Optimised Graph Convolution for calorimetry event classification

Confrence on Computing in High Energy and Nuclear Physics

Matthieu Melennec[1]

Frédéric Magniette[1]   Shamik Ghosh[1]

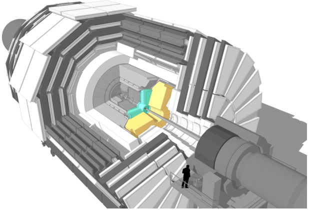
[1]Laboratoire Leprince-Ringuet, Ecole Polytechnique, Institut Polytechnique de Paris, CNRS Nucléaire & Particules, Palaiseau, France

22nd of October, 2024

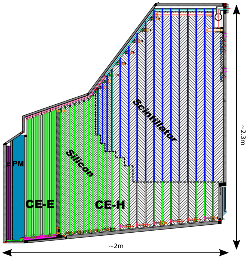
HL-LHC and CMS HGCAL

High-Luminosity LHC (HL-LHC):

- More rare events (Higgs production and BSM physics)
- Increased reconstruction complexity (up to 200 Pile-up events)



- CMS High Granularity Calorimeter (HGCAL):
  - New CMS end-cap sampling calorimeter
  - High granularity: 6M channels on 47 layers
  - Si and Scintillator based



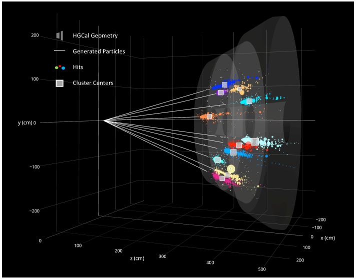
Point Cloud Data in HEP

Point cloud data:

- Points $P_i \in \mathbb{R}^{k\geq 3}$: Euclidean coordinates + $(k-3)$ "colours"
- **Unordered**, **sparse** and with **variable size**

HGCAL output: Point clouds

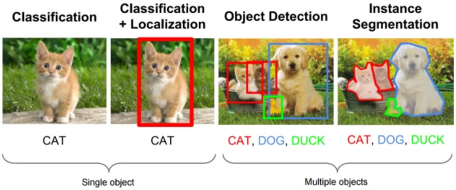
- Hits: 3D points with energy measurement and timing
- Variable granularity
- Graph convolution promising approach



CNNs and Point Cloud Data

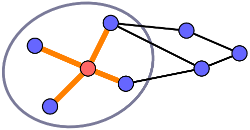
Convolutional Neural Networks:

- Excellent at classification and segmentation tasks
- Identifies geometric patterns



How to generalise the success of CNNs to point-cloud data?

- Graph convolution



Message Passing Graph Convolution

Formalism:

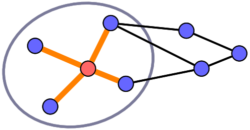
Aggregator: symmetric and normalised (e.g. mean/max) combines all messages

$$x_v^{(t+1)} = \gamma_{\theta_\gamma}\left(x_v^{(t)}, \underset{w \in \mathcal{N}(v)}{\square} \phi_{\theta_\phi}\left(x_v^{(t)}, x_w^{(t)}, e_{vw}\right)\right)$$

Update function: combine messages with own features

Message function: collects neighbour features

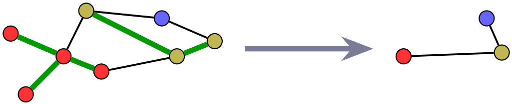
Gilmer et al., *Neural message passing for quantum chemistry*, 2017



Graph Pooling

Aim: Coarsen graph to increase the range of the convolution

1. **Selection** (or clustering): Select which nodes to pool (e.g. by selecting edges to "collapse")
2. **Reduction**: Combine features of pooled nodes (using max or sum pooling)
3. **Connection**: update adjacencies (inherited or dynamic)



