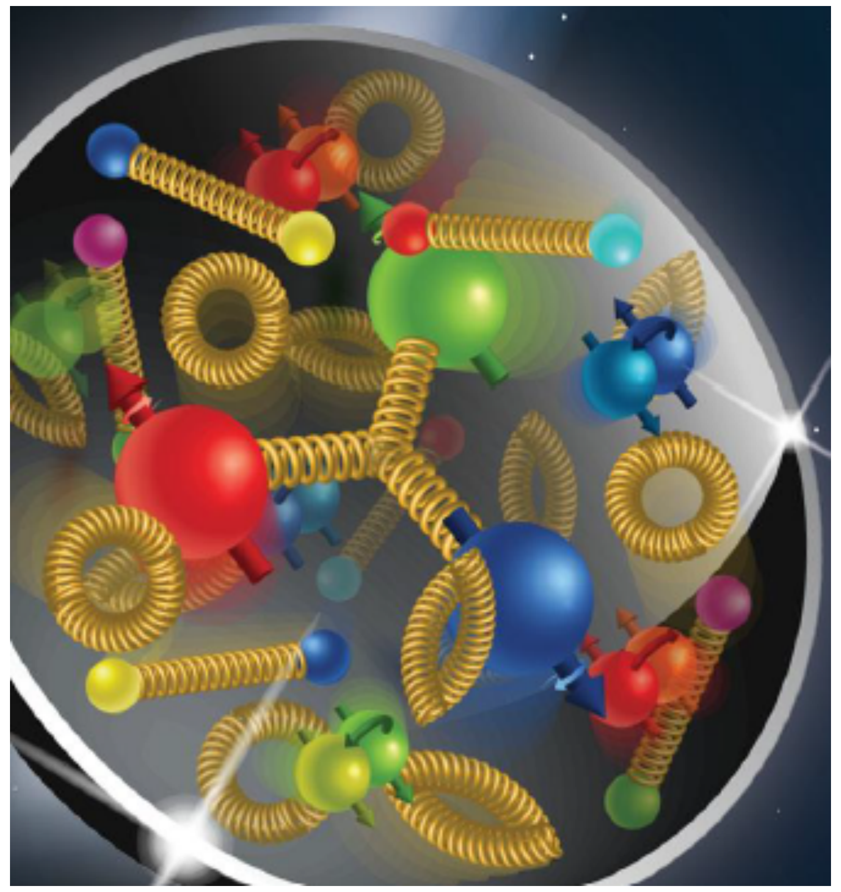
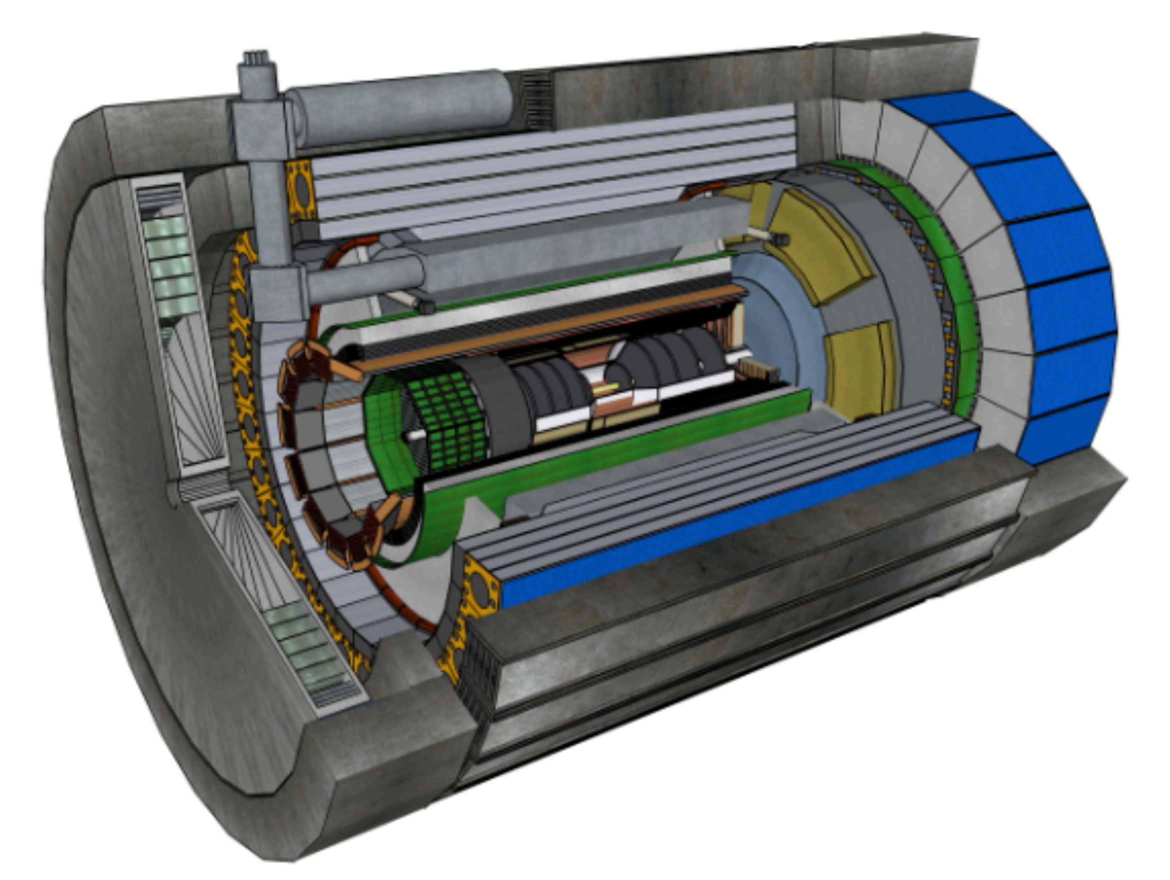
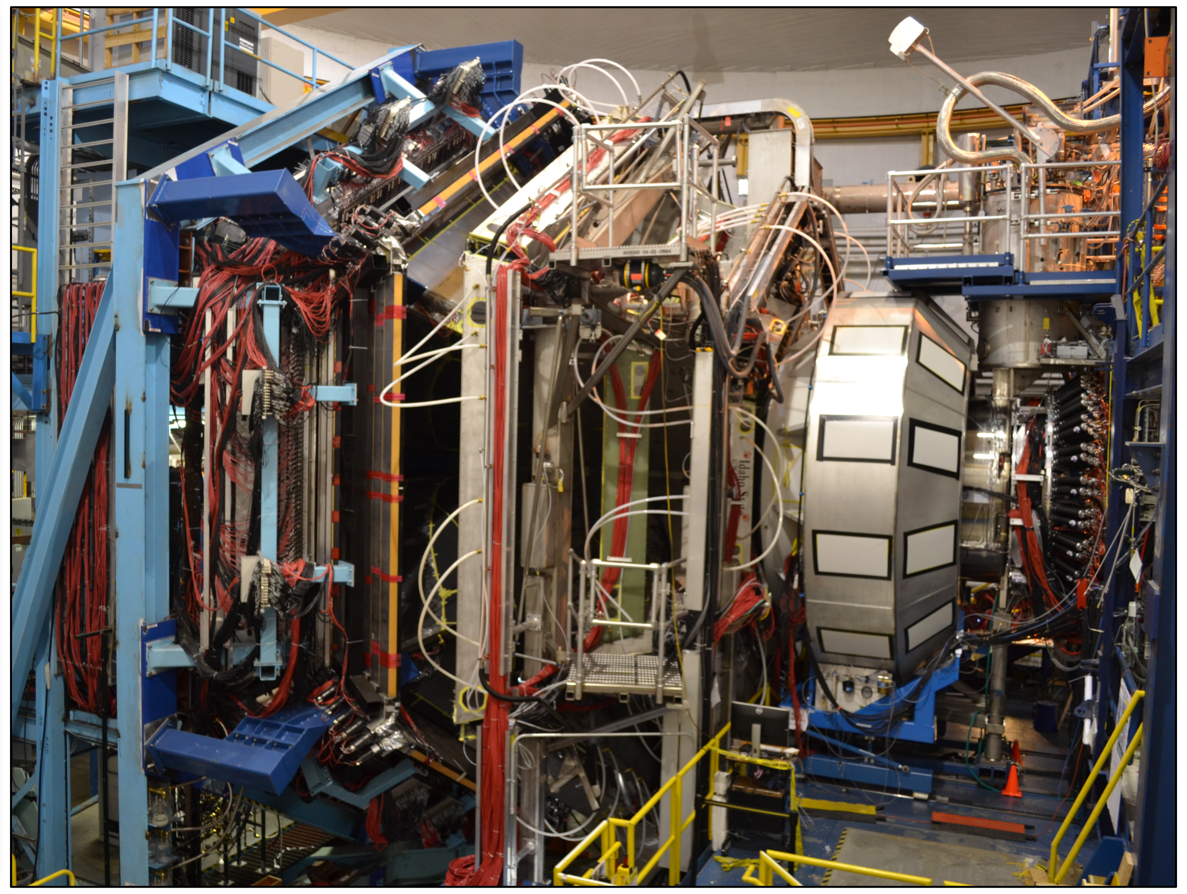
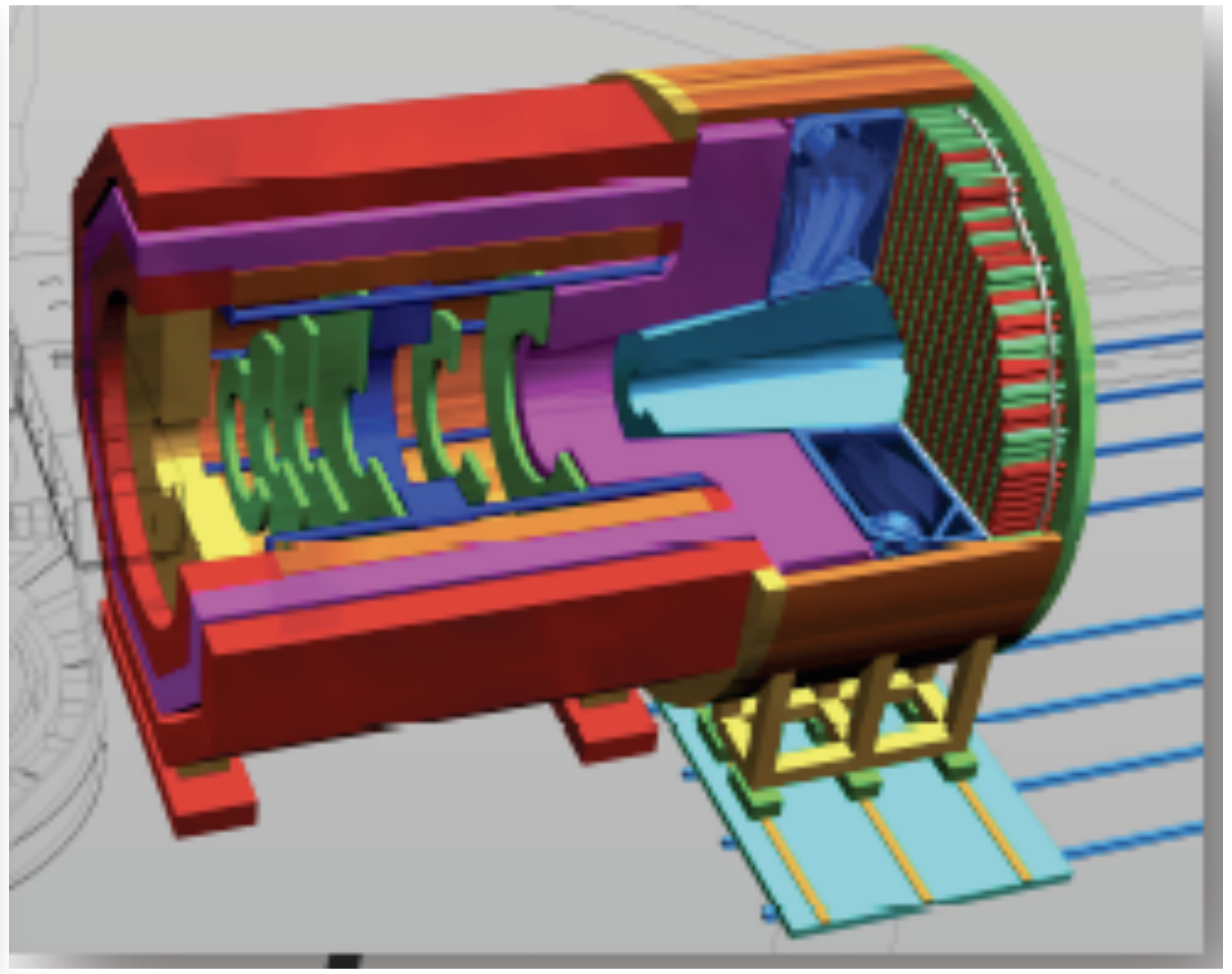








AI-supported methods for Real-time data analysis

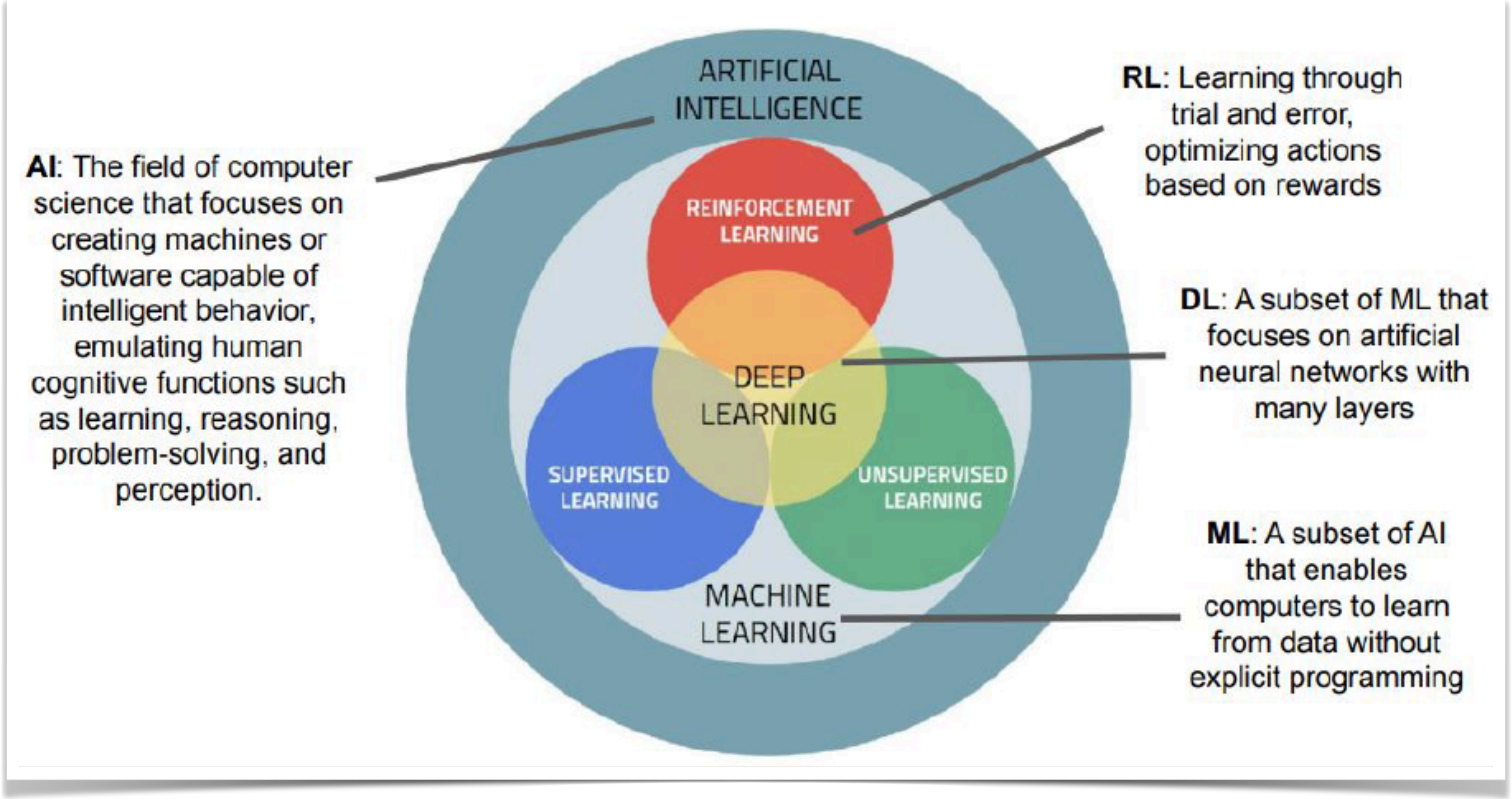
Part I - HEP/NP data and DAQ

M.Battaglieri (INFN)

Why AI for real-time data analysis (and why me)?

★ Streaming readout, the new paradigm in DAQ, calls for sophisticated real-time analysis (including AI/ML)

★ AI is a large and rapidly evolving field
- data scientists, mathematicians computer nerds,…
- what about physicists (theory, experimental)?



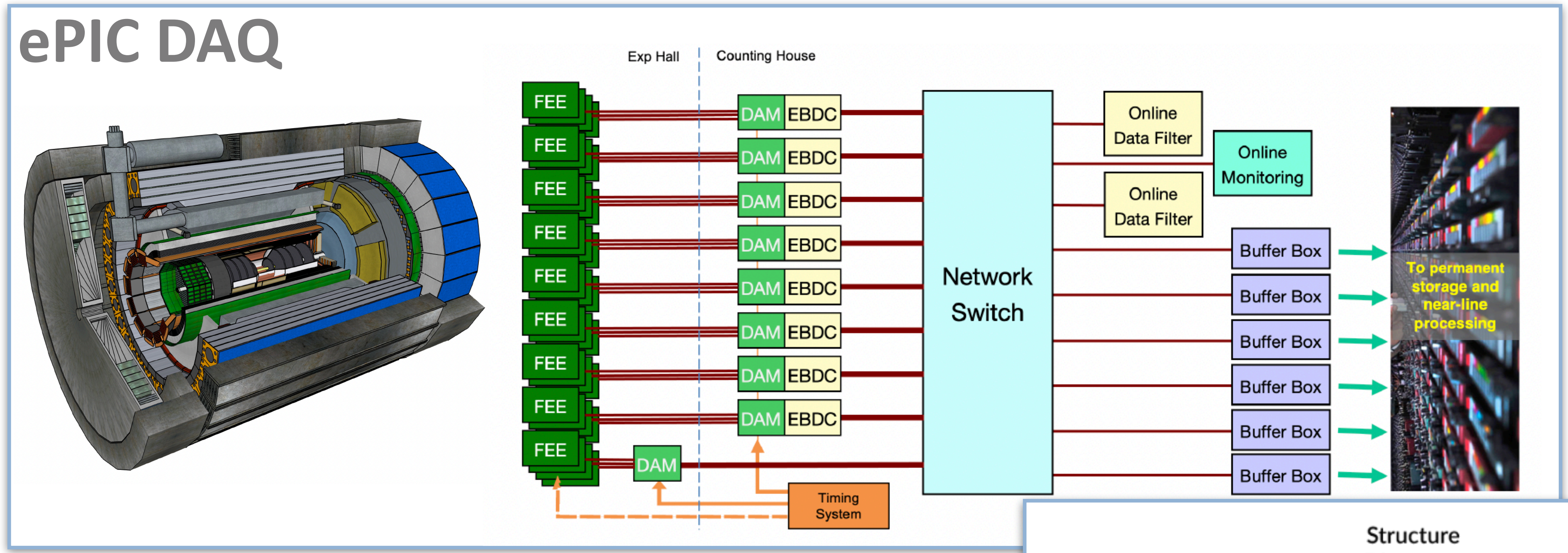
AI is widely used to (try to!) solve (too!) complex problems
- no analytic solutions
- too many data

I'm an experimental physicist with a long experience in detector's design and deployment and streaming readout data acquisition systems.
Despite I spent a large time of my life developing software tools for HEP and NP data analysis I'm:
- not particularly expert in computing, languages, coding, …
- not an expert in AI/ML
- pretty skeptical about AI as a magic wand for everything

… so, why me??? What is the advantage for you???
- unbiased (physicist-like) view of AI/ML uses
- a simple and basic picture of what AI/ML is
- a focus on physics (problems): AI/ML is simply a (powerful) tool
- I see (several!) advantages of using AI vs traditional approaches (and v.v.)
- far away from technical details (and complications)
- easy to fill your shoes (assuming you are a beginner in AI/ML!)

Only recently AI peeped out in physics (and it is spreading fast!)
- the *black box* approach is opposite to the scientific method (as articulated so far)
- AI is perpendicular to reductionism (we daily use in physics)
  - take a problem
  - identify the main features (making smart approximations)
  - extract from experimental data the (simple) underlying law
  - make the complexity simple ….
- Physics is a mature science: hundreds of years are difficult to reach (and beat!)

