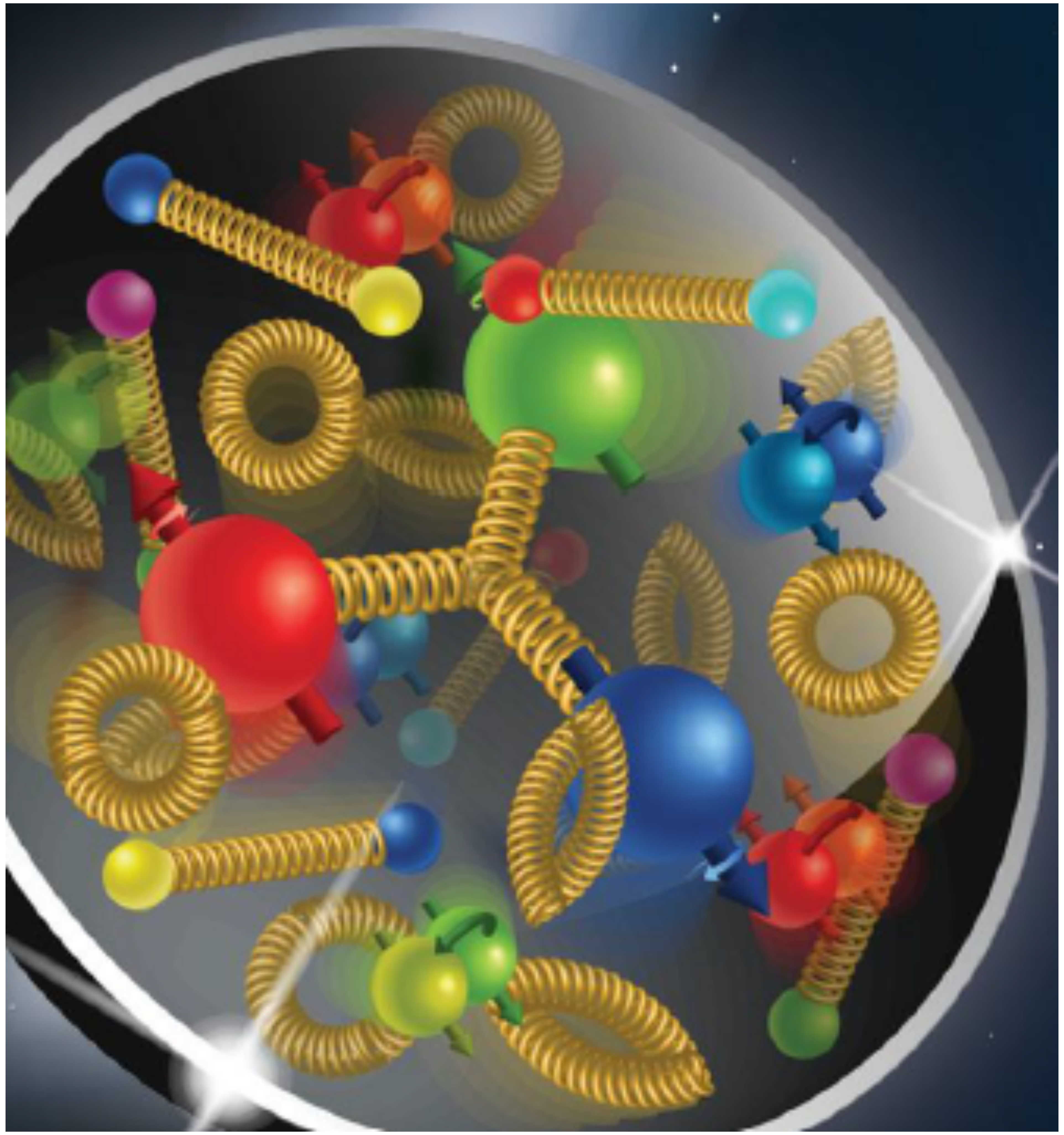
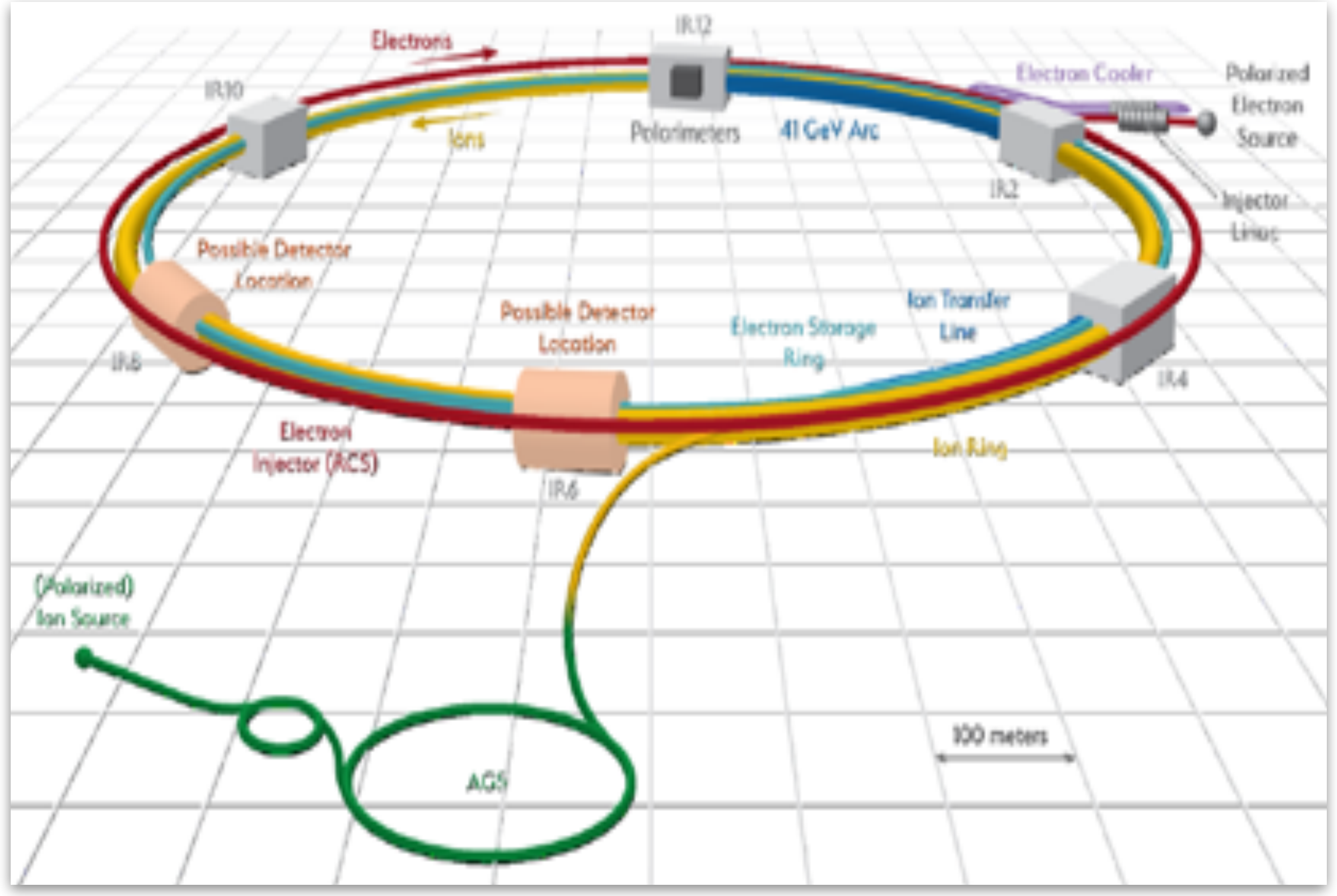
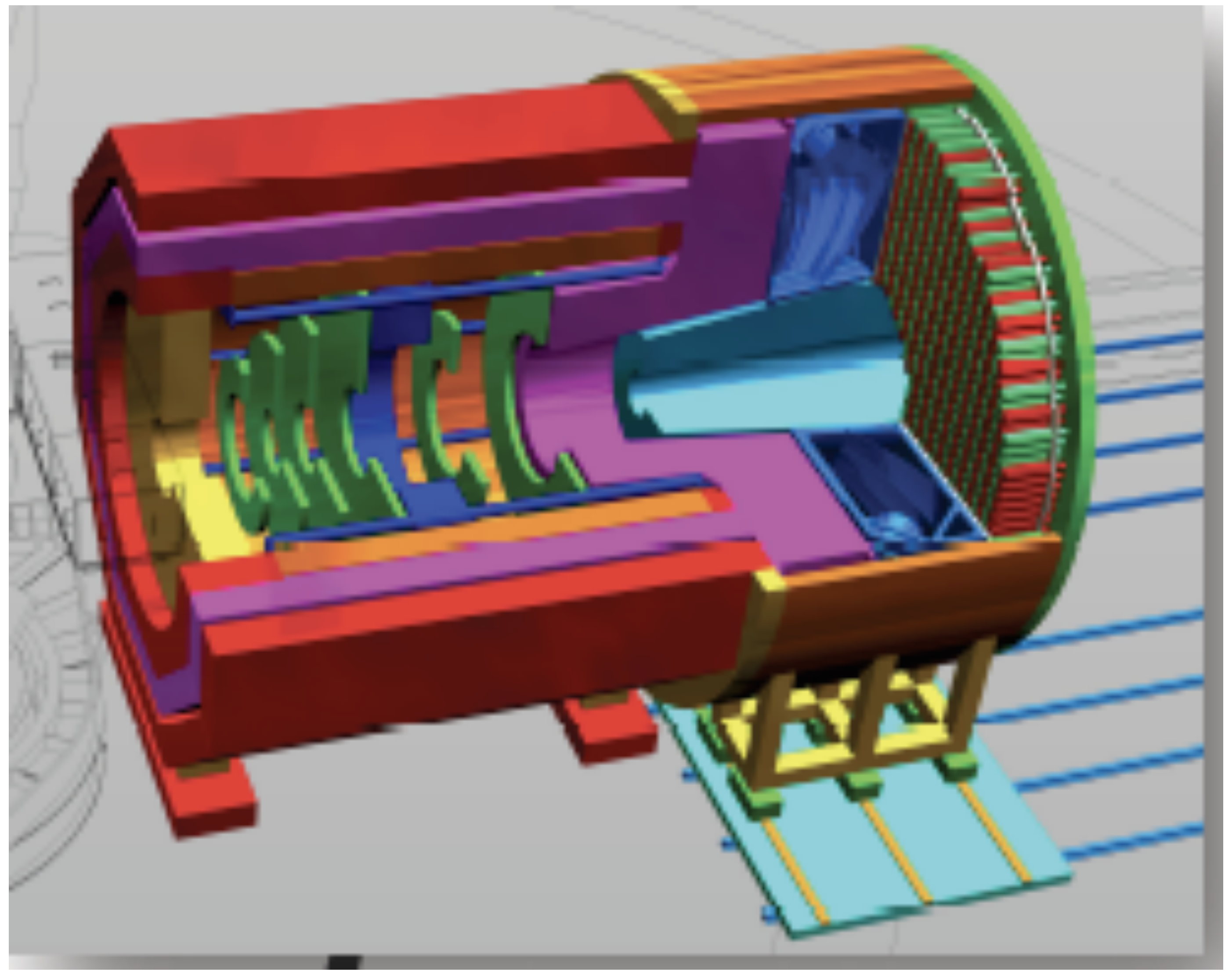






Fifth ML-INFN Hackathon: Advanced Level

Nov 13 – 16, 2023
INFN Pisa
Europe/Rome timezone

AI in streaming readout data acquisition and real-time inference

M.Battaglieri (INFN), F.Rossi (INFN)

Outline

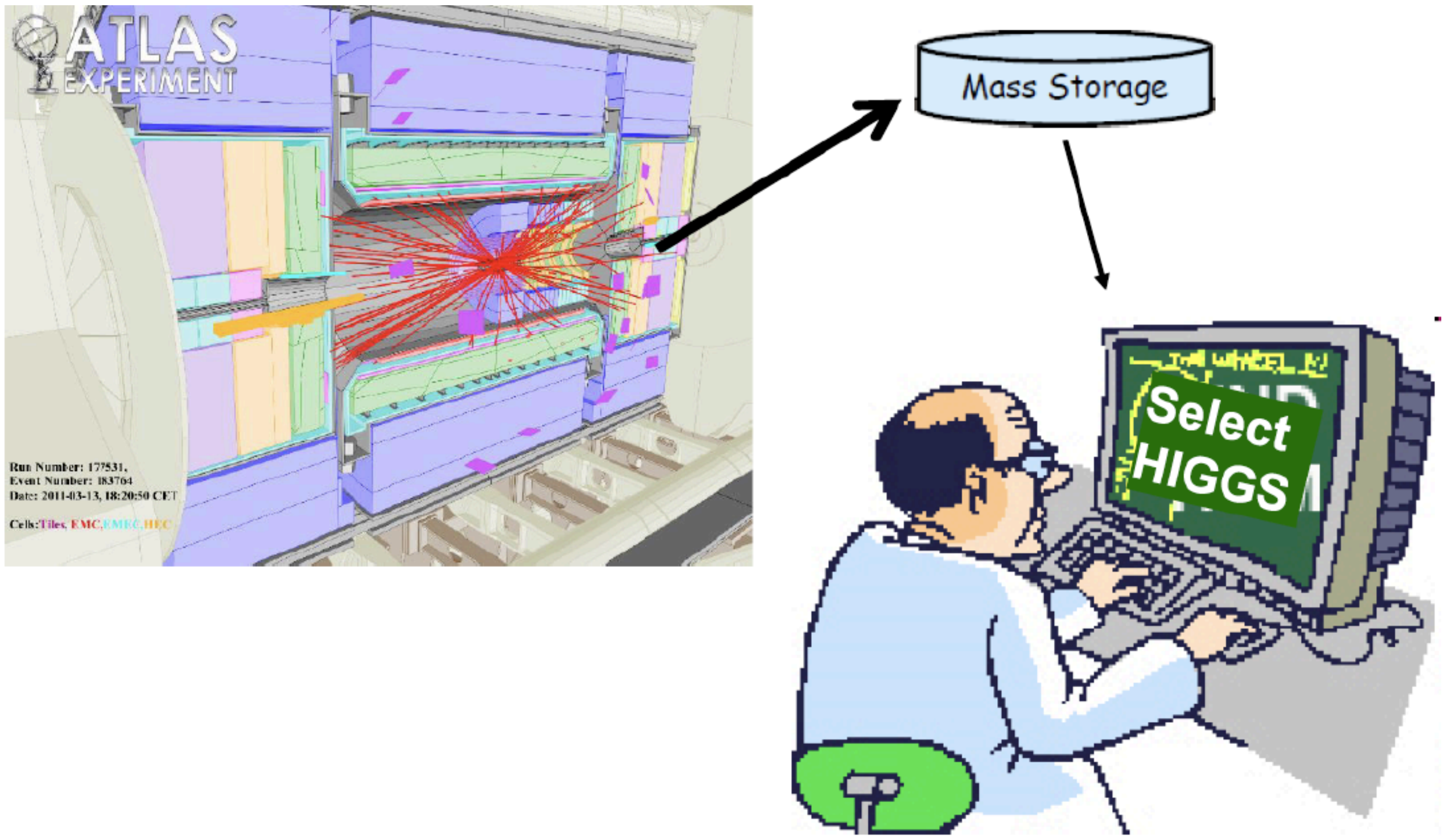
Part I (MarcoB)

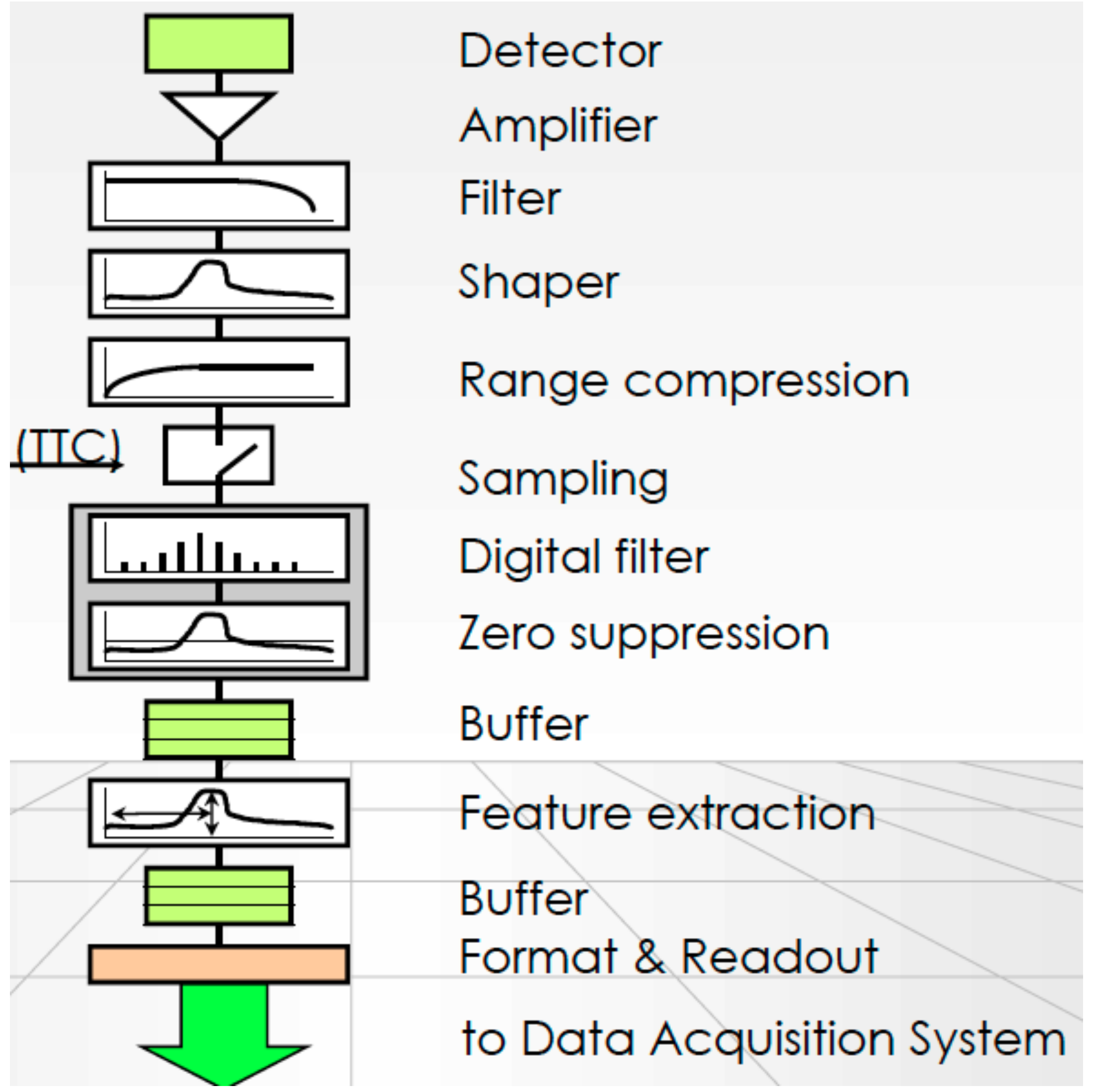
- DAQ and streaming readout: triggered vs untriggered
- SRO requirements and opportunities
- An example: (future) ePIC@EIC (BNL) SRO scheme
- AI in real-time data analysis
- Partial realtime data reconstruction (clustering)
- Fast inference
- Data reduction

Part II (FabioR)