Grattarola et al. *Understanding Pooling in Graph Neural Networks*, 2024

OGCID Project

Optimal Graph Convolution for particle IDentification

Efficient algorithms for event reconstruction in particle detectors

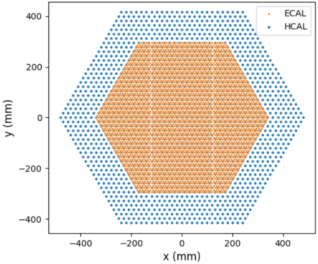
- Reduction of graph construction complexity
- Segmented implementation
- Optimising the network design and adapt it to the electronic implementation (FPGA...)
- Multi-task (Online and Offline CMS HGCAL reconstruction, Hyper-Kamiokande DSNB discrimination)

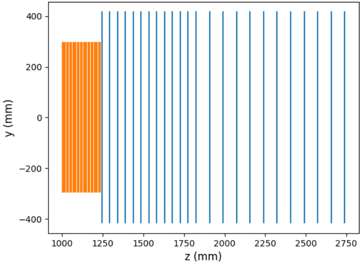
Funded by the Agence Nationale de la Recherche (ANR), ANR-21-CE31-0030

HGCAL-like simulation

Simulate HGCAL-like calorimeter using GEANT4

- $\sim 10^5$ Si sensors
- 26 ECAL layers with Pb absorbers
- 24 HCAL layers with stainless steel absorbers

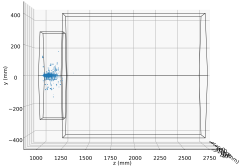




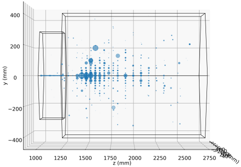
Simulation Samples

Simulated e$^-/\gamma$, π^+ and μ^- events in the detector

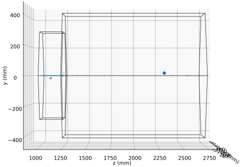
- Energies 10 GeV to 100 GeV
- Each hit corresponds to the energy deposited in the detector in the corresponding sensor



e$^-/\gamma$ event

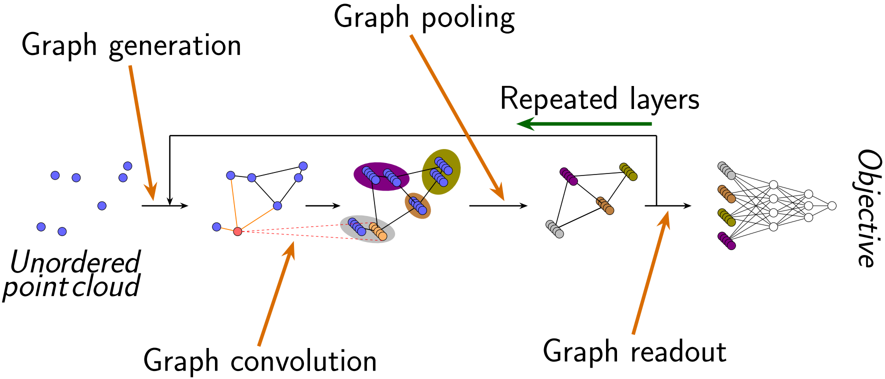


π event



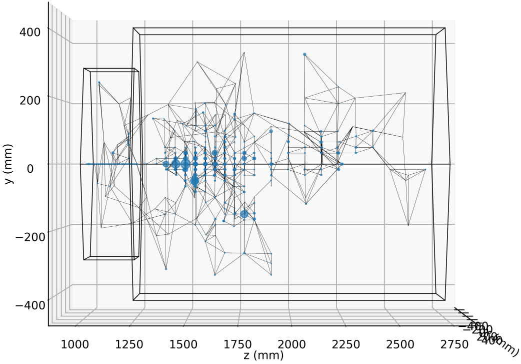
μ event

GCNN Structure



Graph Generation

- Build arbitrary edges between sparse, multi-dimensional data-points
- Typically: k nearest neighbours (KNN)
  - Ensures geometric locality
  - Complexity: worst-case = mean = $\mathcal{O}\left(n^2\right)$