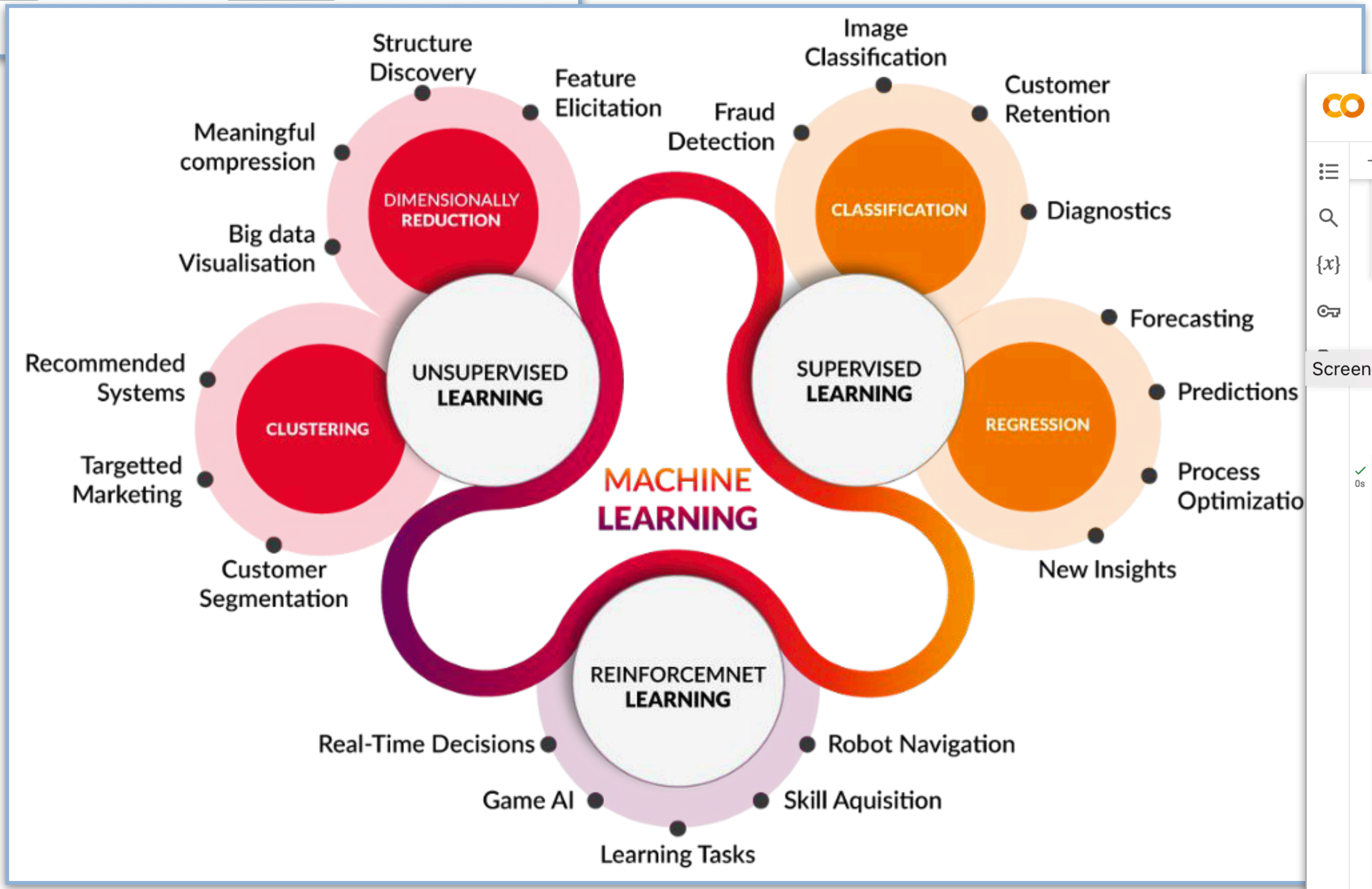
AI for Real time data analysis: CFNS lectures



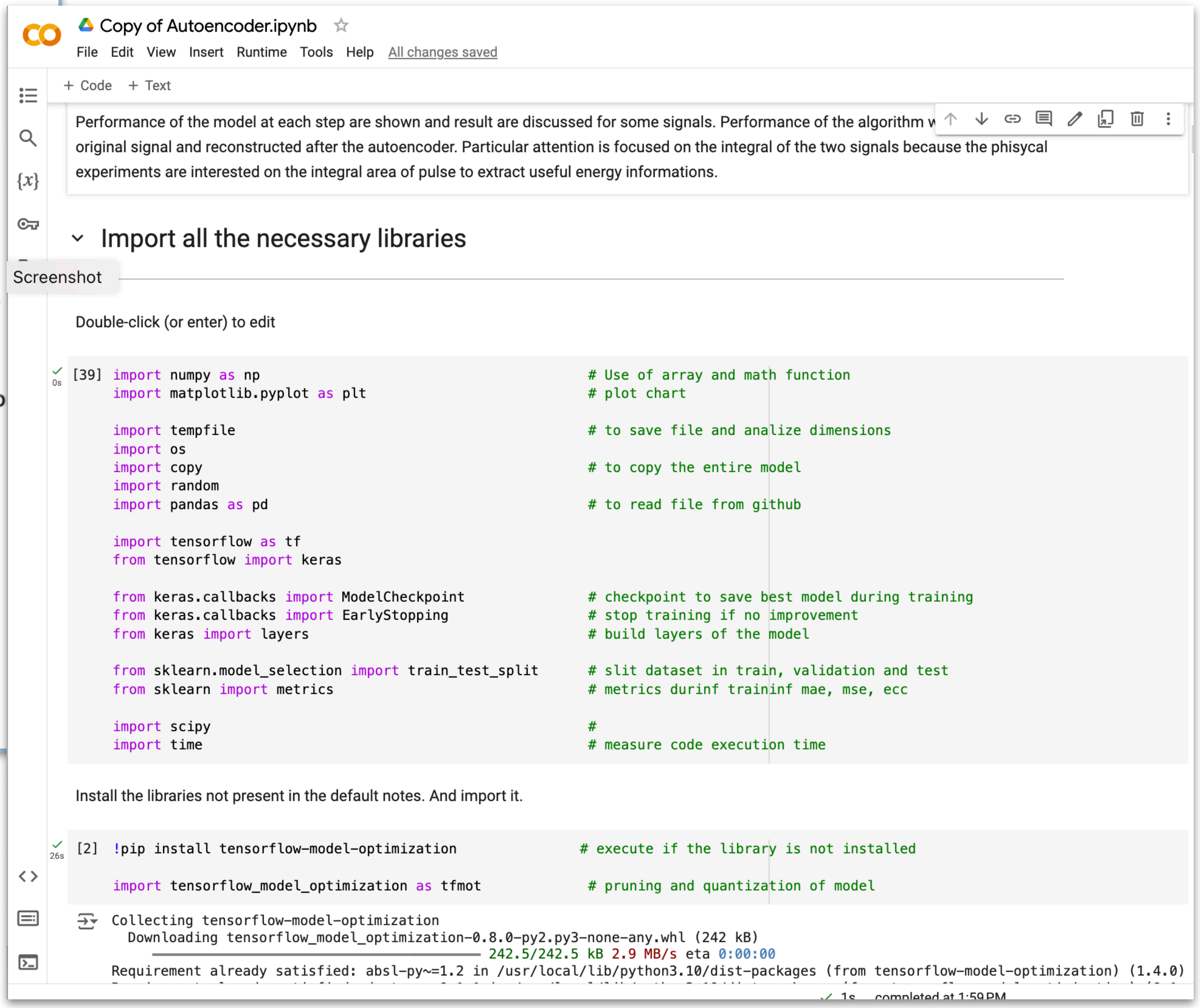
- Data Acquisition
- Particle detector's signals
- Trigger and Streaming readout
- AI in real-time data analysis

What can be done in 3 hours?

… not much



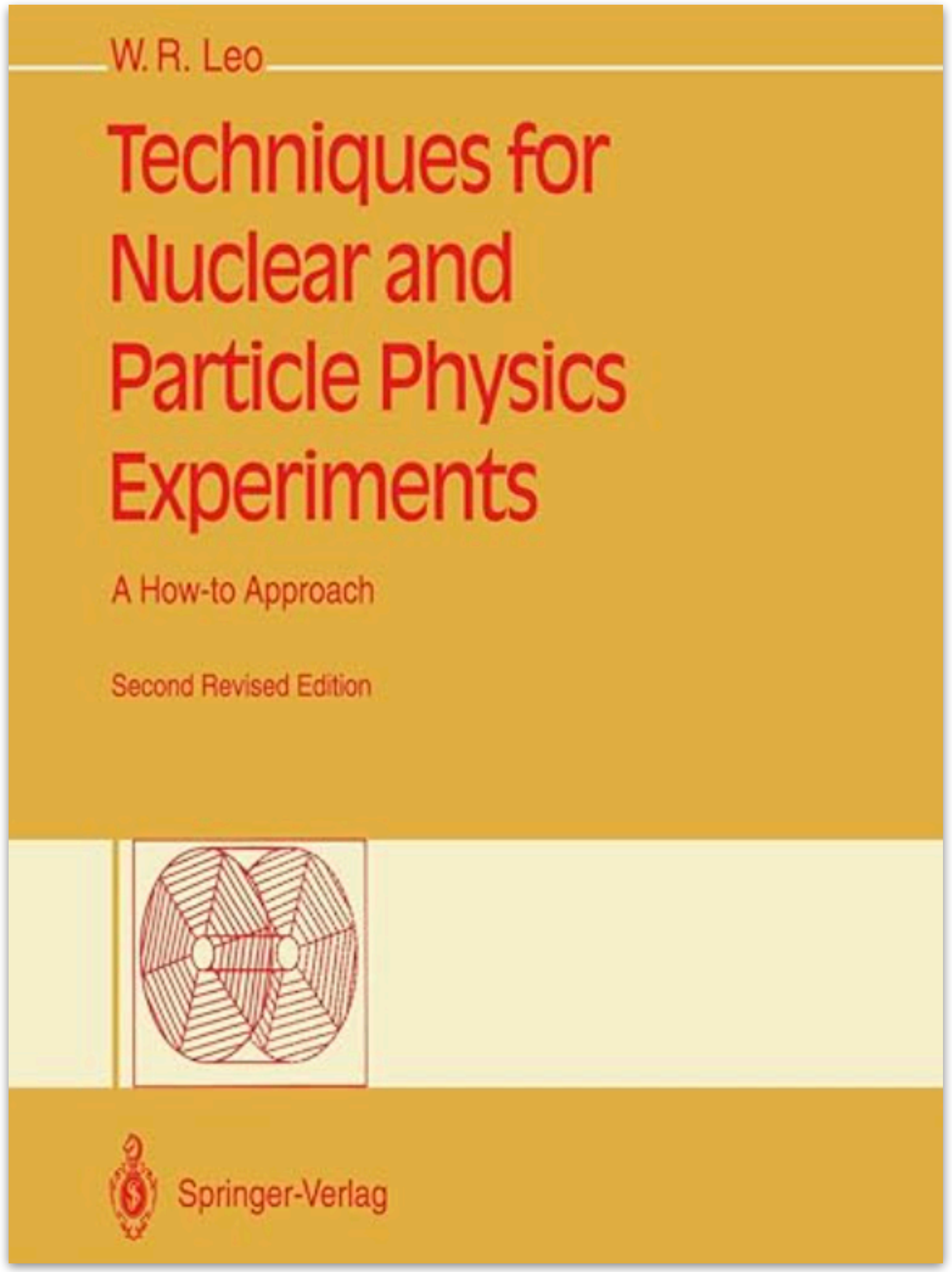
- AI/ML basics
- AI algorithms
- Linear regression
- Gradient Descent
- Neural Networks
- Autoencoder



- (only) one application (from A to Z)
- auto encoder for RT data reduction

https://colab.research.google.com/drive/1fSif01Wc6wXP_TyQdI3trReSf_4ltYph?usp=sharing#scrollTo=eLKiOiAvRUWA

Resources



ePIC Software & Computing Report

The ePIC Streaming Computing Model

Marco Battaglieri[1], Wouter Deconinck[2], Markus Diefenthaler[3], Jin Huang[4], Sylvester Joosten[5], Jeffery Landgraf[4], David Lawrence[3] and Torre Wenaus[4]
for the ePIC Collaboration

[1]Istituto Nazionale di Fisica Nucleare - Sezione di Genova, Genova, Liguria, Italy.
[2]University of Manitoba, Winnipeg, Manitoba, Canada.
[3]Jefferson Lab, Newport News, VA, USA.
[4]Brookhaven National Laboratory, Upton, NY, USA.
[5]Argonne National Laboratory, Lemont, IL, USA.

Abstract

This document provides a current view of the ePIC Streaming Computing Model. With datataking a decade in the future, the majority of the content should be seen largely as a proposed plan. The primary drivers for the document at this time are to establish a common understanding within the ePIC Collaboration on the streaming computing model, to provide input to the October 2023 ePIC Software & Computing review, and to the December 2023 EIC Resource Review Board meeting. The material should be regarded as a snapshot of an evolving document.

arXiv > physics > arXiv:1803.08823

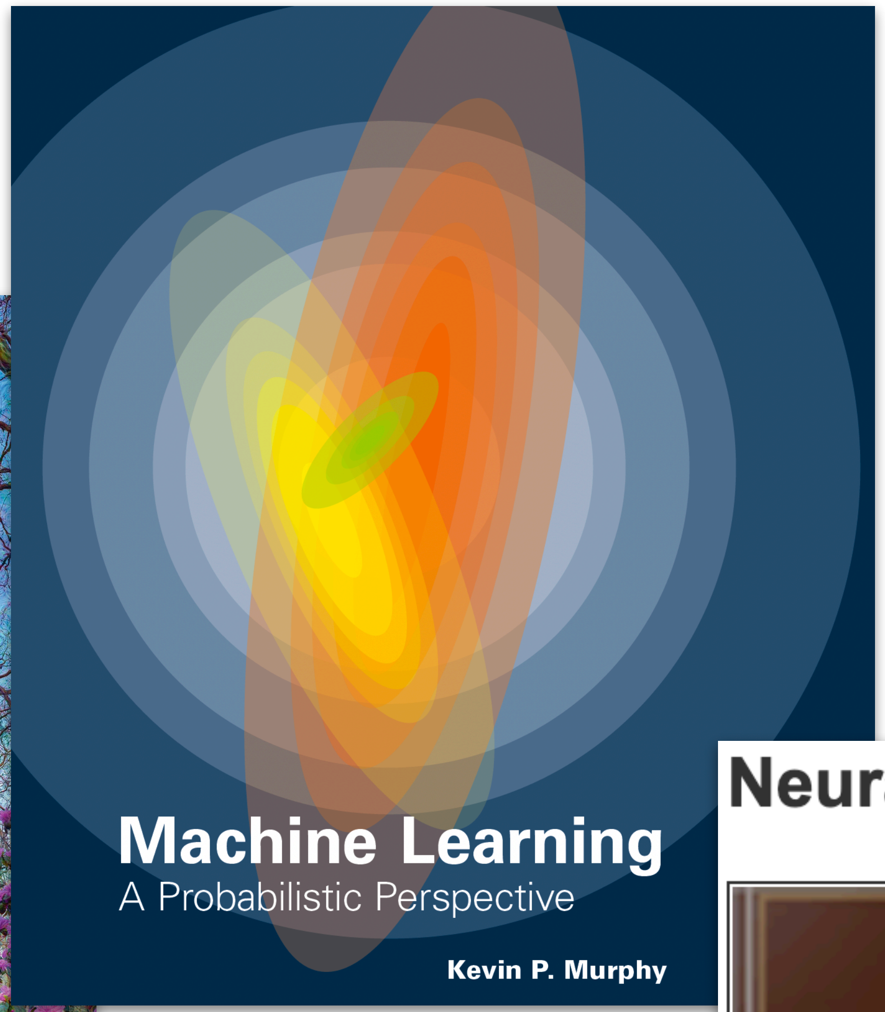
Physics > Computational Physics

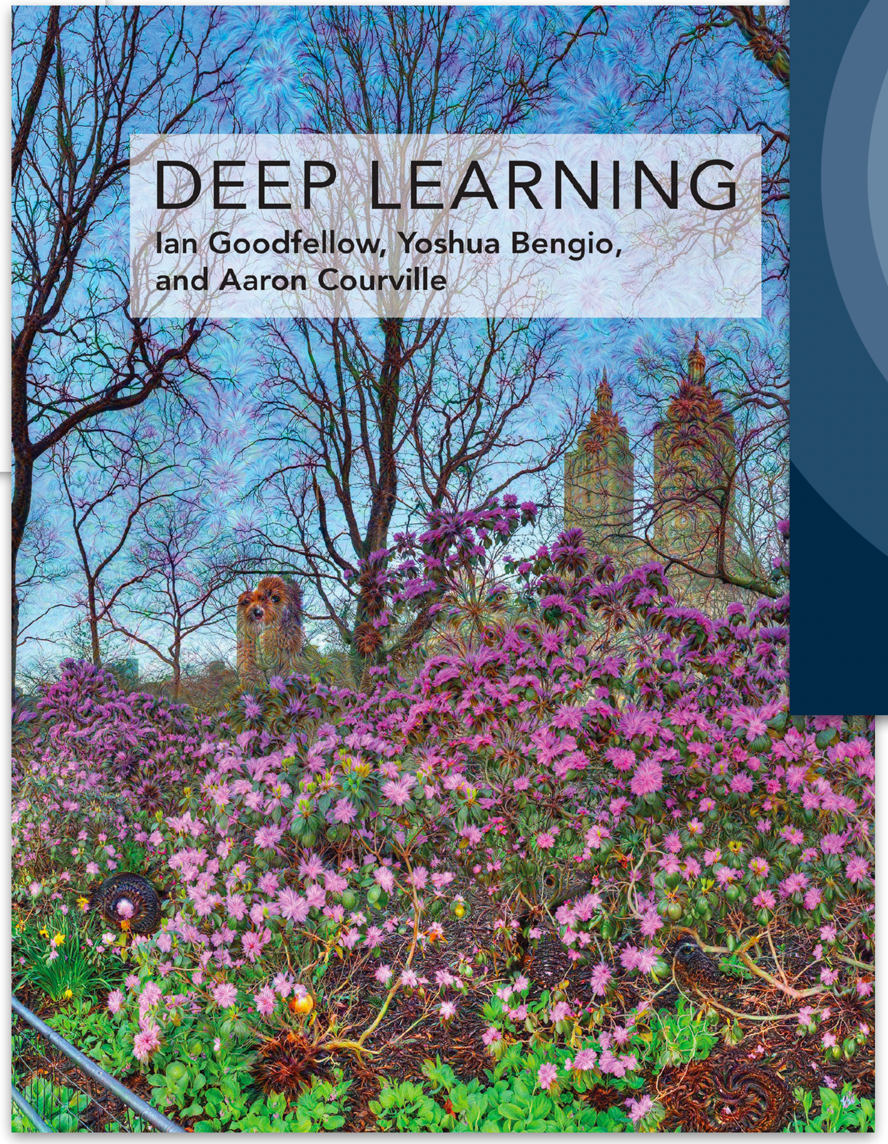
[Submitted on 23 Mar 2018 (v1), last revised 27 May 2019 (this version, v3)]

A high-bias, low-variance introduction to Machine Learning for physicists

Pankaj Mehta, Marin Bukov, Ching-Hao Wang, Alexandre G.R. Day, Clint Richardson, Charles K. Fisher, David J. Schwab

Machine Learning (ML) is one of the most exciting and dynamic areas of modern research and application. The purpose of this review is to provide an introductio concepts and tools of machine learning in a manner easily understood and intuitive to physicists. The review begins by covering fundamental concepts in ML and such as the bias-variance tradeoff, overfitting, regularization, generalization, and gradient descent before moving on to more advanced topics in both supervise learning. Topics covered in the review include ensemble models, deep learning and neural networks, clustering and data visualization, energy-based models (inc models and Restricted Boltzmann Machines), and variational methods. Throughout, we emphasize the many natural connections between ML and statistical phys of the review is the use of Python Jupyter notebooks to introduce modern ML/statistical packages to readers using physics-inspired datasets (the Ising Model an simulations of supersymmetric decays of proton-proton collisions). We conclude with an extended outlook discussing possible uses of machine learning for furt understanding of the physical world as well as open problems in ML where physicists may be able to contribute. (Notebooks are available at this https URL)





Neural Networks and Deep Learning

Michael A. Nielsen
Determination Press, 2015 - Back propagation (Artificial intelligence)
"Neural Networks and Deep Learning is a free online book. The book will teach you about: Neural networks, a beautiful
More »

Credits:
- Jin Huang, Jeff Landgraf, Markus Diefenthaler for ePIC SRO
- Fabio Rossi: (INFN-GE): author of JupyterNotebook exercise
- Cristiano Fanelli (W&M): material and pictures

