PIONEER Software Development Update

Jack Carlton
University of Kentucky

Outline

I. PIONEER Refresher
   A. Strategy
   B. Experimental Design
II. DAQ Software Development
   A. Calorimeter Teststand DAQ
   B. ATAR Candidate DAQ: HDSoC
   C. ATAR Candidate DAQ: SAMPic
   D. Deadtime Tests for ATAR DAQ Candidates
   E. Example Supplementary Software: Flexible Data Quality Monitor
III. Simulation Software Development
   A. ATAR Pattern Finding
   B. ATAR AI Reconstruction
IV. PSI 2025 Testbeam
   A. “Soccer Ball” Design
   B. Energy Resolution Measurement
V. Graduation Planned Timeline

You can find this presentation in my notes

Links:
https://jaca230.github.io/joplin_notes_page/
or
https://tinyurl.com/jack-uky-notes

PIONEER Refresher

j.carlton@uky.edu

Primary PIONEER Goal

e/μ Branching Ratio

$$R_{e/\mu} = \frac{\Gamma(\pi^+ \to e^+\nu(\gamma))}{\Gamma(\pi^+ \to \mu^+\nu(\gamma))}$$

With a precision < 0.01%

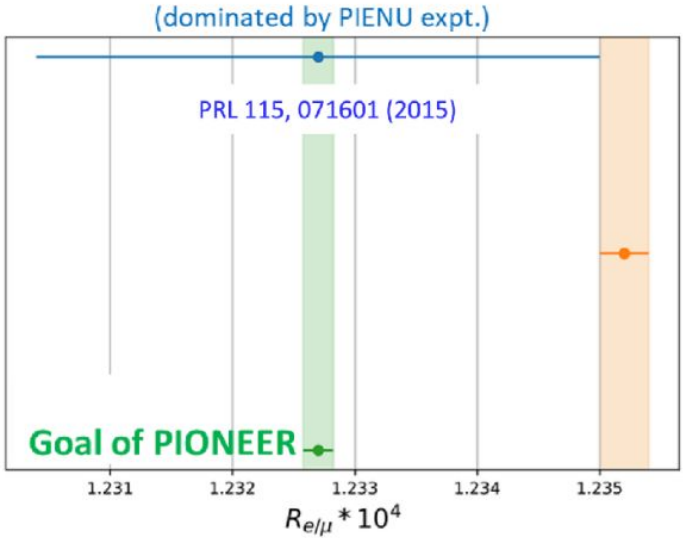
$$R_{e/\mu}(\text{SM}) = 1.23524(015) \times 10^{-4}$$

D. Bryman et al. arXiv:2111.05338

$$R_{e/\mu}(\text{EXP}) = 1.23270(230) \times 10^{-4}$$

PDG

Factor 15 improvement



Lepton Flavour Universality Test

PIONEER Strategy

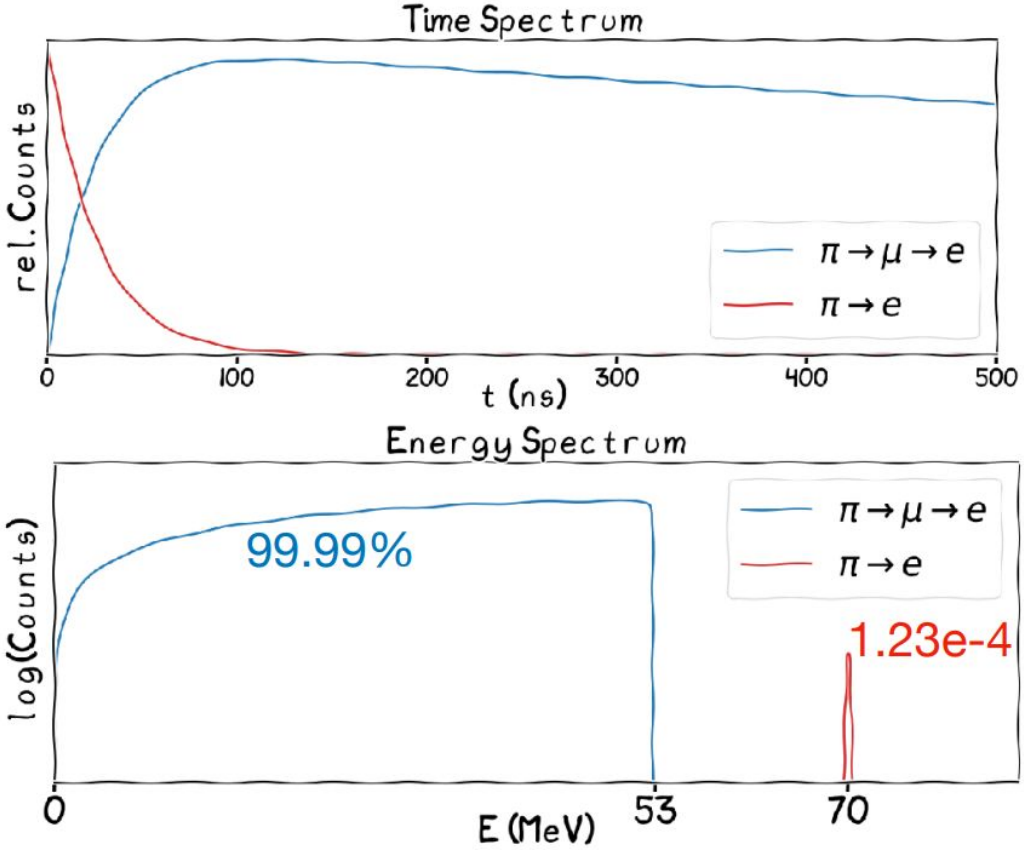
- Calorimeter measures just positron energy
  - Muon stopped by target
  - Neutrinos invisible
- Prompt 70 MeV = π→e
- Delayed <53 MeV = π→μ→e
- Plots are for idealized, perfect detectors (in a “simple world”)

$\tau_{\pi^+} = 26$ ns, $m_{\pi^+} \approx 140$ MeV

$\tau_{\mu^+} = 2197$ ns, $m_{\mu^+} \approx 106$ MeV

$\pi^+ \to \mu^+\nu \quad E_e < m_\mu/2 \approx 53$ MeV

$\pi^+ \to e^+\nu \quad E_e = m_\pi/2 \approx 70$ MeV

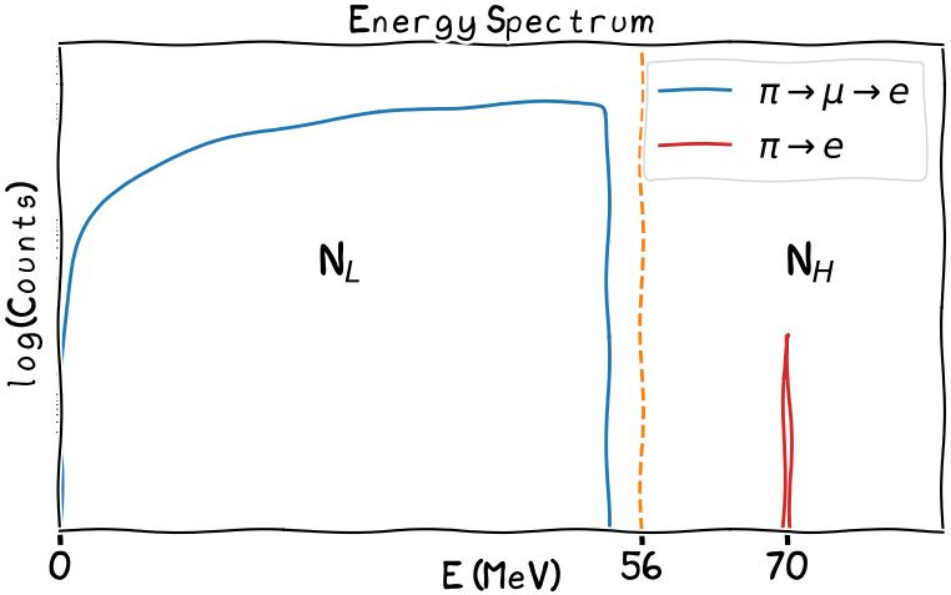


j.carlton@uky.edu

PIONEER Strategy

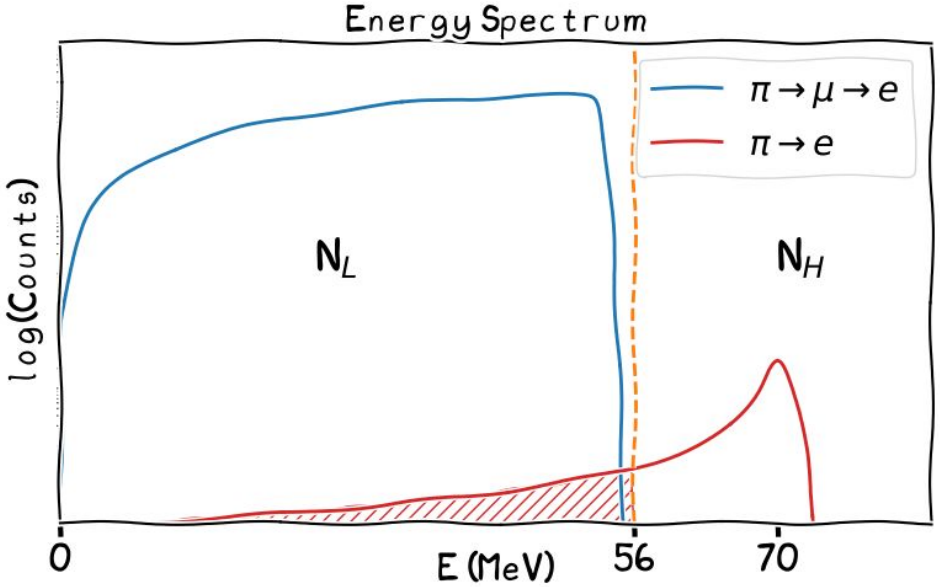
- At a glance the experiment is simple
- Just look at energy deposited on the calorimeter and count events above and below $E_{thr} \equiv 56$ MeV
- In reality, not so simple…

$$R_{e/\mu} = \frac{\Gamma(\pi^+ \to e^+\nu(\gamma))}{\Gamma(\pi^+ \to \mu^+\nu(\gamma))} \approx \frac{N_H}{N_L}$$



PIONEER Strategy

- Finite resolution and incomplete energy collection in the calorimeter result in a tail
- To compete with theoretical uncertainty, we need to characterize this tail to ~1 part in 10,000 ($\frac{\delta\Gamma_e}{\Gamma_e} \sim 10^{-4}$)



$$R_{e/\mu} = \frac{\Gamma(\pi^+ \to e^+\nu(\gamma))}{\Gamma(\pi^+ \to \mu^+\nu(\gamma))} \approx \frac{N_H}{N_L} \times (1 + c_T)$$

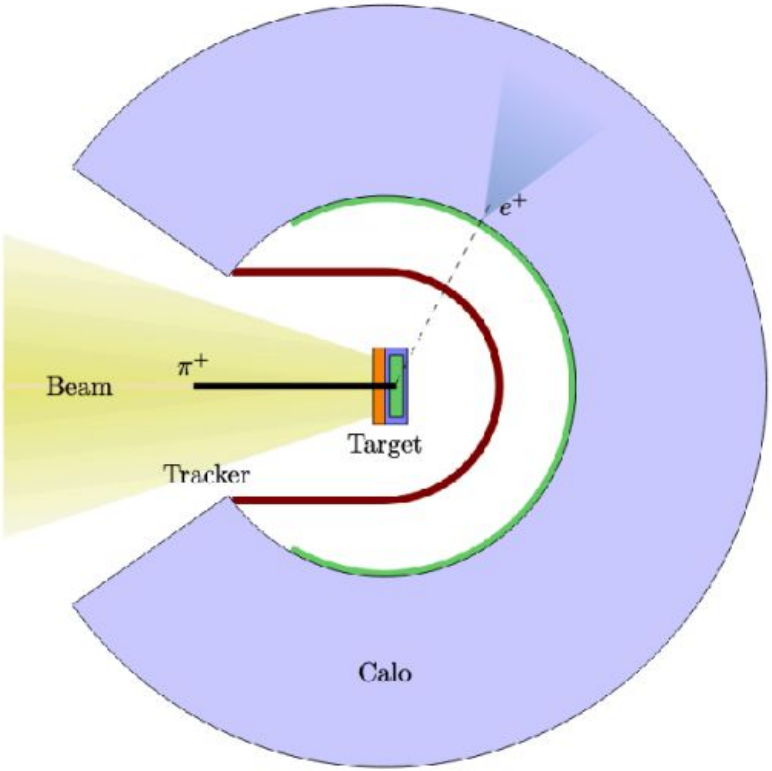
Tail Correction

$c_T \approx 5 \times 10^{-3}$

$$c_T \approx \frac{N_L^{\pi \to e}}{N_H^{\pi \to e}}$$

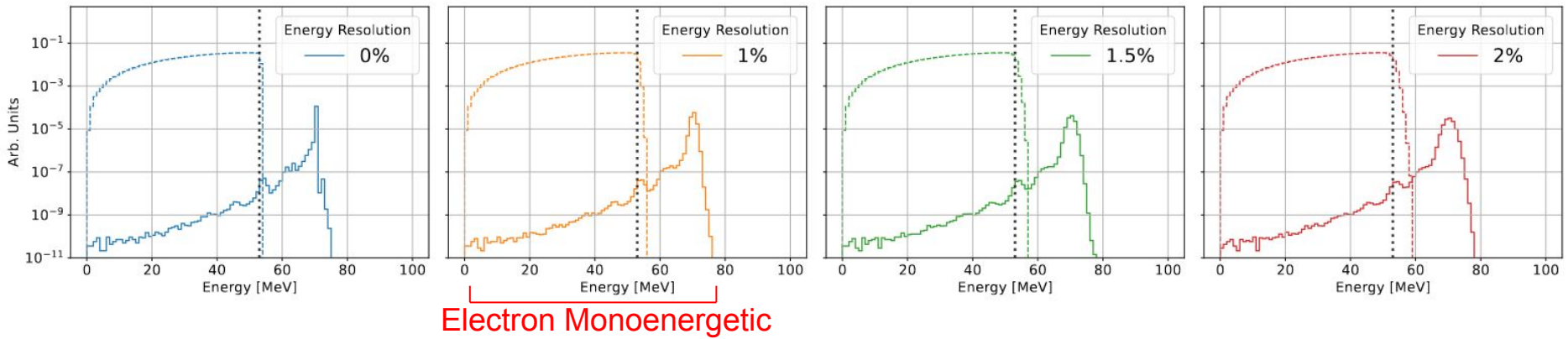
PIONEER Experimental Proposal

- LYSO (or LXe) as a calorimeter material has short decay time and good energy resolution
  - $\sigma_t \approx 100$ ps (timing resolution)
  - $\Delta E/E \approx 2\%$ (energy resolution)
- Experiment to run at much higher beam rate than PIENU
  - ~300kHz (phase 1)
  - ~2000kHz (phase 2 and 3)
- “active target”, muons and pions are “tracked” while being stopped



Calorimeter (CALO) Purpose

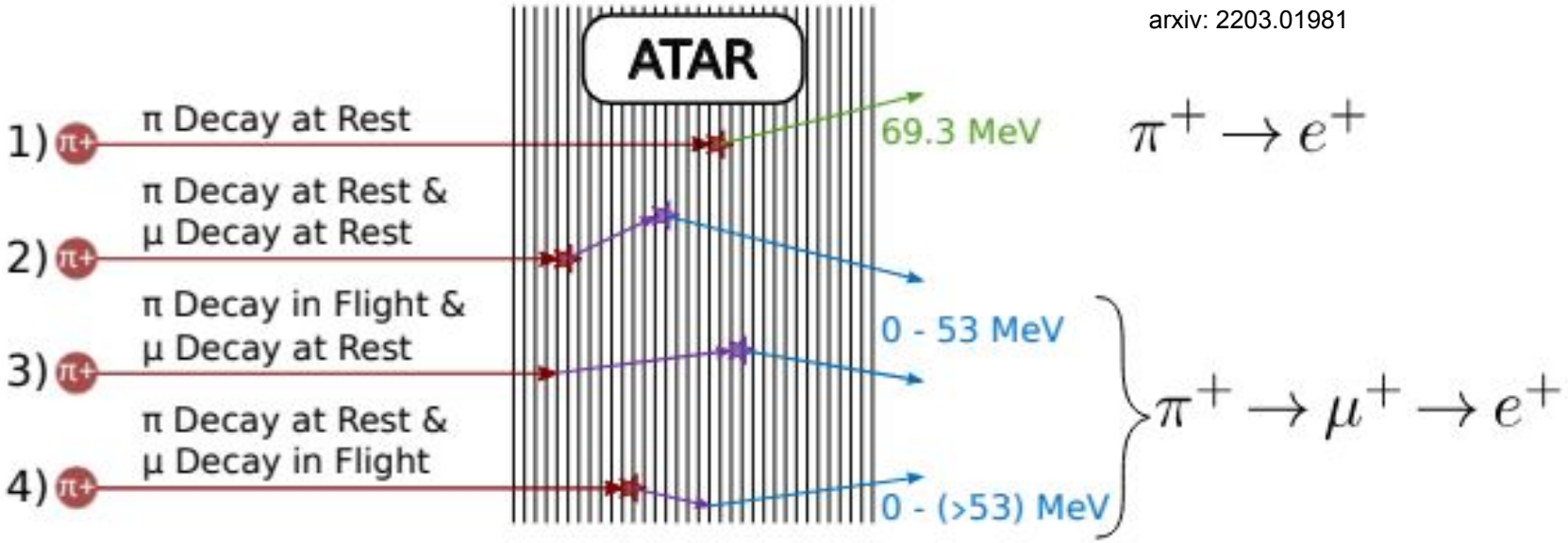
Muon Michel



Electron Monoenergetic

- Muon's intrinsically follow a Michel Spectrum
  - Additionally width comes from energy resolution
- Positrons follow monoenergetic spectrum
  - Deviation from 69.8 MeV mostly due to radiative decays
- Good energy resolution is crucial for event reconstruction
  - The less likely muons decays are to contaminated the electron tail, the lower we can set the energy threshold. Therefore, the tail correction becomes smaller and the ATAR can afford to characterize it with worse relative precision

Active Target (ATAR) Purpose



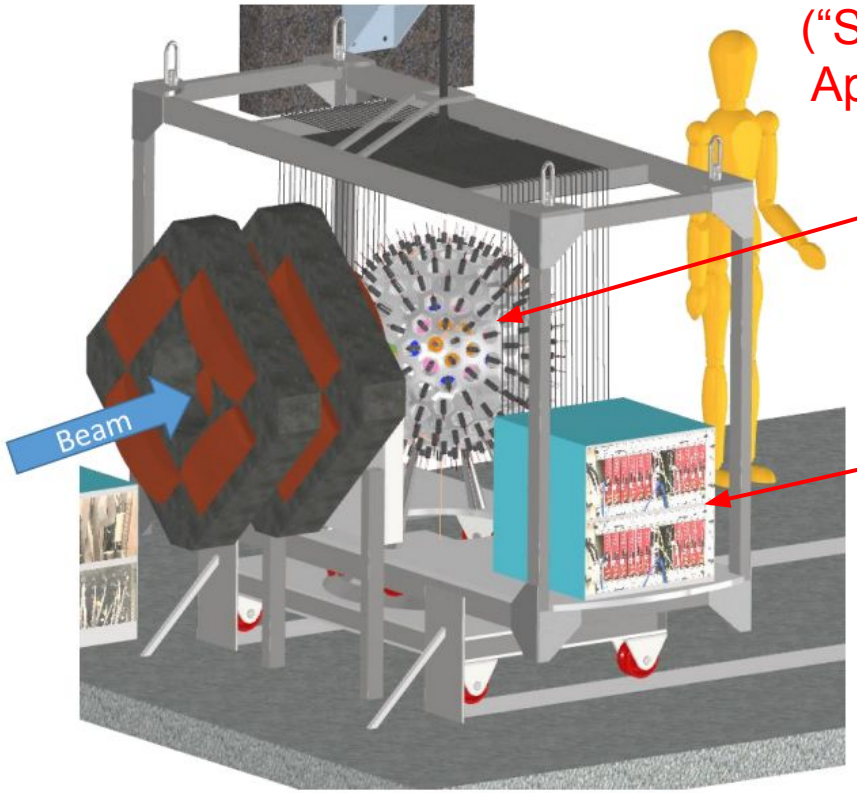
- ATAR allows classifying between π→e and π→μ→e based on 5D (x,y,z,t,E) information
  - Worse energy resolution than calorimeter, cannot do experiment on it's own
- Case 5. π and μ decay in flight negligible
  - ATAR designed to stop heavy "slow" particles within their decay time; decay in flights are are rare thus π and μ decays in flight within the same event are negligible.

j.carlton@uky.edu

3D Render Experiment

Not Pictured:

- ATAR (inside Calo)
- Tracker (a shell around ATAR, inside Calo)
- VETOs, T0, etc.
- DAQ Computers



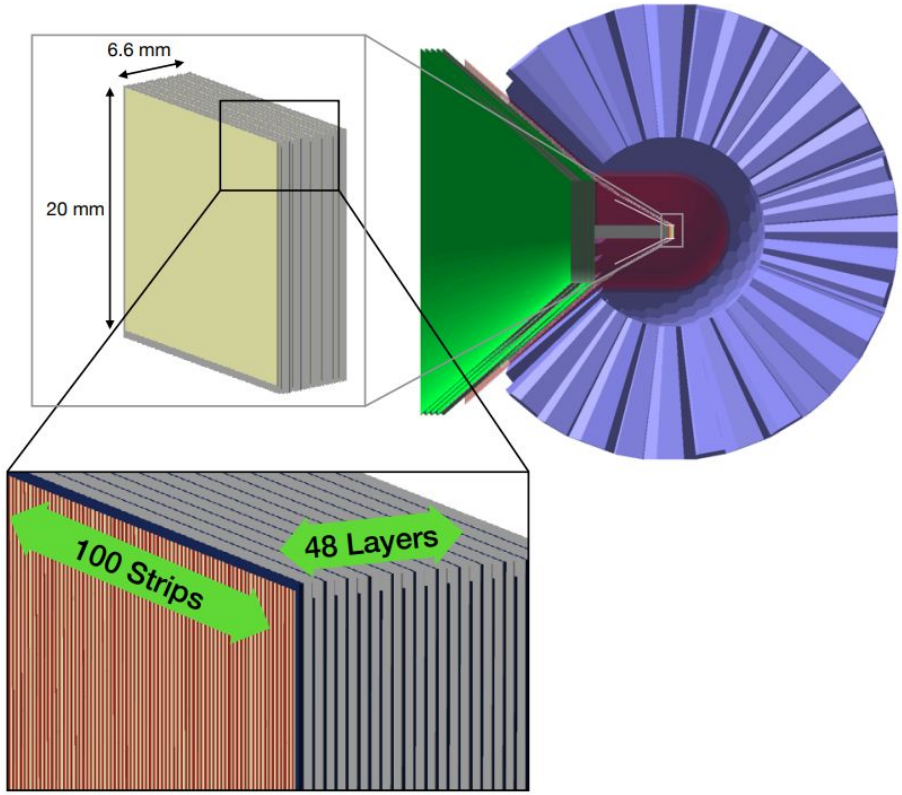
(6,0) Goldberg-Polyhedron Calorimeter Design ("Soccer Ball" Geometry) Approximately spherical