Proximity Tables

Particle detectors: Static and known geometry

Pre-compute proximities of sensors

- For each sensor, order its neighbours by increasing distance in a "proximity table" (PT)

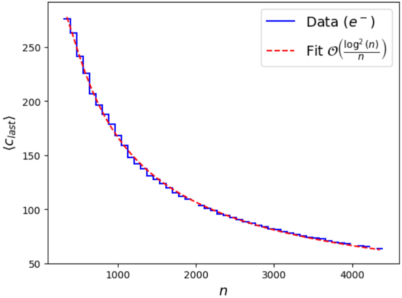
Sensor IDs	Increasing order wrt. metric →			
1	17	38	...	42
2	75	16	...	68
...				
99	3	98	...	22

- Arbitrary choice of metric used for ordering (e.g. Euclidean, adding a radiality term, correlation...), but no correlation on model performance in our study: take Euclidean distance

Proximity Table KNN

PT-KNN: iterate over rows until k neighbours found

- Worst-case complexity: $\mathcal{O}\left(n^2\right)$
- Reduces best case complexity to $\mathcal{O}(kn)$
- On average, $\langle c_{\text{last}} \rangle = \mathcal{O}\left(\frac{\log^2(n)}{n}\right)$



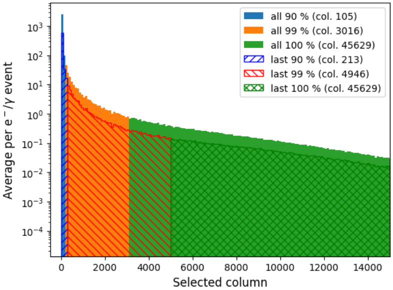
PTs reduce the mean complexity of KNN from

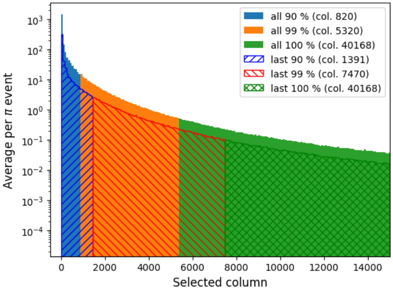
$$\mathcal{O}(n^2) \quad \text{to} \quad \mathcal{O}\left(\log^2(n)\right)$$

Reducing PTs

Proximity Tables: $10^5 \times 10^5$ entries

- Can cut PT to remove rarely explored columns
- Allows FPGA implementation



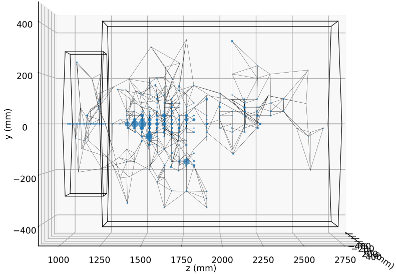


Resulting Graphs

We obtain graphs:

- **Nodes** v:
  - Sensor energy x_v
  - Position $\vec{u}_v$
- **Edges** e_{vw}:
  - End nodes v, w
  - Length $d(v,w) = \|\vec{u}_v - \vec{u}_w\|$
- **Graph-level features** (pid, energy...)

Radial symmetry in detector $\Rightarrow$ Positions $\vec{u}_v$ carried as "hidden features", not used in convolution



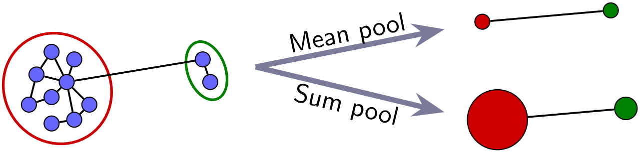
Message Passing Convolution

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

- Message function Φ_θ: Linear combination with trainable weights θ
  $\Phi_\theta \in \mathbb{R}^{2n\times(2n+1)}$, i.e. doubles number of features
- Aggregator $\square$: Feature-wise pooling (classification: max, regression: mean)
- Update function γ: Self-loop (i.e. aggregate with message from itself)