- Application to data reduction

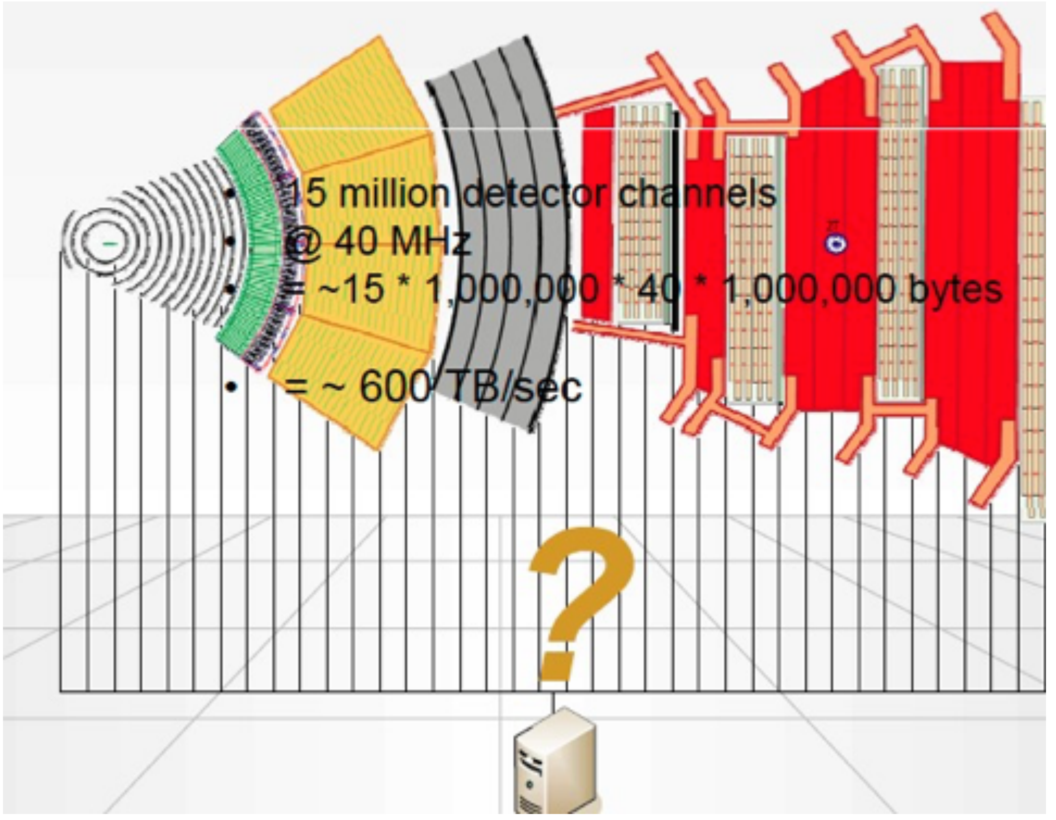
From signals to physics



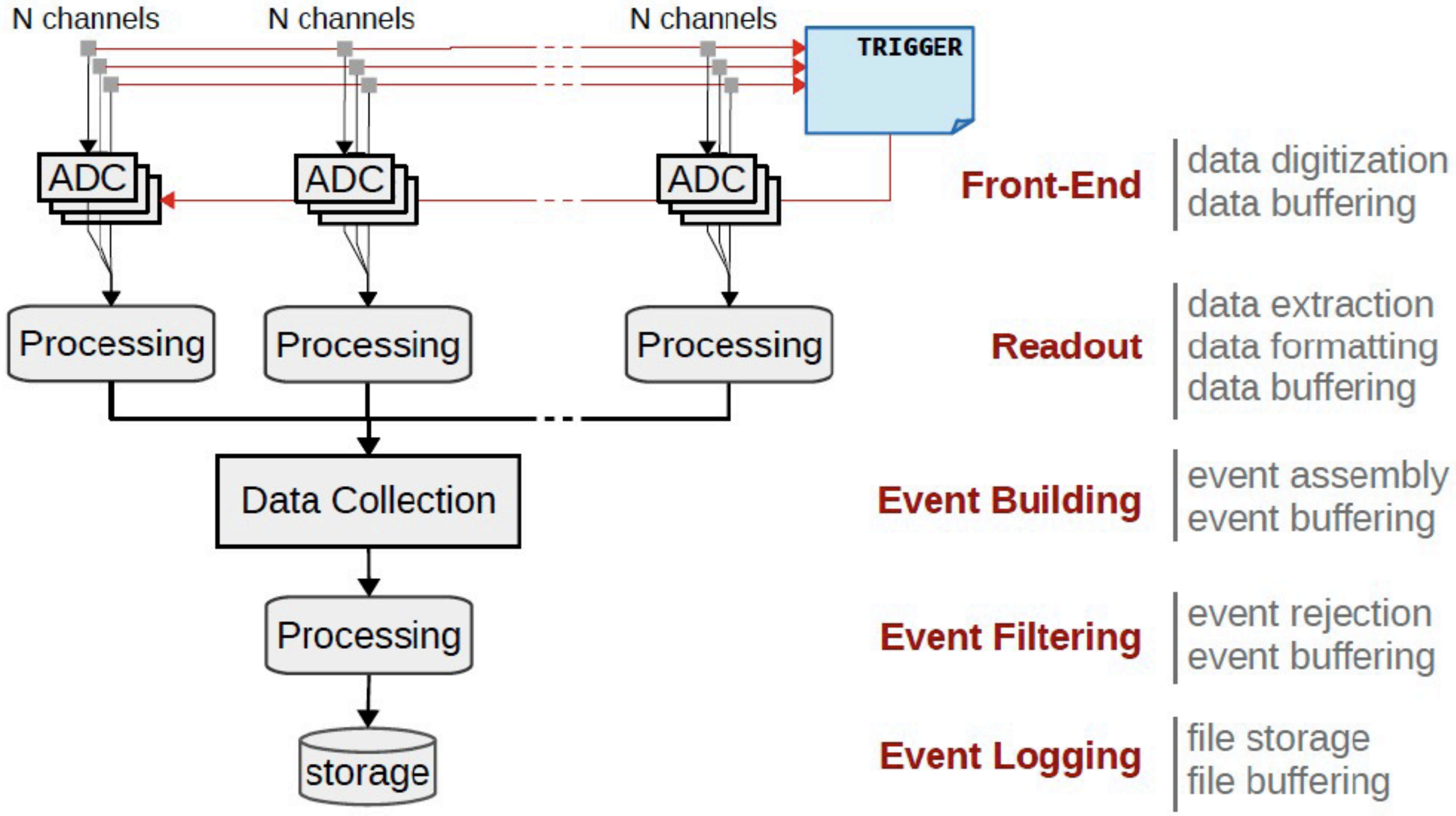


DAQ chain

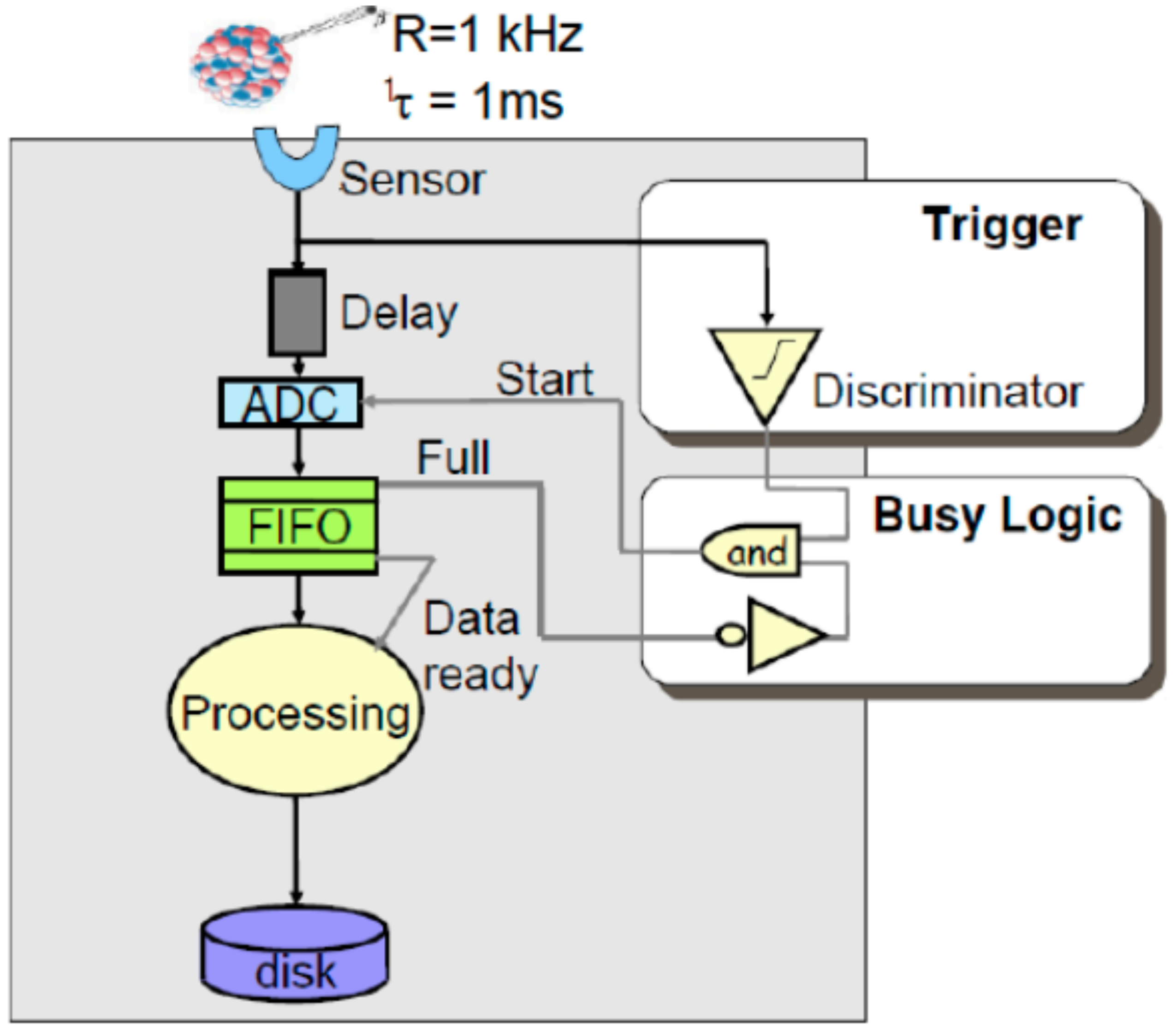
CMS@LHC



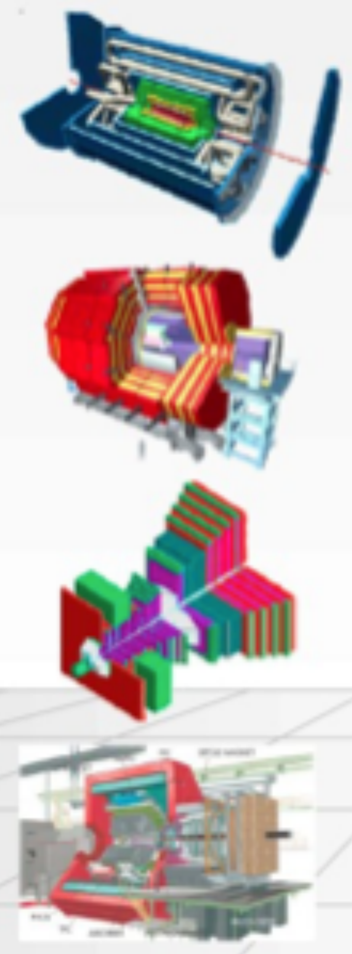
- O(10^7) canali
- Word-size = 1-14 bit
- Rate (bunch crossing) ~40 MHz (1/25ns)
- Rate = 600TB/s (!!!!)



Triggered DAQ



LHC Experiments DAQ

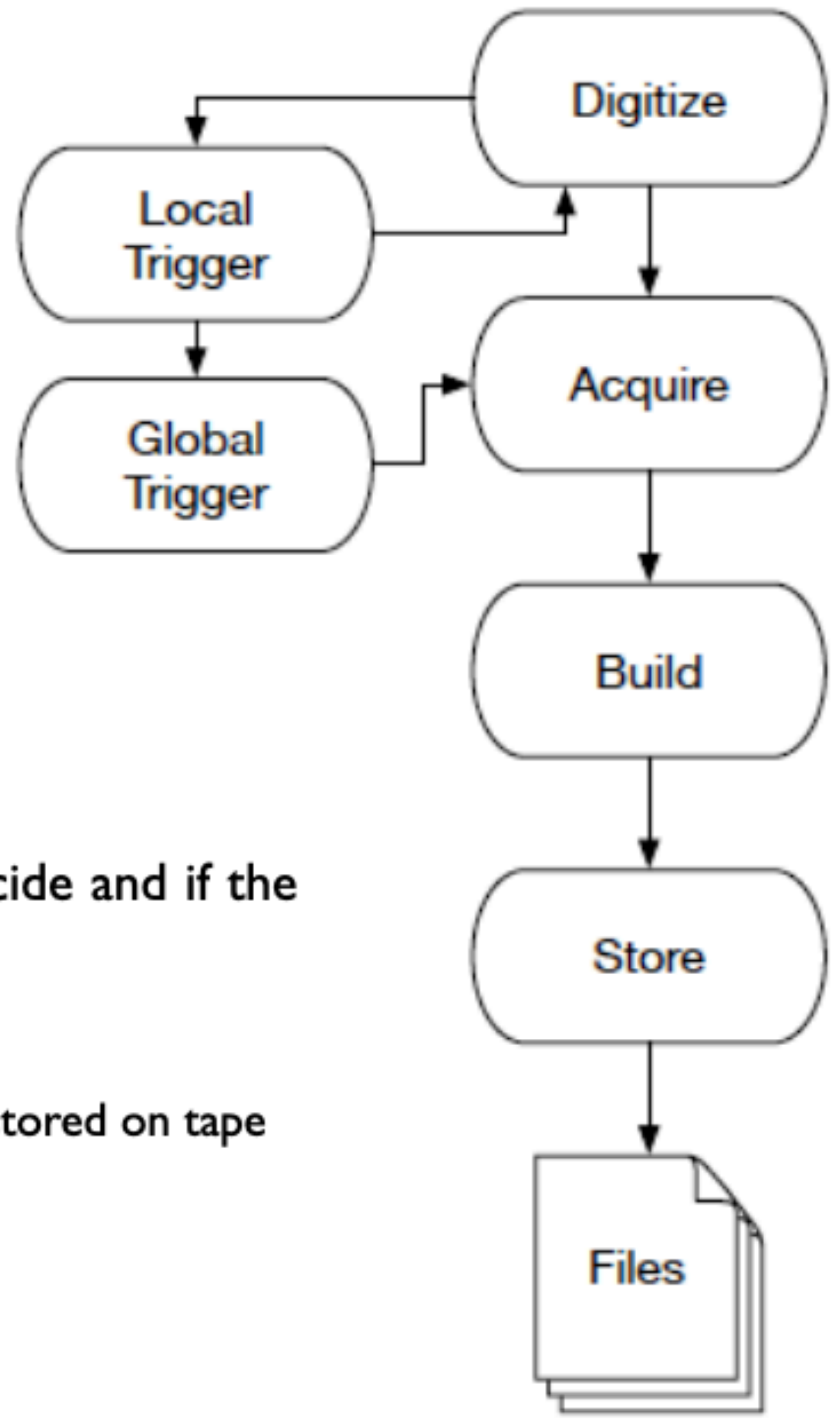


	Level-1 kHz	Event MByte	Storage MByte/s
ATLAS	100	1	100
CMS	100	1	100
LHCb	1000	0.04	80
ALICE	1	25	1250

Traditional (triggered) DAQ

Traditional triggered

* (few) trigger Channels participating send (partial) information to trigger logic



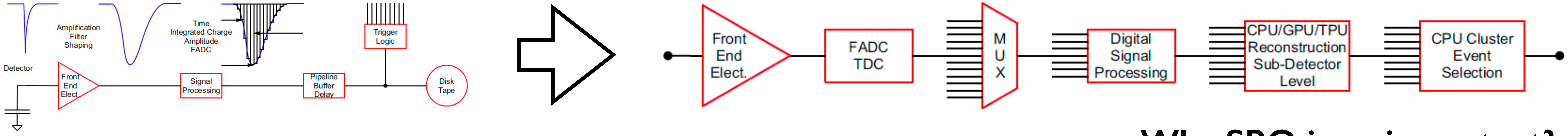
* All channels continuously measured, hits stored in short term memory

* Trigger logic takes time to decide and if the trigger condition is satisfied:
    - a new 'event' is defined
    - trigger signal back to the FEE
    - data read from memory and stored on tape

Traditional triggered DAQ

- **Pros**
    - we know it works reliably!
- **Drawbacks:**
    - only few information forms the trigger
    - Trigger logic (FPGA) difficult to implement and debug
    - not easy to change and adapt to different conditions

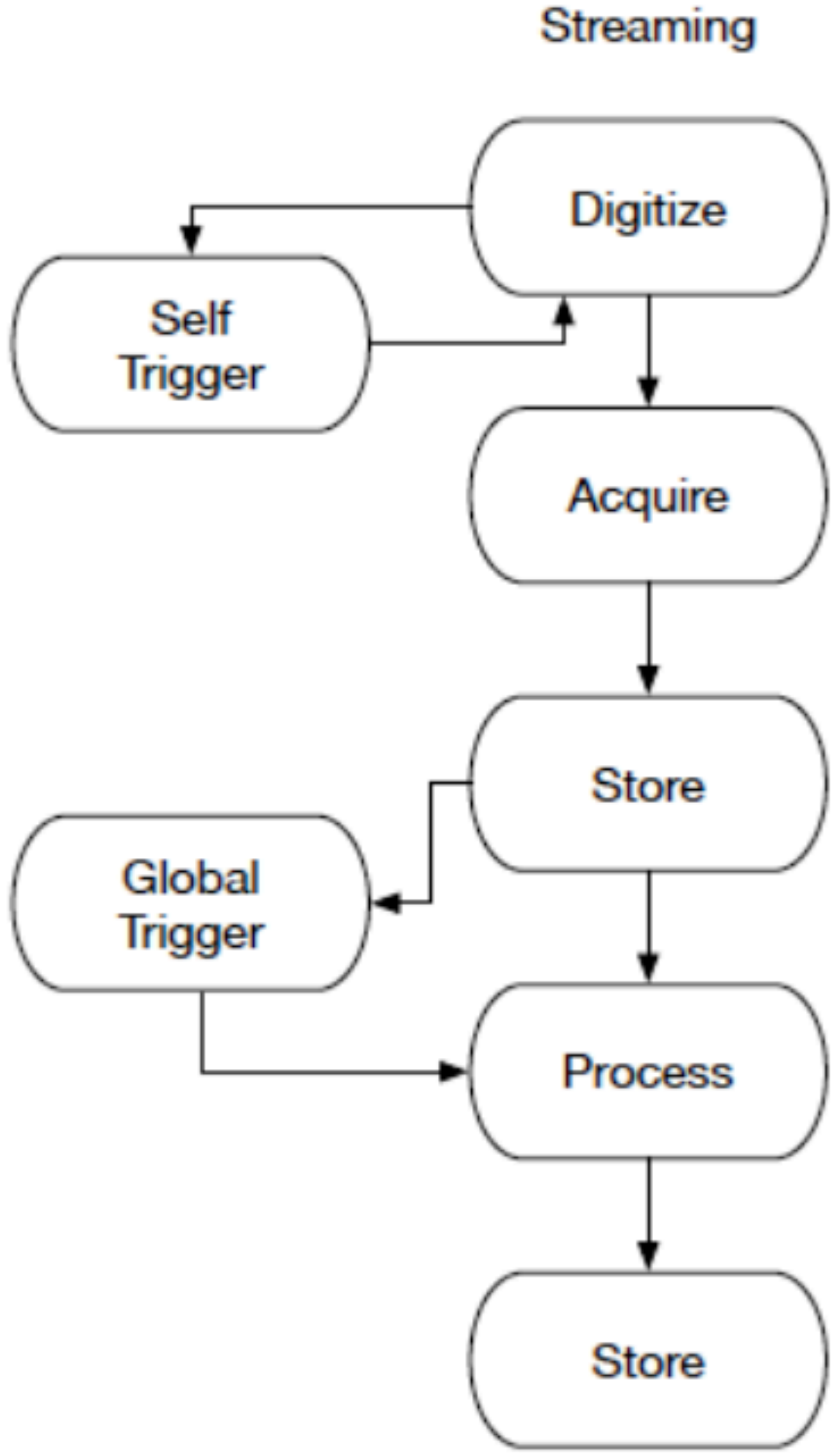
Streaming RO



Streaming read out (SRO)

* A HIT MANAGER receives hits from FEE, order them and ship to the software defined trigger

* Software defined trigger re-aligns in time the whole detector hits applying a selection algorithm to the time-slice
  - the concept of 'event' is lost
  - time-stamp is provided by a synchronous common clock distributed to each FEE



* All channels continuously measured and hits streamed to a HIT manager (minimal local processing) with a time-stamp

SRO DAQ

- **Pros**
  - All channels can be part of the trigger
  - Sophisticated tagging/filtering algorithms
  - high-level programming languages
  - scalability
- **Drawbacks:**
  - we do not have the same experience as for TRIGGERED DAQ

Why SRO is so important?

