FaDER

Fast **D**ata and **E**vent **R**eduction

Lowering data volumes in high-intensity experiments

Andrea Coccaro, Carlo Schiavi, ***Alessandro Zaio***

Università di Genova

ICSC

Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

The FaDER Project

Structured in two main areas of intervention, related to high-intensity particle physics problems

Real-time data processing

Target case: BDX @ JLAB, where no standard trigger and event definitions are applicable.
Focus on developing streaming readout (SRO) systems capable of reducing data size during extraction.
To achieve required speed of operation, heterogeneous computing systems with accelerators are targeted.

Real-time data reconstruction

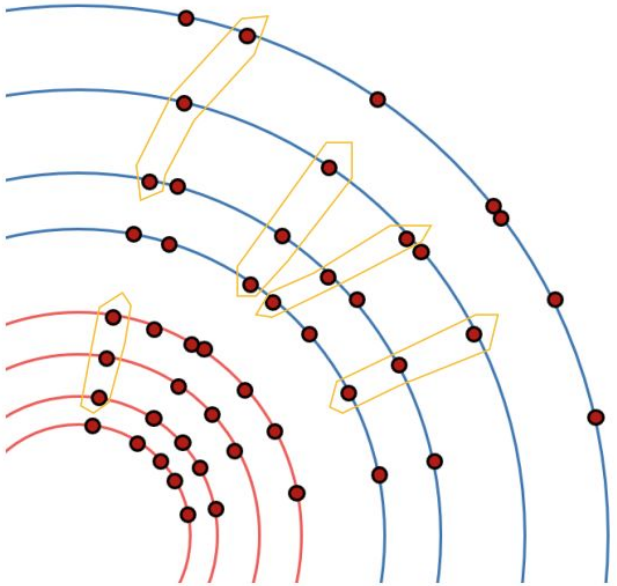
Target case: ATLAS @ CERN, where high luminosity requires real-time physics object reconstruction.
These are heavily based on track reconstruction in the innermost Silicon detectors.
Development of ML models capable of pre-selecting "interesting" detector hits, to assist the tracking task:
→ filtering out noise and pile-up hits (CNN) → **will focus on this in the next slides**
→ extracting physics info, like the position of the primary interaction vertex (Transformer)
→ taking care of a part of the tracking task, clustering hits from the same track (object condensation)

Track Reconstruction in the HLT

The High Level Trigger operates a fast event reconstruction in order to decide whether the event should be saved or rejected.

A key step in the HLT is the track reconstruction (***tracking***) of the charged particles passing through the innermost layers of the detector.

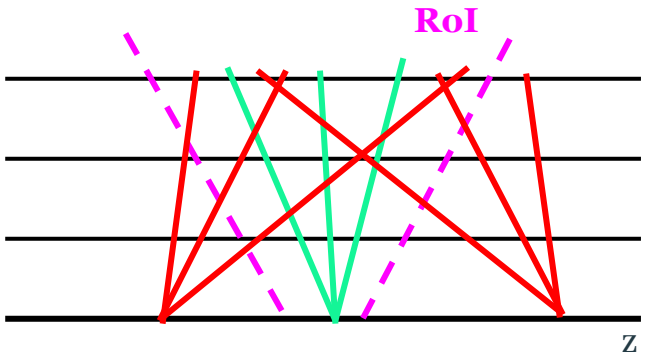
Tracking is the most time and CPU consuming algorithm inside the HLT and is also a strongly **combinatorial problem,** since the number of computations increases at least with N^2 , with N the number of hits being considered.



Moreover an upgrade for the LHC is planned for the 2029, which will increase the number of *pp* collisions and hence hits in the detector, so that the latency for tracking will further increase.

Assisting real-time tracking with hit filtering

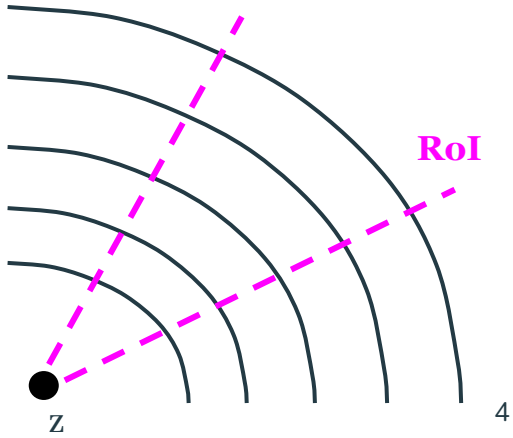
In order to save time, tracking is performed in limited regions of the detector, also called Regions of Interest (RoIs).
Most of the hits inside an RoI are originated by charged tracks coming from outside the RoI, from *pp* interactions which we are not interested in.



A way to reduce the latency for tracking could be to develop a **filtering algorithm to remove the background hits from the RoI** so that less combination have to checked by tracking.

The algorithm that we are developing is based on a Convolutional Neural Network (CNN) which received the images with both signal and bkg hits and filters away the noise.

CNN → easily accelerated on FPGA

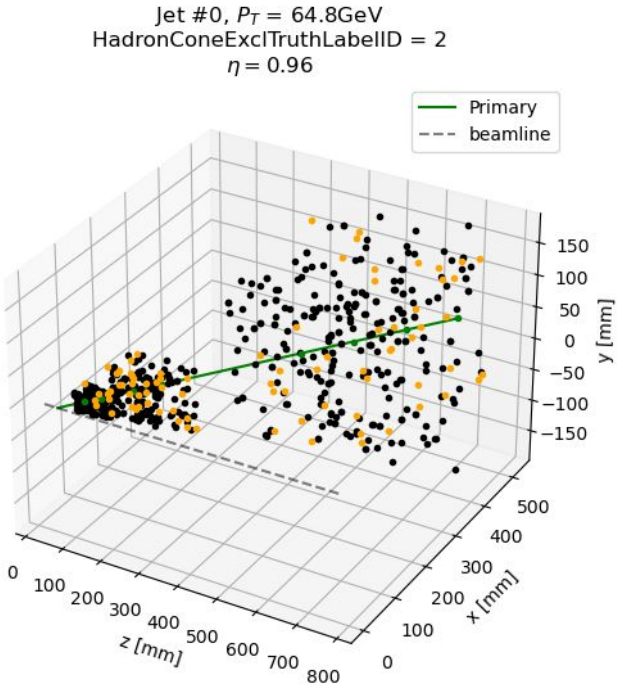


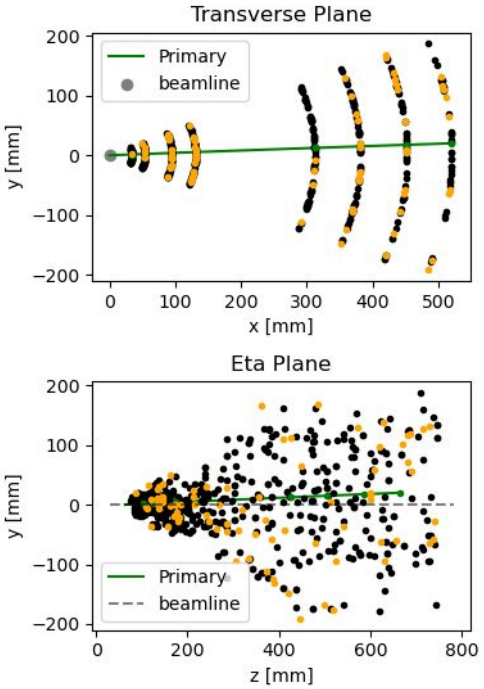
The Toy Dataset

We have built a toy event generator, which generates **signal and background collisions** and then overlays them in such a way to mimic the number of collisions seen in real Run 3 LHC events.

We then apply an **RoI** filter, in order to select the hits in a tunable angular cone. Possibility to remove the z cut.