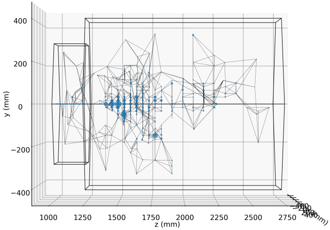
Pooling

1. **Selection** with **Treclus**: Collapse all edges shorter than a threshold ε
   - Choice of ε using the number of resulting nodes: Convolution doubles n° features $\Rightarrow$ pooling halves n° nodes
2. **Reduction**: Combine nodes v in cluster $\mathcal{C}$
   - Feature-wise pool $\{x_v\}_{v \in \mathcal{C}}$ (classification: max, regression: sum)
   - Choose at random a destination node in $\{\vec{u}_v\}_{v \in \mathcal{C}}$

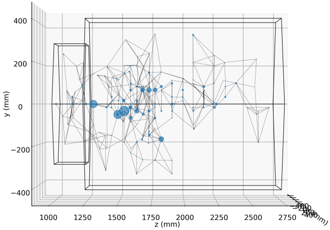


3. **Connection**: Inherited adjacency from nodes $v \in \mathcal{C}, w \in \mathcal{C}'$ neighbours $\Rightarrow \mathcal{C}, \mathcal{C}'$ neighbours

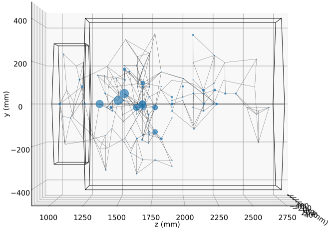
Example Pooling



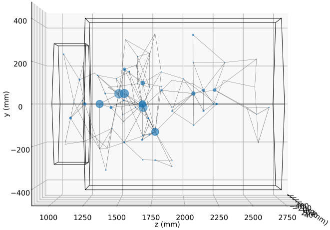
Original graph



Pooling step 1



Pooling step 2



Pooling step 3

Readout

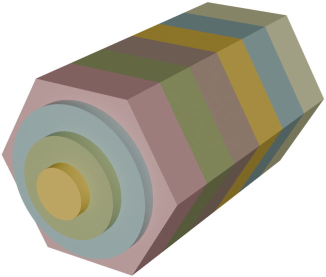
Readout problematic:

