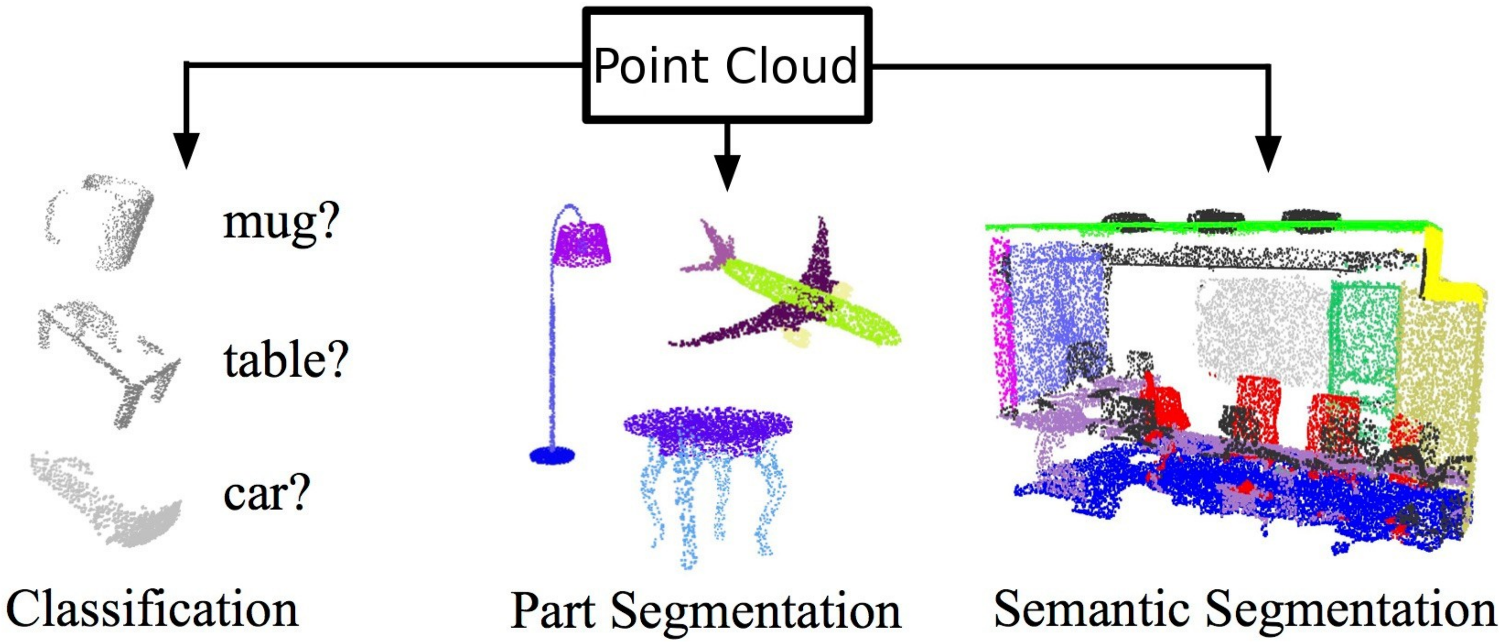
Point Cloud Data

Input

- Extract information directly from a point cloud
- Points P_i in R^k $(k \geq 3)$ Euclidian 3D coordinates + (k-3) « colors »
- 4 main properties
  - unordered : need for a permutation invariant operator
  - Interaction among points : the metric distance defines meaningful neigbourings
  - Invariance under transformation : rotation and translation should not modify the result
  - Sparsity

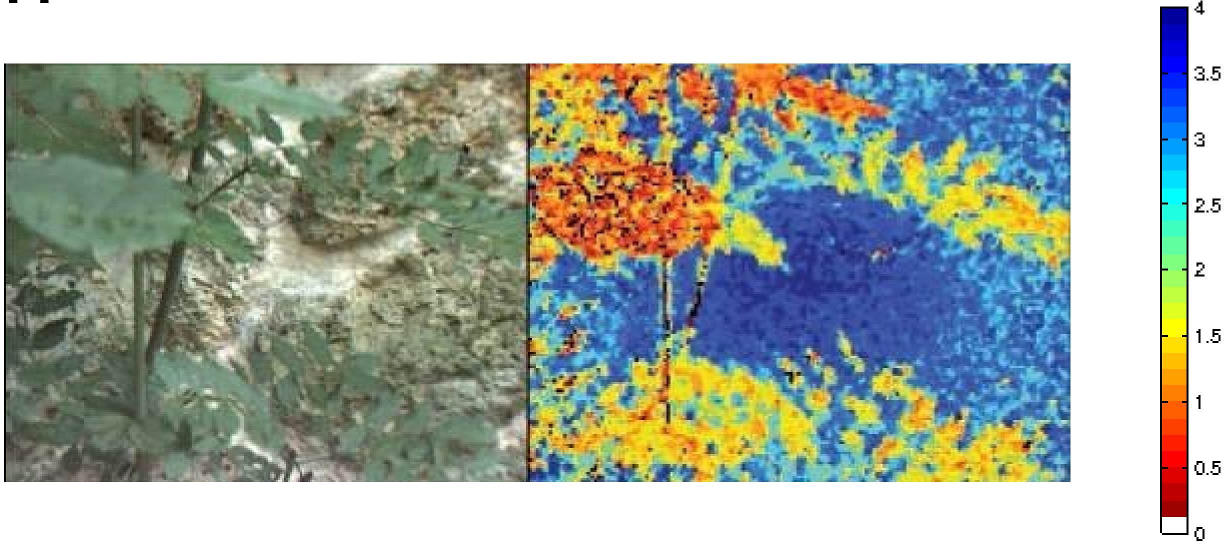
Problematics

- Three problematics
  - Classification
  - Part segmentation
  - Semantic segmentation



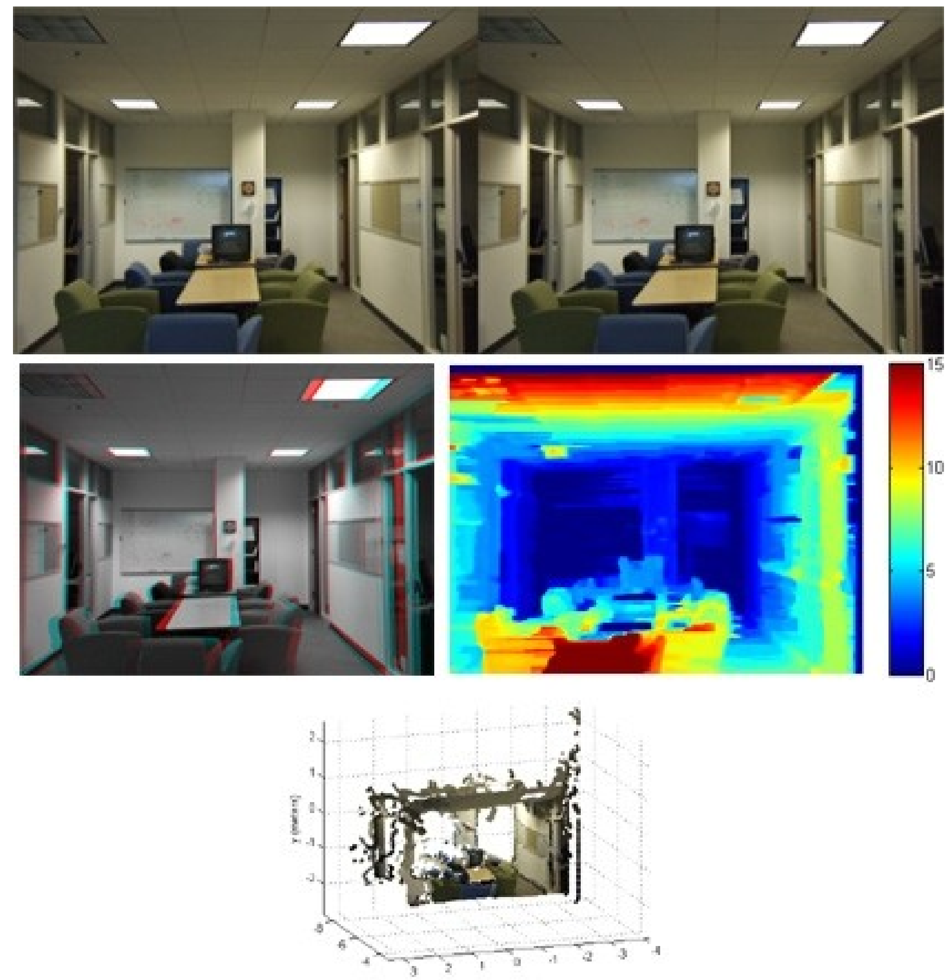
Input device: RGB-D Camera

- color + depth
- Analysis of defocusing blur → distance
- Can be converted to partial 3D representation