Digitization Electronics

j.carlton@uky.edu

ATAR Design

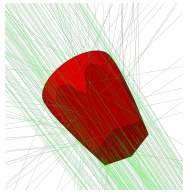
- Highly segmented and instrumented pion stopping target made of silicon low gain avalanche diodes (LGADs)
- Each sensitive strip has a size of 20 mm x 200 μm x 120 μm
- Total of 4800 channels arranged in pattern alternating x and y planes
- 5D track information:
  - Transverse positions x,y ~ O(200 μm)
  - Longitudinal position z ~ O(120 μm)
  - Time t ~ O(100 ps) per single hit
  - Energy E, ΔE/E ~ O(10%)

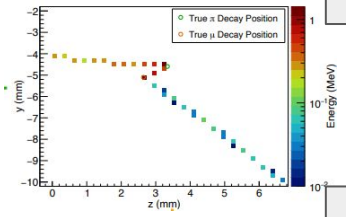


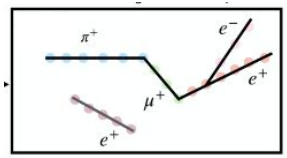
j.carlton@uky.edu

PIONEER Simulation Framework

- Series of software steps to simulate the PIONEER experiment
  - Work in progress
  - Adapts as we develop our detectors/strategy
- Designed as a proof of concept
  - Experiment should work in simulation before real experiment is run
- Reconstruction designed to be used on simulated **and** real detector data







Monte Carlo (Geant4 Based)

Truth level physics interaction data in space of detectors

Detector Response (Gaudi Based)

Realistic detector output data

Event Reconstruction (Gaudi Based)

Physical events reconstructed from detector data

Reconstructed Data

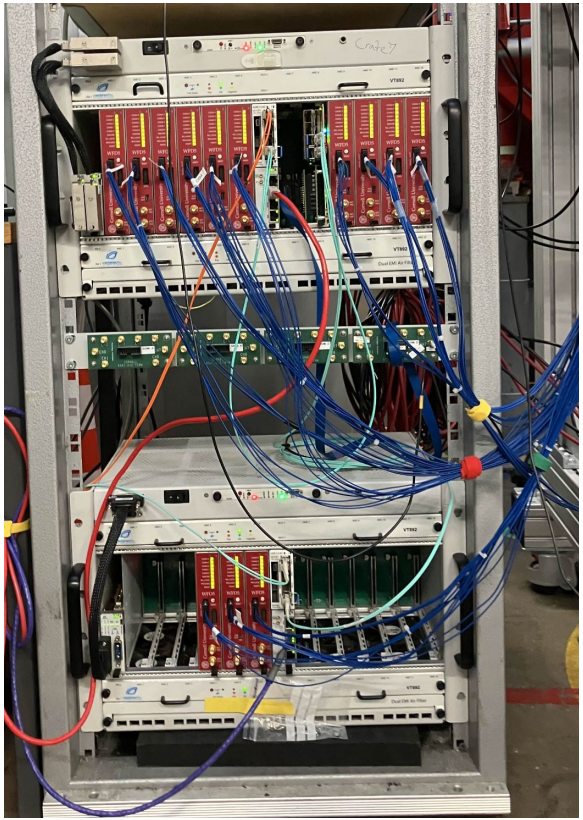
DAQ Software Development

Calo teststand DAQ

- Iteration on previous LYSO test beam

Run	Crates	Channels Used	Event Rate (Hz)	Data Rate (MB/s)
2023 PSI LYSO Test Beam	1	~50	~400	~30
2025 PSI LYSO Test Beam	2	~60	~2500	~200

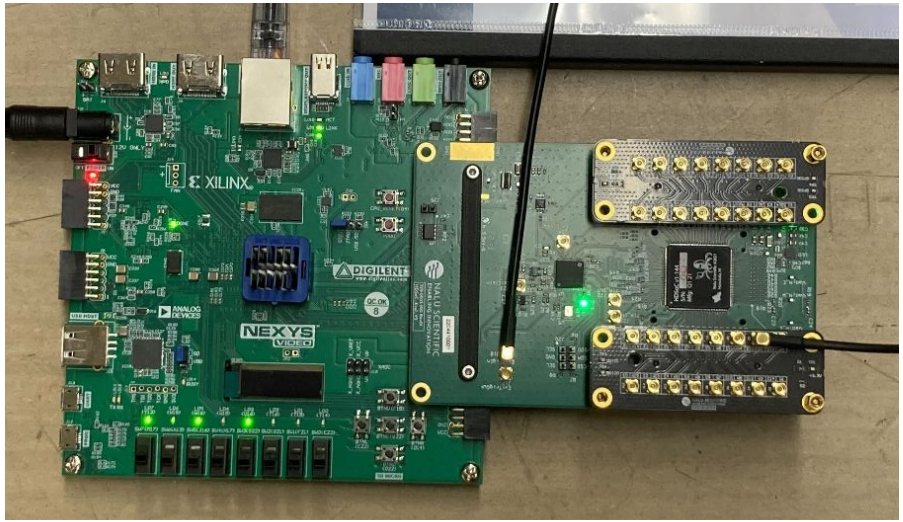
- Uses the same hardware as the g-2 experiment
  - 10 GbE based readout system
  - 800 MSPS Cornell digitizers
- Modified g-2 MIDAS readout software for broader test-stand use
  - LYSO calorimeter test stands
    - Used at PSI in November 2023 rectilinear LYSO crystal array test beam
    - Used at PSI in August 2025 tapered LYSO crystal array test beam
  - LXe calorimeter test stands



Two crate setup used at PSI LYSO Testbeam

ATAR Candidate Digitizer: HDSoC DAQ

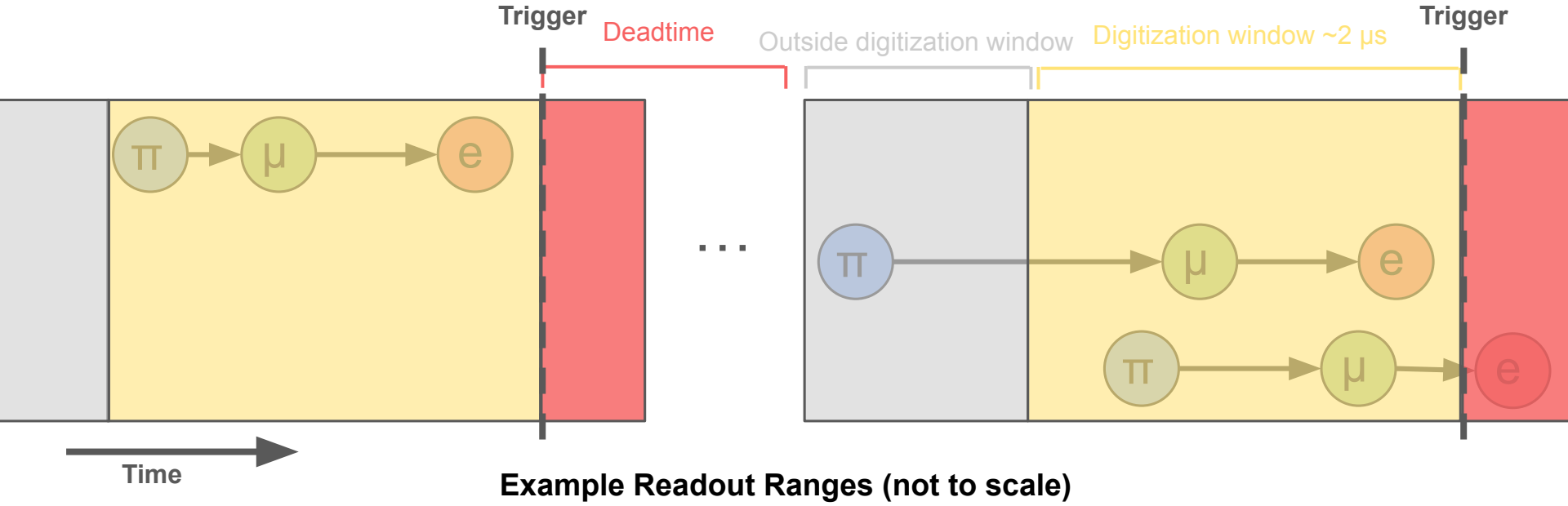
- HDSoC is a candidate ATAR digitization system
  - 1 GbE based readout system
  - 32 channel HDSoC digitizer board in UKy
  - Produces “events”
    - 32 to 1984 samples per triggered channel
- Midas frontend
  - Handles configuration and readout
    - Self triggering/External triggering modes supported
    - Built on existing naludaq python library
  - Handles HDSoCv1 rev2 max data rate of ~55 MB/s
- Documentation (manual) available
  - Details setup, configuration, etc.
- Unpacker app available
  - Unpacks midas data into ROOT tree, stores in analyzable .root files



Nalu’s HDSoC FMC attached to a Nexys A7 Video Card

ATAR Candidate Digitizer: HDSoC DAQ - Strategy

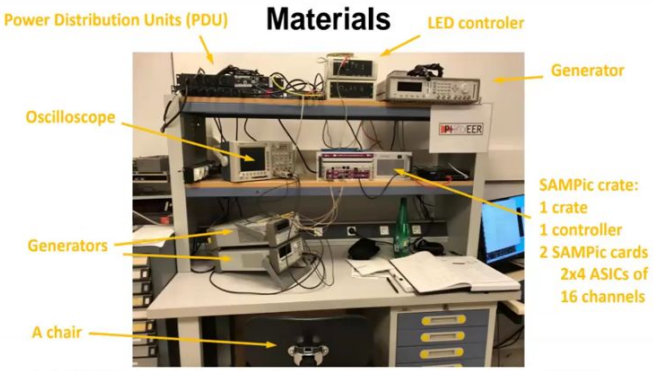
- Given a trigger on positron hit in calo, readout last ~2μs in ATAR
- Challenges:
  - HDSoC needs to zero suppress (currently cannot) or data rate is too large (~10 MB/event or ~100 GB/s)
  - Becomes analysis task to properly construct decay sequences/"patterns"
  - Lookback limited to HDSoC storage ~2μs



Example Readout Ranges (not to scale)

ATAR Candidate Digitizer: SAMPic DAQ

- SAMPic is a candidate ATAR digitization system
  - 1 GbE based readout system
  - 256 channel digitizer crate teststand in LPNHE, Paris
  - Produces “hits”
    - 32 samples from a triggered channel
- Midas frontend
  - Handles configuration and readout
    - Built on existing SAMPic-256ch C library and all it’s different configuration modes
- Unpacker app available
  - Unpacks midas data into ROOT tree, stores in analyzable .root files

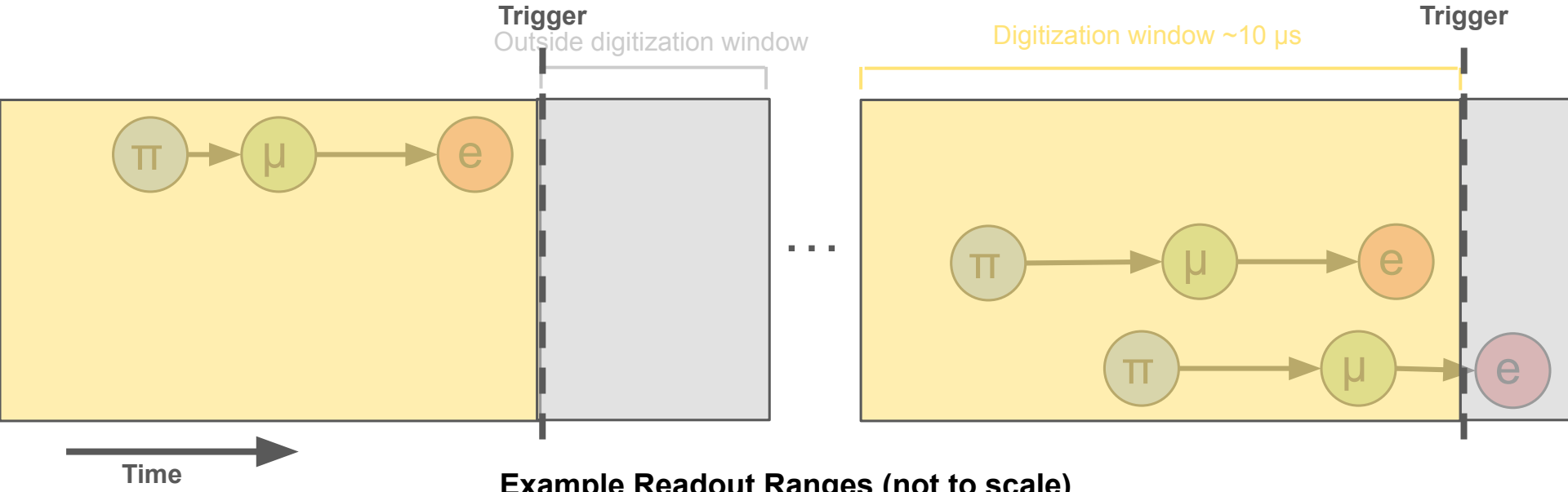


SAMPIC teststand with remote access capabilities

SAMPic Boards

ATAR Candidate Digitizer: SAMPic DAQ - Strategy

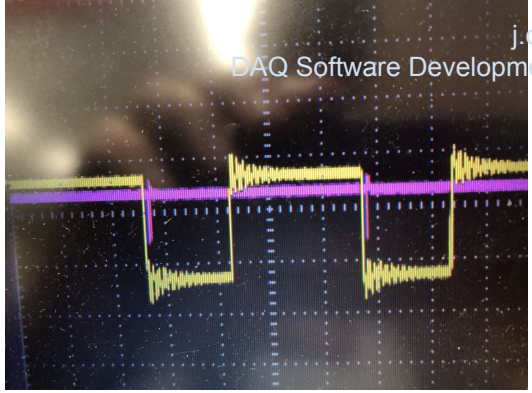
- Given a trigger on positron hit in calo, readout last ~10 μs with signals over threshold in ATAR
- Channels self trigger and store last ~10 μs
- Challenges:
  - Each channel has an individual deadtime ~ 2 μs
  - Deadtime mitigation: use “ping-pong” mapping (1 ATAR channel → 2 SAMPic channels) to “remove” deadtime
    - Readout channel count doubles 4800 → 9600



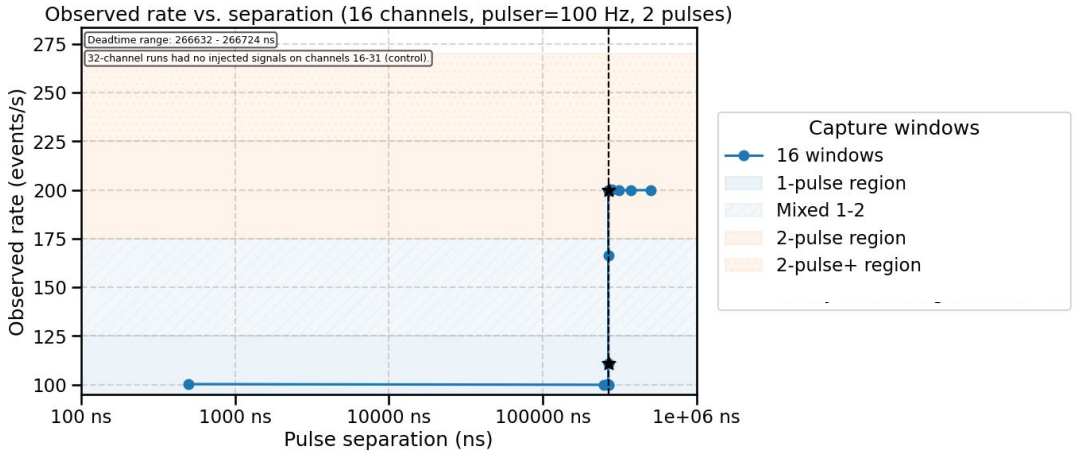
Example Readout Ranges (not to scale)

HDSoC Deadtime Scan

- Used Raspberry Pi Pico W + NIM modules to generated configurable double pulse signal
- Used HDSoC DAQ to observe event rate for varying parameters
- See spike corresponding to deadtime



Example Double pulse on Oscilloscope

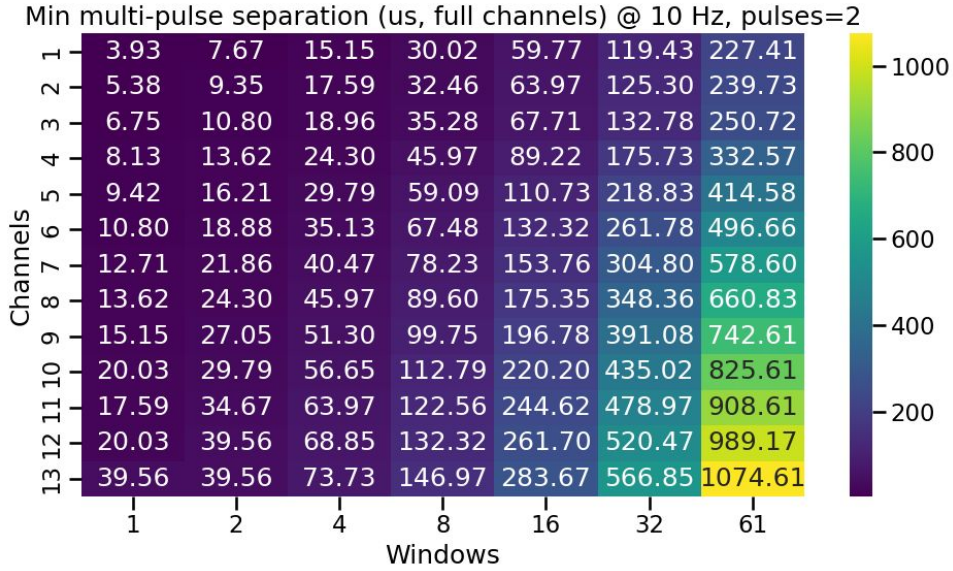


Example Rate Response to Double Pulse Separation

HDSoC Deadtime Scan

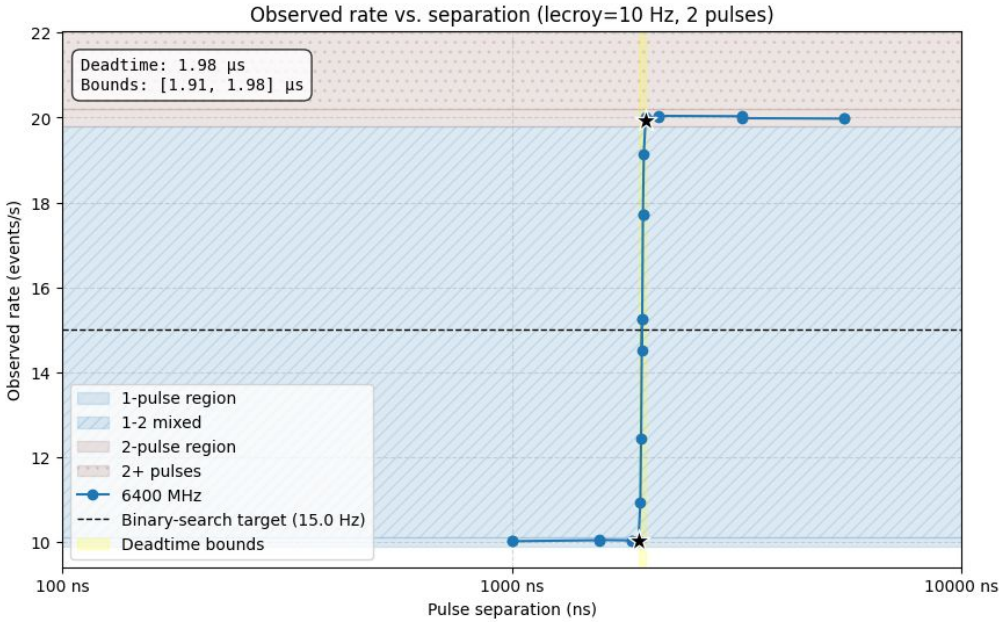
- “Simultaneous” input pulses on each channel, self triggered
- Deadtime scales with number of channels and digitization window
  - 1 window == 32 samples
  - 1 Gsps sampling rate
- Conclusions:
  - Cannot operate experiment in SAMPic style strategy
    - deadtime too long
    - No “ping-pong” mode
    - Deadtime appears to be “chip wide”, i.e. not channel independent
  - Only valid strategy is long (~2us) readout windows per event
    - Long deadtimes make running at high rates a challenge

Parameter Space (91 combinations):
Channels = [1,2,...,13]
Windows = [1,2,4,8,16,32,61]



SAMPic Deadtime Scan

- Used Lecroy 9210 Pulse Generator to create a configurable double pulse signal
- Used sampic_256ch_lib observe hit rate for varying parameters
- See spike corresponding to deadtime

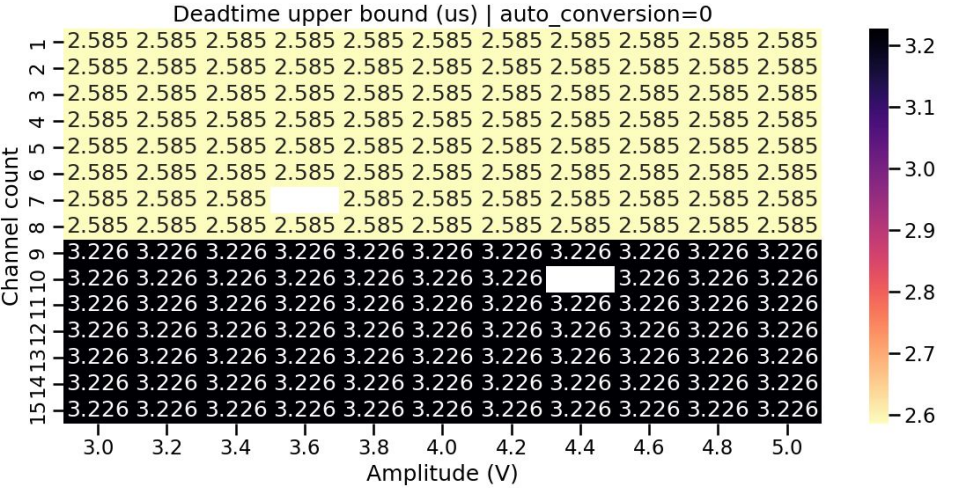


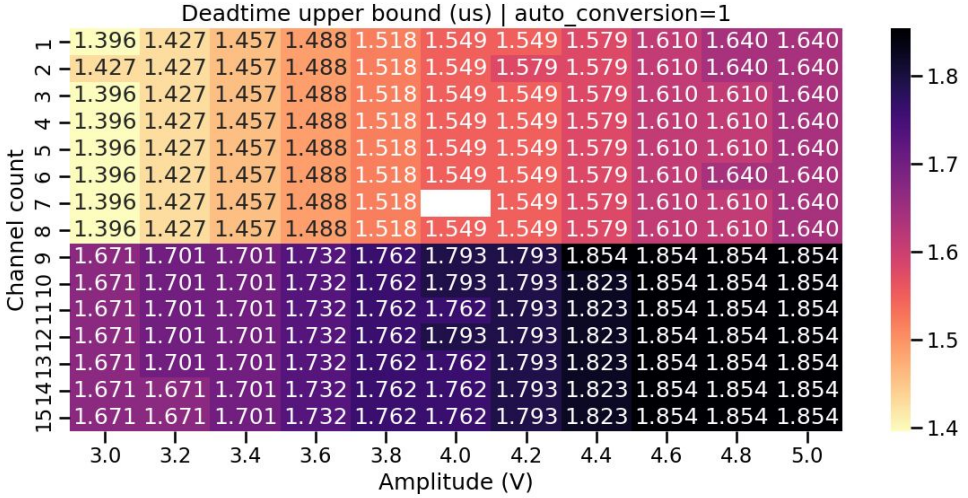
Example rate response to double pulse separation

SAMPic Deadtime Scan

- “Simultaneous” input pulses on each channel, self triggered
- Deadtime doesn’t scale with # of channels (independent channel deadtimes)
  - Channel digitizes 32 samples at specified sample rate, around 1-10 Gsps
- SAMPic has a fixed digitization window of 32 samples
- Unknown Lecroy channel specific behavior causes difference in deadtime
  - Channel count 1-8 is just lecroy channel A
  - Channel count 9-15 includes both lecroy channel A and B

