The LHCb Calorimeter reconstruction challenge

Núria Valls Canudas
Smart Society Research Group - La Salle URL Barcelona
On behalf of the LHCb Real Time Analysis Project

3rd COMCHA School
23 – 31 October 2023

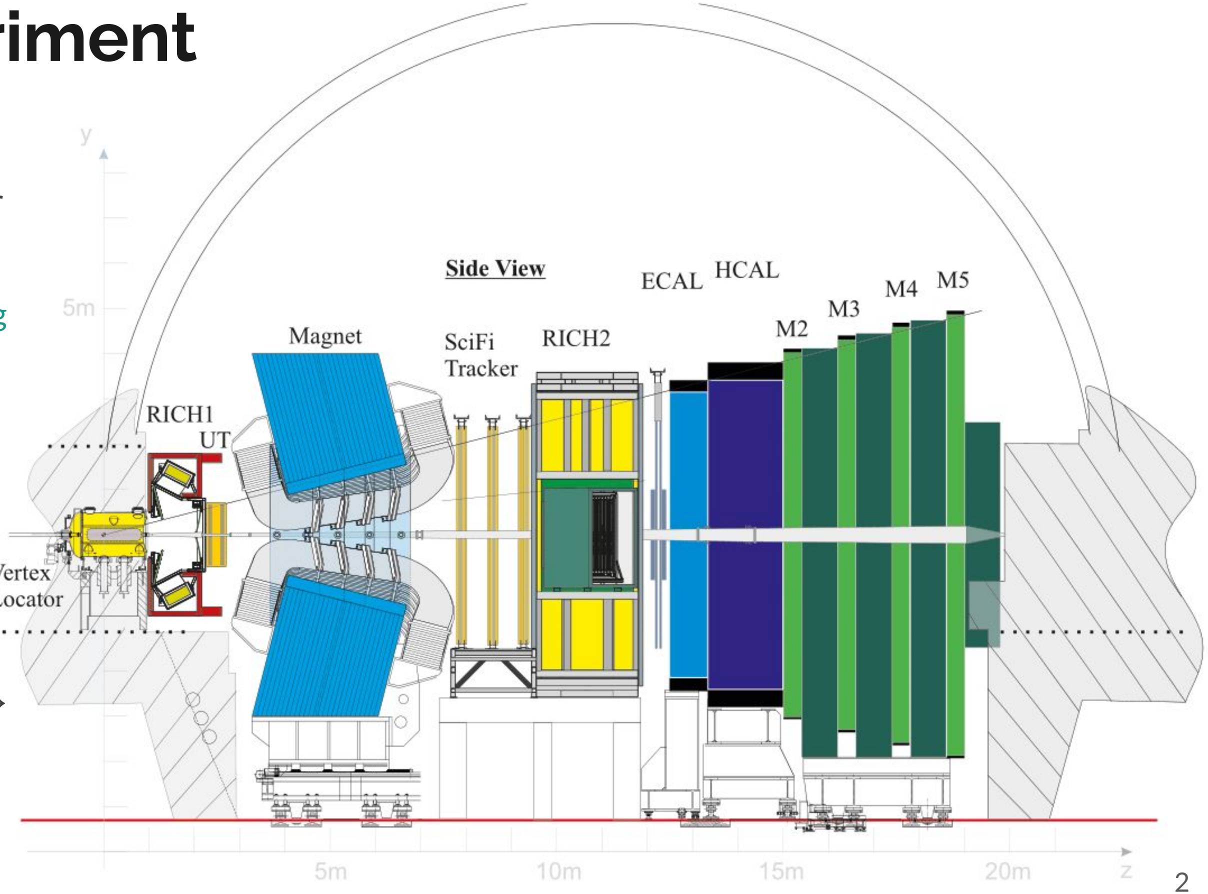
The LHCb Experiment

- Single-arm forward spectrometer for high-precision flavour physics
- High precision tracking and vertexing and PID
- Since 2022 (Upgrade I), operates at an increased luminosity → major upgrade in all sub-detectors

$$\mathcal{L} = 2 \times 10^{33} cm^{-2} s^{-1}$$

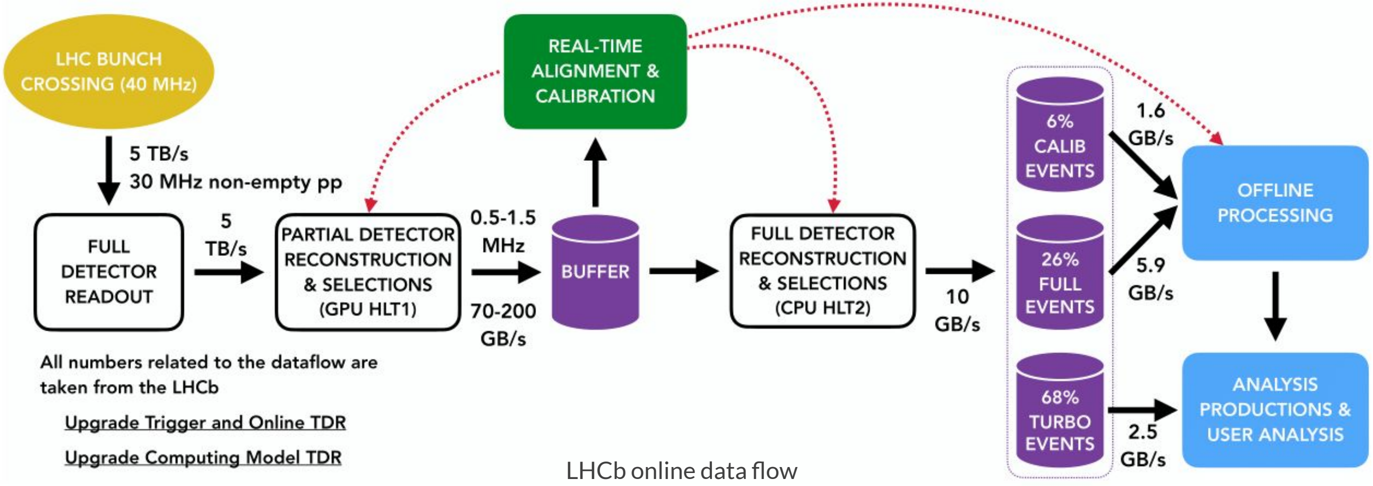
- Can't effectively trigger on heavy flavour using hardware signatures → full software trigger

[The LHCb upgrade I]