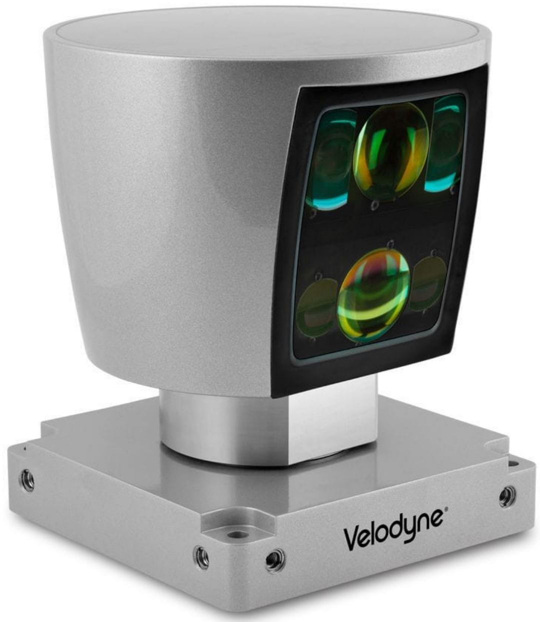
Input device: Stereo Camera

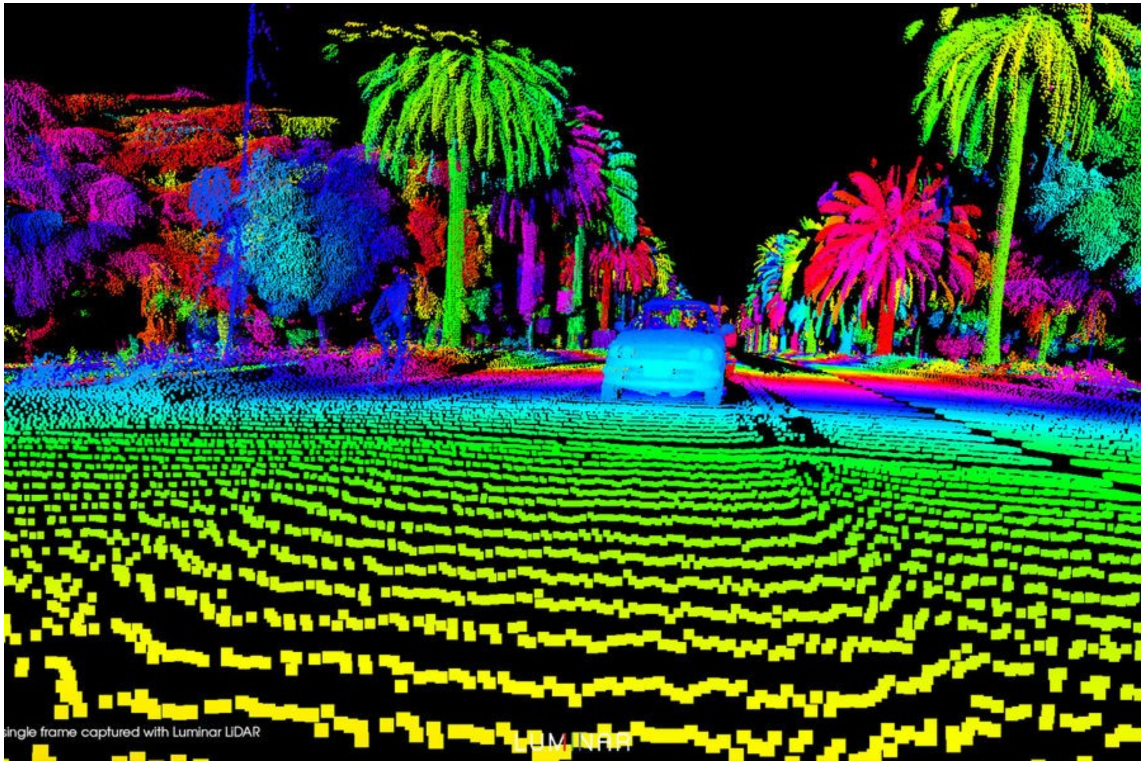
- Take 2 images at the same time
- Stereoscopy : Calculate the distance from the shift



Input device: LIDAR

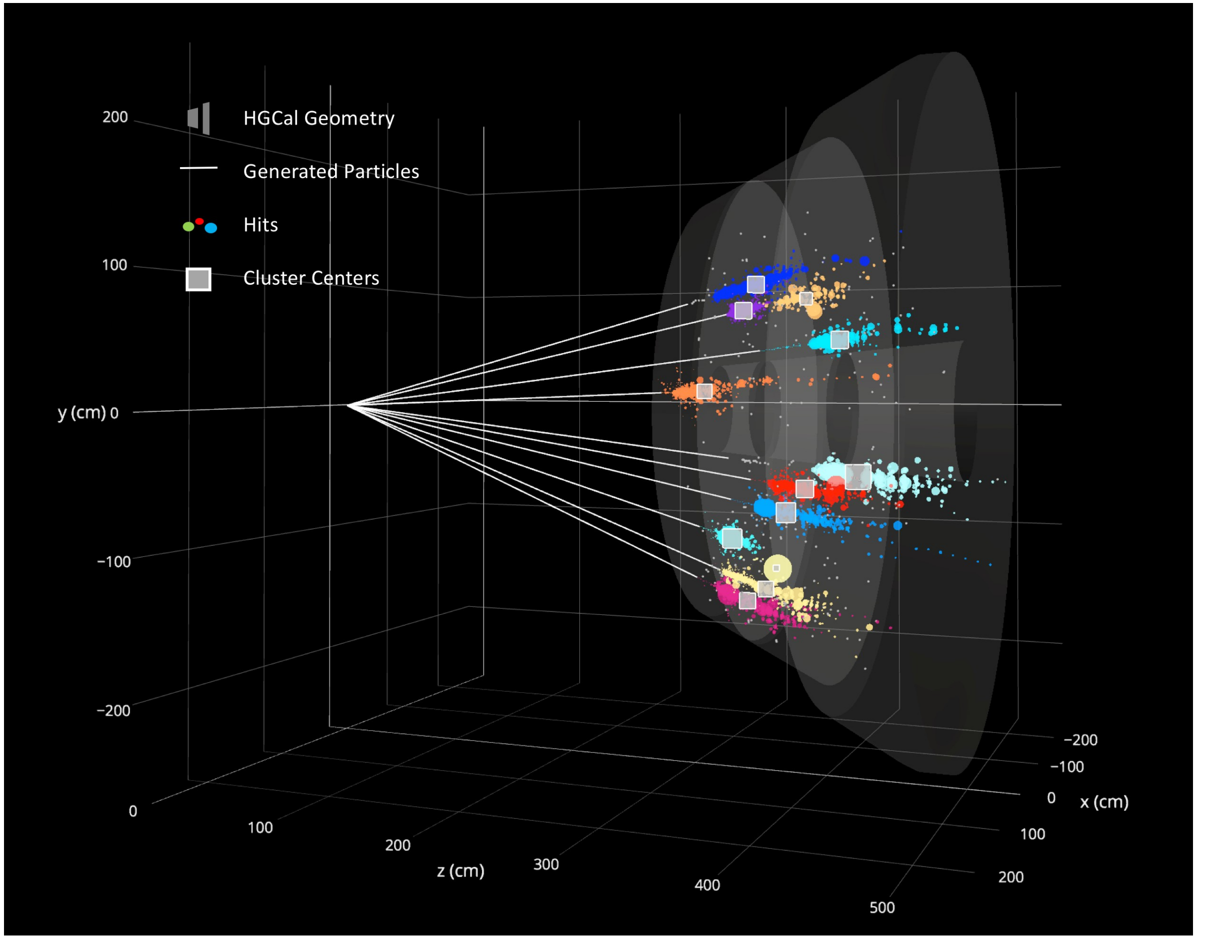
- LIDAR : light detection and ranging
- Emitting visible laser light
- Analyse the return of the light
- Can also measure the speed by Doppler effect
- Used for advanced robotics





Input device : Particle detectors

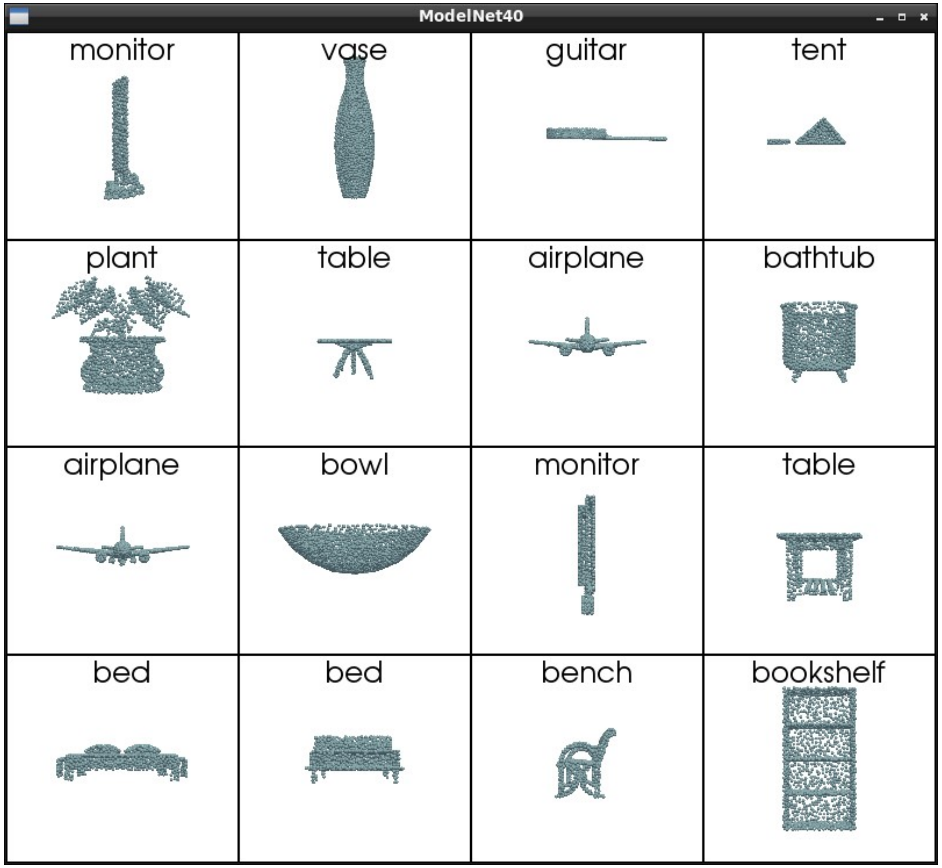
- Hits : 3D point with energy measurement and timing → 5D points
- Different granularity
- Barycenter of sensors



ModelNet40

- CAD models in 40 categories
- 1024/2048 point clouds
- Around 12k models
- Canonical dataset for point cloud classification

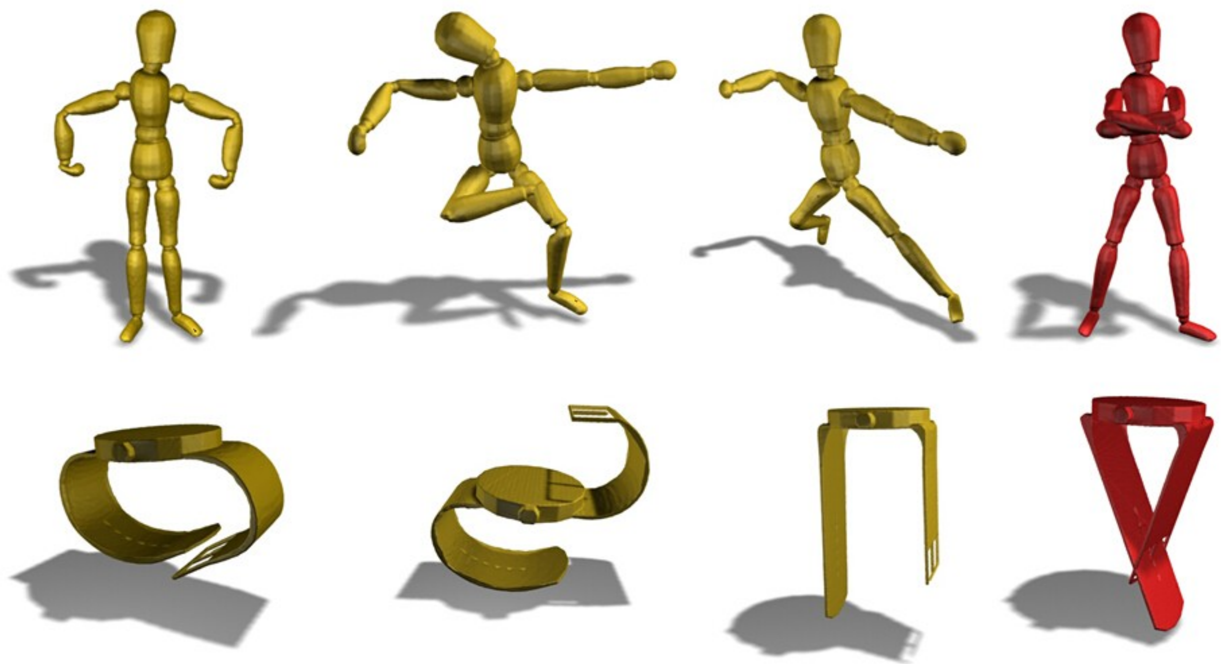
Wu & al, 3D ShapeNets: A Deep Representation for Volumetric Shapes, 2015