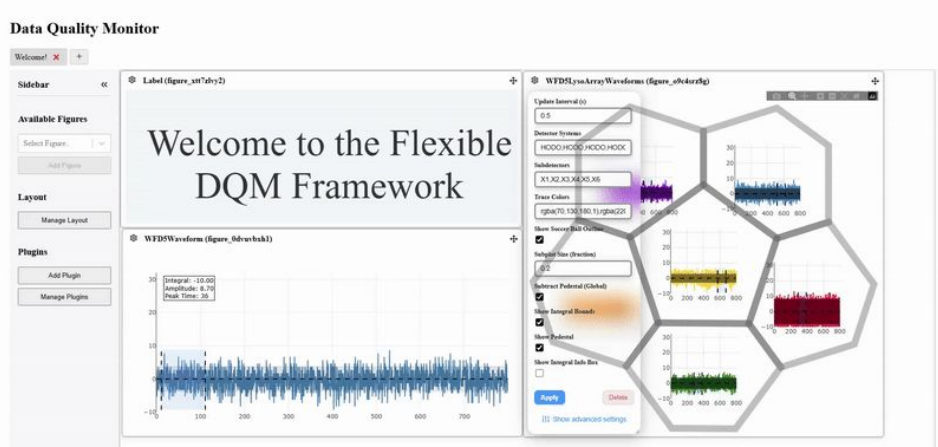
Parameter Space (330 combinations):
Channels = [1,2,...,13,14,15]
Amplitude = [3.0,3.2, … 5.0]
Auto_conversion = [False, True]



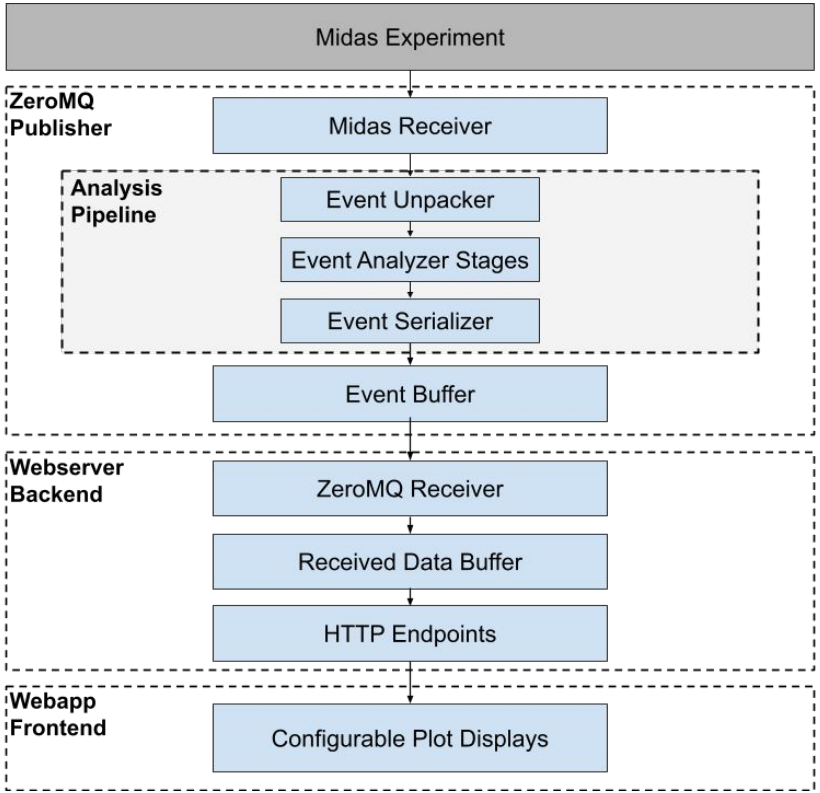


Supplementary Software - Flexible DQM

- Used in PSI 2025 LYSO Testbeam
  - Plugins system for configurability to a variety of experiments (with boilerplate additions)
    - Customizable plot layouts
    - Customizable preliminary “live” processing
- Manual, wikis, tutorials, and repos available



Example DQM webapp page view

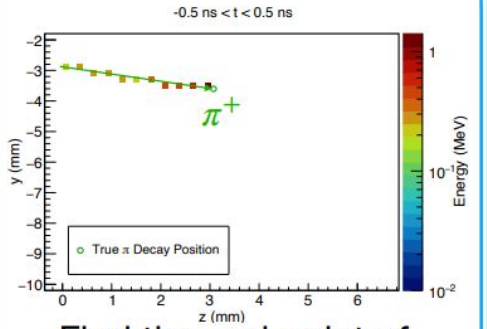


Data flow diagram for DQM

Reconstruction Software Development

Why ATAR Events Are Hard to Reconstruct Traditionally

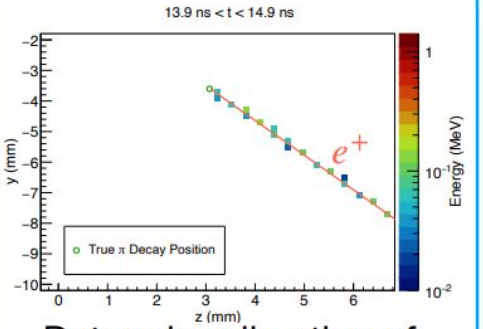
Pion Stop Location



Find the end point of the pion track.

$r_\pi(x_\pi, y_\pi, z_\pi)$

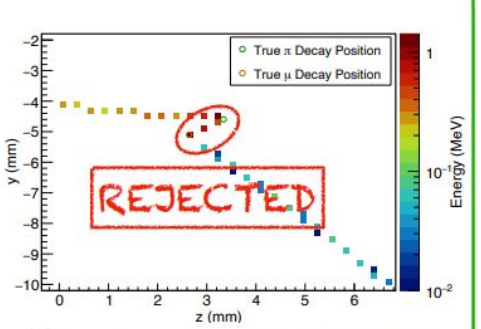
Positron Direction



Determine direction of emitted positron

θ_e, ϕ_e

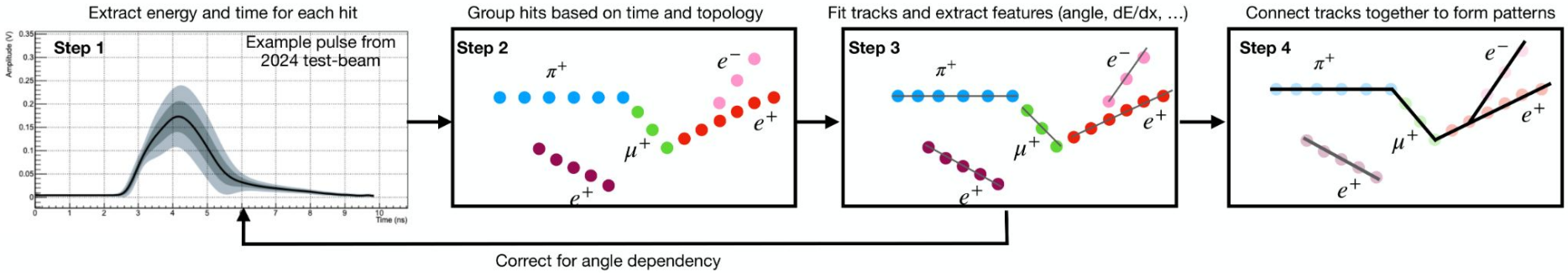
Tail Analysis



Tag $\pi \rightarrow \mu \rightarrow e$ events. Provide 7 orders of magnitude suppression

- Precise pion stop location and positron direction are essential for rebuilding events in our detector geometry
- Must be very efficient at rejecting π→e events that are actually π→μ→e
  - π→μ→e decays are ~10^4 times more frequent than π→e, and $R_{e/\mu}$ must be measured to ~10^{-4} relative precision, the probability for a π→μ→e event to be misidentified as π→e must be suppressed to the ~10^{-8} level
  - Timing and energy information help us so we need ~10^{-7} suppression from tagging

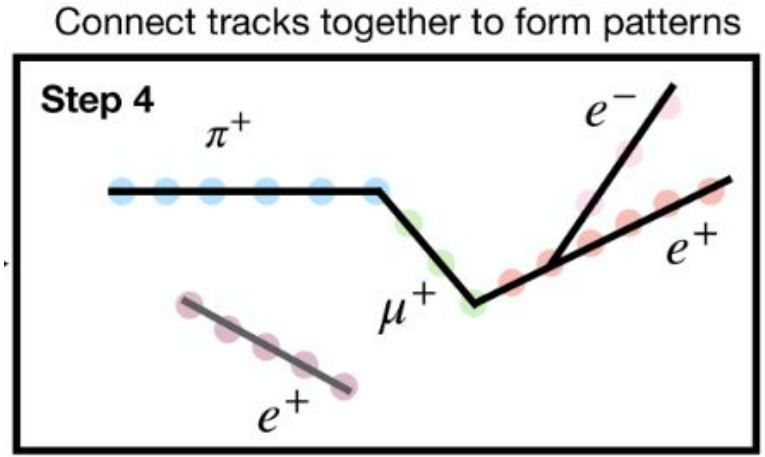
PIONEER Simulation Reconstruction



- Traditional reconstruction is a Gaudi based pipeline
- A series of steps to ultimately classify event types (ex. as π→e or π→μ→e)
- Designed to operate on simulated **and** real detector data
- This is the stage where machine learning/AI approaches can work well
  - Large quantities of detector data with truth targets available from simulation

PIONEER Rules Based Reconstruction: Pattern Finding

- Focused work particularly on last step
  - Collaborative effort with (former) UKy postdoc Sean Foster
- Iterative process involving:
  - Pattern finding algorithm prototyping in custom pattern finding playground framework
    - Python based, allowing for more rapid development
  - Benchmarking performance of pattern finding against truth information from simulation
  - Integrating tested algorithms into simulation framework
    - C++ based (Gaudi Pipeline)



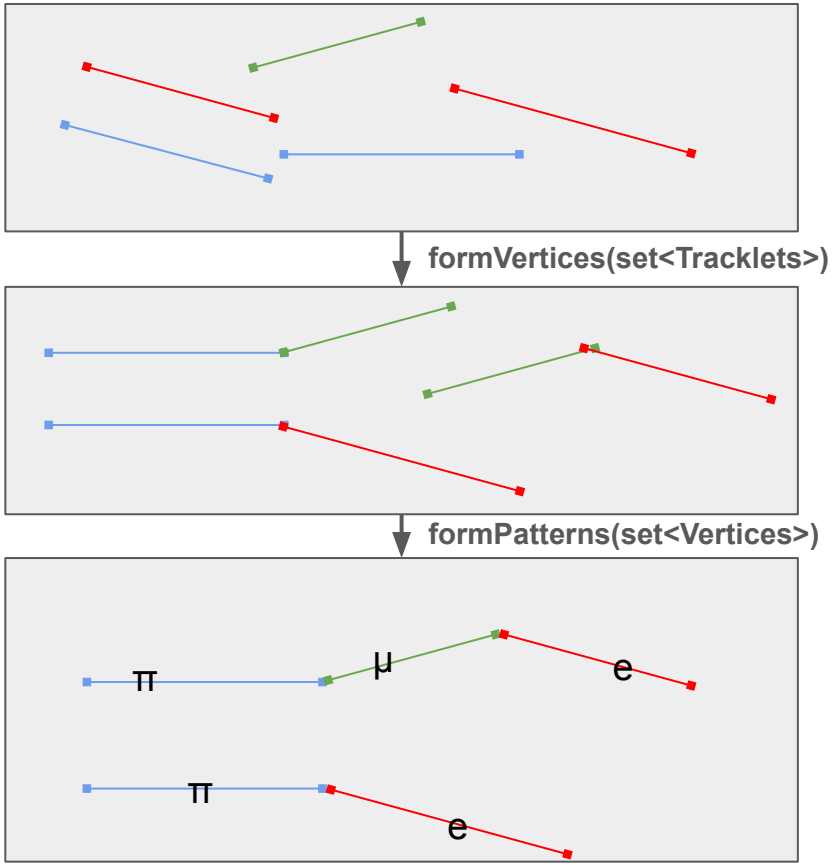
PIONEER Rules Based Reconstruction: Pattern Finding

- **Goal:** Given tracklet objects, form them into physics patterns
- Idea split into two main steps:
  1. Form “Vertices”

$$t_i \equiv \text{"tracklet"}$$
$$v \equiv \{t_1, t_2, \ldots\} \equiv \text{"vertex"}$$
$$t_i \text{ share a common endpoint}$$

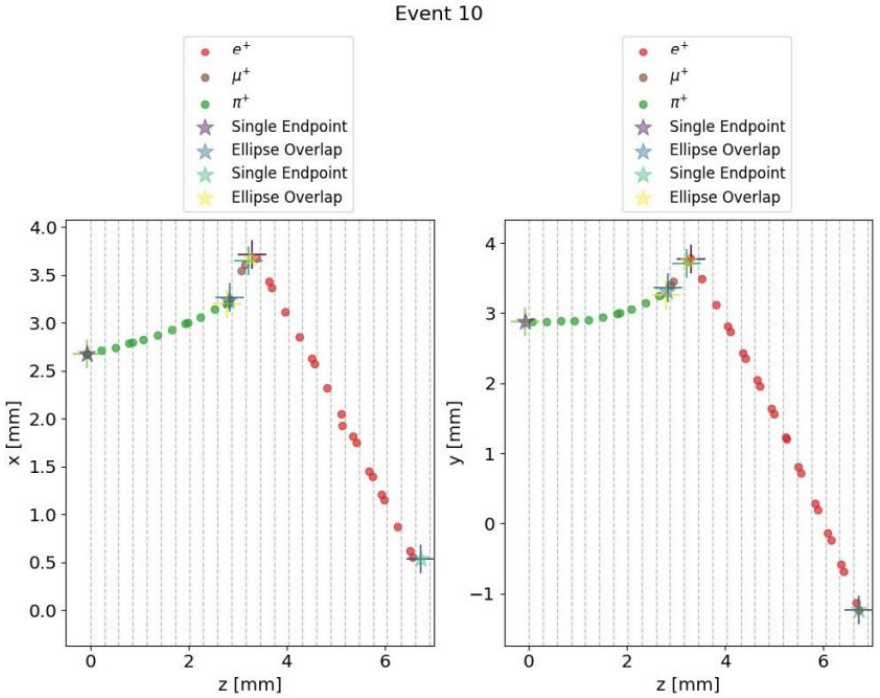
  2. Form “Patterns”

$$G = (V, E)$$
$$v \in V$$
$$(v_i, v_j) \in E \quad \Leftrightarrow \quad v_i \cap v_j \neq \emptyset$$
$$\text{Pattern} \equiv \text{connected component of } G$$



PIONEER Rules Based Reconstruction: Pattern Finding

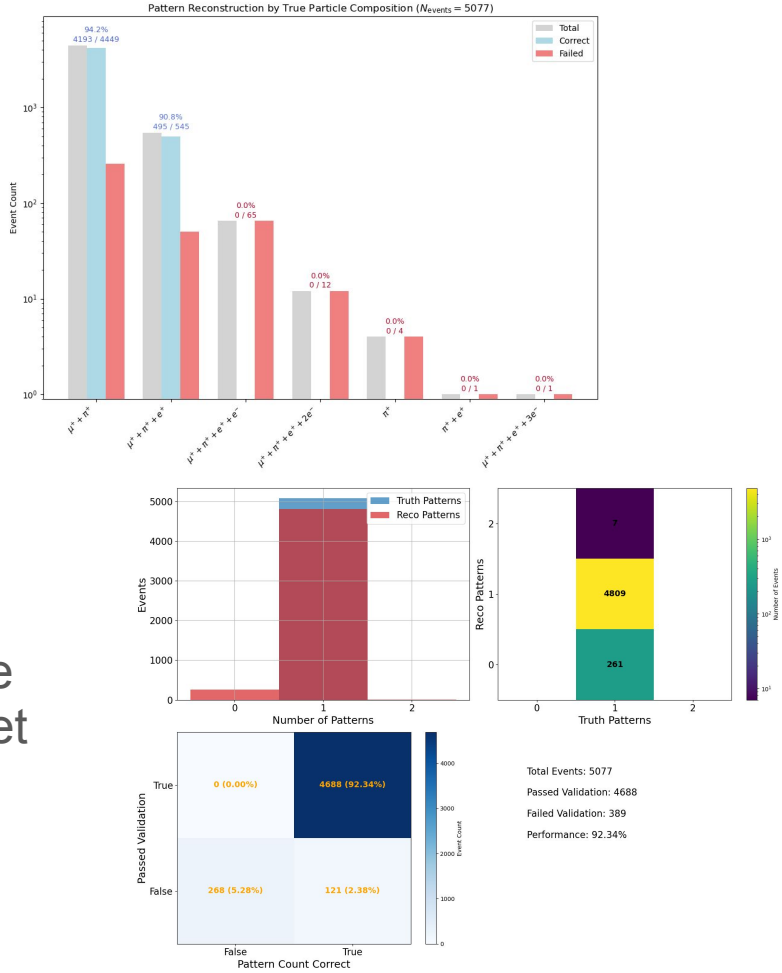
- Vertex forming is the real challenge
  - Pattern forming reduces to depth first search (DFS) on the small (usually <10 nodes) vertex graph
- Vertex Finding Algorithm:
  - Starting from pion and then muon tracklets, endpoint overlaps with other tracklets are evaluated independently in the XZ and YZ projections
  - Overlaps are scored using distance between endpoints, the shortest distance is selected to form a vertex



Example Event (π -> μ -> e):
Tracklets stitched together into vertices based on distance between tracklet endpoints

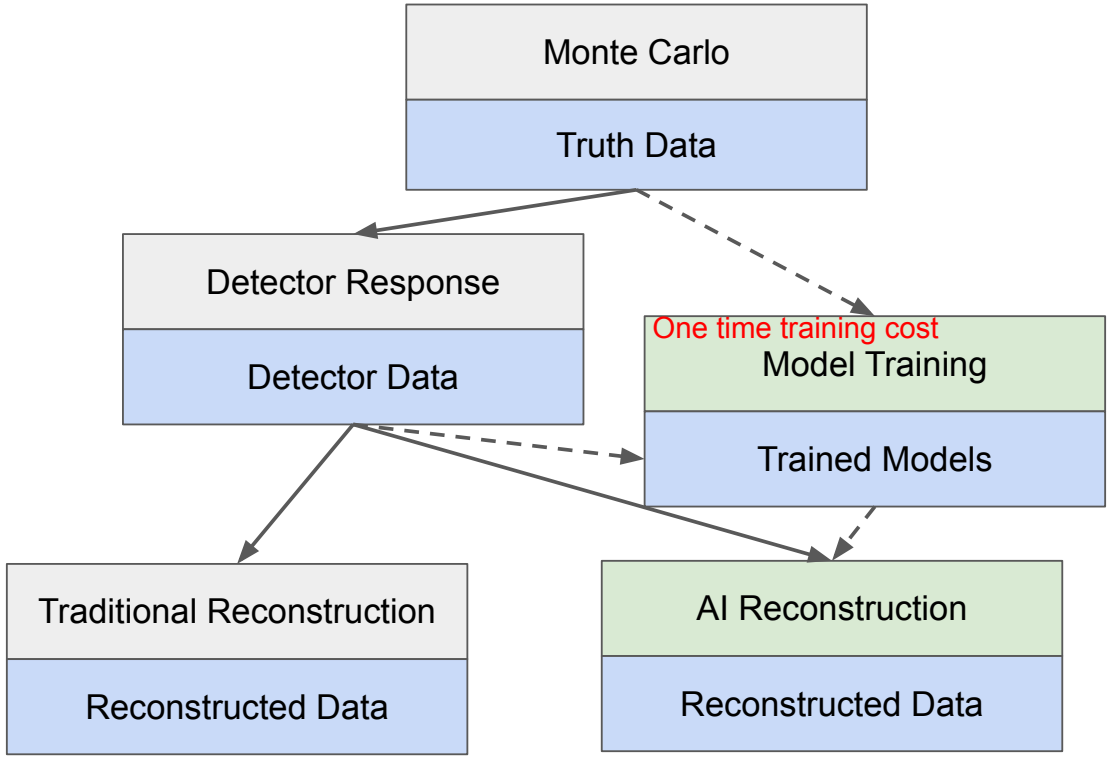
Example Reconstruction Accuracy

- Find the correct pattern **~92%** of time
  - “Pattern finding” is heavily dependent on previous stage “tracklet finding”
    - If tracklet finding fails to appropriate tag particles, pattern finder often fails
  - These statistics are for events without pile-up
  - This statistic is only correlated with pattern finding performance
    - Need work to define better physics based metrics to evaluate pattern finding performance in isolation
- **Takeaway:** still work to be done to refine the edge cases of the pattern finding and tracklet finding algorithms



Where AI/ML Reconstruction Falls in the Pipeline

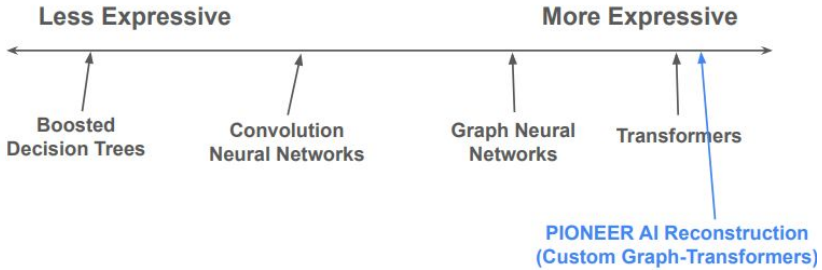
- Simulation provides all needed information
  - Geant4 simulation gives truth targets (ex. Positron angle, true pion stop)
  - Detector response gives inputs (ex. ATAR strip hit 5D information)
- The simulation can produce large quantities of data needed for training
- Project where training and inference is handled in a scalable way
  - Designed to stream data to arbitrary number of compute nodes (ex. GPU cluster)



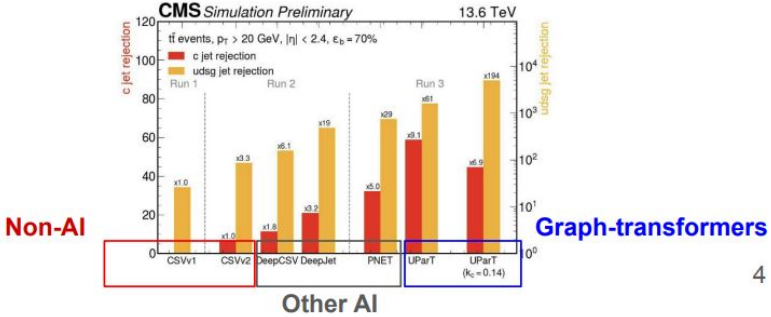
CMS Results Motivate Graph-Transformer Reconstruction

- Among the most expressive ML architectures
  - Solve a broader class of reconstruction problems
- Straightforward to implement with PyG
- Attention based transformers allow models to learn how event information is related
  - Standard GNNs rely on predefined neighborhood aggregation, whereas attention-based models learn which nodes should communicate

AI-based reconstruction:

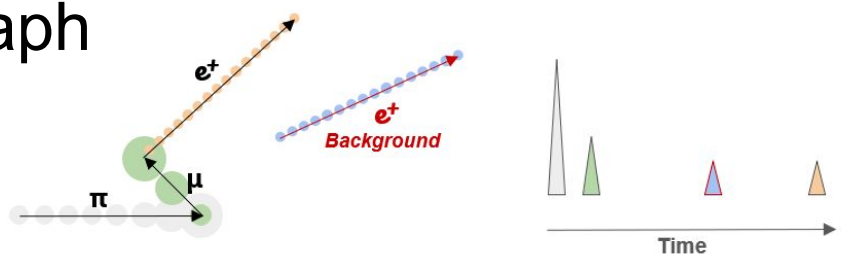


- The UParT clustering model, built on a graph-transformer architecture, is the most successful CMS jet reconstruction algorithm to date (10.22323/1.476.0992)