The LHCb data-flow

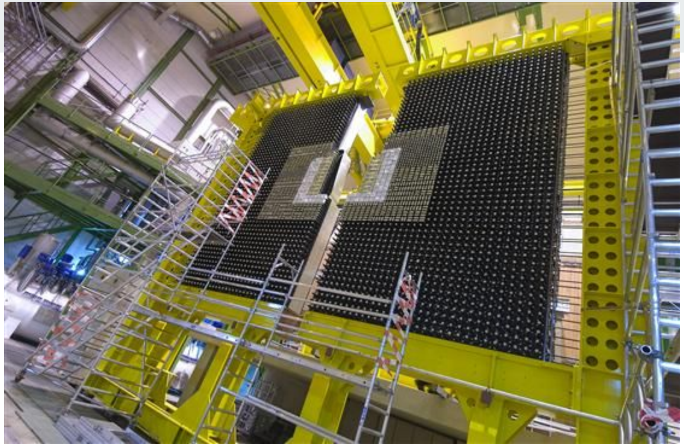
- The data flow generated from the LHCb detector currently reaches **5 TB/s**.
- Before storage, this rate is reduced by a factor 400 with the **trigger system**.
- **Real Time Analysis** approach: full event reconstruction and selection of specific signals of interest enabled by a quasi-real-time alignment and calibration.

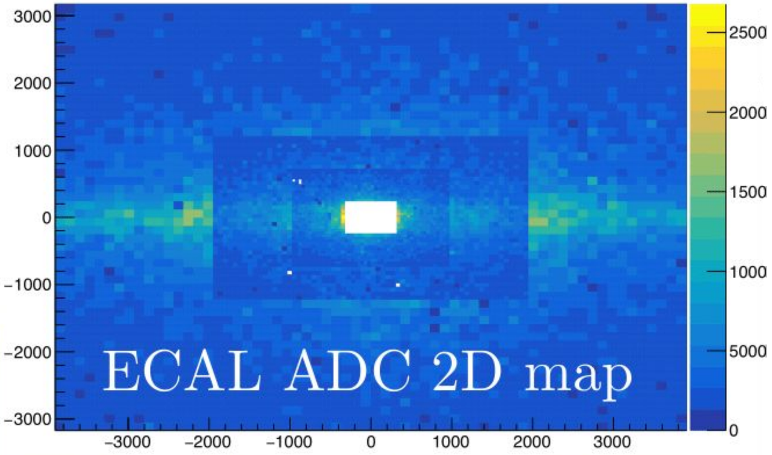


LHCb online data flow
LHCB-FIGURE-2020-016 (2020)

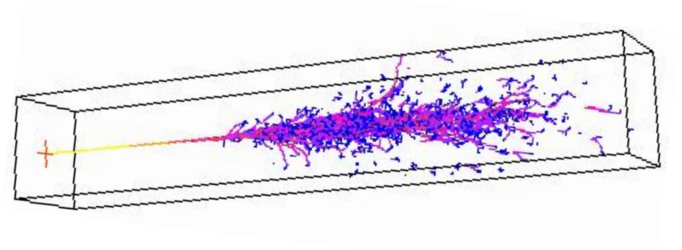
The ECAL detector

- The **electromagnetic calorimeter** (ECAL) is used for photon and electron identification.
- Makes high precision measurements of position and energy deposited.
- 2D grid of Shashlik modules with three active regions:
  - Inner → 4x4 cm^2 cell size
  - Middle → 6x6 cm^2 cell size
  - Outer → 12x12 cm^2 cell size
- Output data: Grid of “digits” → energy deposits (MeV) on each calorimeter cell.



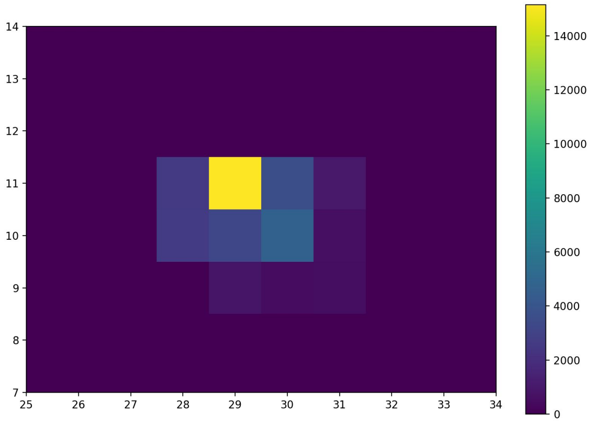


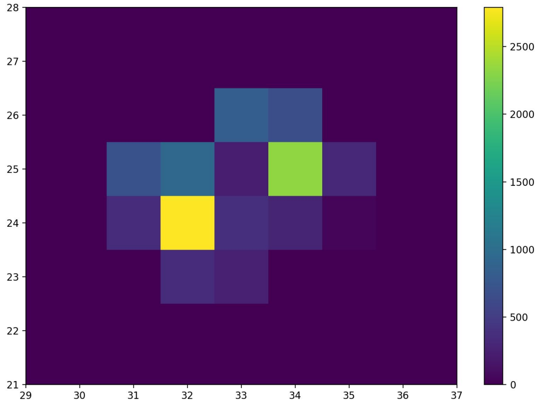
ECAL detector and ECAL ADC 2D map
LHCb-TALK-2022-163

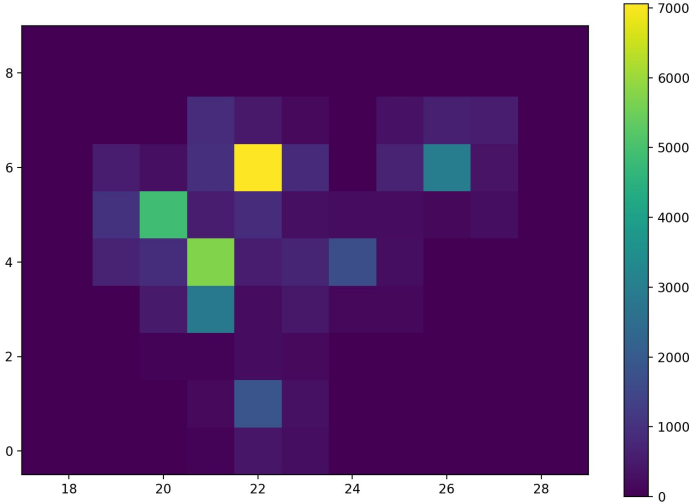


ECAL data reconstruction

- Reconstruction: Cluster the digit deposits from the same particle (typically 3x3).
    - Look for local maxima → cluster seeds
    - Expand the clusters to neighbouring deposits