* **High luminosity experiments**
  - Write out the full DAQ bandwidth
  - Reduce stored data size in a smart way (reducing time for off-line processing)

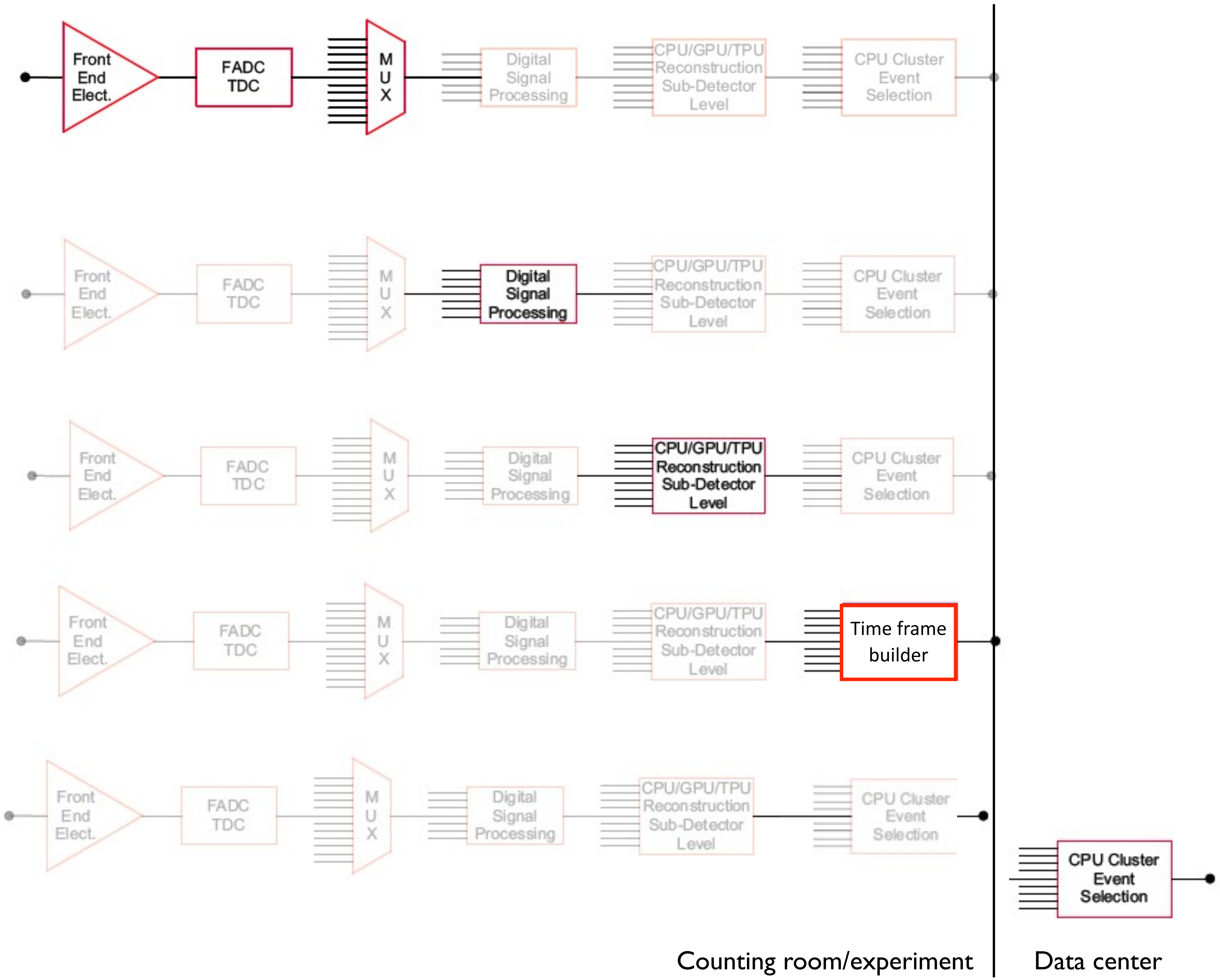
* **Shifting data tagging/filtering from the front-end (hw) to the back-end (sw)**
  - Optimize real-time rare/exclusive channel selection
  - Use of high-level programming languages
  - Use of existing/ad-hoc CPU/GPU farms
  - Use of available AI/ML tools
  - (future) use of quantum-computing

* **Scaling**
  - Easier to add new detectors in the DAQ pipeline
  - Easier to scale
  - Easier to upgrade

Many NP and HEP experiments adopt a SRO DAQ

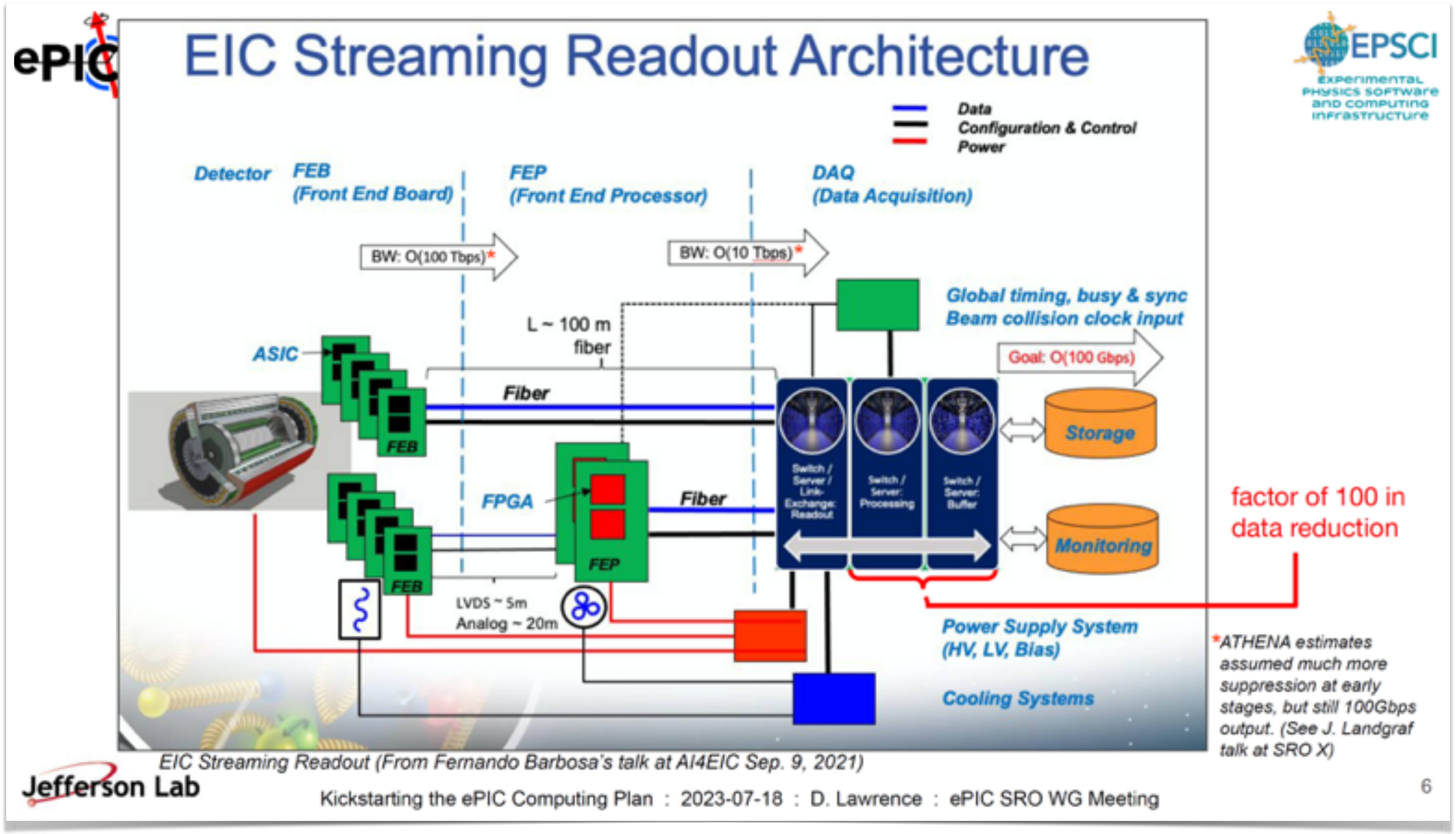
- CERN: LHCb, ALICE, AMBER
- FAIR: CBM
- DESY: TPEX
- FRIBS: GRETA
- BNL: sPHENIX
- JLAB: SOLID, BDX, CLAS12, …

Streaming RO



- FEE optimised for SRO
  - ASICS (cheap) or fADC (multiplexing) at (O($10/ch)
  - TDC if necessary to replace fADC
  - Zero-suppression mode
  - Fast readout (optical link)
- Signal pre-processing with fast hw (dedicated FPGA)
  - de-multiplexing fADC info
  - Charge, time, amplitude
  - Data compression
  - Data monitoring
  - Add other information (e.g. ch_ID eTimeStamp)
- CPU/GPU/TPU sub-detector analysis (single stream)
  - Local clusters, track segments, PID, …
  - Time-frame building
  - If necessary only store high-level data dumping raw
- TF-Router Time frame construction
  - Use time stamps to reorganise data from all streams in time frames
- Full reconstruction CPU analysis (for each time frame)

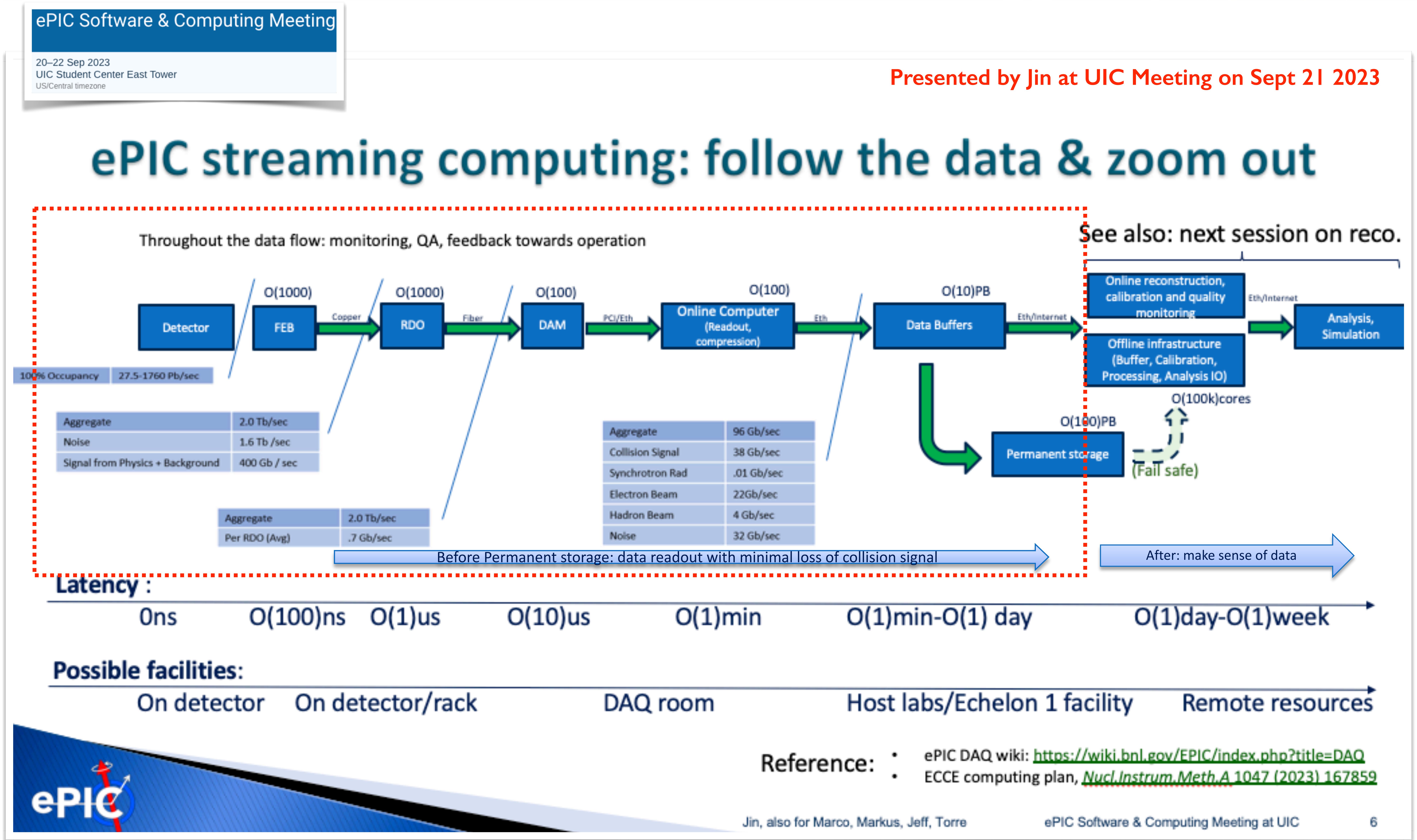
ePIC Streaming Computing



Streaming RO for ePICS

- Full consensus for SRO within the EIC community (Yellow Paper, DAQ models in ECCE, ATHENA, …)
- Rates at ePICS are not comparable to LHC HI-LUMI but advantages of SRO remain:
  - multiple channels to trigger on
  - Holy Grail: to manage (storage) an unbiased (un-triggered) data set for further analysis
  - on/off-line event selection with full detector information

ePIC Streaming Computing



Interfaces

- Each step in the workflow has a different latency
- Identify interfaces for a 'service-oriented' approach

Within the 'control room'

- Each stage in data flow requires IO specs (based on CPU, GPU, FPGA reduction)
- 'control room' boundary based on permanent data storage

Outside the control room

- Networking
- CPU/GPU farm
- Local/remote resources
- on/off-line analysis

AI-supported algorithms for SRO

Real Time data analysis

- In the SRO scheme, data analysis is performed online [this does not prevent to save unbiased frames for further analysis!]
- A św trigger is released based on real-time data analysis
- SRO and real-time data processing NEED AI to adapt data analysis to the changed conditions of the run (e.g. thresholds)
- Identify data features in real-time (e.g.clusters)
- Use a data subset to extract calibration constants
- Define algorithms to run (fast!) in real time on heterogeneous systems (e.g. CPU+GPU+FPGA)

Partial Real-Time data reconstruction: clustering

- Look at all detector information (hit: x, y, t, E) to learn correlations: clusters of objects share common features
  - Define a metric in a space and identify cluster features
- Tests on minimum bias trigger data before real-time
- Hyperparameters optimization based on data

Fast inference

- Fast algorithms to extract data features to be used in data selections (and reduction)
- Mimicking a smart 'trigger'
- provide partial reconstructed quantity quickly

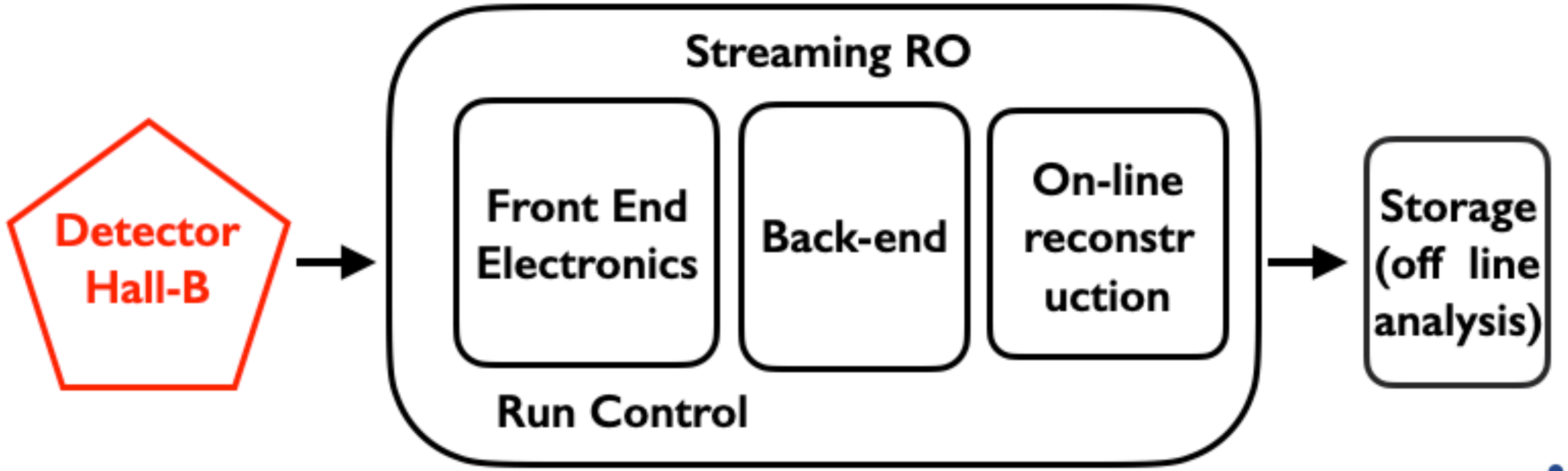
Calibration

- Use smart algorithms to extract data features and correct detector parameters varying over time
- toward a self-calibrating detector