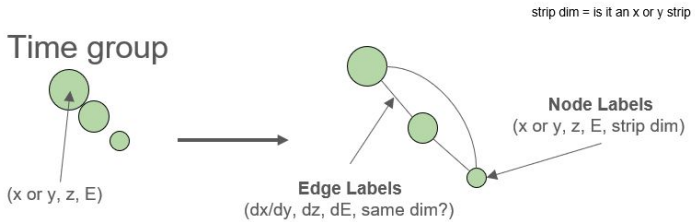


Representing the Data as a graph

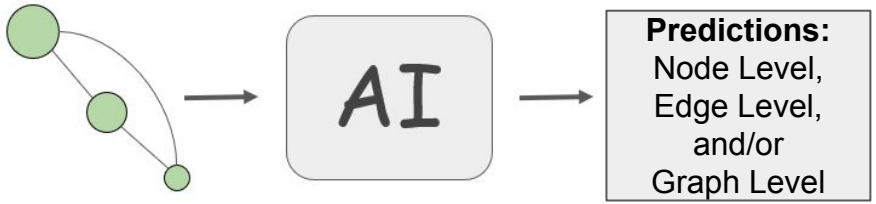
- Start with an detector level event
  - Set of ATAR hits over ~2-10 μs before trigger
- Split into time groups
  - Sequences of hits separated by at most 1ns
- From each time group, create a fully connected graph from hits
  - Nodes contain detector info:
    - (x or y, z, E, strip dim)
  - Edges:
    - (dx or dy, dz, dE, same strip type?)
- Labeled fully connected graphs are inputs to AI models



Example time grouping



Example node and edge labeling



Example of model input and output

= Not an ML stage

= An ML (trained) stage

Current AI/ML ATAR Reconstruction Pipeline

Simulation Pipeline:
Create ML Ready Dataset

Monte Carlo + Detector Response

Truth level descriptions of entire events + detector response; optional event mixing

ML Ready Dataset Gaudi Pipeline

Forms ML ready dataset (parquet files, efficient columnar storage)
Applies preprocessing stages to derive additional feature columns (ex. Time group labels)

AI/ML Pipeline:
Append predictions to events, each stage follows the following prescription:
Event Data → Graphs → Model → Graph Level Predictions → Event Level Predictions

Group Classifier

Adds group level particle classes

Group Splitter

Adds hit level particle classes

Endpoint Regressor

Adds endpoint (x,y,z) estimates

Pattern Finder

Adds affinities between time group

Pion Stop Regressor

Adds pion stops (x,y,z) estimates

Positron Angle Finder

Adds positron angles ($\sin\theta\cos\varphi$, $\sin\theta\sin\varphi$, $\cos\theta$) estimates

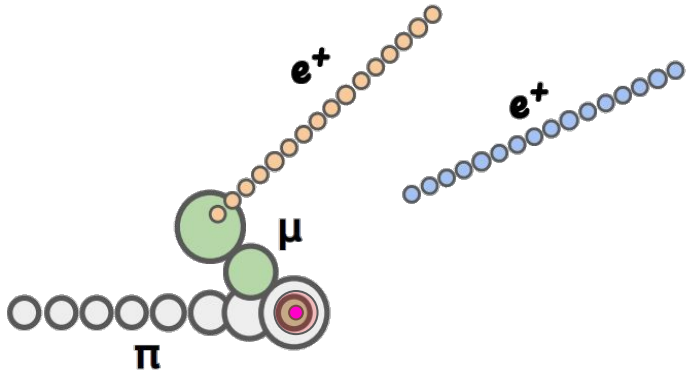
Final Goal: Unify all stages into a single end-to-end ML model

Pion Stop Regressor

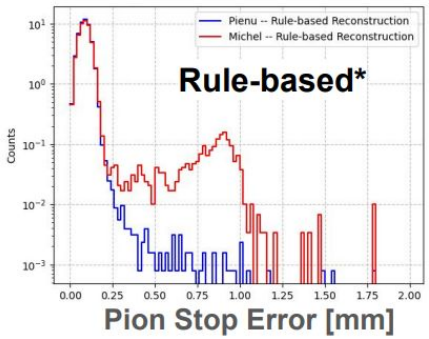
- **Quantile graph transformer regressor**
  - Graph level task
- Inputs (represented as a graph):
  - Detector response event information
  - Time grouping labels
  - Predicted time group level particle class
  - Predicted hit level particle classes
  - Predicted affinities
- Targets:
  - Truth level pion stop
  - Particularly: `[x_stop,y_stop,z_stop]`
- Outputs:
  - Predicted pion stop `[[[x_stop_q1, y_stop_q1, z_stop_q1], …]`
- Notes:
  - Quantiles effectively give uncertainties if chosen to be top 84%, top 50%, and top 16% quantiles = median ± 1 standard deviation

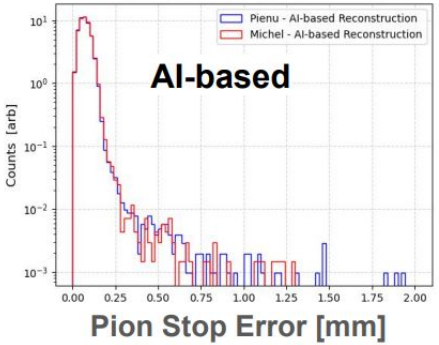
● = Pion stop estimate

● = Error range determined from quantiles



Cartoon example of pion stop estimate

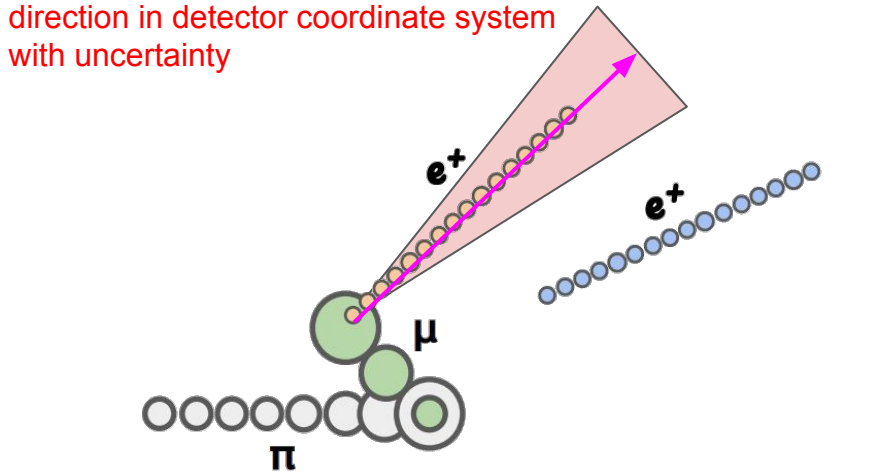




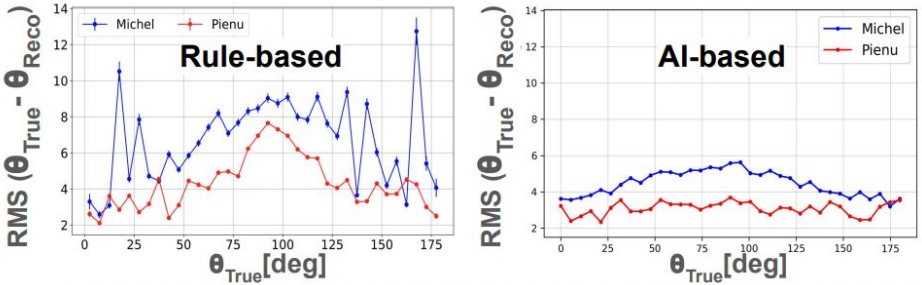
j.carlton@uky.edu

Positron Angle Regressor

- **Quantile graph transformer regressor**
  - Graph level task
- Inputs (represented as a graph):
  - Detector response event information
  - Time grouping labels
  - Predicted time group level particle class
  - Predicted hit level particle classes
  - Predicted affinities
- Targets:
  - Truth level direction vector for positron angle
  - Particularly: `[(sinθcosφ), (sinθsinφ), (cosθ)]`
- Outputs:
  - Predicted direction vector for positron angle `[[(sinθcosφ)_q1, (sinθsinφ)_q1, (cosθ)_q1], …]`



Cartoon example of positron angle estimate

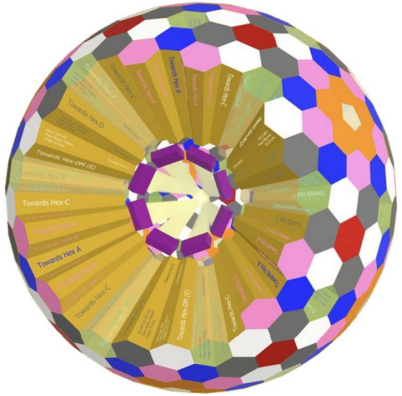


2025 PSI Test Beam

j.carlton@uky.edu

Motivation

- PIONEER LYSO Calorimeter design is a (6,0) Goldberg-Polyhedron with an opening
  - 8 different crystal shapes
    - 1 class of pentagon
    - 7 classes of hexagons
  - 311 crystals for full calo (362 for full sphere)
- **Goal:** Show a LYSO “demonstrator” i.e. segment of a (6,0) Goldberg-Polyhedron meets our key performance parameters (KPPs)
  - **Energy Resolution**
  - Timing Resolution
  - Position Resolution



LYSO Calorimeter: (6,0) Goldberg-Polyhedron with opening. Shape classes color coded

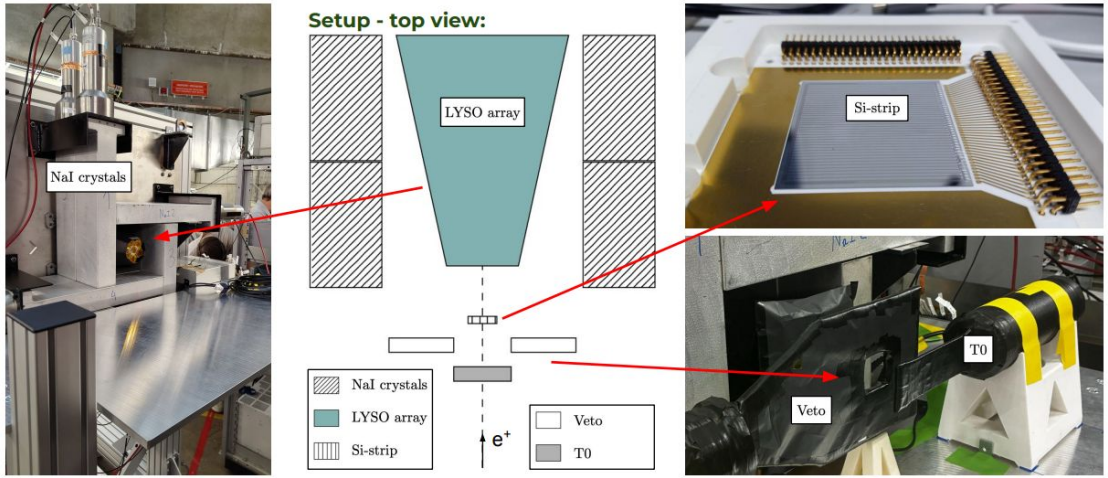
KPPs	Value	Stretch
Effective energy resolution	2 % at 70 MeV 2.5 % at 50 MeV	1.6 - 2.2 % 2.0 - 2.5 %
Timing resolution	200 ps at 56 MeV	100 ps - 300 ps
Position resolution	1 cm	0.5 cm - 3 cm

Key performance parameters (KPPs) targets to meet PIONEER target precision

j.carlton@uky.edu

Beamtime Setup

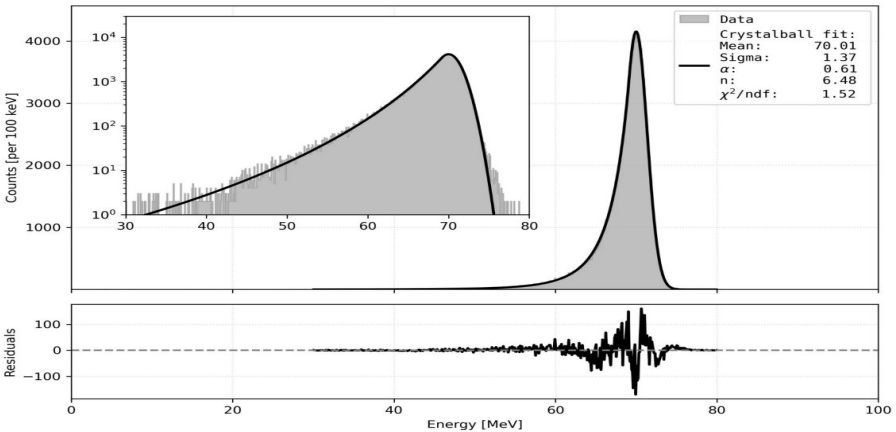
- piE1 beamline
  - Used to steer 20 - 80 MeV positrons towards our detectors
  - Located at PSI, Switzerland
- Core components of our detector setup:
  - **T0**: Defines event start time
  - **Veto**: Used to reject events outside our detector volume
  - **Si-strips**: “Hodoscope” detector used to make more refined spatial cuts on the beam
  - **LYSO array + PMTs**: Used to reconstruct energy of the event
  - **NaI crystals +PMTs**: Used to make light leakage cuts



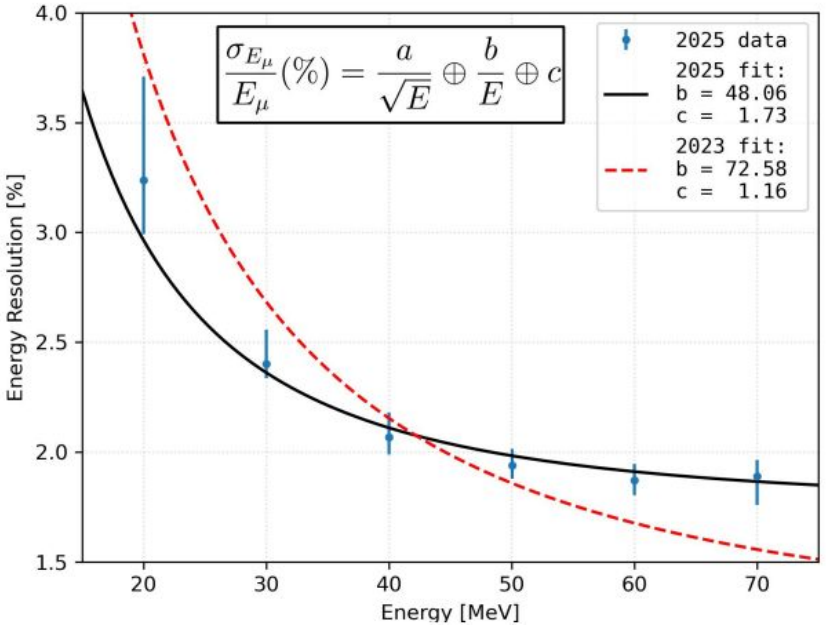
Our test beam experimental setup

Energy Resolution Measurement

- < 2% Energy resolution above 50 MeV
  - Meets KPP target for PIONEER precision
- Similar results to 2023 testbeam
  - More or less expected



Energy reconstruction fit at 70MeV

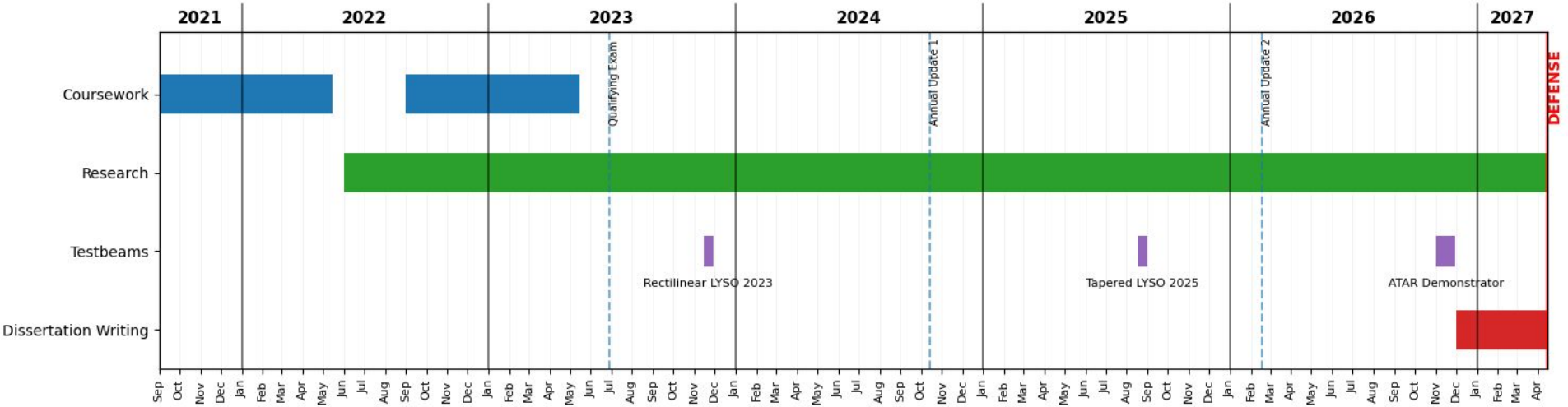


Energy resolution as a function of beam energy

Graduation Planned Timeline

Graduation Timeline

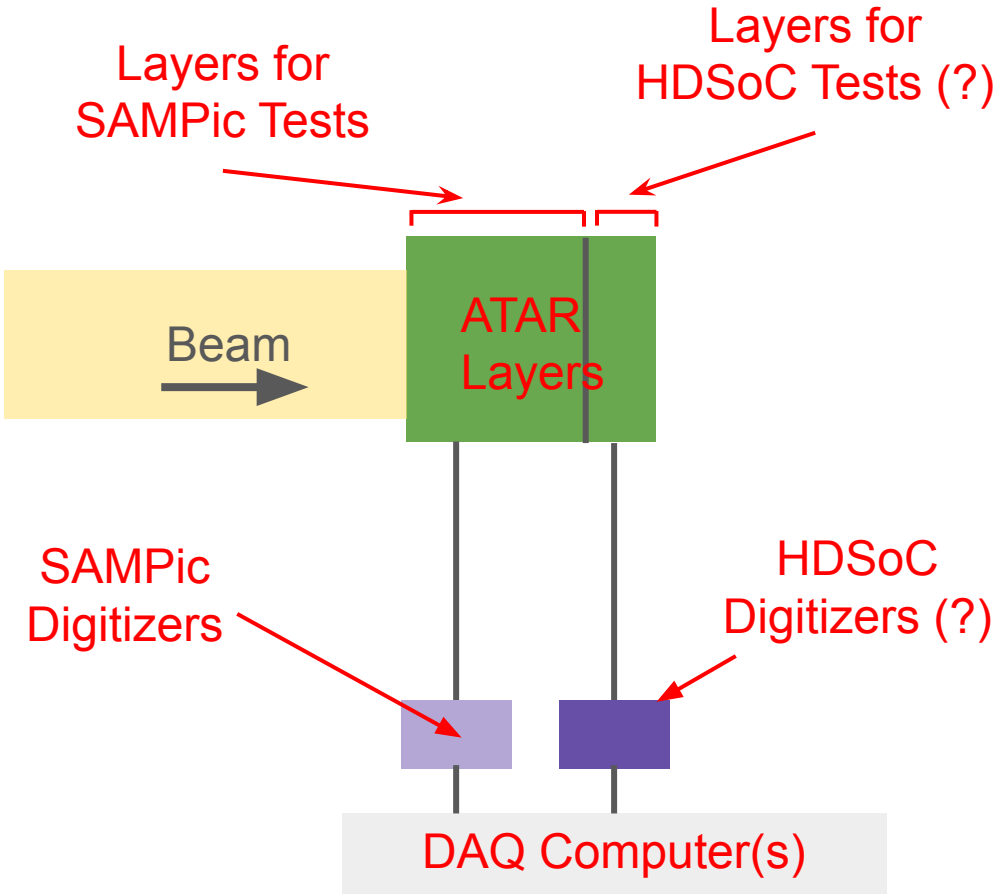
- ATAR Demonstrator planned near end of 2026
  - Provide data to run atar reconstruction code on and evaluate performance
  - This can act as the capstone analysis project for the Ph.D.
- Defend spring semester 2027
  - Gives 4-6 months for dissertation writing and analysis



Research timeline with milestones and testbeams

PIONEER ATAR Demonstrator

- Prototype ATAR demonstrator hardware
  - Smaller number of ATAR layers (16 layers)
- DAQ handles event construction
- Ideas to test both HDSoC and SAMPic in this testbeam
  - Up in the air
- Plan:
  - Run reconstruction (traditional and ML based) on this data for dissertation



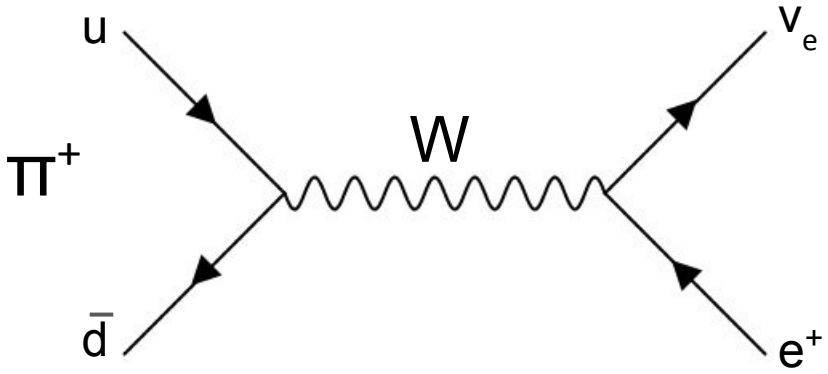
Auxiliary Slides

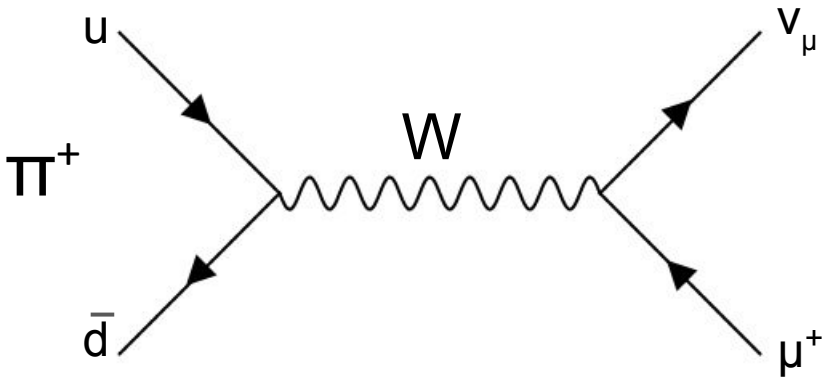
j.carlton@uky.edu

Background Physics

j.carlton@uky.edu

$\pi \rightarrow e\nu_e$ and $\pi \rightarrow \mu\nu_\mu$



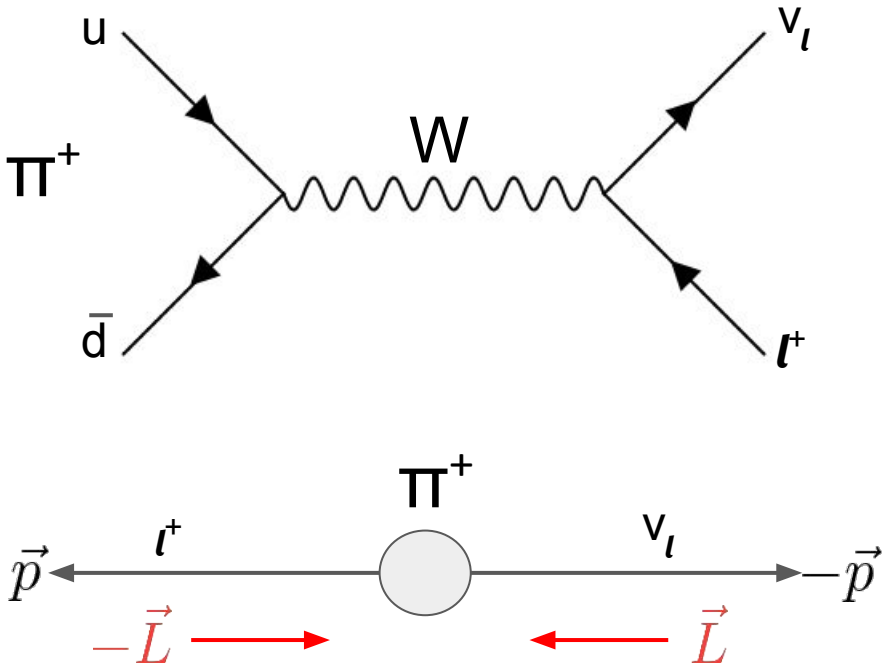


- Corresponding diagrams for π^-
- Tau decay forbidden
  - tau too massive ~ 1000 MeV/c^2
  - Pion ~ 100 MeV/c^2
- Muon decay more likely
  - branching fraction of 0.999877

j.carlton@uky.edu

Helicity Suppression (Why is Muon Decay Most Likely?)