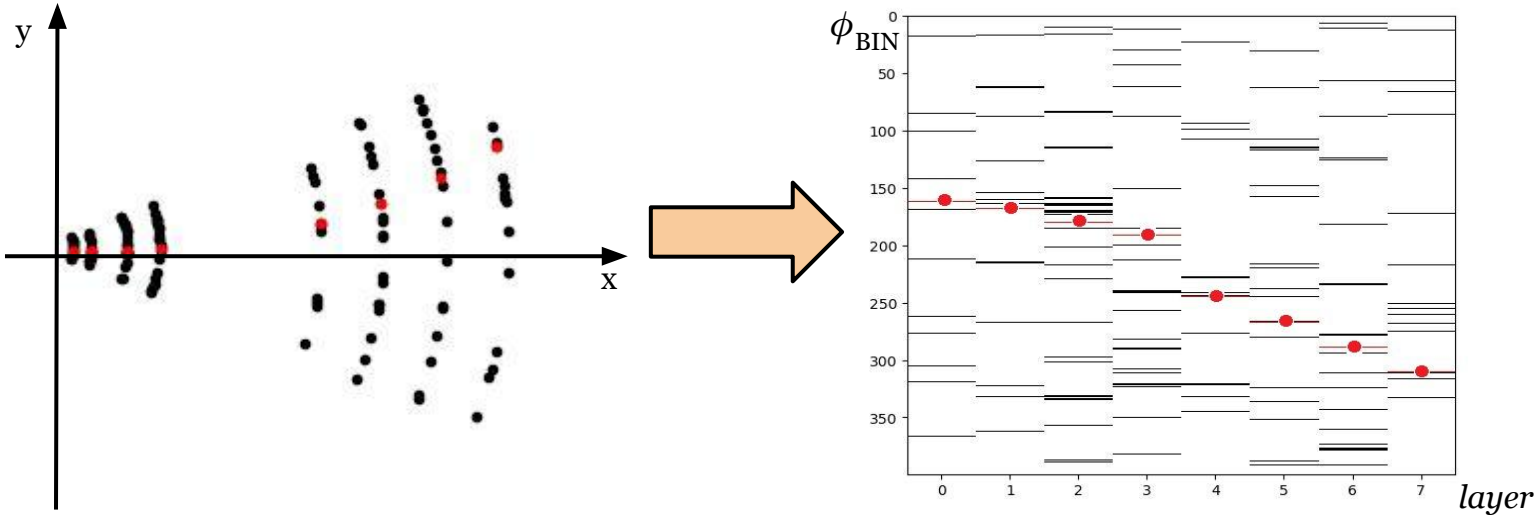
The simplified detector geometry follows the one for the ATLAS Inner Detector, limited to the Barrel part.





The Event Representation

We need to find a proper representation of the hit-information for the network.
To do so we start by taking the 2D projection of the hits on the x-y plane.
In order to obtain a grid-like and fixed-size representation suited for the CNN, we apply the following change of coordinates: $(x,y) \rightarrow (layer, \phi)$ defining $\phi_{\text{BIN-WIDTH}}$=0.02 rad



The CNN Model

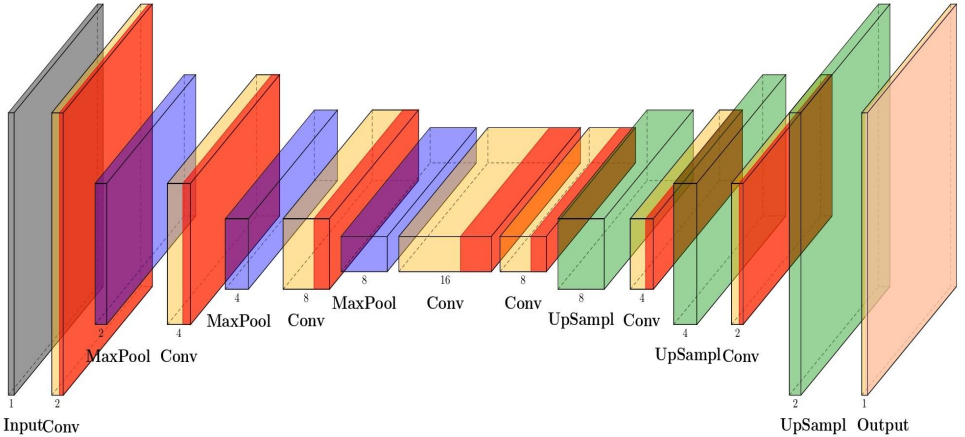
The labels of the grid-like input are associated in this way:

- if the bin contains either ***signal or noise hits*** → label has value **1**
- if the bin contain ***no hits*** → label has value **0**

The goal of the network is to obtain output images such that:

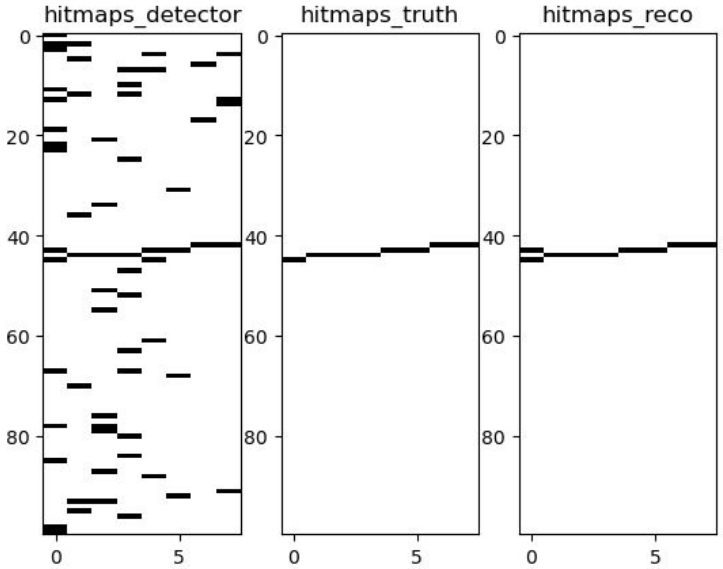
- if the bin contains ***signal hits*** → label=**1**
- if the bin contain ***no hits or noise hits*** → label=**0**

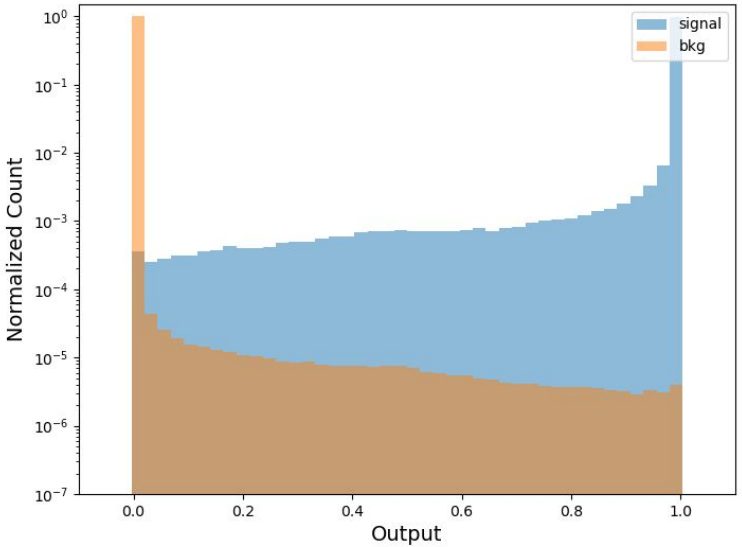
We use a **weighted BCE loss**, computed on the bins that had either signal or noise hits in input. Furthermore we use an Auto-Encoder architecture, frequently used for denoising problems.



Results

We conducted a training using a toy sample of **1M events** with an RoI of **0.1 rad**, which is typical size adopted at the HLT for e.g. single-muon candidates.
The signal tracks are generated in a ϕ interval of [-0.1 rad, 0.1 rad] and $p_T \in$[20GeV, 50GeV] which again follows the interesting target values of single μ transverse momentum.
The training has thus far been run on INFN GPUs, on our way to obtain Leonardo resources.

