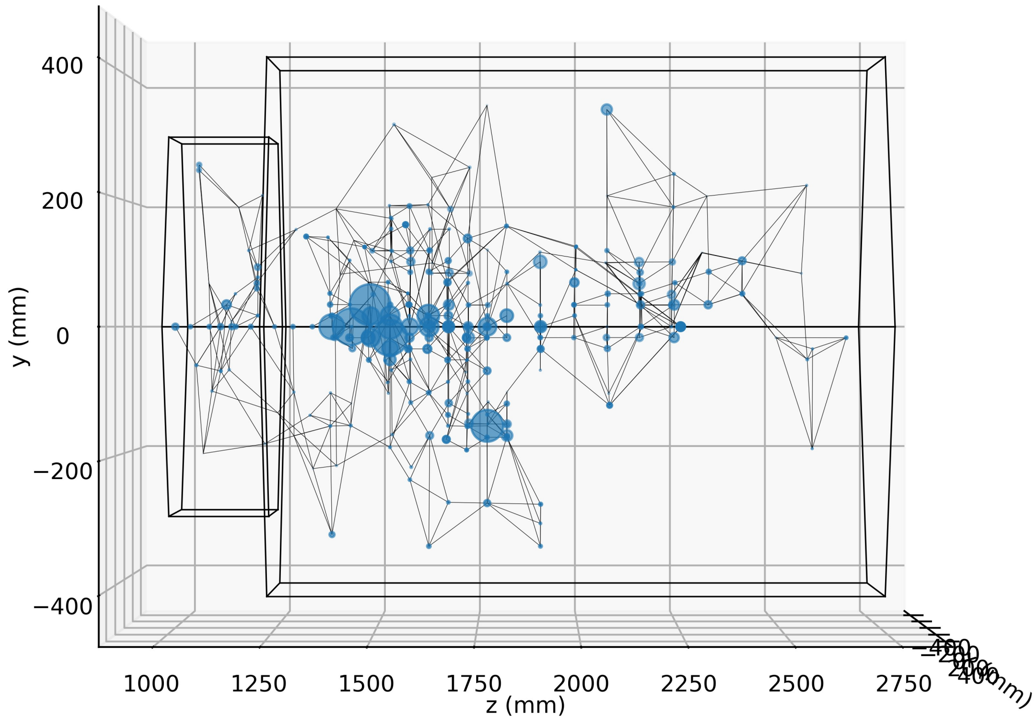
Graph Convolutional Neural Networks for HEP

Frédéric Magniette



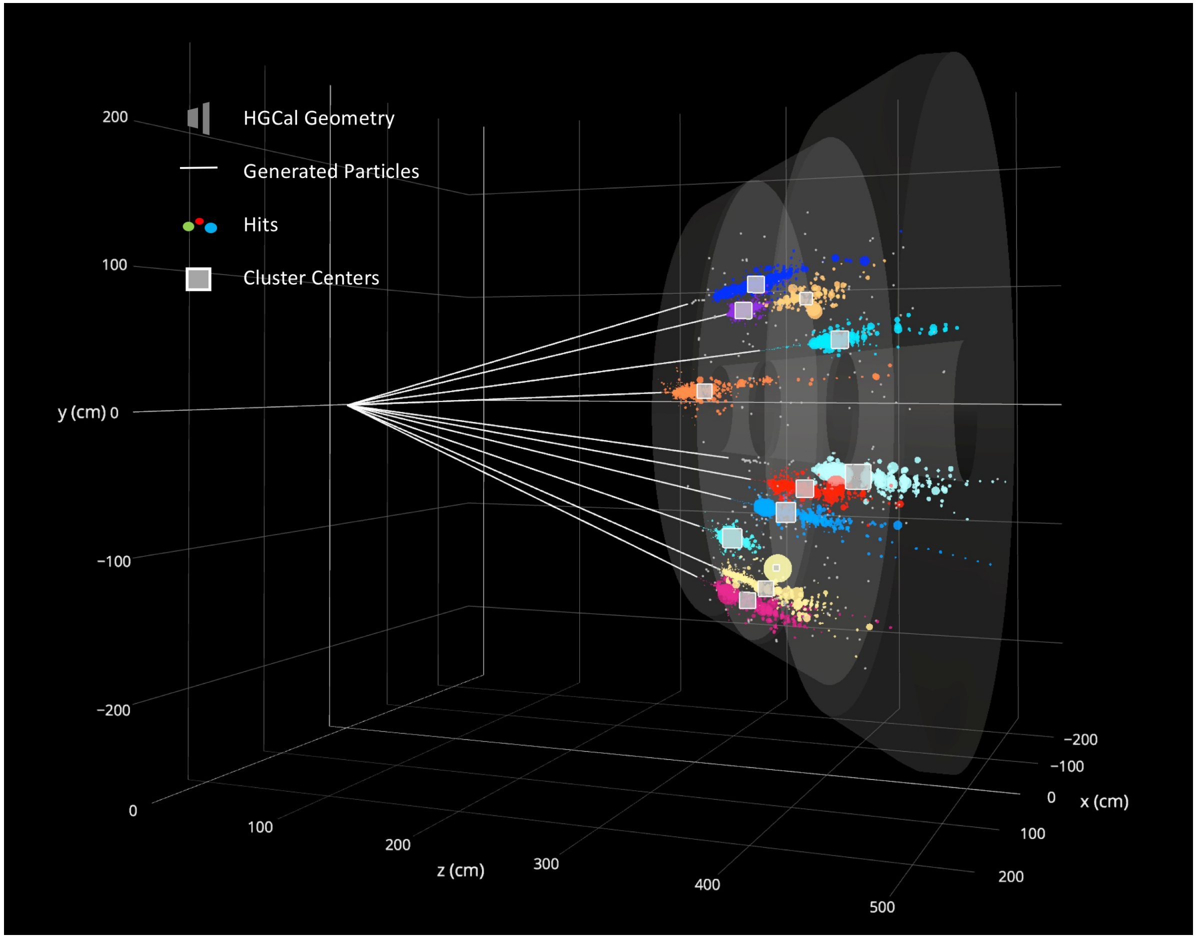
ÉCOLE POLYTECHNIQUE

IP PARIS

Laboratoire Leprince-Ringuet

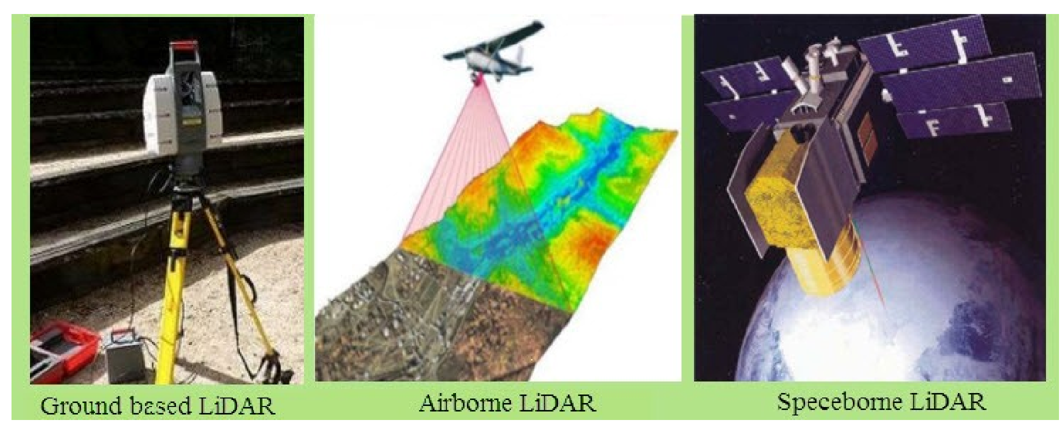
Introduction

- Uniformity of HEP data
- 3D point with energy measurements and timing
- Different granularity
- Barycenter of sensors
- Fixed geometry
- How to handle such data with neural networks?



Similar Applications

- 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
- New dedicated algorithms from 2015 to now

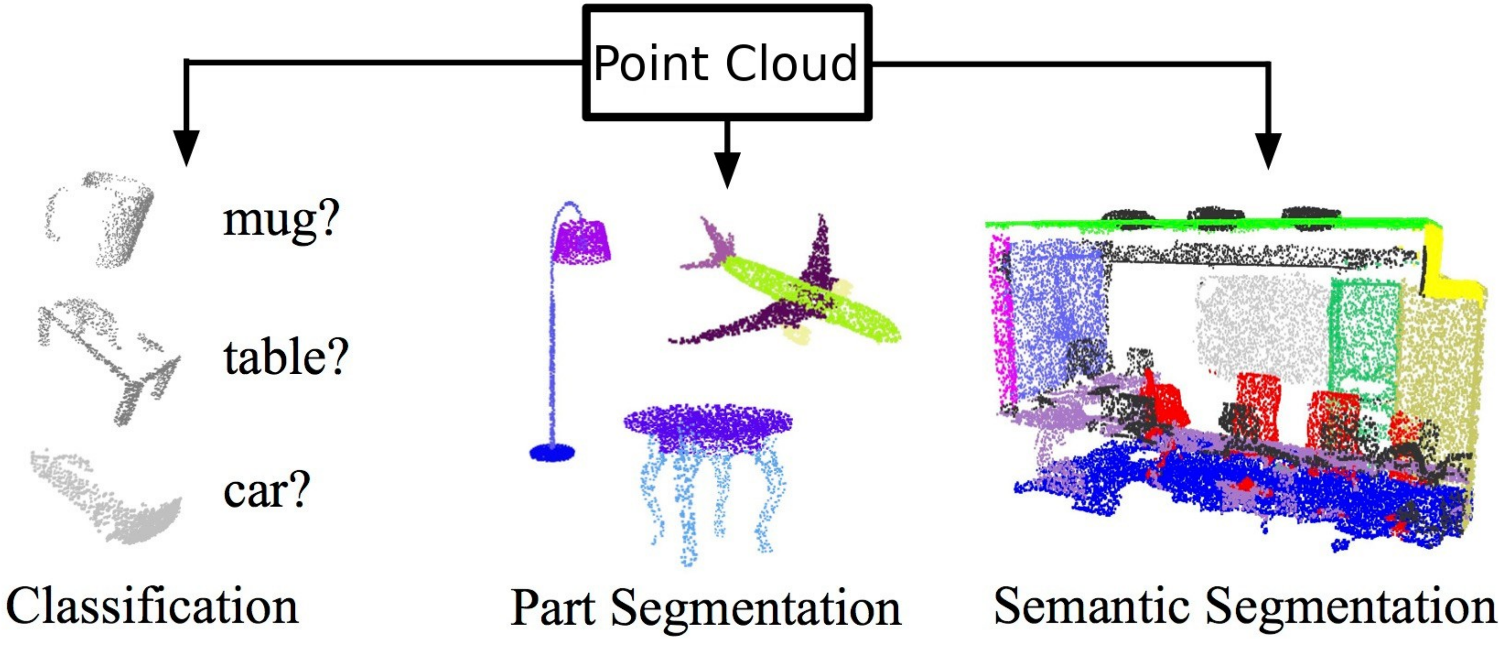


Point cloud data

- Points P_i in R^k ($k \geq 3$)
  - 3D coordinates
  - (k-3) features : energy measurements, timing, calibration...
- 4 main properties
  - unordered : need for a permutation invariant operator
  - Interaction among points : the metric distance defines meaningful neigbourings
  - Invariance under transformation : some rotations and translations should not modify the result
  - Sparsity

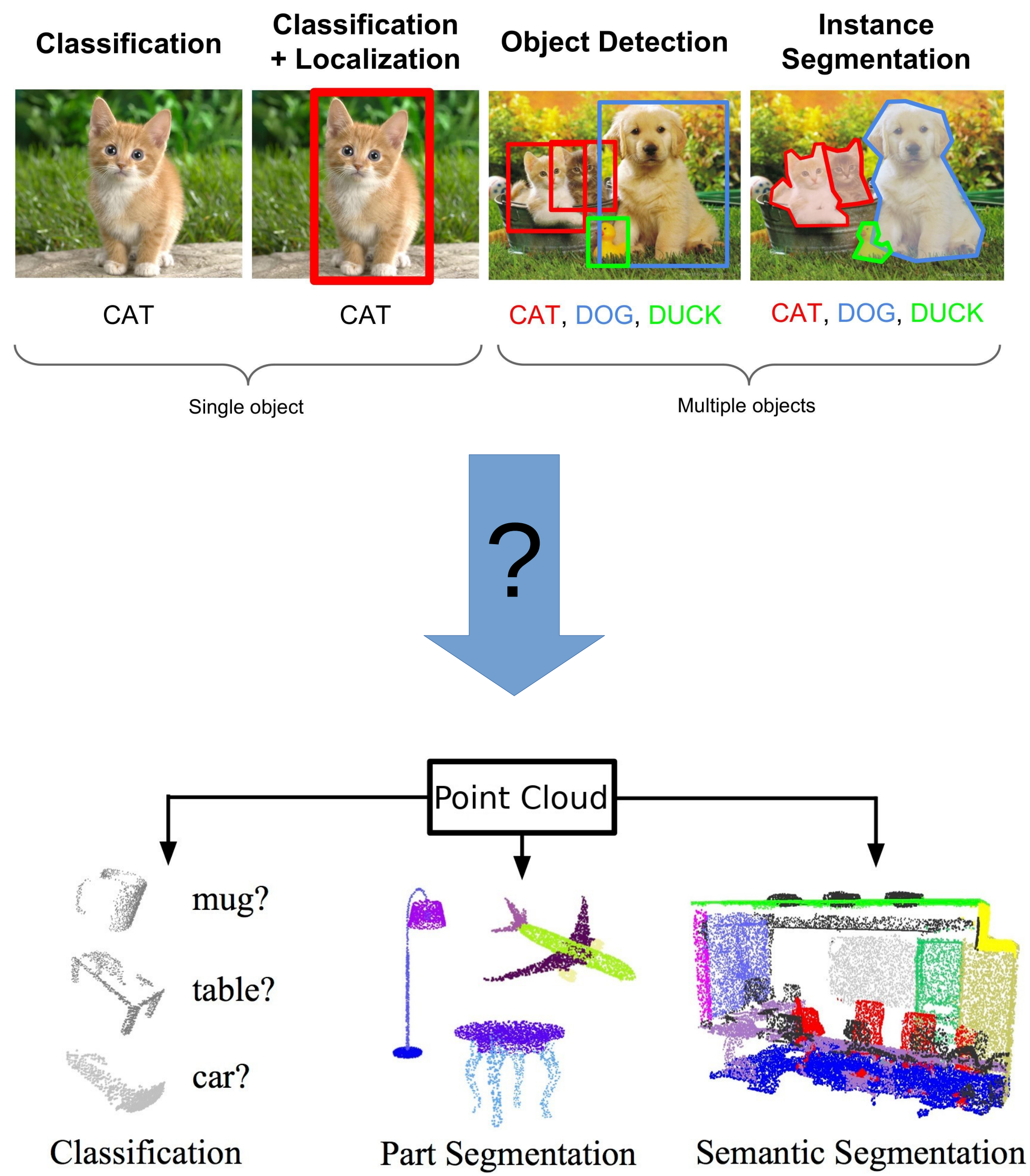
Problematics

- Three problematics
  - Classification
  - Part segmentation
  - Semantic segmentation