SHREC 15

Lian & al, Non-rigid 3D Shape Retrieval, 2015

- Non rigid shapes
- 1200 3d shapes
- different poses of the same 3D model
- Classified in 50 categories



ScanNet

- RGB-D video dataset
- 2.5 million views
- 1500 scans
- annotated with
  - surface reconstructions
  - instance-level semantic segmentations

Dai& al, Scannet: Richly-annotated 3d reconstructions of indoor scenes, 2017

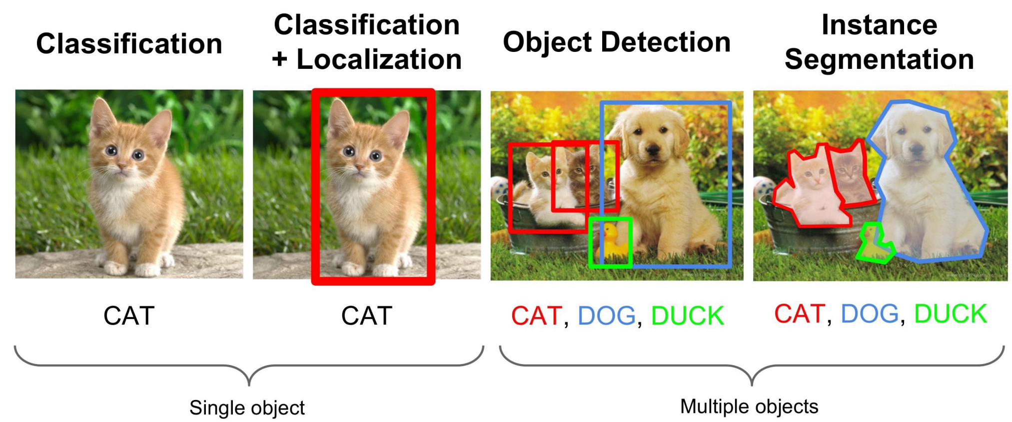
ShapeNet

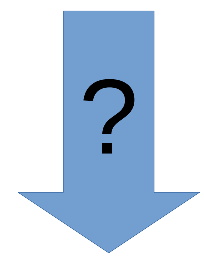
Yi & al, A scalable active framework for region annotation in 3D shape collections, 2016

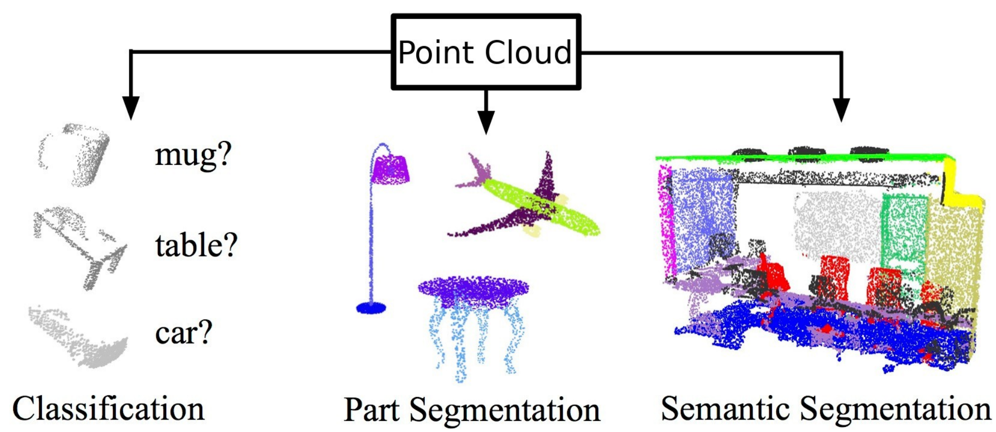
- Part of object data from 50 different part denomination
- 16881 CAD models from 16 categories
- 2048 points samples

Question

- How to transpose the tremendous success obtained with 2D image convolution to 3D point cloud ?
- Before 2015 : handmade feature
  - specific spatial configuration
  - Dedicated to a specific problem
  - unable to be transfered to similar problem
- A lot of work based on neural network from 2015 to now





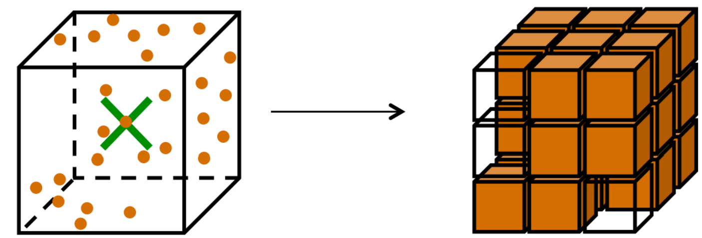


Point Cloud Neural Networks

Three main techniques

Precision

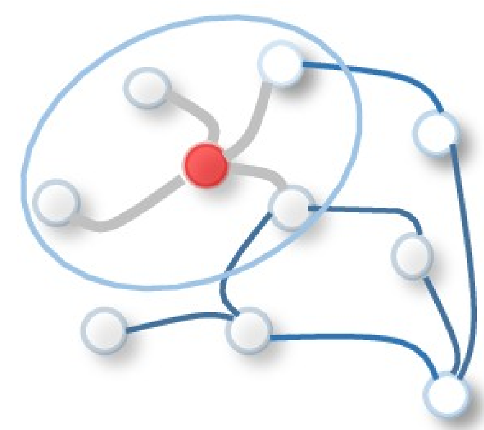
– Voxelization and 3D Convolution (2015-2016)



– Symmetric pooling (2017-2018)

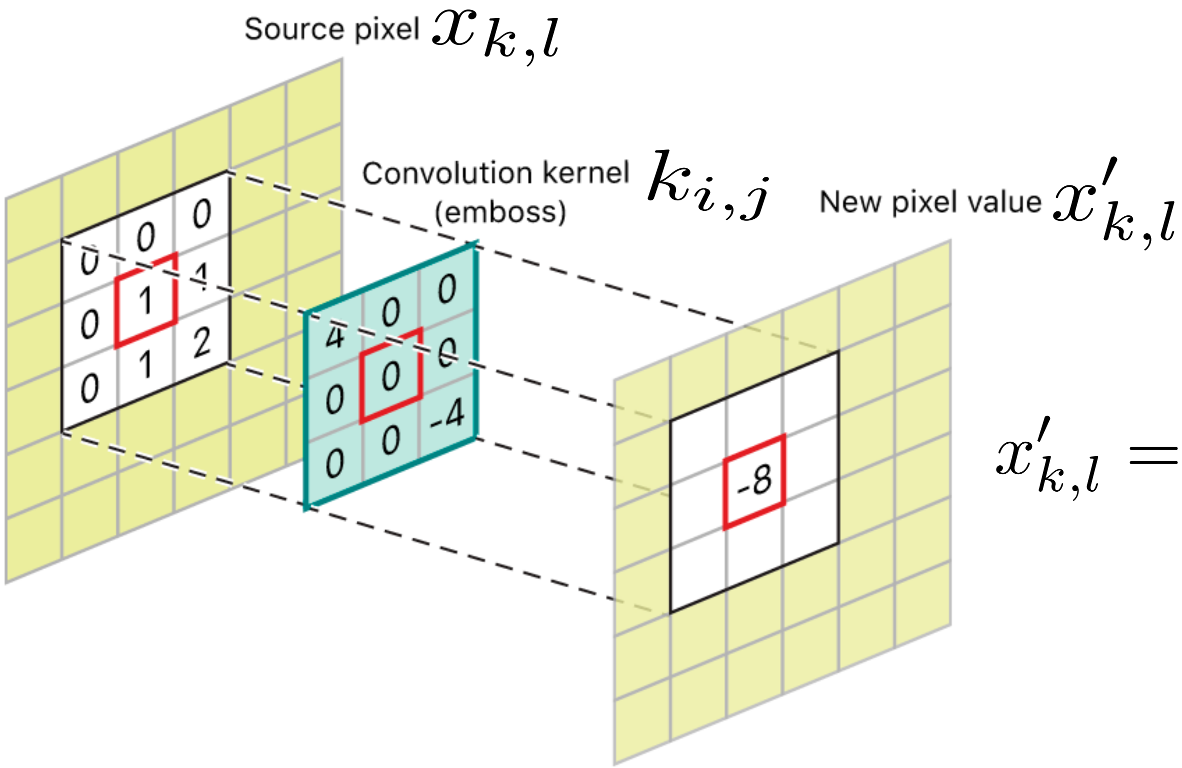
$$f(x_1, \ldots, x_n) \approx g(h(x_1), \ldots, h(x_n))$$

– Graph Convolution (2017-now)