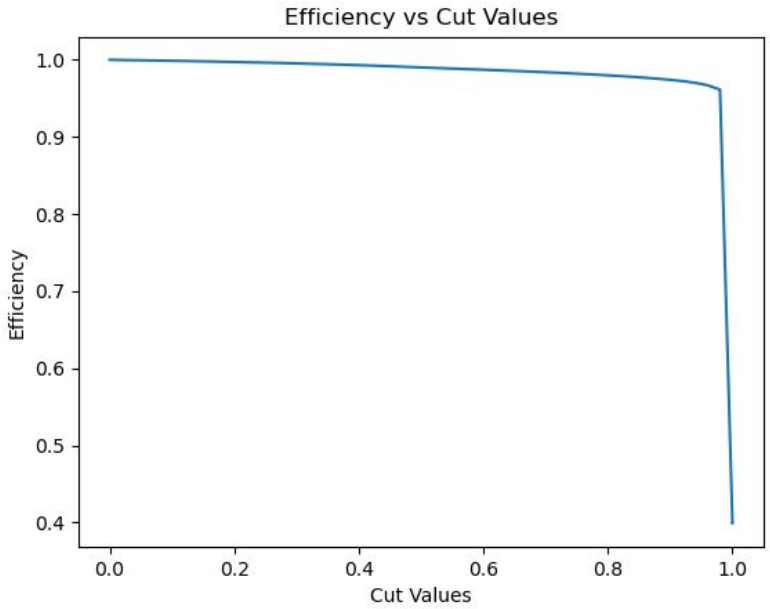


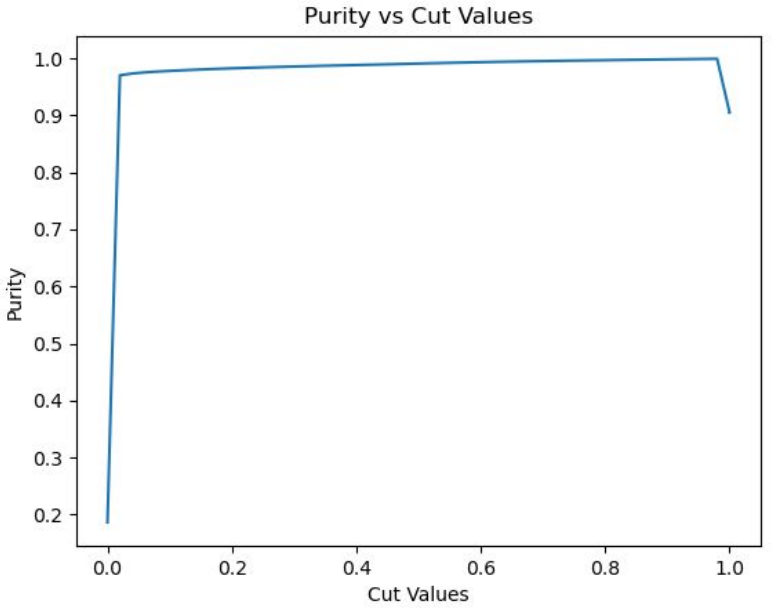


Results

$$\text{Efficiency} = \frac{\textit{num. of Signal Hits making the cut}}{\textit{total num. of Signal Hits}}$$

$$\text{Purity} = \frac{\textit{num. of Signal Hits making the cut}}{\textit{num. of Signal and Noise Hits making the cut}}$$





Next Steps

The network has shown potential to perform noise-hit filtering to assist tracking!

The following are some of the next steps we are planning:

- Extend to other case studies: larger RoIs or lower p_T ranges → larger bending of the tracks.
- Use both CPU and GPU **Leonardo** resources to extend the toy data production and training.
- **FPGA implementation** of the CNN using hls4ml to convert the ML model.

Thank you very much for listening! :)

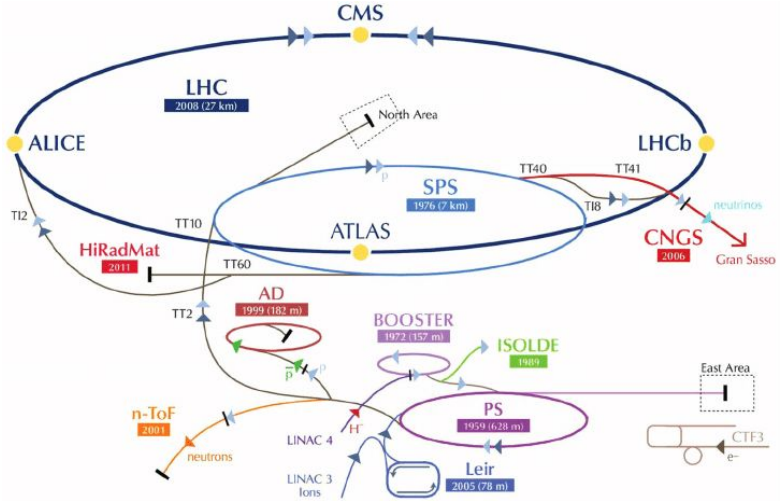
BACKUP SLIDES

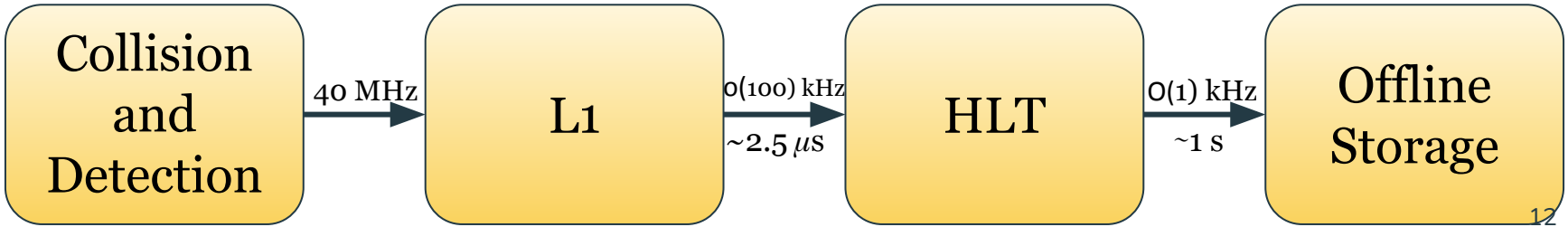
Trigger Systems of LHC Experiments

Within the Large Hadron Collider (LHC) at CERN proton beams are accelerated up to 10km/h from the speed of light and then collided at 4 experiments.

A collision inside each detector happens every 25 ns, so that the data volume corresponds to ~60 TB/s. It's necessary to use a trigger system to selects the most relevant events and store them.

- **Level-1 Trigger** (L1): hardware system.
- **High Level Trigger** (HLT): software system.



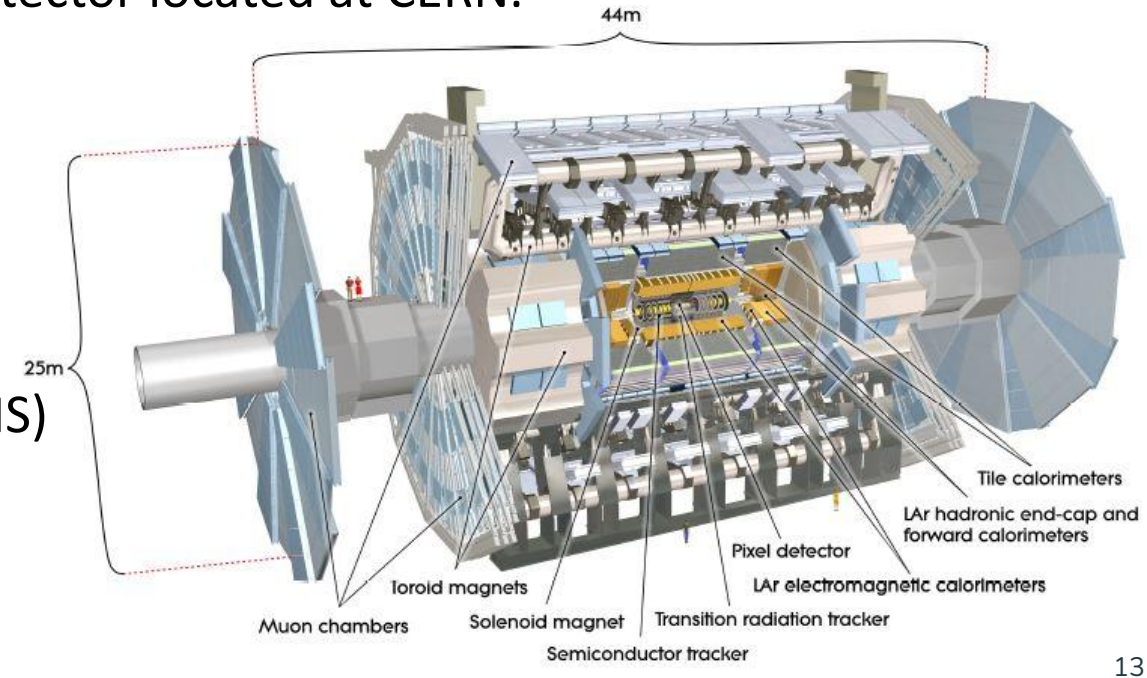


ATLAS Detector

The ATLAS (**A** **L**arge **T**oroidal **A**pparatu**S**) detector is the largest general-purpose particle detector located at CERN.