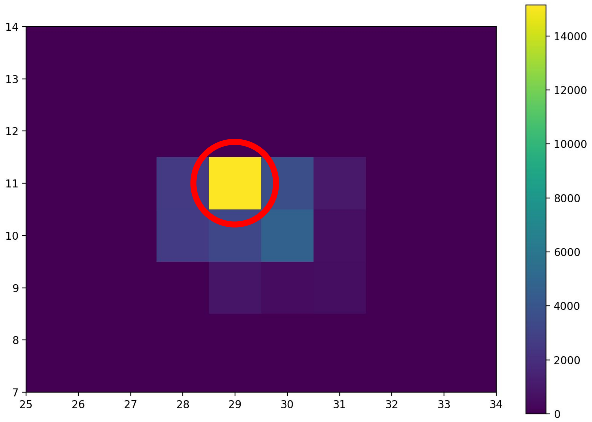


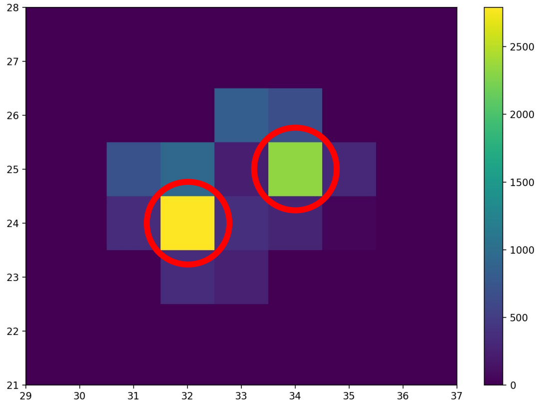


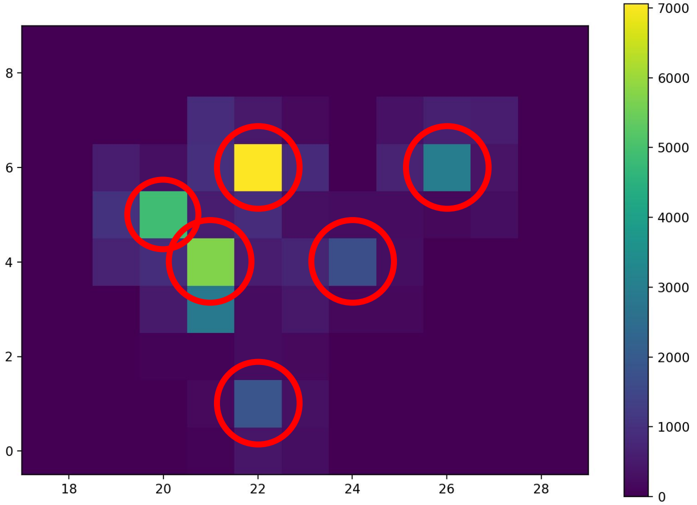


ECAL data reconstruction

- Reconstruction: Cluster the digit deposits from the same particle (typically 3x3).
    - Look for local maxima → cluster seeds
        - Adjacent peaks → merged pi0 candidates

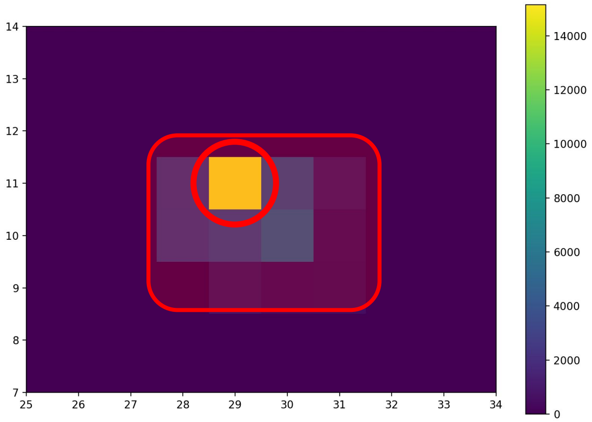


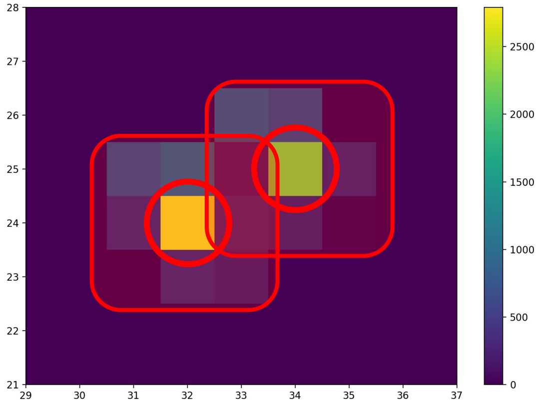


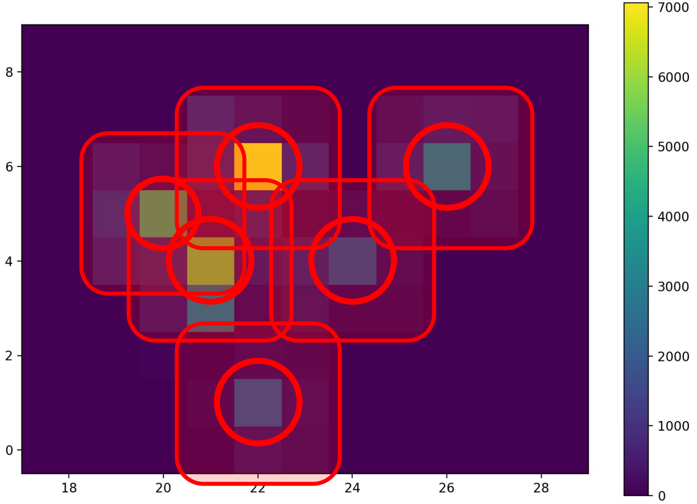


ECAL data reconstruction

- Reconstruction: Cluster the digit deposits from the same particle (typically 3x3).
  - Expand the clusters to neighbouring deposits
    - Overlapping clusters → distribute the energy in between







Classic approach to ECAL reconstruction

Peak detector	Cluster reconstructor	Overlap solver	Cluster corrections
Find local maxima	Tag energy cells around maxima	Fraction energy values for overlapping cells	Can be done with reconstructed data
One iteration for all cells at least	One iteration through all the cells at least	Needs more than local information	
Is "Cellular Automata (CA)-like"	Is a CA		