- 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 →

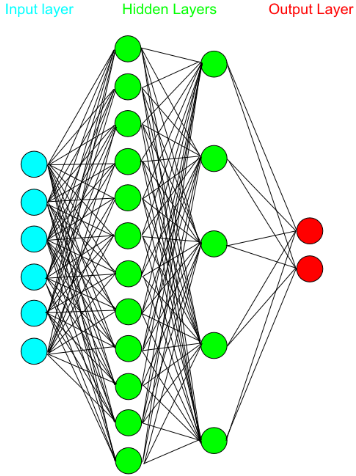
Readout

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

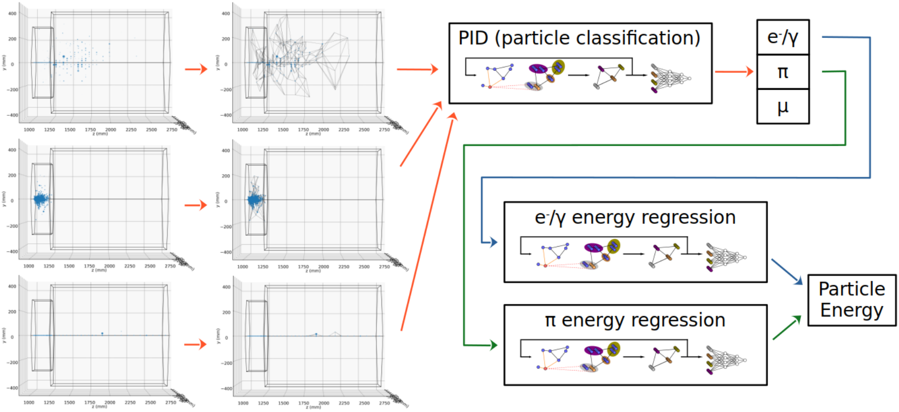


Multi-Layer Perceptron

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

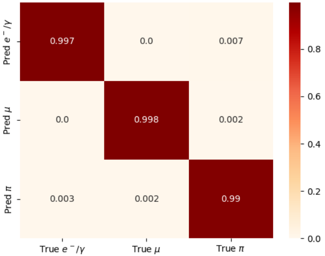


Full Pipeline



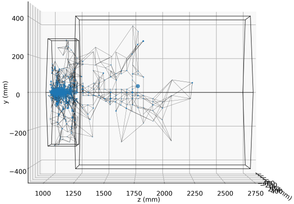
Pipelines have 3 CP layers, 6 hidden MLP layers $\sim 10^4$ parameters
Readout granularity adapted to task

Particle ID Performance

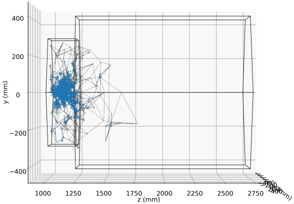


- Classify $e^-/\gamma, \mu, \pi$ with $E \in [10, 100]$ GeV
- Balanced set of 10^5 events
- State of the art performance
- Some difficult PID tasks

e^- induced Hadronic jet

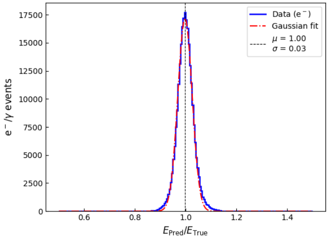


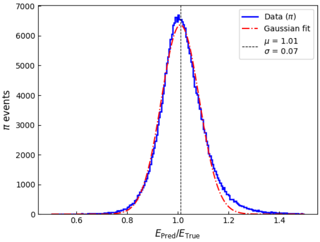
Early showering π



Energy Regression

- Trained on 2×10^6 graphs (75% training)
- Regression performance conform to detector
- e^-/γ better precision that π: different sampling fractions and physics
- Asymmetry of tails: detector properties





Energy Resolution

Energy resolution given by:

Stochastic fluctuations in shower development

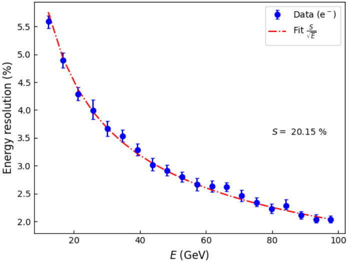
Noise from read-out electronics

$$\left(\frac{\sigma\left(\frac{E_{\text{Pred}}}{E_{\text{True}}}\right)}{\left\langle\frac{E_{\text{Pred}}}{E_{\text{True}}}\right\rangle}\right)^2 = \frac{S}{\sqrt{E_{\text{True}}}} + \frac{N}{E_{\text{True}}} + C$$

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

- Noise not emulated:

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



Conclusion

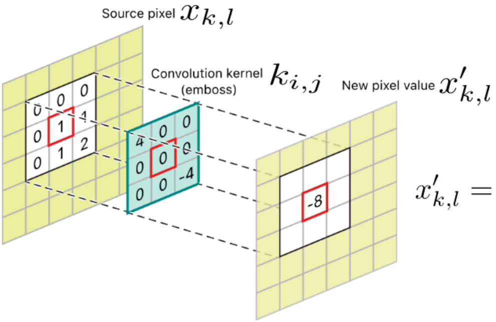
- Graph convolution powerful tool for HEP data
- Recover state of the art results
- Algorithmic optimisation allows online implementation (e.g. FPGAs)

Perspectives:

- More difficult PIDs
- Bigger energy range
- Pile-up Segmentation
- Extension to other detectors (e.g. diffuse supernovae background in Hyper-Kamiokande)

Thank you for listening...
Any questions?