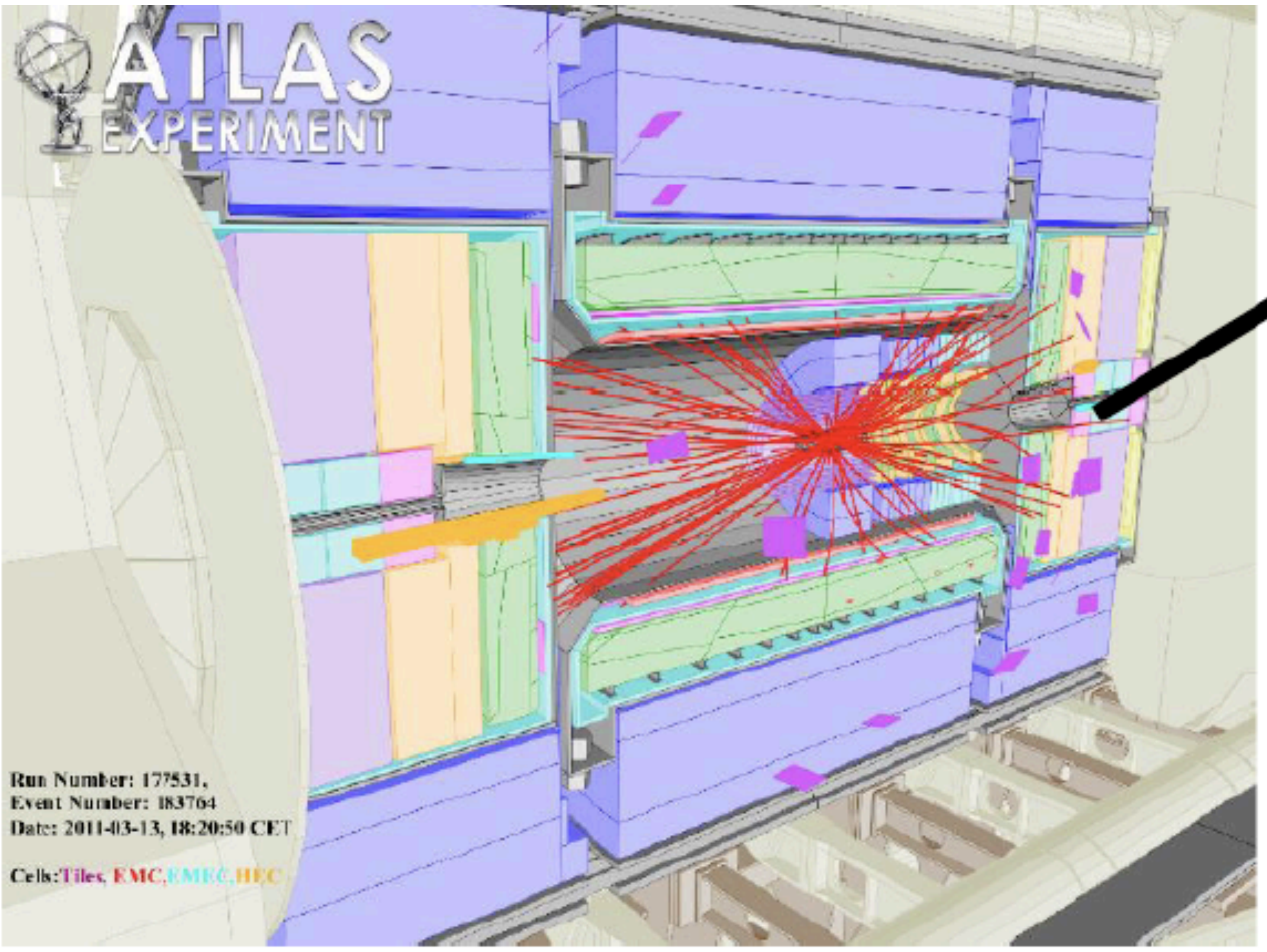
Outline

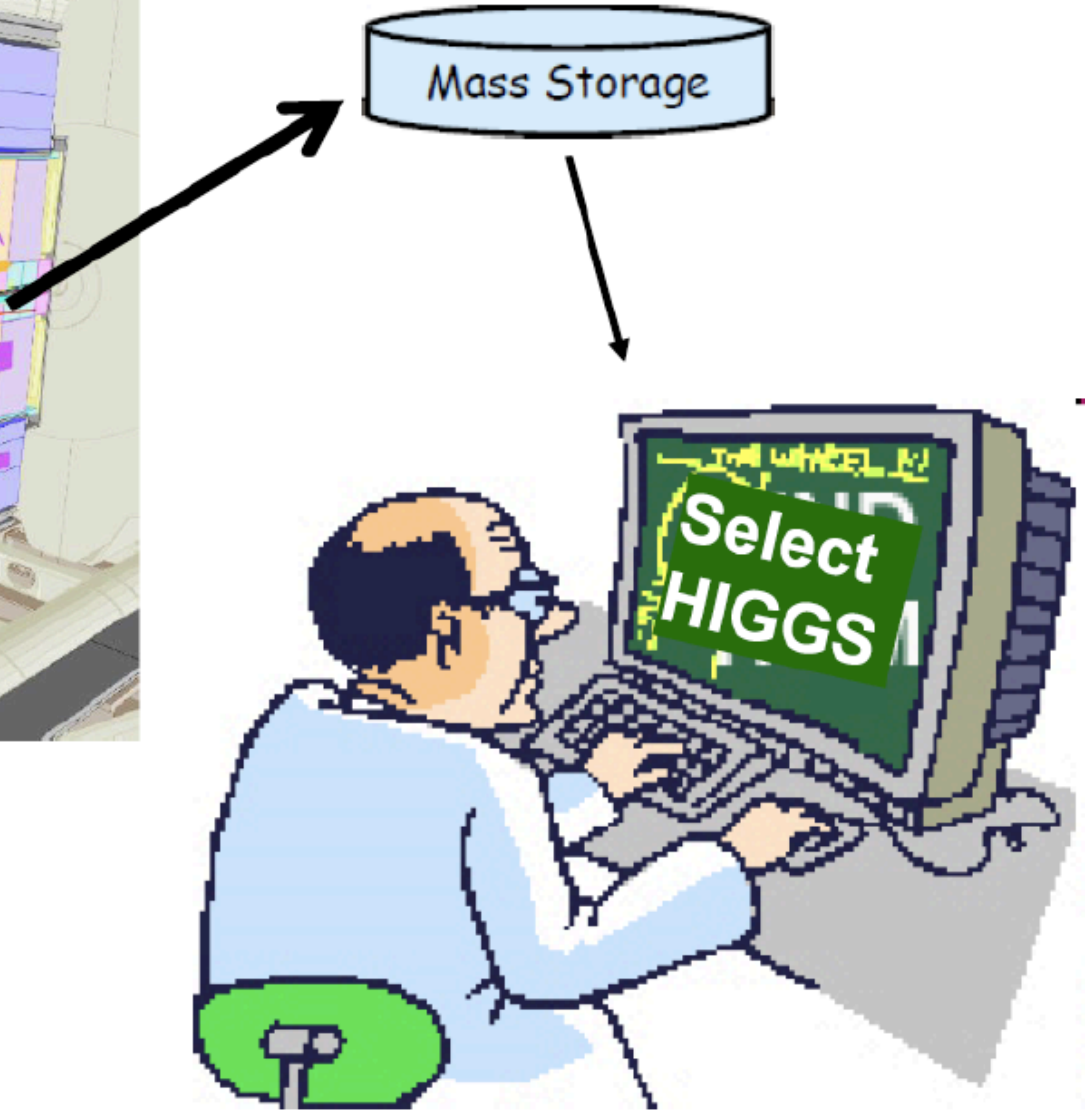
Part I

HEP/NP data and DAQ

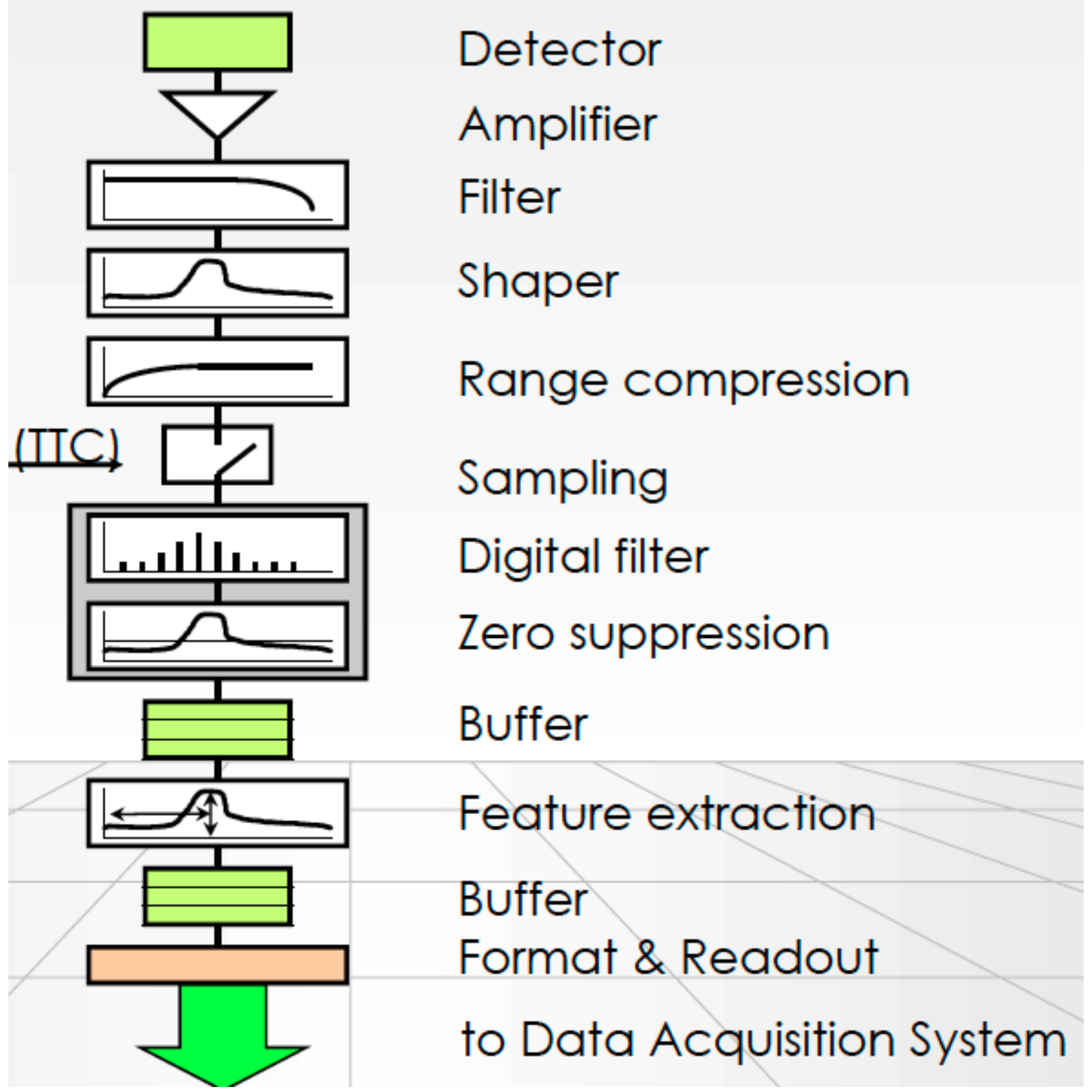
- Analog vs. digital
- Charge and time (features extraction)
- DAQ and streaming readout: triggered vs untriggered
- SRO requirements and opportunities
- An example: (future) ePIC@EIC (BNL) SRO scheme
- AI in real-time data analysis (clustering, tracking, calibration)
- Fast inference
- Data reduction

From detector's signals to physics





DAQ chain



Elementary particle detector

Particle

- Position: x, y, z
- Momentum: px, py, pz
- Energy: $E^2 = (M^2 + p_x^2 + p_y^2 + p_z^2)$
- Electric charge

Particle trough the detector

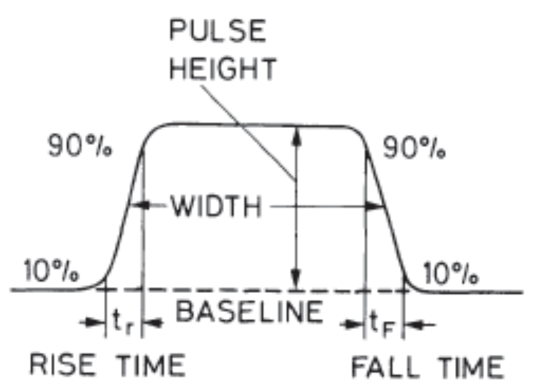
Detector

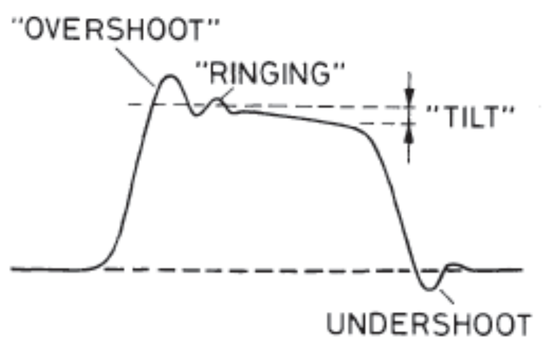
- Position
- Time
- Energy

★ Interaction TIME
★ Interaction POSITION
★ Deposited ENERGY

Signals in HEP and NP physics: analog vs. digital

Analog

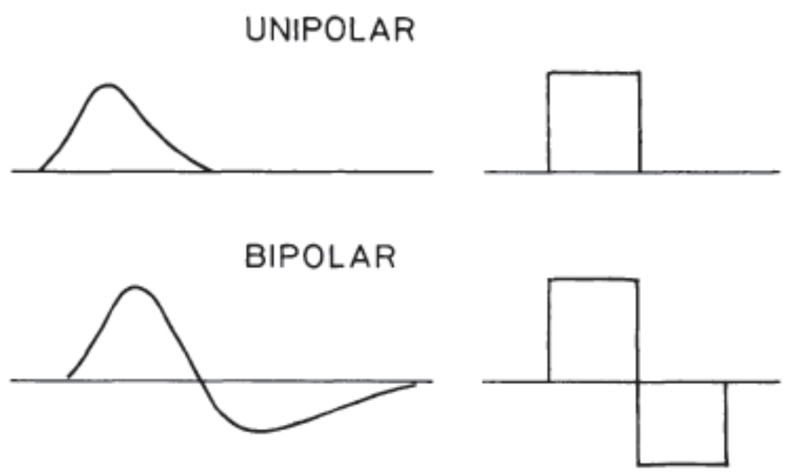




★ The Front End Electronics (FEE) produces analog signals. The information is coded into the signal shape

- PROS
  - coded info: height, length, shape …
- CONS
  - distortions imply a loss

Digital



★ Coding the info in a conventional 'pulse' provides a simpler and effective manipulation of signals

- PROS
  - distortion is not an issue
- CONS
  - coding requires a more complex elaboration and (often) a loss of information

★ The 'pulse' is the precursor of digital coding (Analog-to-digital)

★ Many different formats of 'standard pulse' with well-defined characteristics (within a certain range)

★ Each element in the data manipulation chain 'knows' about the input signal, and produces a well-defined (similar) output

★ Analog signals could be fast (rise time 10ps - 1ns) requiring a fast processing electronics (bandwidth > 1GHz)

★ CRATE (with a bus) + BOARDS for systems with <10k channels (NIM and VME standards are still in use)

★ Dedicated ASICS for large-scale experiments

★ Nowadays data acquisition is heading to streaming readout mode (the border between online and offline is less and less defined)

★ Modern Analog-to-Digital Converters (ADCs) convert analog signals to digitals at the FE

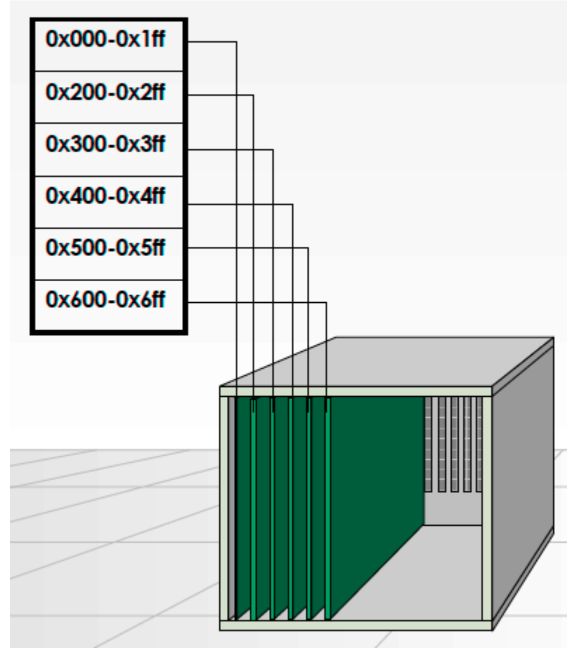
NIM

	Output must deliver	Input must accept
Logic 1	−14 mA to −18 mA	−12 mA to −36 mA
Logic 0	−1 mA to +1 mA	−4 mA to +20 mA

Current into 50 Ω
Neither risetime nor width is defined

- Preamplifiers, Shaping amplifier, Pulse-stracher, Fan-in fan-out, Dealy line, Discriminator, ADC, Logic unit, Scaler Gate and Delay generator, Time-to-Amplitude converter, Attenuator, Splitter, Converter, Filter, …

VME



- Bus-based architecture
- Three different devices: controller, master (write), slave (stream out)
- Addressing hardwired on each board
- Inverted logic (active-low)
- Max speed: ~200 MB/s

Particle detector's signals

Elementary particle detector

Particle

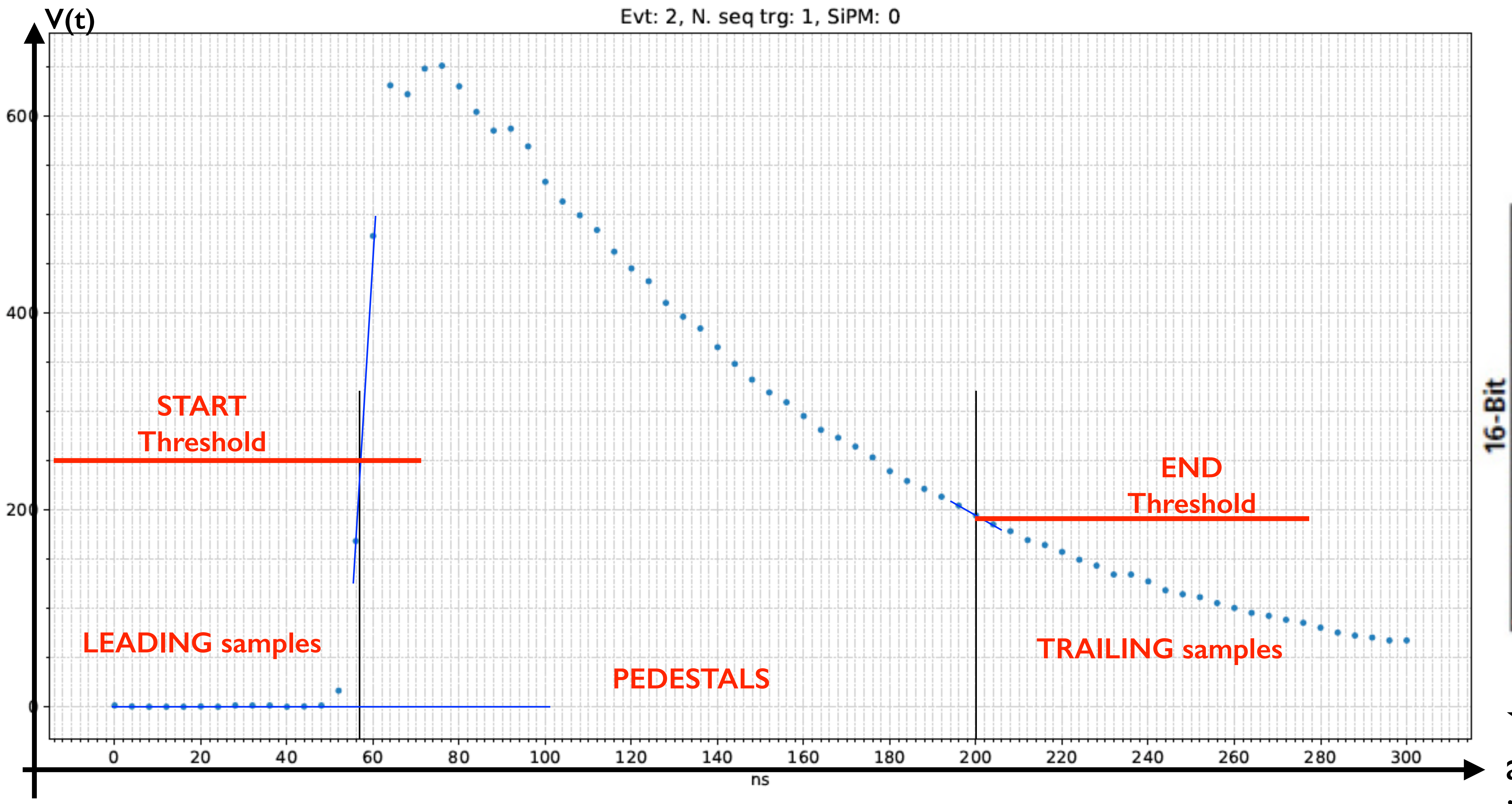
- Position: x, y, z
- Momentum: px, py, pz
- Energy: $E^2 = (M^2 + p_x^2 + p_y^2 + p_z^2)$
- Electric charge

Particle trough the detector

Detector

- Position
- Time
- Energy

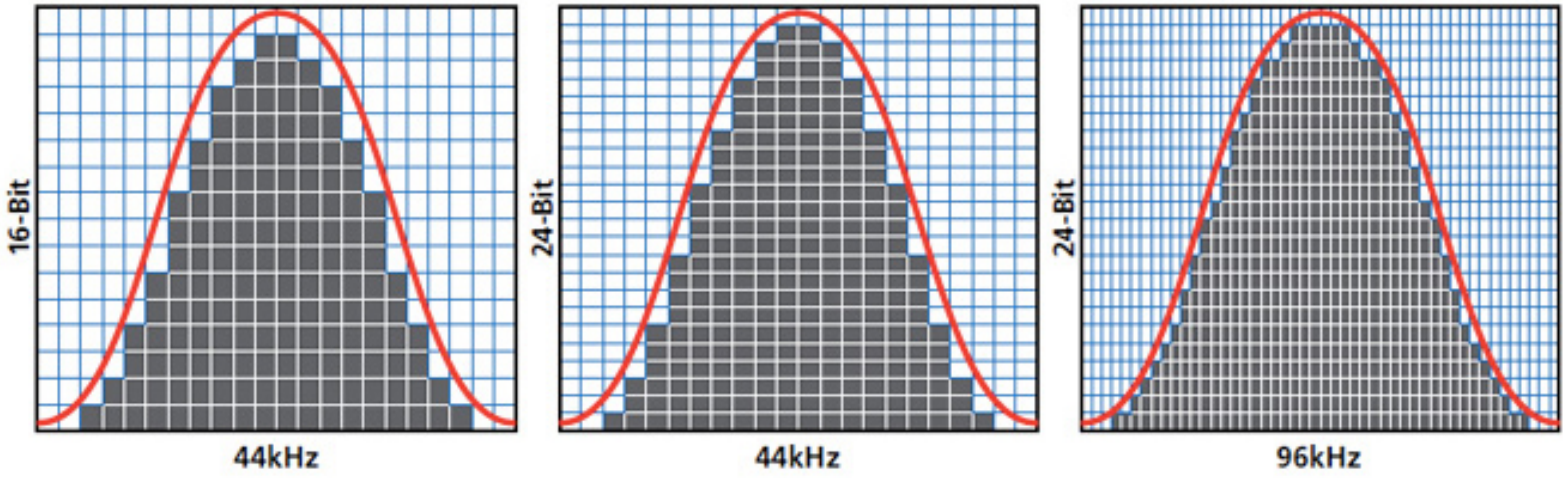
★ Interaction TIME
★ Interaction POSITION
★ Deposited ENERGY



Typical signal

★ Full waveform contains information about the shape of the signal

★ Modern digitizers sample the waveform at a high rate and high definition (bits)



★ CHARGE is measured by summing up samples (within a certain time window) or applying a more sophisticated algorithms (e.g. fitting the wf and integrating the curve or interpolating samples)

$Q = \mathrm{Sum}_{[i=1,N_{sample}]} (\mathrm{Ampl}_i \times 4\mathrm{ns}) / 50\ \mathrm{Ohm}$

Particle detector’s signals

Elementary particle detector

Particle

- Position: x, y, z
- Momentum: px, py, pz
- Energy: $E^2 = (M^2 + p_x^2 + p_y^2 + p_z^2)$
- Electric charge