Question

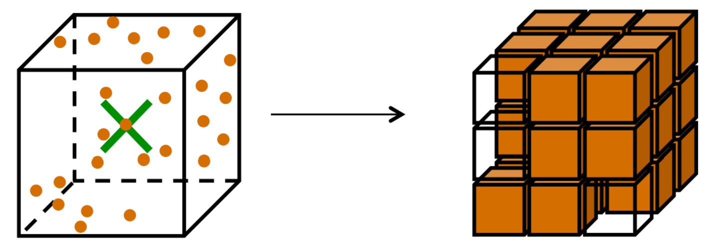
- How to transpose the tremendous success obtained with 2D image convolution to 3D point cloud ?
- Before 2015 : handmade feature
- A lot of work based on neural network from 2015 to now



Point Cloud Neural Networks

Three main techniques

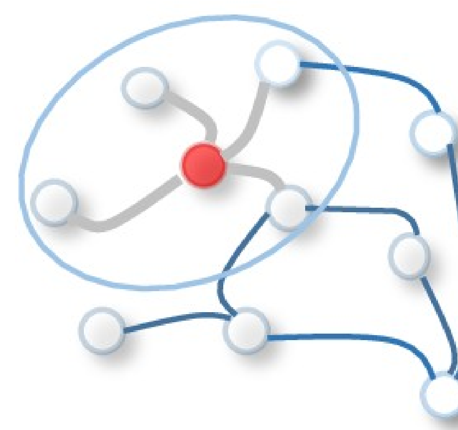
- Voxelization and 3D Convolution (2015-2016)



- Symmetric pooling (2017-2018)

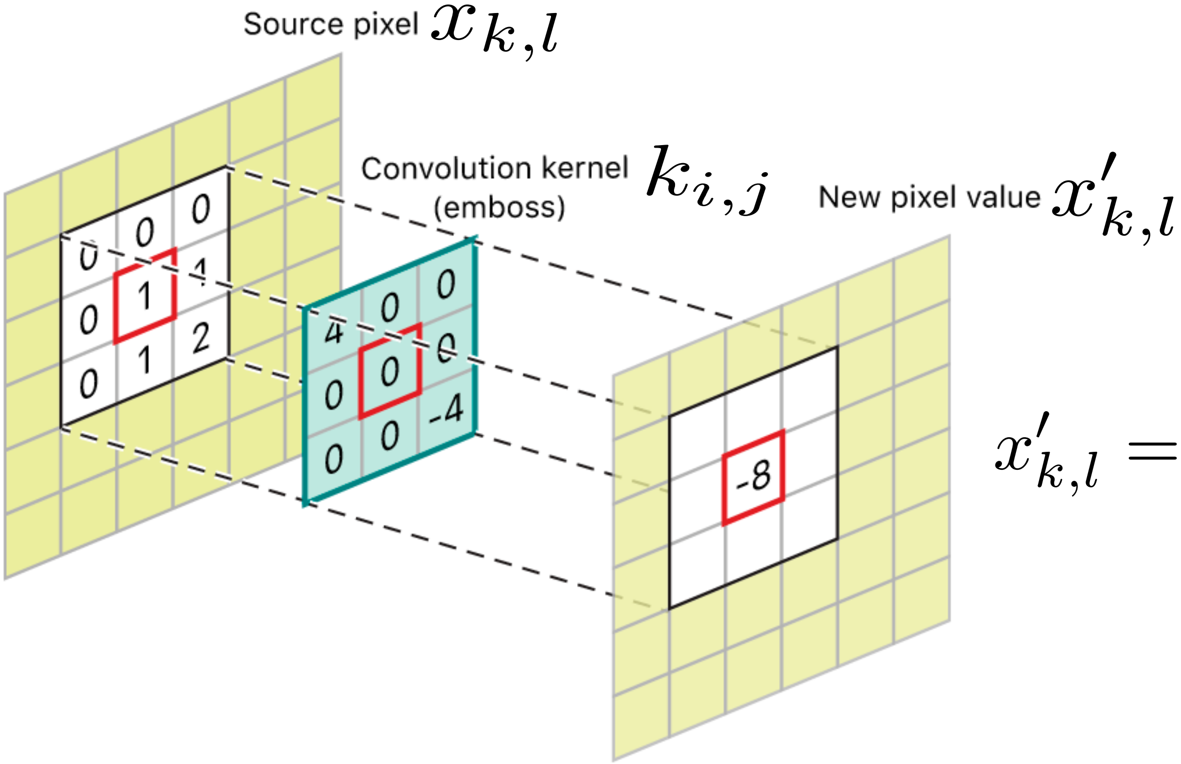
$$f(x_1, \ldots, x_n) \approx g(h(x_1), \ldots, h(x_n))$$

- Graph Convolution (2017-now)



Precision

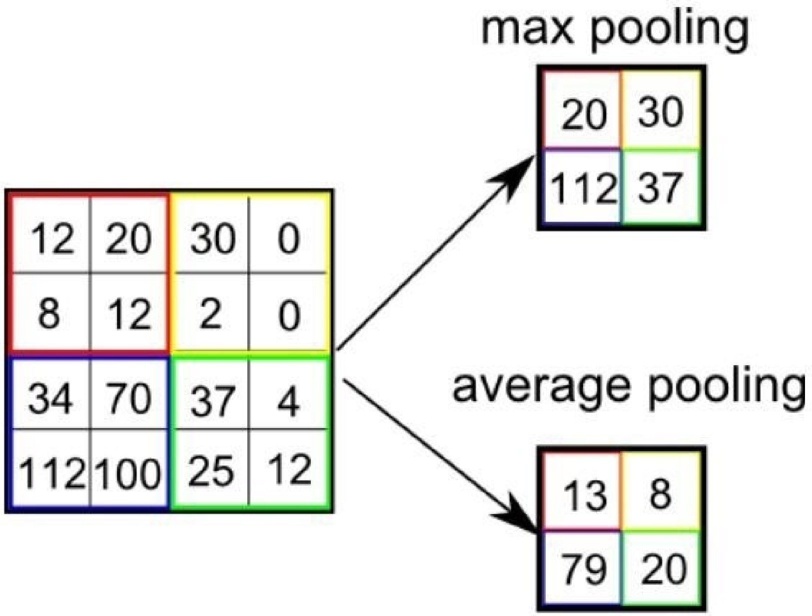
Pixels Convolution



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

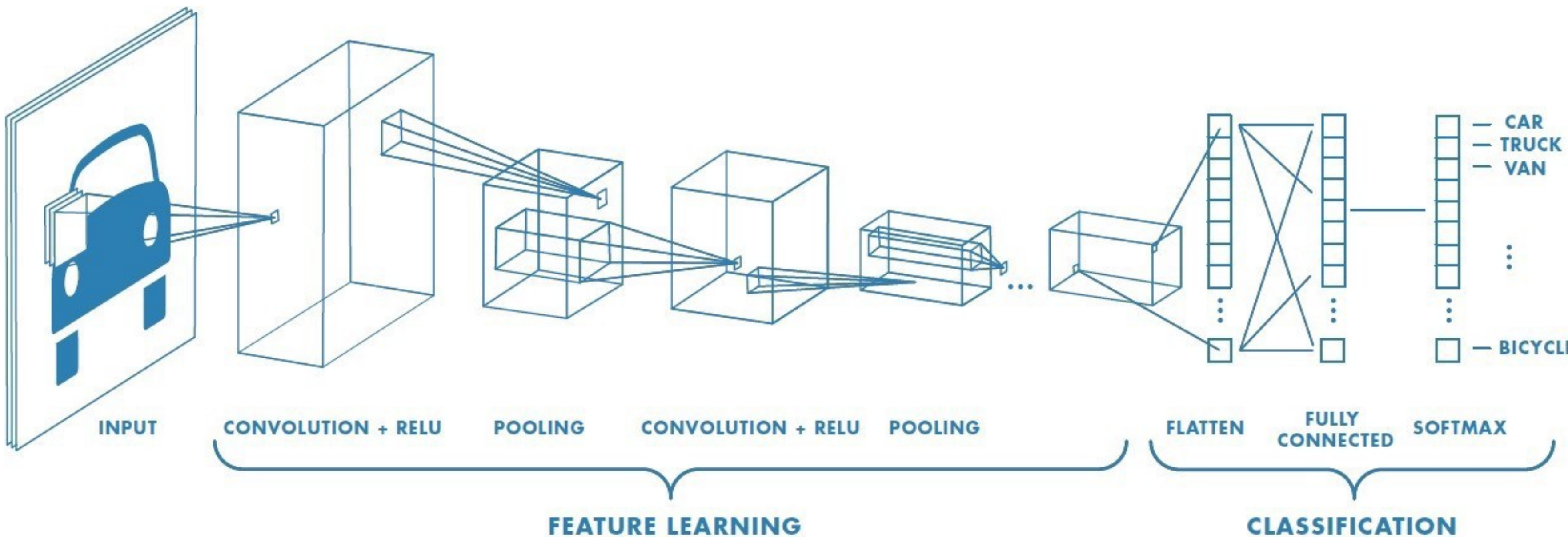
- Apply kernel on image (like the convolution filter)
- kernel is learnable ($k_{i,j}$)
- Filter is shared over the whole picture
- Idea : creating maps of features (one kernel per feature)

Pooling



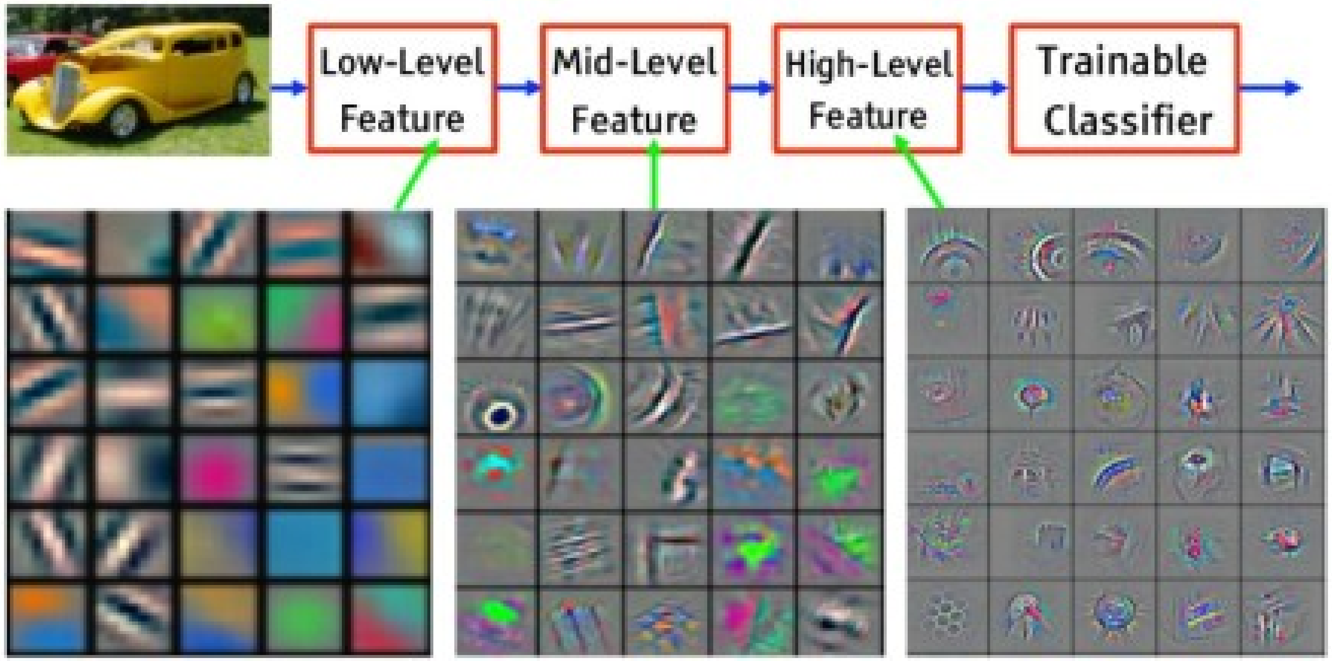
- Reduce the dimensionality of the feature maps
- Move to higher level of abstraction
- Max pool is widely used

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 Euclidean nature
  - Euclidean space → global translation-invariance (stationarity) → convolution
  - local translation-invariance → pooling