The detector is divided into three sub-detectors:

- Inner Detector (ID)
- Calorimeter System
- Muon Spectrometer (MS)

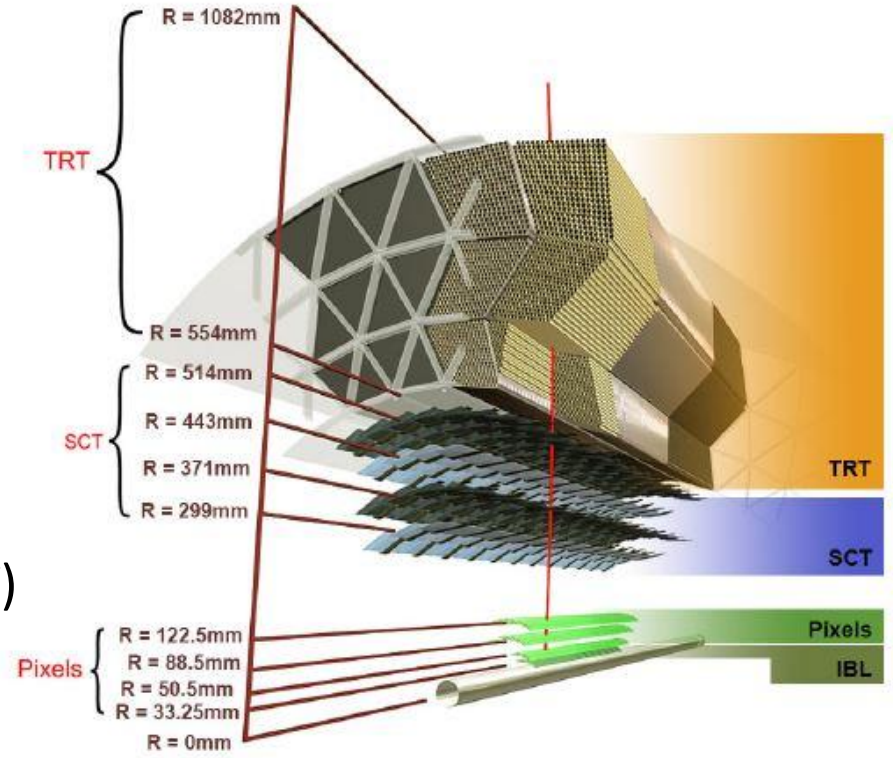


Inner Detector

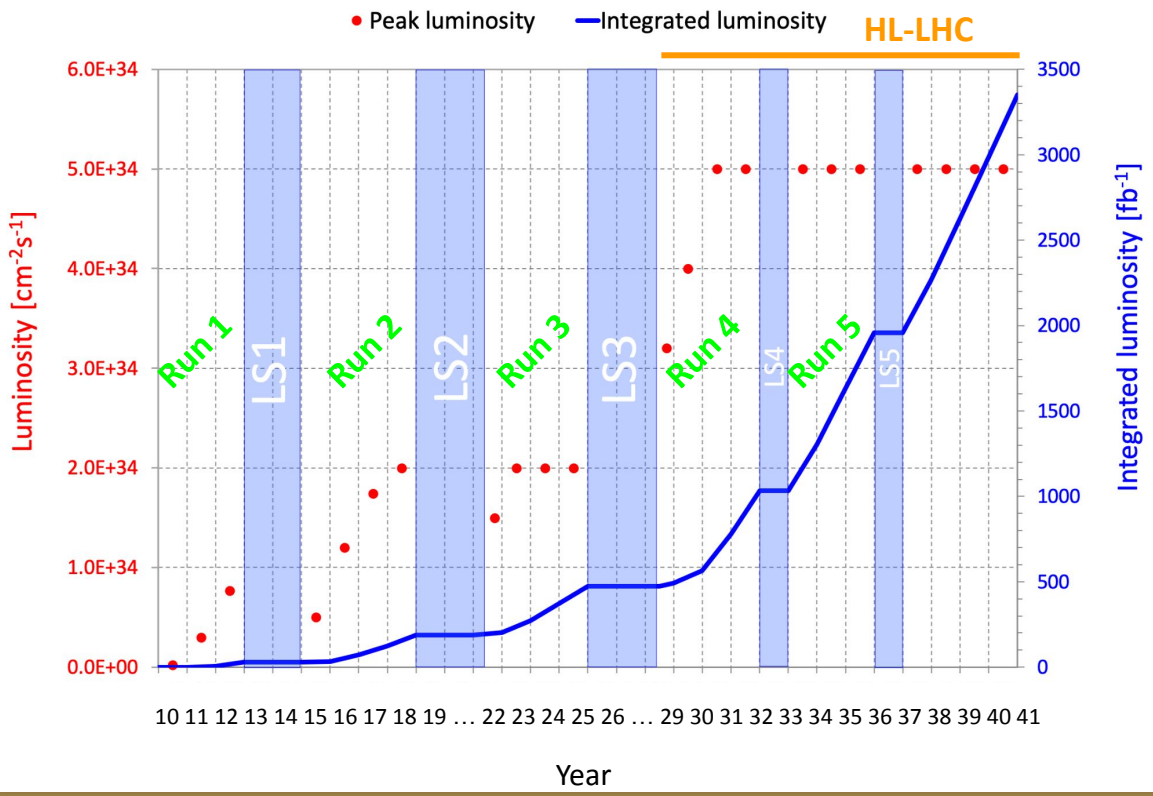
We focus on the ID, designed to provide precise tracking information for charged particles.

The ID is composed of:

- Pixels
- Semi-Conductor Tracker (SCT)
- Transition Radiation Tracker (TRT)



High-Luminosity LHC (HL-LHC)



After Run 3 a series of upgrades will increase the instantaneous luminosity. Also new tracking system.

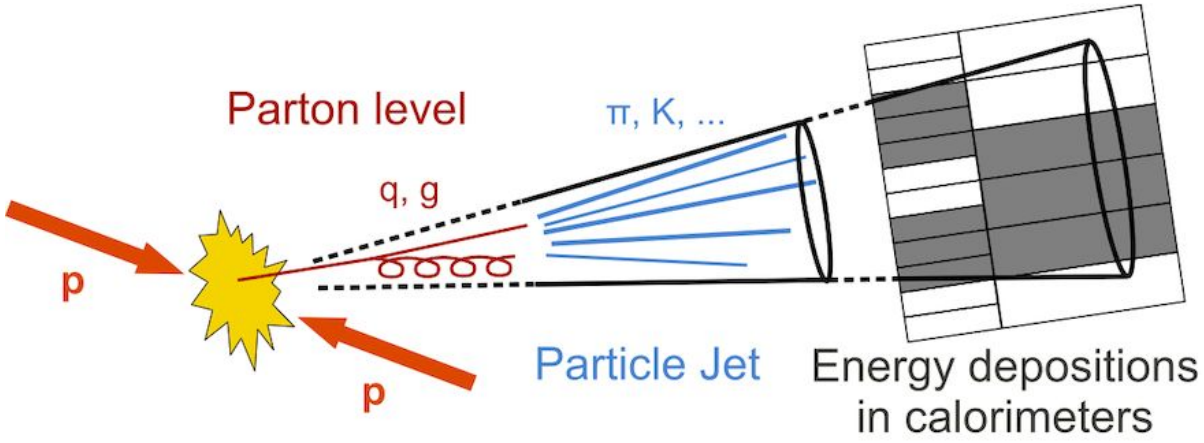
Starting from 2029:

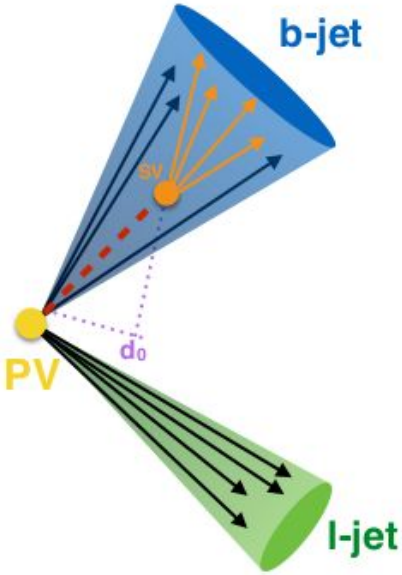
HL-LHC

Challenge for the Trigger System

Jets

A quark in a final state is observed as a shower of collimated hadrons.





Jets are found from energy clusters in the calorimeters.

Flavour of the jets → **Tracks are necessary**

At HLT events are selected according to the number and the energy of the jets reconstructed.