- Naively, $\Gamma \propto p'$ → electron decay more likely
- Weak force only affects left-handed (LH) chiral particle states and right-handed (RH) chiral anti-particle states
- Neutrinos are all LH chirality
- $m_\nu << E$ means LH neutrino chirality→ LH (negative) neutrino helicity
- Conservation of momentum → anti-lepton is LH (negative) helicity



j.carlton@uky.edu

Helicity Suppression (Why is Muon Decay Most Likely?)

- We can write the LH (negative) helicity anti-particle state in the chiral basis:

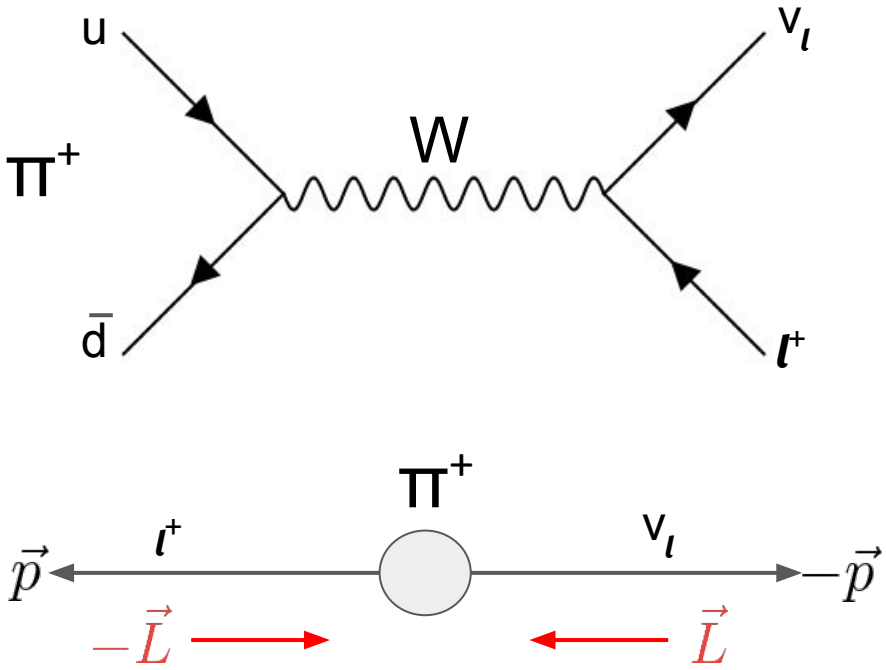
$$v_{\downarrow} = \frac{A}{2}\left[\left(1 - \frac{p}{E+m}\right)v_R - \left(1 + \frac{p}{E+m}\right)v_L\right]$$

- We ignore the LH term (weak force only acts on the RH term), anti-particle's matrix element contribution:

$$\mathcal{M} \sim \frac{1}{2}\left(1 - \frac{p_l}{E_l + m_l}\right) \xrightarrow[m_\nu \to 0]{} \frac{m_l}{m_\pi + m_l}$$

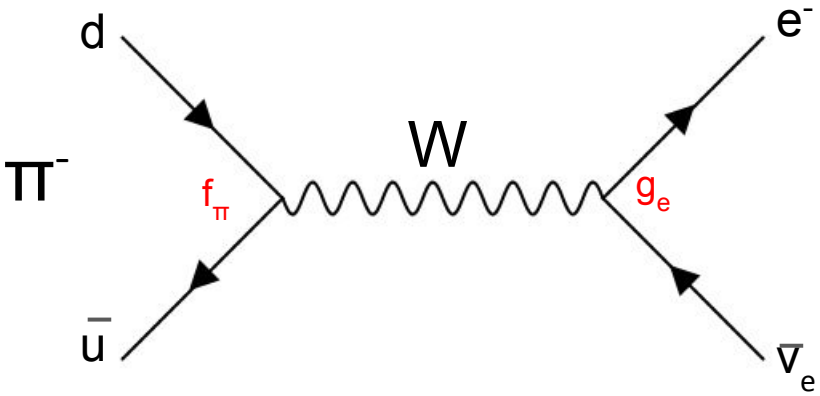
- This effect ends up making the matrix element smaller → decay rate smaller

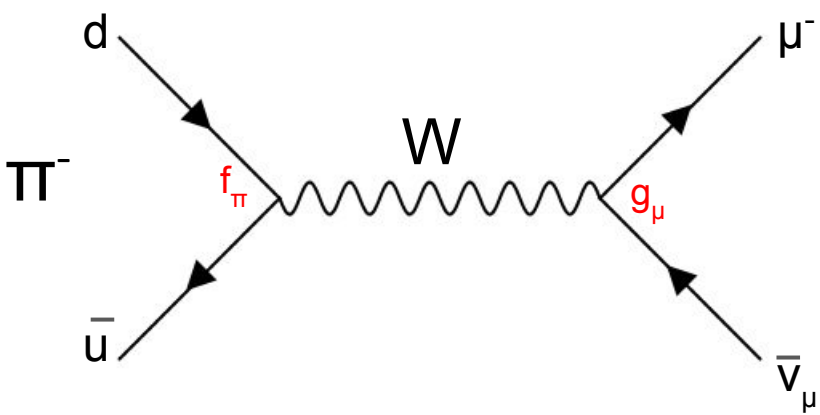
$$\Gamma \propto |\mathcal{M}|^2$$



j.carlton@uky.edu

Lepton Universality





- States coupling strengths (vertices) $g_e = g_\mu = g_\tau$
- Using the Feynman rules for the weak interaction, we can approximate the matrix element

$$\mathcal{M}_{fi} = \underbrace{\left[\frac{g_W}{\sqrt{2}}\frac{1}{2} f_\pi p_\pi^\alpha\right]}_{\text{Pion vertex}} \cdot \underbrace{\left[\frac{g_{\alpha\beta}}{m_W^2}\right]}_{\text{W-boson propagator}} \cdot \underbrace{\left[\frac{g_W}{\sqrt{2}} g_l \bar{u}(p_l)\gamma^\beta \frac{1}{2}(1-\gamma^5) v(p_\nu)\right]}_{\text{Lepton vertex}}$$

j.carlton@uky.edu

Lepton Universality

- After some “massaging” we can find the matrix element to be

$$\mathcal{M}_{fi} = \left(\frac{g_W}{2m_W}\right)^2 f_\pi g_l \cdot \sqrt{m_\pi^2 - m_l^2}$$

- Pion spin zero → no spin averaging needed, i.e.:

$$\langle|\mathcal{M}_{fi}|^2\rangle = |\mathcal{M}_{fi}|^2 = \left(\frac{g_W}{2m_W}\right)^4 f_\pi^2 g_l^2 \cdot (m_\pi^2 - m_l^2)$$

- We can use the general formula for 2-body decay to to find the decay rate

$$\Gamma = \frac{p\langle|\mathcal{M}_{fi}|^2\rangle}{8\pi m_\pi^2} = \frac{f_\pi^2}{16\pi^2 m_\pi^3}\left(\frac{g_W}{2m_W}\right)^4 \left[m_l g_l (m_\pi^2 - m_l^2)\right]^2$$

- Finally, we compute the branching ratio

$$\frac{\Gamma(\pi^- \to e^- \bar{\nu}_e)}{\Gamma(\pi^- \to \mu^- \bar{\nu}_\mu)} = \left(\frac{g_e}{g_\mu}\right)^2 \left[\frac{m_e(m_\pi^2 - m_e^2)}{m_\mu(m_\pi^2 - m_\mu^2)}\right]^2$$

j.carlton@uky.edu

Lepton Universality

$$\frac{\Gamma(\pi^- \to e^- \bar{\nu}_e)}{\Gamma(\pi^- \to \mu^- \bar{\nu}_\mu)} = \left(\frac{g_e}{g_\mu}\right)^2 \left[\frac{m_e(m_\pi^2 - m_e^2)}{m_\mu(m_\pi^2 - m_\mu^2)}\right]^2$$

- Lepton universality assumes $g_e = g_\mu$, so the first factor disappears
- Improving the branching ratio measurement and comparing to the theoretical value acts as a test of lepton universality
- Another test would consider pure leptonic decays, but such decays involving taus are too rare for high precision measurements

j.carlton@uky.edu

Branching Ratio $R_{e/\mu}$

- We can measure the branching ratio by measuring # of decays e and μ decays

$$R_{e/\mu} \equiv \frac{\Gamma(\pi^- \to e^- \bar{\nu}_e)}{\Gamma(\pi^- \to \mu^- \bar{\nu}_\mu)}$$

- Theoretical prediction is simple in first (and second) order
  - No f_π or CKM element V_{ud}

$$R^0_{e/\mu} = \underbrace{\left(\frac{g_e}{g_\mu}\right)^2}_{= 1 \text{ [in theory]}} \left[\frac{m_e(m_\pi^2 - m_e^2)}{m_\mu(m_\pi^2 - m_\mu^2)}\right]^2$$

- 3rd order correction and beyond the pion structure becomes relevant

$$R^{(\text{theory})}_{e/\mu} = R^0_{e/\mu}\left(1 - \frac{3\alpha}{\pi}\ln\left(\frac{m_\mu}{m_e}\right) + \ldots\right)$$

j.carlton@uky.edu

Current state of $R_{e/\mu}$

$$R_{e/\mu}^{exp} = 1.2327(23) \times 10^{-4} \text{ (PIENU collab)}$$

$$R^{theo} = 1.23524(15) \times 10^{-4}$$

- Consistent with each other

- Expect factor of ~10 precision improvement on experimental value from PIONEER
  - “Catches up” with theoretical uncertainty

j.carlton@uky.edu

Common Pion Decay Channels

= Most Common

Leptonic Decay

- $\pi^+ \rightarrow e^+ + \nu_e$
- $\pi^- \rightarrow e^- + \bar{\nu}_e$
- $\pi^+ \rightarrow \mu^+ + \nu_\mu$
- $\pi^- \rightarrow \mu^- + \bar{\nu}_\mu$

Beta Decay

- $\pi^+ \rightarrow \pi^0 + e^+ + \nu_e$
- $\pi^- \rightarrow \pi^0 + e^- + \bar{\nu}_e$

Photon Decay

- $\pi^0 \rightarrow \gamma + \gamma$

Dalitz Decay

- $\pi^0 \rightarrow \gamma + e^- + e^+$

Double-Dalitz Decay

- $\pi^0 \rightarrow e^- + e^+ + e^- + e^+$

Electrons

- $\pi^0 \rightarrow e^- + e^+$

[Note: Dalitz Decays are like photon decays, except the photon(s) are virtual and immediately decay into electron/positron pairs]

j.carlton@uky.edu

Naive Pion Decay, 2-body decay

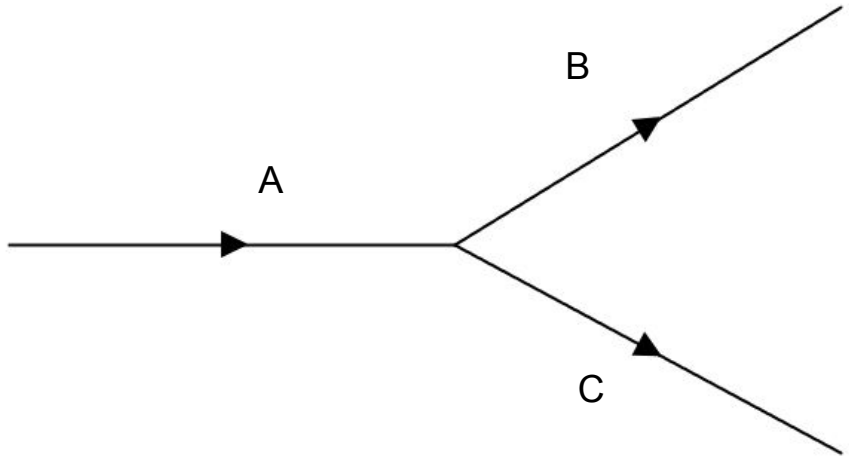
- Without getting into details of QCD, we can treat this as a 3 particle decay
- We can use Fermi's golden rule:

$$d\Gamma = |\mathcal{M}|^2 \cdot \frac{1}{2\hbar m_a} \cdot \left[\frac{c d^3\mathbf{p}_b^2}{(2\pi)^3 2E_b} \cdot \frac{c d^3\mathbf{p}_c^2}{(2\pi)^3 2E_c} \right] \cdot (2\pi)^4 \delta^4(p_a - p_b - p_c)$$

- After integration in the COM frame we find:

$$\Gamma = \frac{|\mathbf{p}|}{8\pi\hbar m_a^2 c} |\mathcal{M}|^2$$
$$\text{where } \mathbf{p} = \mathbf{p}_b = -\mathbf{p}_c$$

- $\longrightarrow \Gamma \propto p$ (not correct)
  - Details hidden in matrix element



j.carlton@uky.edu

Why Massless → Chirality States ~ Helicity States

- Massless → moves at c
- Moves at c → cannot reverse particle direction with Lorentz boost → helicity is Lorentz Invariant
- Chirality is a property of a particle, always Lorentz invariant! → helicity and chirality agree in direction in all inertial reference frames

$$(\gamma^\mu p_\mu - m)u(p) = 0 \quad \text{[Dirac Equation]}$$

$$\Longrightarrow \begin{pmatrix} -mI_{2\times 2} & \sigma\cdot p \\ \bar{\sigma}\cdot p & -mI_{2\times 2} \end{pmatrix} \begin{pmatrix} u_L \\ u_R \end{pmatrix} = 0 \quad \text{[Chiral States]}$$

$$\Longrightarrow \begin{cases} (\sigma\cdot p)u_R - mu_L = 0 \\ (\bar{\sigma}\cdot p)u_L - mu_R = 0 \end{cases}$$

$$m \to 0 \Longrightarrow \begin{cases} (p_0 - \boldsymbol{\sigma}\cdot\mathbf{p})u_R = 0 \\ (p_0 + \boldsymbol{\sigma}\cdot\mathbf{p})u_L = 0 \end{cases}$$

$$\Longrightarrow \begin{cases} \frac{\boldsymbol{\sigma}\cdot\mathbf{p}}{|\mathbf{p}|}u_R = u_R \\ \frac{\boldsymbol{\sigma}\cdot\mathbf{p}}{|\mathbf{p}|}u_L = -u_L \end{cases}$$

$$\hat{h} = \frac{\mathbf{S}\cdot\mathbf{p}}{|\mathbf{p}|} = \frac{1}{2}\frac{\boldsymbol{\sigma}\cdot\mathbf{p}}{|\mathbf{p}|} \quad \text{[Helicity operator]}$$

$$\Longrightarrow \begin{cases} \hat{h}u_R = \frac{1}{2}u_R \\ \hat{h}u_L = -\frac{1}{2}u_L \end{cases} \quad \text{[Chiral states are eigenstates of helicity operator]}$$

j.carlton@uky.edu

LH (negative) helicity spinor to chiral components

An negative helicity antiparticle can be written as

$$v_{\downarrow} = \sqrt{E+m} \begin{pmatrix} \frac{|\mathbf{p}|}{E+m} \cos(\frac{\theta}{2}) \\ \frac{|\mathbf{p}|}{E+m} \sin(\frac{\theta}{2}) e^{i\phi} \\ \cos(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) e^{i\phi} \end{pmatrix}$$

Where (θ,φ) define the direction of the momentum. Without loss of generality, assume the momentum is in the z direction

$$v_{\downarrow} = \sqrt{E+m} \begin{pmatrix} \frac{|\mathbf{p}|}{E+m} \\ \frac{|\mathbf{p}|}{E+m} \\ 1 \\ 1 \end{pmatrix} \equiv A \begin{pmatrix} \tau\xi_R \\ \xi_R \end{pmatrix}$$

j.carlton@uky.edu

LH (negative) helicity spinor to chiral components

We can use the chiral projection operations to project this helicity state to chiral state

$$v_\downarrow = P_L v_\downarrow + P_R v_\downarrow$$

$$P_R = \frac{I_{4\times4} + \gamma^5}{2} = \begin{pmatrix} I_{2\times2} & I_{2\times2} \\ I_{2\times2} & I_{2\times2} \end{pmatrix}$$

$$P_L = \frac{I_{4\times4} - \gamma^5}{2} = \begin{pmatrix} I_{2\times2} & -I_{2\times2} \\ -I_{2\times2} & I_{2\times2} \end{pmatrix}$$

$$v_\downarrow = \frac{A}{2}\left[(1-\tau)\begin{pmatrix} -\xi_R \\ \xi_R \end{pmatrix} + (1+\tau)\begin{pmatrix} \xi_R \\ \xi_R \end{pmatrix}\right] \equiv \frac{A}{2}(1-\tau)v_R - \frac{A}{2}(1+\tau)v_L$$

Where the left and right chiral anti-particle states are defined by

$$P_L v_R = v_R \text{ and } P_R v_L = v_L$$

j.carlton@uky.edu

LH (negative) helicity spinor to chiral components

Looking at the chiral projection of a negative helicity state, we can see in general there are left **and** right chiral components, so the weak force **can** act on a LH (negative) anti-particle helicity state

$$v_{\downarrow} = \frac{A}{2}\left[\left(1 - \frac{p}{E+m}\right) v_R - \left(1 + \frac{p}{E+m}\right) v_L\right]$$

It should also be clear as m→0, the LH (negative) helicity state coincides with the LH chiral state.

This means W boson decay to two massless leptons is forbidden! One of the particles must have the wrong chirality, and thus low mass decays will be suppressed.

j.carlton@uky.edu

Matrix Element Details

$$\mathcal{M}_{fi} = \left[\frac{g_W}{\sqrt{2}}\frac{1}{2} f_\pi p_\pi^\alpha\right] \cdot \left[\frac{g_{\alpha\beta}}{m_W^2}\right] \cdot \left[\frac{g_W}{\sqrt{2}} g_l \bar{u}(p_l)\gamma^\beta \frac{1}{2}(1-\gamma^5) v(p_\nu)\right]$$

Move to pion rest frame so only $p^0 = m_\pi$ remains:

$$\mathcal{M}_{fi} = \frac{g_W^2 f_\pi g_l}{4 m_W^2} m_\pi \bar{u}(p_l)\gamma^0 \frac{1}{2}(1-\gamma^5) v(p_\nu)$$

Using the identity: $\bar{u}(p_l)\gamma^0 = u^\dagger(p_l)\gamma^0\gamma^0 = u^\dagger(p_l) I_{4\times 4} = u^\dagger(p_l)$

$$\mathcal{M}_{fi} = \frac{g_W^2 f_\pi g_l}{4 m_W^2} m_\pi u^\dagger(p_l) \frac{1}{2}(1-\gamma^5) v(p_\nu)$$

j.carlton@uky.edu

Matrix Element Details

For a neutrino m << E so helicity eigenstate is essentially the chiral eigenstate:

$$\frac{1}{2}(1-\gamma^5)v(p_\nu) = v_\uparrow(p_\nu) \implies \mathcal{M}_{fi} = \frac{g_W^2 f_\pi g_l}{4m_W^2} m_\pi u^\dagger(p_l) v_\uparrow(p_\nu)$$

By letting the lepton go in the z-direction we can write:

$$u(p_l) = u_\uparrow(p_l) + u_\downarrow(p_l) = \sqrt{E_l + m_l}\left[\begin{pmatrix} 1 \\ 0 \\ \frac{p}{E_l+m_l} \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 0 \\ \frac{-p}{E_l+m_l} \end{pmatrix}\right] \quad \text{and} \quad v(p_\mu) = v_\uparrow(p_\mu) = \sqrt{p}\begin{pmatrix} 1 \\ 0 \\ -1 \\ 0 \end{pmatrix}$$

Negative helicity lepton down state disappears when “dotted” with the neutrino state:

$$\mathcal{M}_{fi} = \frac{g_W^2 f_\pi g_l}{4m_W^2} m_\pi \sqrt{E_l + m_l}\sqrt{p}\left(1 - \frac{p}{E_l + m_l}\right)$$

j.carlton@uky.edu

Matrix Element Details

$$\mathcal{M}_{fi} = \frac{g_W^2 f_\pi g_l}{4m_W^2} m_\pi \sqrt{E_l + m_l}\sqrt{p}\left(1 - \frac{p}{E_l + m_l}\right)$$

We can re-write E_l and p in the limit where the neutrino mass is zero:

$$E_l = \frac{m_\pi^2 + m_l^2}{2m_\pi} \text{ and } p_l = \frac{m_\pi^2 - m_l^2}{2m_\pi}$$

$$\implies \mathcal{M}_{fi} = \frac{g_W^2 f_\pi g_l}{4m_W^2} m_\pi \cdot \frac{m_\pi + m_l}{\sqrt{2m_\pi}} \cdot \left(\frac{m_\pi^2 - m_l^2}{2m_\pi}\right)^{\frac{1}{2}} \cdot \frac{2m_l}{m_\pi + m_l}$$

$$\implies \mathcal{M}_{fi} = \frac{g_W^2 f_\pi g_l}{4m_W^2} \cdot m_l (m_\pi^2 - m_l^2)^{\frac{1}{2}}$$

j.carlton@uky.edu

Lepton Universality

$$\text{Note: } x^2 = 1 + 2(x-1) + \mathcal{O}(x^2)$$

$$\text{Let: } \left(\frac{g_e}{g_\mu}\right) \equiv (1 + \Delta_{\frac{g_e}{g_\mu}}) \equiv x$$

$$R_{e/\mu} = \frac{\Gamma(\pi^- \to e^- \bar{\nu}_e)}{\Gamma(\pi^- \to \mu^- \bar{\nu}_\mu)} \approx (1+2\Delta_{\frac{g_e}{g_\mu}}) \left[\frac{m_e(m_\pi^2 - m_e^2)}{m_\mu(m_\pi^2 - m_\mu^2)}\right]^2$$

$$\text{Let: } \Delta R_{e/\mu} = R_{e/\mu} - \left(R_{e/\mu}\right)_{\text{theory}}$$