Data reduction

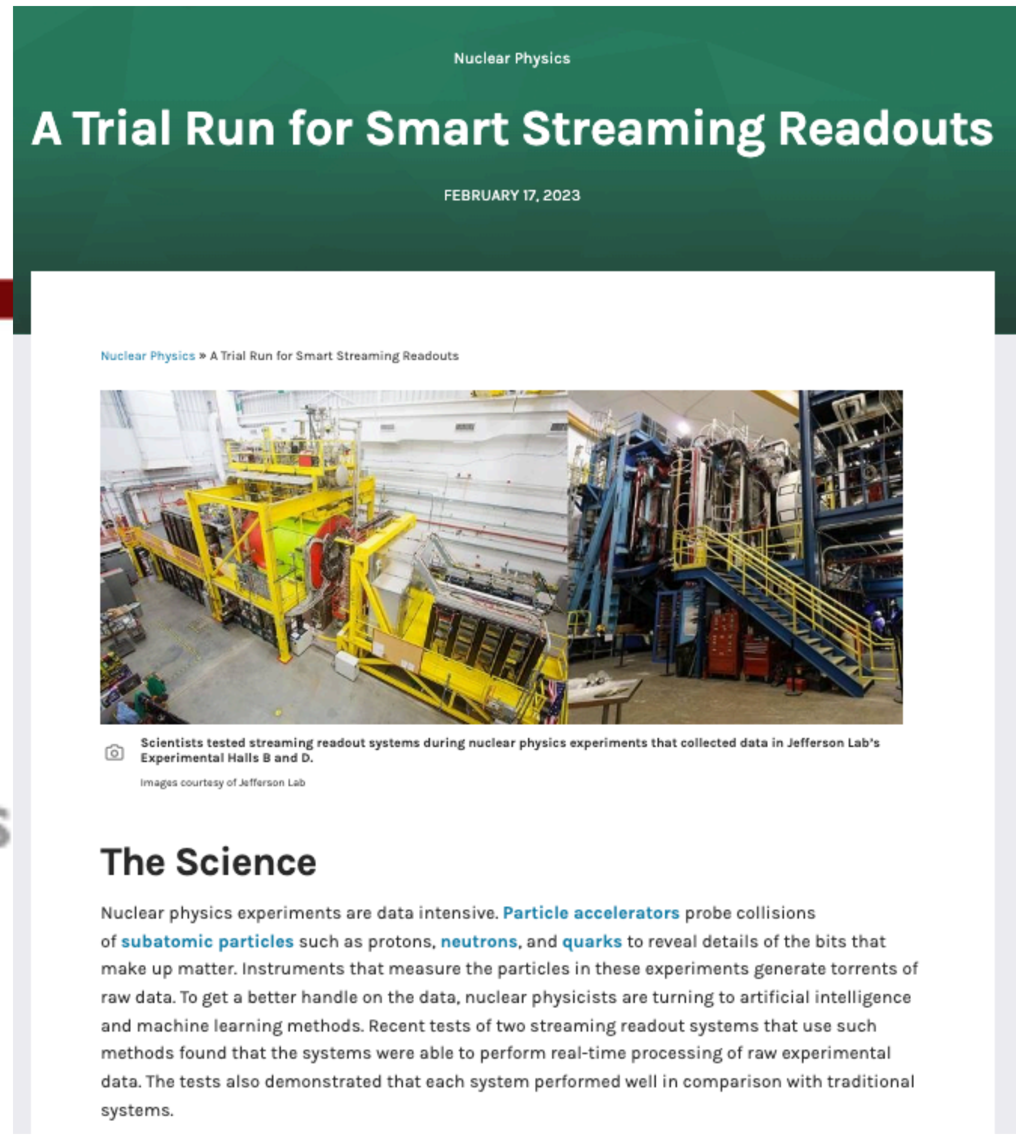
- reduce data volume to a manageable level with minimum bias

Streaming RO tests



SRO concept validation

1) Assemble SRO components
2) Test SRO DAQ in lab
3) Test SRO DAQ on-beam


Nuclear Physics

A Trial Run for Smart Streaming Readouts

FEBRUARY 17, 2023

Nuclear Physics » A Trial Run for Smart Streaming Readouts

Scientists tested streaming readout systems during nuclear physics experiments that collected data in Jefferson Lab's Experimental Halls B and D.

Images courtesy of Jefferson Lab

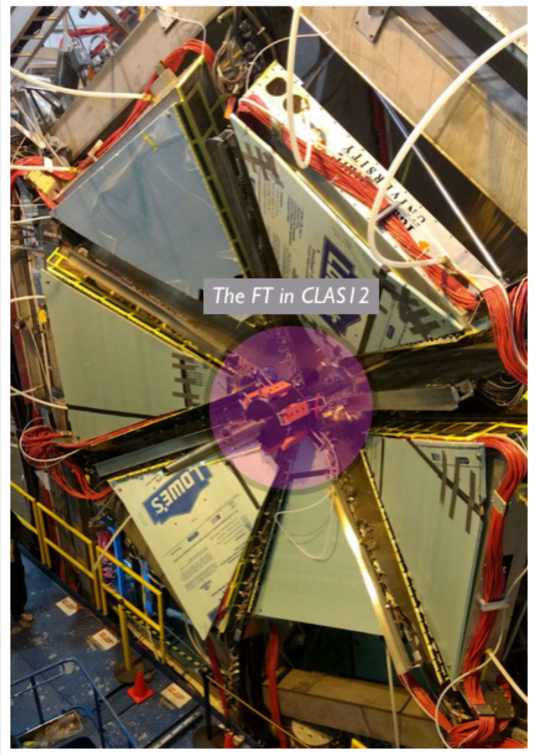
The Science

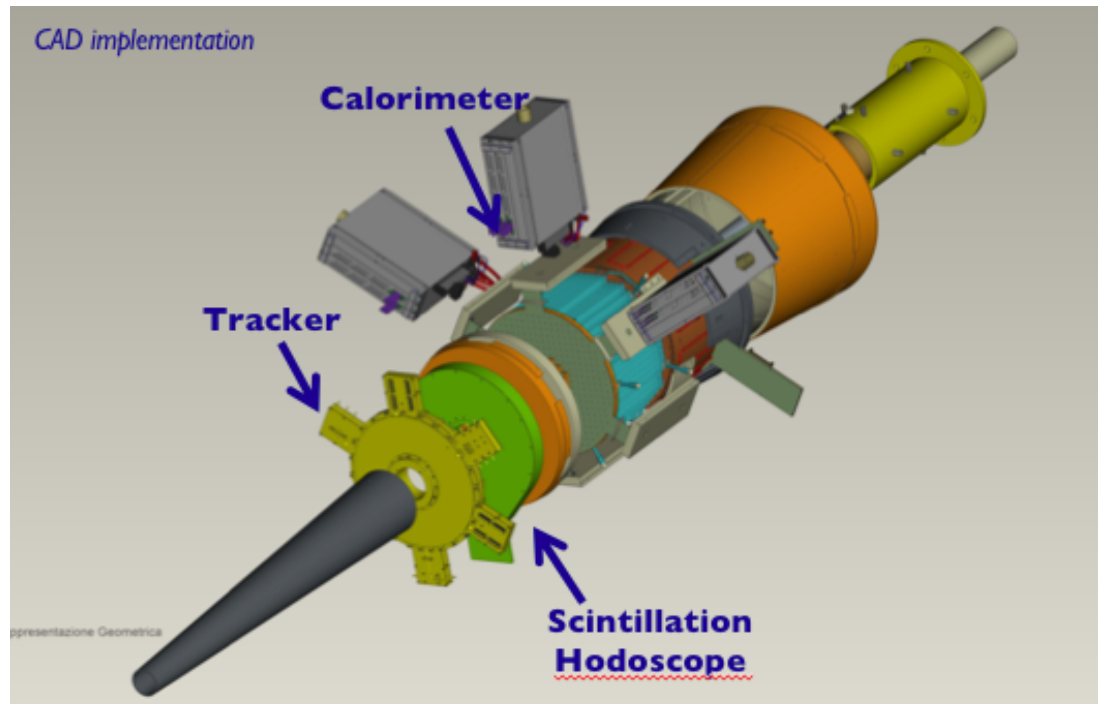
Nuclear physics experiments are data intensive. Particle accelerators probe collisions of subatomic particles such as protons, neutrons, and quarks to reveal details of the bits that make up matter. Instruments that measure the particles in these experiments generate torrents of raw data. To get a better handle on the data, nuclear physicists are turning to artificial intelligence and machine learning methods. Recent tests of two streaming readout systems that use such methods found that the systems were able to perform real-time processing of raw experimental data. The tests also demonstrated that each system performed well in comparison with traditional systems.

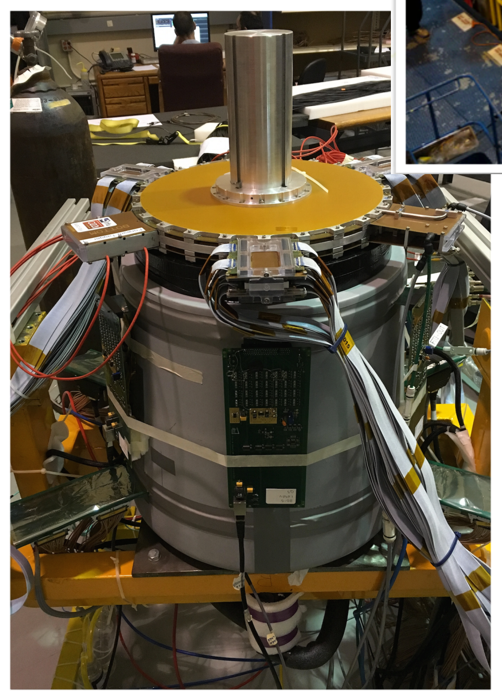
JLab SRO validation

✳ CLAS12 Forward Tagger

- Complete system that include calorimetry, PiD, Traking in a simpler (than CLAS12) set up
- FT-ECAL: 332 PbWO crystals, APD readout
- FT-HODO: 224 plastic scintillator tiles, SiPM readout
- FT-TRK: ~3000 channels, MicroMegas
- fADC250 digitizers + DREAMs for MM



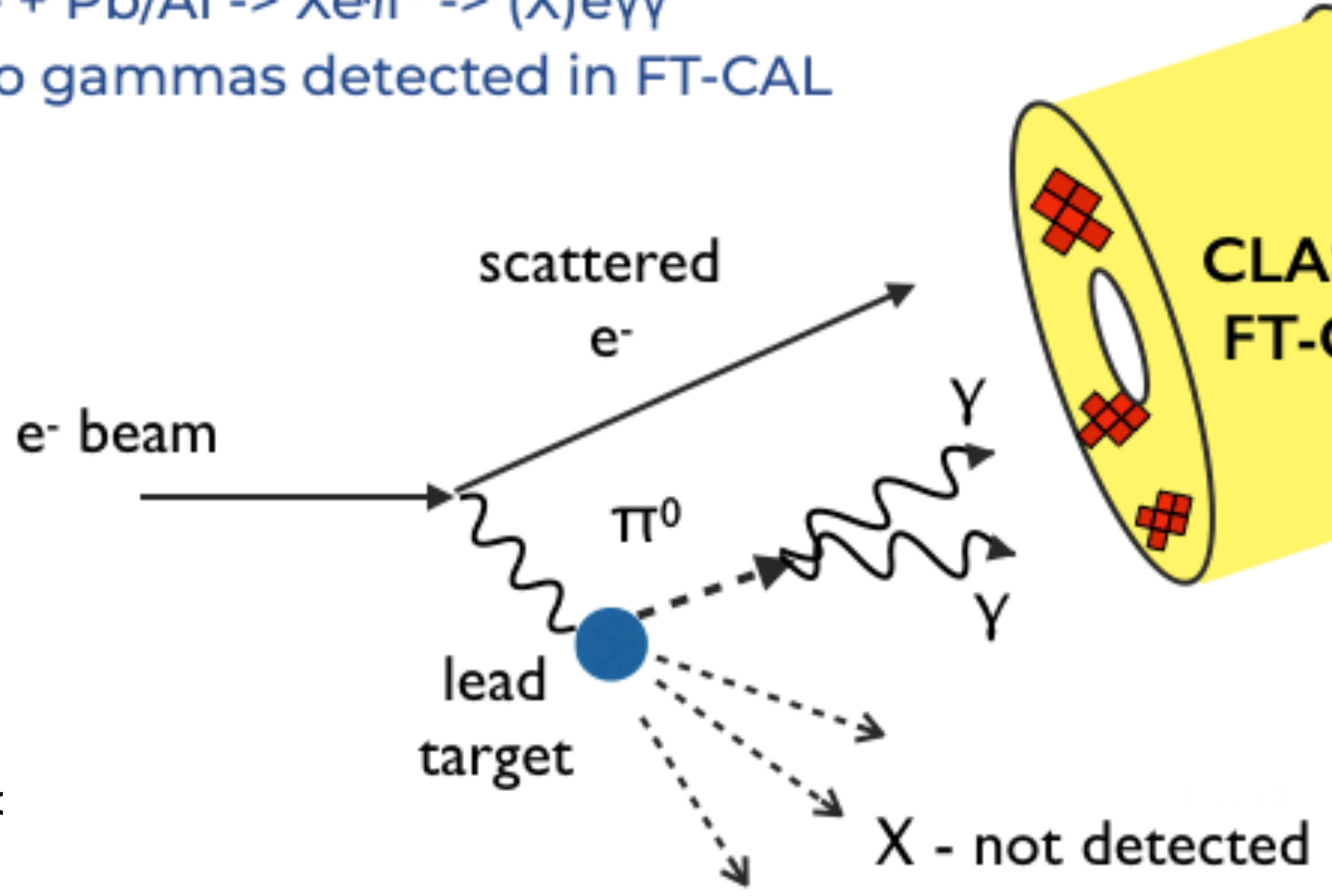




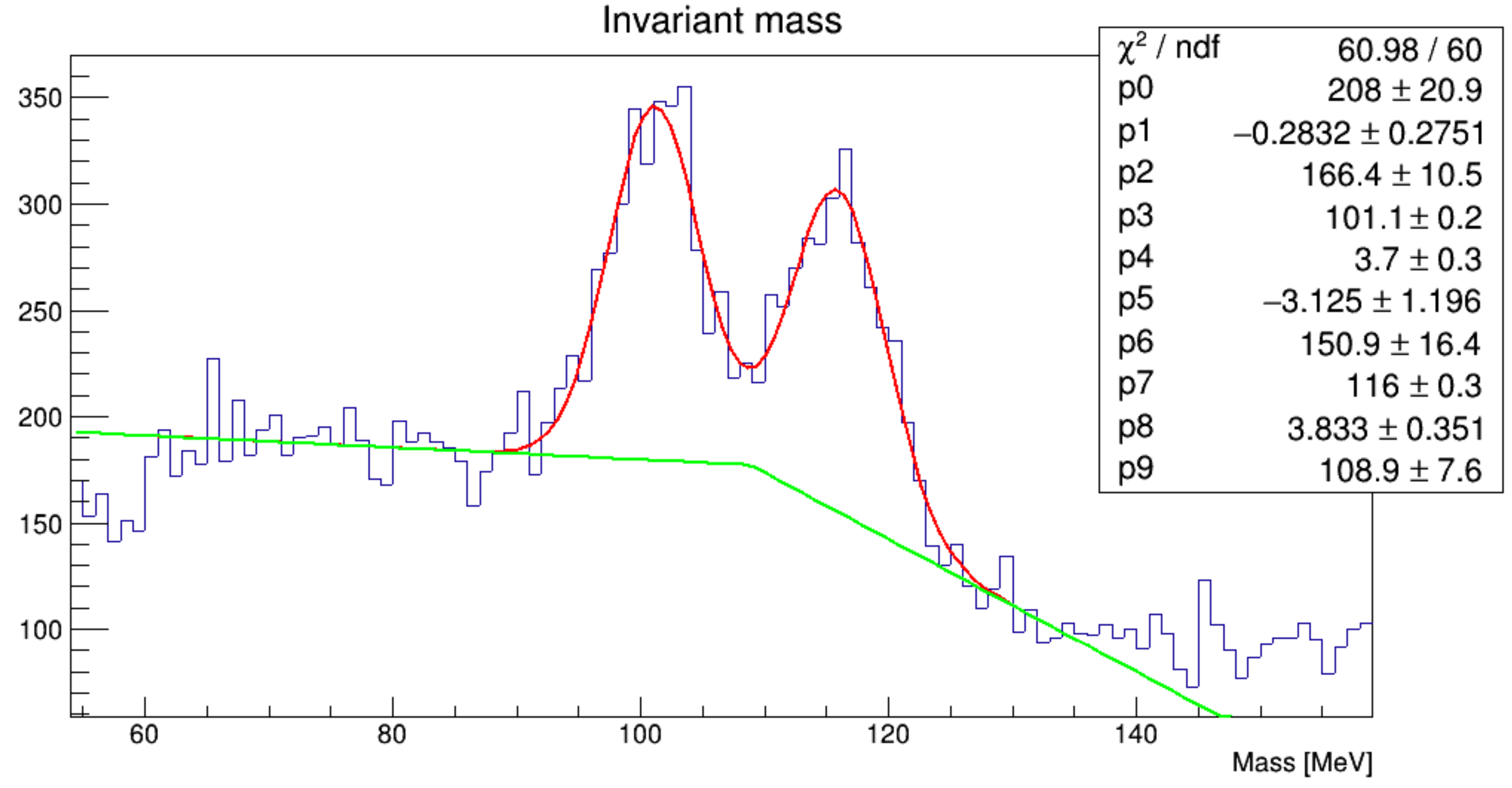
✳ CLAS12 Forward Tagger

- Inclusive pi0 electroproduction
- Two gammas detected into FT-CAL
- EM clusters identification, anti coincidence with FT-Hc
- Self-calibration reaction (pi0 mass)

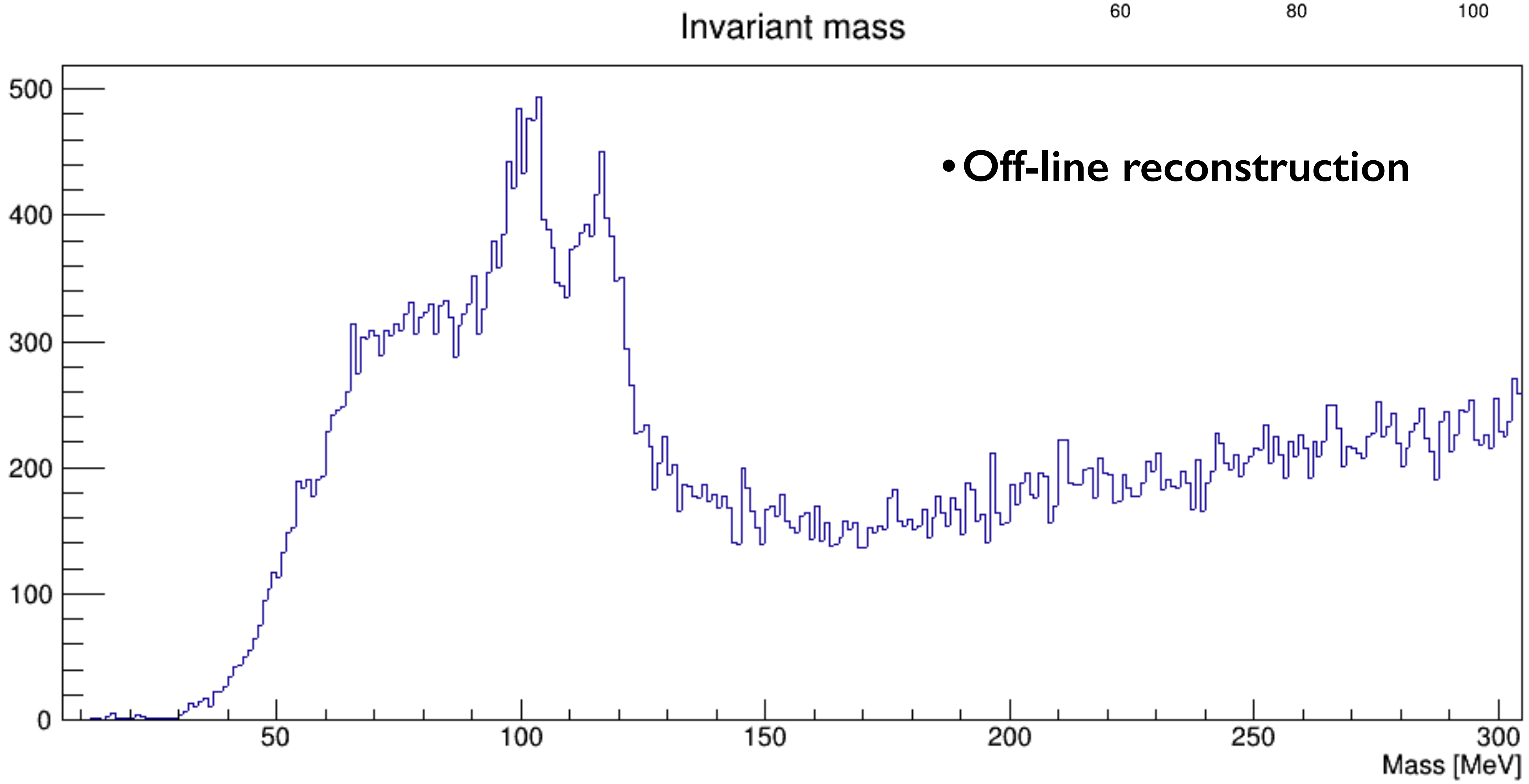
- On-beam tests:
  - 10.4 GeV e- beam on thin Pb/Al target
  - Inclusive pi0 production
    - e + Pb/Al -> Xeπ^0 -> (X)e$\gamma\gamma$
  - Two gammas detected in FT-CAL