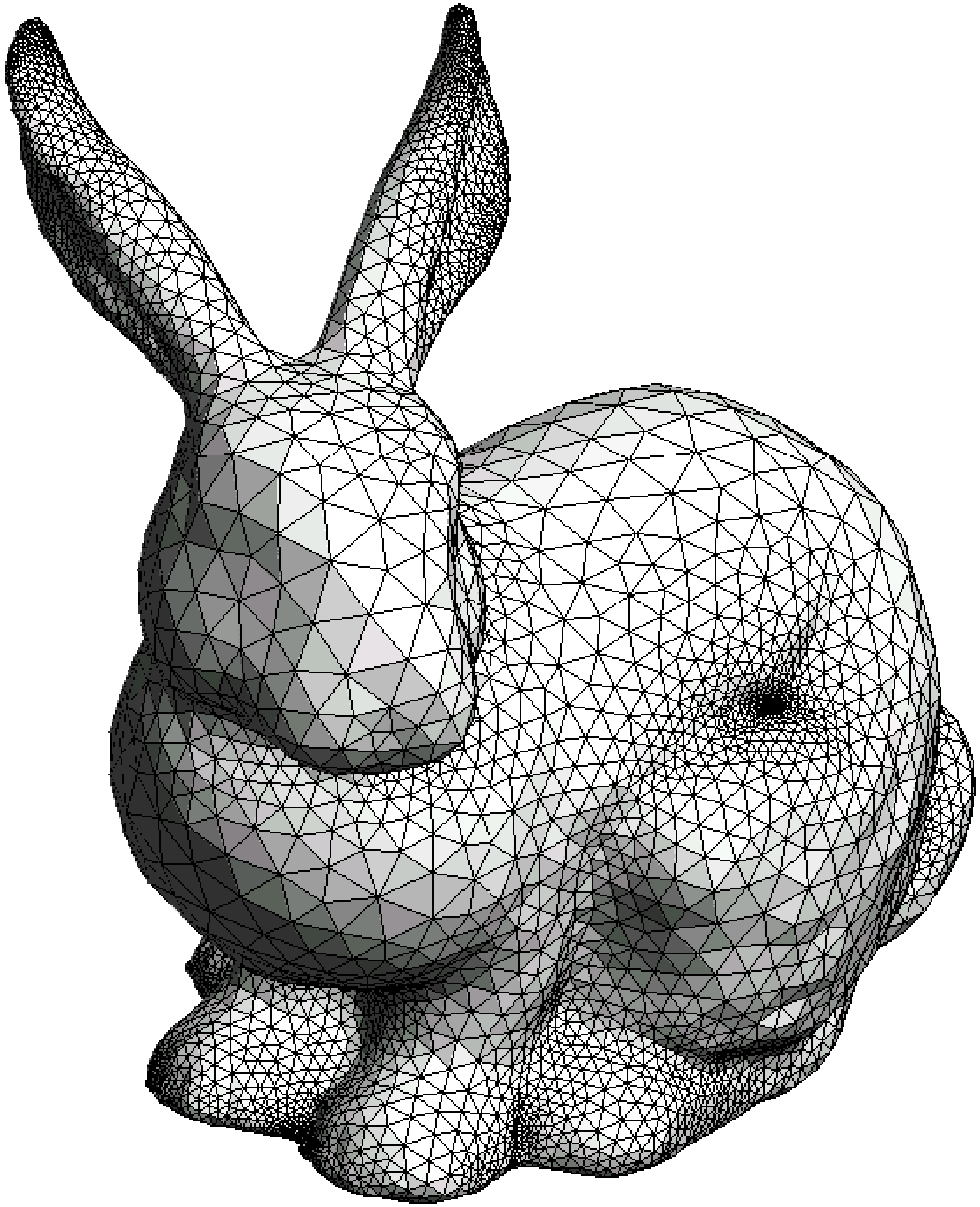
- Dream complexity
  - O(1) parameters per filter (independant of image size)
  - O(n) complexity in time per layer (n=#pixels)

Cat

Cat

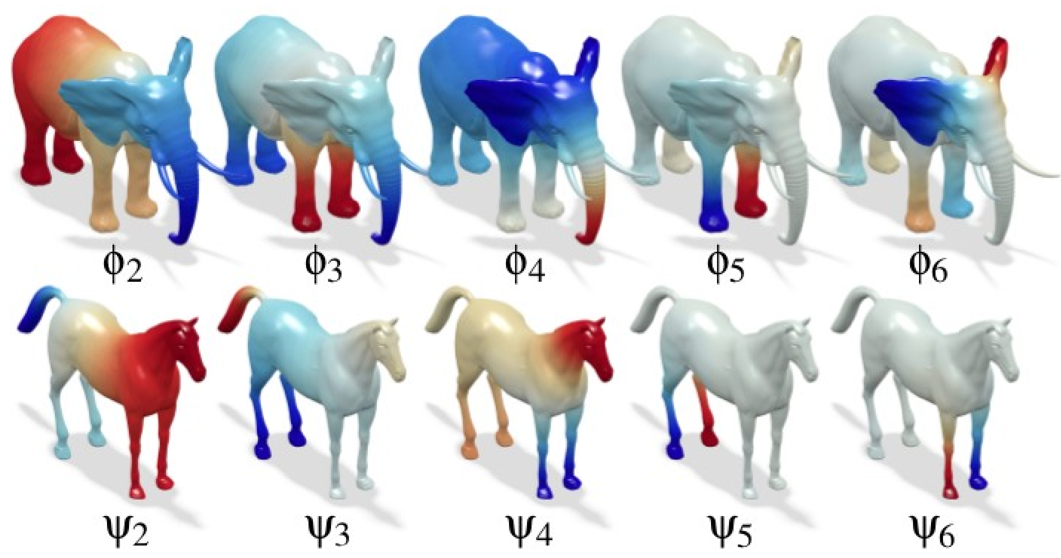
Idea of Graph convolution

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



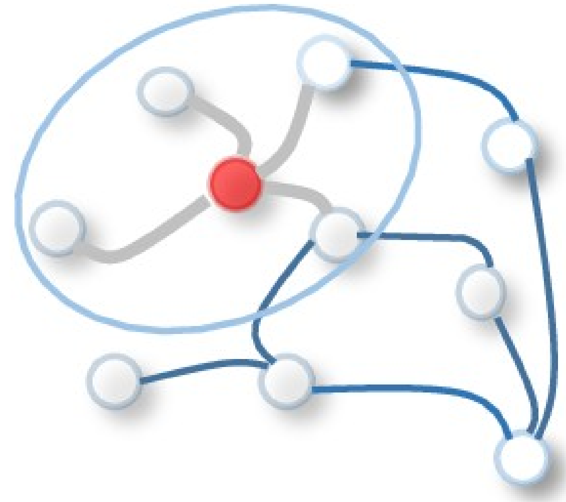
Spectral vs Spatial

- 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

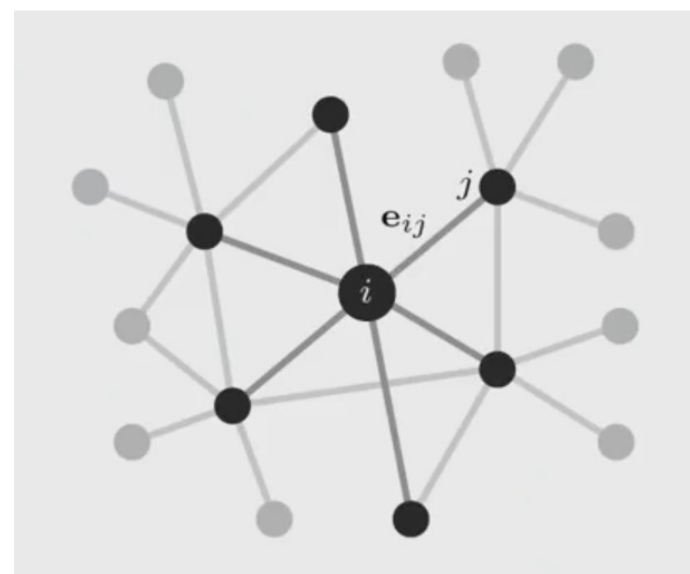


Laplacian eigenbases

Neural Message Passing Network



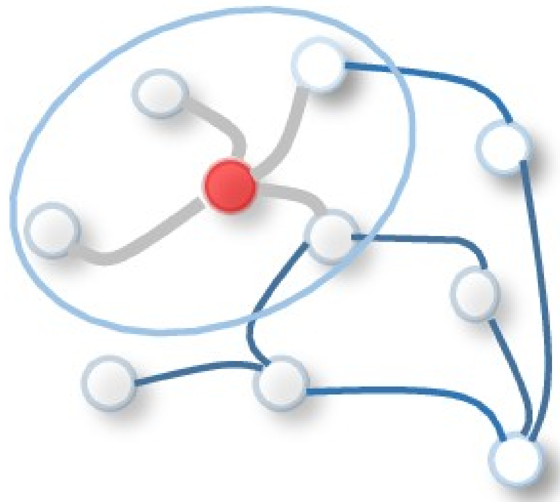
- Generic recipe for spatial graph convolution
  - Convolves the central node x_i with its neighbors x_j in N(v)
  - Iterate



- Nice complexity O(m)

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

Formalism

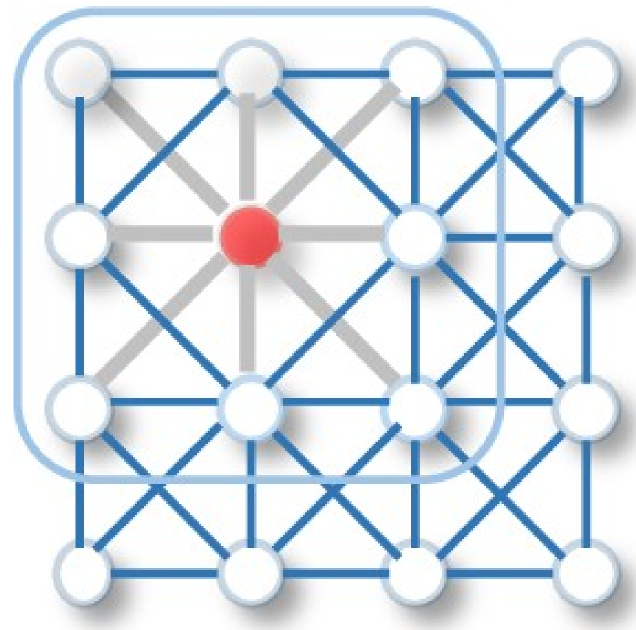


- Every node has a feature vector changing at each iteration (convolutional step)
- x_i^t is the feature vector of node i at convolutional step t
- Every edge between x_i and x_j has a feature vector $e_{i,j}$
- Convolution step which convolves the central node x_i with its neighbors x_j in N(v)

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

- □ 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



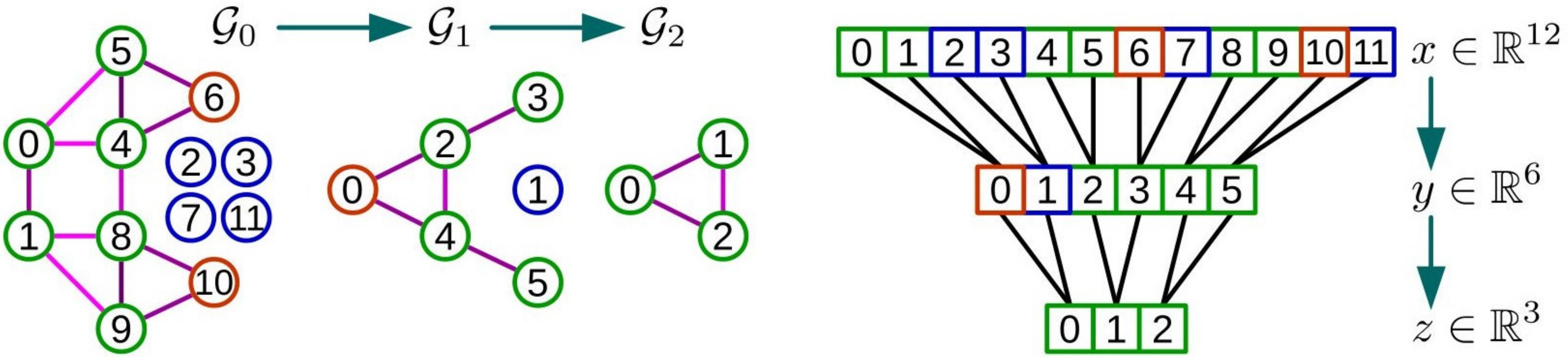
- $\Phi_\theta(x_i, x_j, e_{ij}) = x_j * \theta_{ij}$
- $\square$ = sum
- Regular graph (no weight)
- Every vertex is self looped

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

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

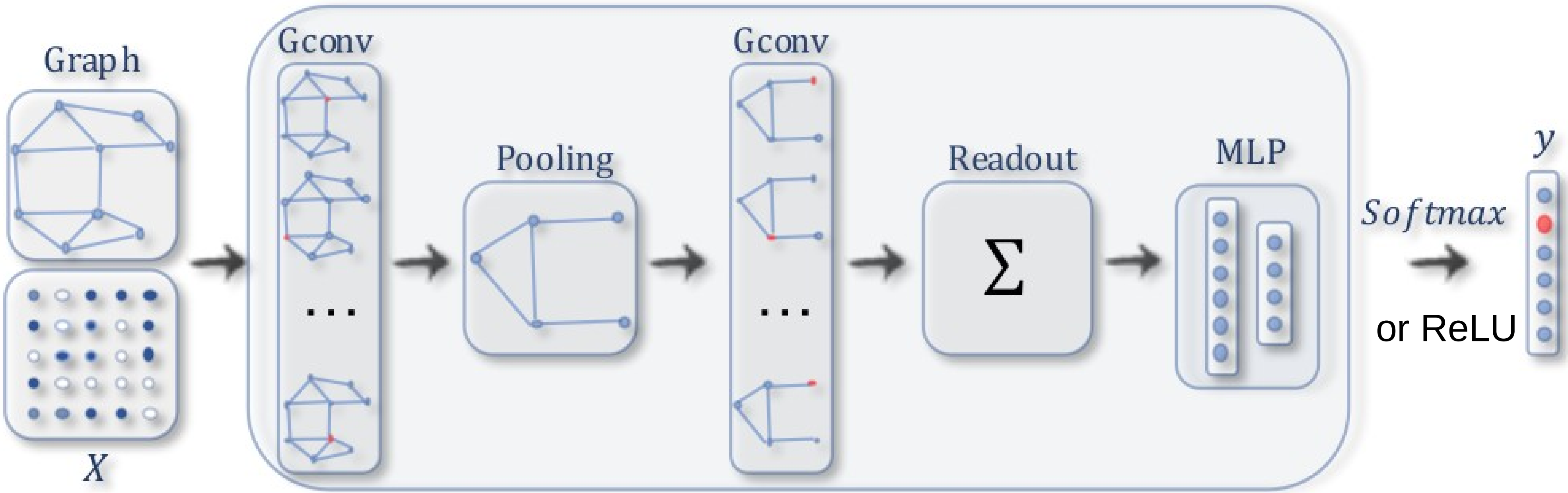
→ Euclidian CNN

Graph pooling



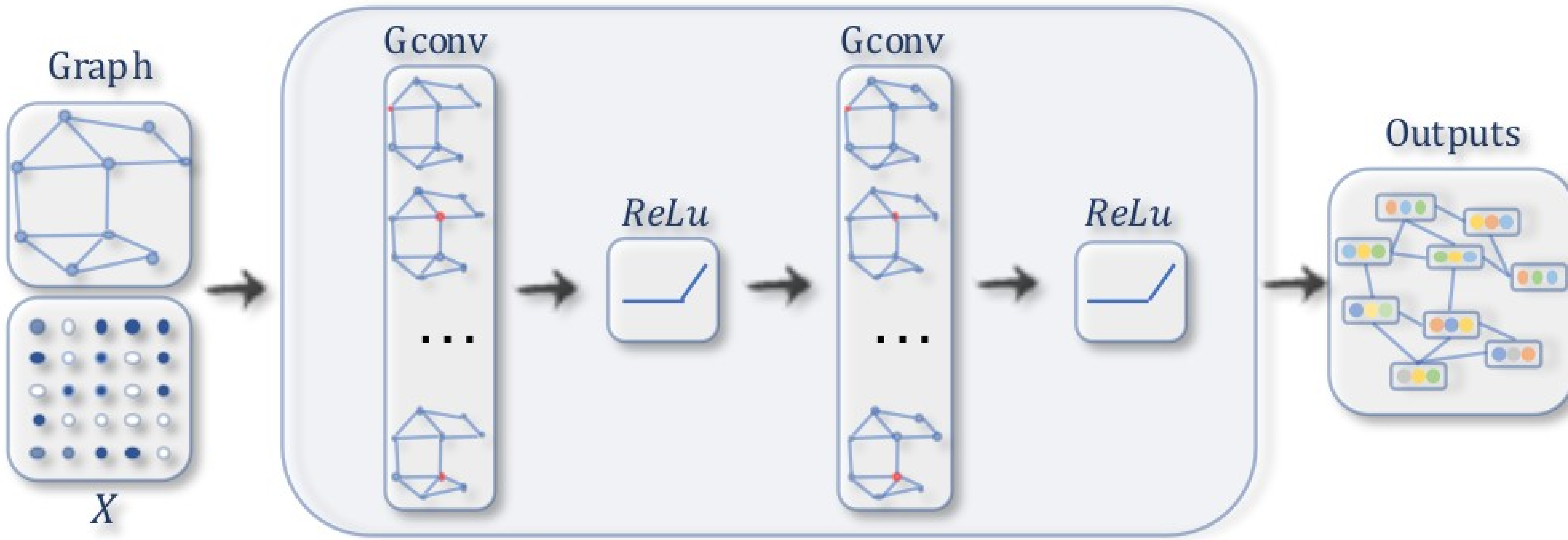
- Produce a sequence of coarsened graphs
- Graclus algorithm
- Fusion of vertices
  - Connected by a common edge
  - Max, sum or average pooling of collapsed vertices