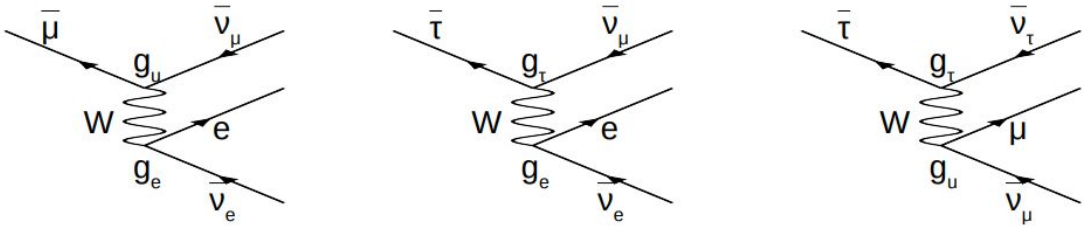
$$\boxed{\frac{\Delta R_{e/\mu}}{\left(R_{e/\mu}\right)_{\text{theory}}} = 2\Delta_{\frac{g_e}{g_\mu}}} \implies$$

Small discrepency in g_e/g_μ and 1 can cause twice as big discrepency in measured $R_{e/\mu}$ and theory $R_{e/\mu}$

j.carlton@uky.edu

Another Test for Lepton Universality

$\mu \to e\, \nu\, \bar{\nu}$ $\tau \to e\, \nu\, \bar{\nu}$ $\tau \to \mu\, \nu\, \bar{\nu}$



Fermi constant, $G_F = g^2 / 4\sqrt{2}M_W^2$

$G_{\mu e} = 1.166\ 378\ 7(6) \times 10^{-5}\ GeV^{-2}$ (0.5 ppm)

$G_{\tau\mu} = 1.1665(28) \times 10^{-5}\ GeV^{-2}$ (0.2%)

$G_{\tau e} = 1.1665(28) \times 10^{-5}\ GeV^{-2}$ (0.2%)

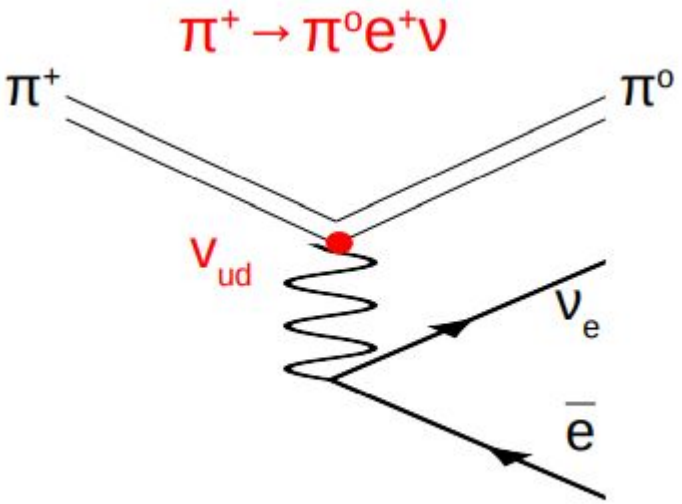
weak coupling, g

$g_e : g_\mu : g_\tau$ 1 : 1.0011(24) : 1.0006(24)

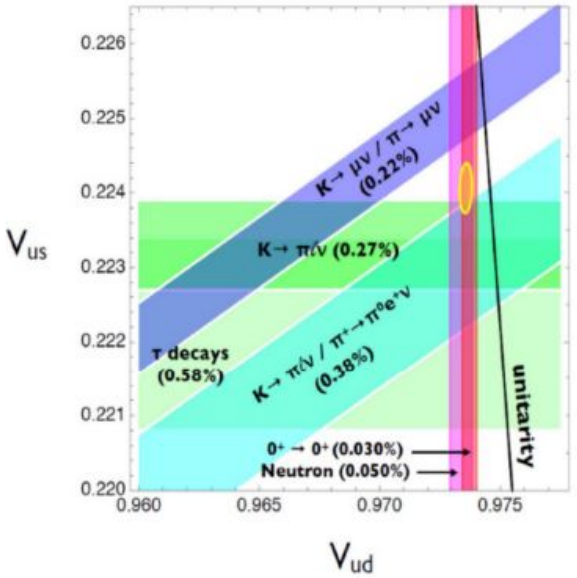
Citation: Testing Lepton Flavor Universality and CKM Unitarity with Rare Pion Decays in the PIONEER experiment, PIONEER collab (pg. 21)
PIONEER Seminar, Tim Gorringe (Slide 12)

j.carlton@uky.edu

CKM Unitary Test



arXiv:2203.05505



- Pion beta decay gives a precision measurement of V_{ud}
- These decays are lower rate than $\pi \rightarrow e\nu_e$ and $\pi \rightarrow \mu\nu_\mu$
- Experimental measurements do not agree

Citations: A measurement of the scintillation decay time constant of nuclear recoils in liquid xenon with the XMASS-I detector, XMASS collab, arxiv 1809.05988
Scintillation yield of liquid xenon at room temperature , XMASS collab, arxiv 0803.2888
Berkeley Nucleonics (https://www.berkeleynucleonics.com/nai-sodium-iodide)

j.carlton@uky.edu

Some Information about LXe and NaI

- LXe has singlet and triplet state decay constants:
  - $\tau_S = 4.3 \pm 0.6$ ns
  - $\tau_T = 26.9^{+0.7}_{-1.1}$ ns
- LXe light yield:
  - ~29 photons/keV at room temp

- NaI decay constant:
  - ~ 250 ns
- NaI light yield:
  - 38 photons/keV at room temp

Scintillation from excited Xe (Xe*):

$$Xe^* + Xe + Xe \rightarrow Xe_2^* + Xe\,,$$
$$Xe_2^* \rightarrow 2Xe + h\nu\,.$$

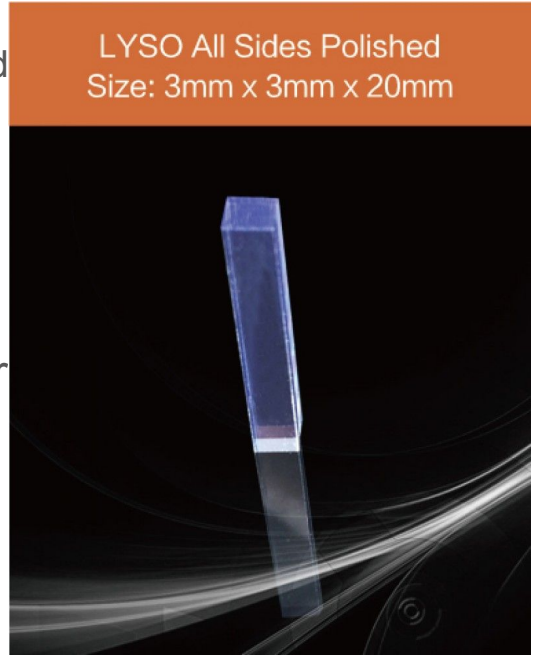
Scintillation from ionized Xe (Xe^+):

$$Xe^+ + Xe \rightarrow Xe_2^+\,,$$
$$Xe_2^+ + e^- \rightarrow Xe^{**} + Xe\,,$$
$$Xe^{**} \rightarrow Xe^* + heat\,,$$
$$Xe^* + Xe + Xe \rightarrow Xe_2^* + Xe\,,$$
$$Xe_2^* \rightarrow 2Xe + h\nu\,.$$

j.carlton@uky.edu

LYSO Information

- LYSO – lutetium–yttrium oxyorthosilicate
  - Lutetium (73%), Oxygen (18%), Silicon (6%), Yttrium (3%), and a Cerium scintillation dopant (~ 0%)
- Density = **7.4 g/cm^3**
- X_0 = **1.14 cm** = “Radiation length” = distance for an electron's energy to be reduced by a factor of 1/e
- R_M = **2.07 cm** = “Moliére radius” = radius of a cylinder containing on average 90% of the shower's energy deposition
- Light Yield = **30,000 photons/MeV**
- Scintillating decay time = **40 ns**



j.carlton@uky.edu

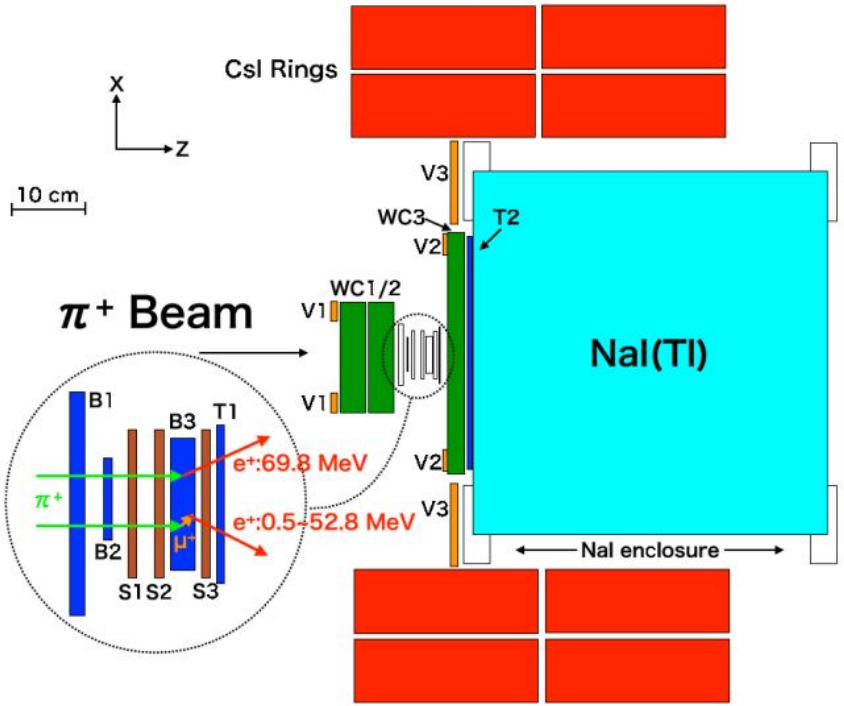
PIONEER Experiment

Citation: Status of the TRIUMF PIENU Experiment, PIENU collab, (arxiv 1509.08437)
https://pienu.triumf.ca/

j.carlton@uky.edu

Past Experimental Approach (PIENU)

- NaI has a long primary decay time
  - ~ 250 ns
- Event pileup forces the experiment to run at a low rate
  - ~70 kHz
- “inactive target”, muons aren’t tracked
- CsI Rings for shower leakage detection

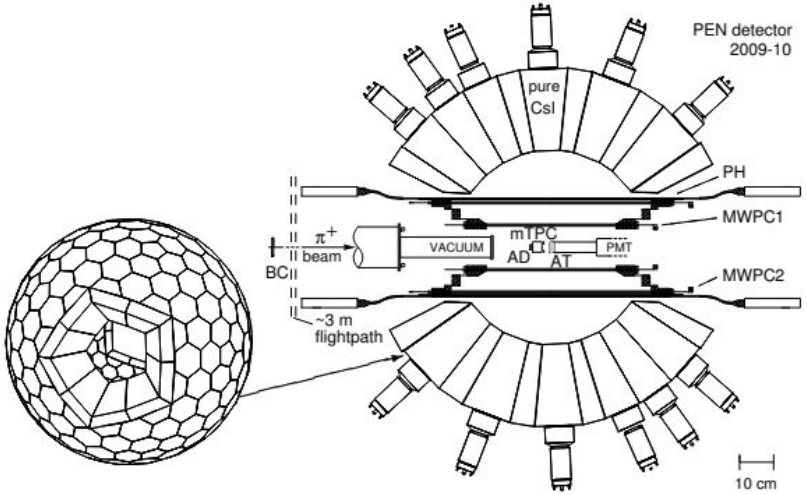


Citation:PEN: a low energy test of lepton universality ,
PENcollab, (arxiv: 1701.05254)

j.carlton@uky.edu

PEN

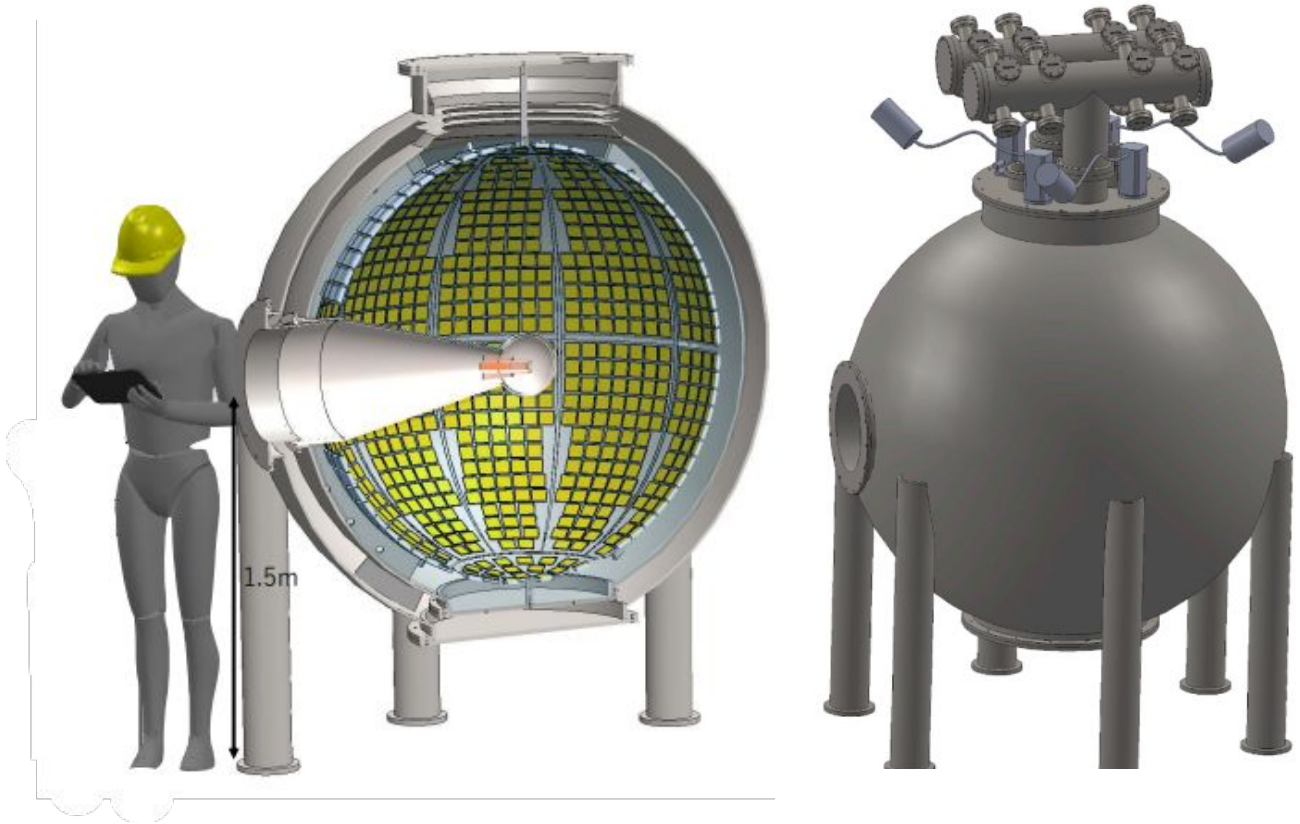
- Similar to PIENU
  - Segmented
  - Better timing
- Many channels of pure CSI
  - 240 channels
- Active target



Citations: PIONeer A next-generation rare pion decay experiment, David Hertzog, (https://pioneer.npl.washington.edu/docdb/0002/000281/001/PIONEER%20Orientation%20and%20Status%20June%202024.pdf)

j.carlton@uky.edu

Another Calorimeter 3D Render (Liquid Xenon)



Citations: PIONEER Slides from ACTS Workshop, Patrick Schwendimann (Slide 7)

PIONEER Strategy Tail Correction

$N_H^{\pi e} \equiv N(\pi \to e\nu,\ E_{\text{reco}} > E_{\text{cut}}) \equiv$ true $\pi \to e\nu$ above threshold
$N_H^{\pi\mu} \equiv N(\pi \to \mu\nu,\ E_{\text{reco}} > E_{\text{cut}}) \equiv$ true $\pi \to \mu\nu$ above threshold ≈ 0 (by construction)
$N_L^{\pi e} \equiv N(\pi \to e\nu,\ E_{\text{reco}} < E_{\text{cut}}) \equiv$ true $\pi \to e\nu$ below threshold
$N_L^{\pi\mu} \equiv N(\pi \to \mu\nu,\ E_{\text{reco}} < E_{\text{cut}}) \equiv$ true $\pi \to \mu\nu$ below threshold

$N_H \equiv N(E_{\text{reco}} > E_{\text{cut}}) \equiv$ observed events above threshold
$N_L \equiv N(E_{\text{reco}} < E_{\text{cut}}) \equiv$ observed events below threshold

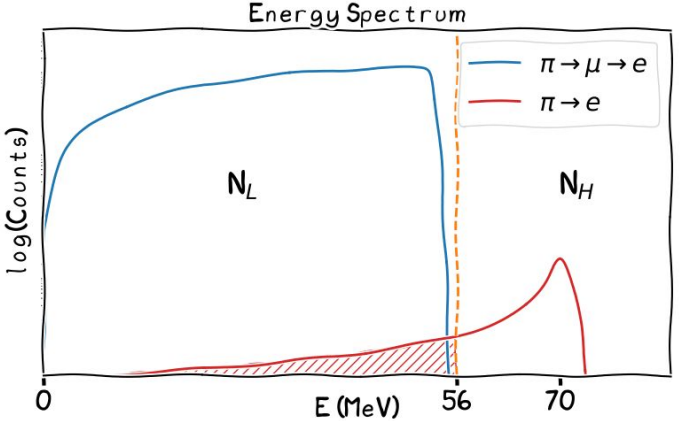
$N_{\text{total}} \equiv$ all observed events

$$\frac{N_H}{N_L} = \frac{N_H^{\pi e} + N_H^{\pi\mu}}{N_L^{\pi e} + N_L^{\pi\mu}} \approx \frac{N_H^{\pi e}}{N_L^{\pi e} + N_L^{\pi\mu}}$$

Assume negligible leakage of muon decays above threshold

$$R_{e/\mu}^{\text{true}} \equiv \frac{N_H^{\pi e} + N_L^{\pi e}}{N_H^{\pi\mu} + N_L^{\pi\mu}} \approx \frac{N_H^{\pi e} + N_L^{\pi e}}{N_L^{\pi\mu}}$$

$$R_{e/\mu}^{\text{true}} = \left[\frac{N_H}{N_L} \cdot \frac{1}{\frac{N_H}{N_L}}\right] \cdot \frac{N_H^{\pi e} + N_L^{\pi e}}{N_L^{\pi\mu}} = \left[\frac{N_H}{N_L} \cdot \frac{N_L^{\pi e} + N_L^{\pi\mu}}{N_H^{\pi e}}\right] \cdot \frac{N_H^{\pi e} + N_L^{\pi e}}{N_L^{\pi\mu}}$$



$$R_{e/\mu}^{\text{true}} = \frac{N_H}{N_L}\left(\frac{N_L^{\pi e}N_H^{\pi e} + (N_L^{\pi e})^2 + N_L^{\pi\mu}N_H^{\pi e} + N_L^{\pi\mu}N_L^{\pi e}}{N_H^{\pi e}N_L^{\pi\mu}}\right)$$

$$= \frac{N_H}{N_L}\left(\frac{N_L^{\pi e}N_H^{\pi e} + (N_L^{\pi e})^2 + N_L^{\pi\mu}N_H^{\pi e} + N_L^{\pi\mu}N_L^{\pi e}}{N_H^{\pi e}N_L^{\pi\mu}}\right) \cdot \left[\frac{\frac{1}{N_{\text{total}}^2}}{\frac{1}{N_{\text{total}}^2}}\right]$$

$$= \frac{N_H}{N_L}\left(\frac{\frac{N_L^{\pi e}N_H^{\pi e}}{N_{\text{total}}^2} + \frac{(N_L^{\pi e})^2}{N_{\text{total}}^2} + \frac{N_L^{\pi\mu}N_H^{\pi e}}{N_{\text{total}}^2} + \frac{N_L^{\pi\mu}N_L^{\pi e}}{N_{\text{total}}^2}}{\frac{N_H^{\pi e}N_L^{\pi\mu}}{N_{\text{total}}^2}}\right)$$

$$\frac{N_L^{\pi e}}{N_{\text{total}}} \sim 10^{-5}, \quad \frac{N_H^{\pi e}}{N_{\text{total}}} \sim 10^{-4}, \quad \frac{N_L^{\pi\mu}}{N_{\text{total}}} \sim 1 - 10^{-4}$$

Removed terms ~10^{-8} relative precision, far beyond goal of ~10^{-4}

$$\approx \frac{N_H}{N_L}\left(\frac{\frac{N_L^{\pi\mu}N_H^{\pi e}}{N_{\text{total}}^2} + \frac{N_L^{\pi\mu}N_L^{\pi e}}{N_{\text{total}}^2}}{\frac{N_H^{\pi e}N_L^{\pi\mu}}{N_{\text{total}}^2}}\right) = \frac{N_H}{N_L}\left(\frac{N_H^{\pi e} + N_L^{\pi e}}{N_H^{\pi e}}\right) = \frac{N_H}{N_L}\left(1 + \frac{N_L^{\pi e}}{N_H^{\pi e}}\right)$$

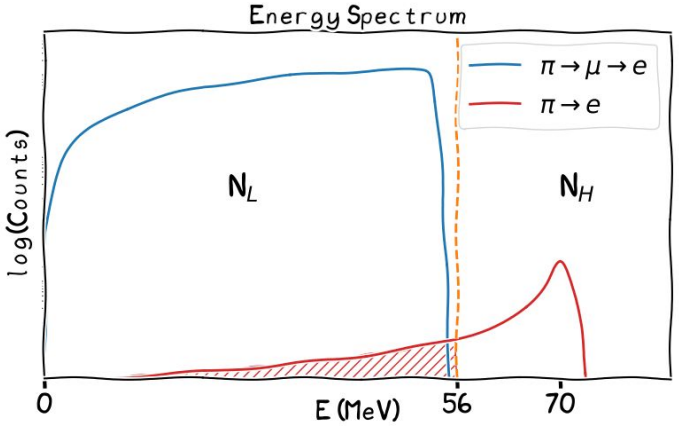
$$\equiv \frac{N_H}{N_L}(1 + C_T)$$

where $c_T \approx \frac{N_L^{\pi e}}{N_H^{\pi e}}$

Citations: PIONEER Slides from ACTS Workshop, Patrick Schwendimann (Slide 7)

PIONEER Strategy Acceptance Correction

- In reality, our efficiency for high and low energy are not the same, requiring another correction term



$$R^{\varepsilon} \equiv \frac{\varepsilon_H}{\varepsilon_L}$$

ε_H : probability that a true $\pi \to e\nu$ event is counted in N_H

ε_L : probability that a true $\pi \to \mu \to e$ event is counted in N_L

$$R_{e/\mu} = \frac{\Gamma(\pi^+ \to e^+\nu(\gamma))}{\Gamma(\pi^+ \to \mu^+\nu(\gamma))} \approx \frac{N_H}{N_L} \times (1+c_T) \times R^{\epsilon}$$