Convolution Recall

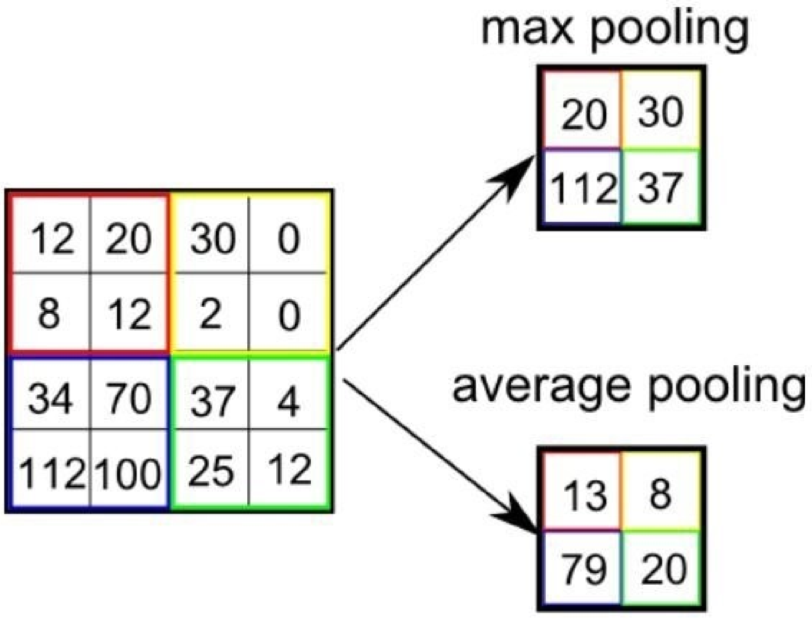
Convolution



$$x'_{k,l} = \sum_{i,j \in [-s..s]^2} x_{k+i,l+j} * k_{i,j}$$

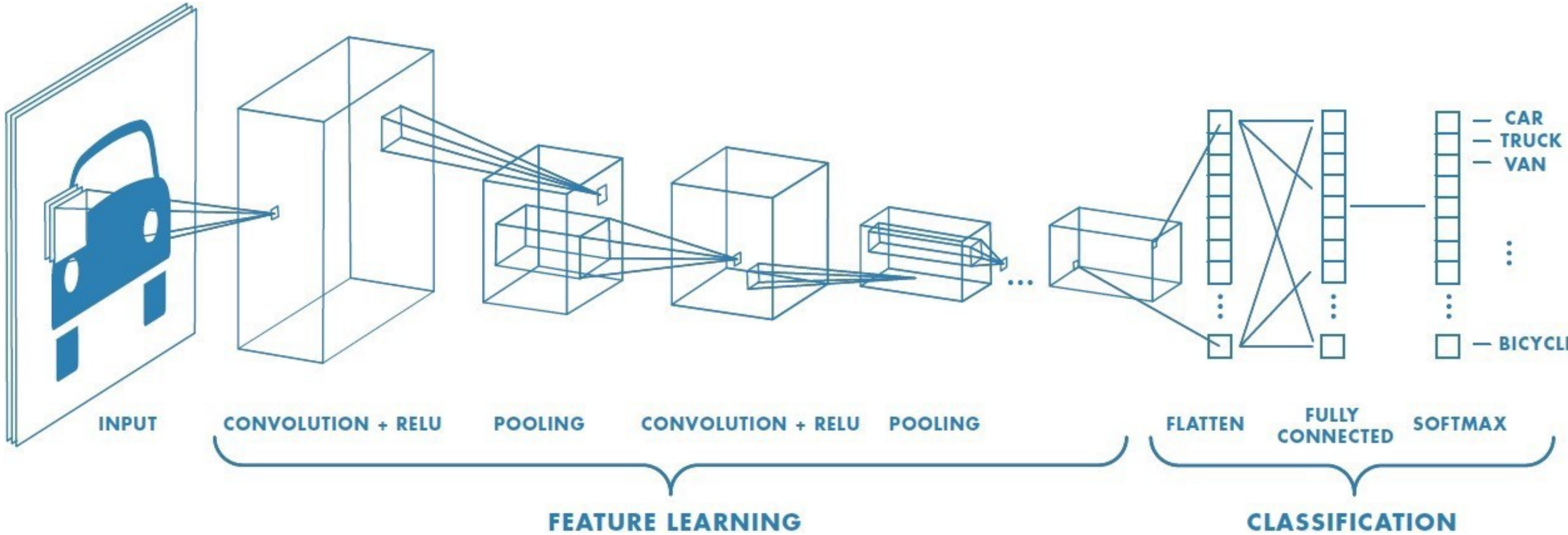
- Apply kernel on image (like the convolution filter)
- kernel is learnable $(k_{i,j})$
- Filter is shared over the whole picture
- Idea : creating maps of features (one kernel per feature)

Pooling



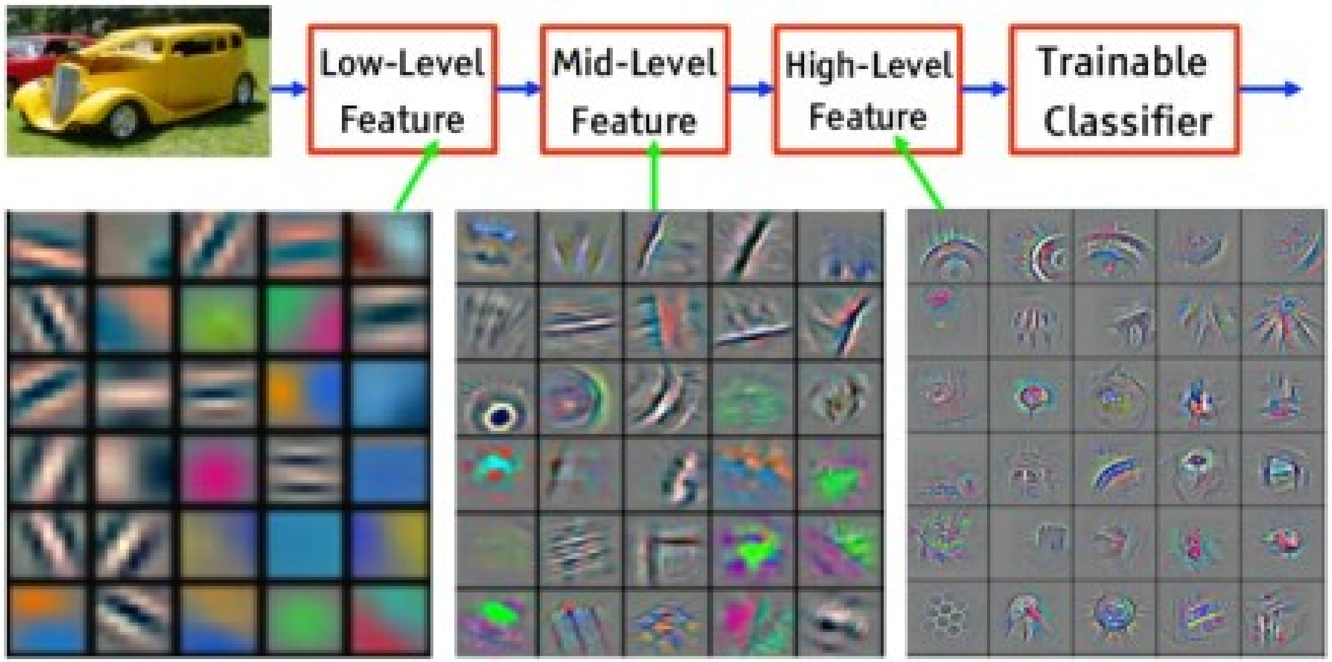
- Reduce the dimensionality of the feature maps
- Move to higher level of abstraction
- Max pool is widely used

Convolutional network



- Network structure :
  - Alternance of convolution & pooling
  - Flattering (sometimes called readout)
  - Multi-layer perceptron

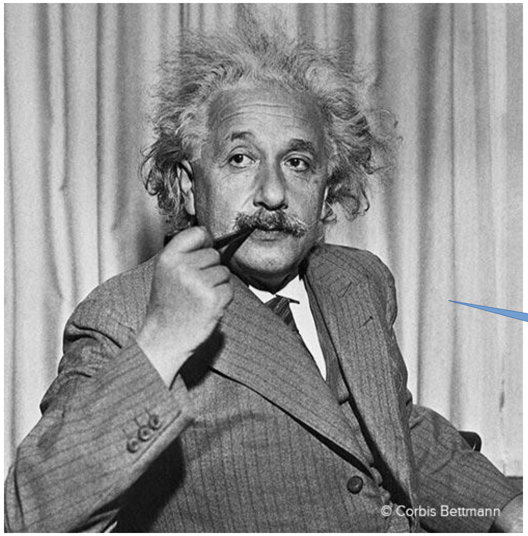
How it works ?



- Feature maps aggregates more and more details to converges to high level recognition patterns
- Flattened high-level feature map is input for multi-layer perceptron

Why it works ?

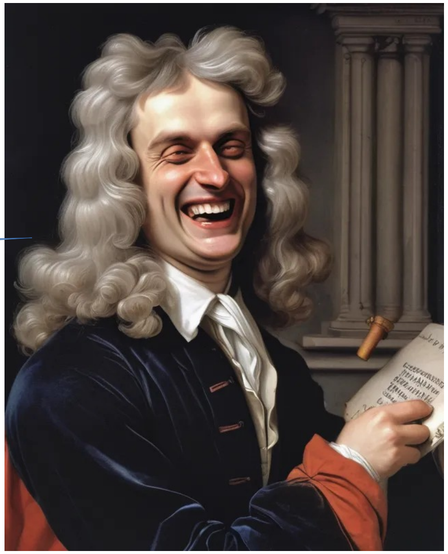
- The two operations derive naturally from local space Euclidian nature
  - Euclidian space → translation-invariance (stationarity) → convolution
  - Scale-separability (compositionality) → downsampling
- Dream complexity
  - O(1) parameters per filter (independant of image size)
  - O(n) complexity in time per layer (n=#pixels)



You see, I told you my little Albert ...

LOL

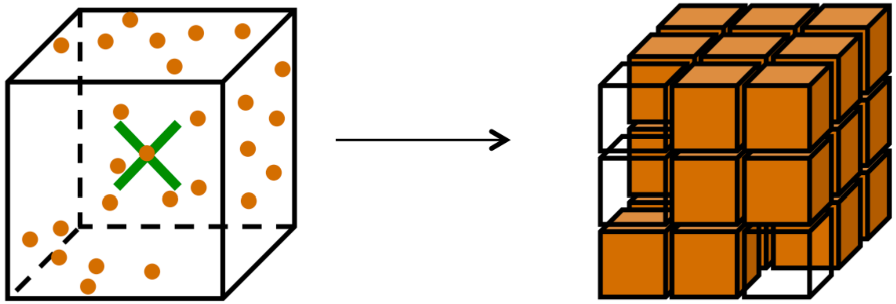
Oh, shut up, Euclid !
And you too, Newton !!!!