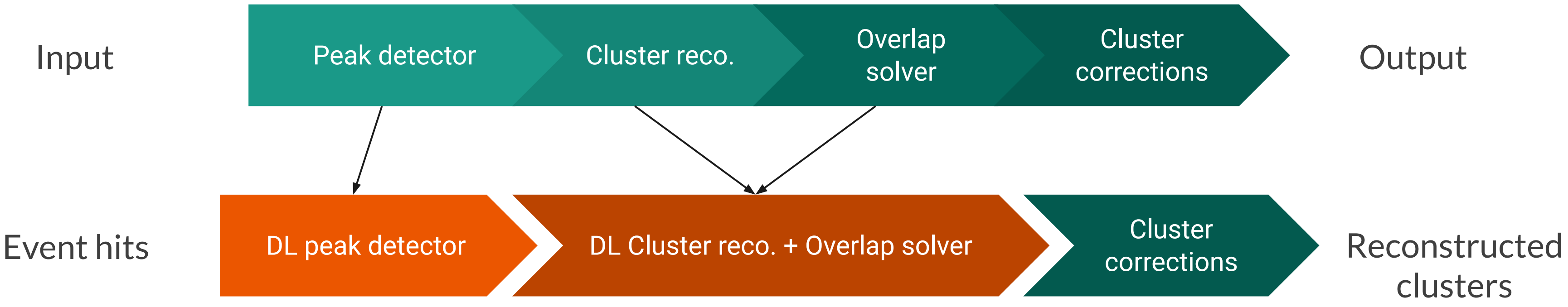
Calorimeter reconstruction takes 25% of High Level Trigger 2 sequence time

LHCb Public Results

Deep Learning approach

Base idea → The rules of a Cellular Automata can be **learned** by a deep convolutional architecture

Gilpin, W. (2019). Cellular automata as convolutional neural networks. *Physical Review E*, *100*(3), 032402



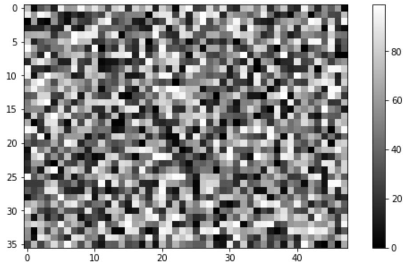
DL peak detector → DL Cluster reco. + Overlap solver → Cluster corrections

Peak finder

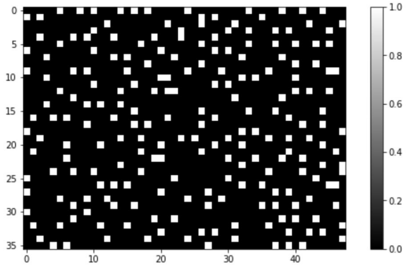
The ruleset defines **comparisons**: Identify local maxima **independently of the numerical value scale**.

Input | Output

Training:
1k images of random values (0-99)

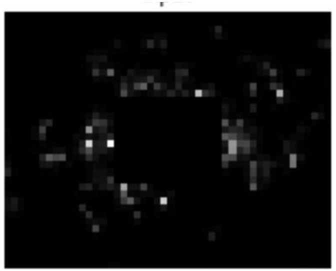


→ CNN →



2D Conv. + Dense layers
1272 parameters

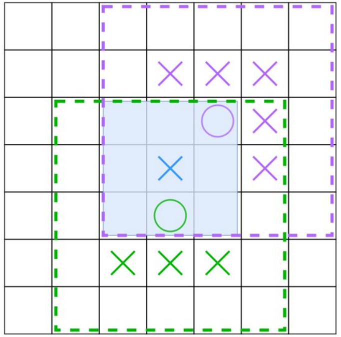
Testing:
with ECAL simulation data
0.9973 acc



→ CNN →

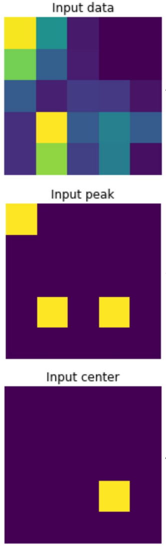
DL peak detector → DL Cluster reco. + Overlap solver → Cluster corrections

Cluster reco + overlap



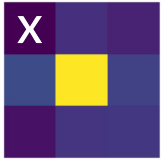
The ruleset defines **fractions**: Need to model **non-linearities** → 2D Convol. Layers would require a very complex kernel (lots of parameters)

→ We need a **MLP** to be the **kernel of the convolution**

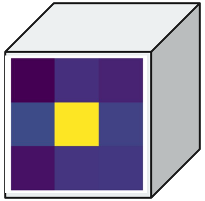


Regression MLP:
5 Hidden Layers
(~100k params)

X
Reconstructed single cell value



Aggregate 9 cell values to get the reconstructed cluster



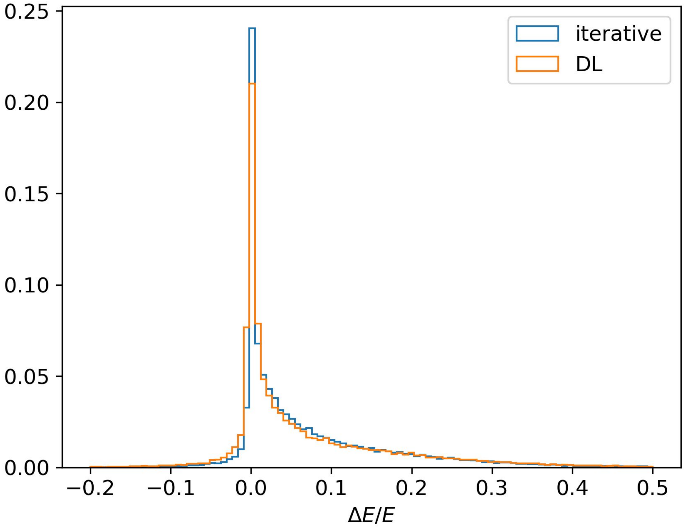
Stack the clusters of all the windows to get the **reconstructed cluster stack** of an event

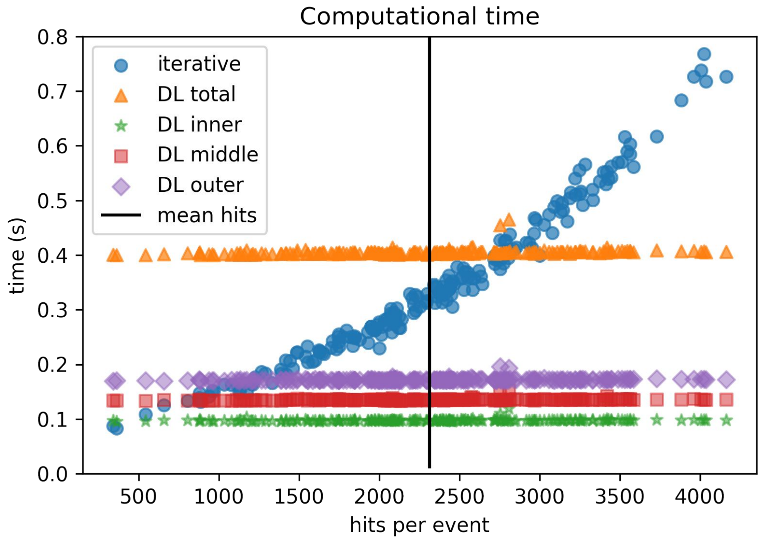
Dataset: 200k balanced 5x5 images from 2k events