Tail Correction

$c_T \approx 5 \times 10^{-3}$

Acceptance Correction

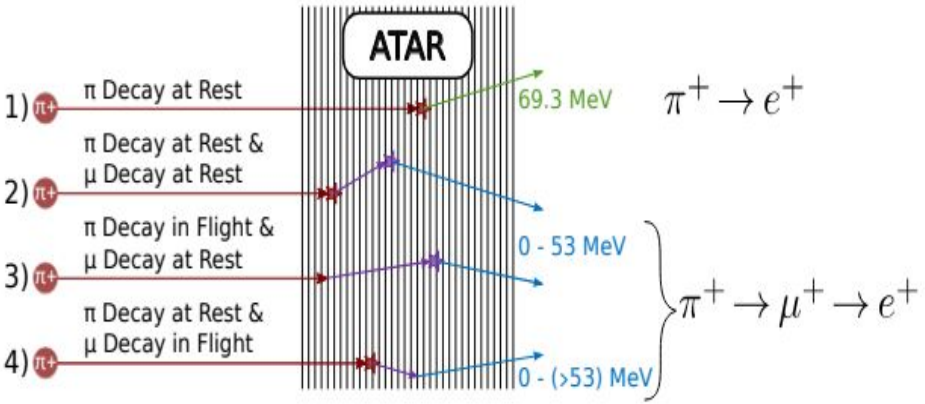
Citation: PSI Ring Cyclotron Proposal R-22-01.1 PIONEER: Studies of Rare Pion Decays, PIONEER collab, (pg. 19 arxiv: 2203.01981)

j.carlton@uky.edu

More ATAR details

- Pion and muon decays deposit energy into ATAR
- Allow event types to be distinguished
- Muons decaying in flight can boost positron energy past 53 MeV (big issue!)
    - ATAR can give information to rebuild event, and correctly classify a muon decay

arxiv: 2203.01981

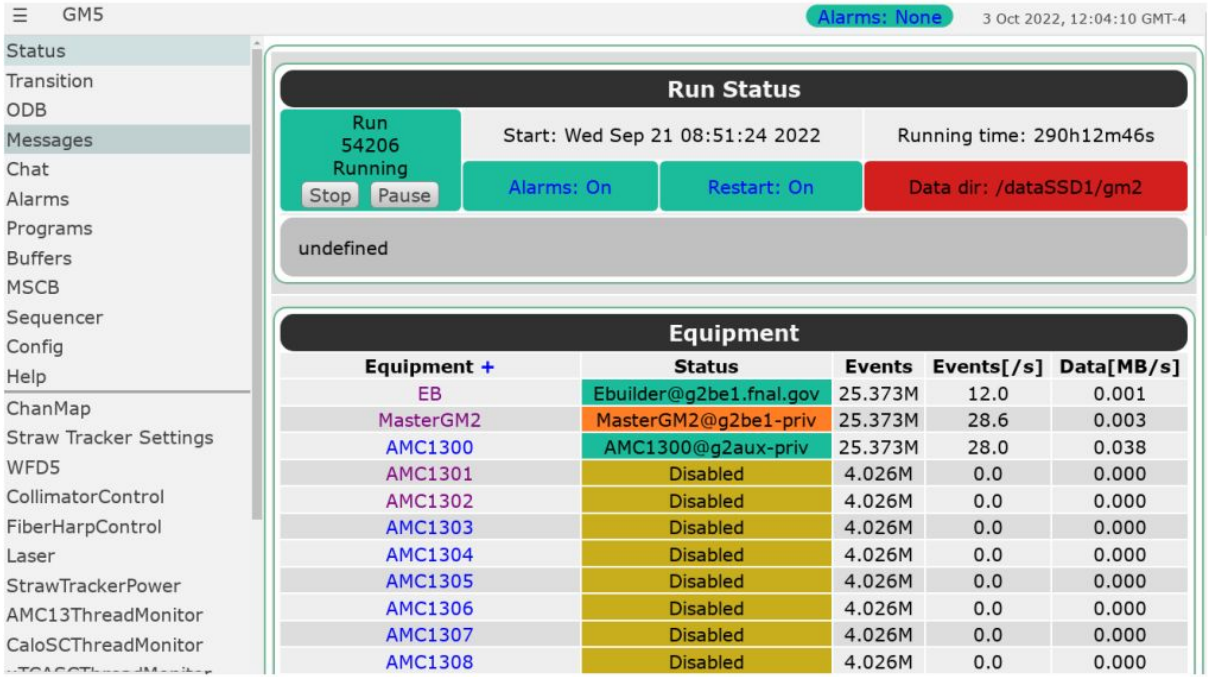


How PIONEER Will Improve the $R_{e/\mu}$ Measurement

- 5D space-time-energy active pion stopping target (ATAR)
  - Reduce e^+ energy tail, identify beam pileup, identify $\pi \rightarrow \mu\nu_\mu$ decays
- Large acceptance, deep radiation length calorimeter
  - LXE or LYSO for high resolution, fast response, small tail
- Fast electronics, high-speed acquisition
  - Giga sample/second digitizers, new gen PCIe readout
- PSI high intensity pion beams
  - 2 mA proton beam, large acceptance beamline

Midas Framework

- C/C++ (mostly) package of modules for
  - run control,
  - expt. configuration
  - data readout
  - event building
  - data storage
  - slow control
  - alarm systems
  - Etc.
- Can link with custom software

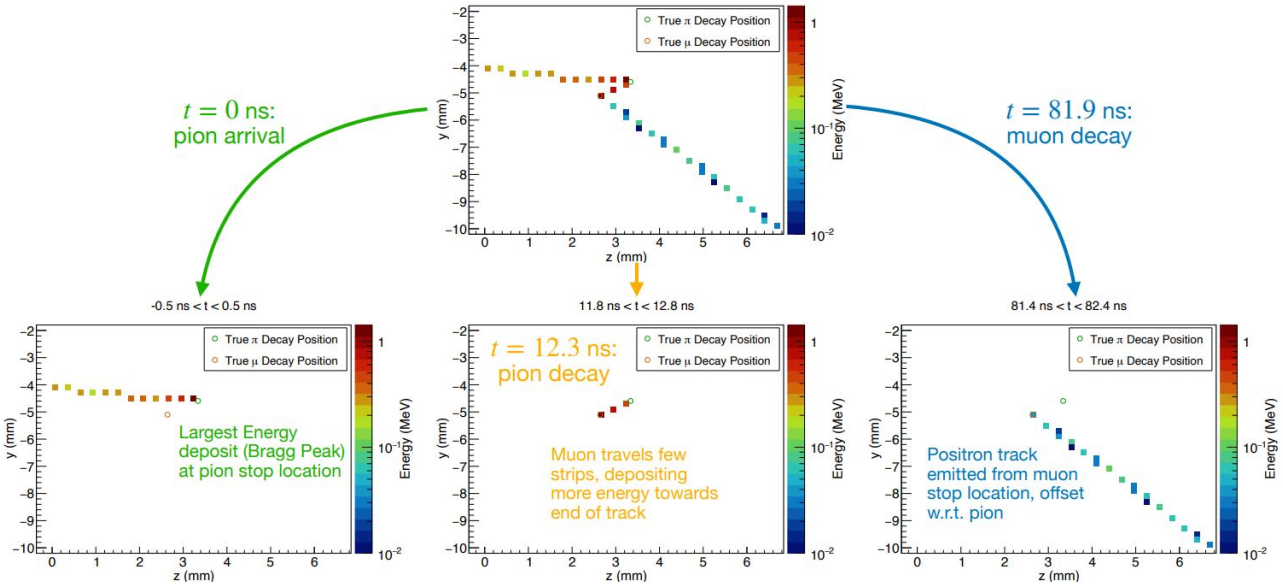


Example g-2 Midas Webpage

Citations: PIONEER Slides from ACTS Workshop, Patrick Schwendimann (Slide 22)

Why Do We Need an ATAR?

Example of pristine π→μ→e event data in ATAR

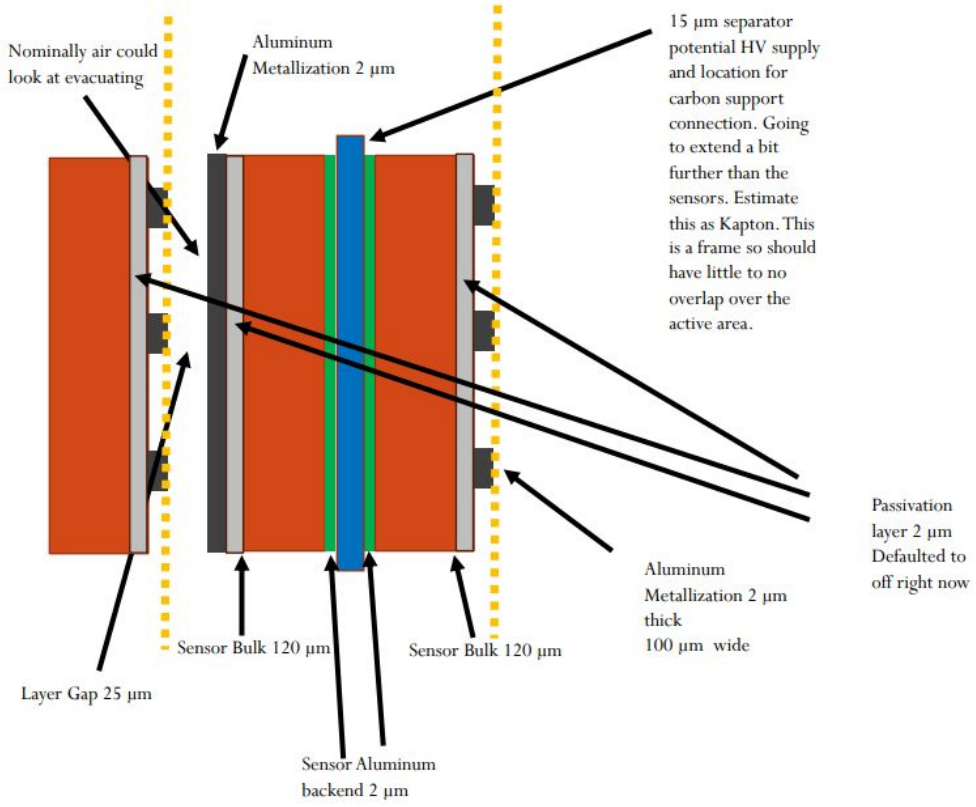


- Energy information alone is insufficient to chracterizethe π→e positron tail at the 10^{-8} level in $R_{e/\mu}$
- ATAR topology and timing are required to identify intermediate muons and correct for this effect

Citations: Update on ATAR region supports, Adam Molnar (Slide 2)

ATAR in Simulation

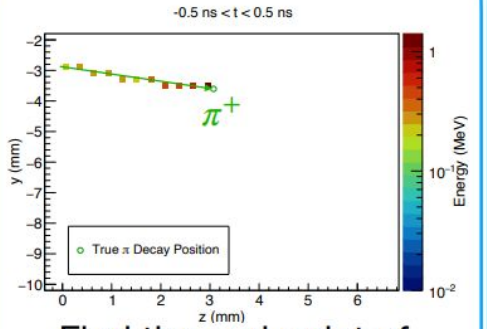
- Geometric descriptions of ATAR in files fed to Geant4 based simulation
  - Geant4 provides simulation of particles interacting with the detector
  - Detector response code turns those interactions into data we'd imagine seeing from the detector
- 24 layers of what's in the yellow dashed lines in the figure
  - Pairs of x y layers in dashed lines, meaning 48 sensor layers



Citations: PIONEER Slides from ACTS Workshop, Patrick Schwendimann (Slide 25)

Why ATAR Events Are Hard to Reconstruct Traditionally

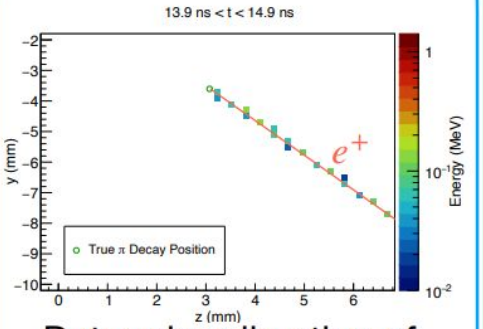
Pion Stop Location



Find the end point of the pion track.

$r_\pi(x_\pi, y_\pi, z_\pi)$

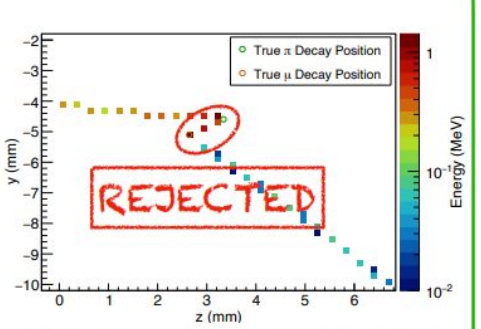
Positron Direction



Determine direction of emitted positron

θ_e, ϕ_e

Tail Analysis

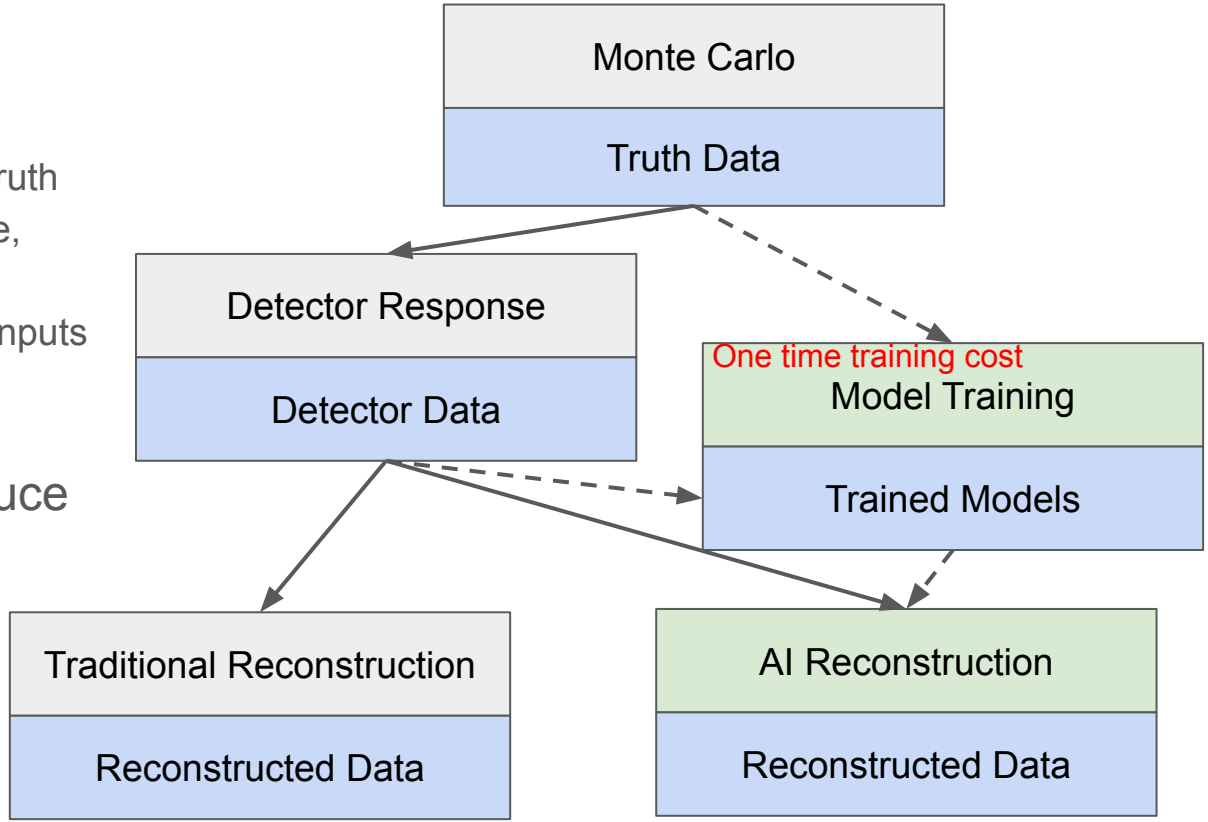


Tag $\pi \rightarrow \mu \rightarrow e$ events. Provide 7 orders of magnitude suppression

- Precise pion stop location and positron direction are essential for rebuilding events in our detector geometry
- Must be very efficient at rejecting π→e events that are actually π→μ→e
  - π→μ→e decays are ~10^4 times more frequent than π→e, and $R_{e/\mu}$ must be measured to ~10^{-4} relative precision, the probability for a π→μ→e event to be misidentified as π→e must be suppressed to the ~10^{-8} level
  - Timing and energy information help us so we need ~10^{-7} suppression from tagging

Why Reconstruction Can be an AI task

- Simulation provides all needed information
  - Geant4 simulation gives truth targets (ex. Positron angle, true pion stop)
  - Detector response gives inputs (ex. ATAR strip hit 5D information)
- The simulation can produce large quantities of data needed for training



j.carlton@uky.edu

Electronics and Data Rates

j.carlton@uky.edu

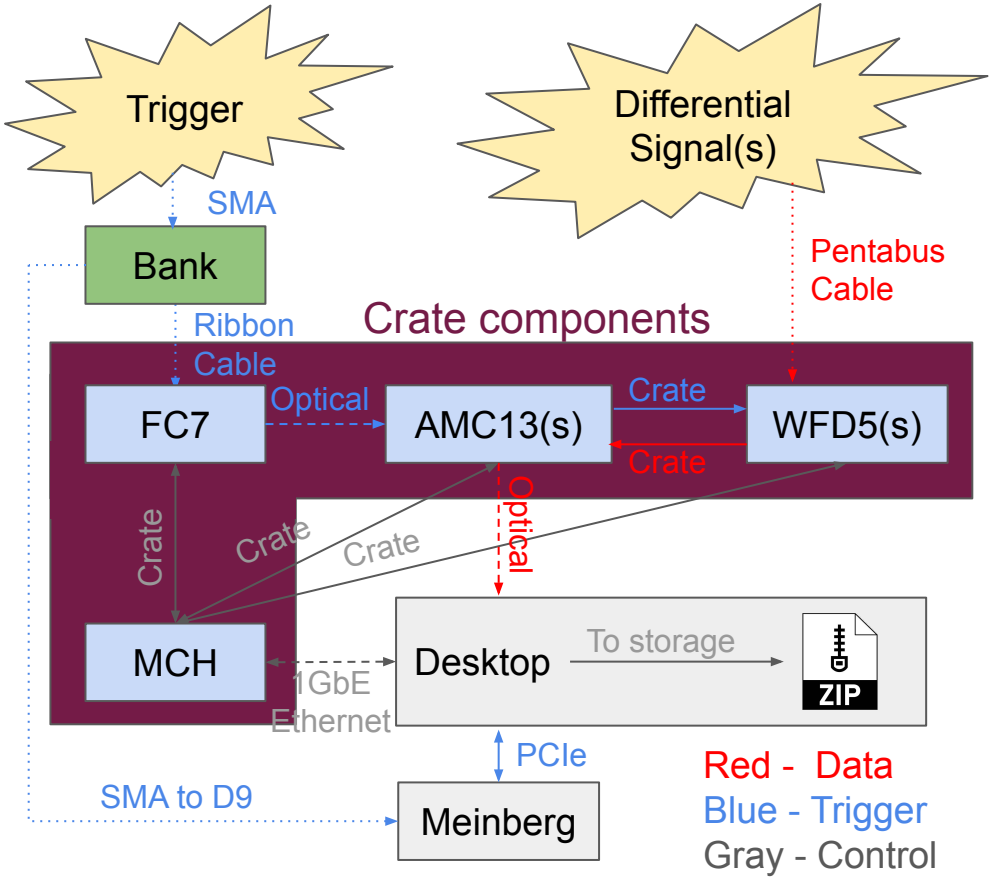
Initialism Cheatsheet

Initialism	Meaning	Example
10GbE	10 Gigabit Ethernet	
FPGA	Field Programmable Gate Array	
FMC	FPGA Mezzanine Card	FC7 SFP Interface
CPU	Central Processing Unit	Intel Core i7-12700K
GPU	Graphics Processing Unit	NVIDIA A5000
µTCA (uTCA)	Micro Telecommunications Computing Architecture	
WFD	Waveform Digitizer	WFD5
FC	Flexible Controller	FC7
AMC	Advanced Mezzanine Card	AMC13 (also FC7 and WFD5)
MCH	MicroTCA Carrier Hub	
DDR	Double Data Rate	DDR3, DDR4 (RAM)
PCIe	Peripheral Component Interconnect Express	PCIe2, PCIe3, ...
TTC	Timing, Trigger, and Control	
UART	Universal Asynchronous Receiver-Transmitter	

j.carlton@uky.edu

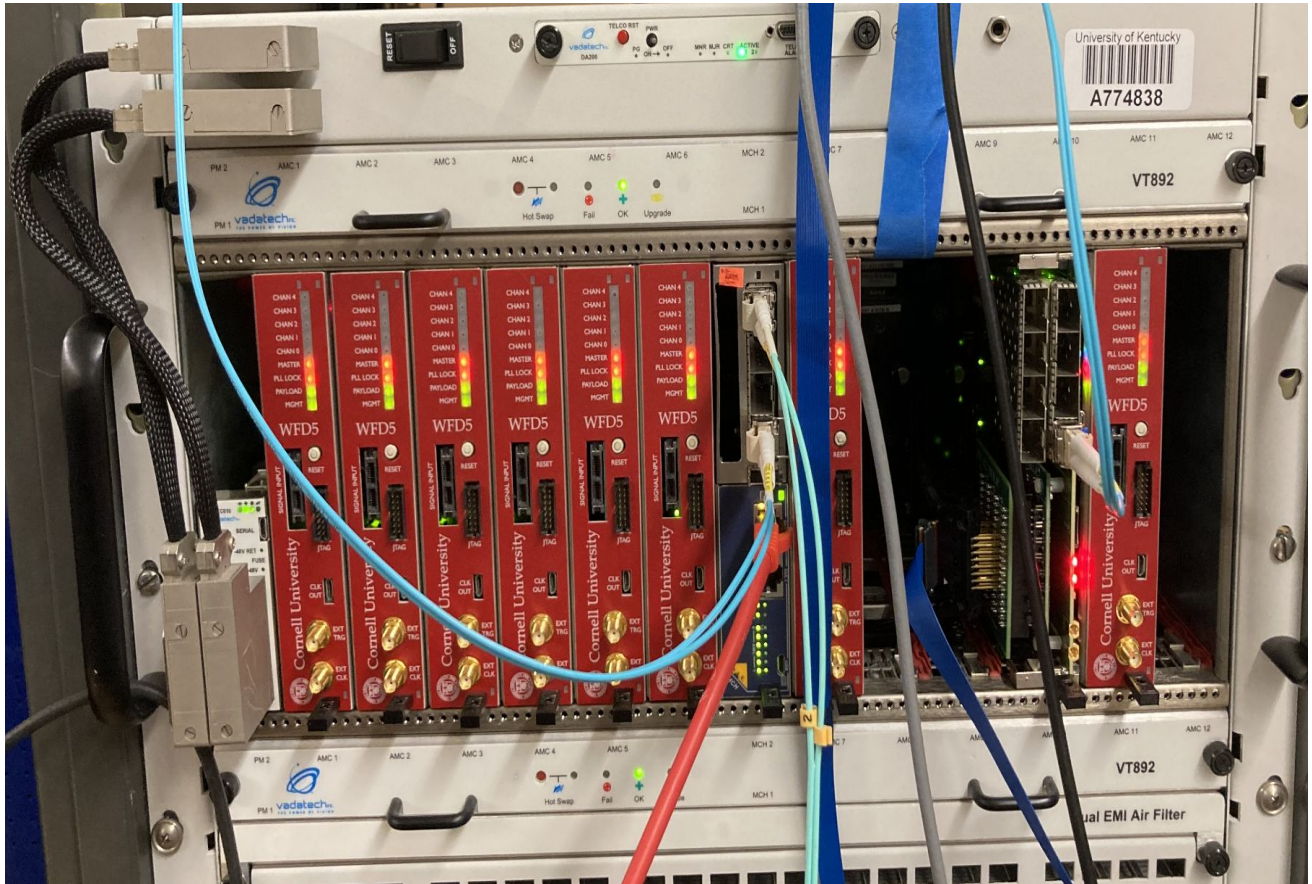
Hardware - Conceptual Diagram

- Differential signal into WFD5 (Waveform Digitizer)
- Trigger signal into FC7 (Flexible Controller)
- AMC13 (Advanced Mezzanine Card) gathers data, sends over 10GbE (10 Gigabit Ethernet) to desktop
- MCH (MicroTCA Carrier Hub) facilitates Desktop↔Crate communication over 1GbE
- Desktop CPU handles event processing
- Meinberg gives trigger timestamp to computer



j.carlton@uky.edu

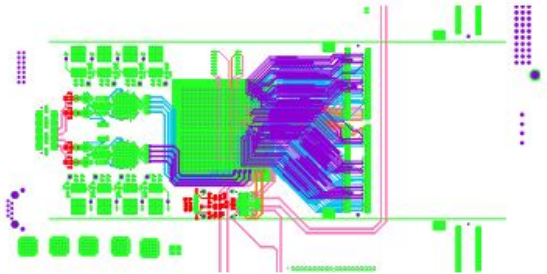
Hardware - Unlabeled Picture

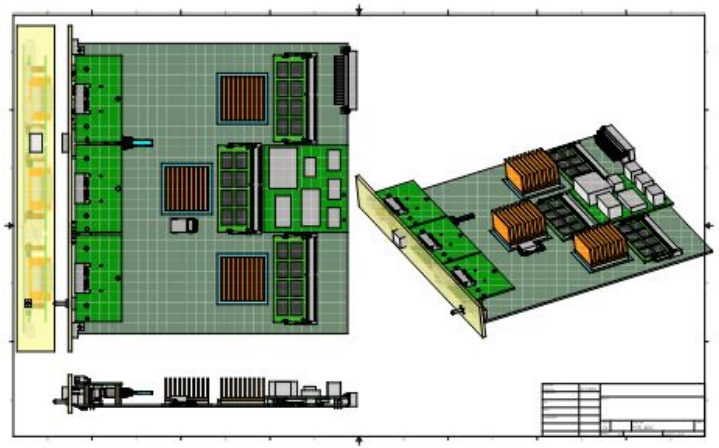


j.carlton@uky.edu

PIONEER DAQ (in a nascent state)