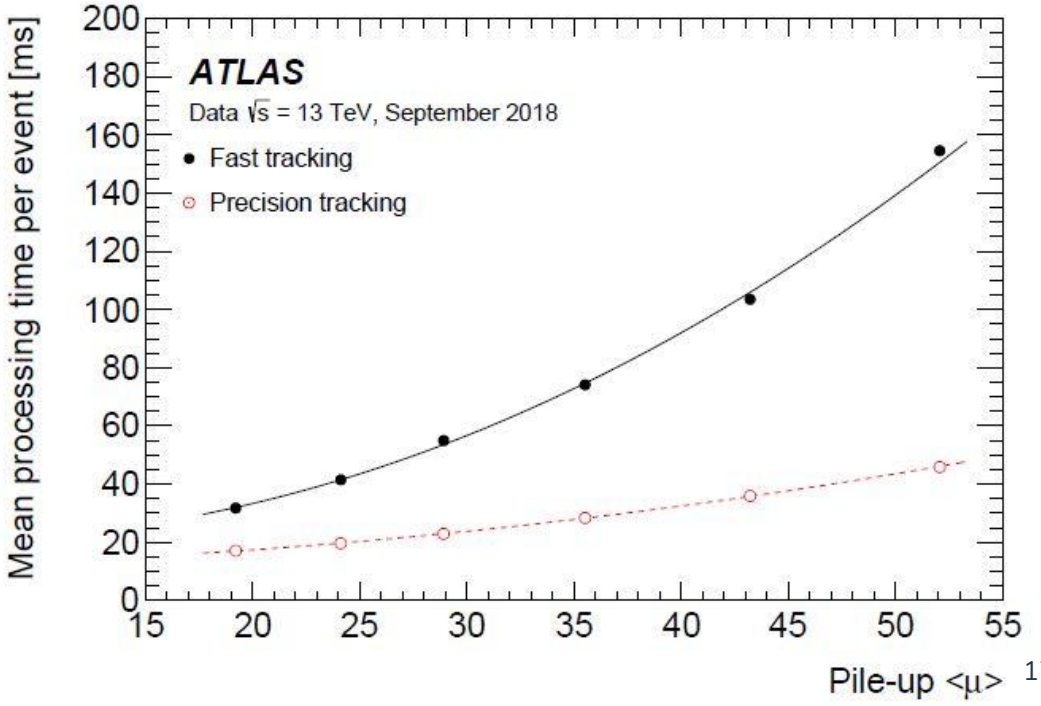
Also *b*-tagging information is used.

Tracking vs Pile-up

Pile-up <μ>: average number of *p-p* interactions associated to the same bunch collision.

Clear **power-like** increase of the FTF processing time as a function of the pile-up.

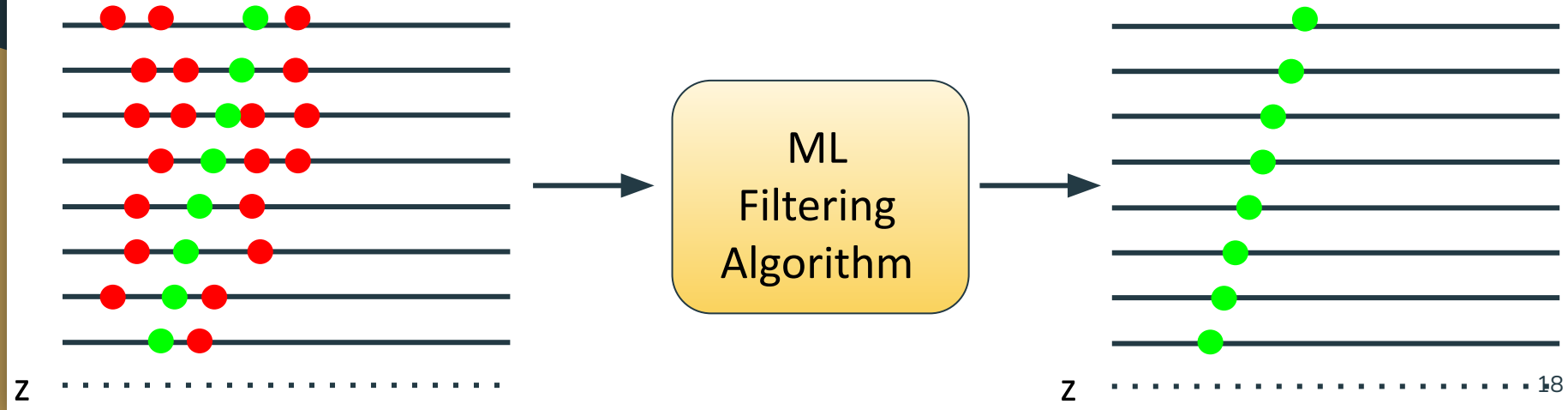
Higher values of <μ> imply
→ higher density of hits
→ higher number of possible combinations checked by the FTF



Assisting Tracking

I propose a new approach: a ML algorithm to **filter out pile-up hits**, in order to reduce the combinatorics of the FTF.

We start with independently generated events, with points from **curved tracks** as the signal and **noise hits** as the background.

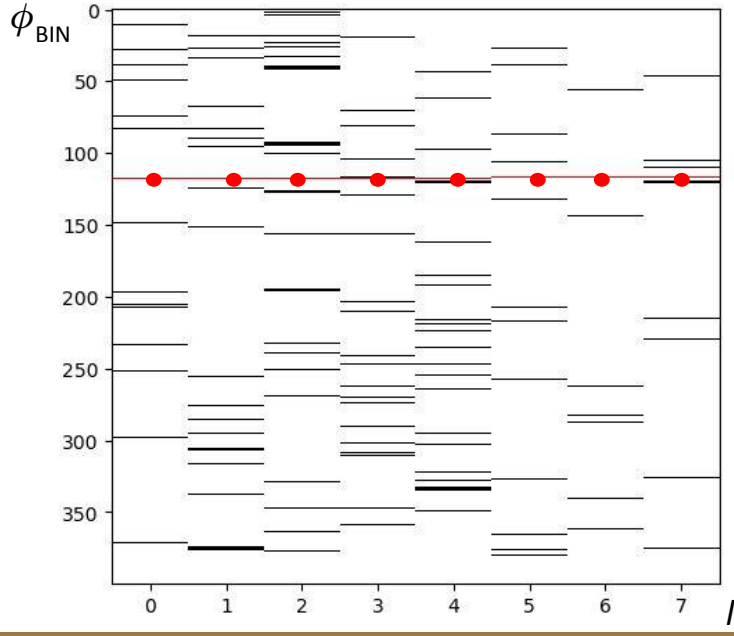


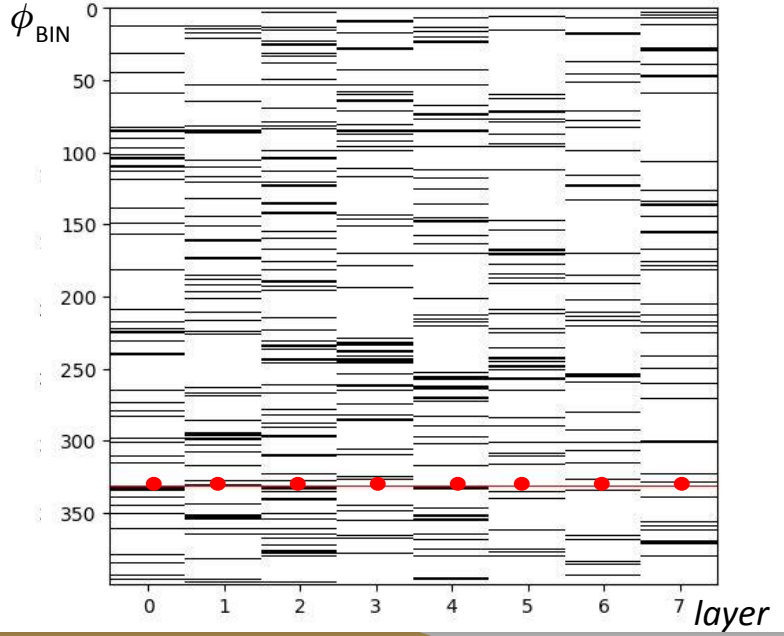
Hit-Maps

We want a format that can easily be used in ML algorithms.

→ We introduce a new set of variables: (*x, y*) → (*layer, ϕ)*

To do so we introduce a binning in ϕ, with ϕ_{BIN}= 0.02 rad.





Filtering Algorithm

Goal of the algorithm: receive images with both track and random hits, and output an image with only track hits.