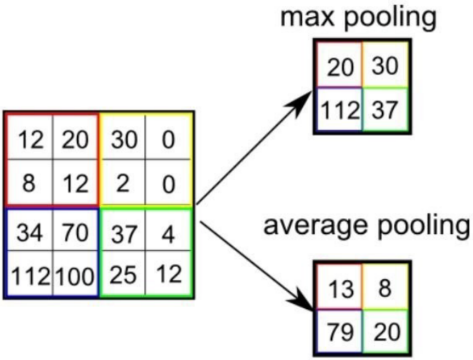
Backups



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

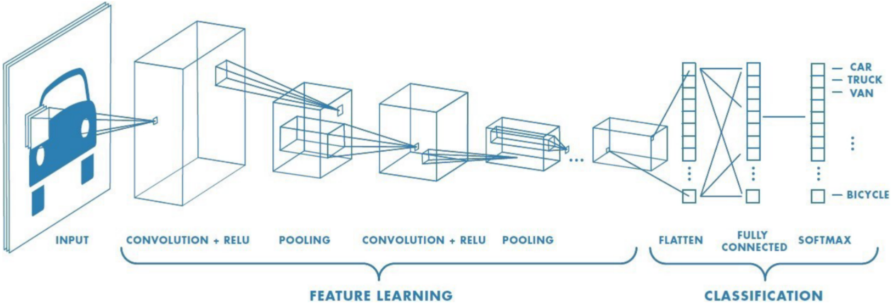
- Apply kernel on image (like the convolution filter)
- Kernel (k_{ij}) is learnable
- Filter is shared over the whole picture
- Idea : creating maps of features (one kernel per feature)

Image Pooling



- Reduce the dimensionality of the feature maps
- Move to higher level of abstraction
- Classification: max pool; Regression: mean pool

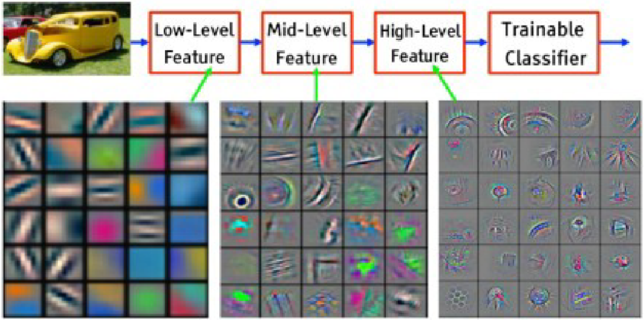
Convolutional Network



Network structure :

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

How Does It Work?



- 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

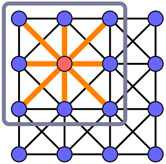
Why Does It Work?

- The two operations derive naturally from local space:
  - Euclidean space $\Rightarrow$ Translation invariance; Respected by convolution
  - Scale-separability $\Rightarrow$ alternated convolution and downsampling
- Dream complexity
  - $\mathcal{O}(1)$ parameters par filter (independent of image size)
  - $\mathcal{O}(n)$ complexity in time per layer (n pixels)

- Message: $\phi_{\theta_\phi}(x_v, x_w, e_{vw}) = x_w * \theta_{\phi_w}$
- Aggregator: $\square = \Sigma$
- Every node is self-looped:

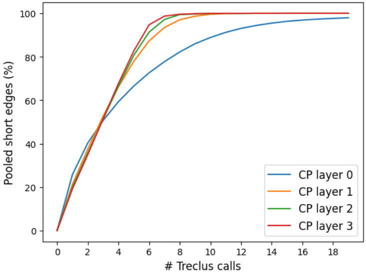
$$x_v^{(t+1)} = \sum_{w \in \tilde{\mathcal{N}}(v)} x_w * \theta_{\phi_w}$$

with $\tilde{\mathcal{N}}(v)$ a regular structure containing $\mathcal{N}(v)$ and v



Pooling

1. **Selection** with **Treclus**: Collapse all edges shorter than a threshold ε
   - Choice of ε using the number of resulting nodes: Convolution doubles n° features $\Rightarrow$ pooling halves n° nodes
   - Make a random matching: avoid chain clusters
   - Treclus cannot collapse all short edges: Call multiple times



Chain clusters