Graph classification architecture



- Non Euclidian convolution with pooling
- Readout to flatten the feature maps
- Multi-layer perceptron
- Can be used for classification (Softmax) and regression (ReLU)

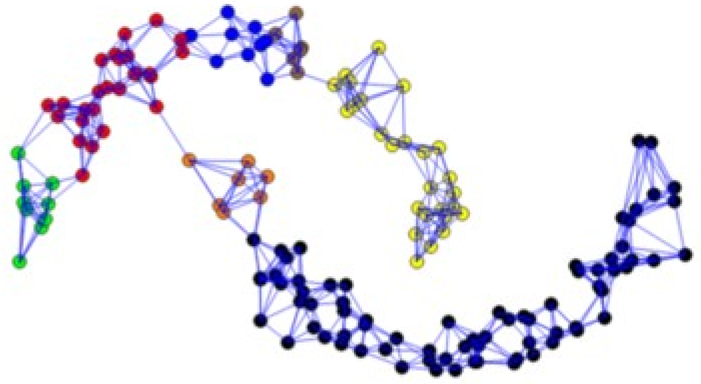
Network inference architecture



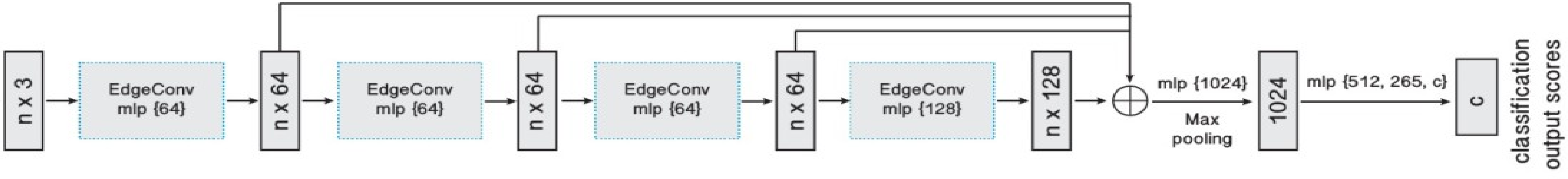
- Successive feature maps induce a new graph
- Semi-supervised learning
- Can be used successfully for segmentation

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)

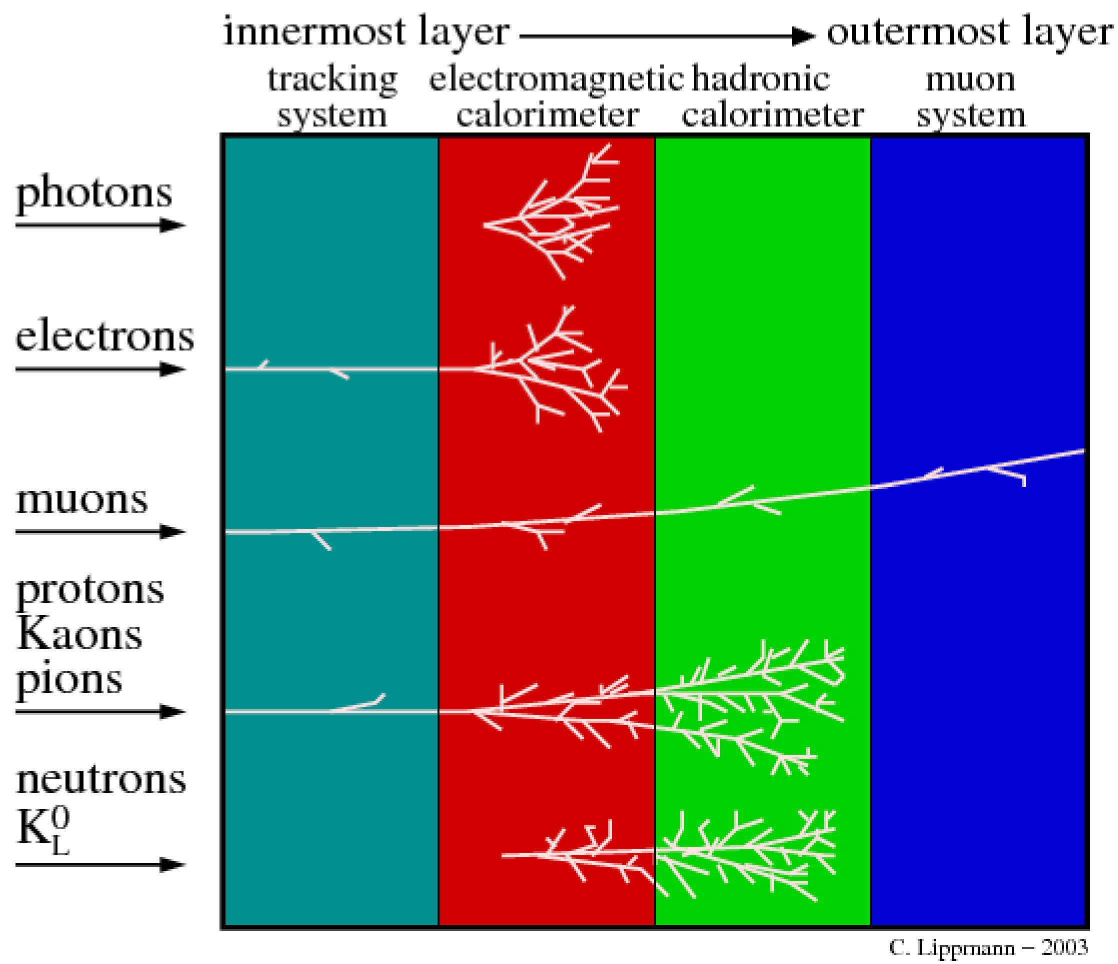


GCNN for HEP

- Main ideas
  - Being agnostic from physics
  - Let the neural network « learn / invent » the discriminating criterions
  - Capture the geometric shapes in a space-time-energy-any_other_features space
  - Get all informations from this point in space

Particle identification and regression

- Convert spacetime_energy_features geometry in classification
- Infer continuous parameters from the geometry (incident particle energy, angle...)

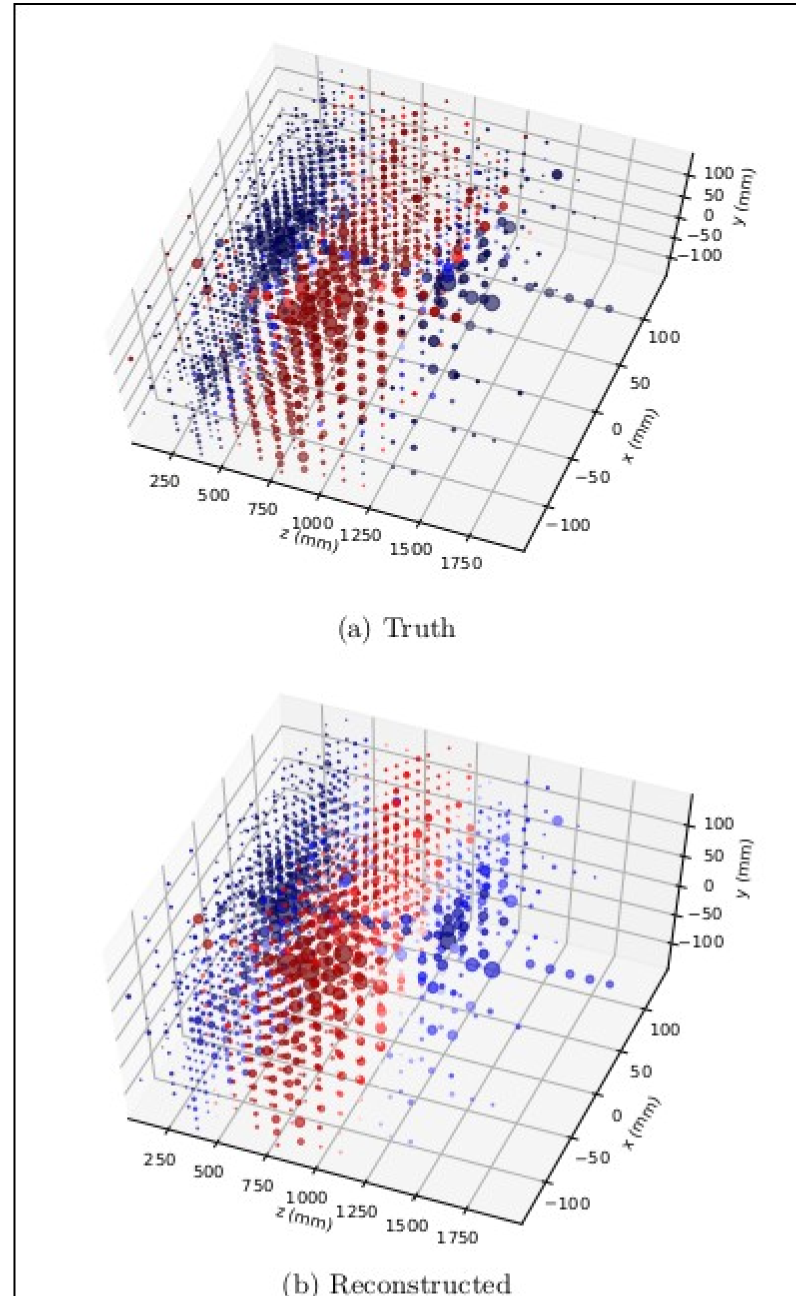


Particle Segmentation

- Prediction of the energy fraction of each sensor belonging to each shower
- Define a loss function for hit segmentation

$$L = \sum_k \frac{\sum_i \sqrt{E_i t_{ik}} (p_{ik} - t_{ik})^2}{\sum_i \sqrt{E_i t_{ik}}}$$

Qasim, Kieseler & al, Learning representations of irregular particle-detector geometry with distance-weighted graph networks, 2019