1.05% relative error in **testing**

Results achieved

Local comparison between an **iterative method** (same complexity as the current algorithm) and the proposed **DL method**





- Coding: Python 3.8
- Networks: Tensorflow 2.3.0
- Test environment: Intel CPU, 4 cores

Results achieved

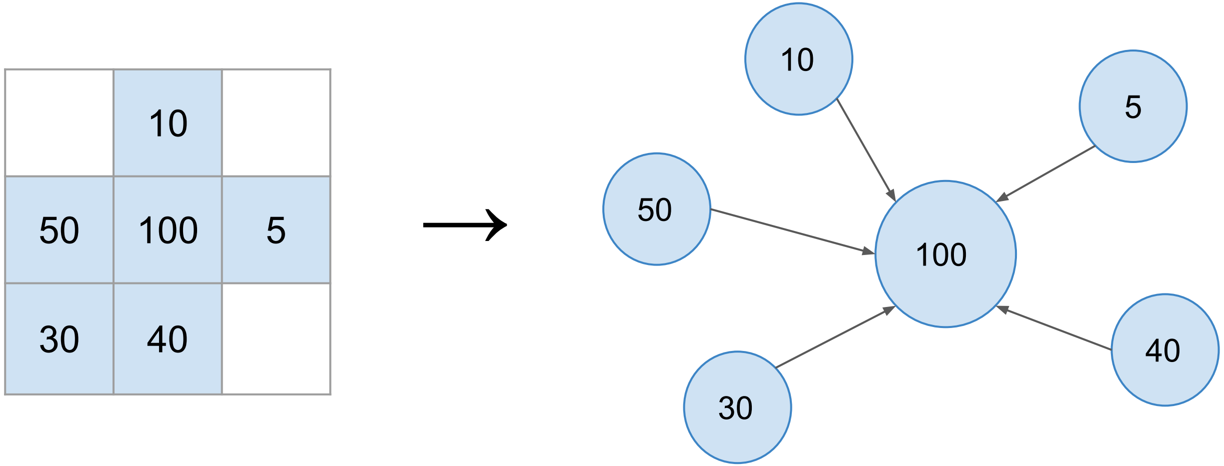
- Resolution not comparable to LHCb but good learning of the rules achieved
- Nearly **constant behavior** with independence of the events complexity
- Only faster in **25%** of the events
- The **volume of data** needed for the network to **generalize knowledge** is much less than training with the whole calo images and can be **artificially generated** in some cases to avoid depending on simulation data
- Each step can be tuned and retrained **adding new features** (e.g. train directly with Monte Carlo data)
- Inference engines are still evolving to provide fast DL inference inside the LHCb framework

New approach: Graph Clustering algorithm

Baseline idea:

- Use **graphs** as a data structure to **store digits**.
- With an **insertion** under certain rules, digits from the same cluster are already **grouped** together.
- **Overlap cases** are contained into independent **connected components** of the graph.

Isolated cluster:



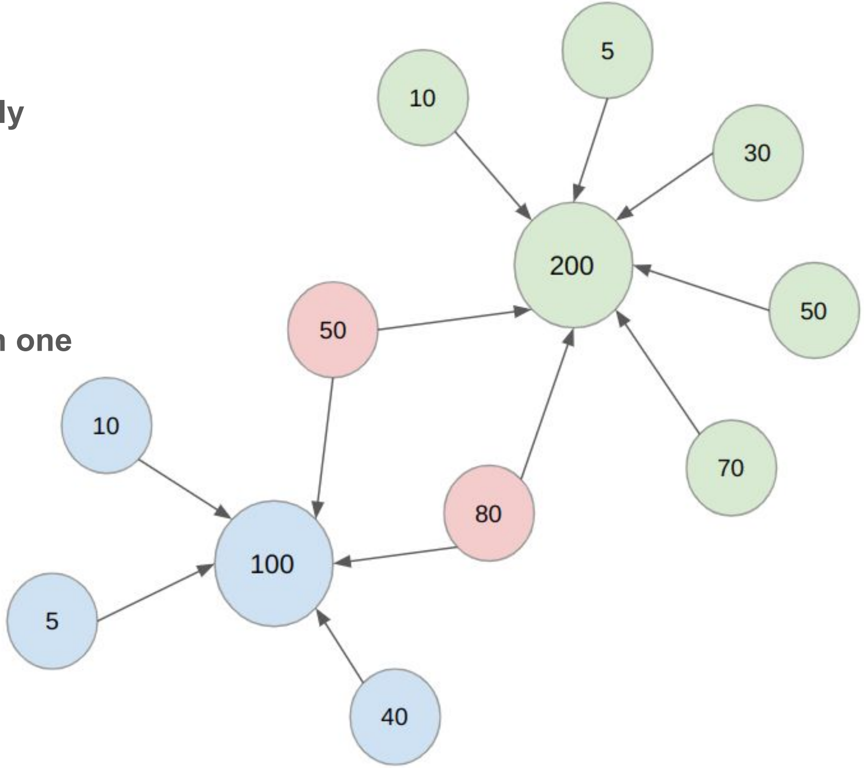
Graph Clustering algorithm

- Each energy **digit** becomes a **node**.
- Local maxima (**seeds**) are identified as nodes with **only input edges**.
- **Edges** between nodes indicate that the source node **belongs to the target** node (seed).
- **Overlapping cells** are identified for having **more than one** output edge.

			10	5	
	10	50	200	30	
5	100	80	70	50	
	40				

Two overlapping clusters

→



Graph Clustering algorithm

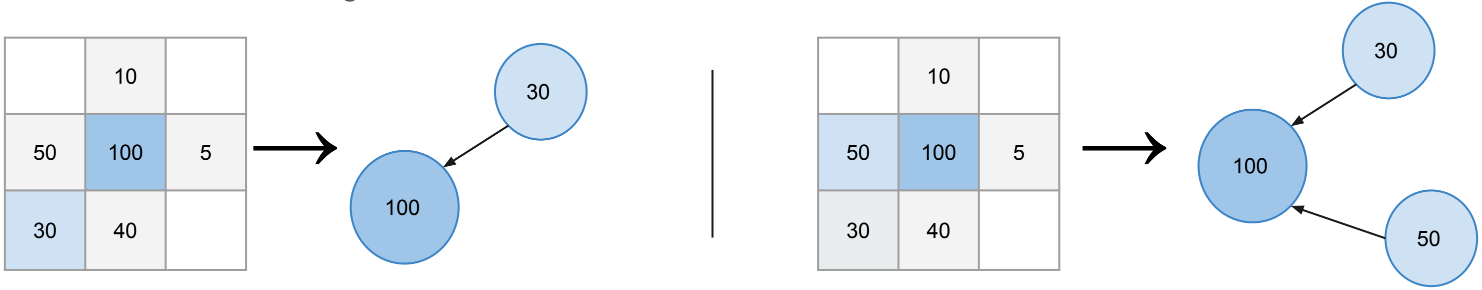
Algorithm steps:

1. **Sort** the event digits by decreasing energy
    - It is needed to make sure the seeds of clusters are inserted in the graph before its neighbor digits
    - 50 MeV is the minimum energy of a digit to be considered a seed
    - Only digits above 50 MeV are sorted by decreasing energy value
2. **Insert** digits into the graph.
3. Get the **connected components** of the graph.
4. **Analyse** each connected component to build the clusters.