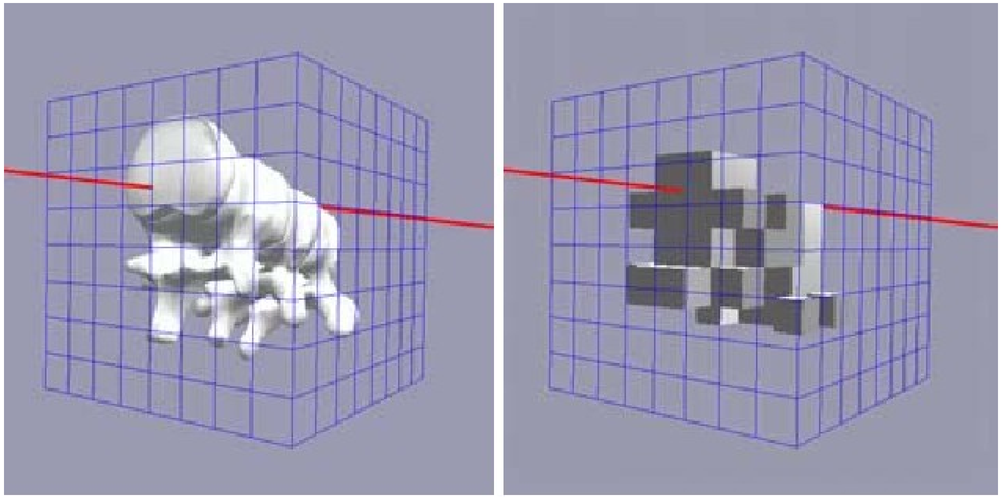


3D Convolution Solutions

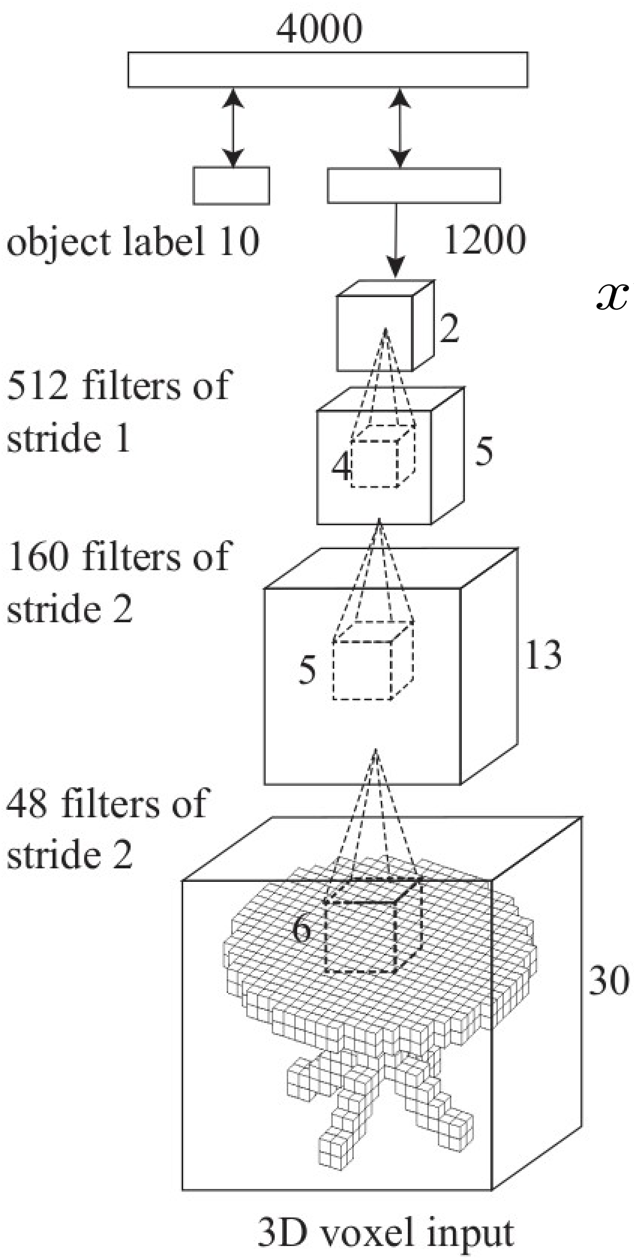
Data Voxelization

- From coordinates to boolean 3D tensor
- Voxel (volume pixel)
- Can be enriched to colored voxel
- Quantization artifact → potential degradation of the recognition





3D Convolution



- Simple extension of 2D formula to voxelized 3D data

$$x'_{k,l,m} = \sigma(b + \sum_{i,j,k \in [-s..s]^2} x_{k+i,l+j,m+k} * k_{i,j,k})$$

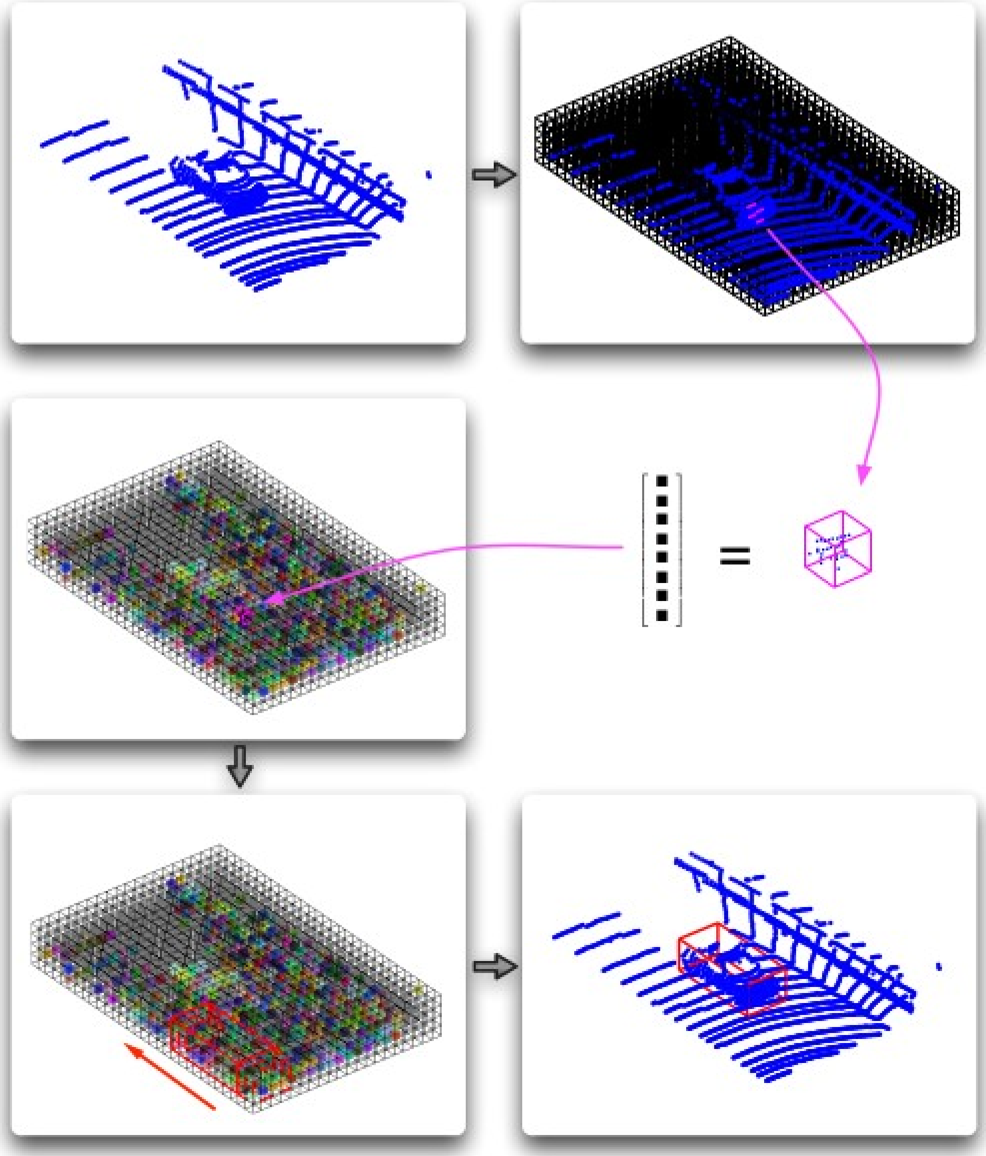
- Cubical complexity $O(n^3)$
- Needs padding -> no exploitation of sparsity
- Need a huge amount of computation
- Limit operations to 30x30x30 resolution
- Tradeoff to find between computation time and precision

Maturana & al, VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition, 2015

Wu & al, 3D ShapeNets: A Deep Representation for Volumetric Shapes, 2015

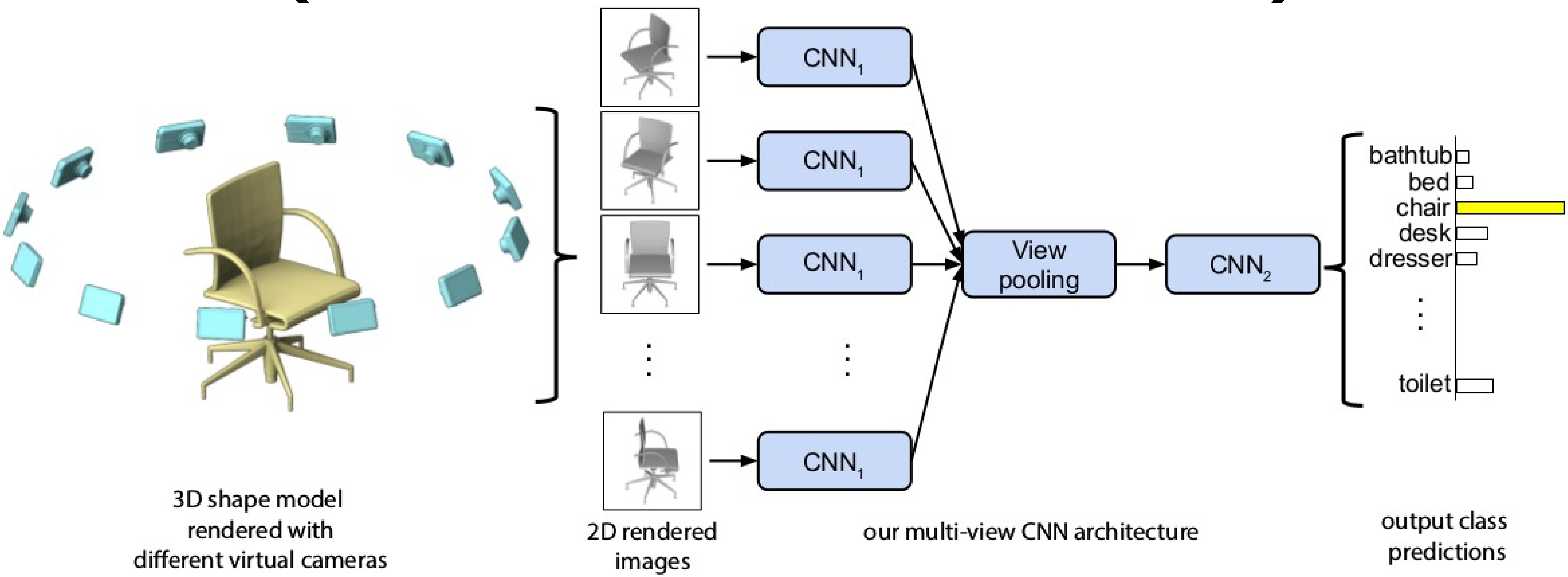
Sparse 3D Convolution

- Try reduce the complexity of convolution by exploiting the data sparsity
- Reduce the number of input points by selecting the interesting/specific parts of the cloud
- Interesting tracks but lower the complexity by reducing the precision...