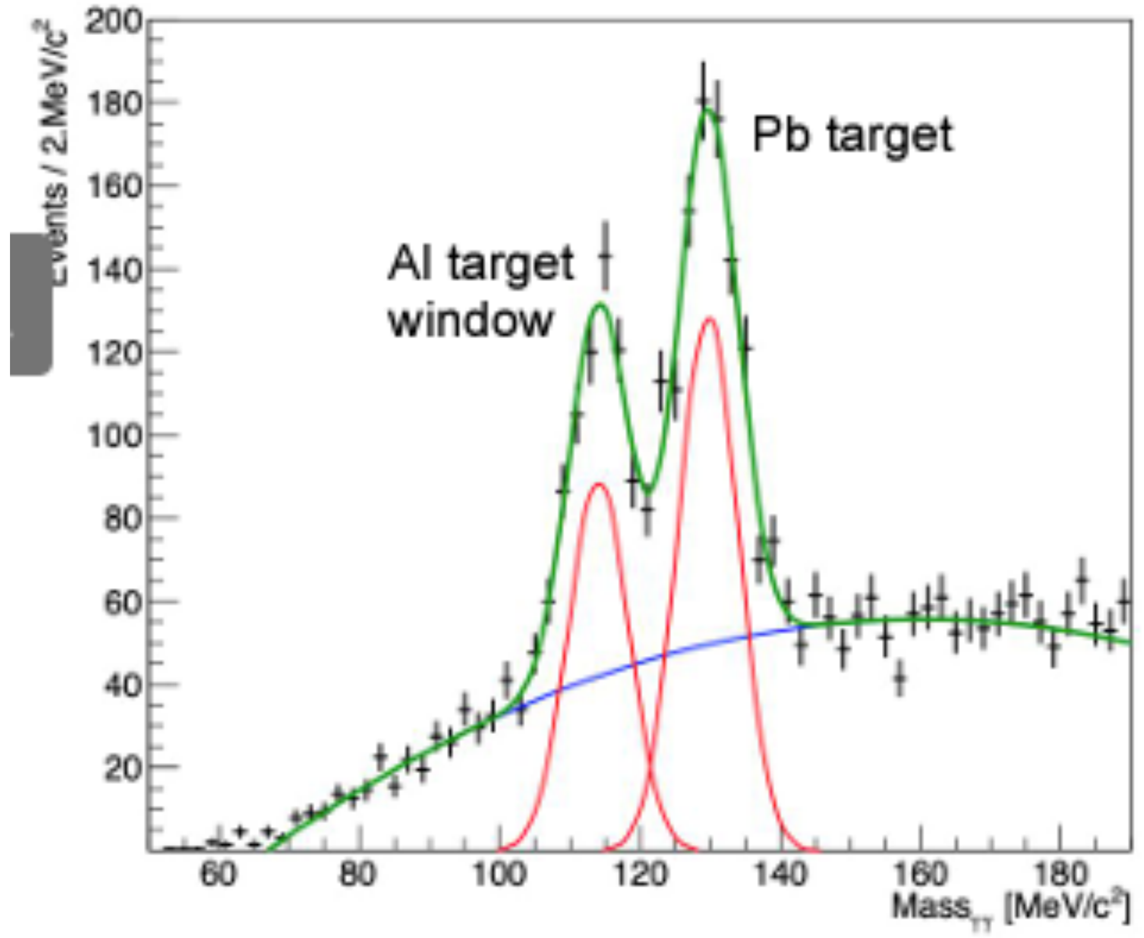
Off-line analysis

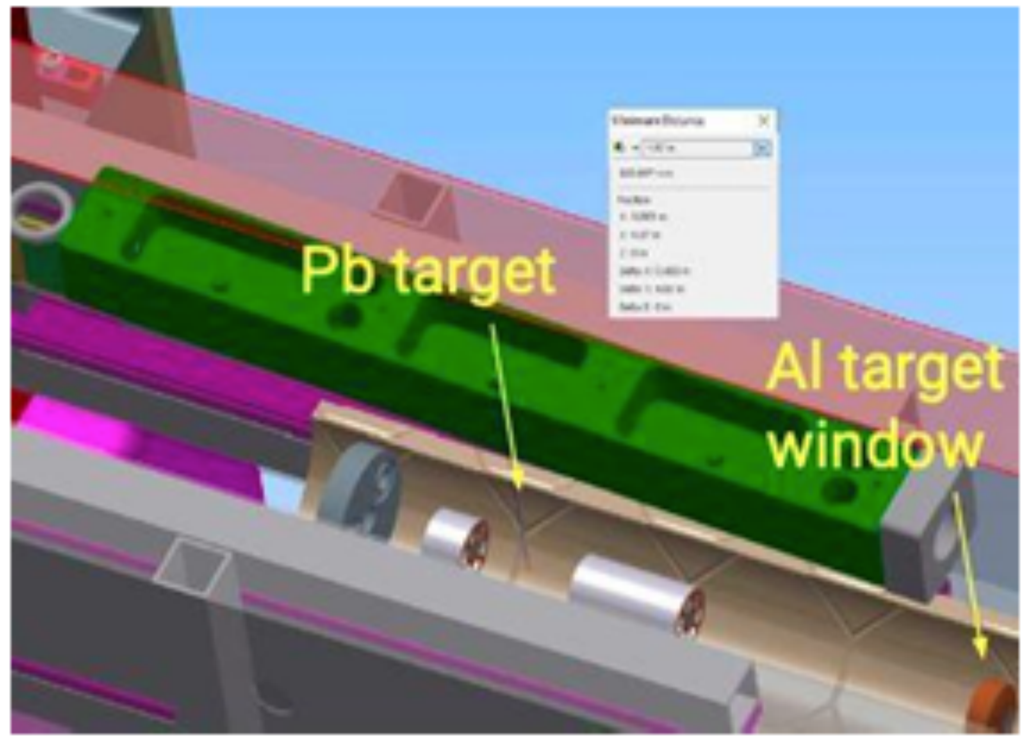


- Two pi0 peaks corresponding to two vertices (and a wrong assumption on the vertex position)



- Off-line reconstruction





Shall we used AI to analyse data real time, extract features (e.g. number of peaks and position)?

Semi-unsupervised: K-means

Semi-supervised Clustering: e.g., K-means

STEP 1: Choose the number K of clusters

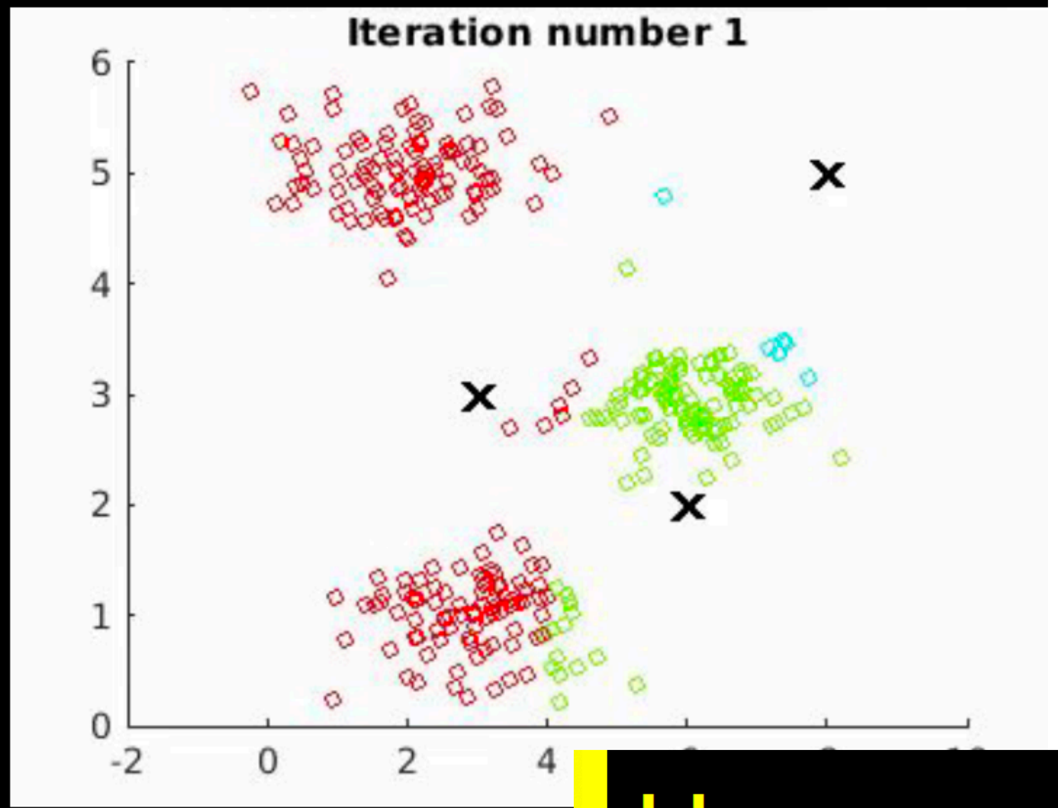
STEP 2: Select at random K points, the centroids
(not necessarily from your dataset)

STEP 3: Assign each data point to the closest centroid
(That forms K clusters)

STEP 4: Compute and place the new centroid of each cluster

STEP 5: Reassign each data point to the new closest centroid
If any reassignment took place, go to STEP 4,
otherwise go to FIN.

Your Model is Ready



Yes, we can: semi unsupervised clustering using K-means

Hyperparameters and metrics

Table 2. The different metrics used for k-means.

metric	description
$(X_{hit} - X_{mean})^2 + (Y_{hit} - Y_{mean})^2$	squared 2D space distance
$\frac{(X_{hit}-X_{mean})^2}{L^2_{cell}} + \frac{(Y_{hit}-Y_{mean})^2}{L^2_{cell}} + \frac{(t_{hit}-t_{mean})^2}{(50\ ns)^2}$	squared 3D space-time distance
$\frac{(X_{hit}-X_{mean})^2}{L^2_{cell}} + \frac{(Y_{hit}-Y_{mean})^2}{L^2_{cell}} + \frac{(t_{hit}-t_{mean})^2}{(50\ ns)^2} + (E_{hit} - E_{mean})^2$	squared 4D space-time-energy distance

Table 3. The main parameters of the k-means algorithm are described and their values reported. For each parameter, the last column shows when it intervenes, either if in the pre-processing or in the clustering phase.

parameter	description	value [units]	phase
t threshold	minimum time of hits	0. ns	preprocessing
E threshold	minimum energy of hits	0. GeV	preprocessing
time_window	time difference between hits	50 ns	preprocessing
count_cells	active neighbor cells for each hit	≥ 1	preprocessing
iterations	k-means updates	10 (30)	clustering
bad_distance	max distance hit-cluster	not used	clustering
bad_time	max time difference hit-cluster	not used	clustering
norm_space	normalization space distance hit-cluster	L_cell (cell length, see Tab. 2)	clustering
norm_time	normalization time difference hit-cluster	50 ns (see Tab. 2)	clustering
norm_ene	normalization energy difference hit-cluster	not used	clustering

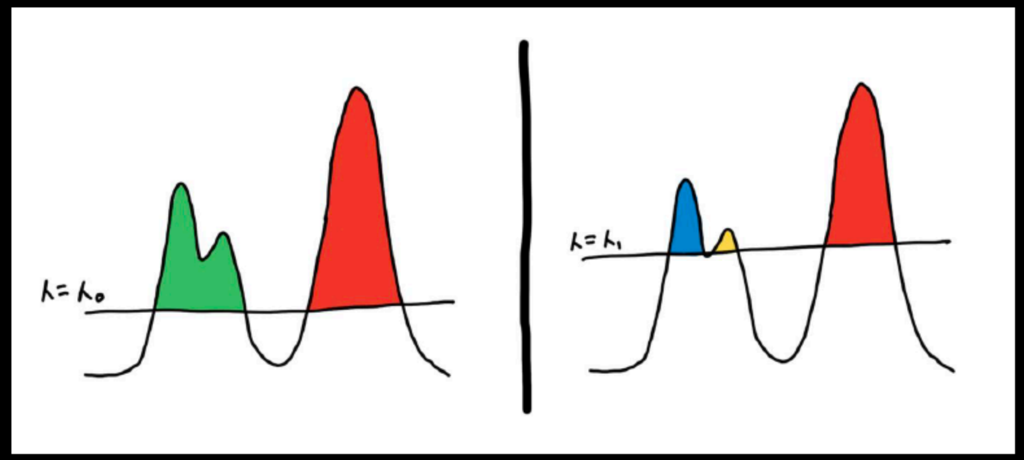
$$bool \ = \ \Delta t < 50\ ns \ \&\& \ \Delta X \ \leq 1 \ \&\& \ \Delta Y \ \leq 1 \ \&\& \ (\Delta X + \Delta Y) \ > \ 0 \qquad (3.1)$$

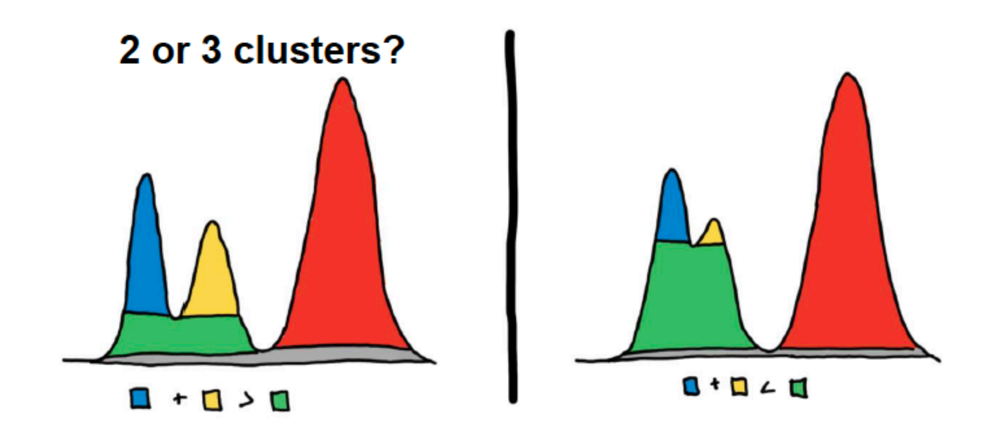
For K-means we need to make some assumptions, in particular we need to provide the seeds.

Unsupervised: hdbscan

Unsupervised: e.g., Hierarchical Clustering

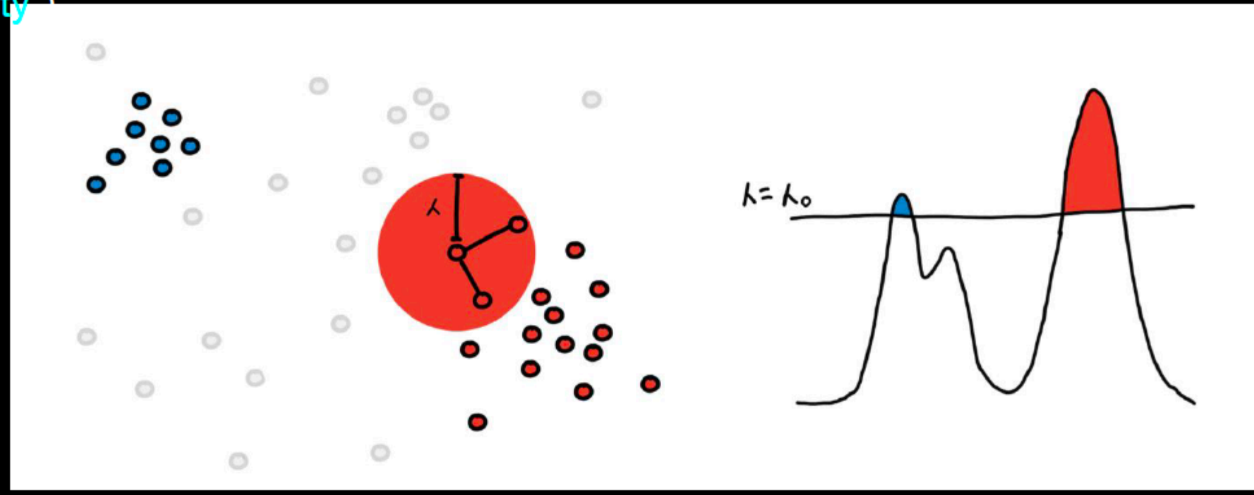
Two different clusterings based on two different level-sets

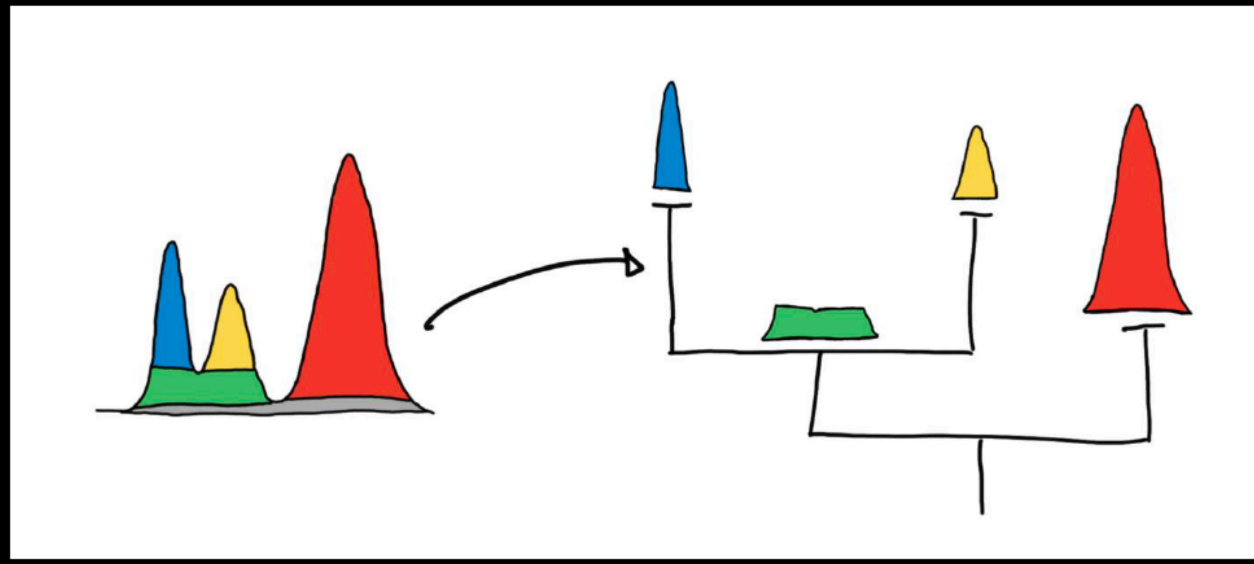




Core distance (defined by a required # of neighbors) as estimate of density

Points have to be in a high density region and close to each other ("mutual reachability")