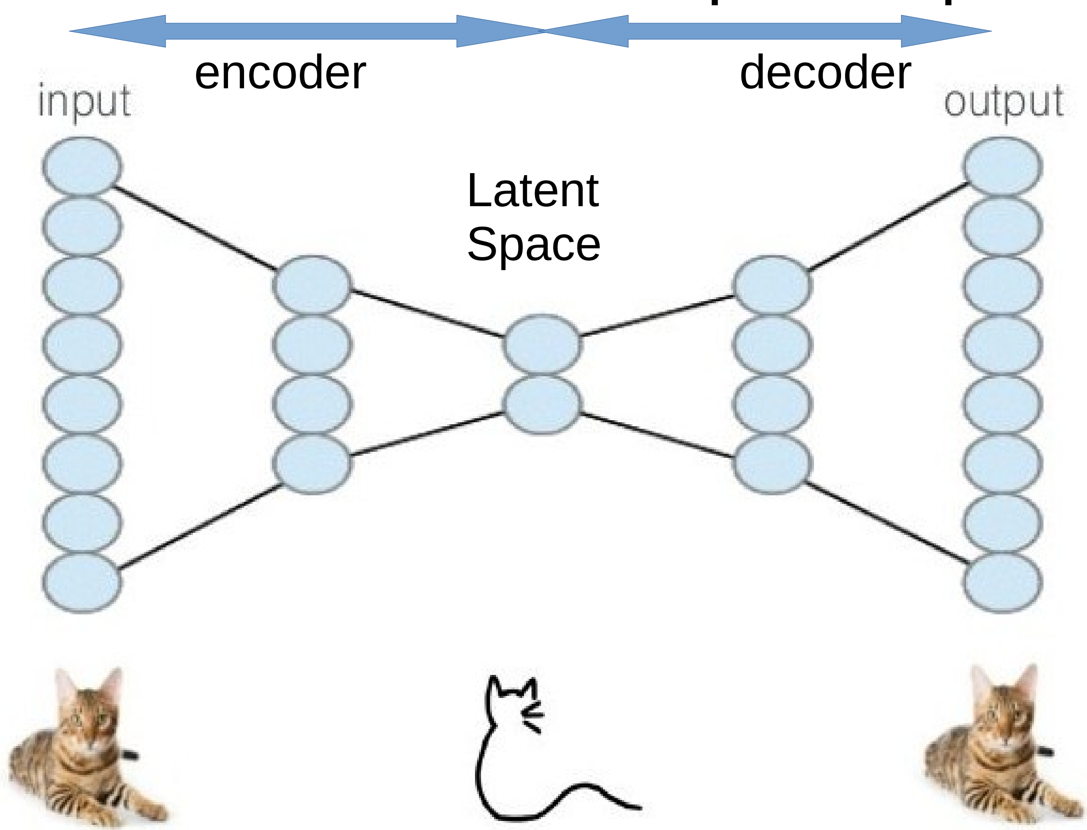


(a) Truth

(b) Reconstructed

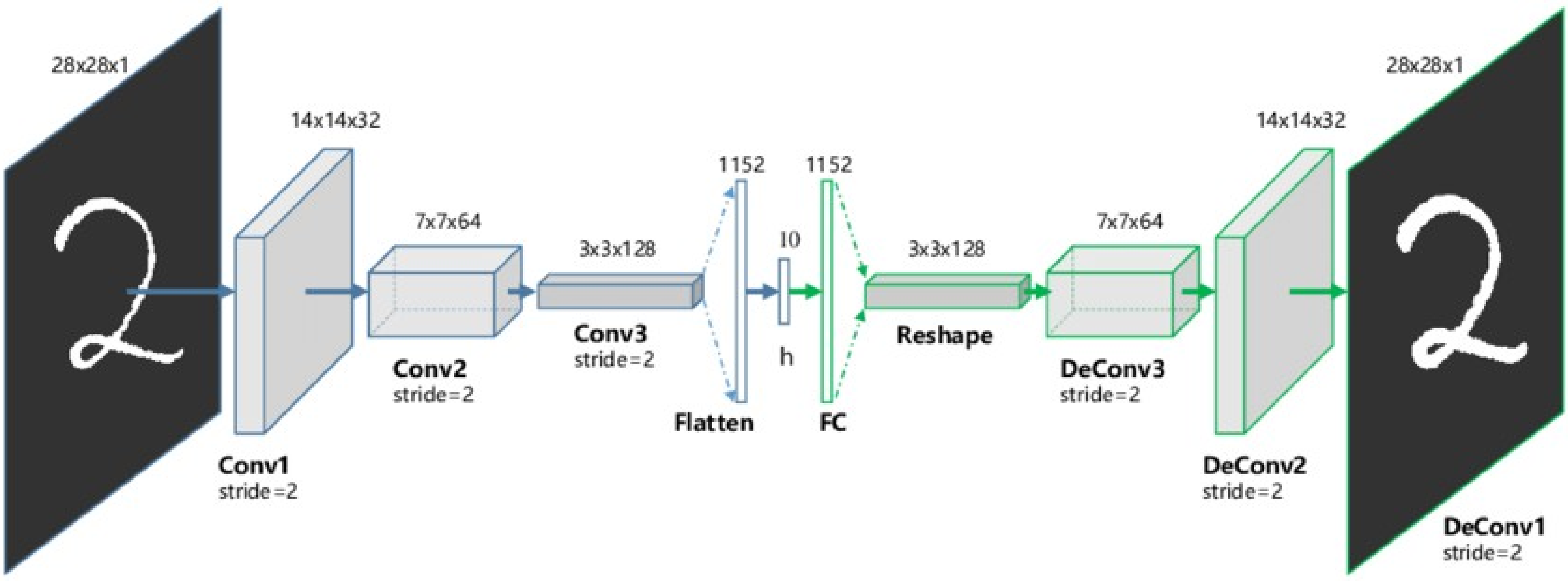
Autoencoder extension

- 3 parts
  - The encoder : input → inner representation
  - The latent space → the space were live the representations
  - The decoder : inner representation → output
- The training is done to maximize input=output



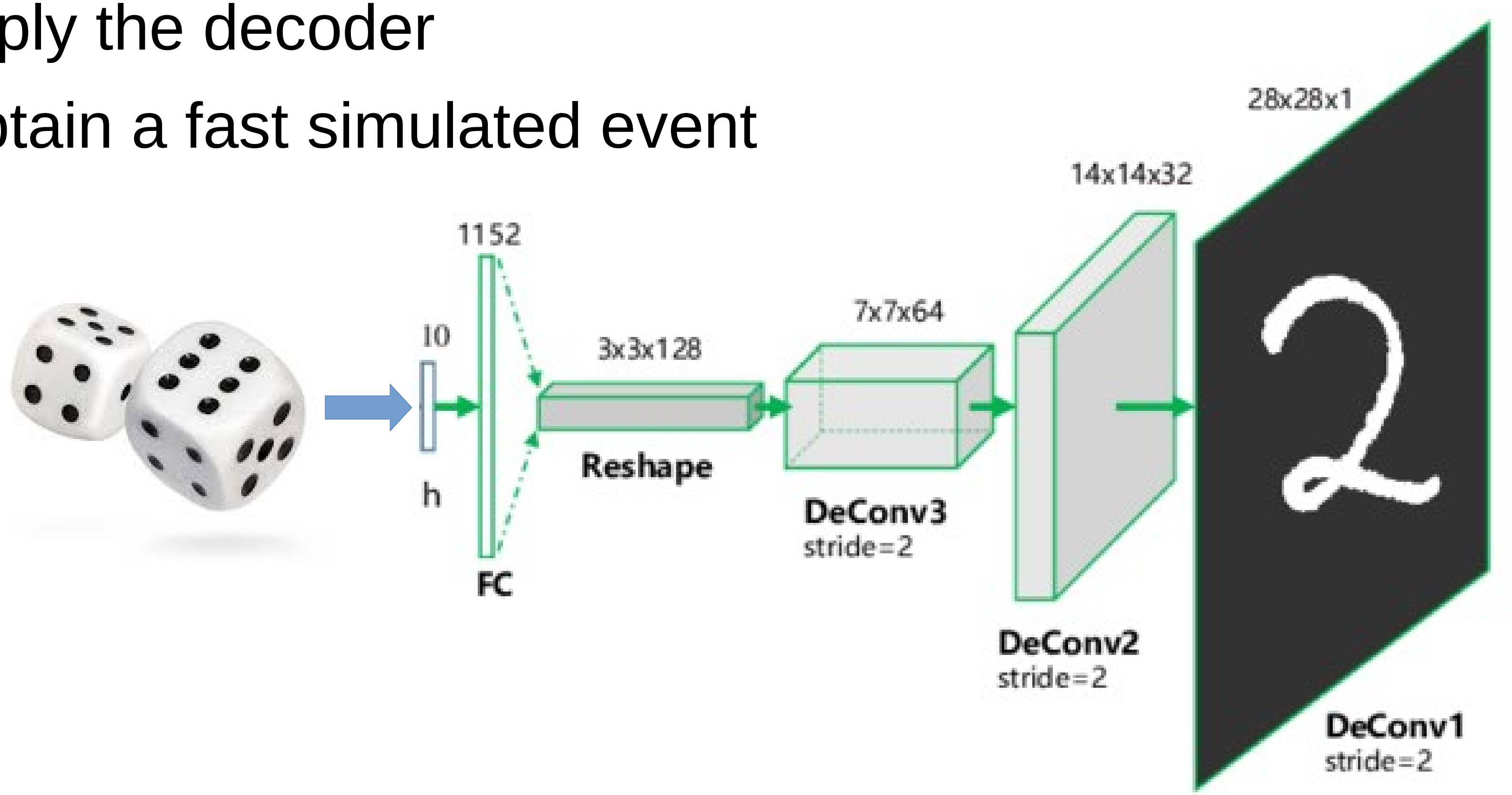
GCNN autoencoder

- Super-complicated architectures
  - variational autoencoder
  - deconvolution, un-pooling, un-readout operations to implement



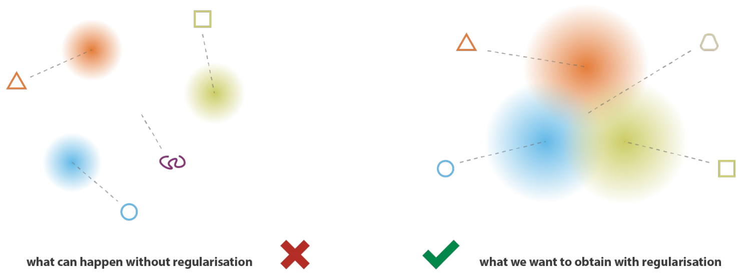
Fast Data Generation

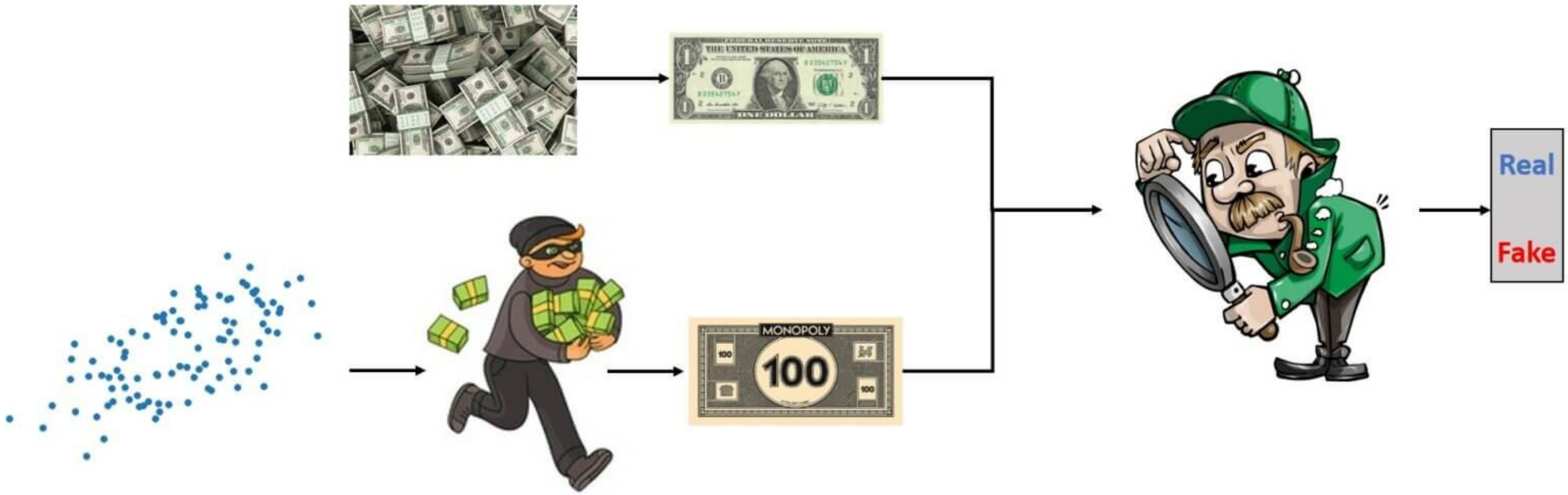
- Generate data directly from latent space
  - Generate random vector of coordinate in latent space
  - apply the decoder
  - Obtain a fast simulated event



Quality problems

- Unknown objects can appear similar to nothing in the training dataset
- Latent space completeness and compacity → variational autoencoders
- High risk of over-fitting (regularization)
- High quality can be obtained by GAN architecture



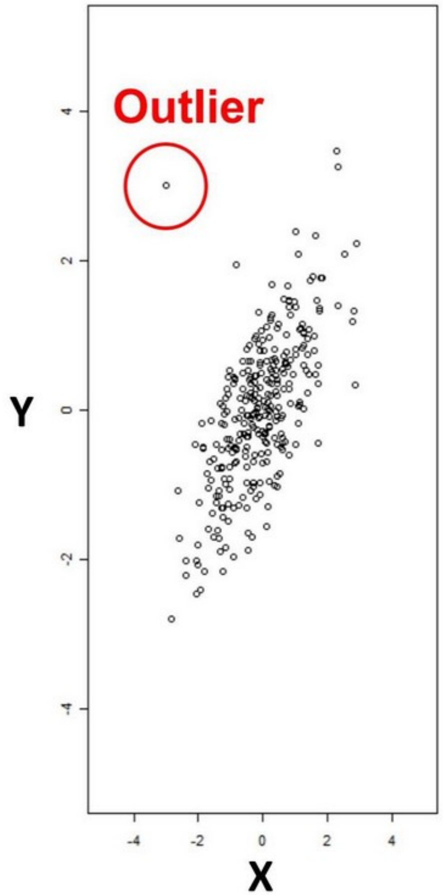


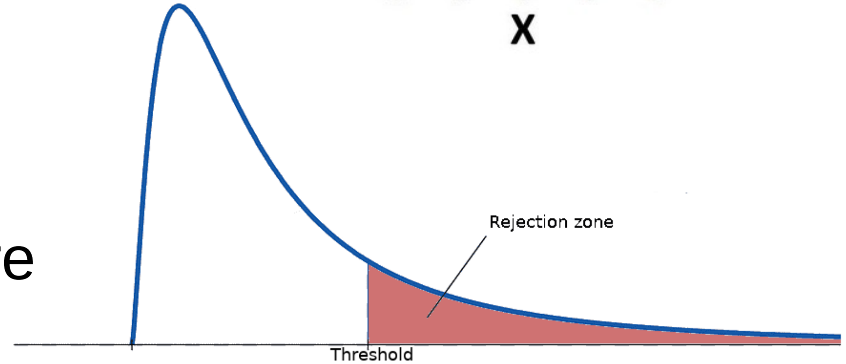
Outlier Detection

- Principle : outlier reconstruction is worse because they fit less in the generative model
- No assumption on the nature of the outlier
- Implementation
  - Calculate the quality of reconstruction in the autoencoder

$$L(x) = (x - \hat{x})^2$$

  - Compare the value to the statistic of training
  - Use a threshold to decide the outlier nature





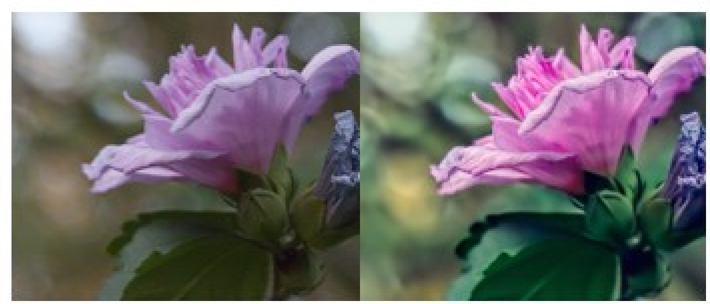
Coloring

- Autoencoders are used every day to color b&w pictures or to adjust color palette
- Could be extended to color graph nodes (tagging)
- Training procedure
  - take colored image x
  - generate b&w image y
  - use y as input and train on the loss $L(x, \hat{x})$