Wang & al, Voting for Voting in Online Point Cloud Object Detection, 2016

Multiple 2D Convolution (2.5D Convolution)



- Improve performance on classification (better resolution of images)
- Still requires huge amount of computation (3D reconstruction + plenty of CNN)
- Does not work for segmentation

Su & al, Multi-view Convolutional Neural Networks for 3D Shape Recognition, 2015

Qi & al, Volumetric and Multi-View CNNs for Object Classification on 3D Data, 2016

Symmetric pooling solutions

Ideas of symmetric pooling

- As the main problem is the non-order of the points
  - Idea 1 : use a symmetric analysing function
    - tends to loose the locality
    - PointNet[++]
  - Idea 2 : order them before analyse
    - Theory : no order can be stable to point perturbation
    - Reality : but could be stable enough to give interesting result
    - PointCNN
  - Idea 3 : treat the input as a sequence in a reccurent network, trained with shuffling to learn symmetry
    - The approximation of the order is not stable either
    - The performance are terrible

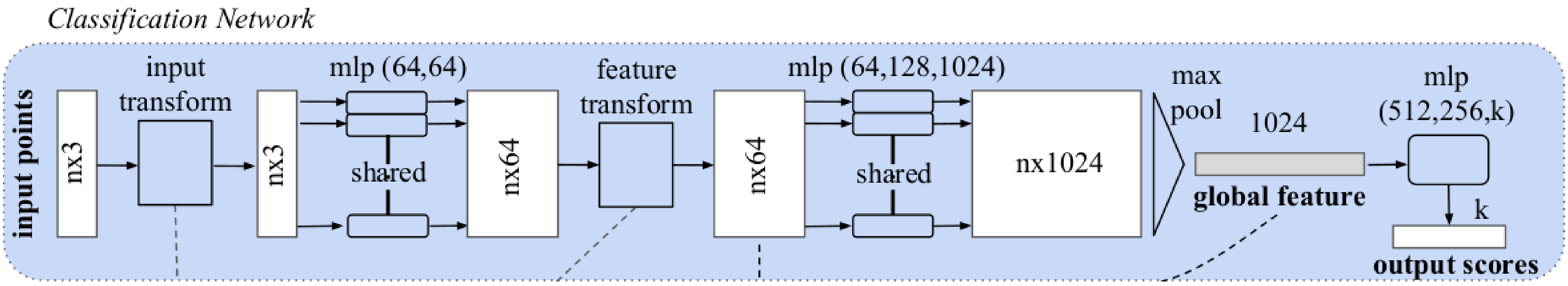
PointNet

Charles R. Qi

- Idea : instead of sorting points, learn a symmetric function g over transformed points h(x)

$$f(x_1, \ldots, x_n) \approx g(h(x_1), \ldots, h(x_n))$$

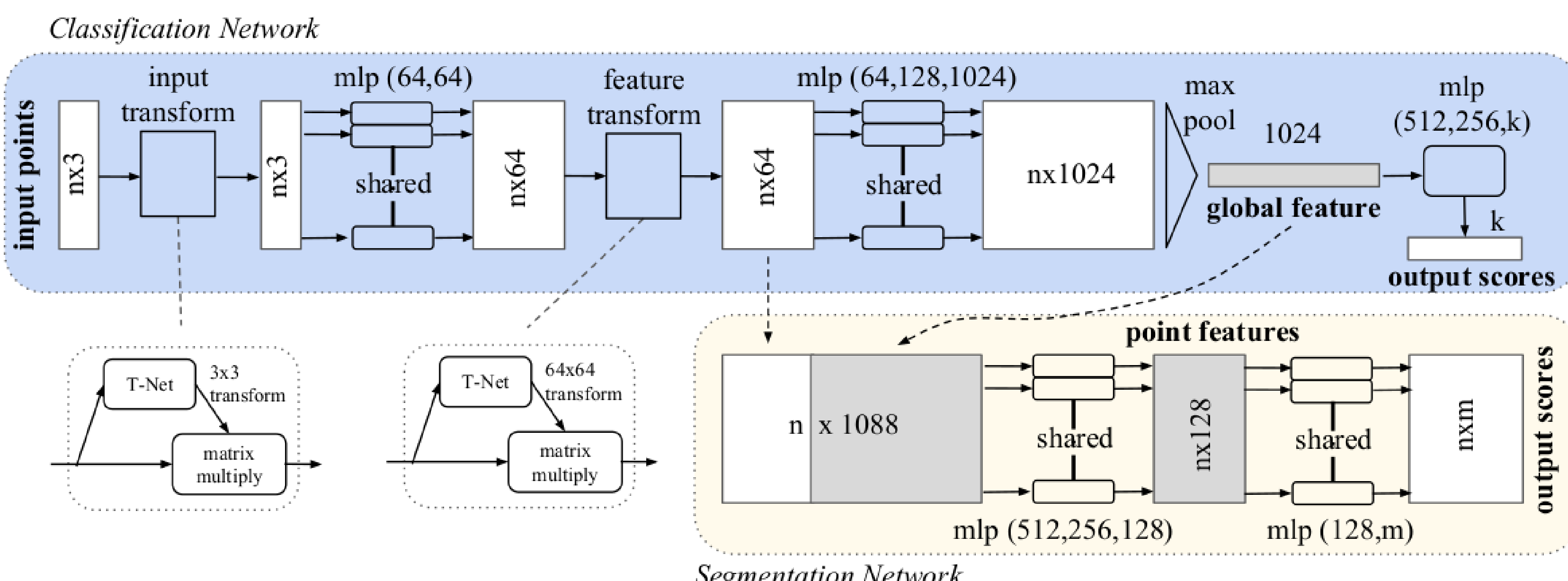
- approximate h by a shared MLP and 2 shared learned linear transformations (normalization)
- Features are ordered by max pooling
- g = max_pool ○ MLP



PointNet (2)

Qi & al, PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, 2017

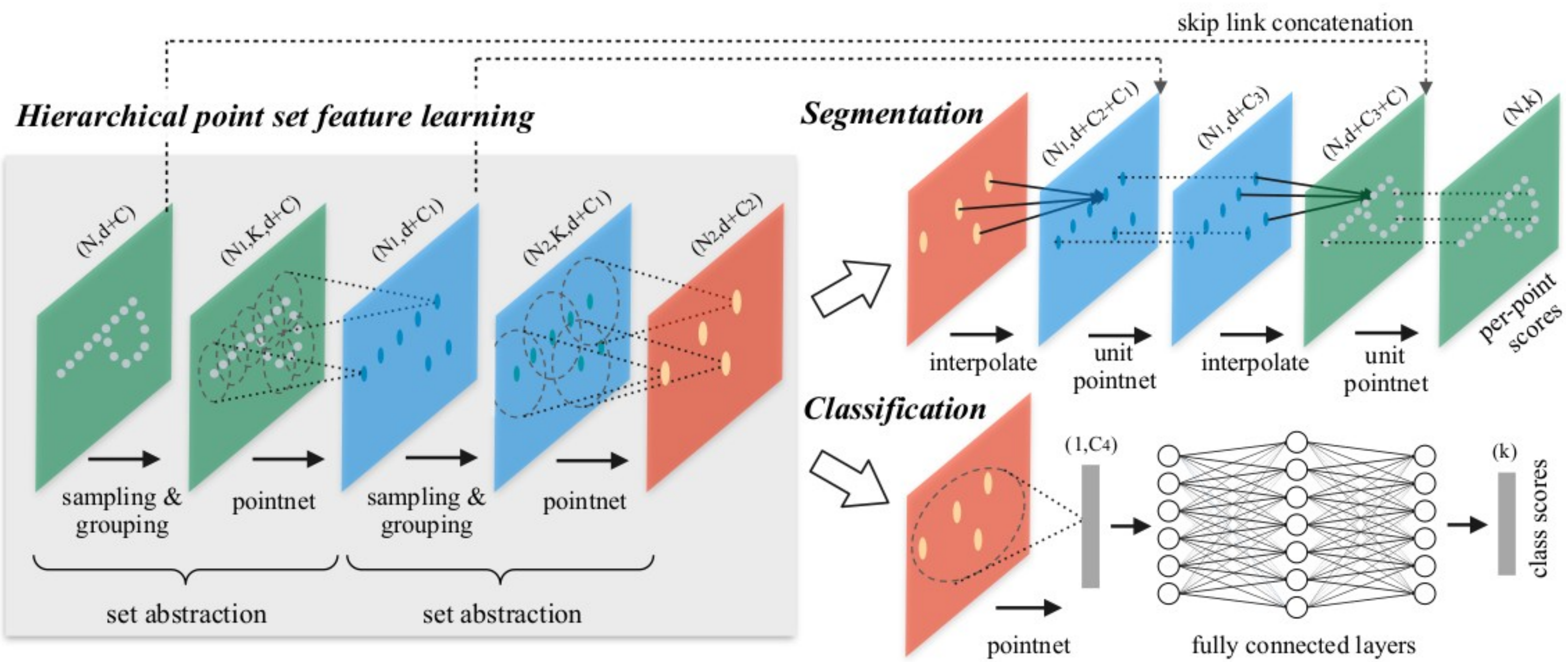
- Reach same overall accuracy as 3D convolution with 440MFlops/sample vs 62057 Mflops/sample for Multiview CNN
- Segmentation extension mixing local and global features
- Drawback
  - does not capture any local feature
  - Cant recognize fine grain patterns



PointNet++

Qi & al, PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space, 2017

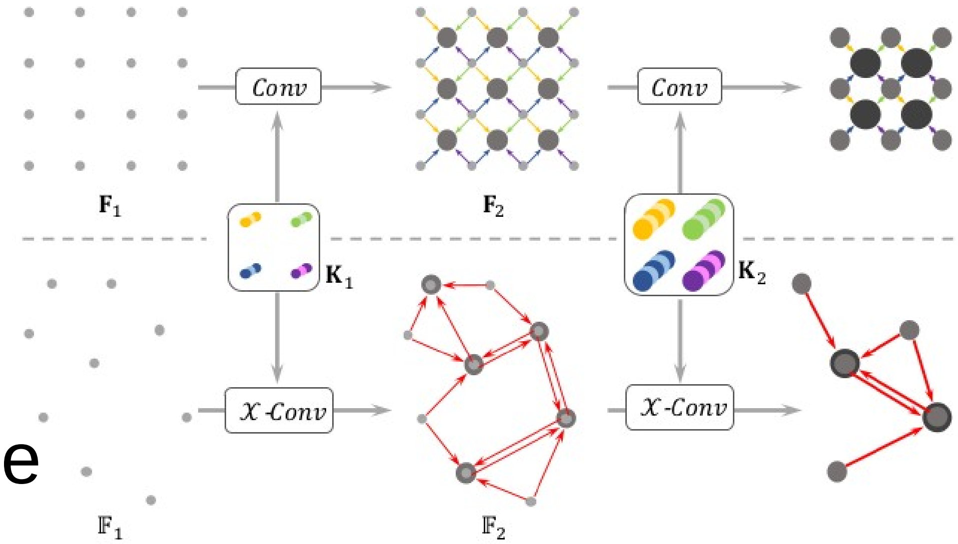
- Hierarchical version of PointNet
- Apply Pointnet recursively on the nested partitions → local features
- Combine learned feature from different scales
- Better perf but still does not understand the relationship between points
- Gain almost 3 % on global accuracy on ModelNet40 → 91.9 %