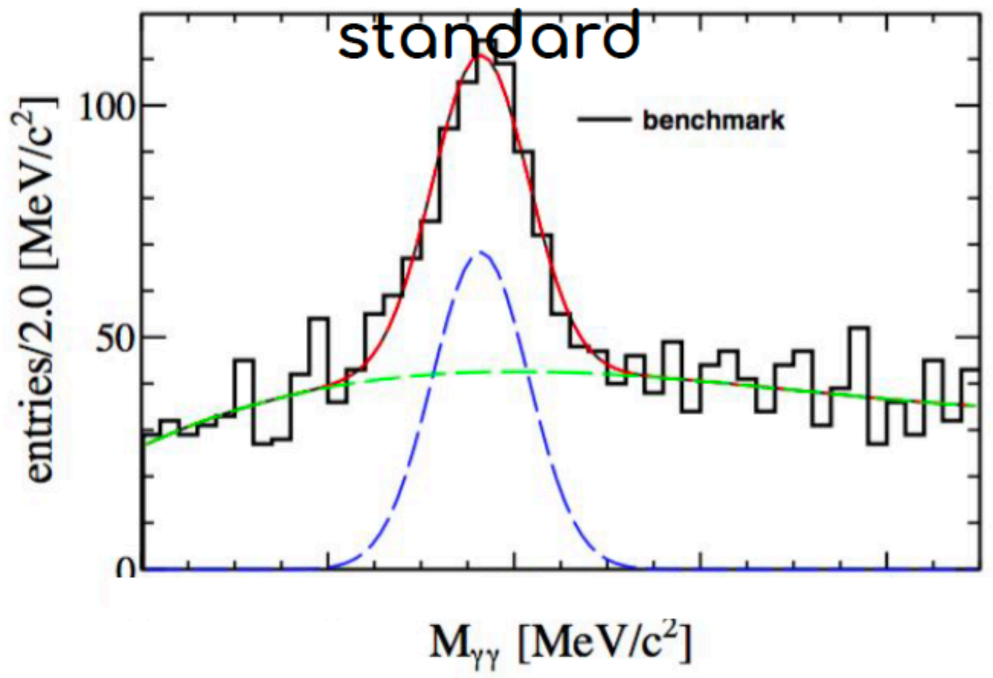


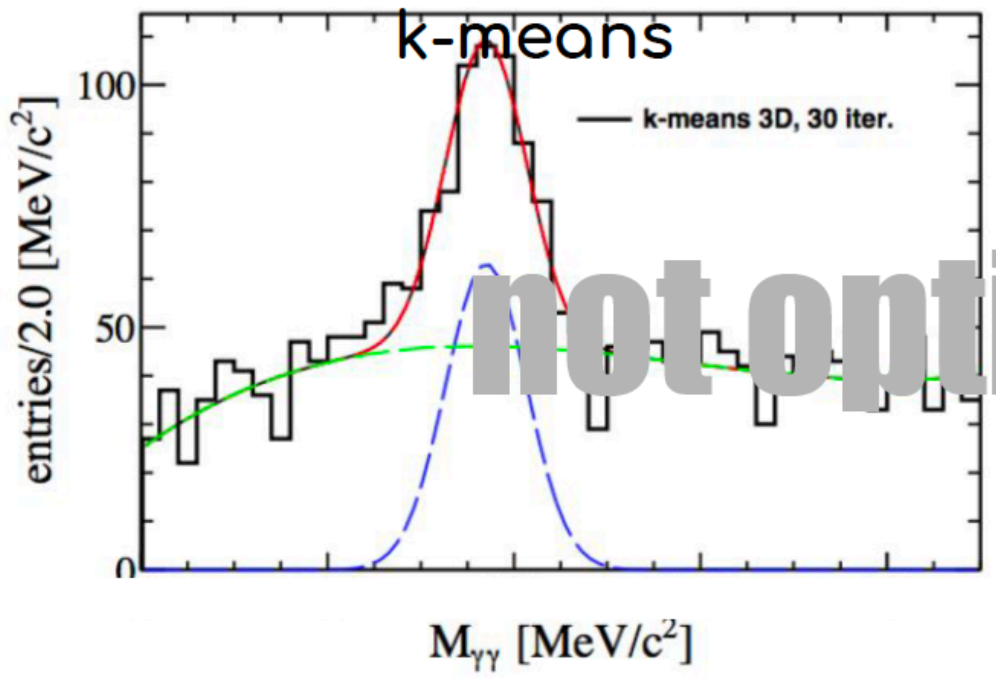
The area of the regions is the measure of "persistence".

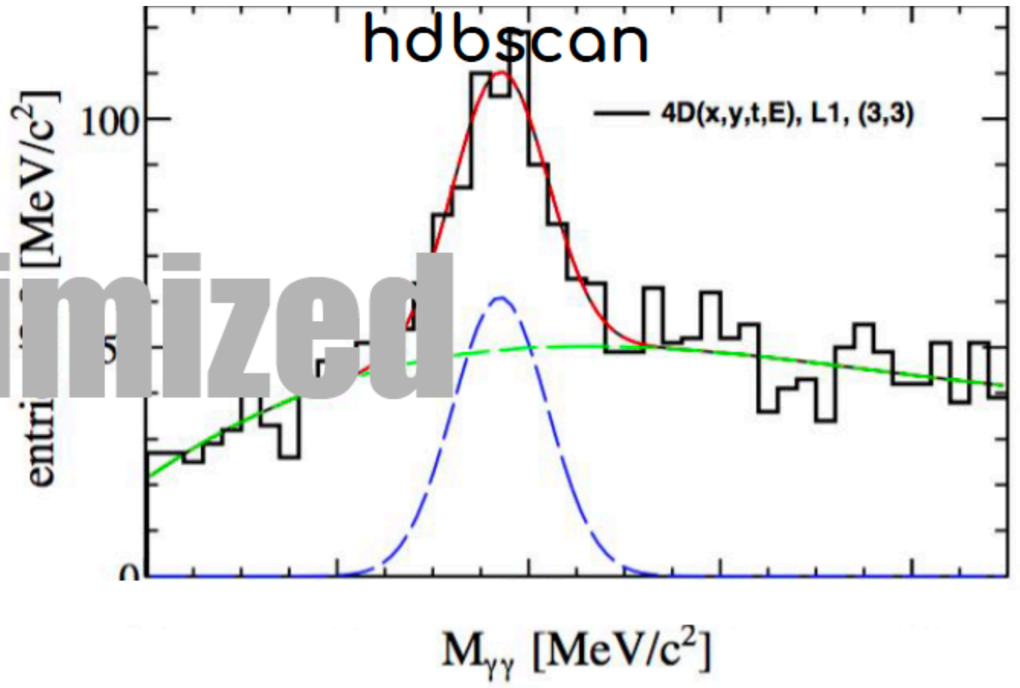
Maximize the persistence of the clusters under the constraint that they do not overlap.

clusters are more likely regions separated by less likely regions -> densities

- **Off-line analysis to tune hyperparameters**







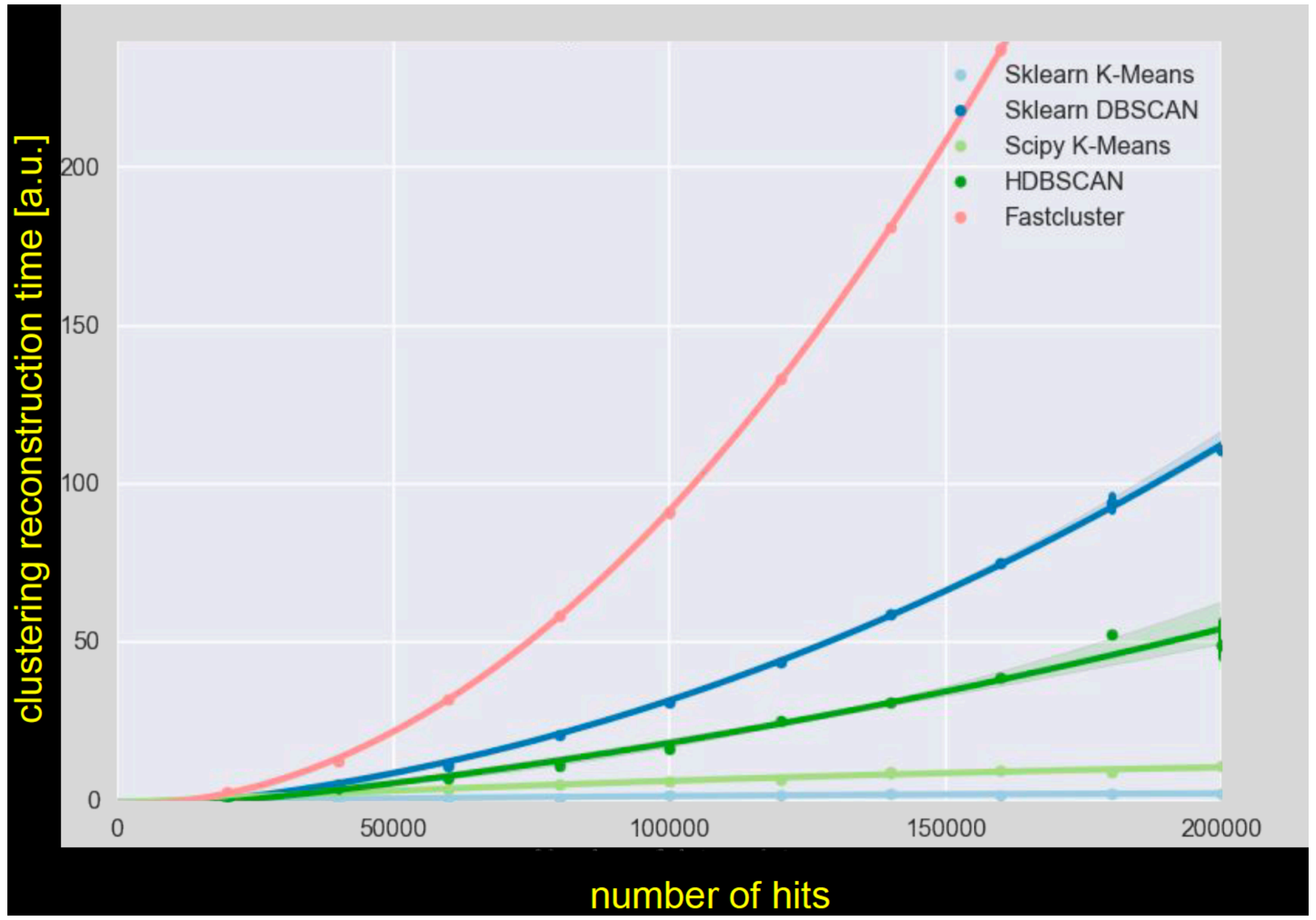
hdbscan vs. K-means

K-means: semi-supervised parametric (K cluster seeds)
Requirements on clusters:

- "round" or "spherical"
- equally sized, dense
- typically most dense in the center
- not contaminated by noise and outliers

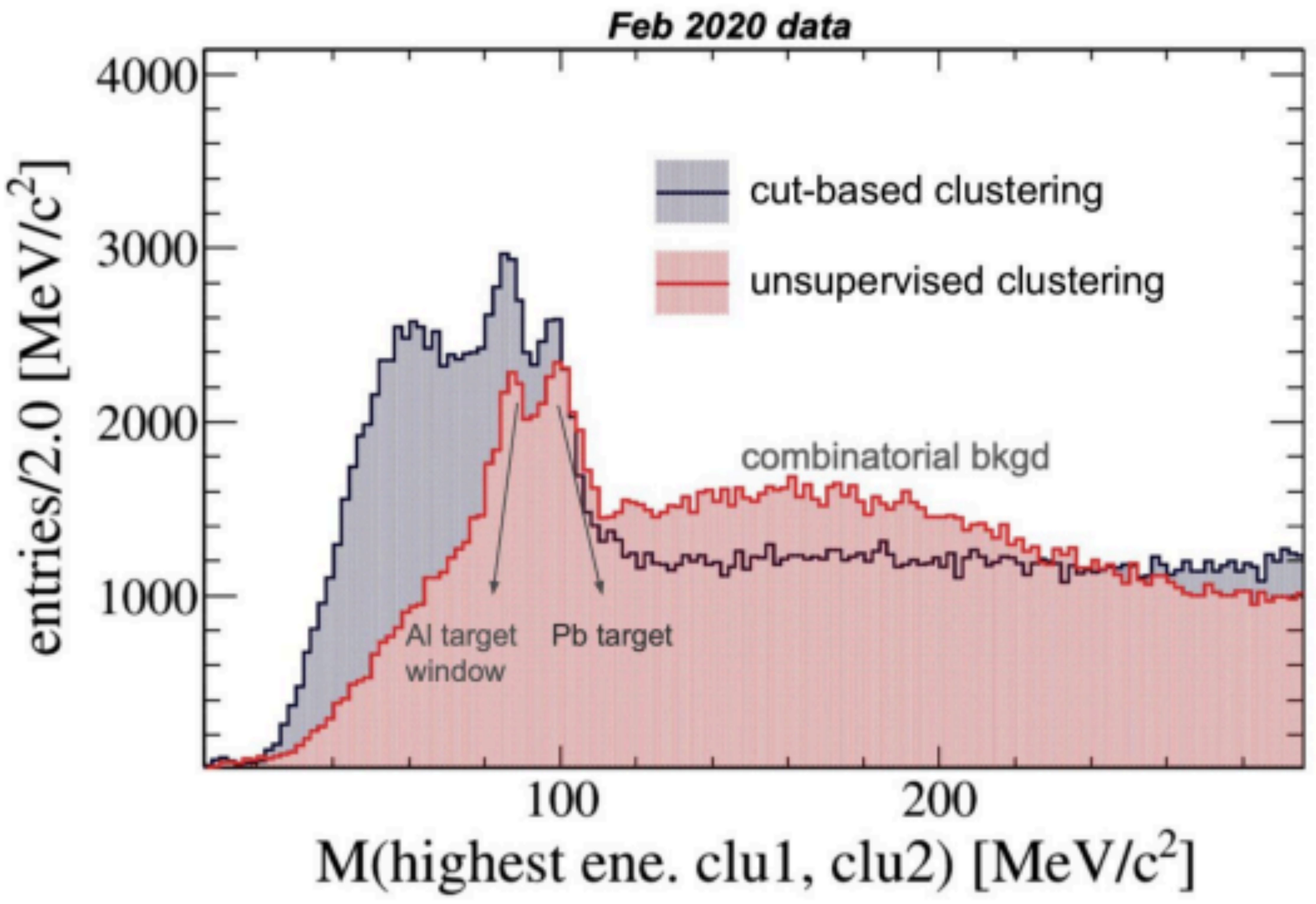
hdbscan: unsupervised hierarchical clustering
Best performance when data are/have:

- arbitrarily shaped clusters
- clusters with different sizes and densities
- noise