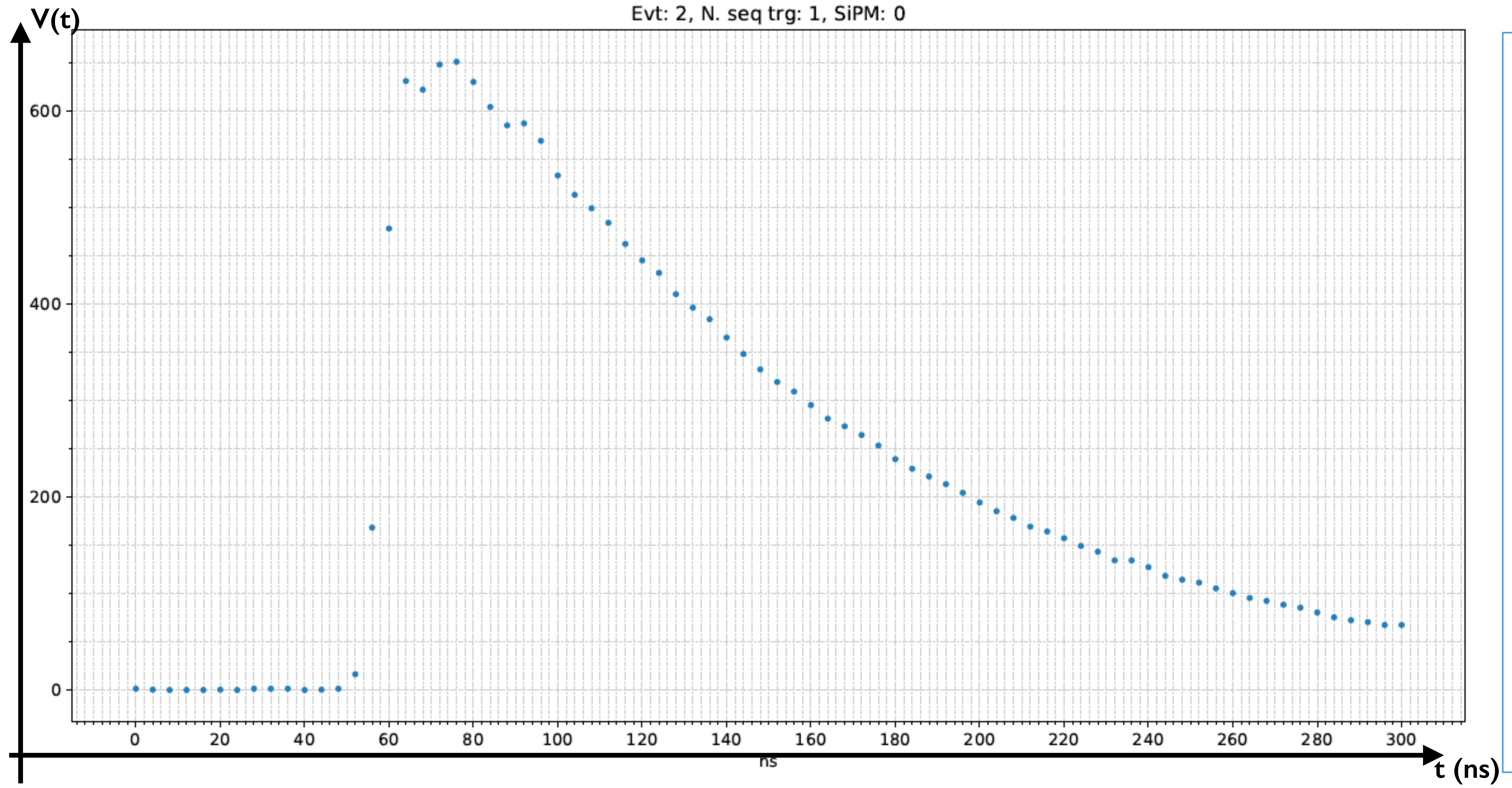
⟺ Particle trough the detector

Detector

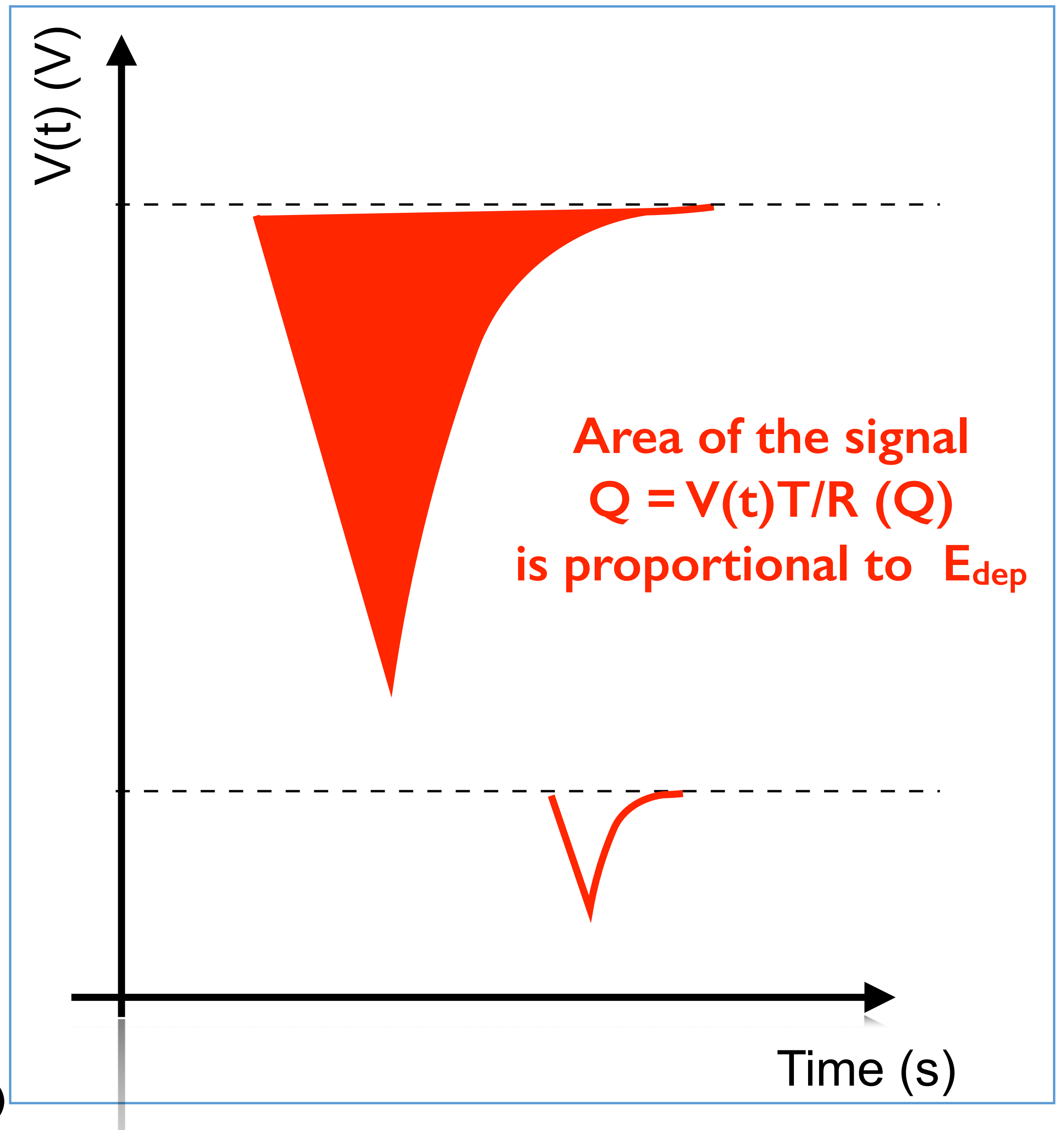
- Position
- Time
- Energy

★ Interaction TIME
★ Interaction POSITION
★ Deposited ENERGY

★ Deposited ENERGY ⇔ CHARGE



Typical signal



$$E_{dep} \sim Q = \int V(t)dt \,/\, R\ (50\Omega)$$

Particle detector's signals

Elementary particle detector

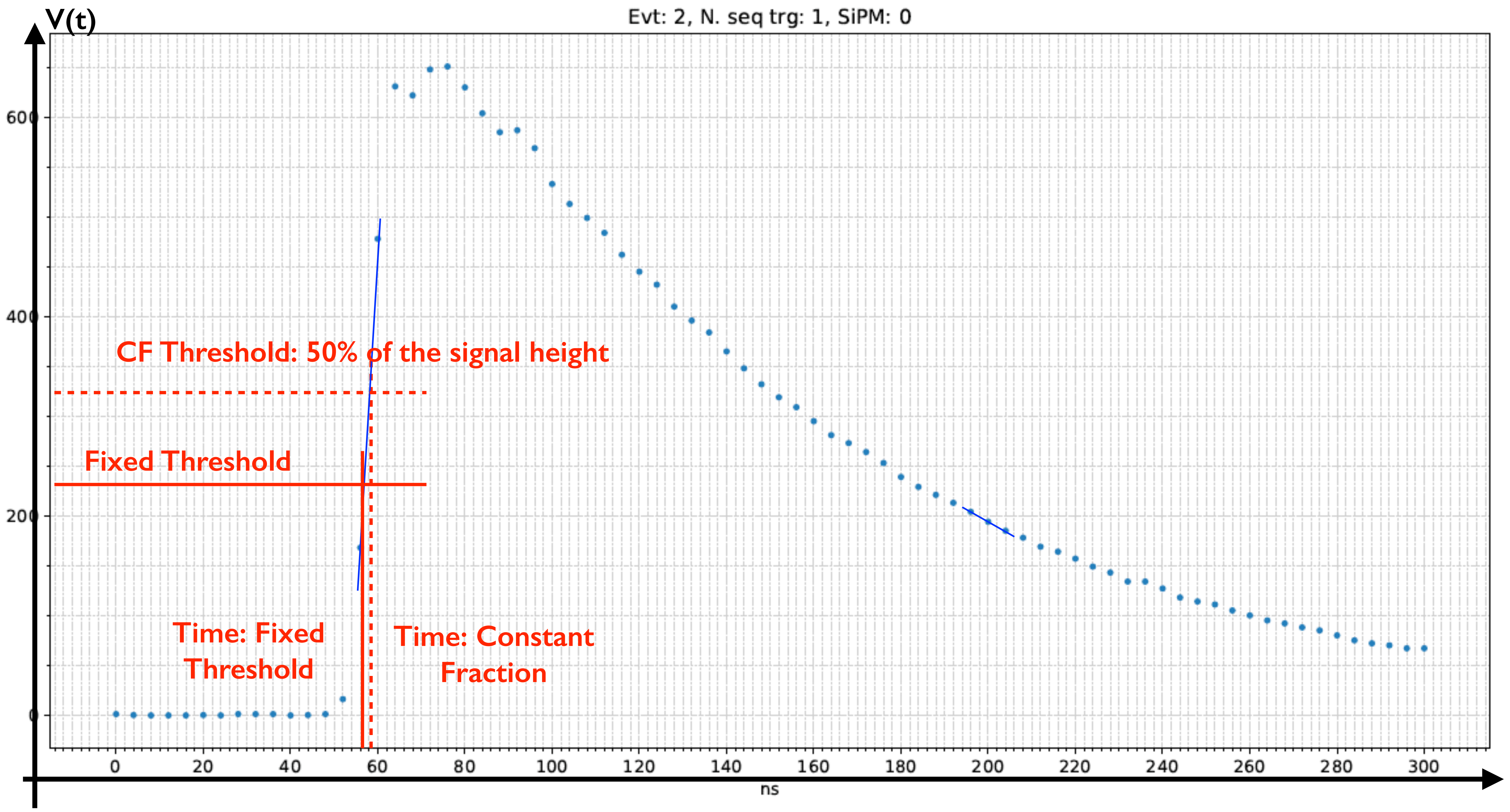
Particle
- Position: x, y, z
- Momentum: px, py, pz
- Energy: $E^2 = (M^2 + p_x^2 + p_y^2 + p_z^2)$
- Electric charge

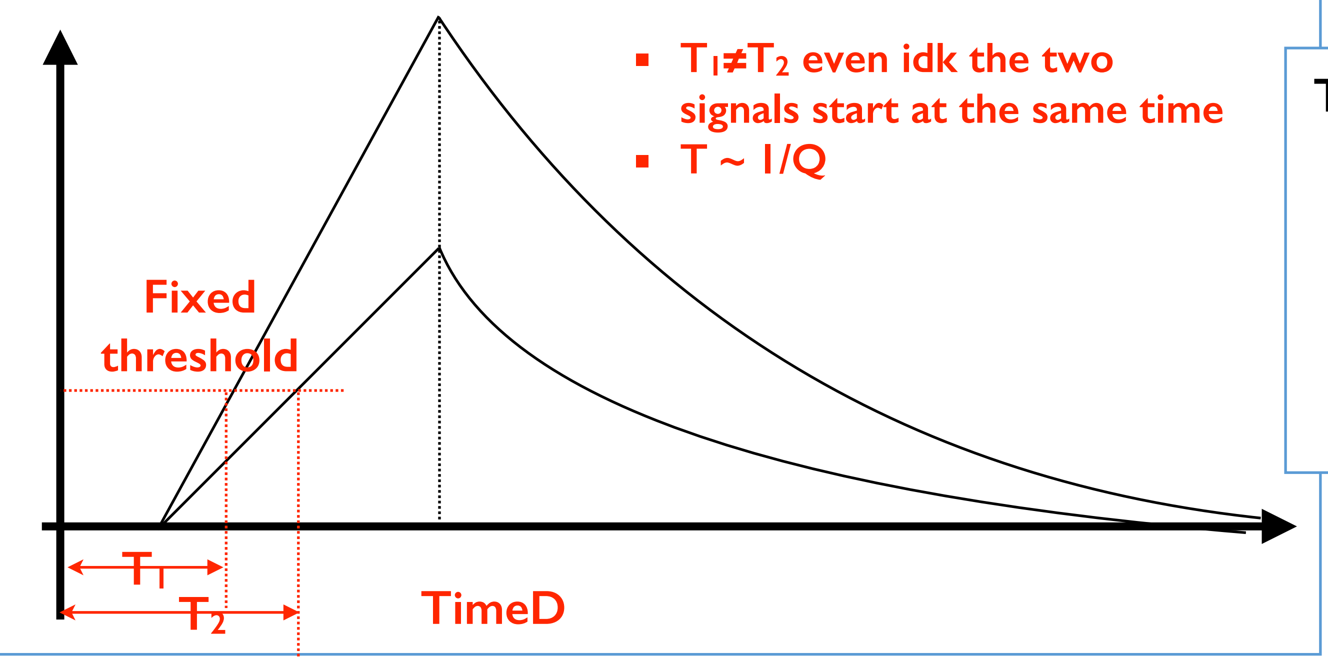
Particle trough the detector

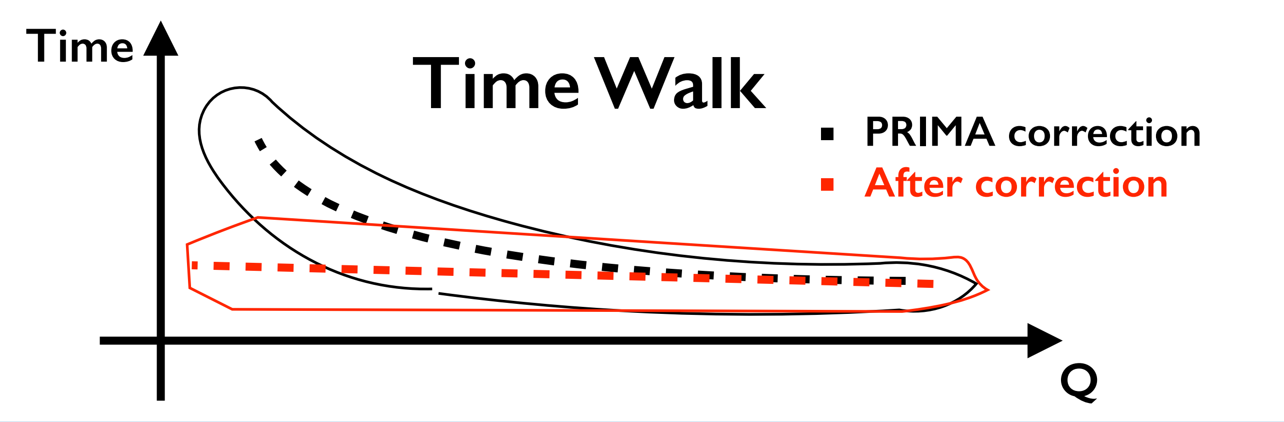
Detector
- Position
- Time
- Energy

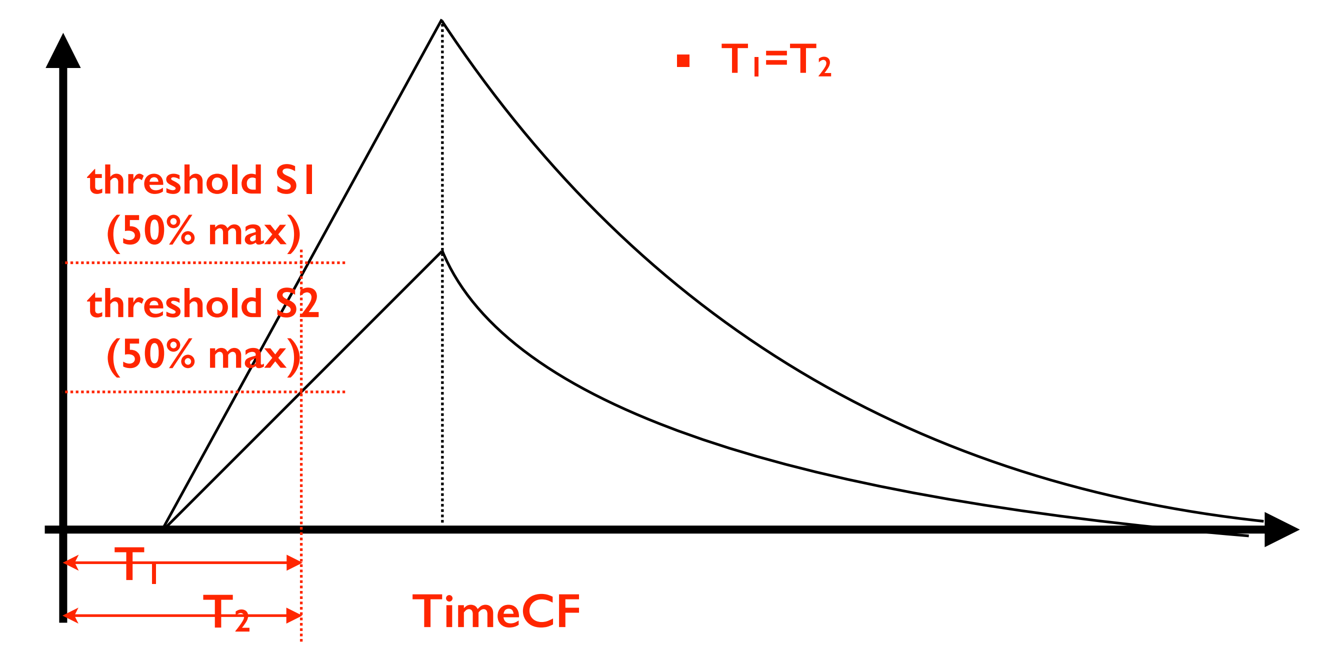
★ Interaction TIME
★ Interaction POSITION
★ Deposited ENERGY

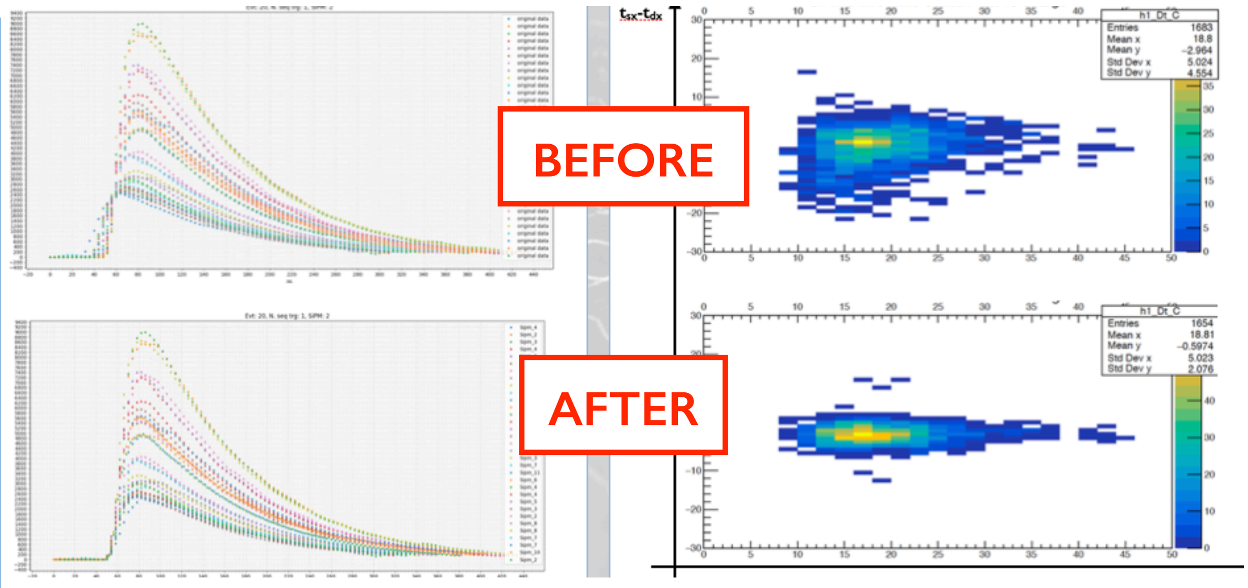
★ Interaction TIME ⇔ Threshold





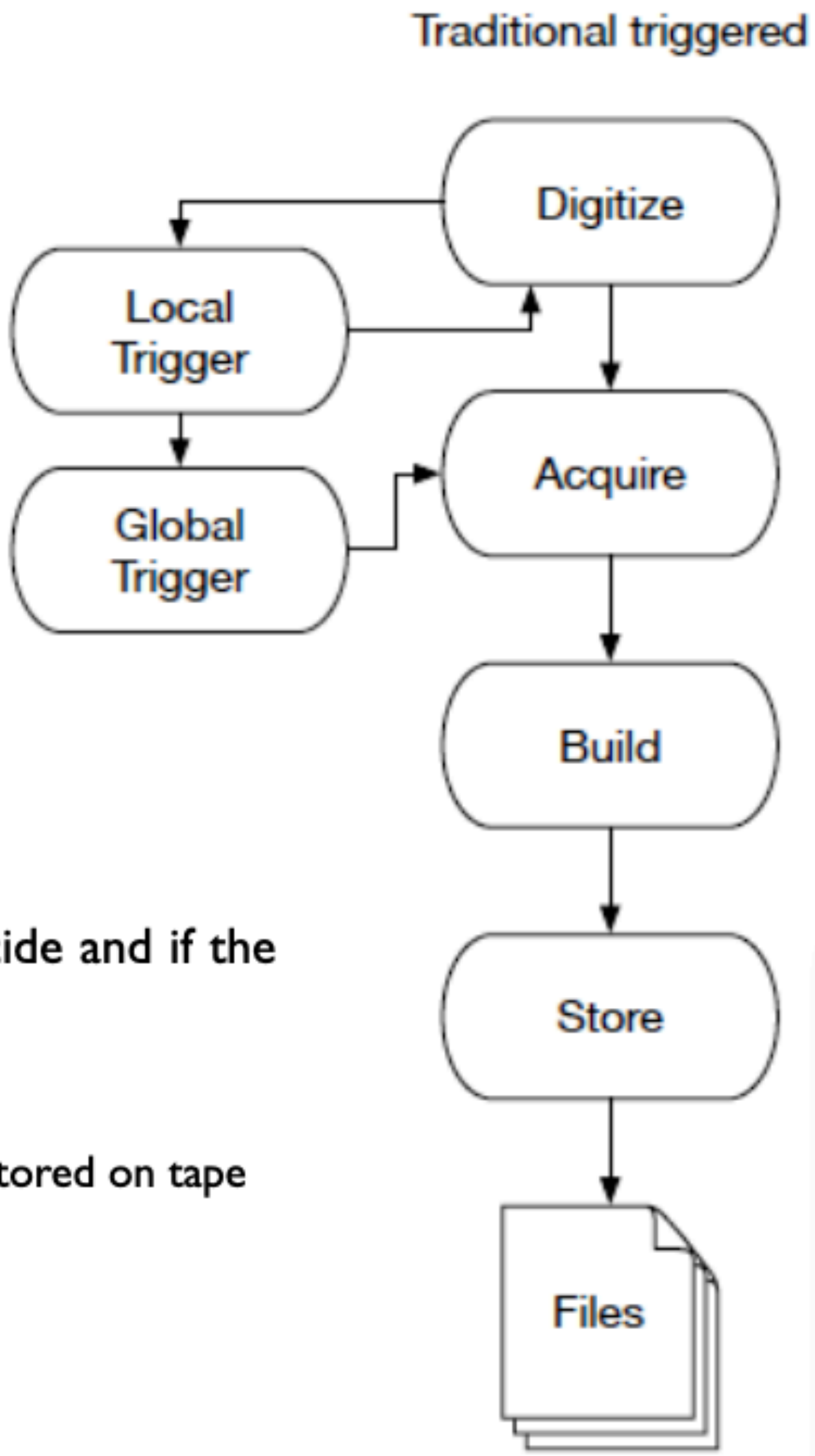






Triggered DAQ

Traditional (triggered) DAQ



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

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

✳ 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

Trigger logic

★ decides if/when to collect detector information

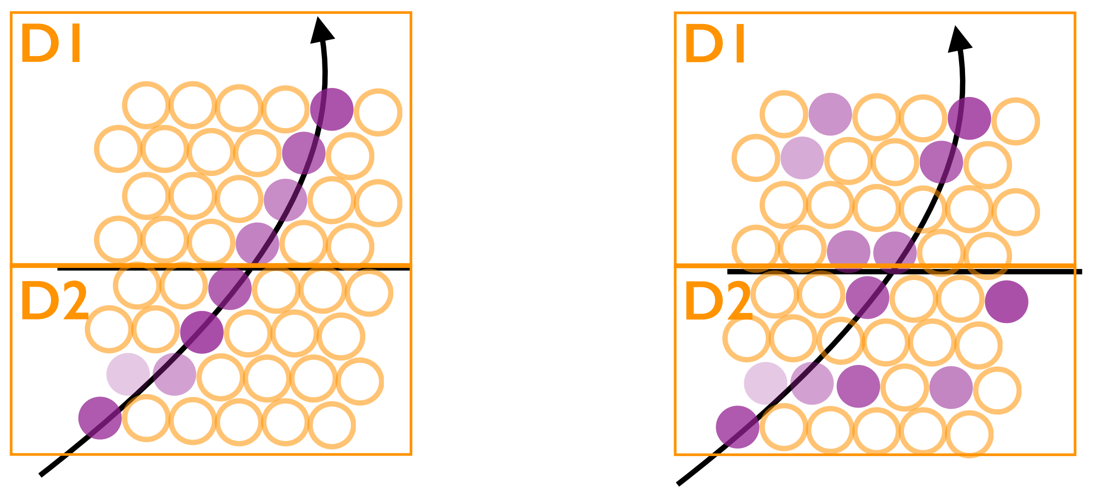
★ Select 'events' over 'background'

★ Save data on disk for further processing

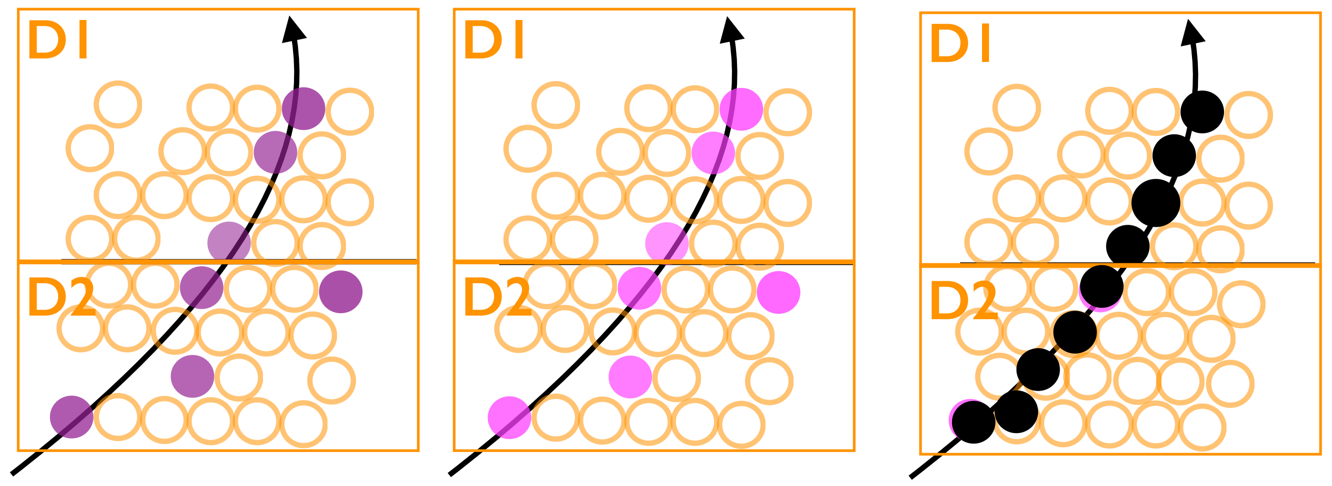
★ Different levels

- L1: threshold on FEE
- L2: combine information from different sub-detector components
- L3: requires info processing

'True' | Real (true+noise)