PointCNN

Li & al, PointCNN: Convolution On X - Transformed Points, 2018

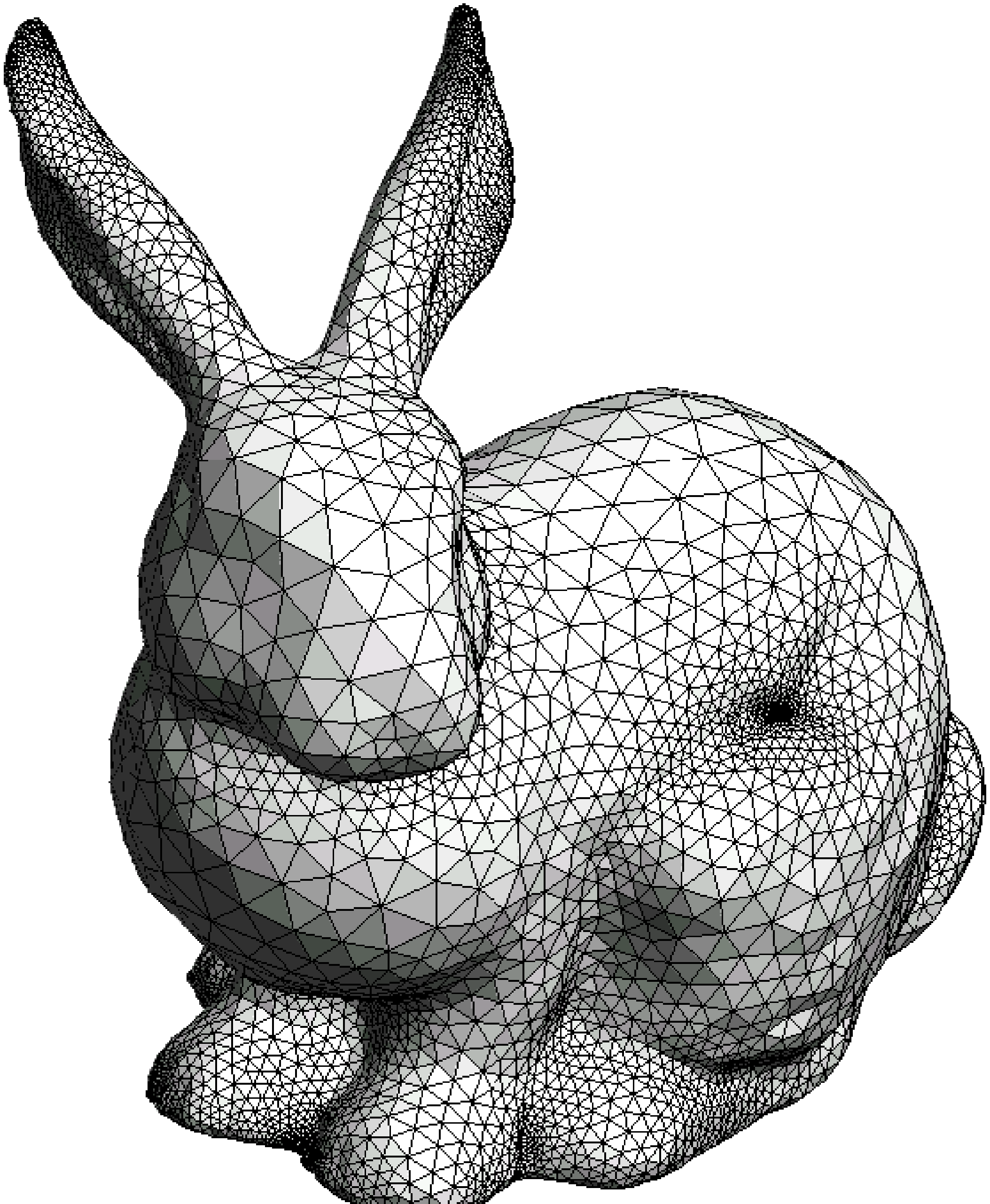
- Convolution on the K proximate neighbours
- Problem : the neighbours are not ordered
- Try to learn a transformation X
  - Weighting the inputs
  - creating a canonical order
- Apply ordinary convolution on the result (X+Conv=XConv)
- Apply pooling on the point set
- Obtain 92.2 % overall accuracy on ModelNet40 (very good)



Graph convolution solutions

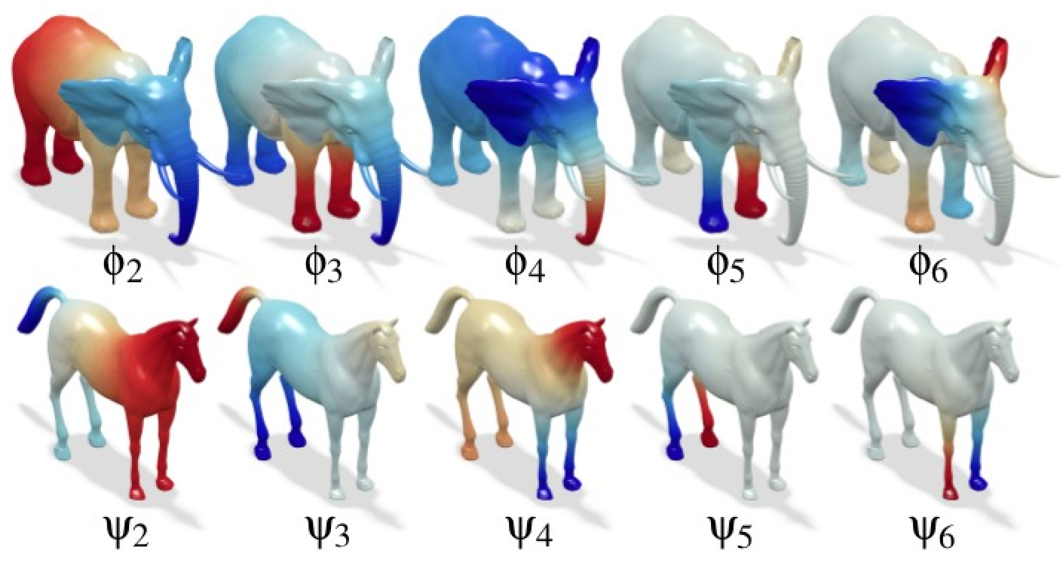
Idea of Graph convolution

- Build a graph structure with the point cloud
- Capture the locality in the graph adjacency
- Apply new techniques of graph convolution



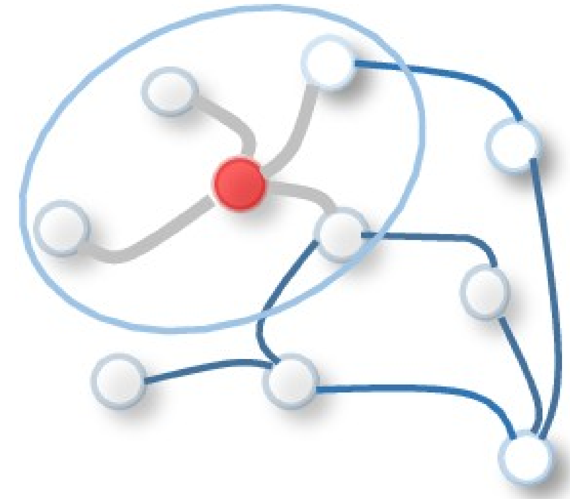
Spectral vs Spatial

- Spectral method has been the first to be developped, based on algebraic / spectral graph theory (80's)
- Contrary to spectral, spatial is stable to graph change
- Nowadays almost only spatial methods are used



Laplacian eigenbases

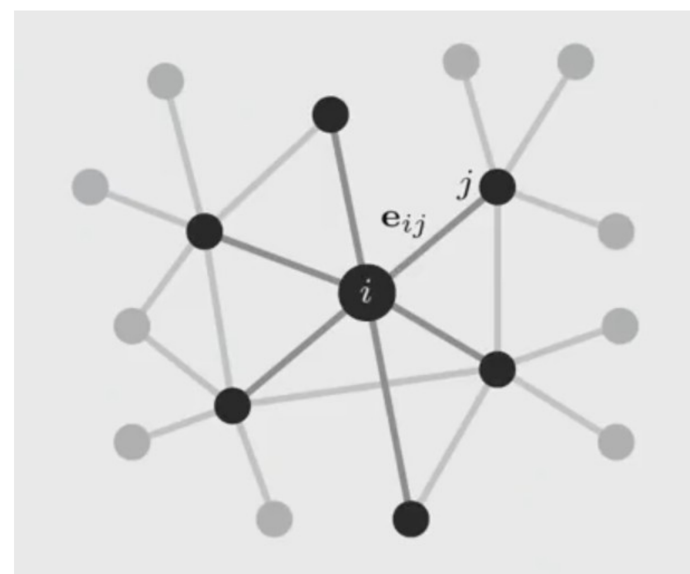
Neural Message Passing Network



- Generic recipe for spatial graph convolution
- Convolves the central node x_i with its neighbors x_j in N(v)

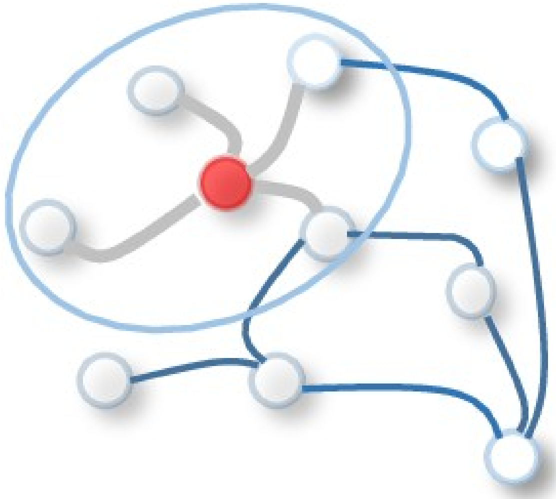
$$x_i^k = \gamma(x_i^{k-1}, \square_{j \in N(i)} \phi_\theta(x_i^{k-1}, x_j^{k-1}, e_{i,j}))$$

- □ is a symmetric normalized operator like mean or max
- Nice complexity O(m)