Graph Clustering algorithm

Algorithm steps:

2. **Insert** digits into the graph
   - Start with the highest energy digits (only possible seeds):
     - If it is already inserted in the graph it is already part of a cluster → cannot be a cluster seed.
     - It not, if it is a local maxima → is a cluster seed.
       - Insert all the distance 1 neighbors to the graph and link them to the seed with a directed edge.

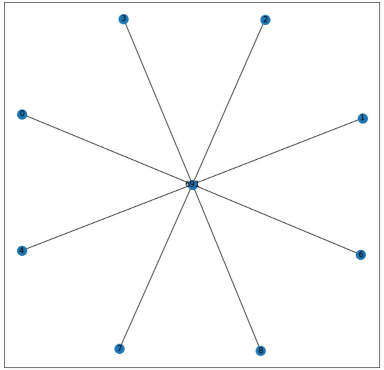


Graph Clustering algorithm

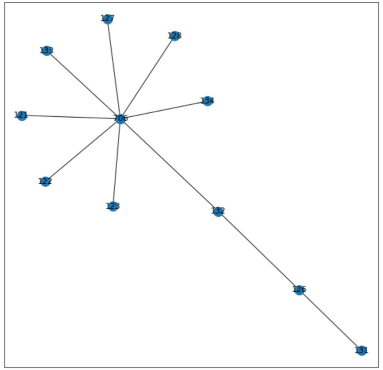
3. Get the **connected components** of the graph.

- Given a directed graph, a weakly connected component (WCC) is a subgraph of the original graph where all vertices are connected to each other by some path, ignoring the direction of edges.
- ~60% of WCCs are already isolated clusters itself