SRO test @ JLAB results: AI vs standard clustering

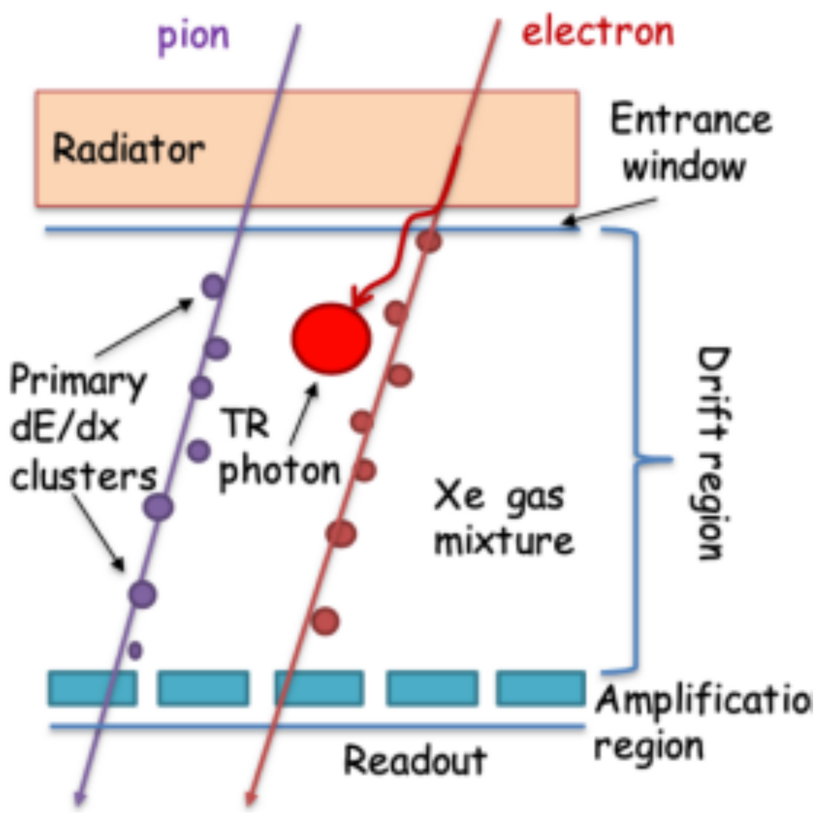
C. Fanelli

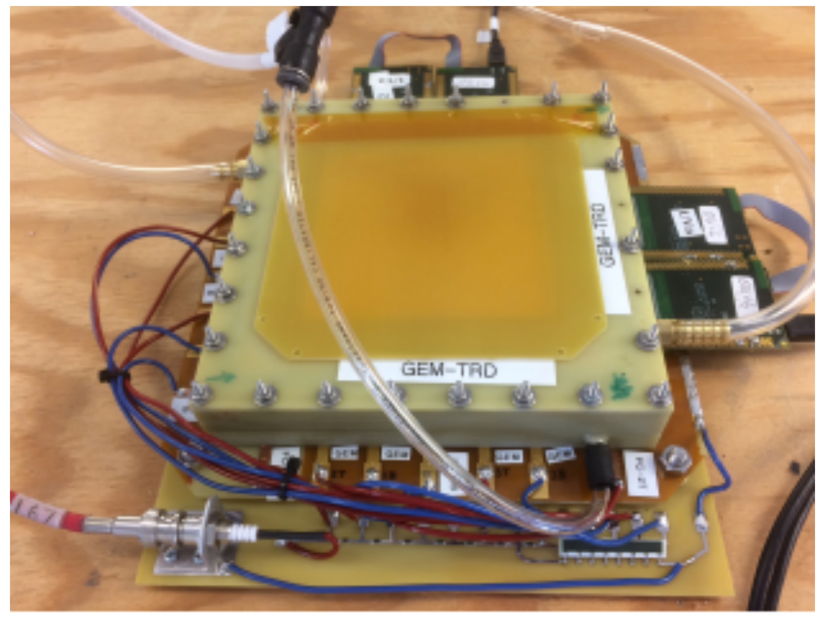


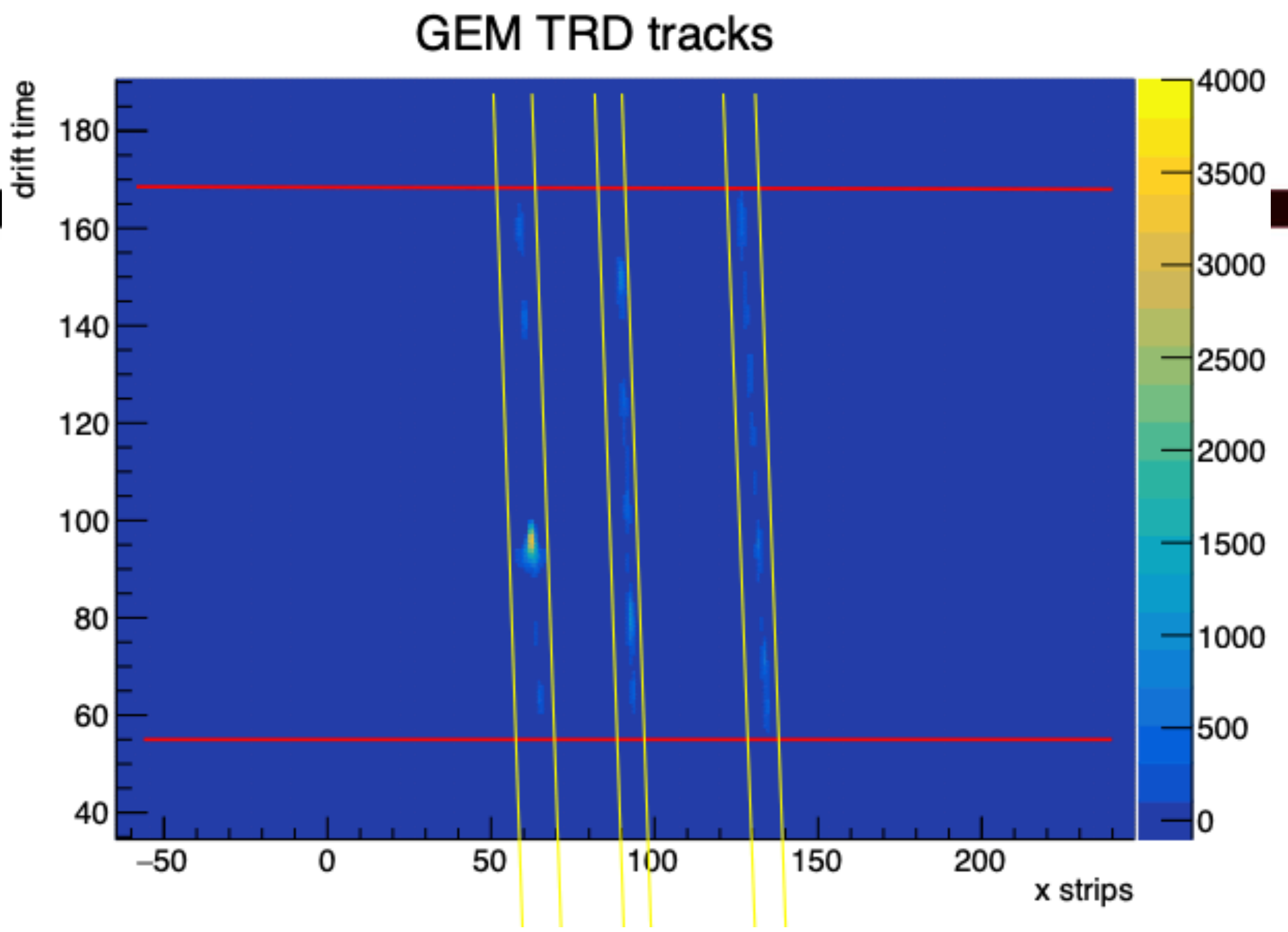
- AI clustering inspired by *Hierarchical Density-Based Spatial Clustering of Applications with Noise* (HDBSCAN)
  - It is not cut-based
  - it is able to cope with a large number of hits
- Compared γγ-invariant mass spectrum obtained utilizing both the standard and the HDBSCAN clustering algorithm
  - AI significantly improves signal-to-background ratio in the π0 region
  - A longer runtime of ~30% relative to the standard clustering algorithm
- AI clustering approach promising alternative to traditional cut-based approaches

F. Ameli et al., Eur. Phys. J. Plus (2022) 137: 958
https://doi.org/10.1140/epjp/s13360-022-03146-z

Fast AI applications: GEM-TRD





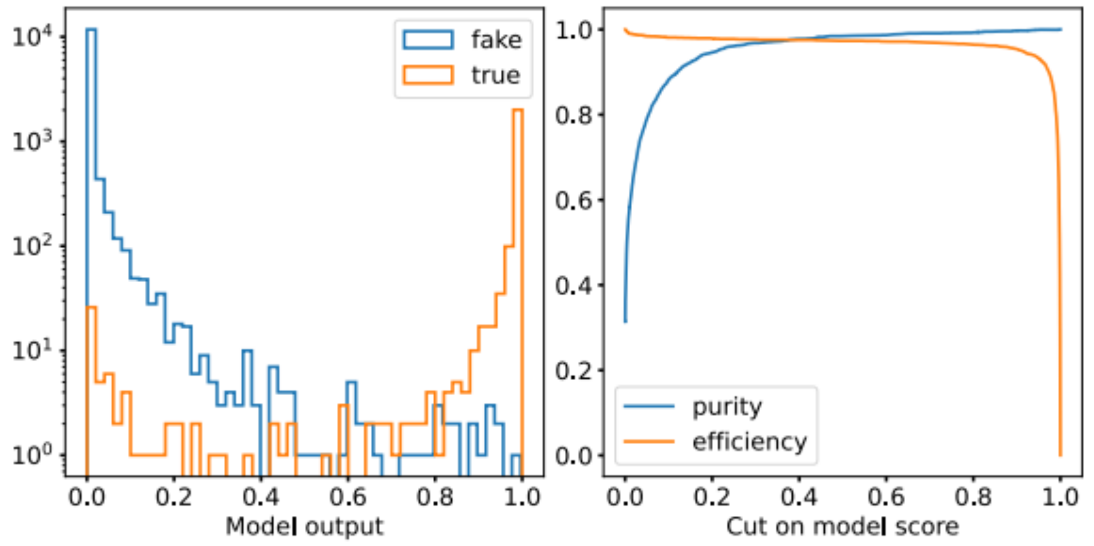


- GEM-TRD copes with multiple tracks
- Fast pattern recognition algorithm: Graph Neural Network (GNN)
- Track fitting: recurrent neural network – LSTM
- Implemented on FPGA using High Level Synthesis (hls4ml)

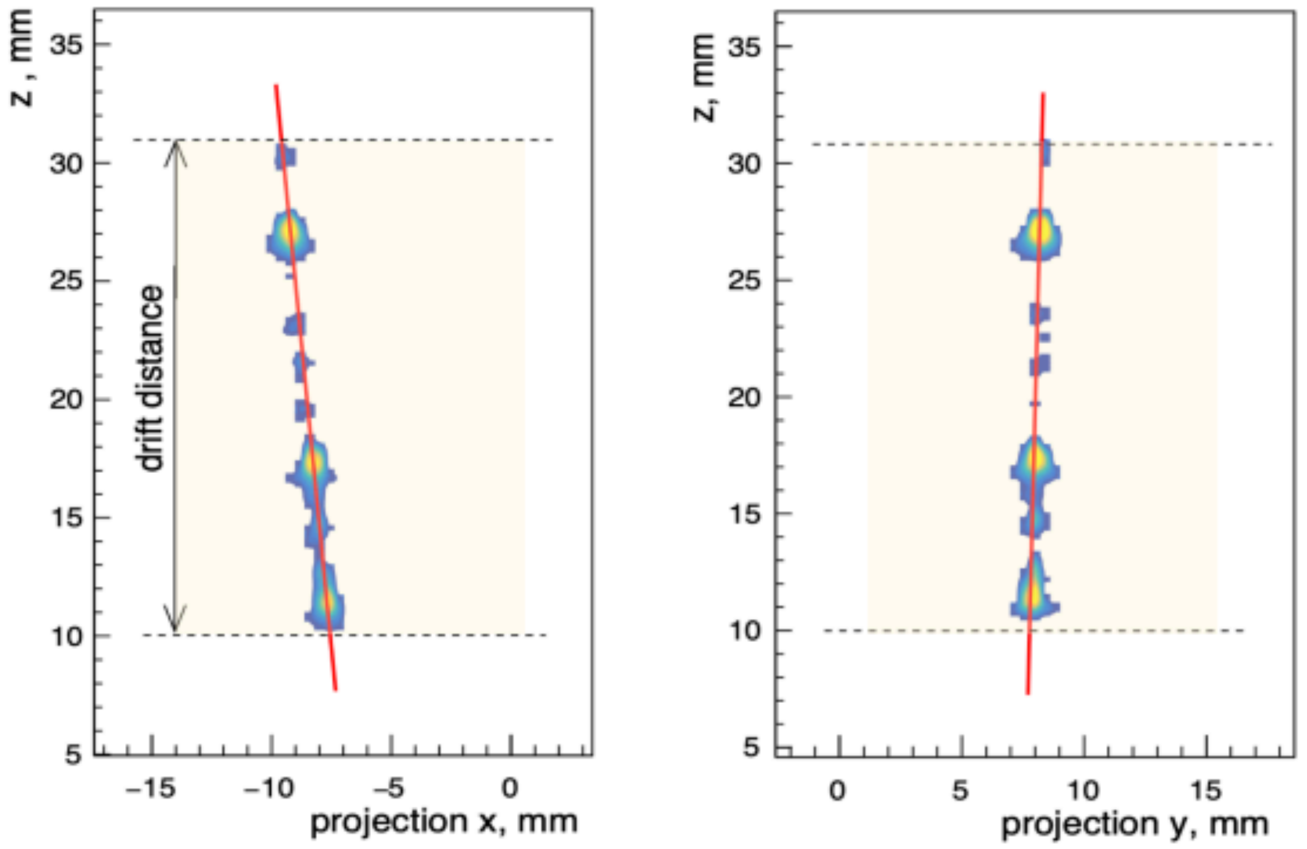
- e/pion separation based on ionization counting along track
- Electrons higher ionization (absorption of TR photons)
  1. detect hits
  2. hits in tracks
  3. ionisation measurement

GNN on FPGAs

- imported by hands
- 1.4us inference time
- Good p(preliminary) results



GEM-TRD can work as micro TPC, providing 3D track segments

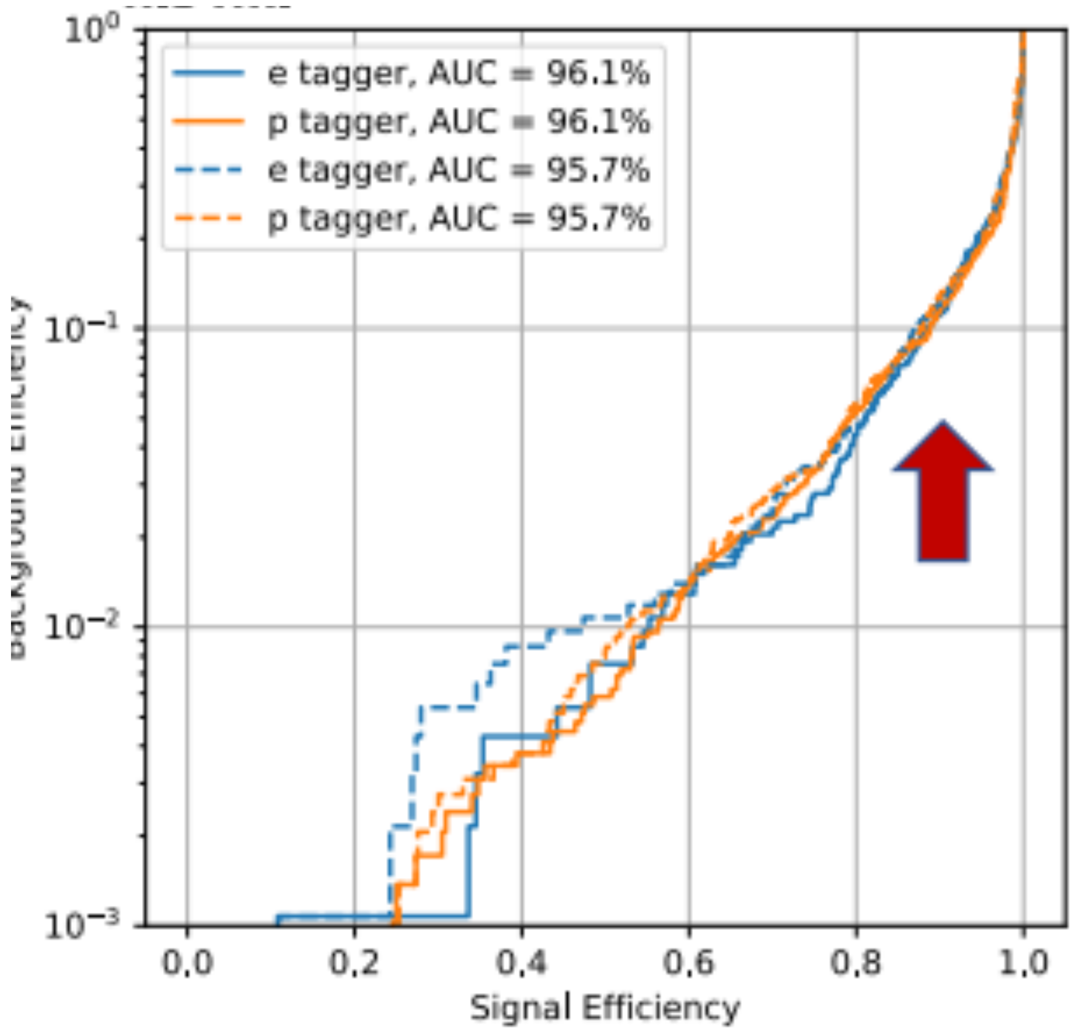


RNN/LSTM on FPGAs

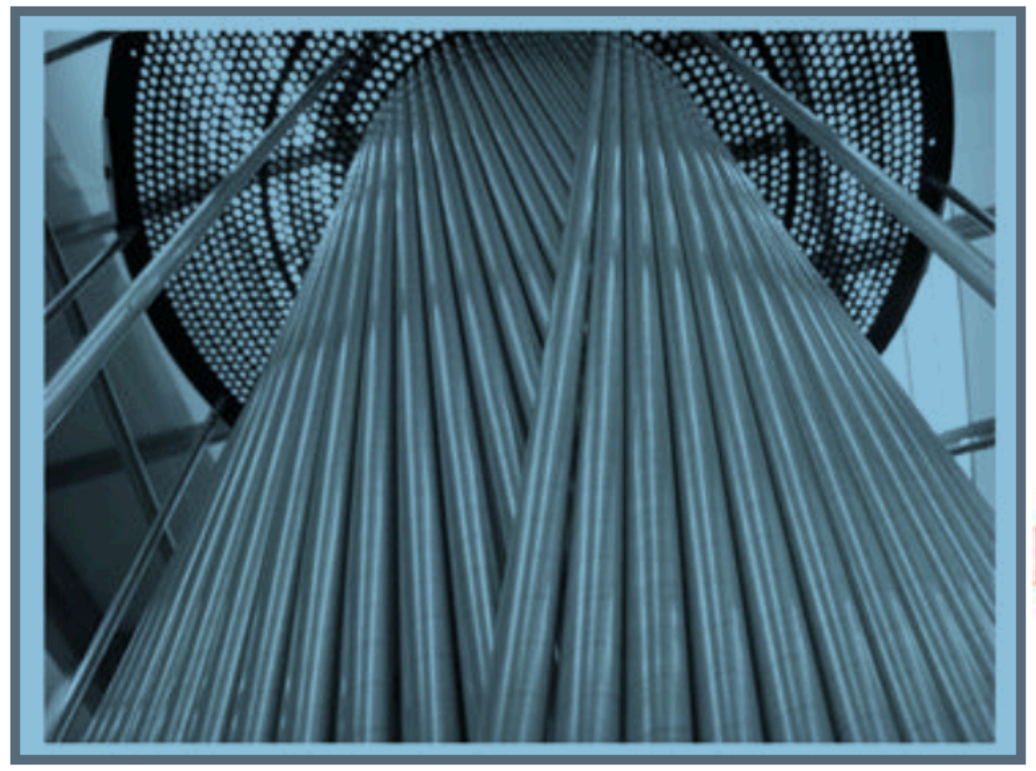
- Only 19% of FPGA resources
- 1us latency time
- Good (preliminary) performance

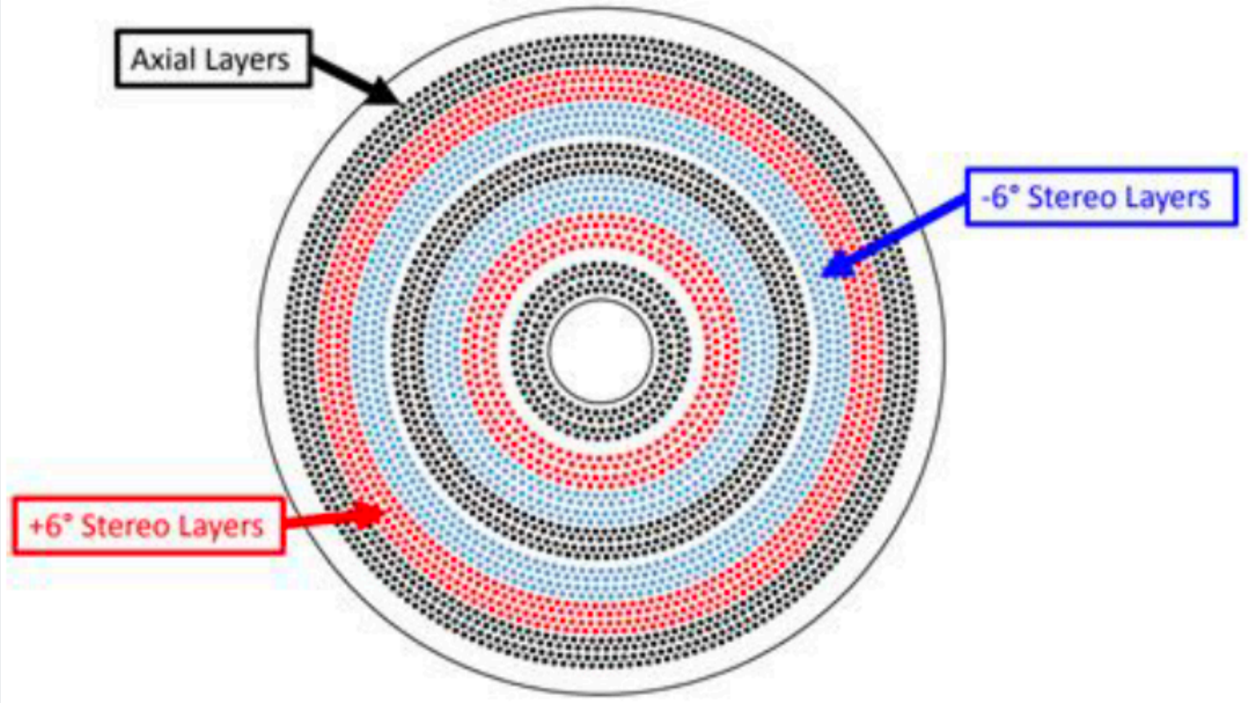
MLP on FPGAs

- Only 3% of FPGA resources
- 65ns latency time
- Good (preliminary) results



AI for a self-calibrating detector: GlueX Central Drift Chambers





Used to detect and track charged particles with momenta p > 0.25 GeV/c

- 1.5 m long x 1.2 m diameter cylinder
- 3522 anode wires at 2125 V inside 1.6 cm diameter straws
- 50:50 Ar/CO2gas mix