IMAGE COLORING

Before After

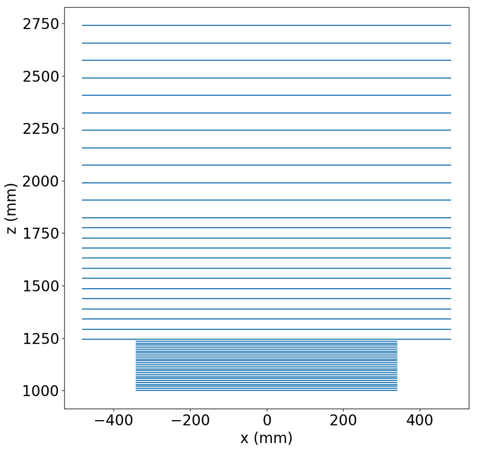
IMAGE NOISE REDUCTION

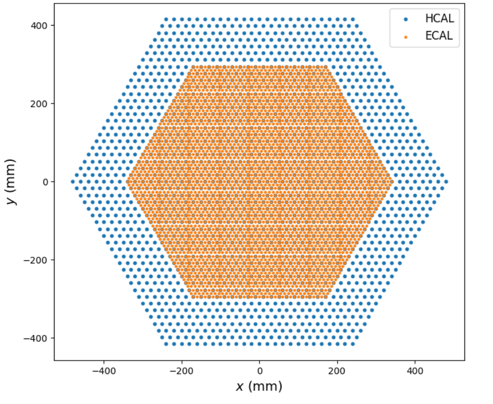


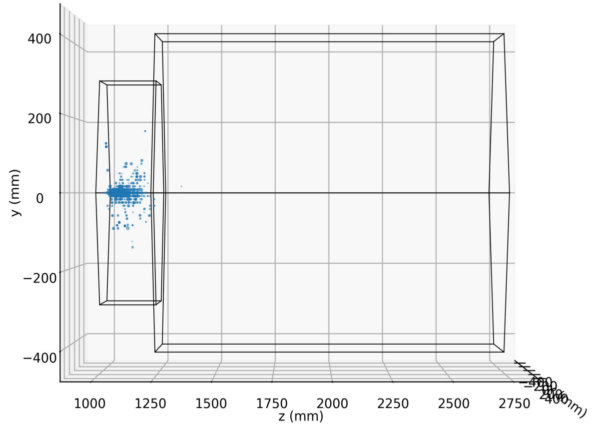
Before After

A simple example

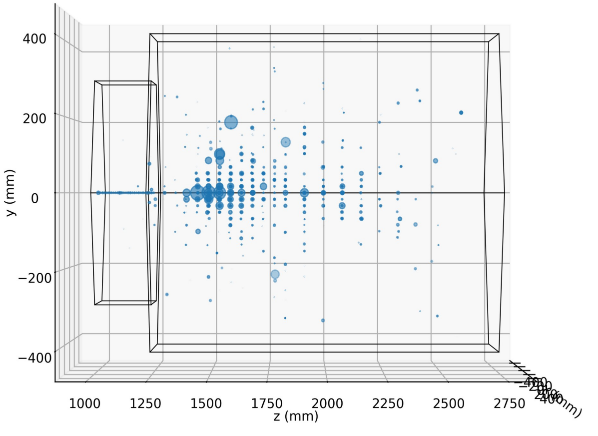
- OGCID Project
- Highly granular sampling calorimeter
- 26 ECAL + 24 HCAL layers
- Different granularity
- Regression and classification of 3 types of events



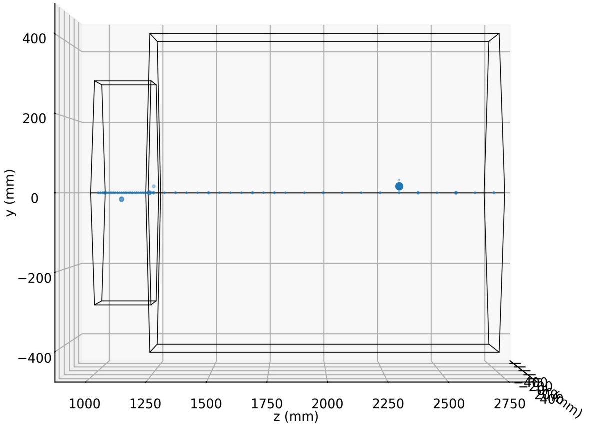




e^-/γ event

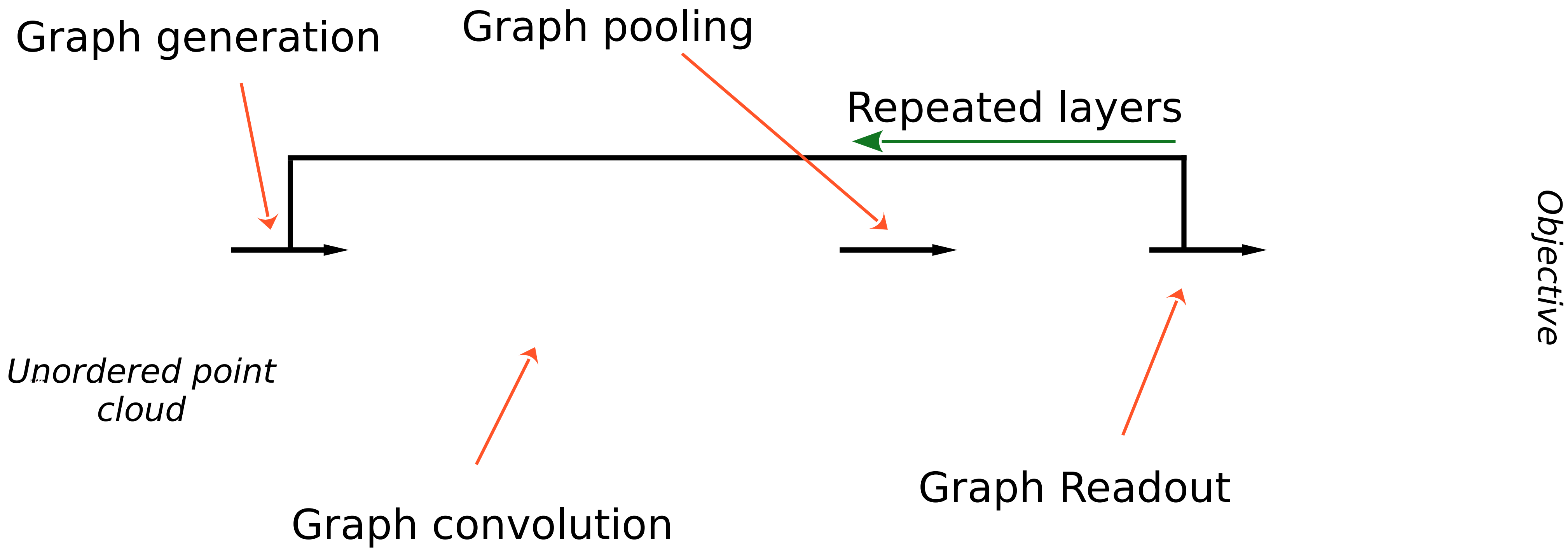


π event



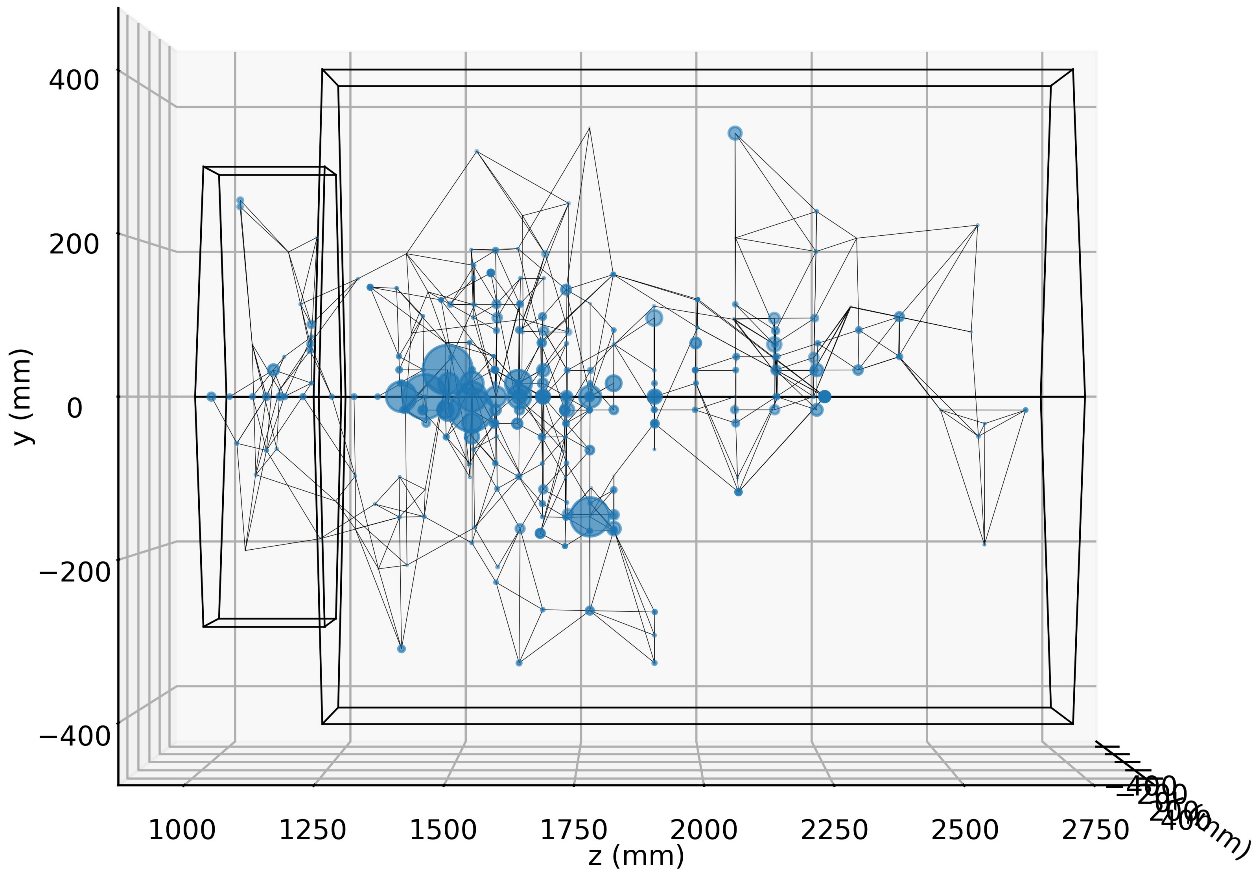
μ event

Graph Convolution Pipeline



Graph Generation

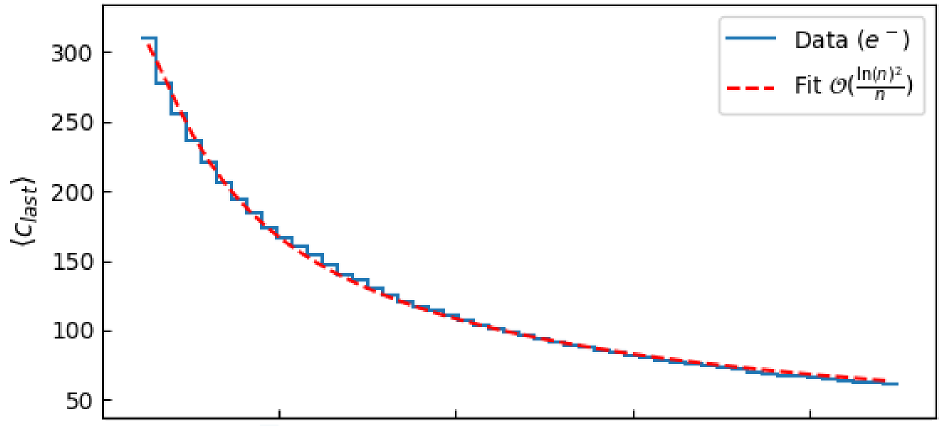
- Build arbitrary edges between sparse, multi-dimensional data-points
- Typically: k nearest neighbours (KNN)



Optimization by Proximity Tables

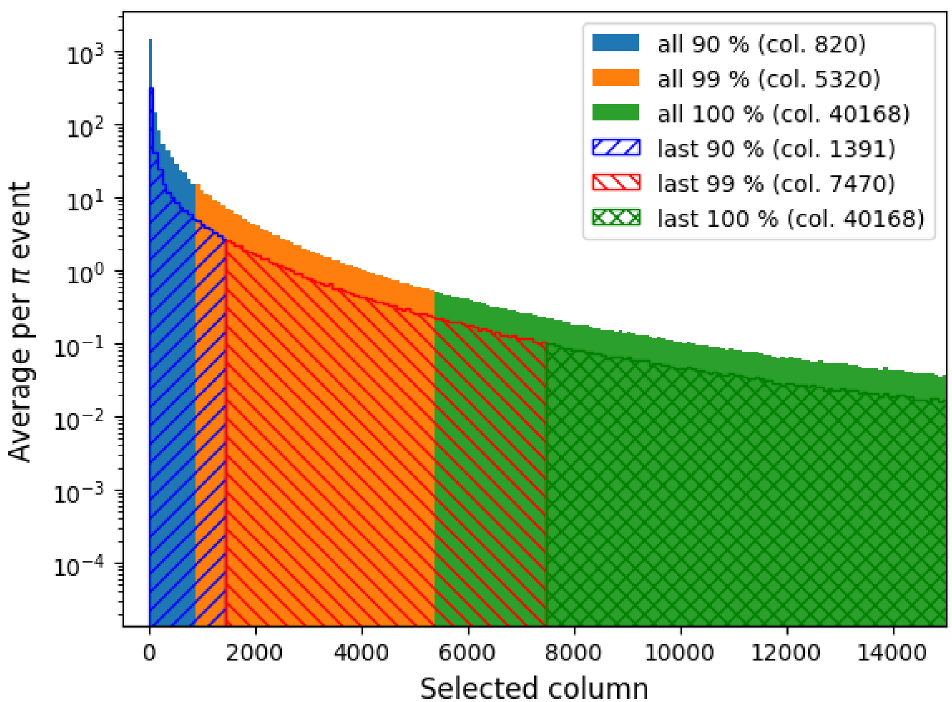
- Exploiting the static geometry of particle detectors
- For each sensor, order its neighbours by increasing distance in “proximity table” (PT)
- Reducing mean complexity from $O(n^2)$ to $O(\log^2(n))$