INPUT:

Signal and **Background**: value 1

Other pixels: value 0

OUTPUT (ideally):

Signal: value 1

Background: value 0

$$L = \frac{1}{N}\sum_{i=0}^{N}(y_i - \overline{y_i})^2$$

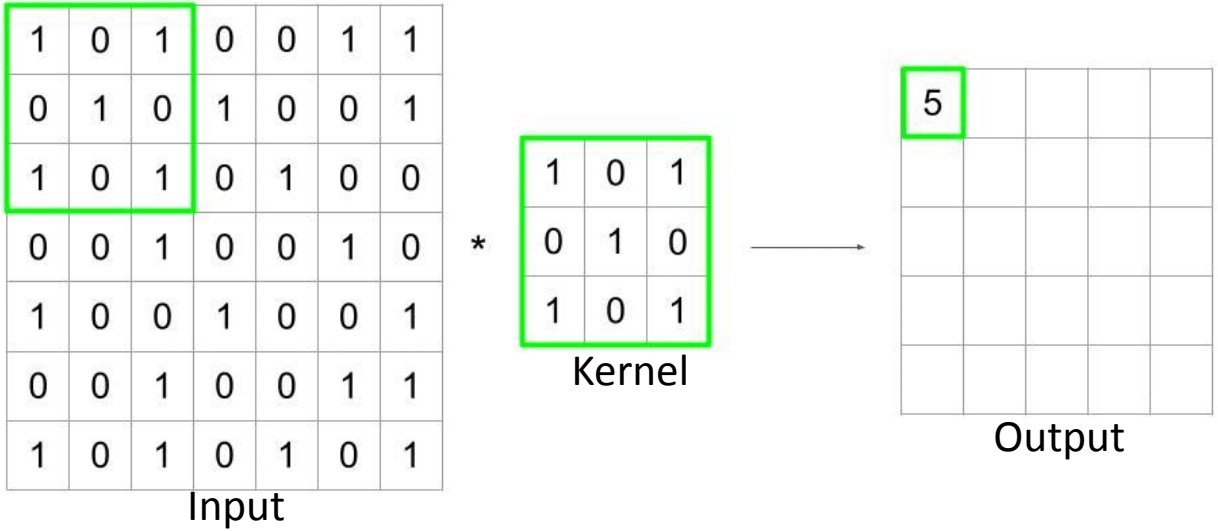
This is the Loss of the algorithm, where:

- $\overline{y}_i$ value of the i_{th} pixel of the target image
- y_i value of the i_{th} pixel of the output image

Convolutional Layer

Convolutional Neural Networks (CNNs) receive images as inputs, and compute an output based on the features of the image.

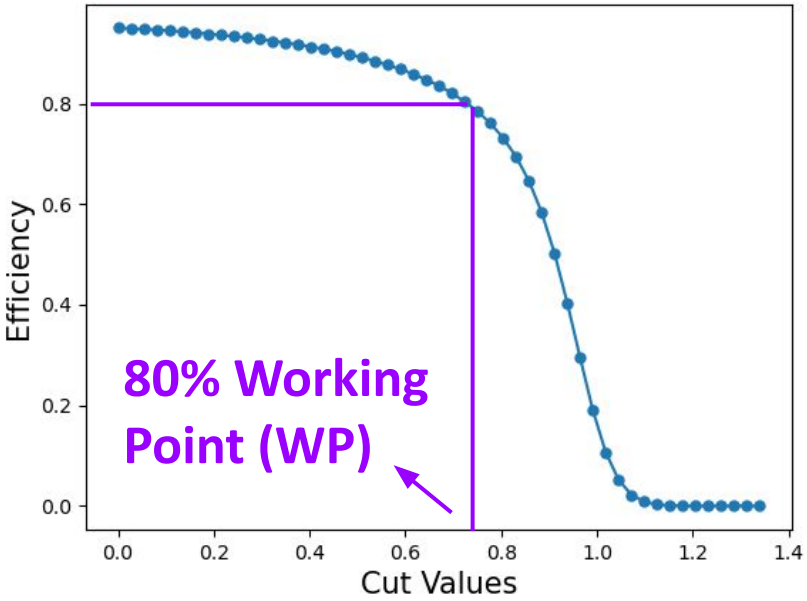
The elements in the kernel are the trainable parameters.

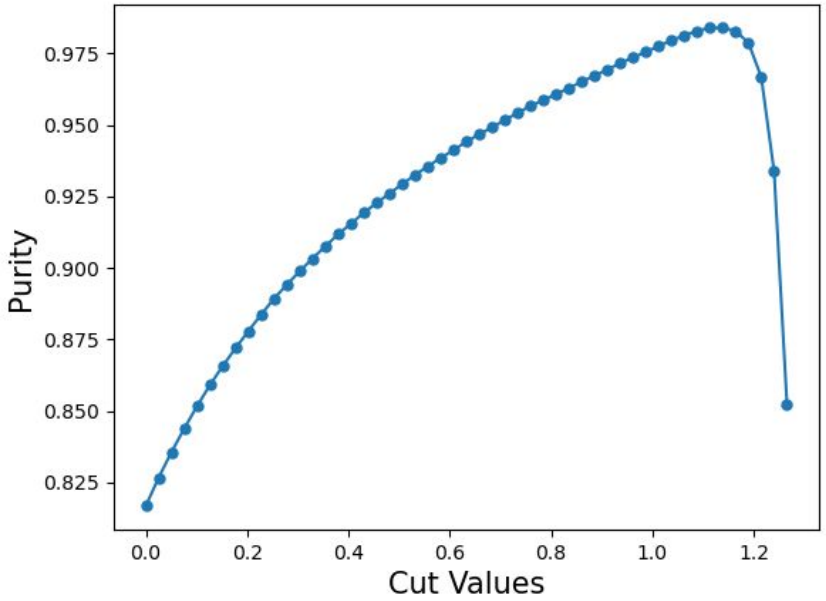


Performance of the Network

$$\text{Efficiency} = \frac{\textit{num. of Signal Hits making the cut}}{\textit{total num. of Signal Hits}}$$

$$\text{Purity} = \frac{\textit{num. of Signal Hits making the cut}}{\textit{num. of Signal and Noise Hits making the cut}}$$

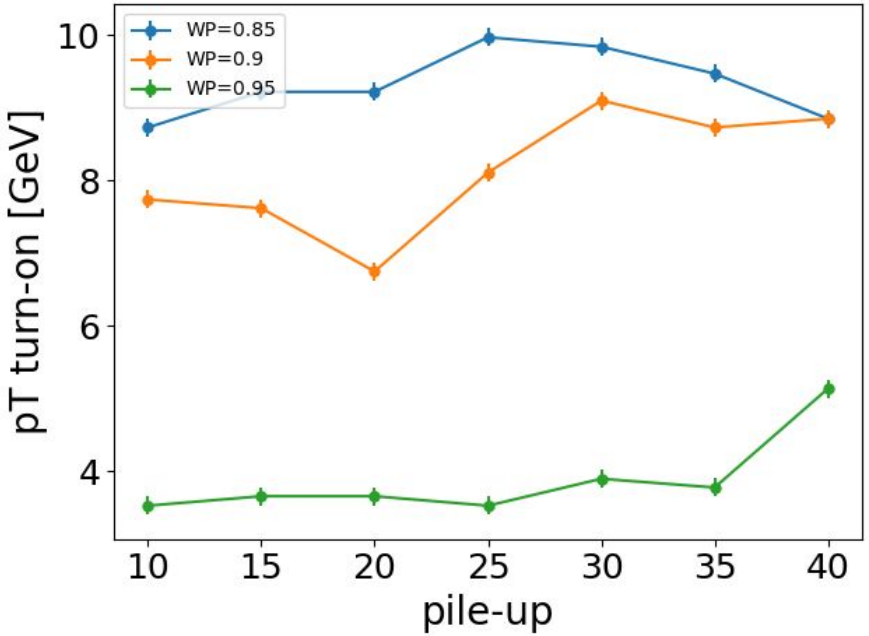




Performance vs Pile-up

I test the (64, 4) model on samples with different pile-up values.

The algorithm shows robustness since the p_T turn-on remains constant.



For $<\mu>=40$ we have:
Average number of input hits ~ 435
Average number of output hits ~ 12

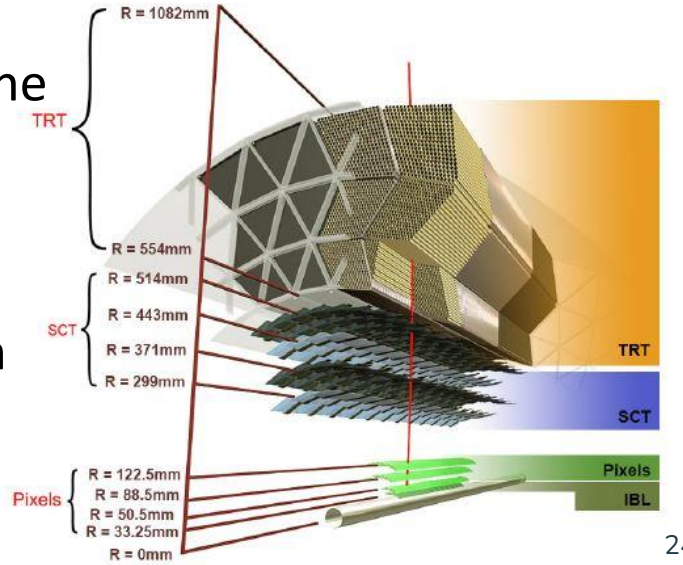
➡ Reduction of a factor ~ 36

➡ Way less combination checked by the FTF $\propto N^2$

Inner Detector (more details)

We focus on the Inner Detector (ID), the innermost system, designed to provide precise tracking information for charged particles, to achieve a high resolution for the particle momentum and the vertex position.
It is composed of:

- Pixels: 4 layers of silicon pixels registering the charge fraction left by the crossing particle.
- Semi-Conductor Tracker (SCT): 4 layers of silicon strip modules.
- Transition Radiation Tracker (TRT): a system of 300k gas-filled straws interleaved by a plastic material.