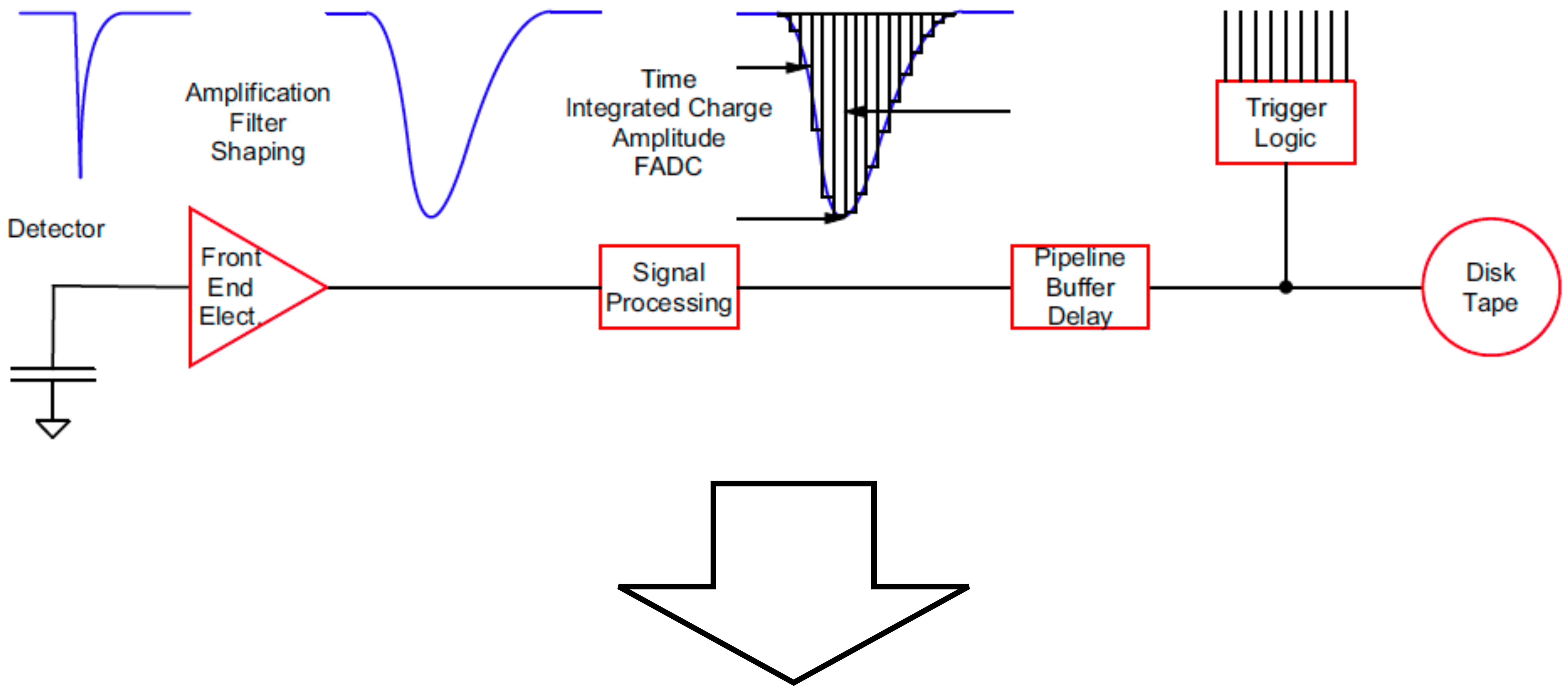


L1: threshold hits | L2: D1+D2 clusters | L3: clusters track

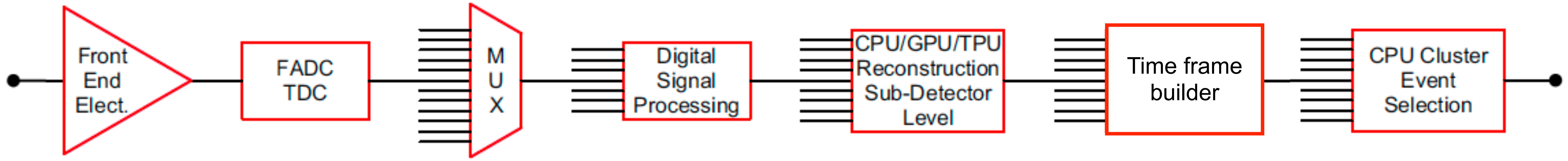


Streaming readout

Triggered DAQ

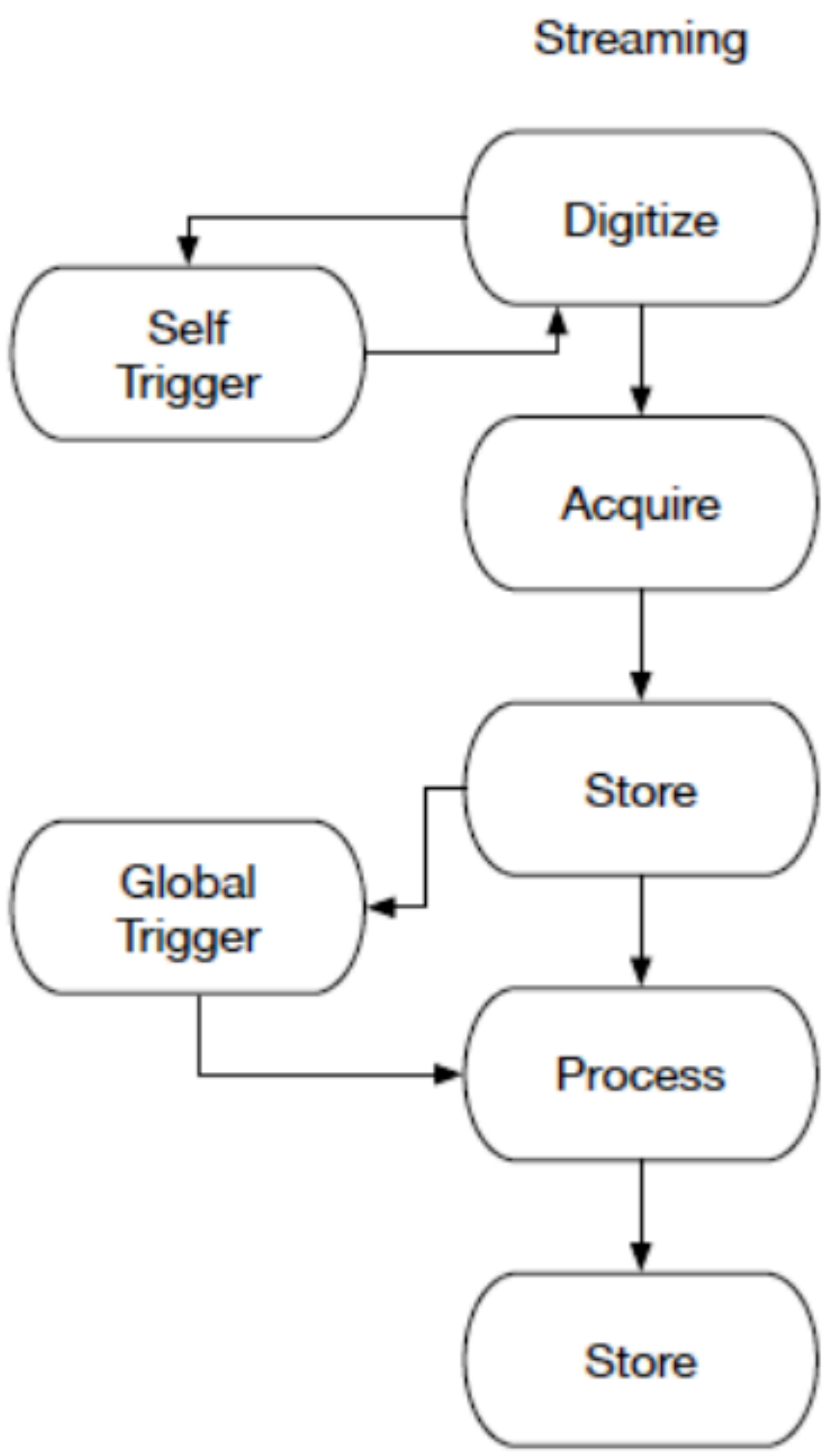


Streaming readout DAQ



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)

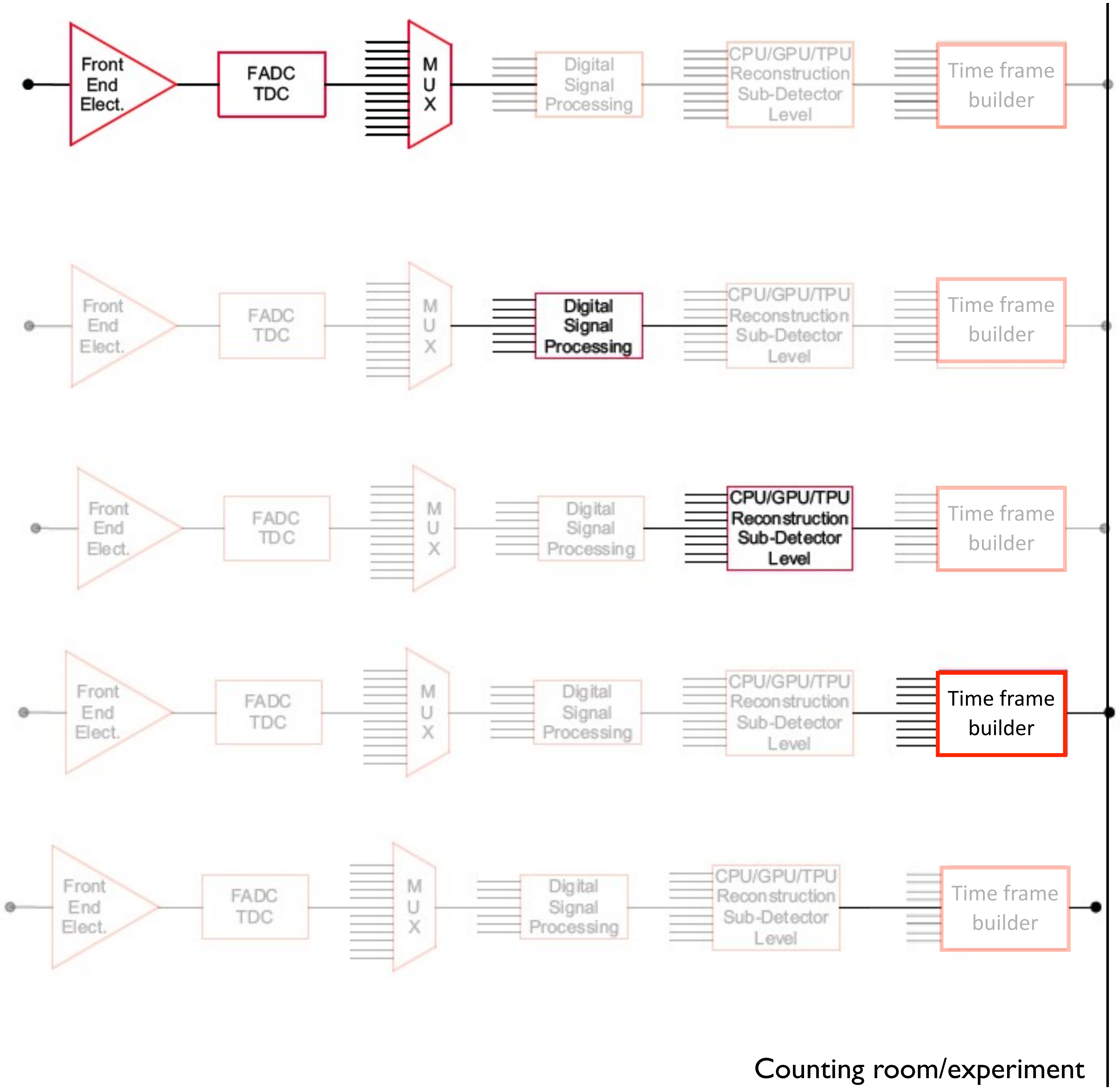
✳ **Shifting data tagging/filtering from the front-end (hw) to the back-end (sw)**
- Optimize real-time rare/exclusive channel selection
- Use of high-level programming languages
- Use of existing/ad-hoc CPU/GPU farms
- Use of available AI/ML tools
- (future) use of quantum-computing

✳ **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.ePIC
- JLAB: SOLID, BDX, CLAS12, …

Streaming RO

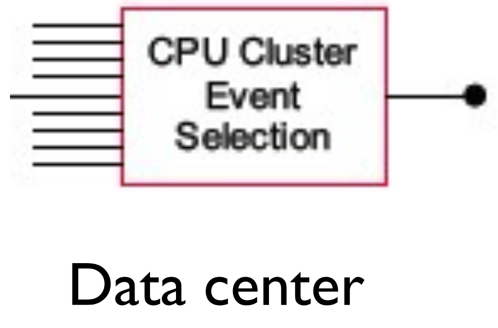


- FEE optimised for SRO
  - ASICS (cheap) or fADC (multiplexing) at (O($10/ch)
  - TDC if necessary to replace fADC
  - Zero-suppression mode
  - Fast readout (optical link)

- 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

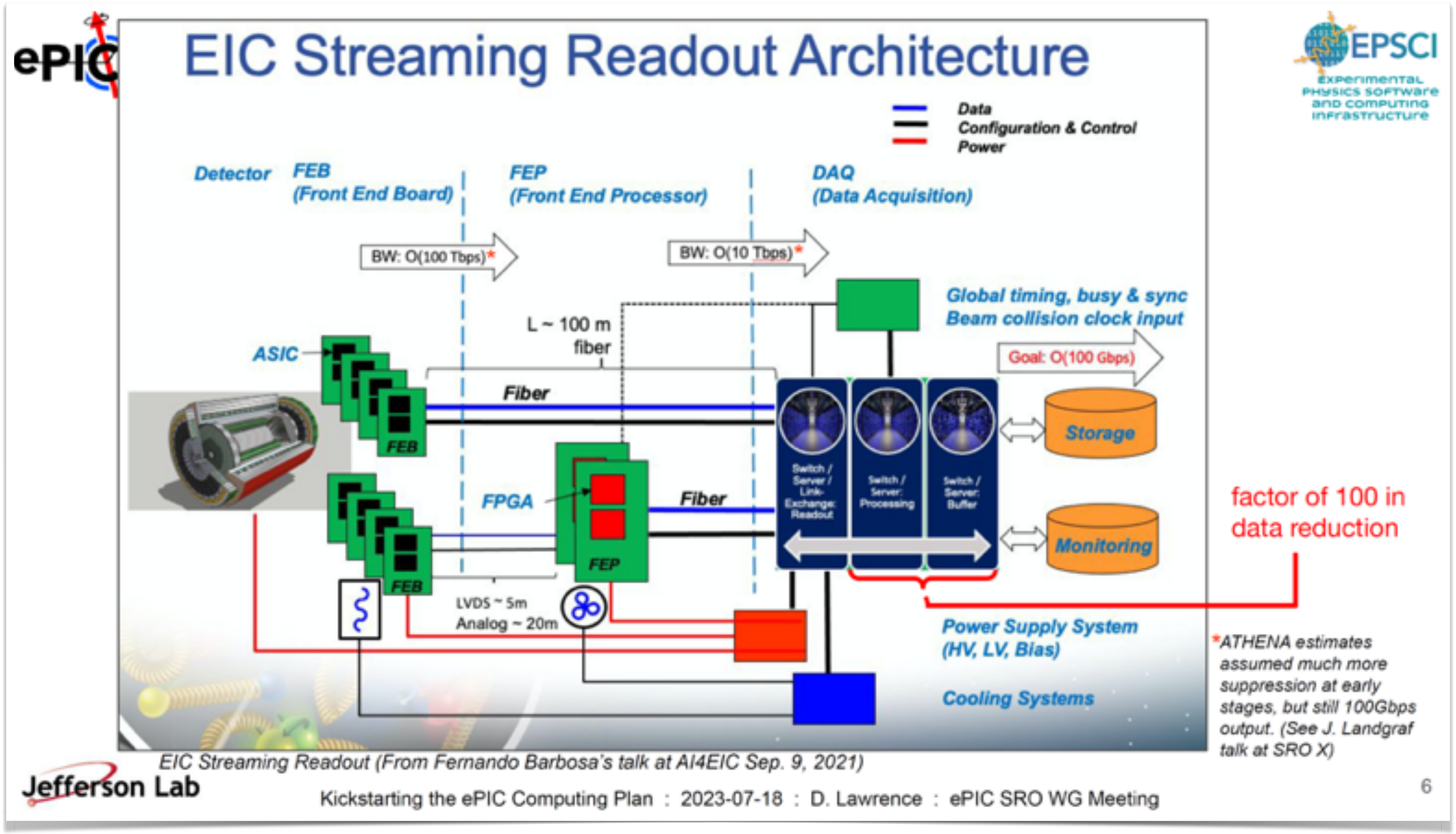
- TF-Router Time frame construction
  - Use time stamps to reorganise data from all streams in time frames



- Full reconstruction CPU analysis (for each time frame)

AI/ML shall play a significant role in each of these steps

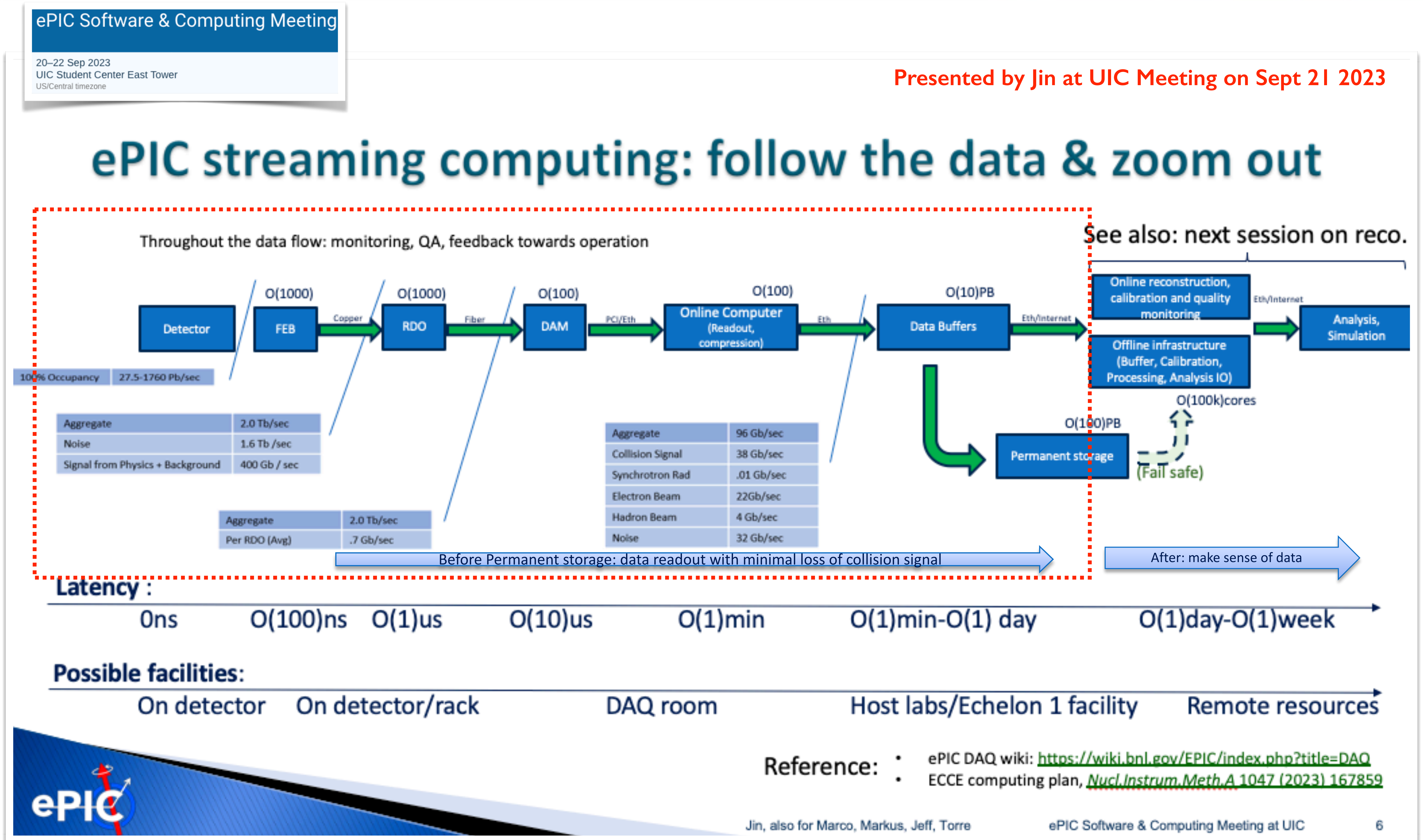
ePIC Streaming Computing



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 shall use AI:
  - to adapt data analysis to the changed conditions of the run (e.g. thresholds)
  - to identify data features in real-time (e.g.clusters)
  - to extract calibration constants from a data sub-set
  - to 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

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 shall use AI:
  - to adapt data analysis to the changed conditions of the run (e.g. thresholds)
  - to identify data features in real-time (e.g.clusters)
  - to extract calibration constants from a data sub-set
  - to 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

AI/ML Autoencoder

Clustering

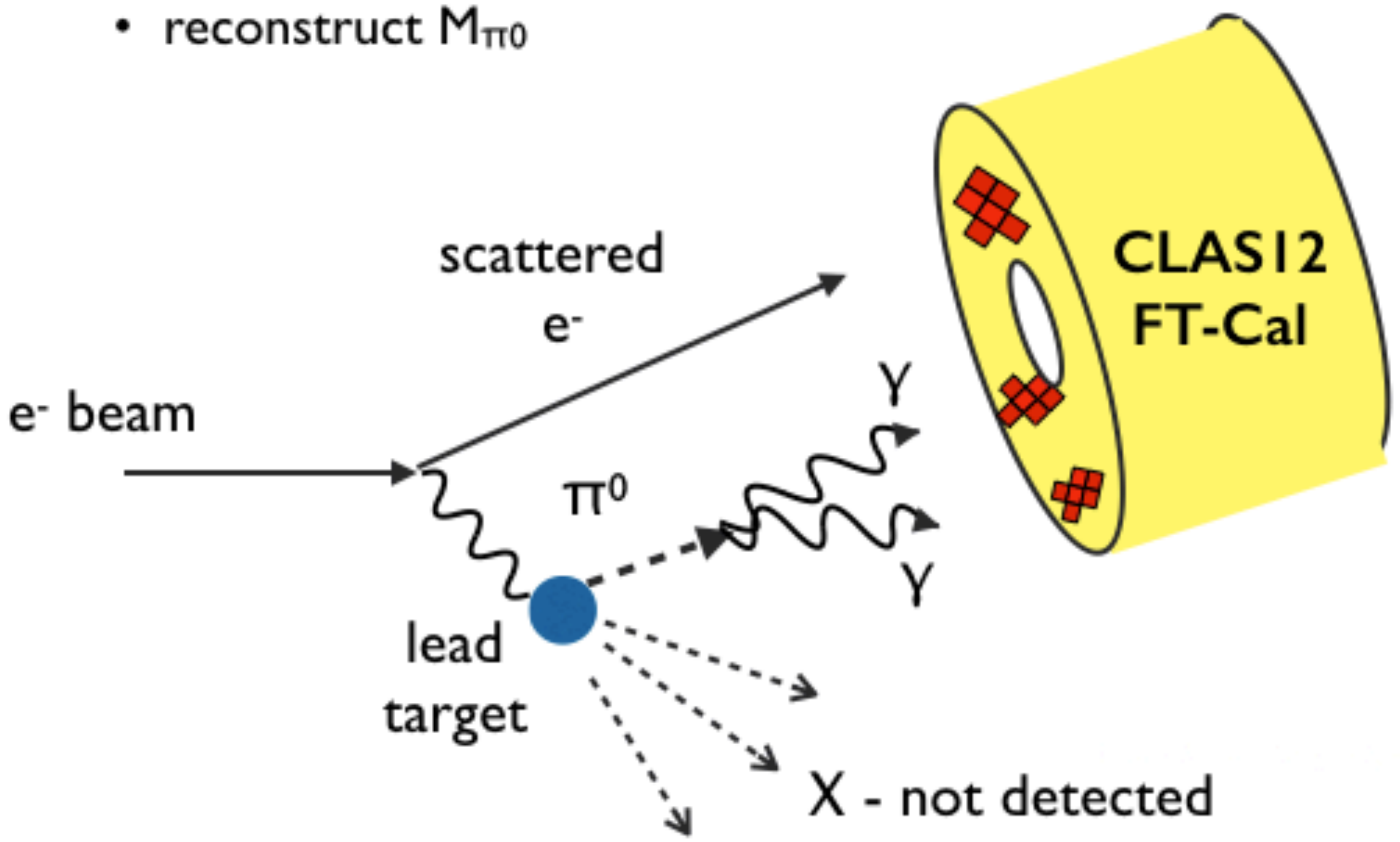
Goal: real-time π^0 identification

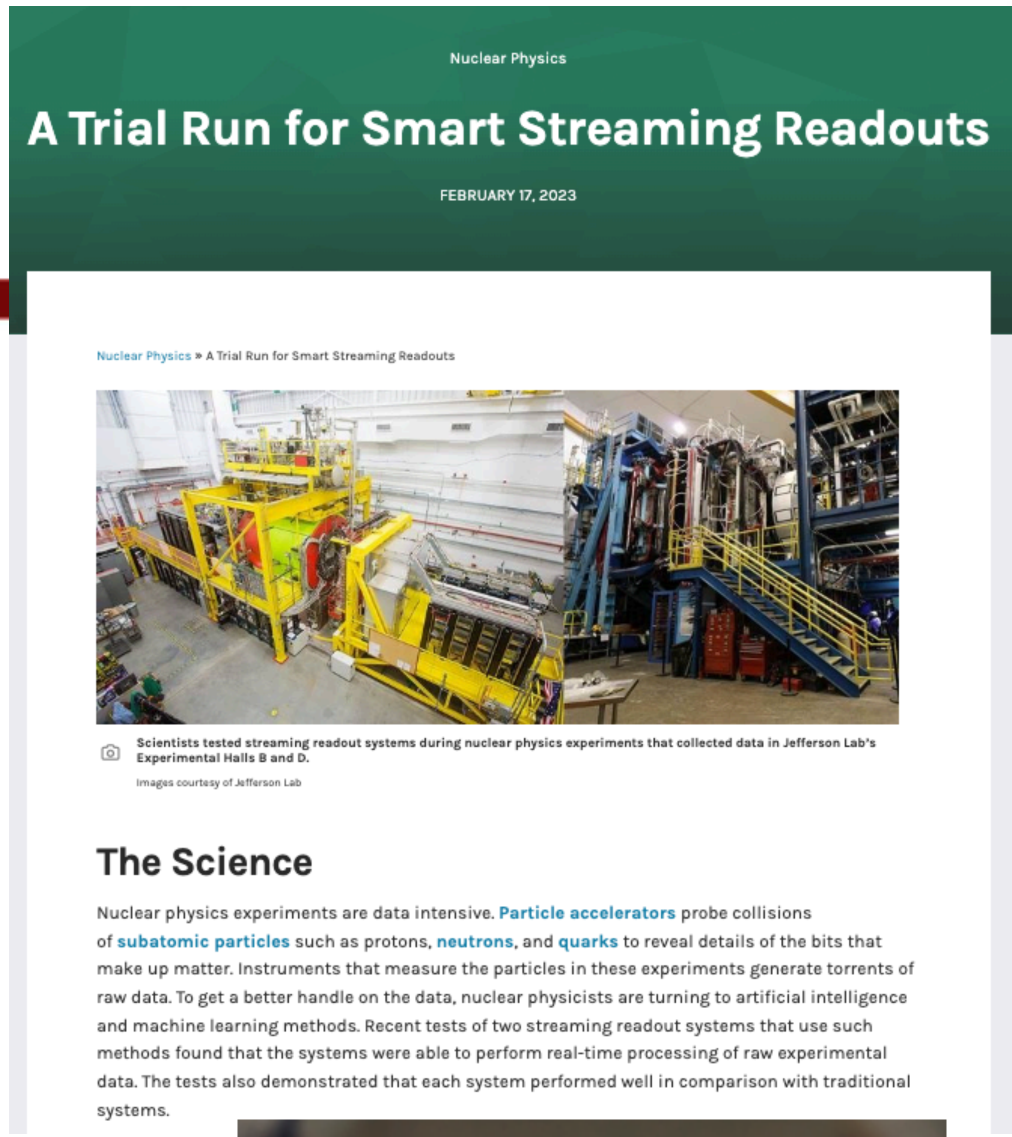
$$m_{\pi^0}^2 = 2E_1E_2(1-\cos\eta)$$

- $\pi^0 \rightarrow \gamma_1 + \gamma_2$
- E1 and E2 = γ's energies
- η = opening angle