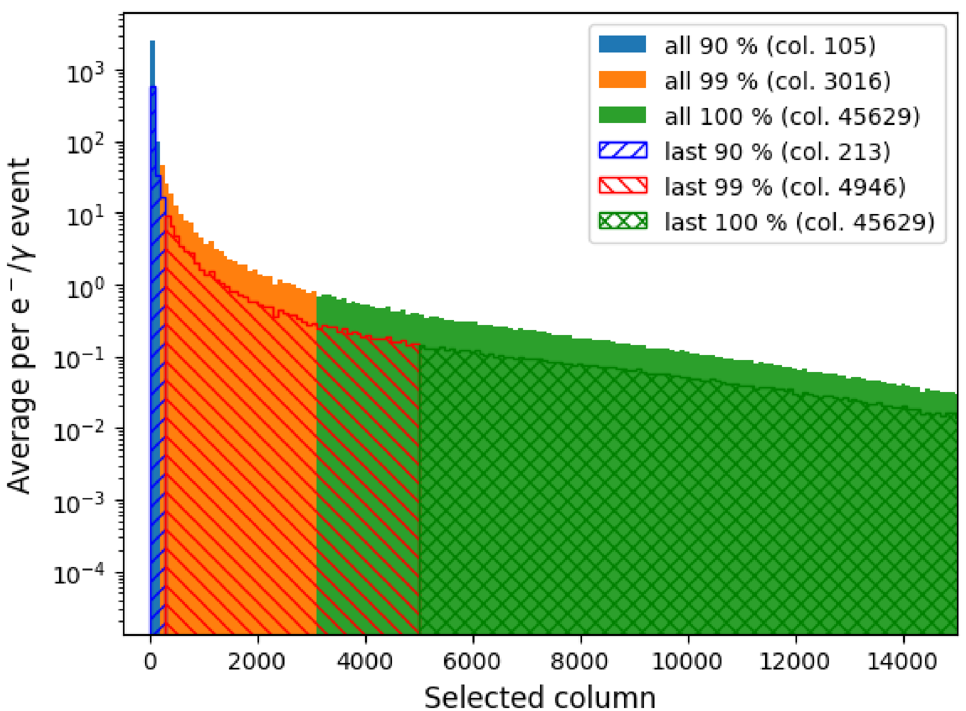
Sensor IDs	Increasing order wrt. metric			
1	17	38	...	42
2	75	16	...	68
...				
99	3	98	...	22



Reducing PTs

- Can cut PT to remove rarely explored columns
- Allows FPGA implementation
- Study of effect on performance in progress





Message Passing Convolution

- Message function Φ : Linear combination, increases the number of features by a factor 2
- Aggregator □ : feature-wise pooling (classification: max, regression: mean)
- Update function γ : Self-loop (i.e. aggregate with message from itself)

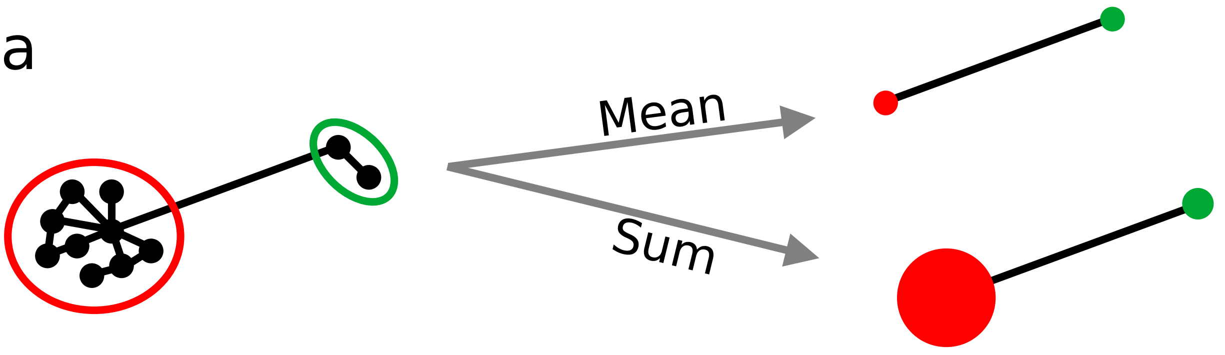
$$x_v^{(t+1)} = \mathop{\square}_{w \in \tilde{\mathcal{N}}(v)} \text{Leaky-ReLU}\left(\phi\left[x_v^{(t)}\ x_w^{(t)}\ d(v,w)\right]\right)$$

Pooling

- Pooling [Grattarola2024]:

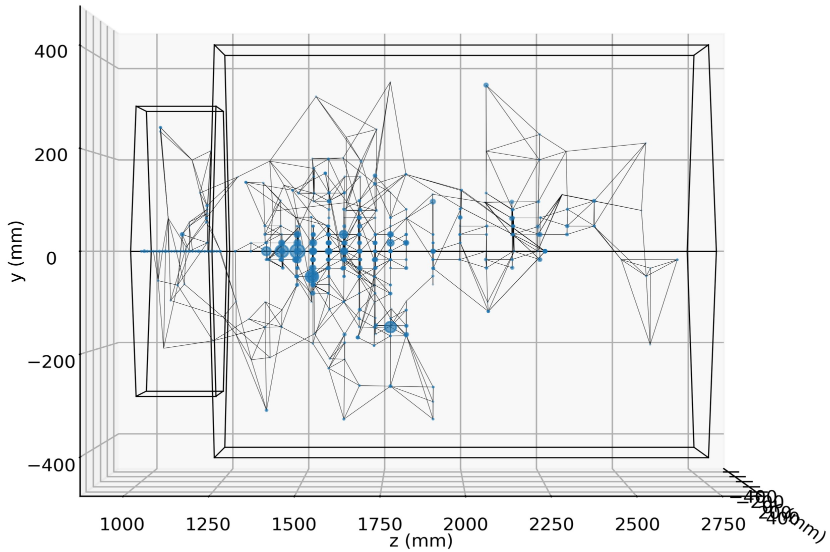
i. Selection : which edges to collapse
ii. Reduction: feature combination
iii. Connection: adjacency update

- Dedicated selection algorithm : Treclus
- Collapse all edges with a distance inferior to an adjustable threshold

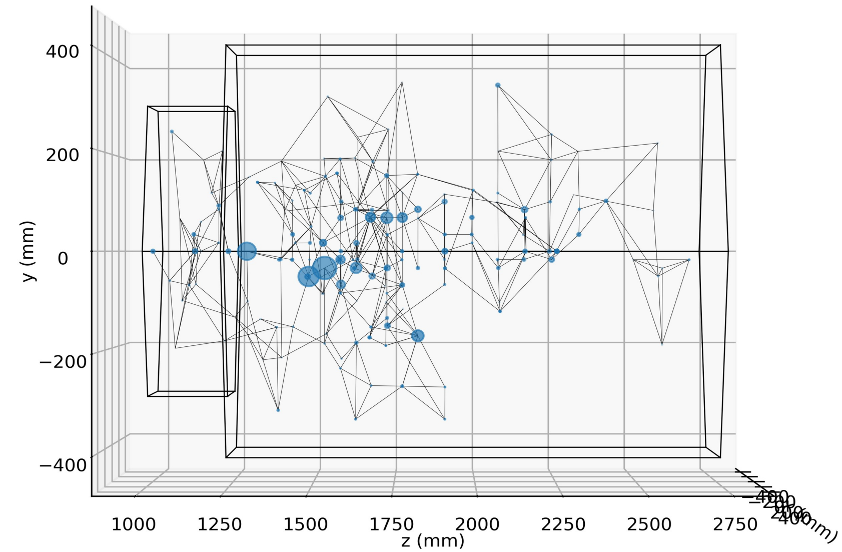


- Reduction
  - choosing randomly a destination node from the cluster
  - using max for classification and sum for regression

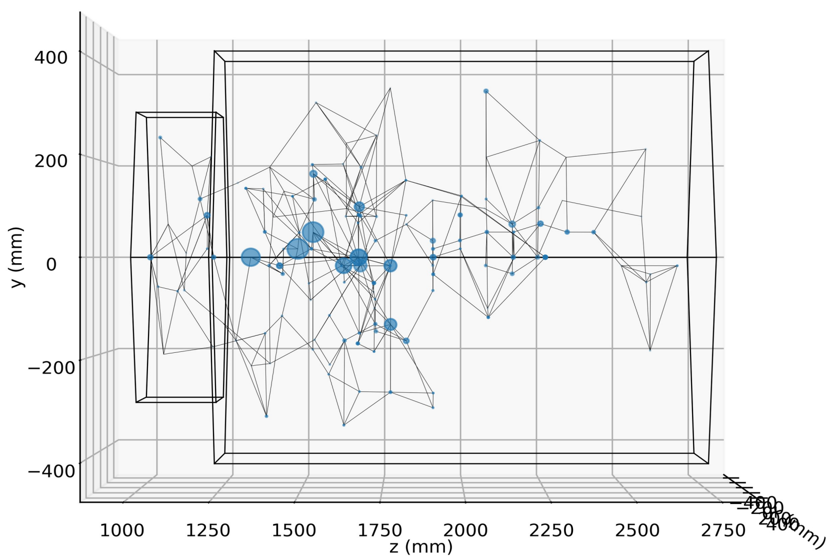
Example of pooling



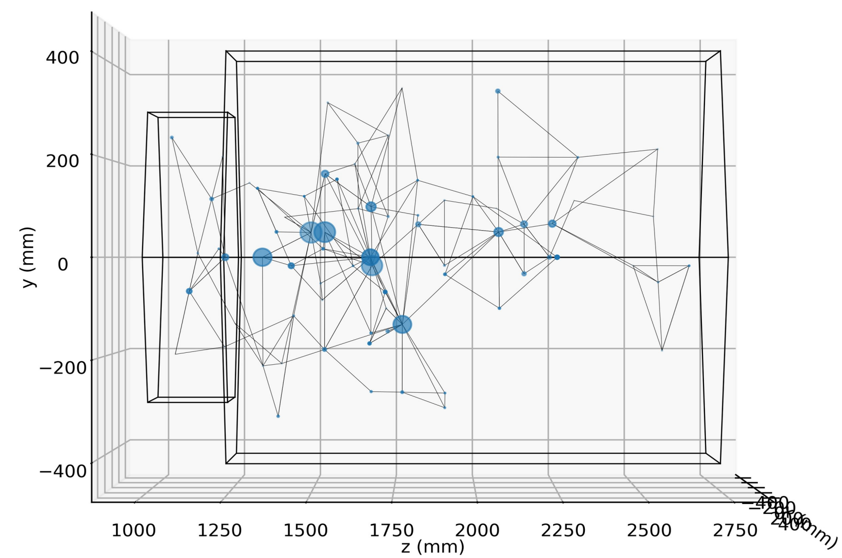
Original Graph



Pooling Step 1



Pooling Step 2

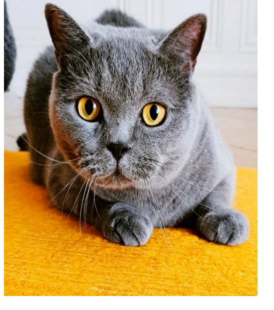


Pooling Step 3

Readout problematics

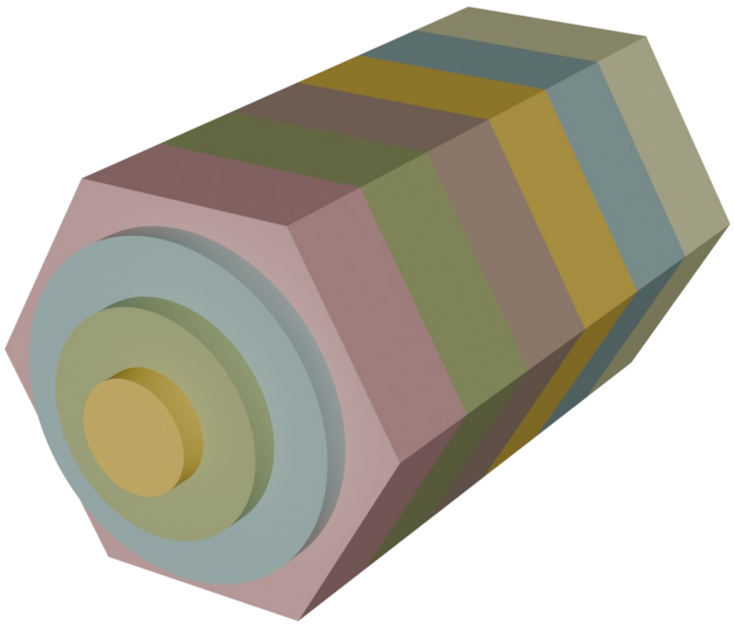
- Need to flatten graph structure as input for an MLP
- Can be tricky to keep graph structural information
  - No order for nodes
  - No order for edges
- Need a consistent approach

Random order of readout unintelligible →



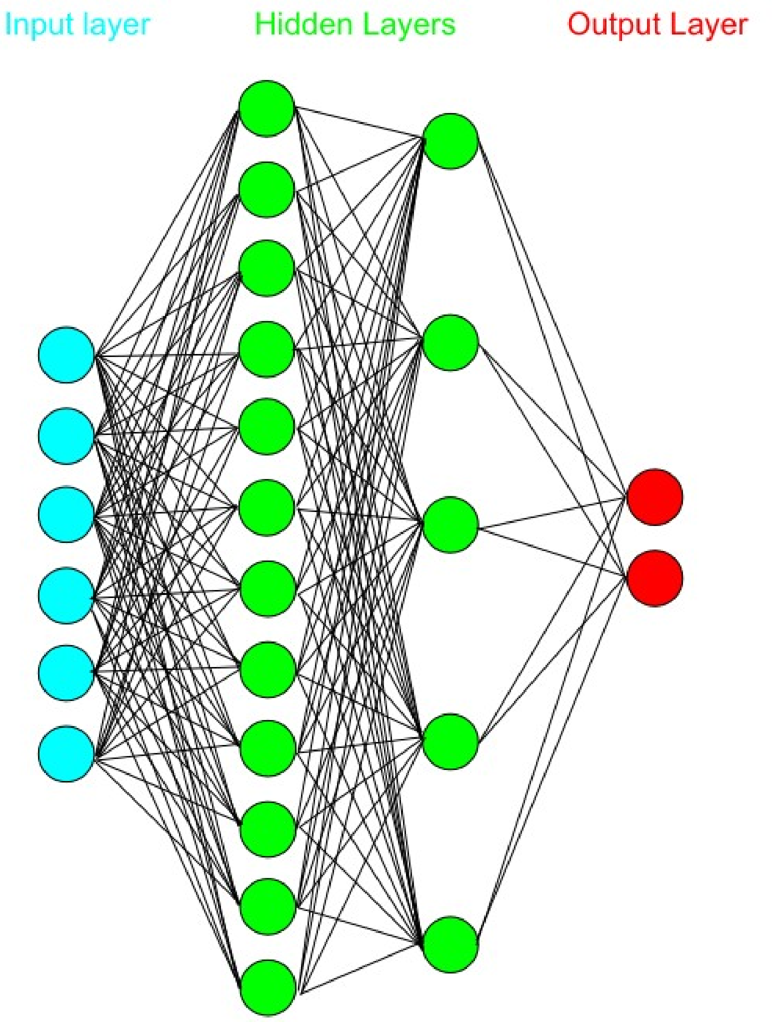
Symmetry Aware Readout

- Known geometry: embed graph back into its geometry
- Detector sliced up in readout regions that respects rotational symmetry
- Pool features within same regions (max or sum)
- Flatten in consistent order