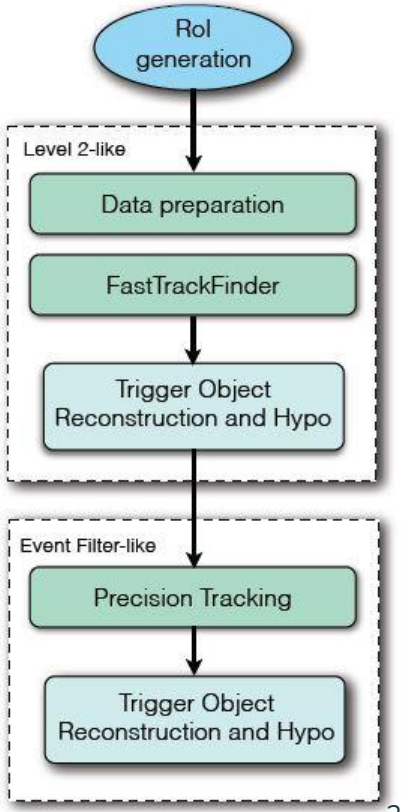


Tracking within the HLT (extended)

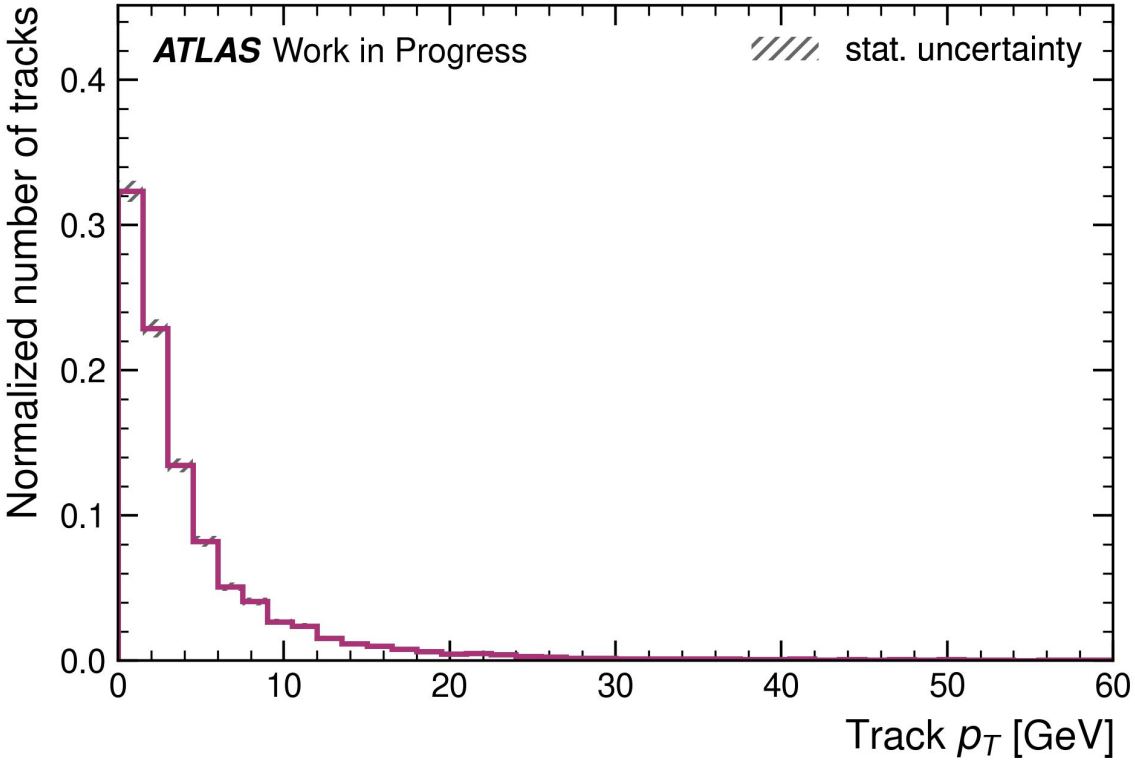
The HLT reconstructs the tracks from the ID data.
First of all the pixel hits are merged into **clusters**, from which **space-points** are computed.

Then the **Fast Track Finder** (FTF) reconstructs the tracks compatible with the space-points, using combinatorial algorithms.

Finally **Precision Tracking** selects from the FTF tracks the ones with higher quality, which are then used to look for relevant physical processes.



Tracks p_T distribution

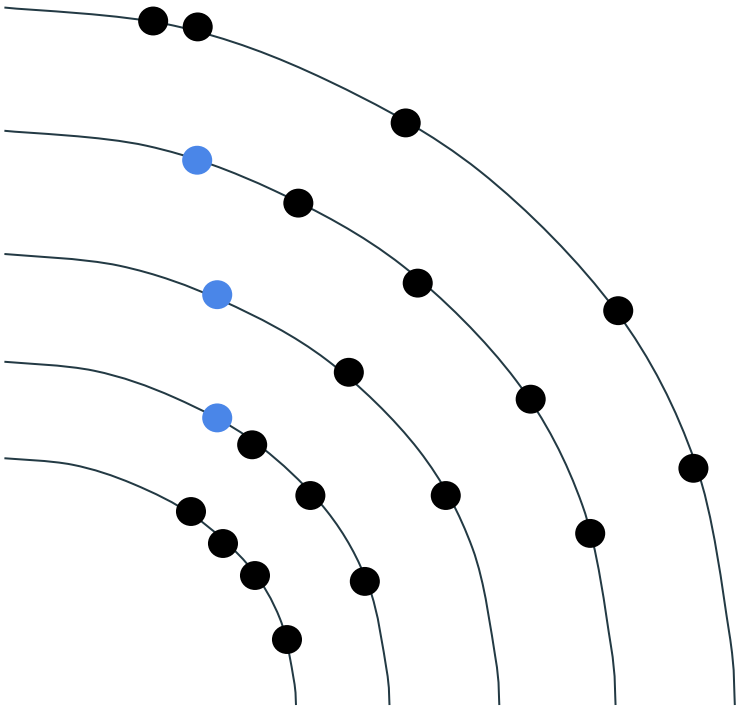


Plot obtained from a MC sample of $t\bar{t}$ jets.

Most of the tracks are in the range [0 Gev, 5 GeV]

⟶ Need to improve the performance for low p_T

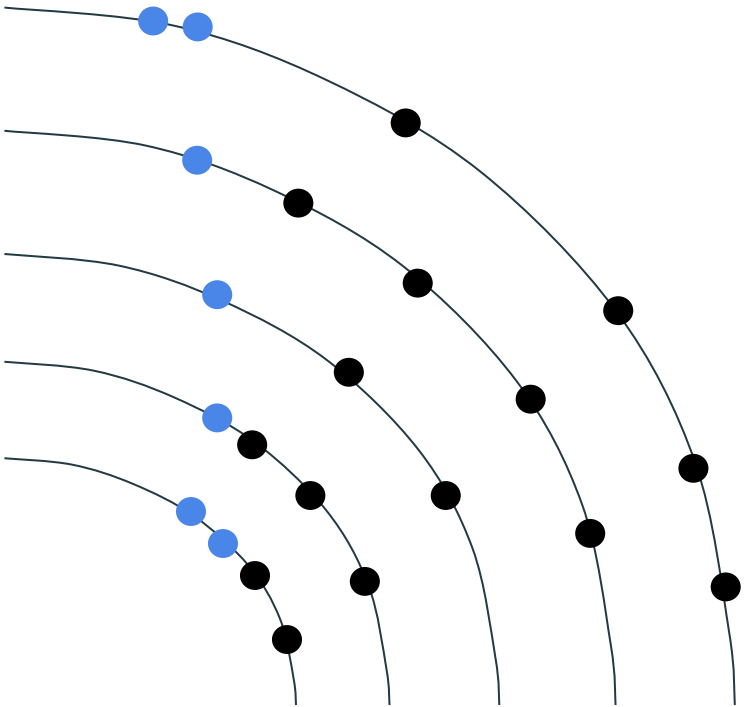
Real-time Tracking



The first stage is the Fast Track Finder (FTF):