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

- reconstruct $M_{\pi 0}$




Nuclear Physics

A Trial Run for Smart Streaming Readouts

FEBRUARY 17, 2023

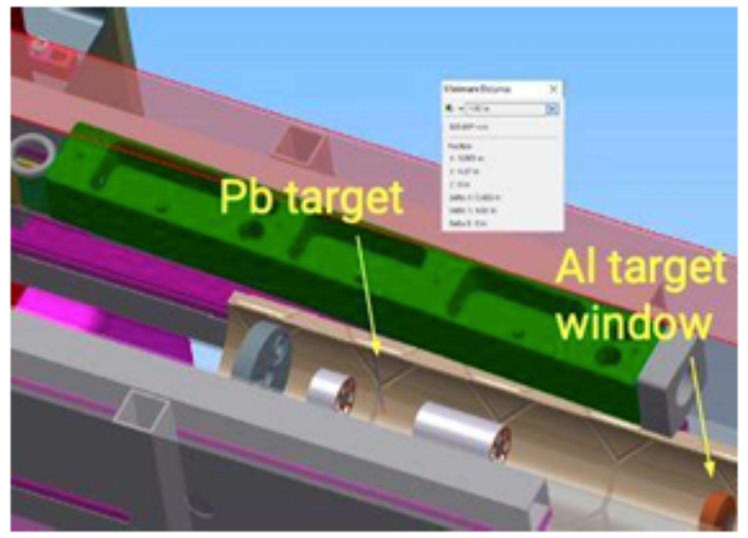
Nuclear Physics » A Trial Run for Smart Streaming Readouts

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

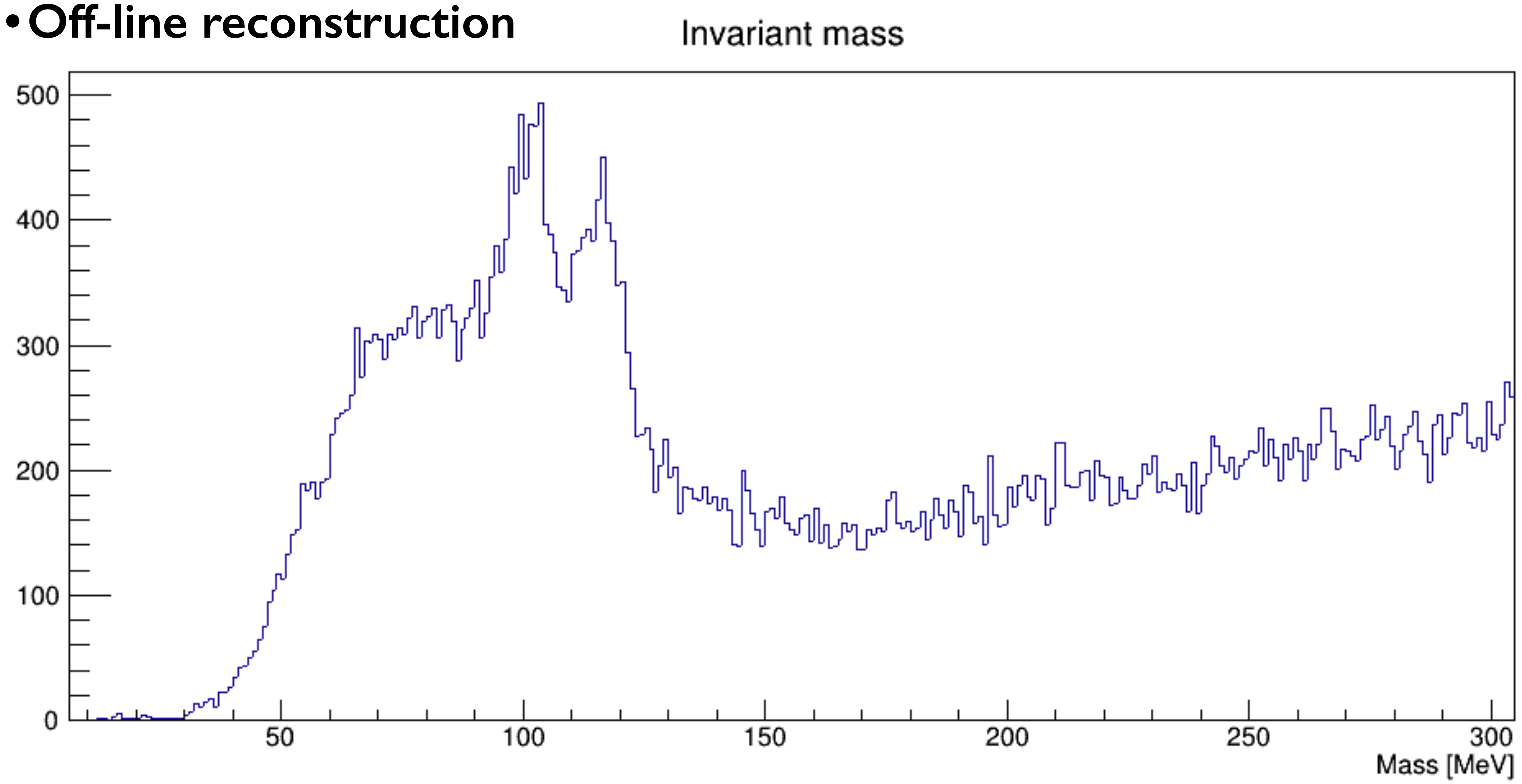
Images courtesy of Jefferson Lab

The Science

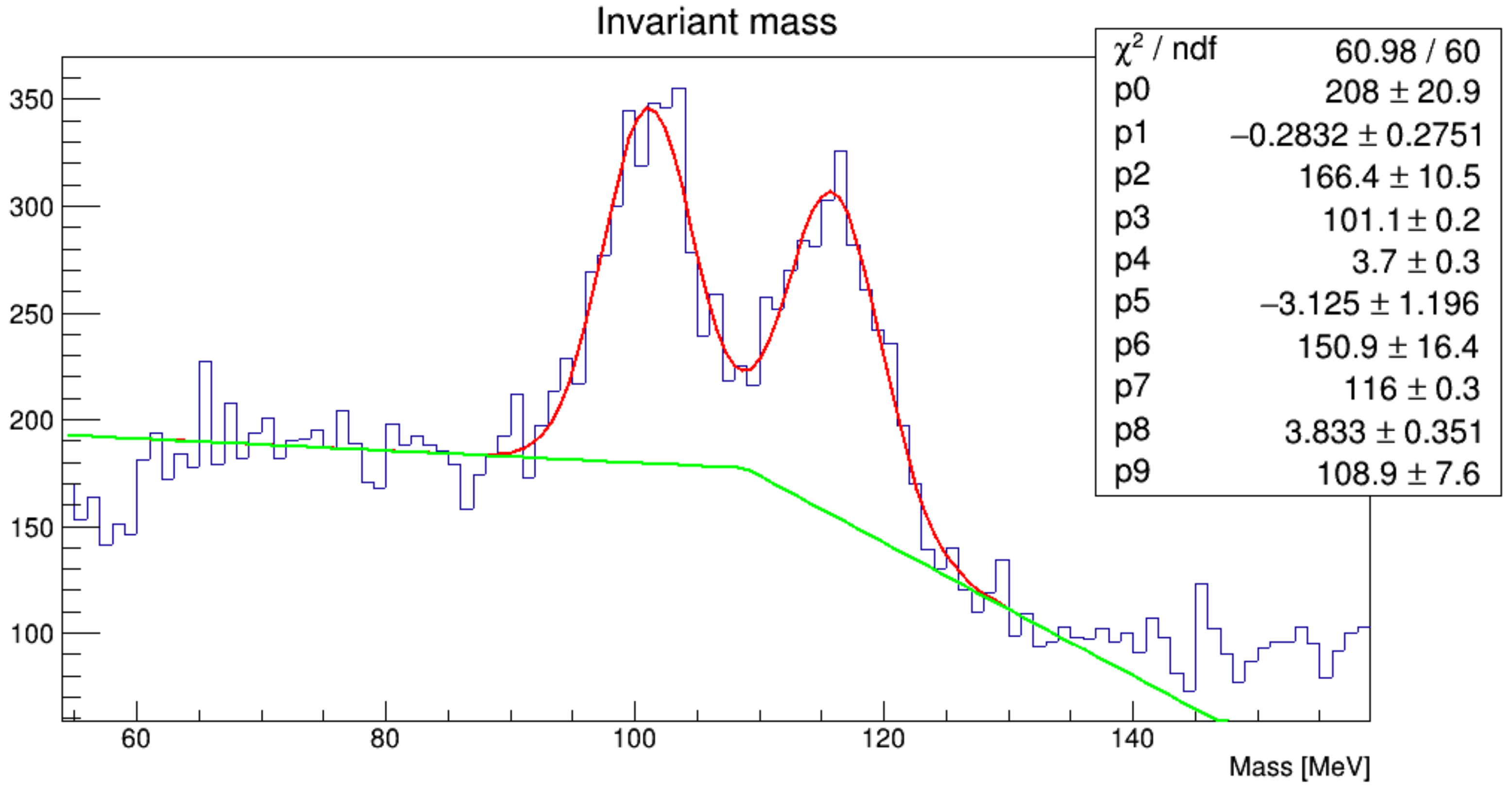
Nuclear physics experiments are data intensive. Particle accelerators probe collisions of subatomic particles such as protons, neutrons, and quarks to reveal details of the bits that make up matter. Instruments that measure the particles in these experiments generate torrents of raw data. To get a better 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.



- **Off-line reconstruction**



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



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

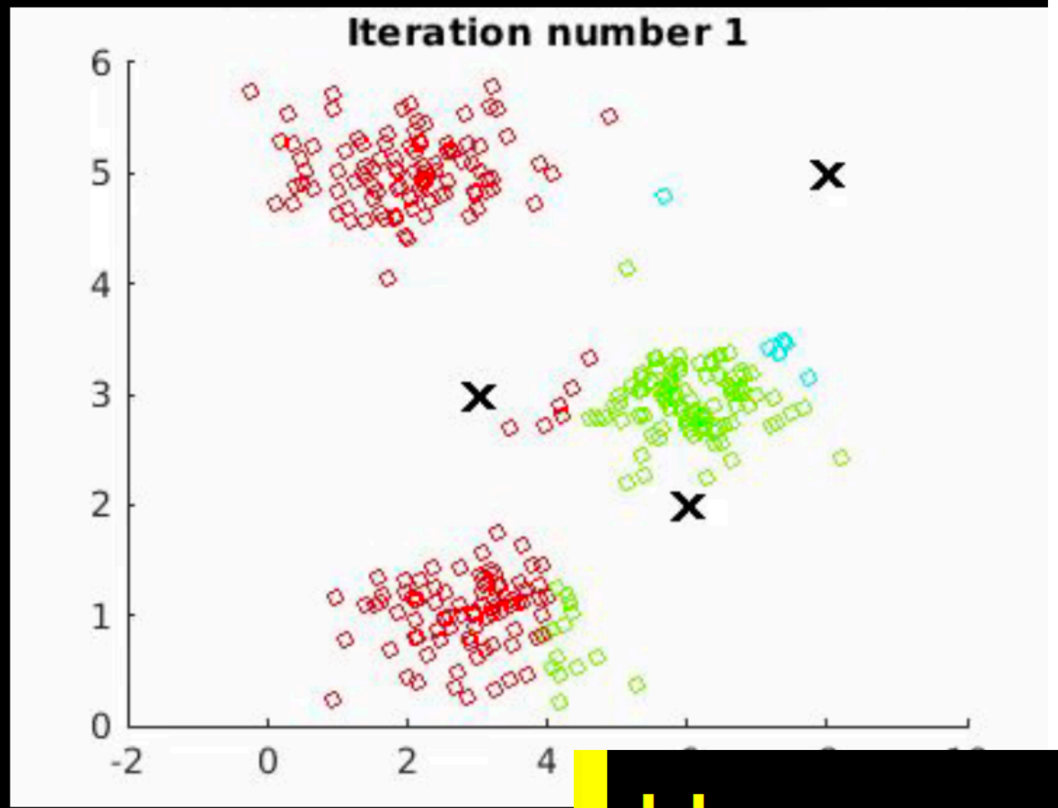
STEP 2: Select at random K points, the centroids (not necessarily from your dataset)

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
$\frac{(X_{hit}-X_{mean})^2}{L_{cell}^2}+\frac{(Y_{hit}-Y_{mean})^2}{L_{cell}^2}+\frac{(t_{hit}-t_{mean})^2}{(50\ ns)^2}$	squared 3D space-time distance
$\frac{(X_{hit}-X_{mean})^2}{L_{cell}^2}+\frac{(Y_{hit}-Y_{mean})^2}{L_{cell}^2}+\frac{(t_{hit}-t_{mean})^2}{(50\ ns)^2}+(E_{hit}-E_{mean})^2$	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

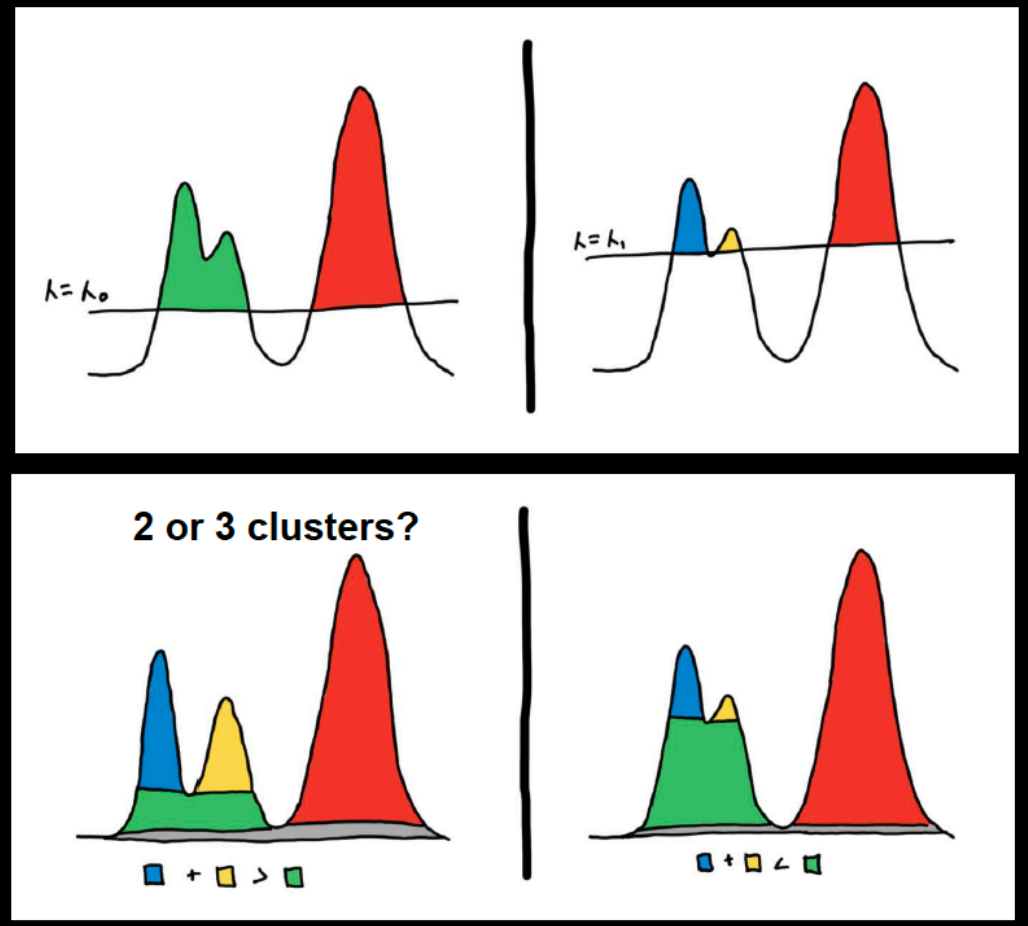
$$bool = \Delta t < 50\ ns\ \&\&\ \Delta X \leq 1\ \&\&\ \Delta Y \leq 1\ \&\&\ (\Delta X + \Delta Y) > 0 \quad (3.1)$$

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

Unsupervised: hdbscan

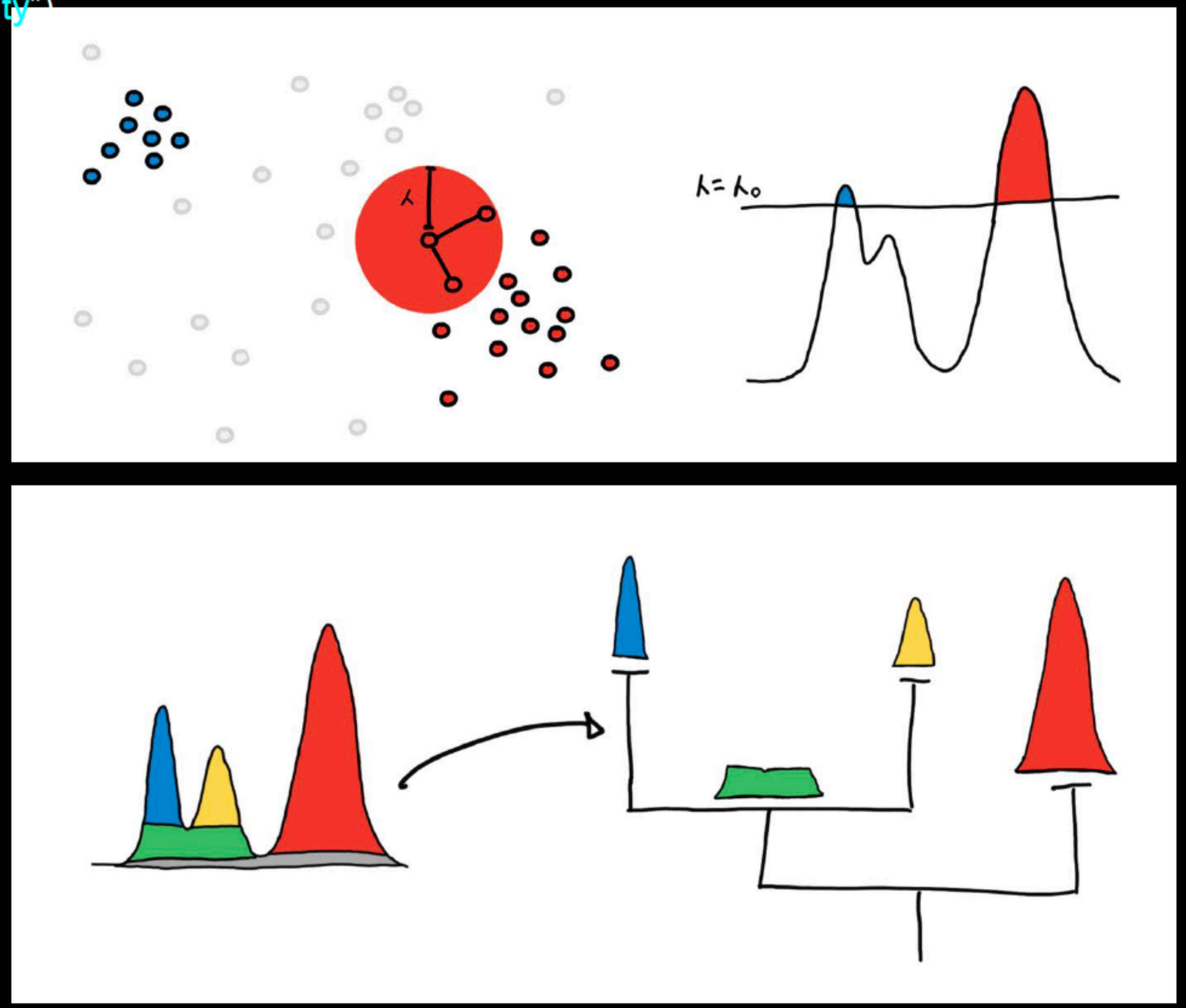
Unsupervised: e.g., Hierarchical Clustering

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")

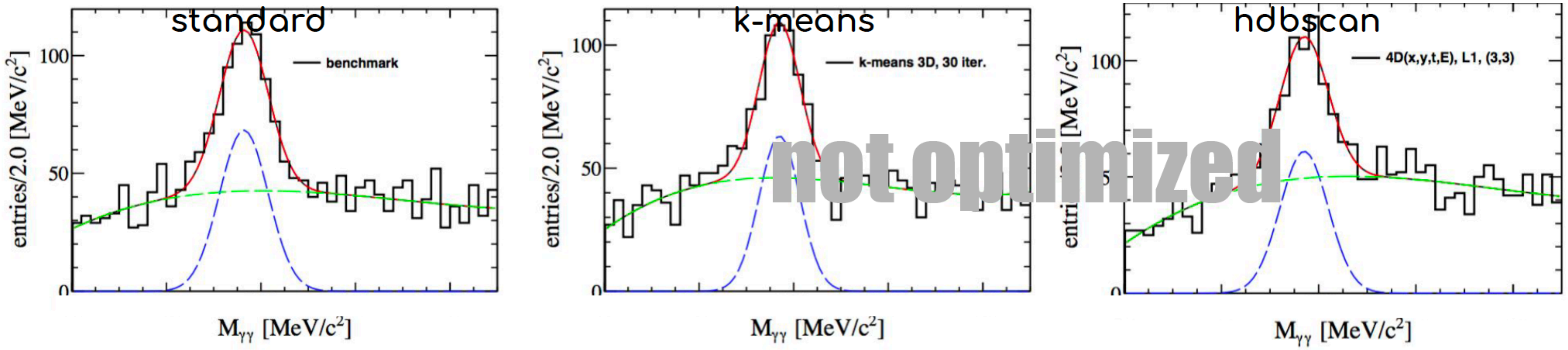


The area of the regions is the measure of "persistence".