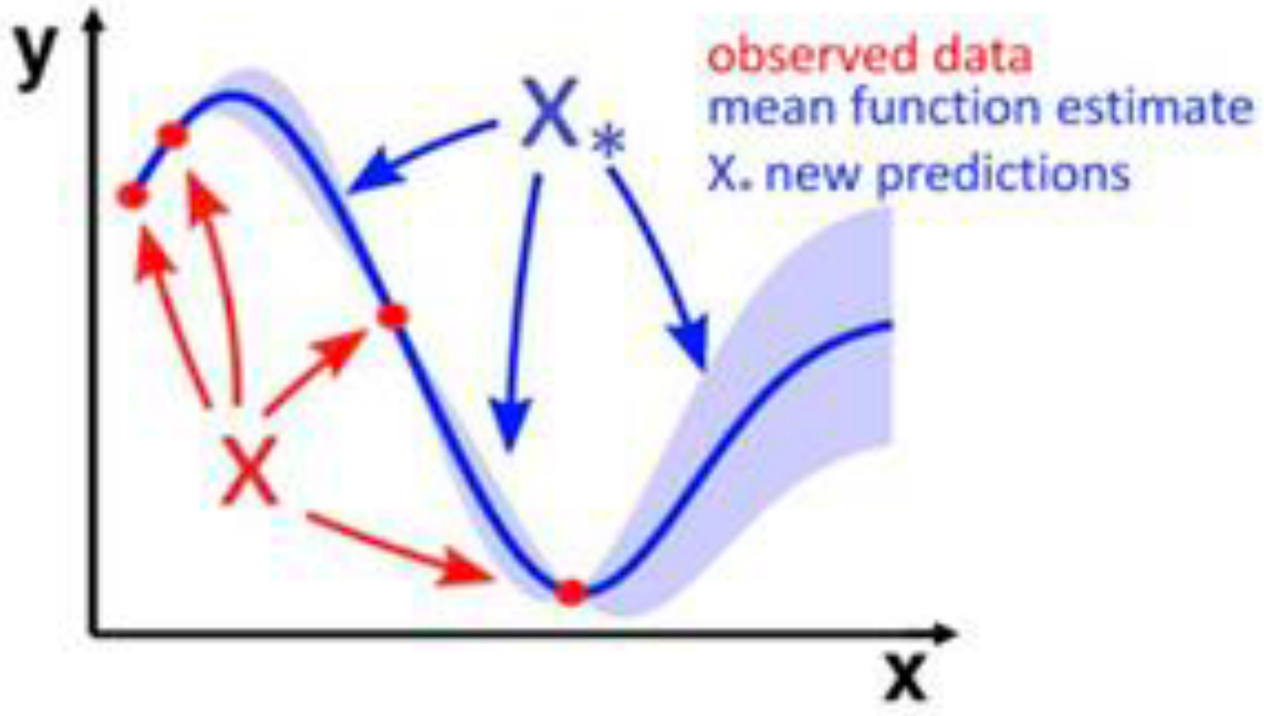
Requires two calibrations: chamber gain and drift time-to-distance

- Gain Correction Factor (GCF): have most variation +/-15%
- Has one control: operating voltage

ML Technique: Gaussian Process (GP)

Target: Provide traditional Gain Correction Factor (GCF)

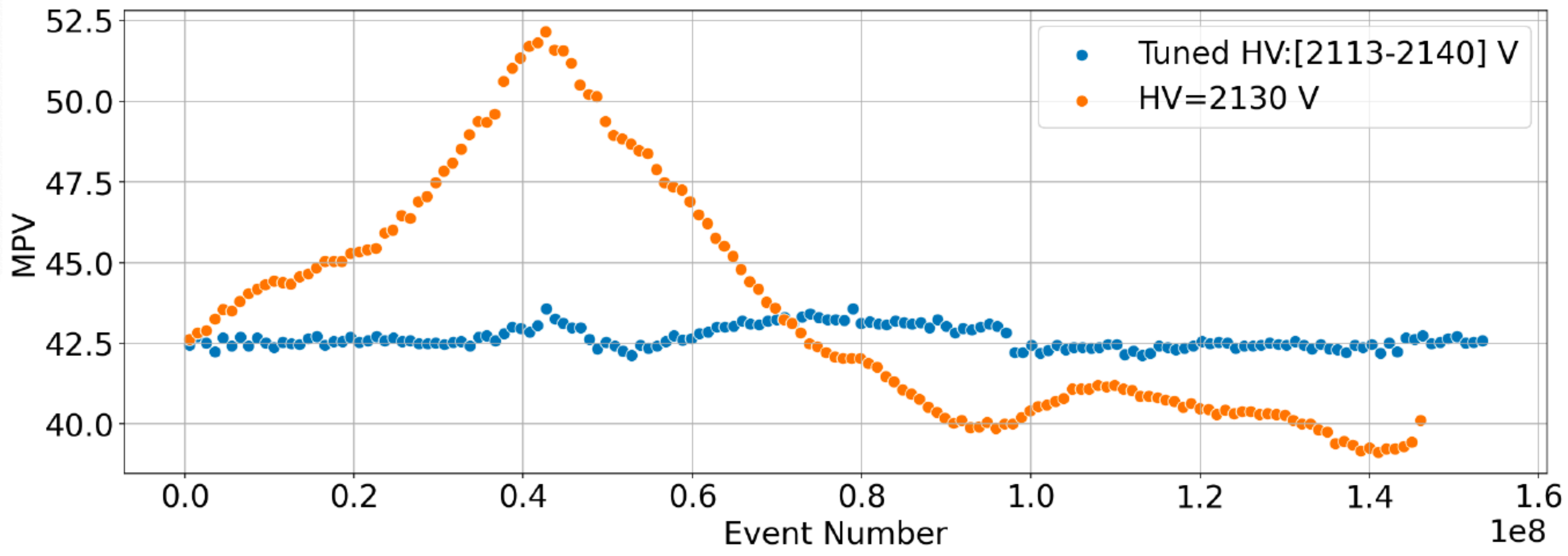
- atmospheric pressure within the hall
- temperature within CDC
- CDC high voltage board current



- GP calculates PDF over admissible functions that fit the data
- GP provides the standard deviationwe can exploit for uncertainty quantification(UQ)

•We used a basic GP kernel: Radial Basis Function + White

It works!



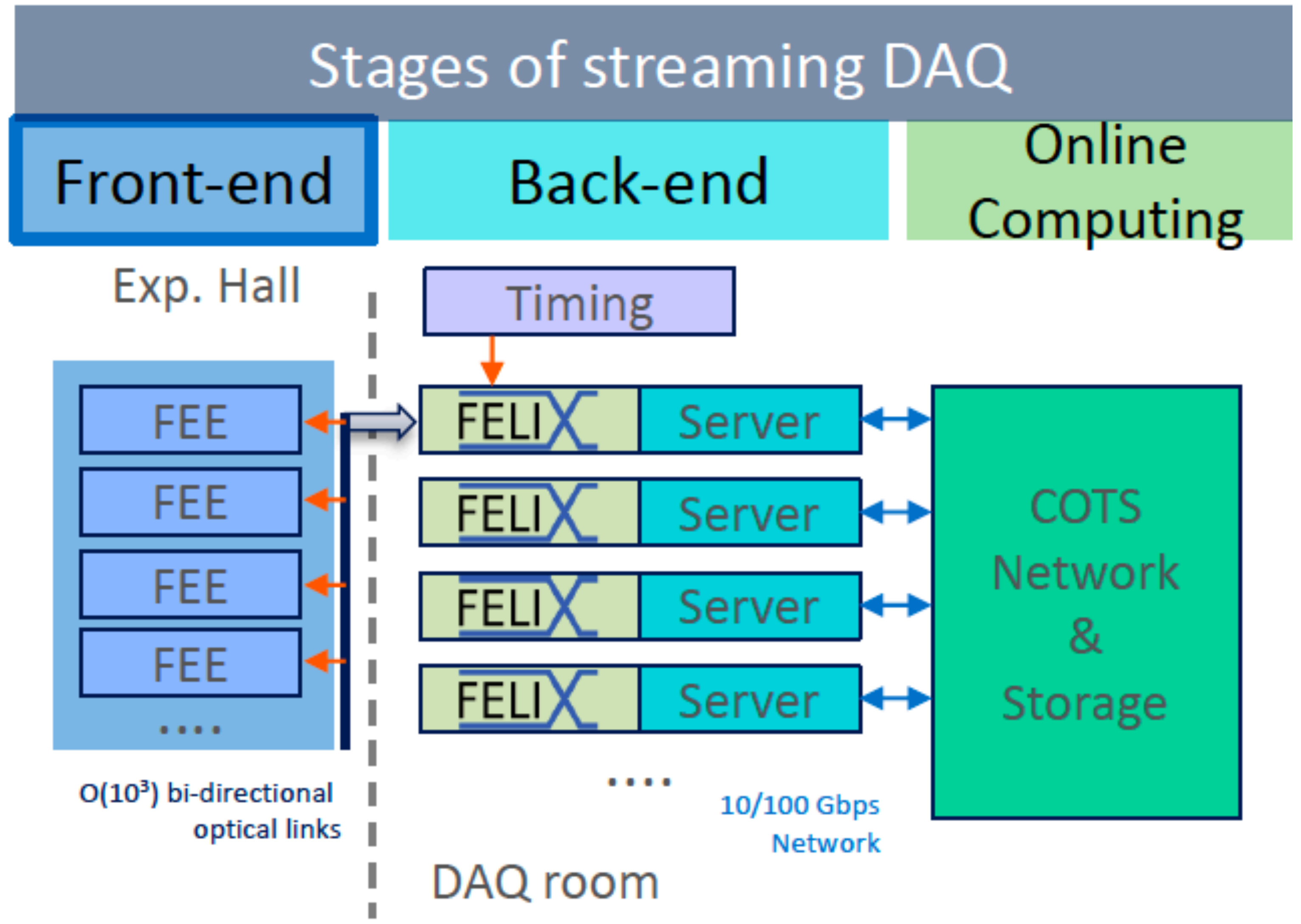
- Half the CDC (orange) at fixed HV, t he other half (blue) had its high voltages adjusted every 5 minutes

Realtime data reduction

Data reduction represents a main challenge in SRO

★Traditional DAQ: triggering (+ high level triggering/ reconstruction and compression) reduces data volume

★Streaming DAQ needs to reduce data real-time: zero-suppression, feature building, lossy compression

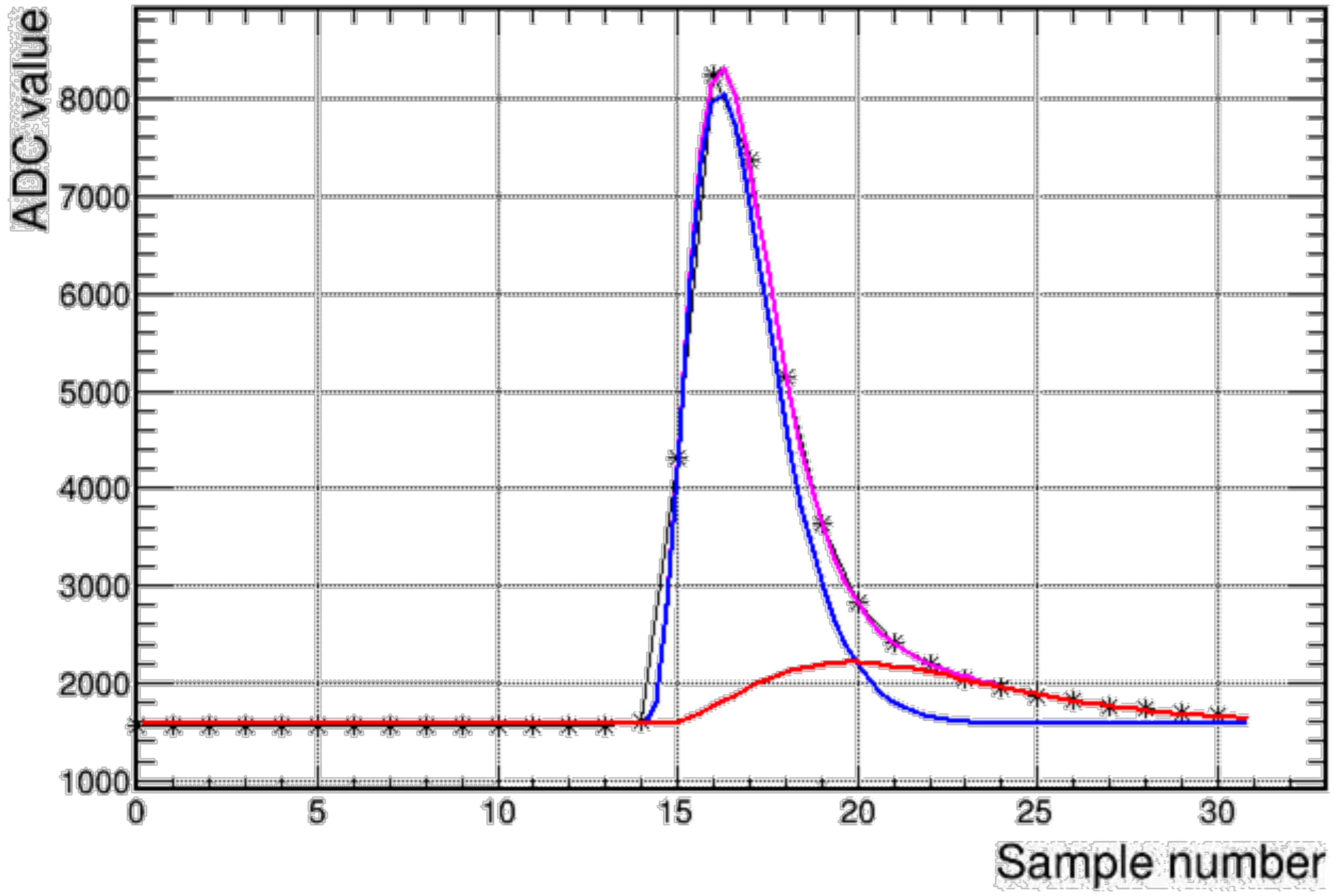


Opportunities for real-time AI but also a challenge

- reliable data reduction
- applicable at each stages of streaming DAQ
  - Front-end electronics
  - Readout Back-end
  - Online computing
- Data quality monitoring, fast calibration/reconstruction

Front end electronics

- Digitization (ADC, TDC, pixel readout)
- Data reduction strategy to immediately apply zero-suppression
- Real-time AI data reductions:
  - Improved zero-suppression (e.g.small signal recovery)
  - Feature building
  - Compression
- Target hardware: ASIC, (smaller) FPGAsCommon requirement of low-power consumption, radiation tolerant

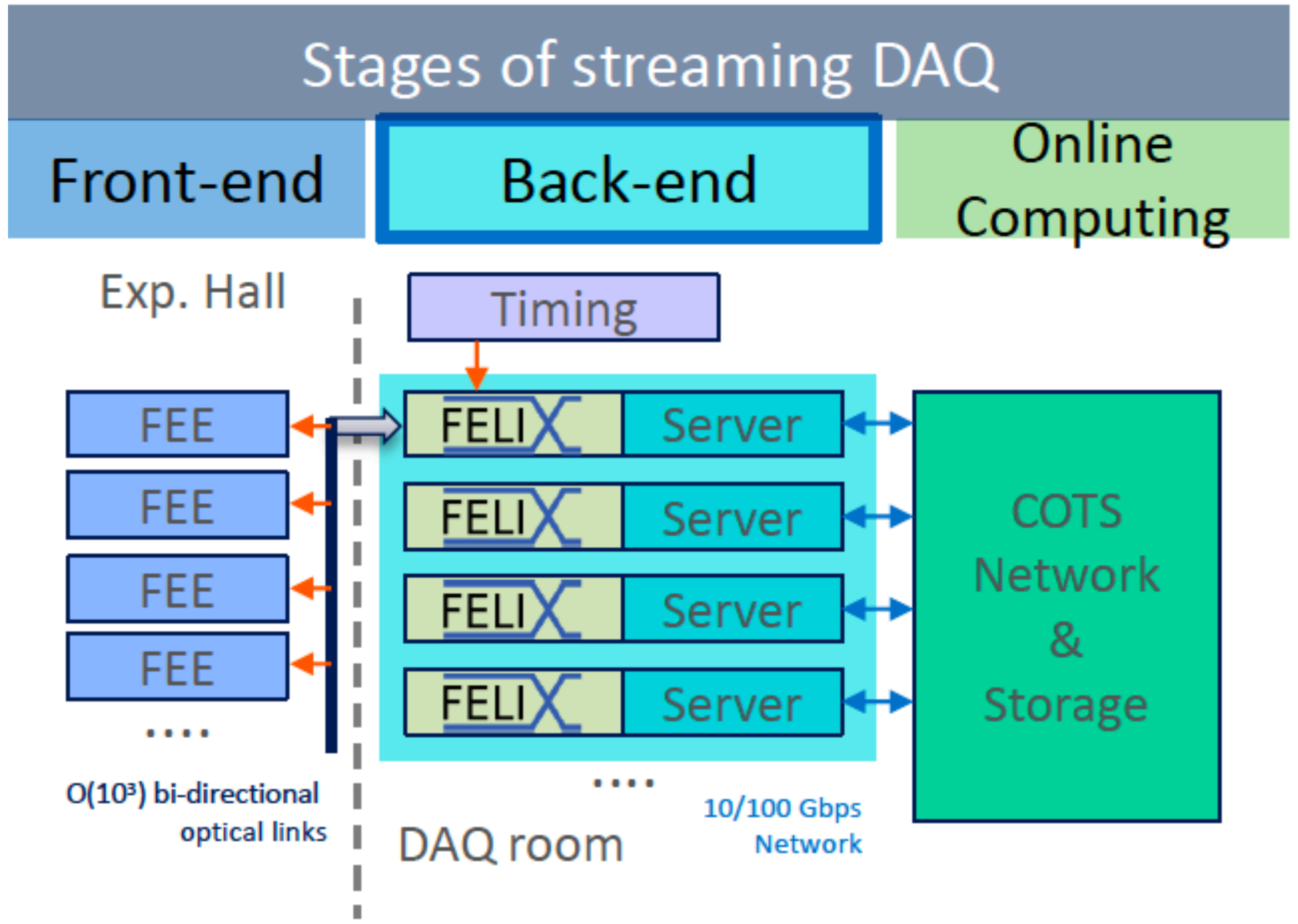


- Waveform digitizer: output data in ADC time series
- NN can be used in the FE to extract features (e.g. amplitude and time)
- Fit limited resources in FEE FPGA or ASIC
- quantized-aware training and pruning

Realtime data reduction

Read out back end

- Data aggregation and flow control
- FPGA as data receiver trough optical link
- Real-time AI data reduction
  - Higher-level feature building
  - Selection of interesting time slices,
  - background/noise rejection
- Target hardware: large-scale FPGAs



Online computing

- Online computing is an integral part of streaming DAQ
- Blending the boundary of online/offline computing
- Real-time AI data reductions
- Lossy compression
- Noise and background filtering
- Higher level reconstruction
- Target hardware: Traditional computing: CPU, GPU (or new AI-oriented hw)

Simple auto-encode neural network