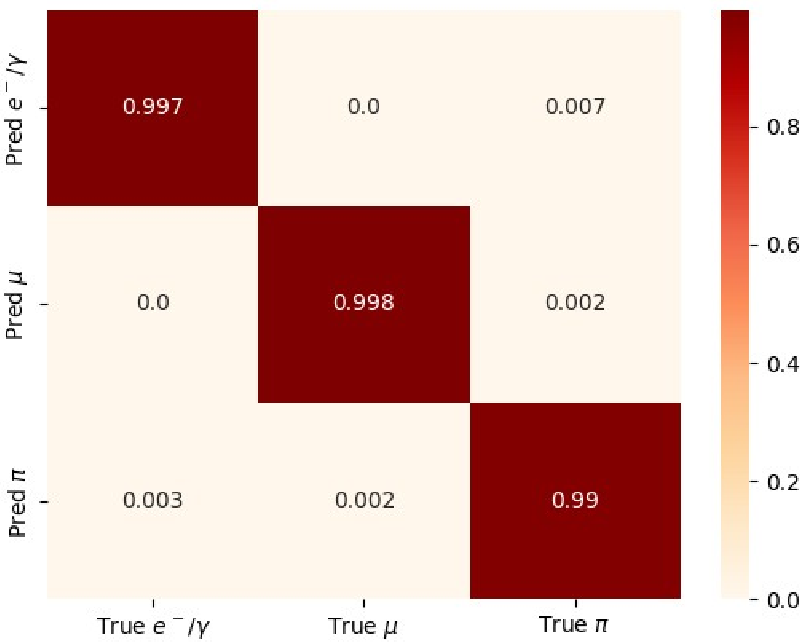


Multi-Layer Perceptron

- Fully connected MLP
- 5-6 hidden layers
- Leaky ReLU activation
- Output size:
  - 3 (PID classification)
  - or 1 (Energy regression)

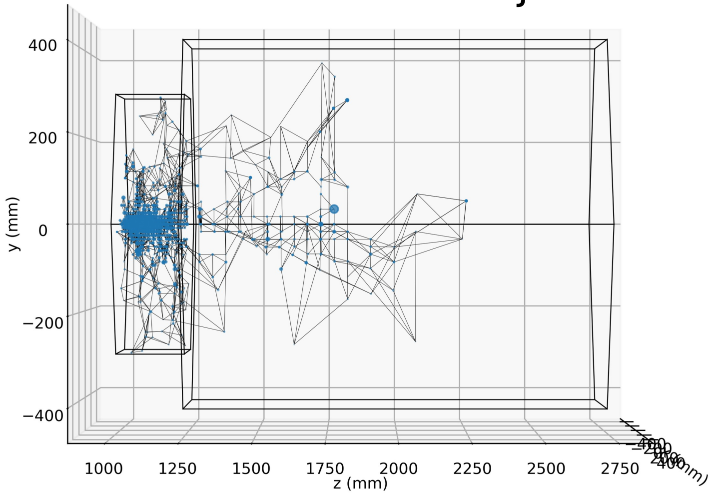


Particle ID performance

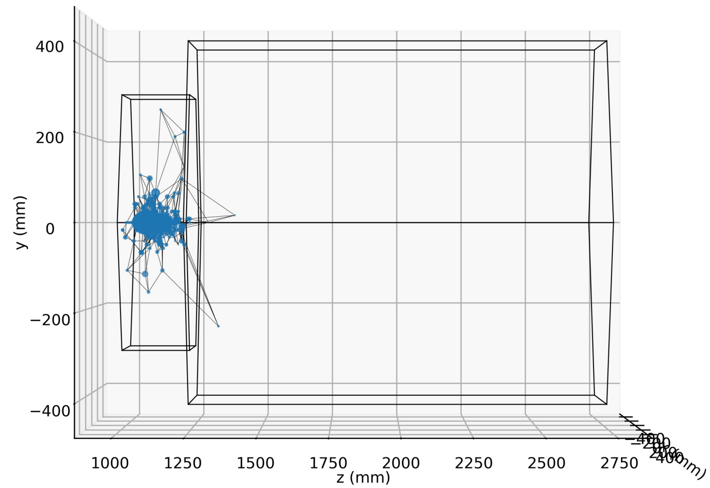


- Simulate particle showers at variable energy (10-100GeV)
- Classify e^-/γ, μ, π
- State of the art performance
- Difficult PID tasks, need more samples

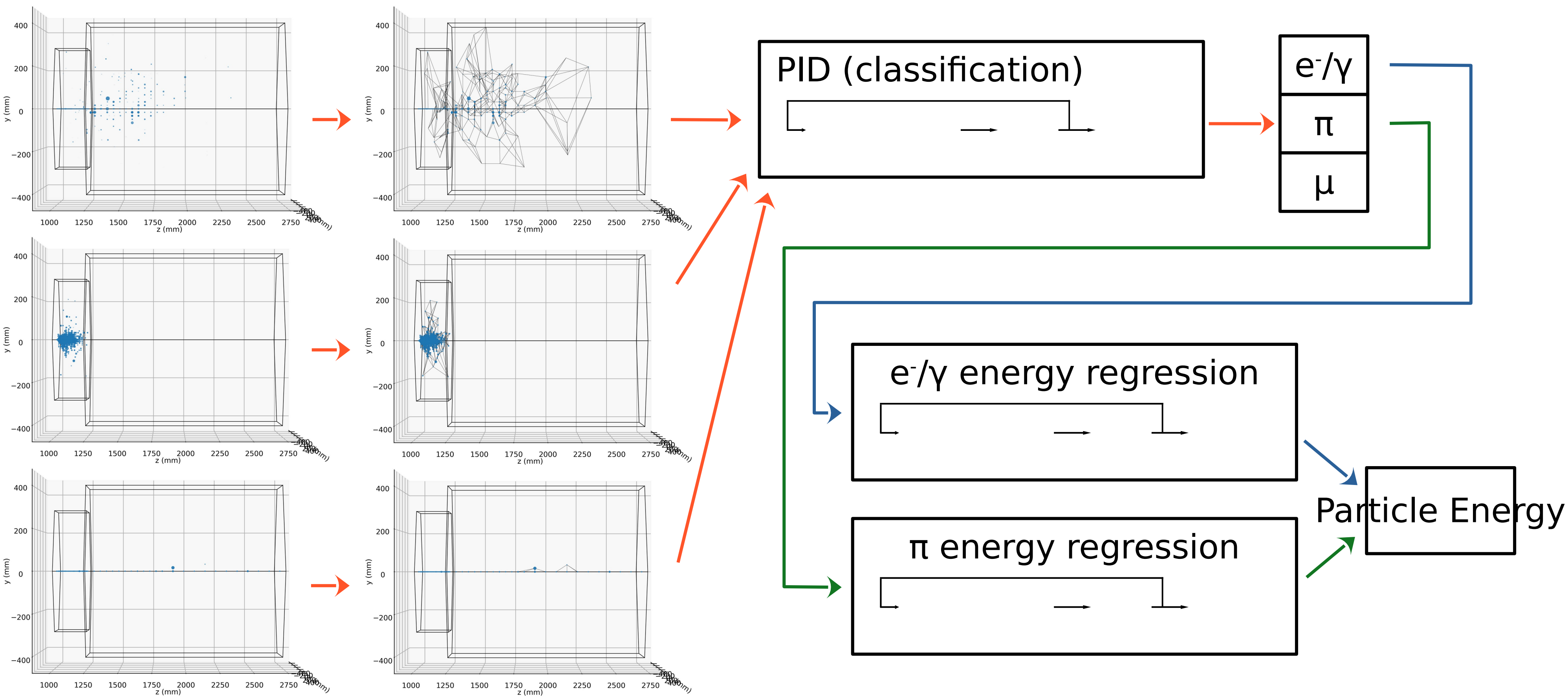
e$^-$ induced hadronic jet



Early showering π

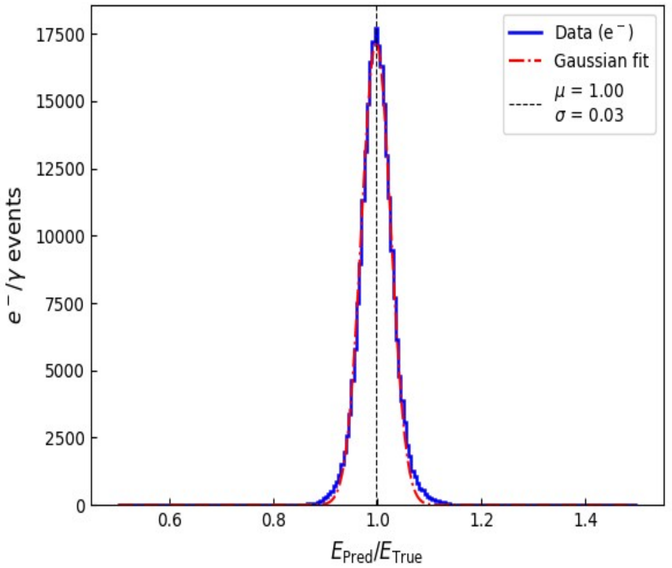


Energy Regression Pipeline

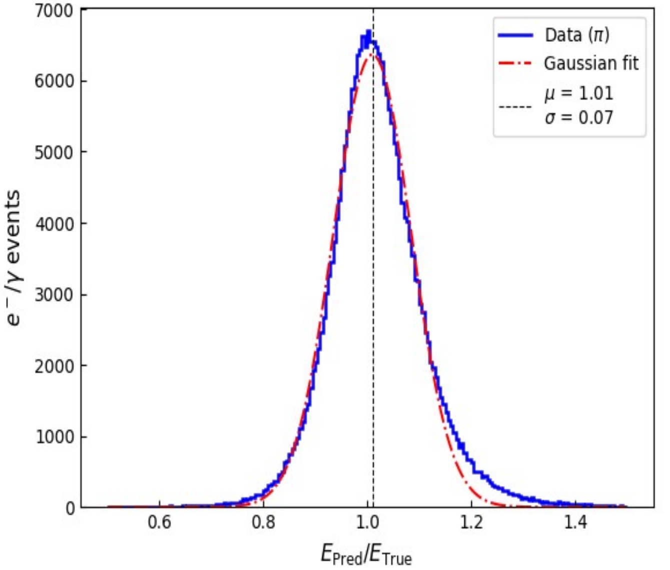


Energy Regression Performance

- Regression precision conform to detector
- e^-/γ better precision than π: different sampling fractions and physics
- Asymmetry of tails: detector properties



e^-/γ energy regression



π energy regression

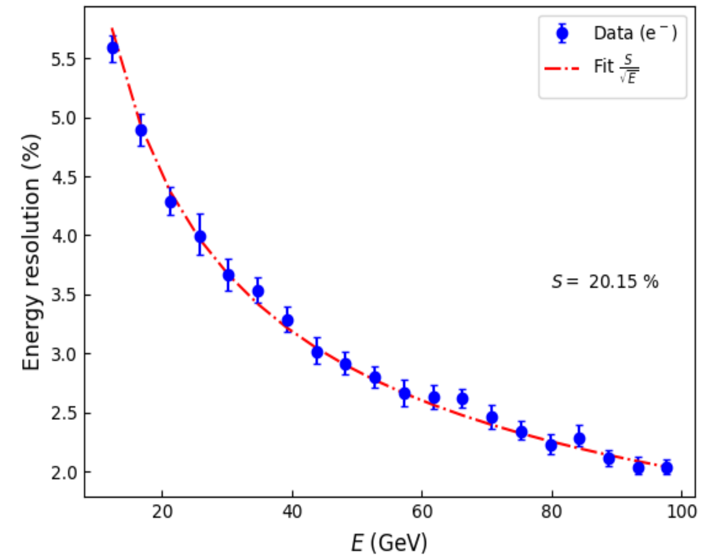
Energy resolution

Stochastic fluctuations in shower development

Noise from readout electronics

$$\frac{\sigma\left(\frac{E_{\mathrm{Pred}}}{E_{\mathrm{True}}}\right)}{\left\langle\frac{E_{\mathrm{Pred}}}{E_{\mathrm{True}}}\right\rangle} = \frac{S}{\sqrt{E_{\mathrm{True}}}} \oplus \frac{N}{E_{\mathrm{True}}} \oplus C$$

Systematic noise (e.g. dark noise...)



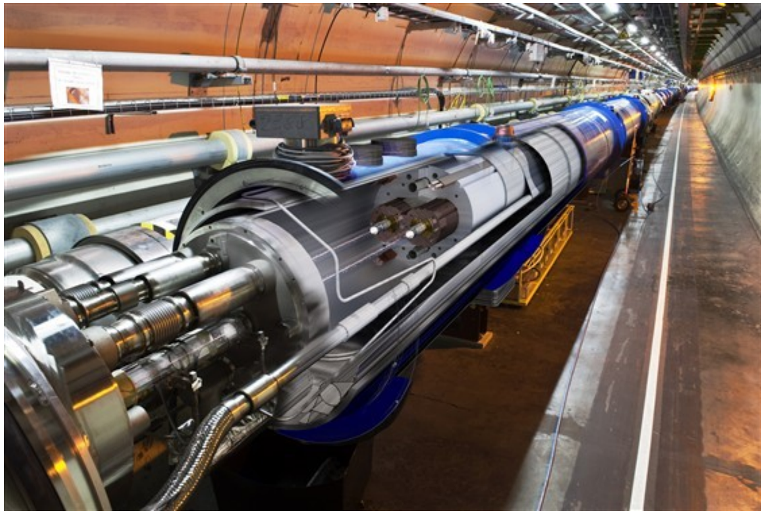
$$\frac{\sigma}{\mu} \propto \frac{1}{\sqrt{E}}$$

- Obtained S value : 20.15 %
- Theoretical prediction : between 19 and 24 %
- Testbeam results 21 % (different detector)

Conclusion

- Graph convolutional neural networks can handle HEP data
- State of the art performance
- Algorithmical optimization increases implementability (high throughput systems, FPGA…)
- Numerous extensions

Perspectives



- More complicated PID : jets !
- Segmentation
- Parallelization
- FPGA implementation
- Fast simulation
- Outlier detection (DQM, trigger)
- Autoencoder tagging