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)

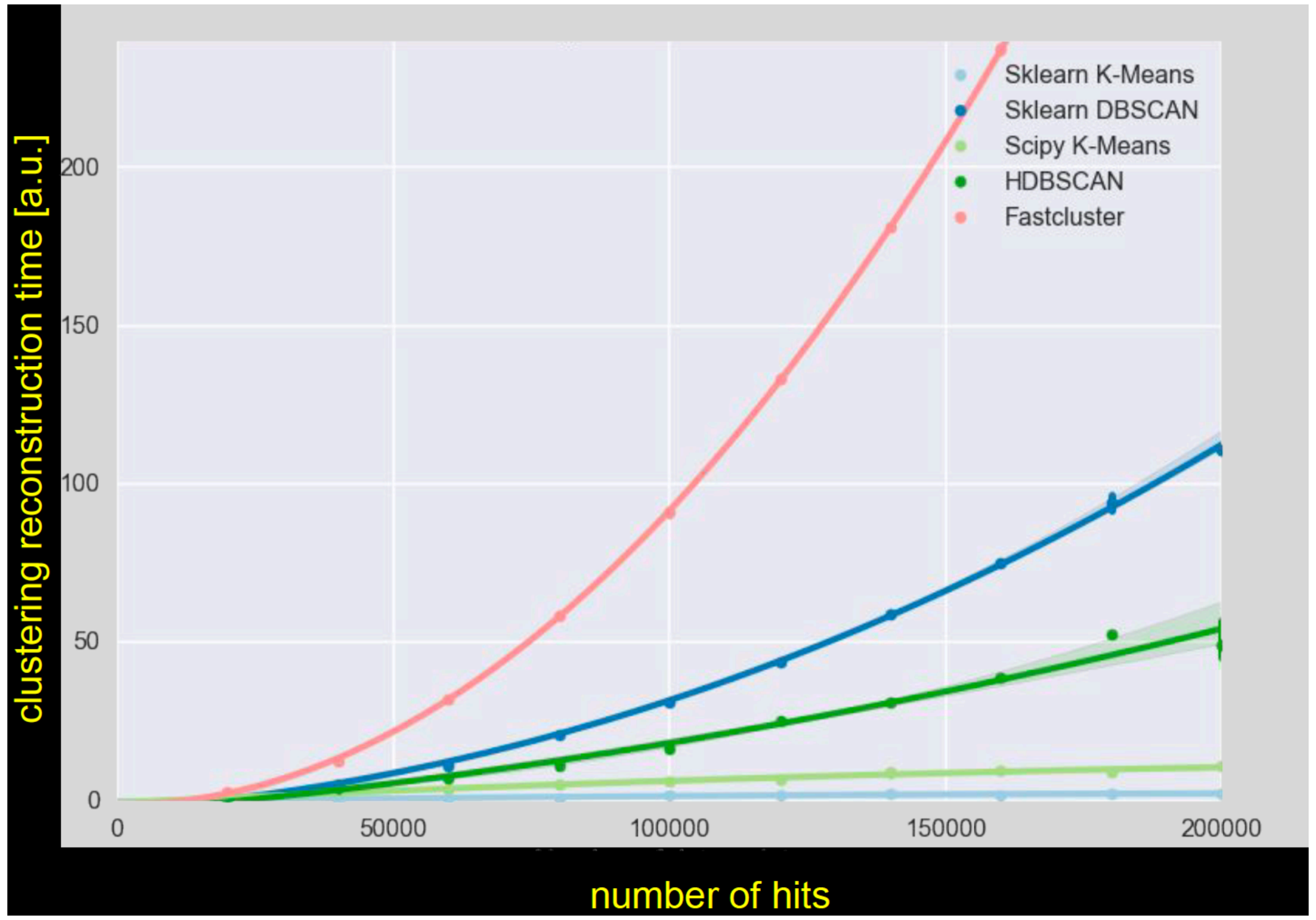
Requirements on clusters:

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

hdbscan: unsupervised hierarchical clustering

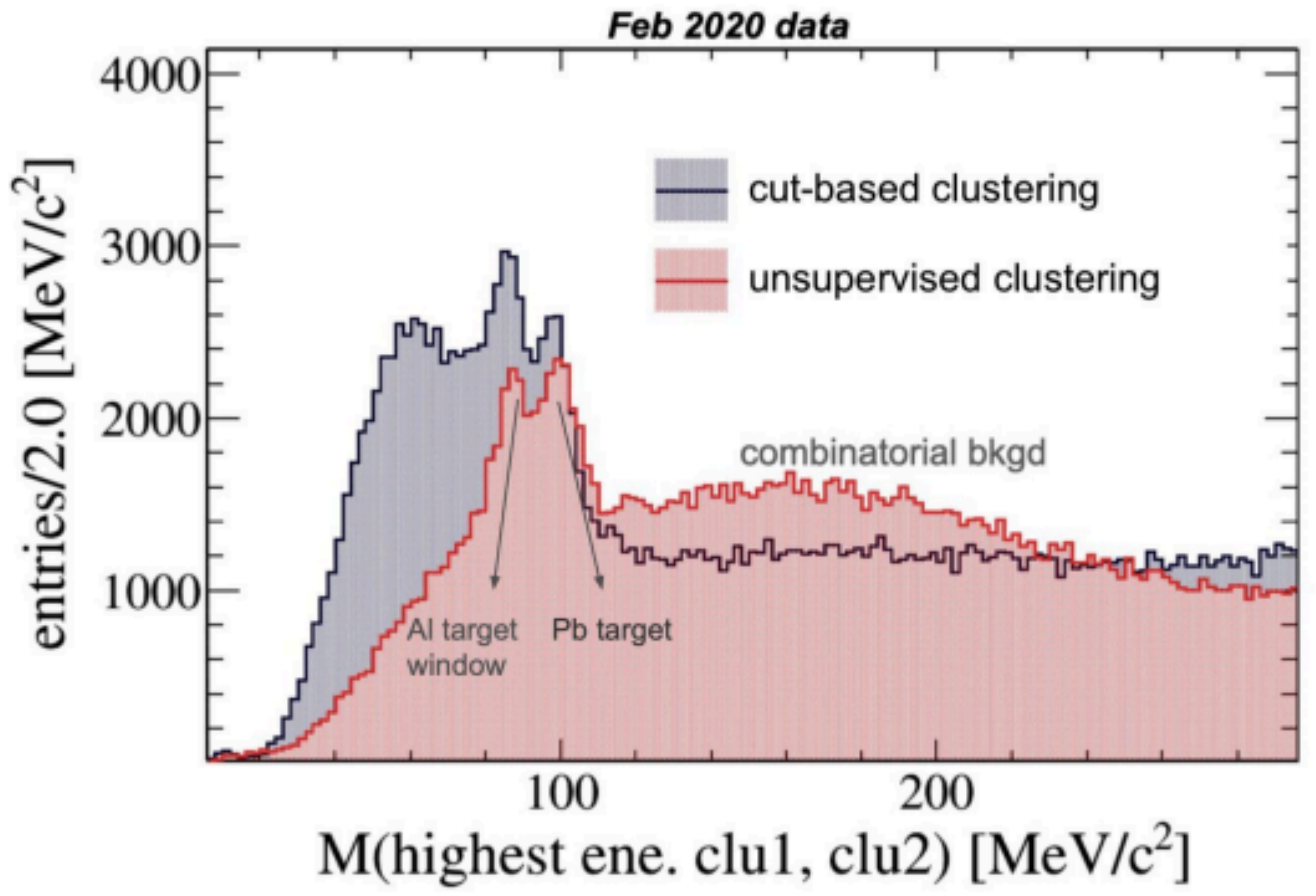
Best performance when data are/have:

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



SRO test @ JLAB results: AI vs standard clustering

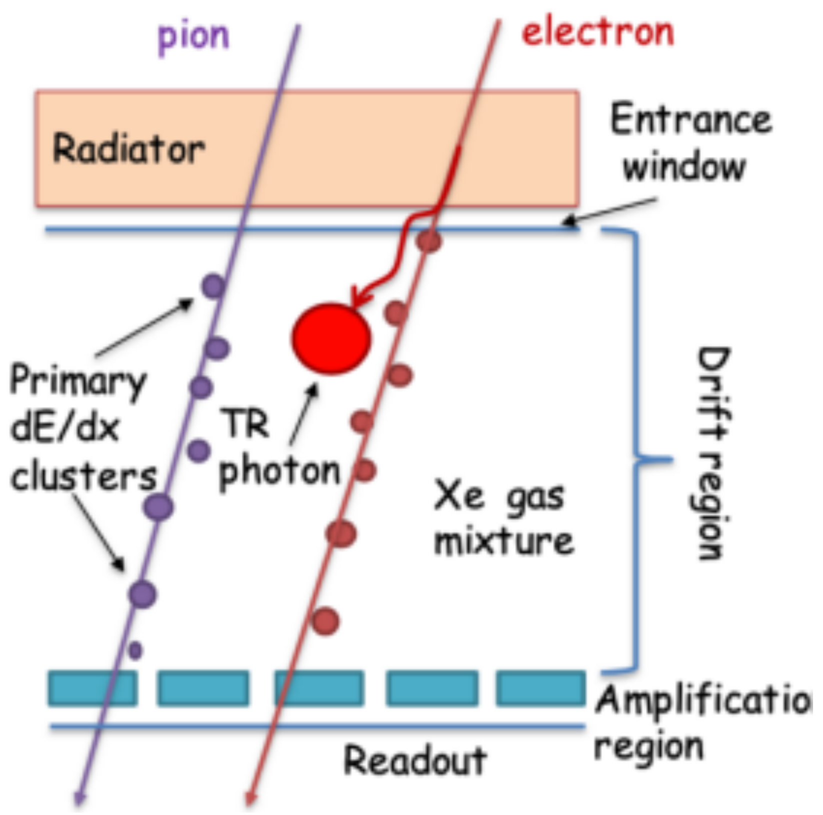
C. Fanelli

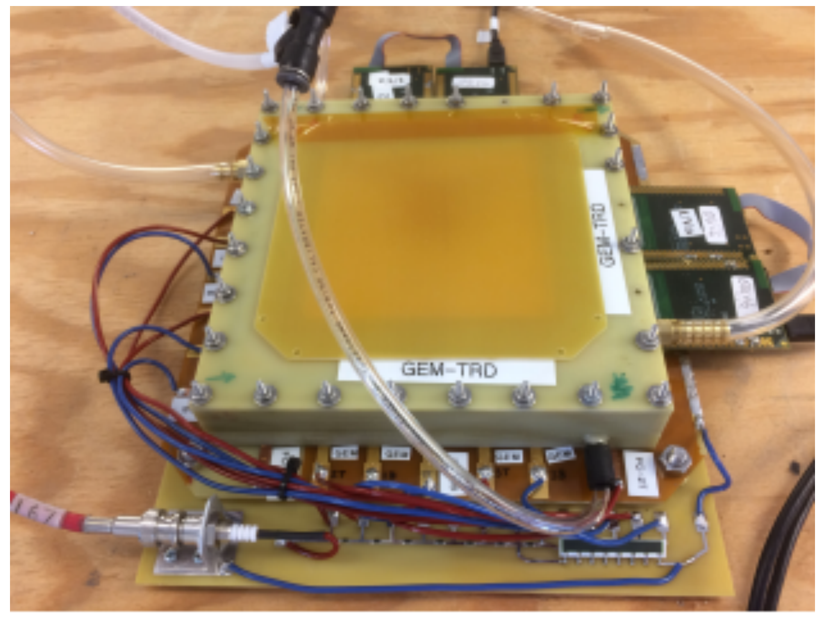


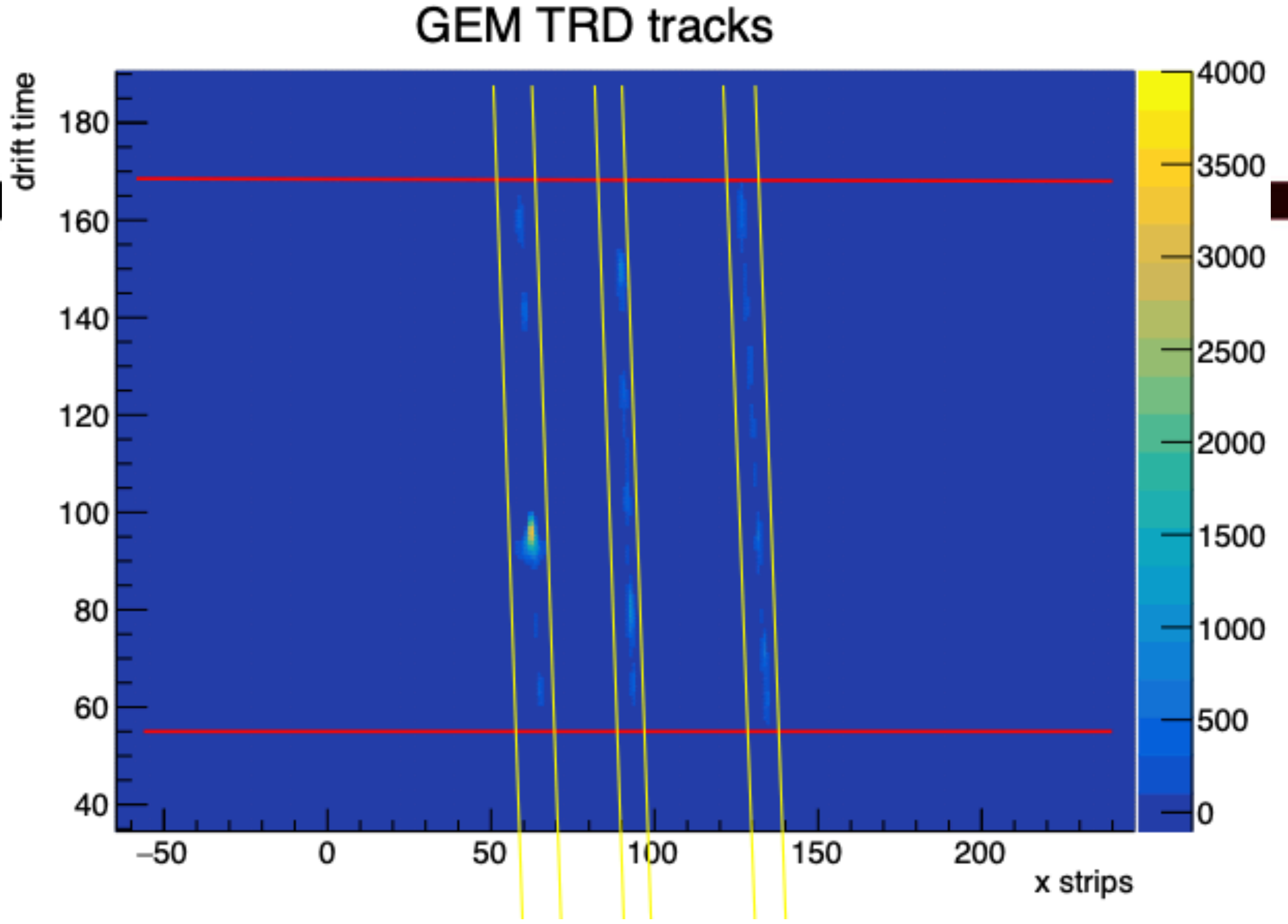
- AI clustering inspired by *Hierarchical Density-Based Spatial Clustering of Applications with Noise* (HDBSCAN)
  - It is not cut-based
  - it is able to cope with a large number of hits
- Compared γγ-invariant mass spectrum obtained utilizing both the standard and the HDBSCAN clustering algorithm
  - 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

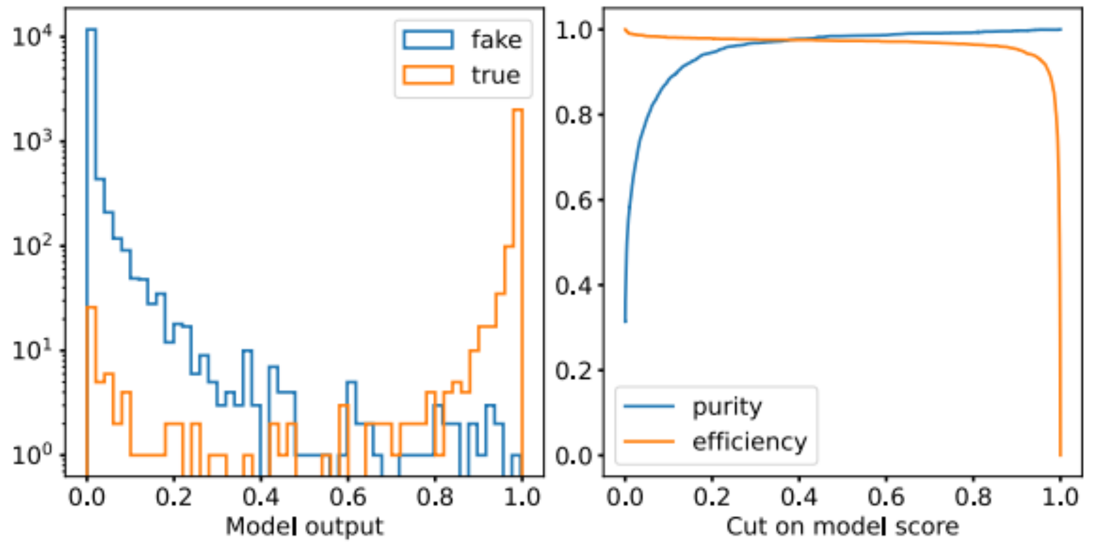
Fast AI applications: GEM-TRD





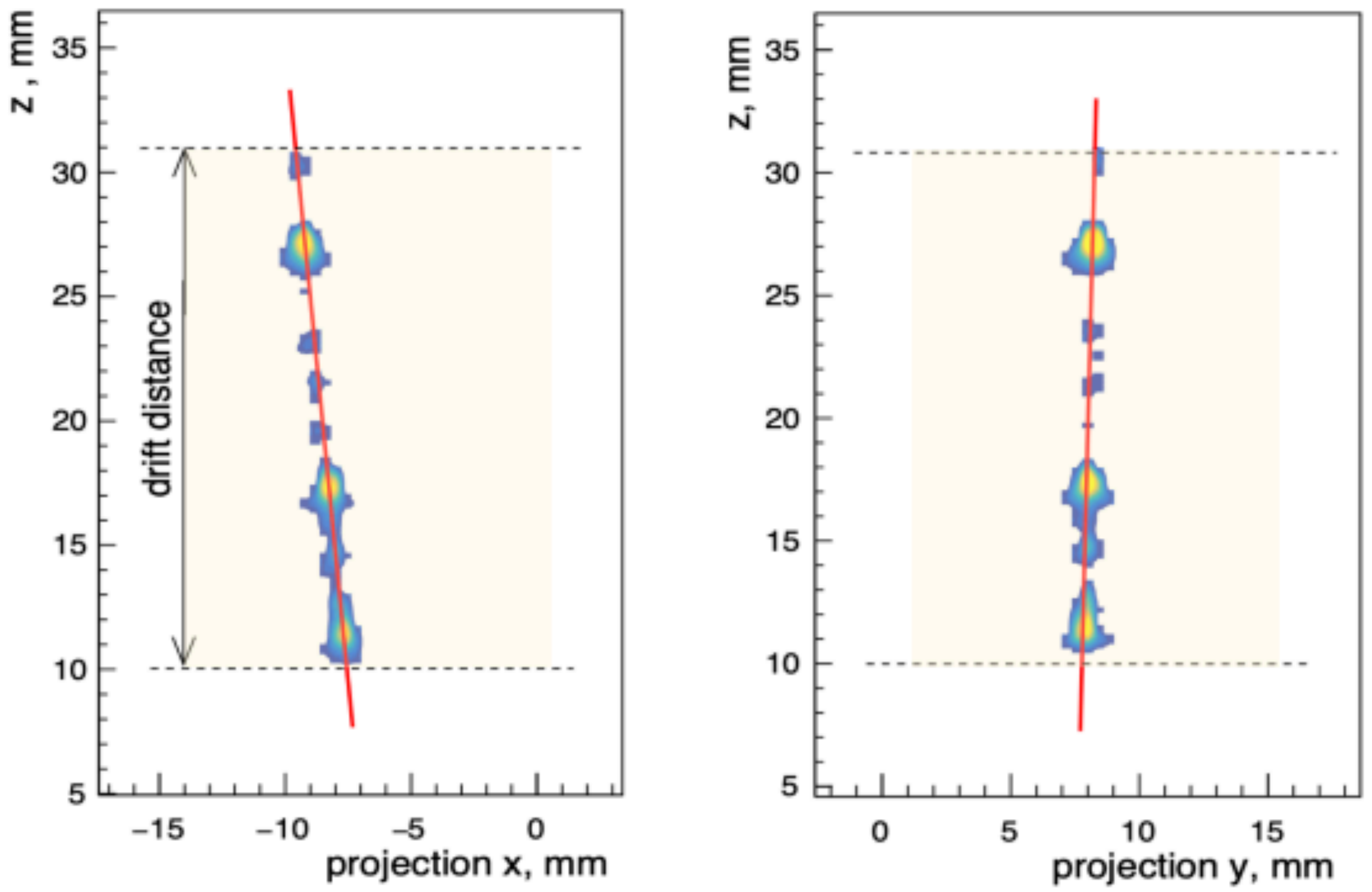


- GEM-TRD copes with multiple tracks
- Fast pattern recognition algorithm: Graph Neural Network (GNN)
- Track fitting: recurrent neural network – LSTM
- Implemented on FPGA using High Level Synthesis (hls4ml)



- 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

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



GNN on FPGAs

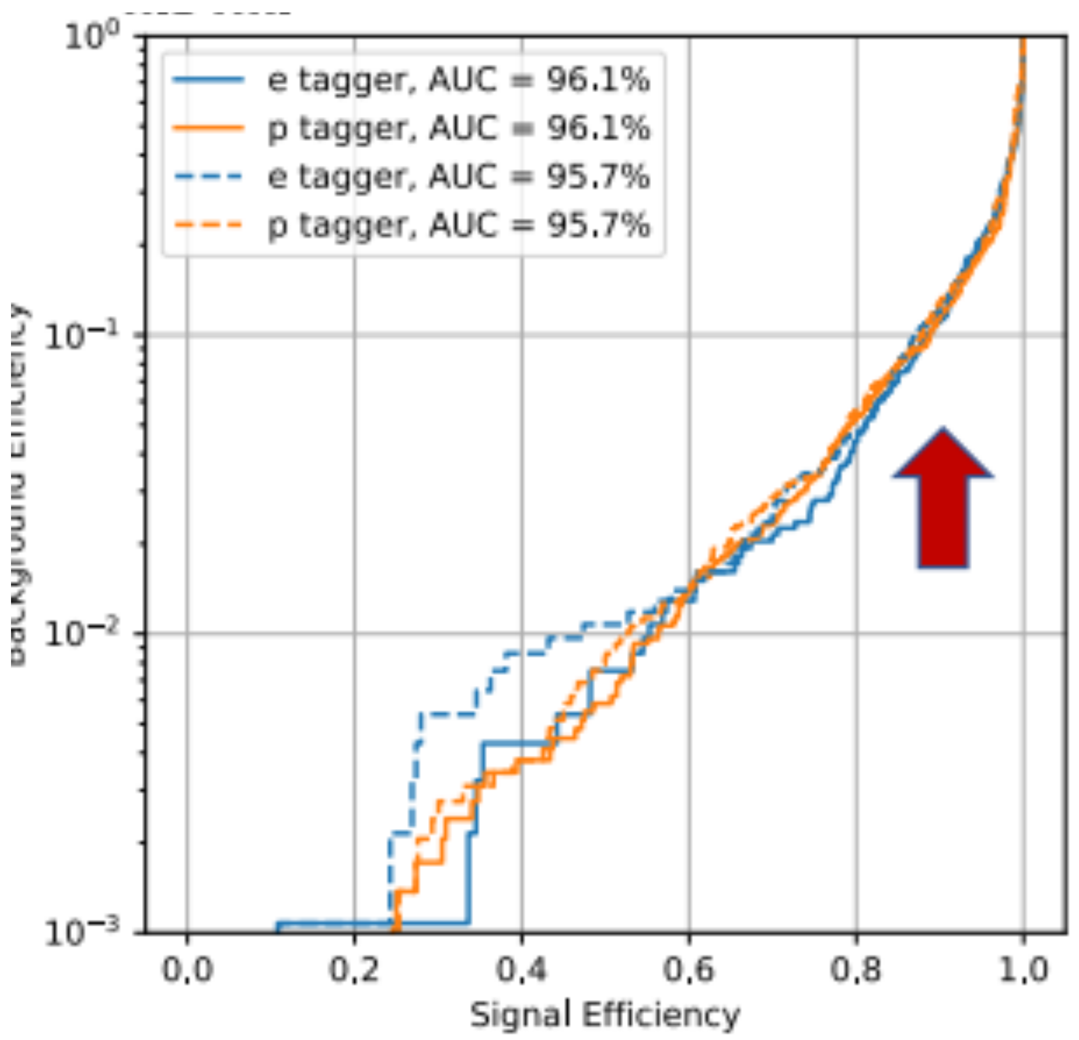
- imported by hands
- 1.4us inference time
- Good (preliminary) results