Gilmer & al, Neural message passing for quantum chemistry, 2017

Formalism



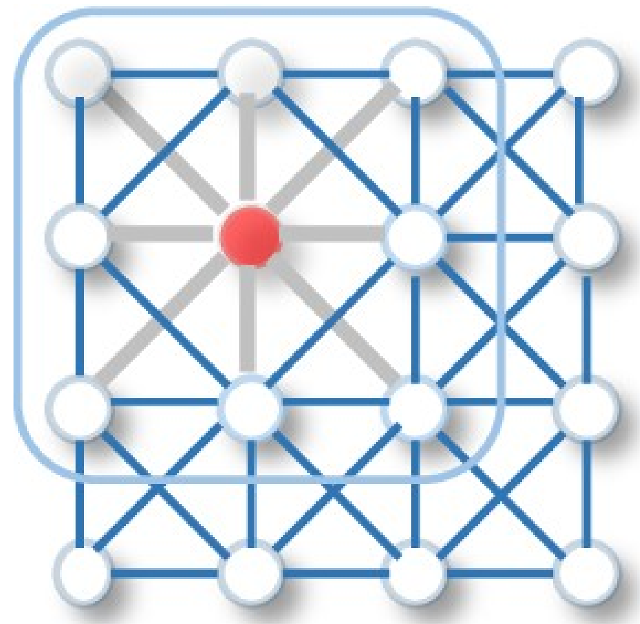
- Every node has a feature vector changing at each iteration (convolutional step)
- x_i^t is feature vector of node i at convolutional step t
- X^t is the feature map of all nodes at step t
- Every edge between x_i and x_j has a feature vector $e_{i,j}$
- Convolution step which convolves the central node x_i with its neighbors x_j in N(v)

$$x_i^{t+1} = \gamma_{\theta_\gamma}(x_i^t, \square_{j \in N(i)} \phi_{\theta_\phi}(x_i^t, x_j^t, e_{i,j}))$$

- $\square$ is the aggregator function (commutative & normalized : max, average..)
- Φ is the message function (learnable parameters)
- γ is the update function (learnable parameters)
- Learnable parameters are θ_γ and θ_Φ

This recipe includes Euclidian CNN

- $\Phi_\theta(x_i, x_j, e_{ij}) = x_j * \theta_{ij}$
- □ = sum
- Regular graph (no weight)
- Every vertex is self looped

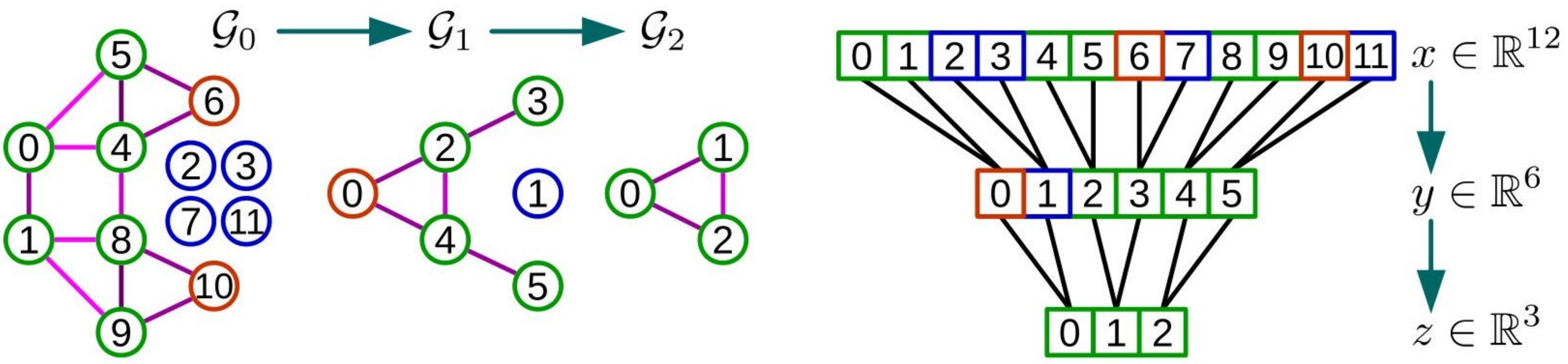


$$x_{k,l}^{t+1} = \sum_{i,j \in [-s..s]^2} x_{k+i,l+j}^{t} * \theta_{i,j}$$

35	40	41	45	50
40	40	42	46	52
42	46	50	55	55
48	52	56	58	60
56	60	65	70	75

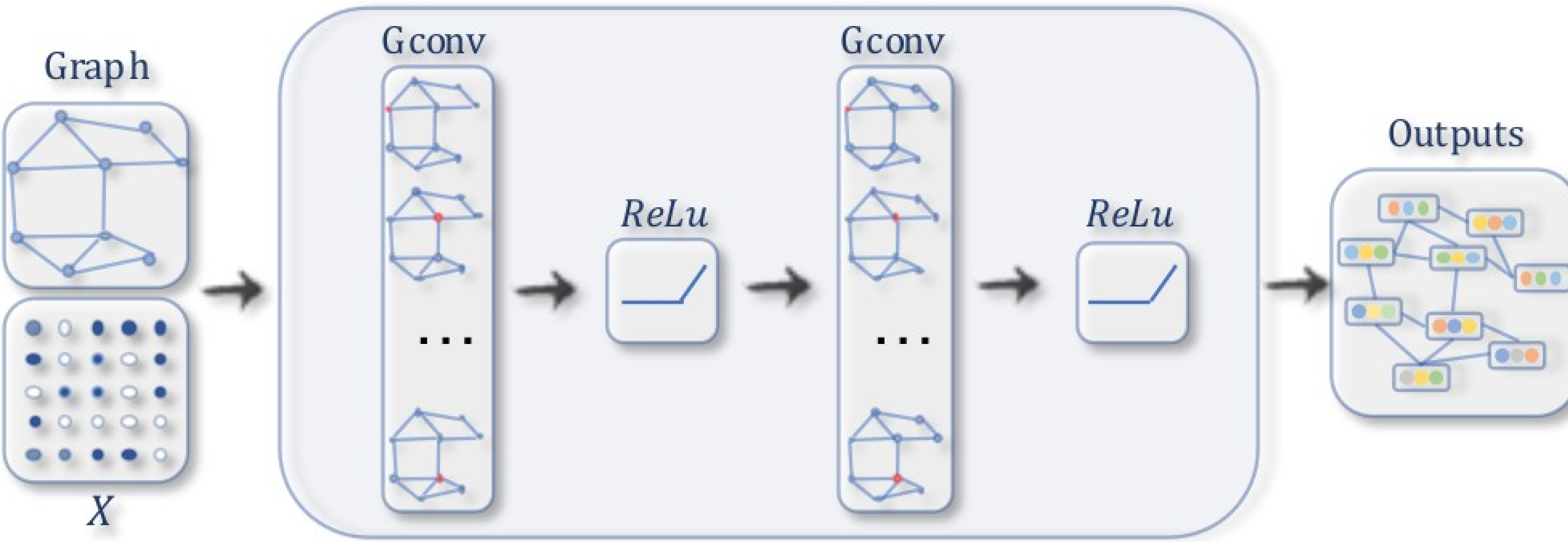
→ Euclidian CNN

Graph pooling



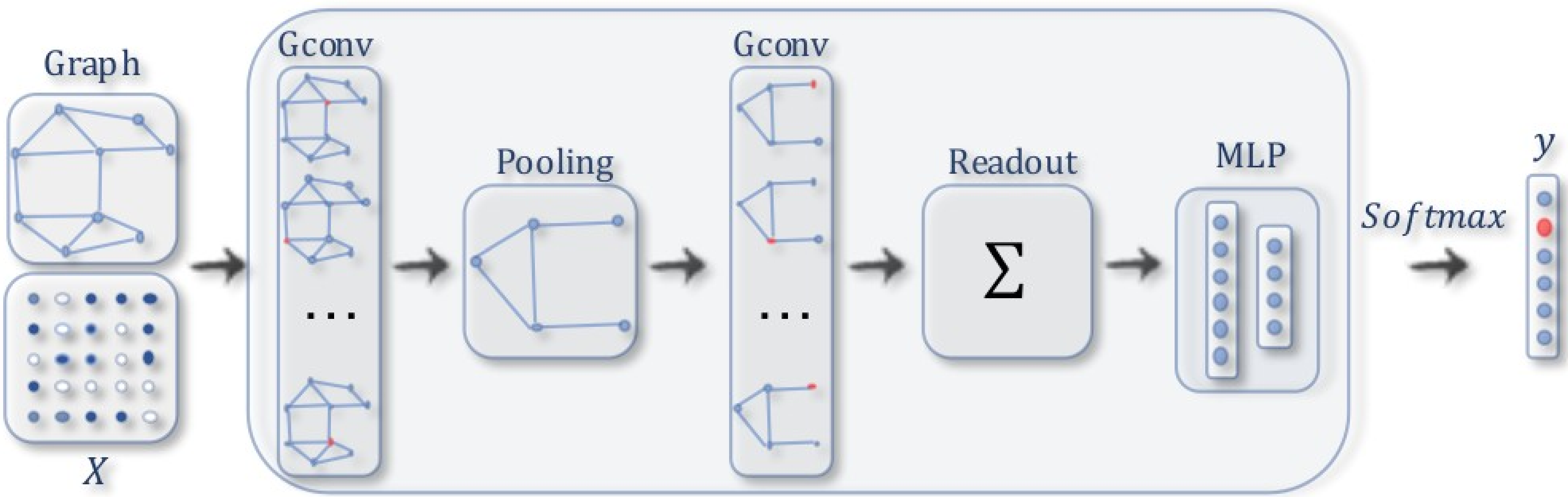
- Produce a sequence of coarsened graphs
- Graclus algorithm
- Fusion of vertices
  - Connected by a common edge
  - Max, sum or average pooling of collapsed vertices

Network inference architecture



- Successive feature maps induce a new graph
- Semi-supervised learning

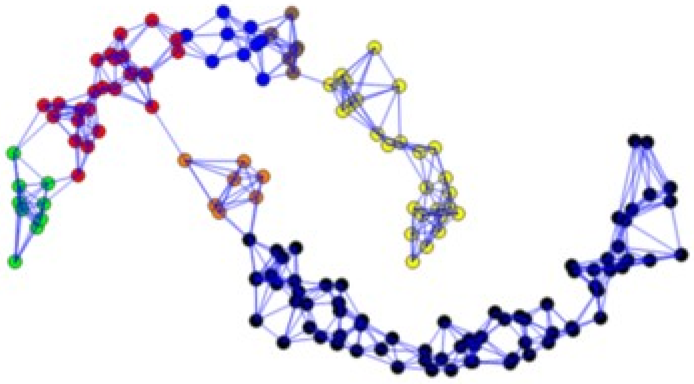
Graph classification architecture



- Non Euclidian convolution with pooling
- Readout to flatten the feature maps
- Multi-layer perceptron with softmax for classification
- Shape recognition (particle interactions)

Dynamic extension

Wang & al, Dynamic Graph CNN for Learning on Point Clouds, 2019



- It is shown to work better if the graph is re-computed at every step
- The network learns how to build the graph
- Cluster similar features in the feature space
- Very resource demanding (multiple KNN)