- **Track seeding**
- Track extension
- Removal of duplicates
- Kalman filter

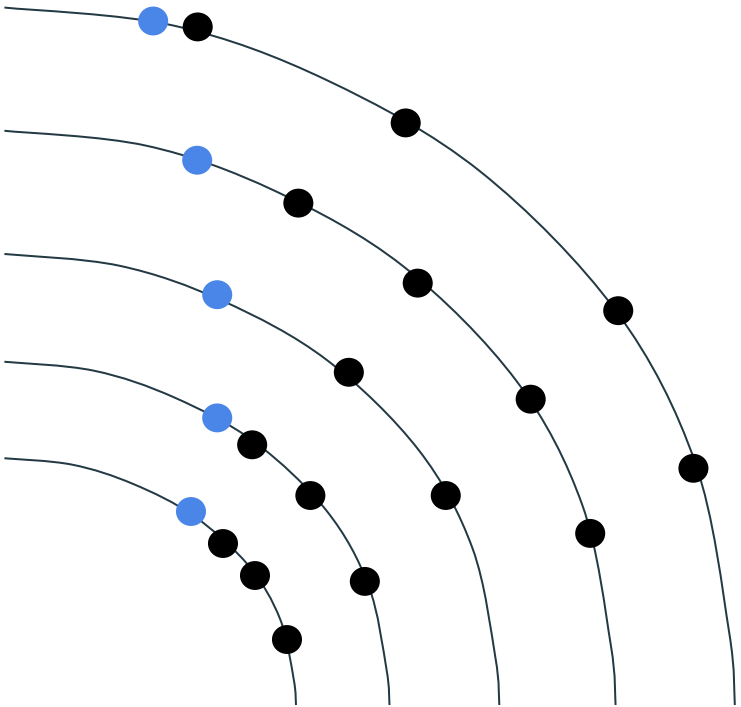
Real-time Tracking



The first stage is the Fast Track Finder (FTF):

- Track seeding
- **Track extension**
- Removal of duplicates
- Kalman filter

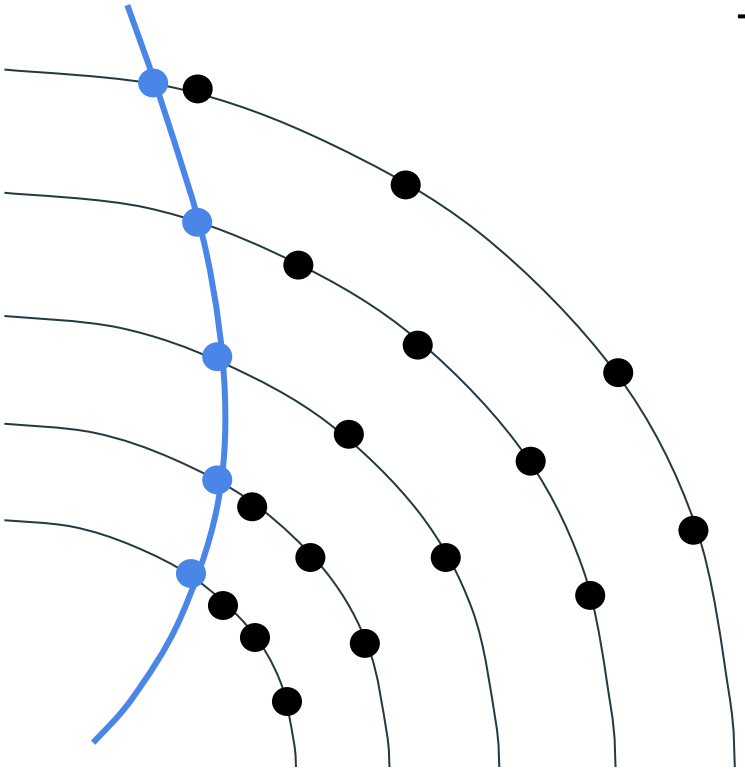
Real-time Tracking



The first stage is the Fast Track Finder (FTF):

- Track seeding
- Track extension
- **Removal of duplicates**
- Kalman filter

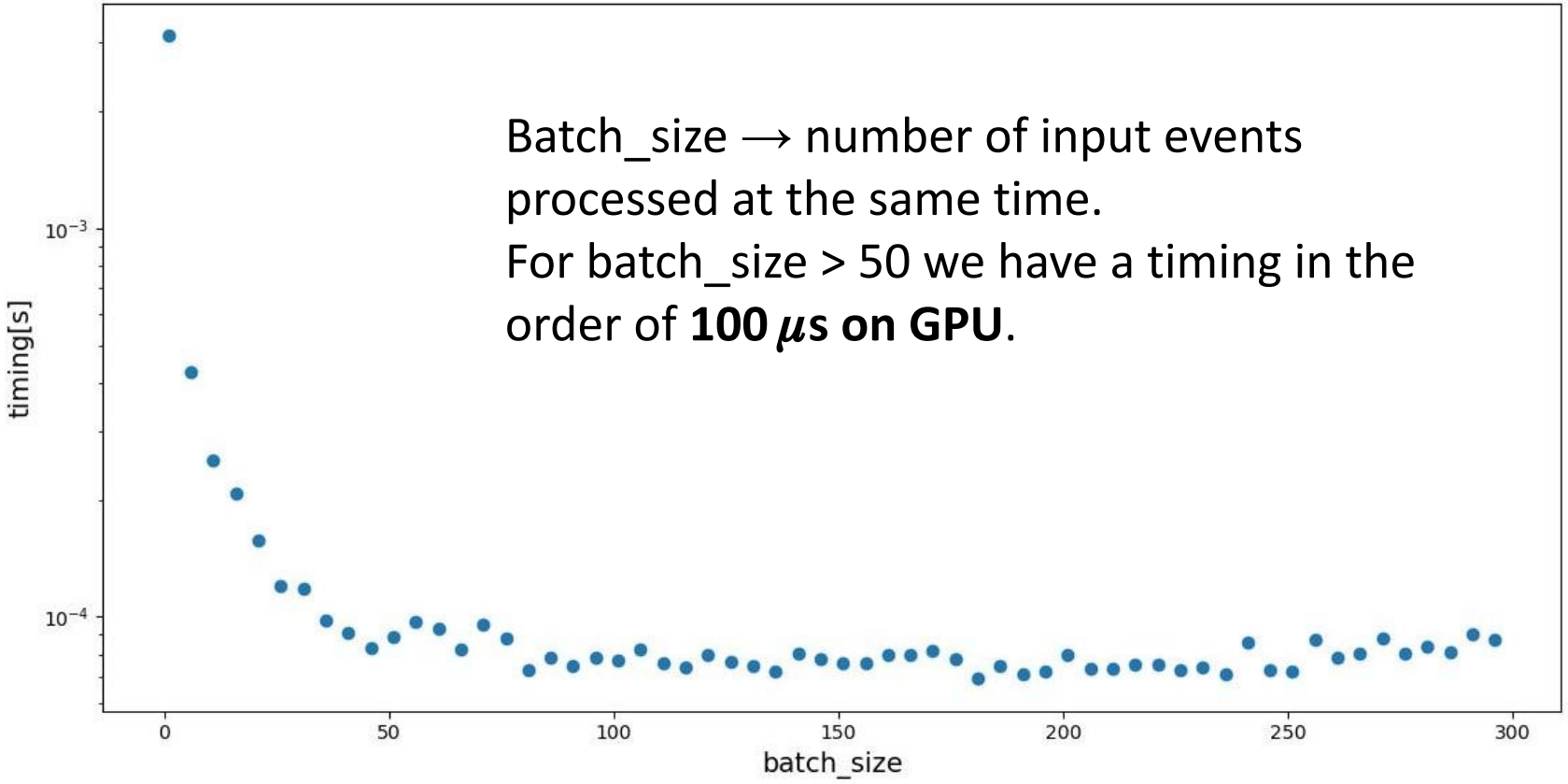
Real-time Tracking



The first stage is the Fast Track Finder (FTF):

- Track seeding
- Track extension
- Removal of duplicates
- **Kalman filter**

Timing of the Algorithm



Batch_size → number of input events processed at the same time.
For batch_size > 50 we have a timing in the order of **100 μs on GPU**.