RNN/LSTM on FPGAs

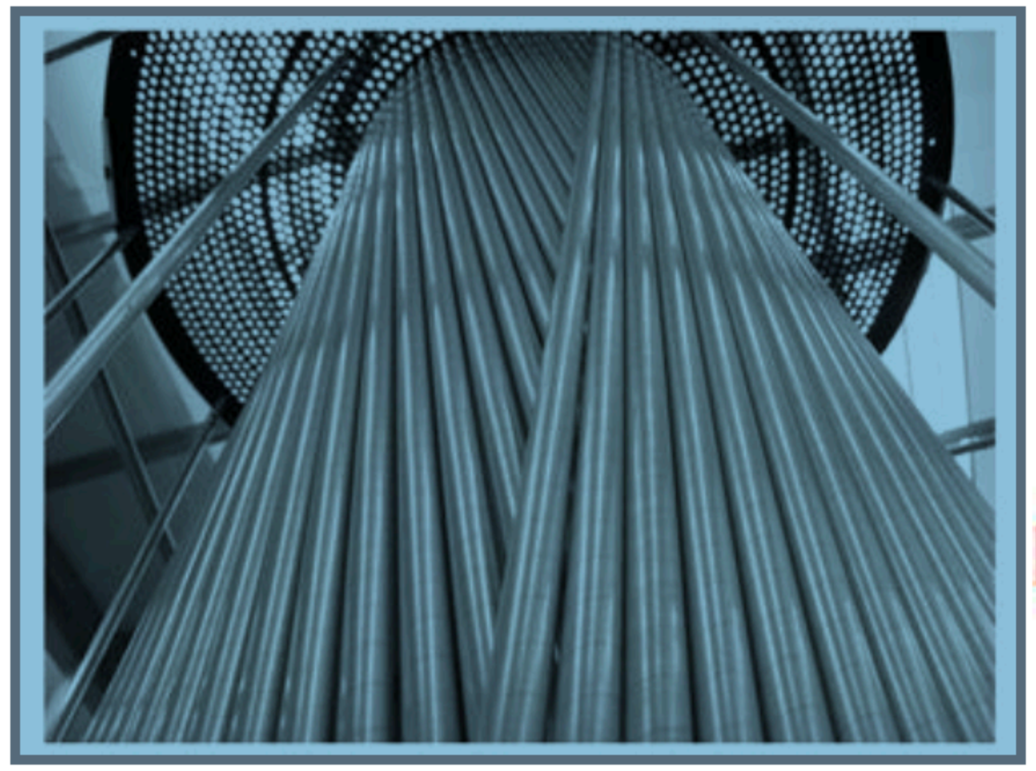
- Only 19% of FPGA resources
- 1us latency time
- Good (preliminary) performance

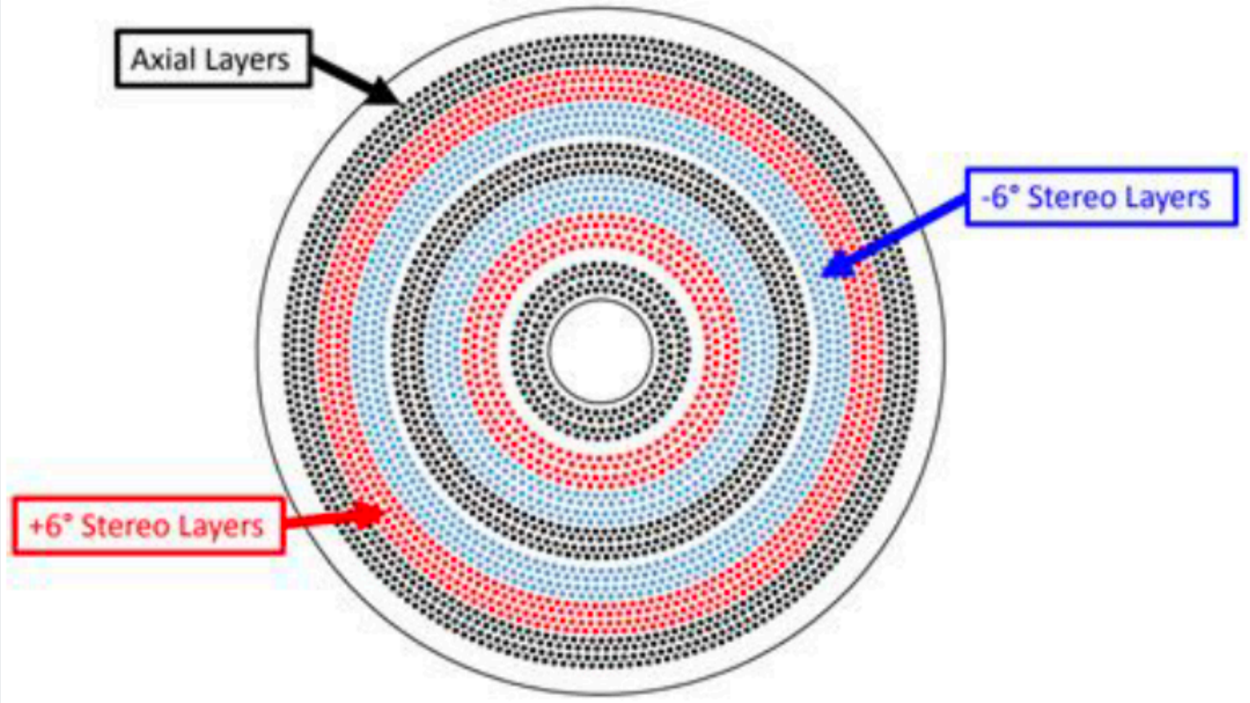
MLP on FPGAs

- 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

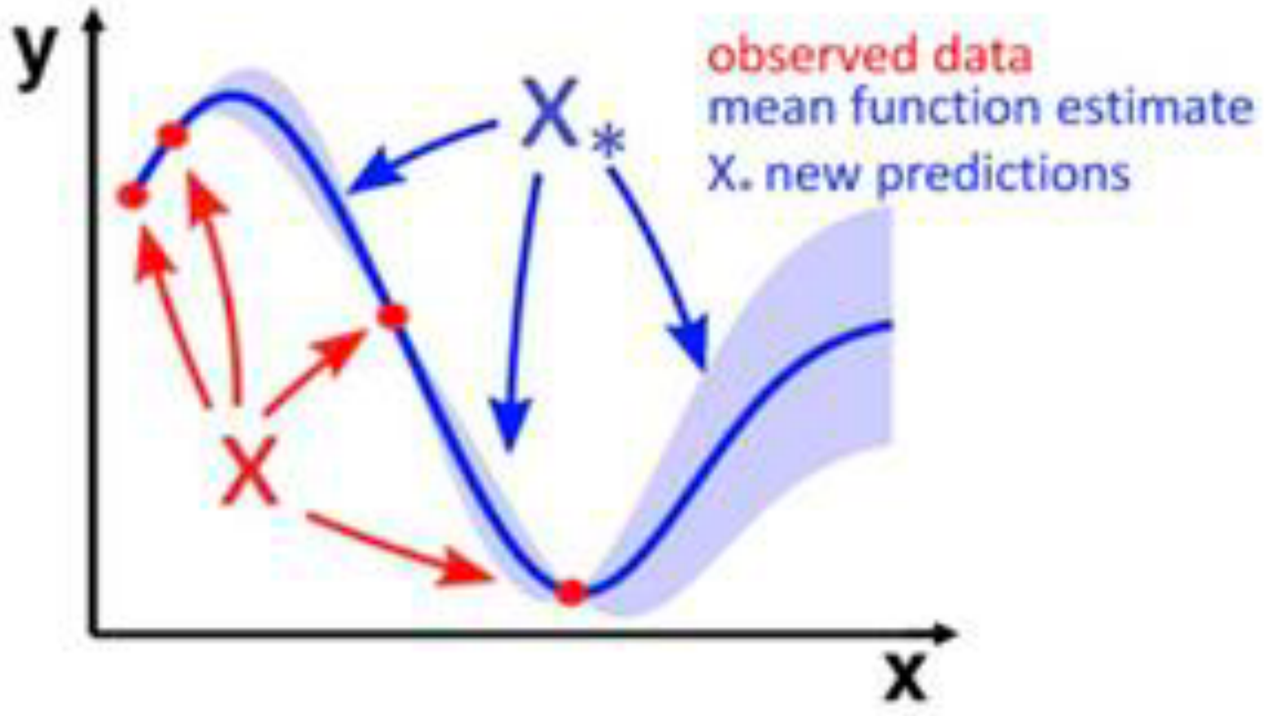
Requires two calibrations: chamber gain and drift time-to-distance

- 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)

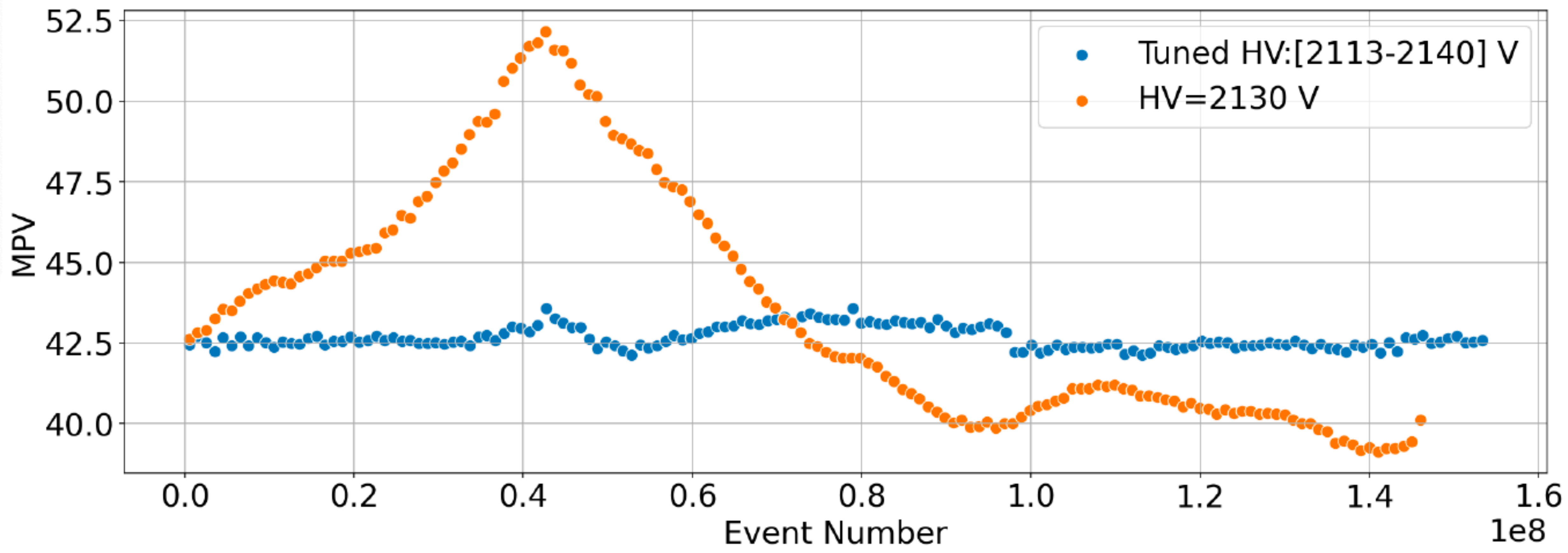
- atmospheric pressure within the hall
- temperature within CDC
- CDC high voltage board current



- 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!

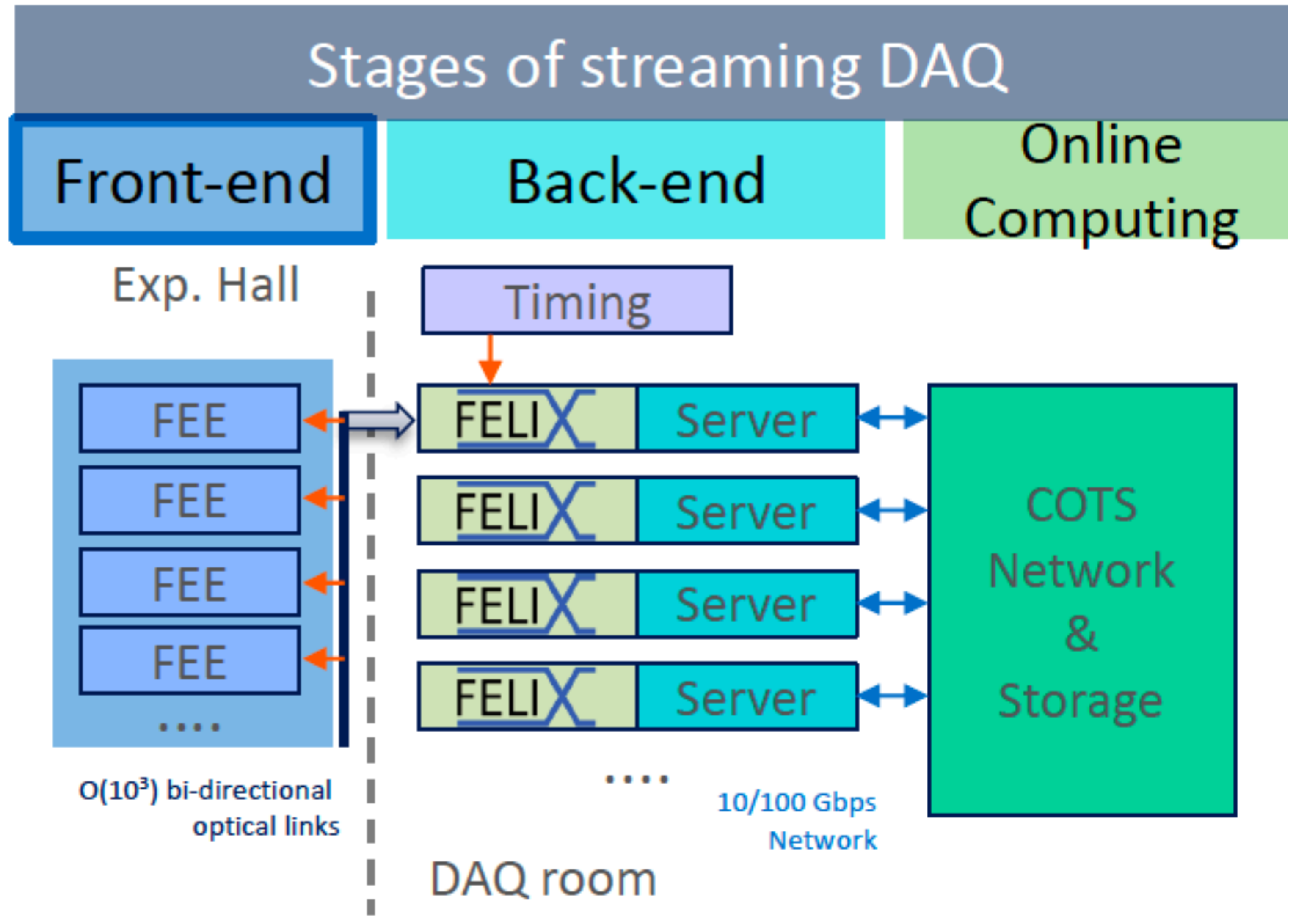


- Half the CDC (orange) at fixed HV, t he other half (blue) had its high voltages adjusted every 5 minutes

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



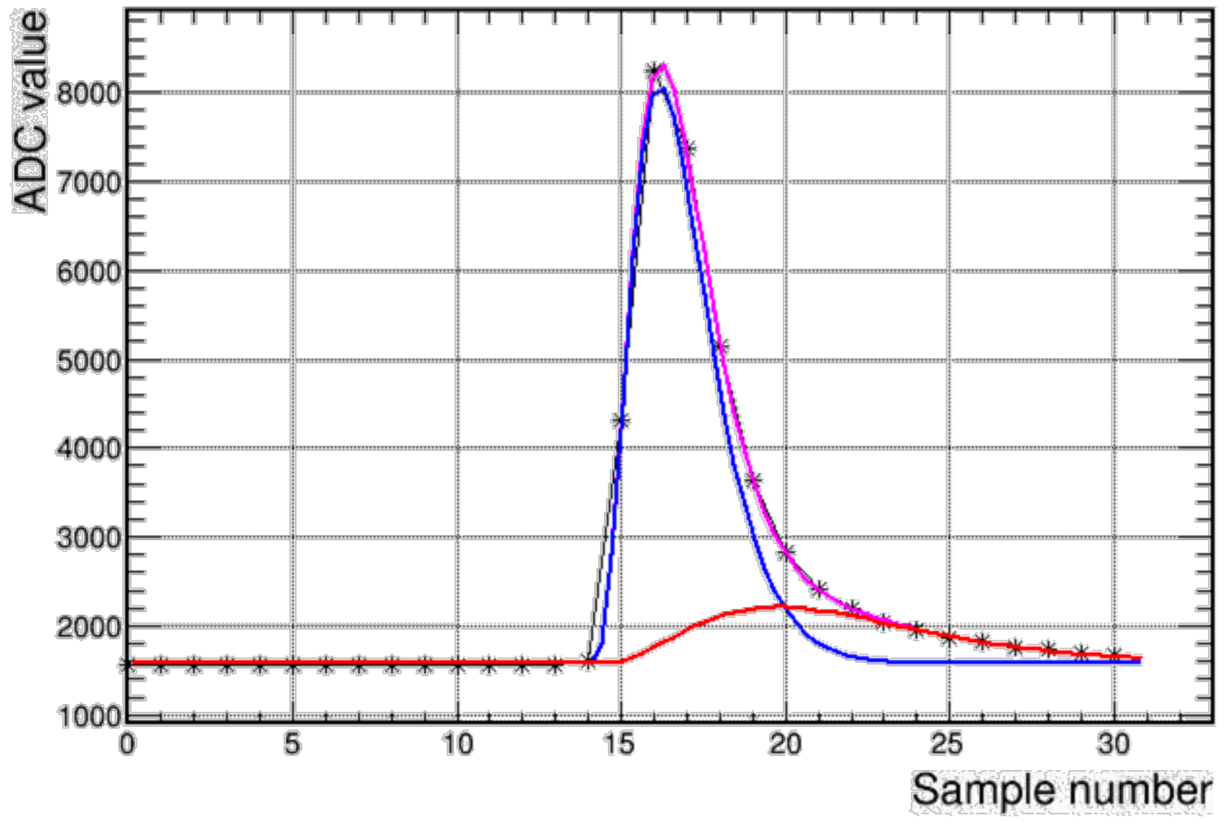
Opportunities for real-time AI but also a challenge:

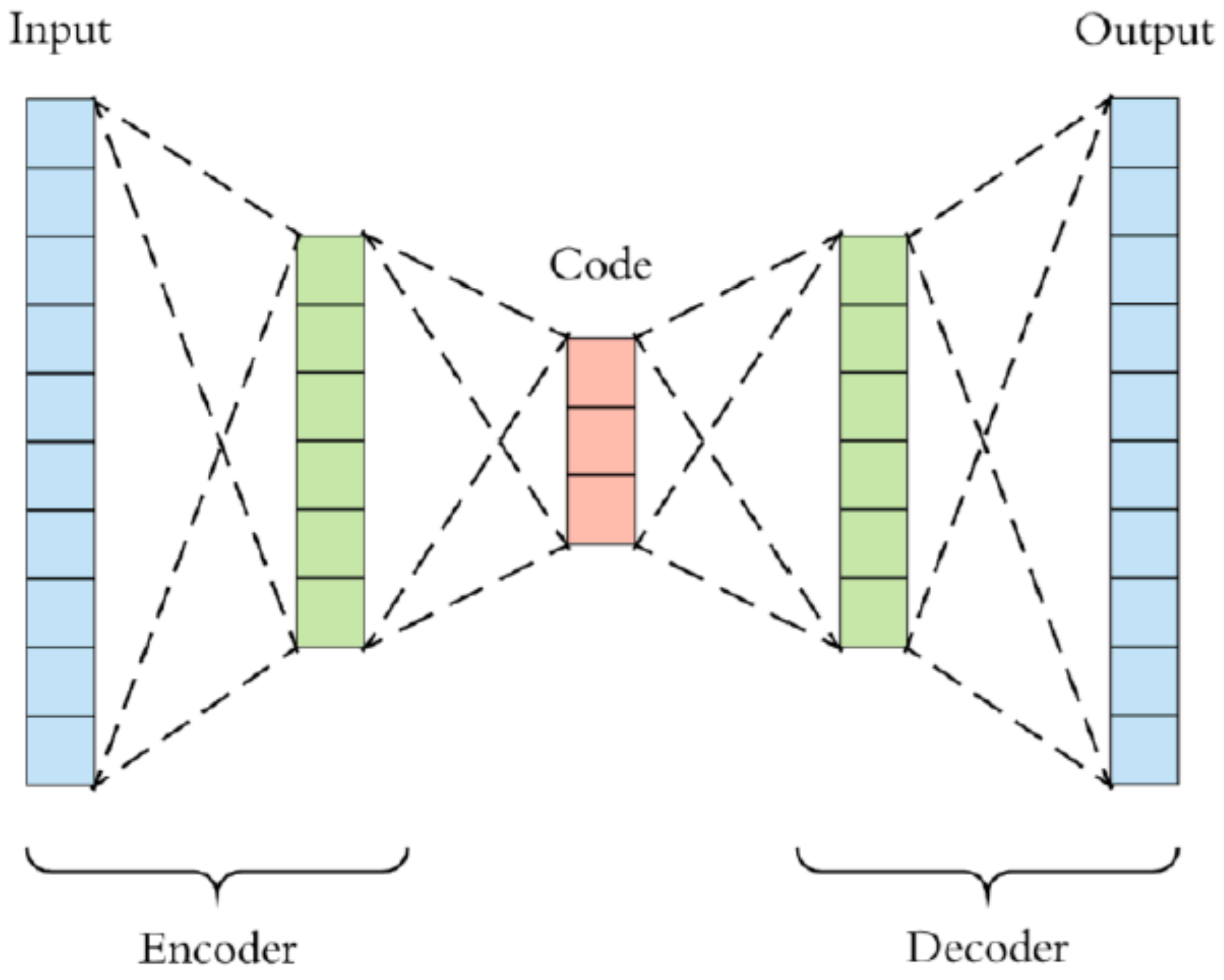
- reliable data reduction
- Applicable at each stages of streaming DAQ (front-end electronics, readout back-end, online computing)
- Data quality monitoring, fast calibration/reconstruction

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) FPGAs Common 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





Autoencoder

- Charge (Energy) and time are compact to stream but partial
- fast and efficient way to preserve the full (anagogic) wave-form information
- Reduce the traffic on the first stages of the SRO DAQ pipeline