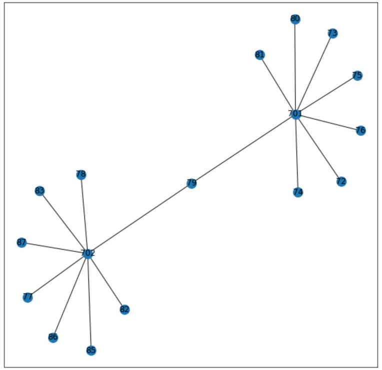
Isolated cluster

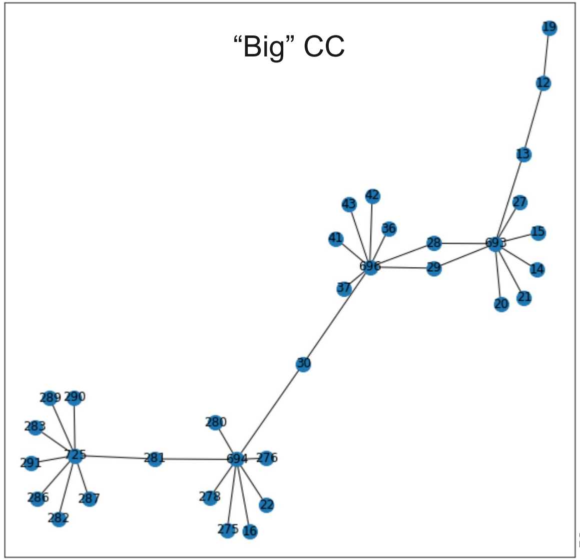


Cluster overlapping with small residual cluster

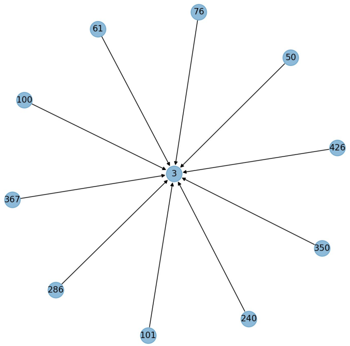


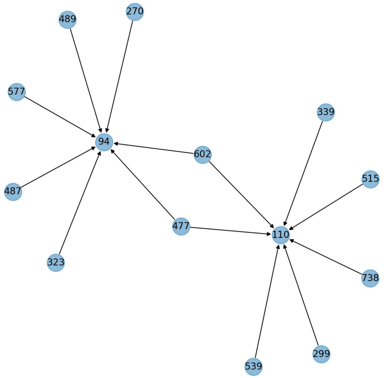
Two clusters overlapping in one cell

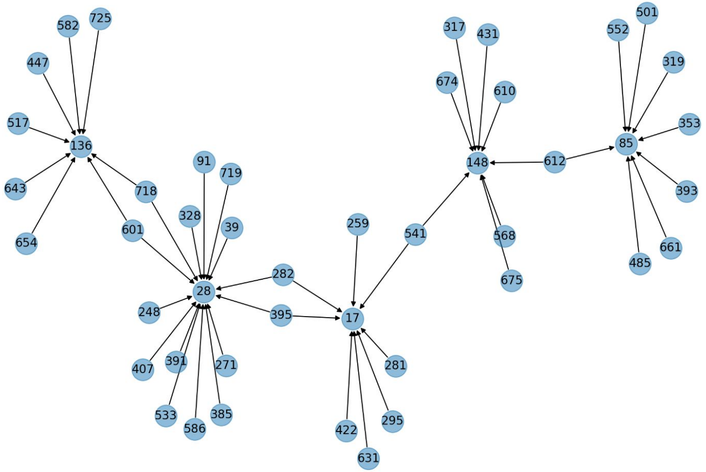


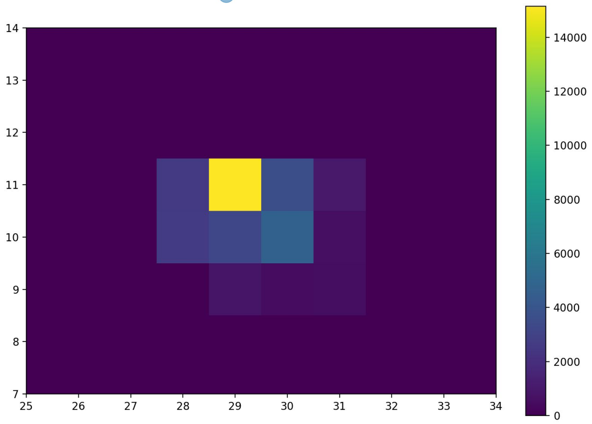


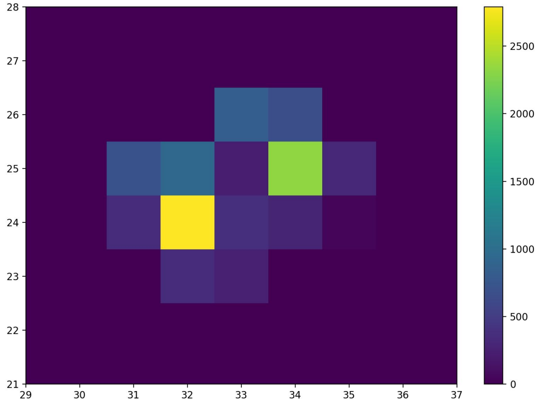
Graph Clustering algorithm

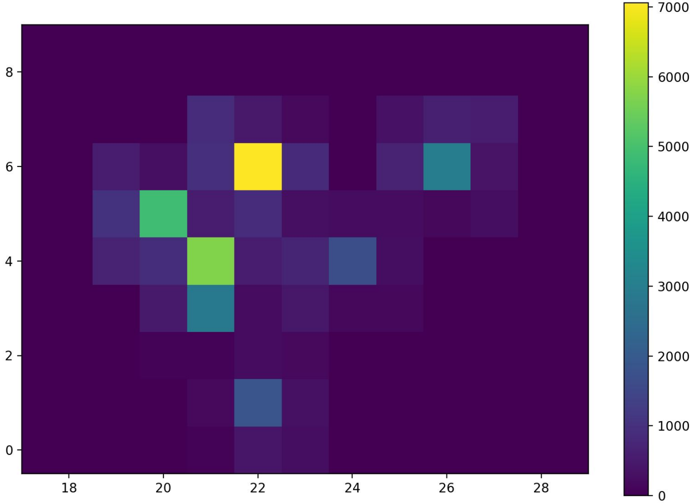












Graph Clustering algorithm

Algorithm steps:

4. **Analyse** each connected component to build the clusters.

    - Identifies overlap cells and calculates which fraction of its energy is assigned to each cluster.
    - Uses an estimation based on the total energy of each cluster:

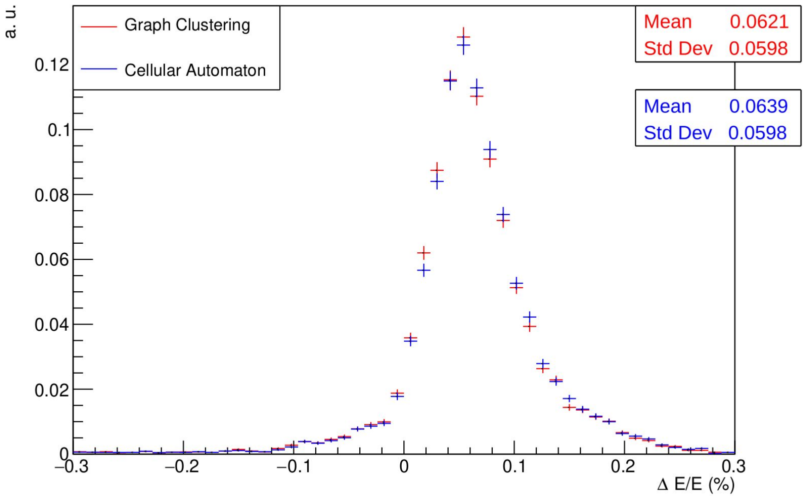
$$fraction_{cluster1} = \frac{E_{cluster1}}{E_{cluster1} + E_{cluster2}}$$

    - Although it takes ~3 iterations to fully converge, with only 1 iteration the efficiencies do not change significantly and execution time is minimized.
    - At the end of this step, the list of clusters is already completed with the information of: `seed digit`, `total energy`, list of `digits in the cluster` with `id`, `energy` and `fraction`.

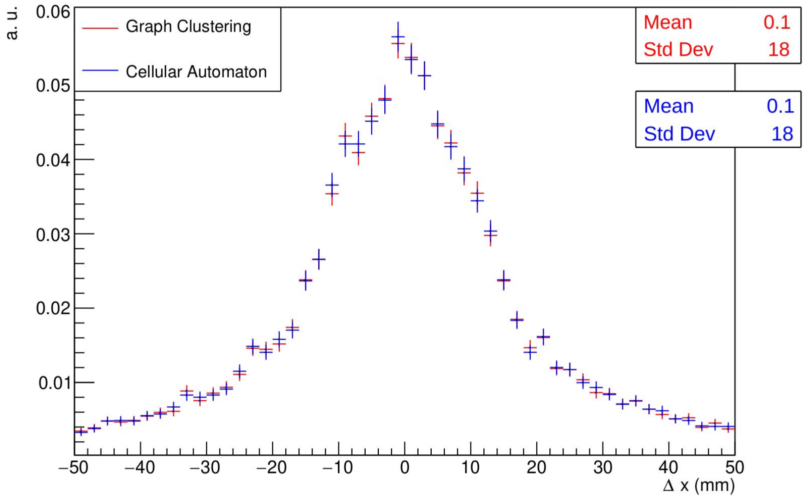
Results

Efficiency, energy and position resolution

Algorithm	Reconstructible	Reconstructed	Efficiency (%)
Graph Clustering	43234	35313	81.68 ± 0.19
Cellular Automaton	43234	34872	80.66 ± 0.19



E resolution with γ samples

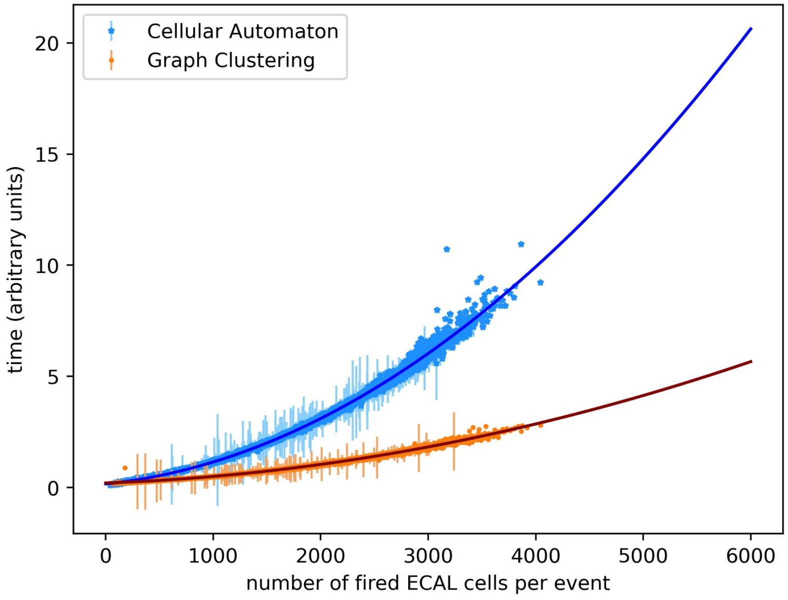


X resolution with γ samples

Results

Execution time

- Measured using the same data, conditions and environment in both algorithms.
- GC is **65.4% faster** than CA on average.
- On low number of digits/event (<150 digits) CA is faster.
- Overall complexity of GC is lower, good for high digits per event scenarios.
- It increases the HLT2 throughput up to 9%.



Conclusions

- Graph Clustering is now the default calorimeter clustering solution for Run 3
- Equivalent efficiency and resolution but improving the computational complexity of the previous method
- Provides a flexible definition of clusters → different shapes per region, future upgrades with different granularity, etc.
- Check the code in GitLab and detailed documentation in EPJ C article!

Thank you!

Any questions?

Núria Valls Canudas
Smart Society Research Group - La Salle URL Barcelona
On behalf of the LHCb Real Time Analysis Project

3rd COMCHA School
23 – 31 October 2023

Backup

Núria Valls Canudas
Smart Society Research Group - La Salle URL Barcelona
On behalf of the LHCb Real Time Analysis Project

3rd COMCHA School
23 – 31 October 2023

DL Limitations and constraints

Need to use inference engines to use neural nets inside the LHCb framework:

ROOT / TMVA

Used in LHCb for offline analysis

Mainly use BDTs and MLPs

Efficient inference is not scalable and hard to maintain

ONNX

Open format built to represent machine learning models

Compatible with C++ HLT2 framework in LHCb

TensorRT

High-performance deep learning inference for CUDA environments

Allows to read and use ONNX files

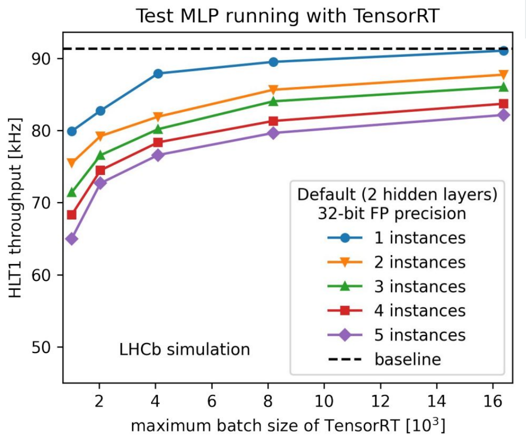
Compatible with CUDA HLT1 framework in LHCb

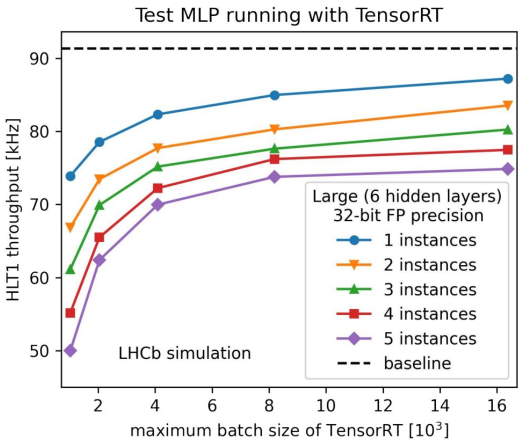
DL Limitations and constraints

Recent studies have started testing inference engines inside HLT1 using TensorRT.

If we extrapolate the throughput impact:

- Three instances of the peak detector O(1k) parameters
  - ~6% throughput reduction
- One instance of the MLP O(100k) parameters
  - ~5% throughput reduction
- Broad estimation of ~11% of throughput reduction for the inference, current ECAL clustering in HLT1 represents 4% of the sequence

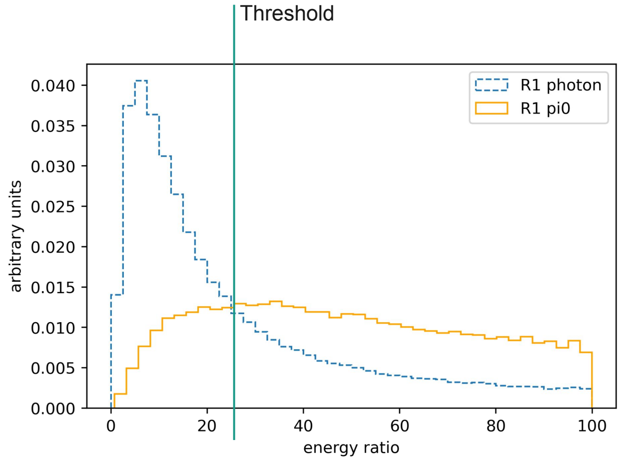


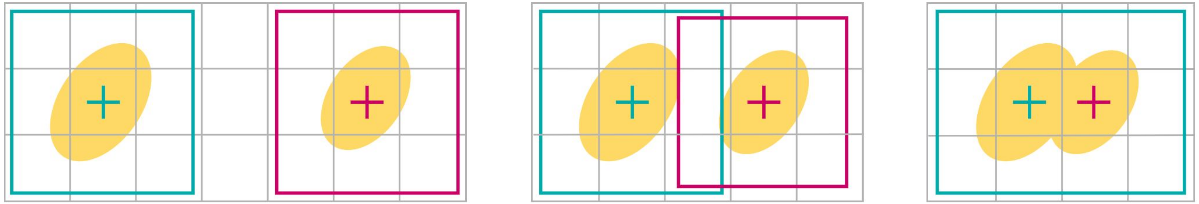


Algorithm steps

2. Insertion: particular case

- Neutral pions decay into two photons before reaching the calorimeter.
- Merge π^0s are reconstructed as a single 3x3 cluster (by default), leaving significant energy deposits out.
    - To avoid this, potential merged π^0s are filtered by the energy ratio between the seed and the second most energetic deposit (R1).
    - This clusters are expanded to the neighbor cells of the second seed.





Separable photons, no overlap   Resolved π^0 Merged π^0