- PIONEER DAQ
  - In nascent development state
  - Design catered to PIONEER full experiment necessities





PIONEER ADC schematic drawings

j.carlton@uky.edu

Data Rates (CALO data rates LXe/LYSO dependant)

arXiv:2203.01981

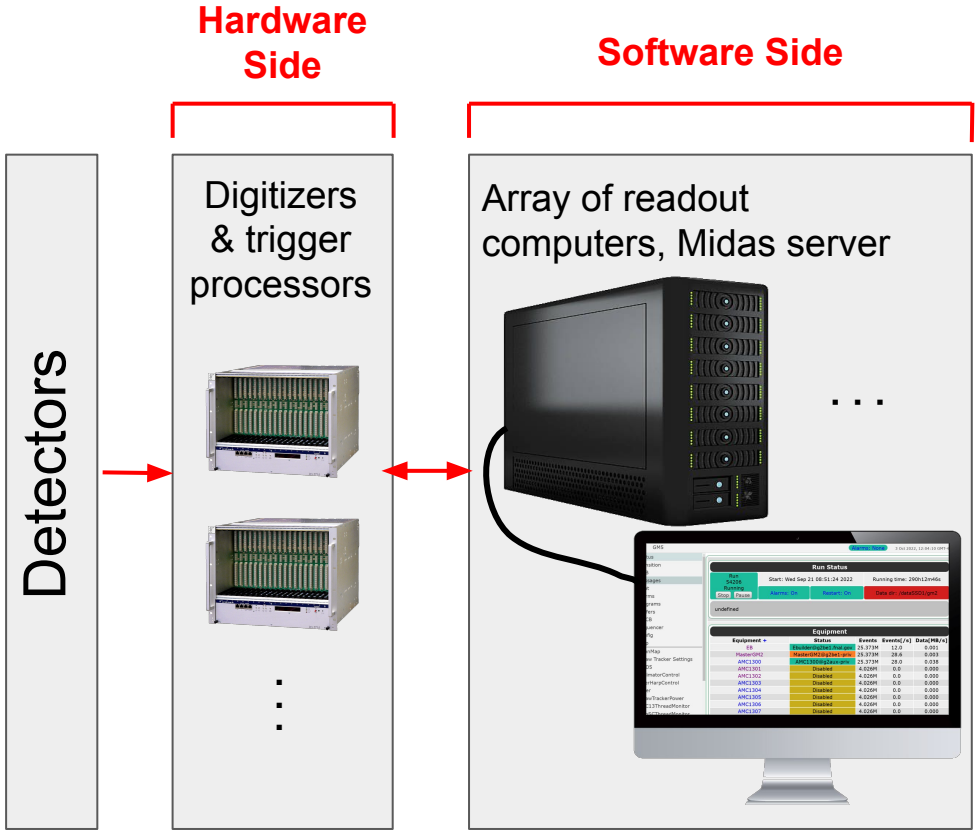
triggers	prescale	range	rate	CALO			ATAR digitizer			ATAR high thres	
		TR(ns)	(kHz)	ΔT(ns)	chan	MB/s	ΔT(ns)	chan	MB/s	chan	MB/s
PI	1000	-300,700	0.3	200	1000	120	30	66	2.4	20	0.012
CaloH	1	-300,700	0.1	200	1000	40	30	66	0.8	20	0.004
TRACK	50	-300,700	3.4	200	1000	1360	30	66	27	20	0.014
PROMPT	1	2,32	5	200	1000	2000	30	66	40	20	0.2

- PIONEER DAQ expects data rate of **~3.5GB/s**
- Considering running time, this is **~35,000 TB/year**
- How do we compress this in real time?
  - Fit data, store fit parameters
  - Compress and store residuals, throw some out
  - Graphics Processing Units (GPUs) used for this operation

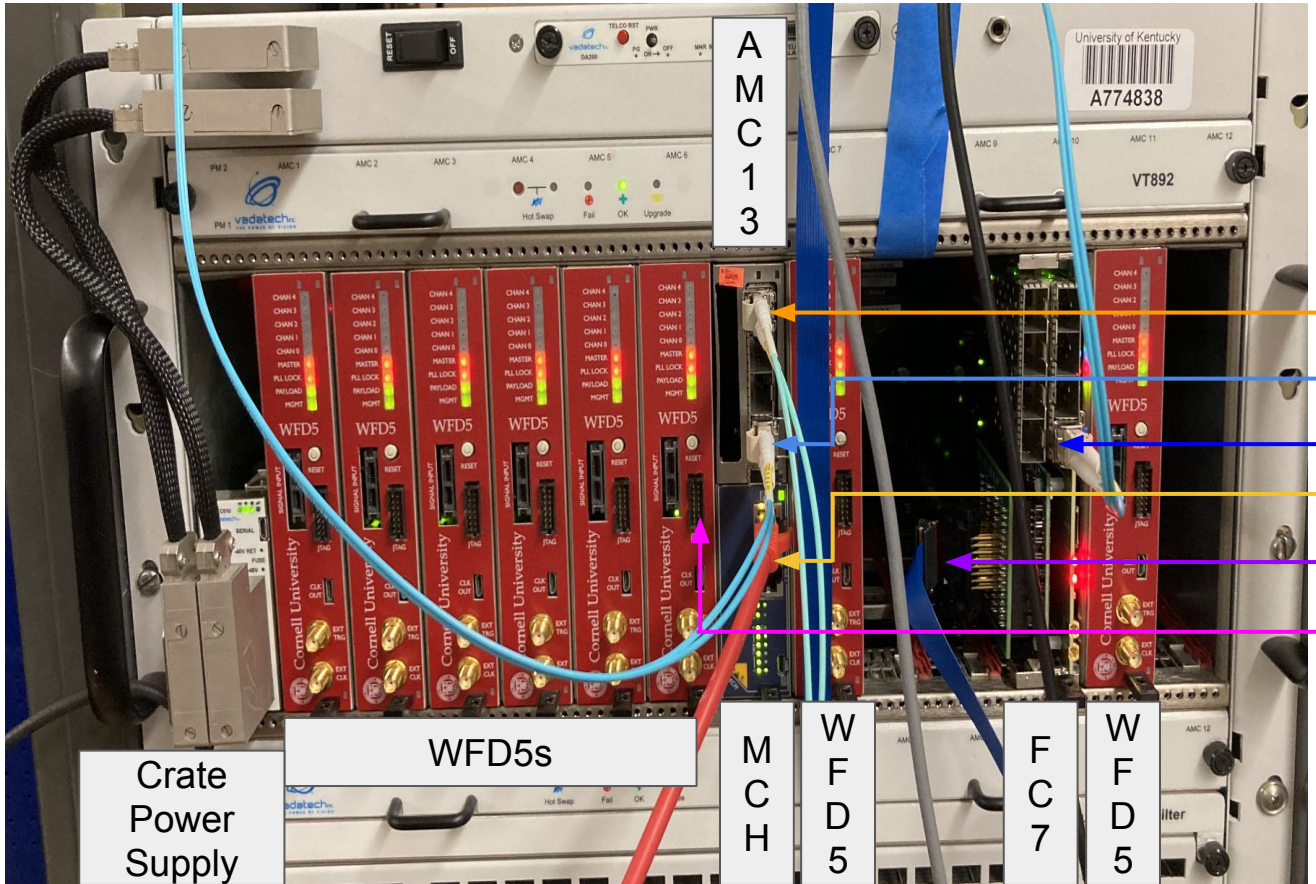
DAQ Software Development

Calorimeter Teststand DAQ

- The test stand DAQ is used throughout the PIONEER collaboration
  - Main uses are for calorimeter development
    - LXe
    - LYSO
- Built on top of g-2 DAQ hardware and software



Hardware - Labeled Crate



Note: AMC13 and MCH are half slot modules

10GbE out (data) AMC13→desktop

Trigger in AMC13

Trigger out FC7

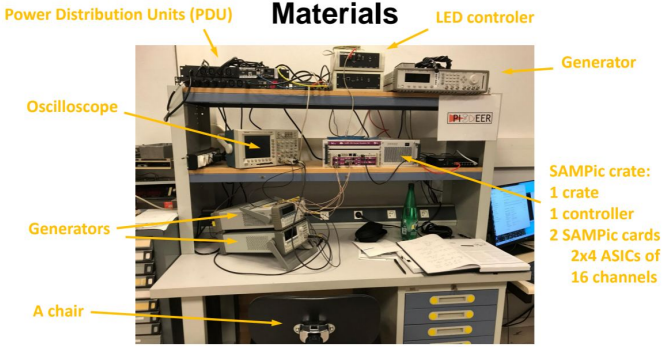
1GbE MCH in/out (comm.)

FC7 Trigger in

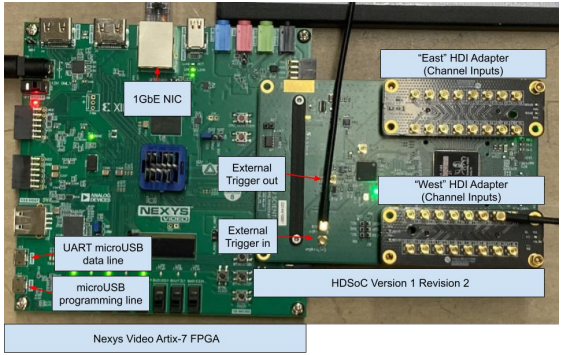
WFD5 5-channel, differential signal in (no connection in this picture)

Readout Systems for ATAR DAQ Candidates

- SAMPic
  - 1 GbE based readout system
  - MIDAS frontend handles configuration and readout, built on existing SAMPic-256ch C library
  - 256 channel digitizer crate teststand in LPNHE, Paris
- HDSoC
  - 1 GbE based readout system
  - MIDAS frontend handles configuration and readout built on existing naludaq python library
  - 32 channel HDSoC digitizer board in UKy
- Comparative studies of rate, triggering, and deadtime performance are currently underway



SAMPic teststand with remote access capabilities

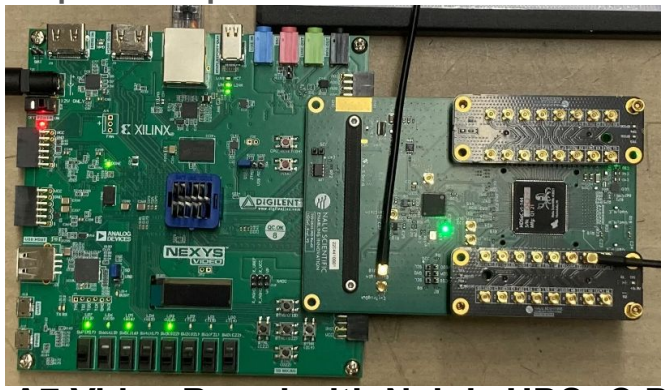


Nalu Scientific's HDSoC FMC attached to a Nexys A7 video card

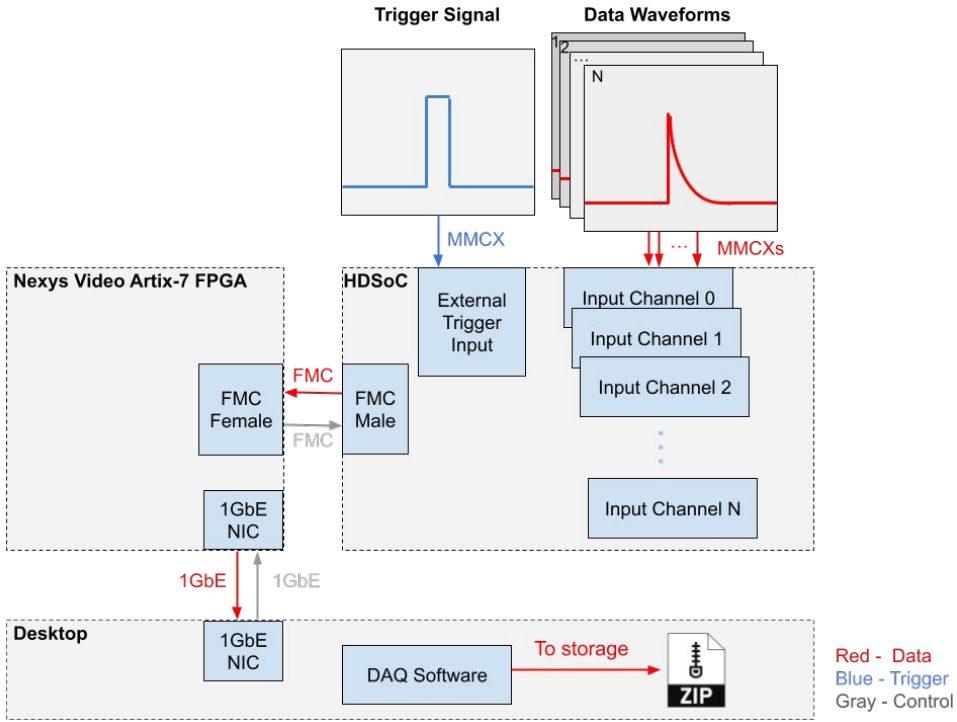
HDSoC DAQ - Hardware

- DAQ is functional and integrated into MIDAS
- Can digitize data rates up to 55 MB/s, event rates up to 30 kHz*

*For specific parameters



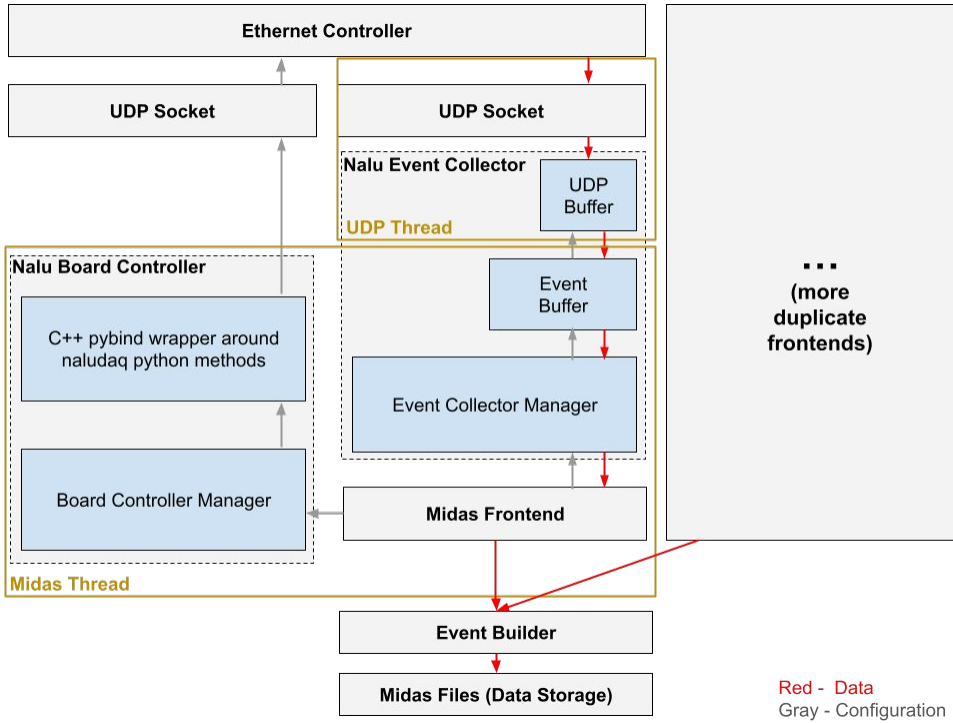
Nexys A7 Video Board with Nalu's HDSoC Digitizer Attached as an FMC Module



Conceptual Hardware Diagram for the HDSoC Readout

HDSoC DAQ - Software

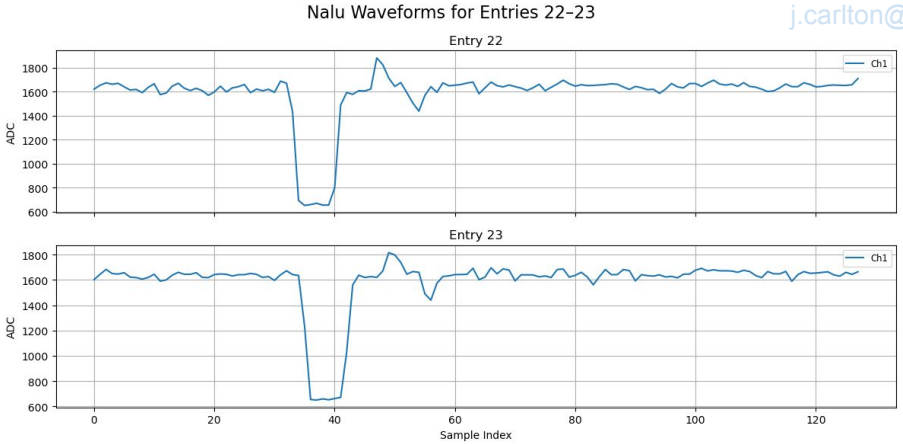
- Wrote a midas frontend that leverages custom libraries created for readout
  - Nalu Board Controller
  - Nalu Event Collector
- Separate branch for rate testing, leveraging custom RP Pico W libraries created for automatic rate testing
  - RP Pico W remote controller
  - RP Pico W board interface



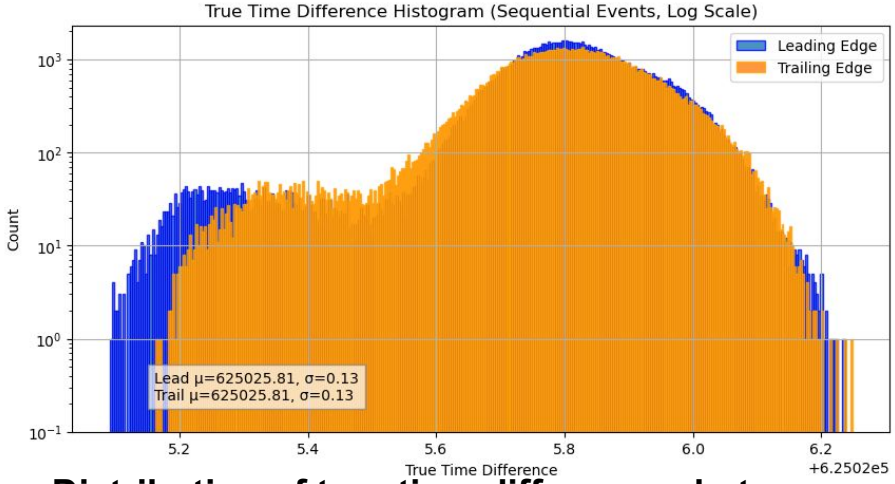
Conceptual Software Diagram for the HDSoC Readout

HDSoC DAQ - Next Steps

- Low priority for the time being
  - Not actively working on development
  - Development focus shifted to to SAMPic system
- Undergraduate Brennan Edwards doing studies on HDSoC digitized data
  - Timing resolution studies
    - Constructs “pseudo time” of leading and trailing edge
    - Then constructs true time differences between events
    - Similar to g-2 analysis



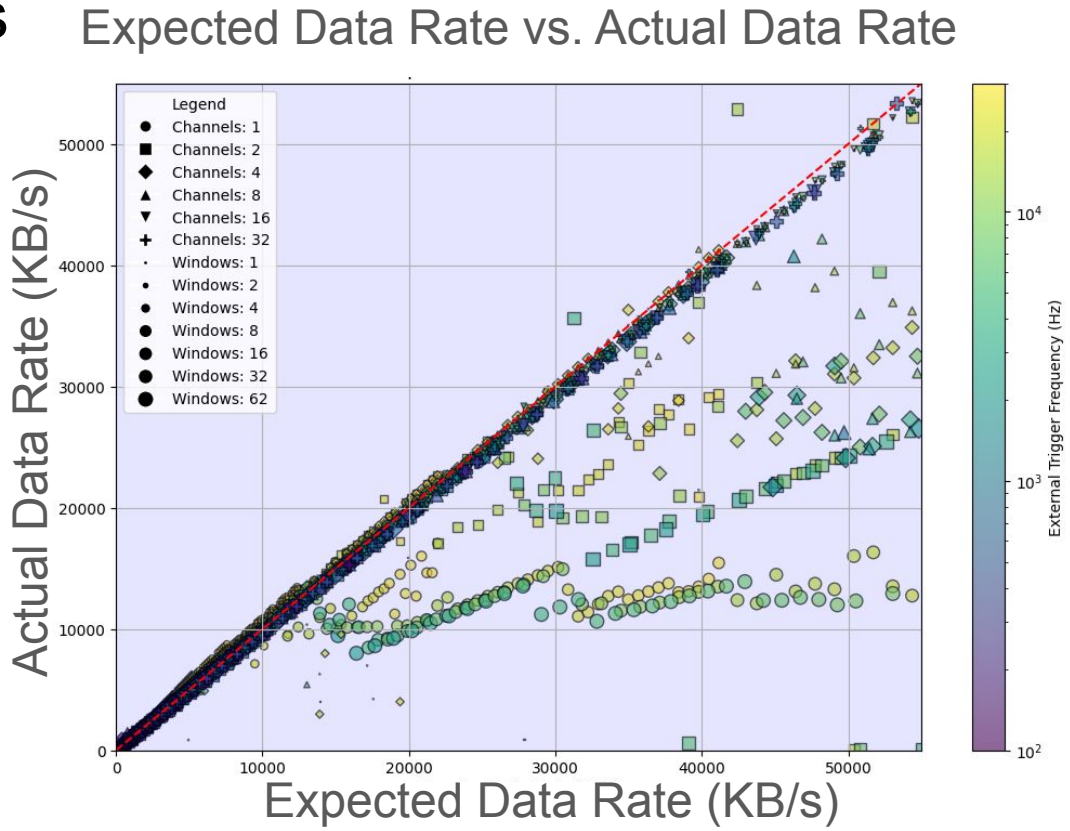
Two consecutive events used to create time difference



Distribution of true time differences between consecutive events [preliminary analysis]

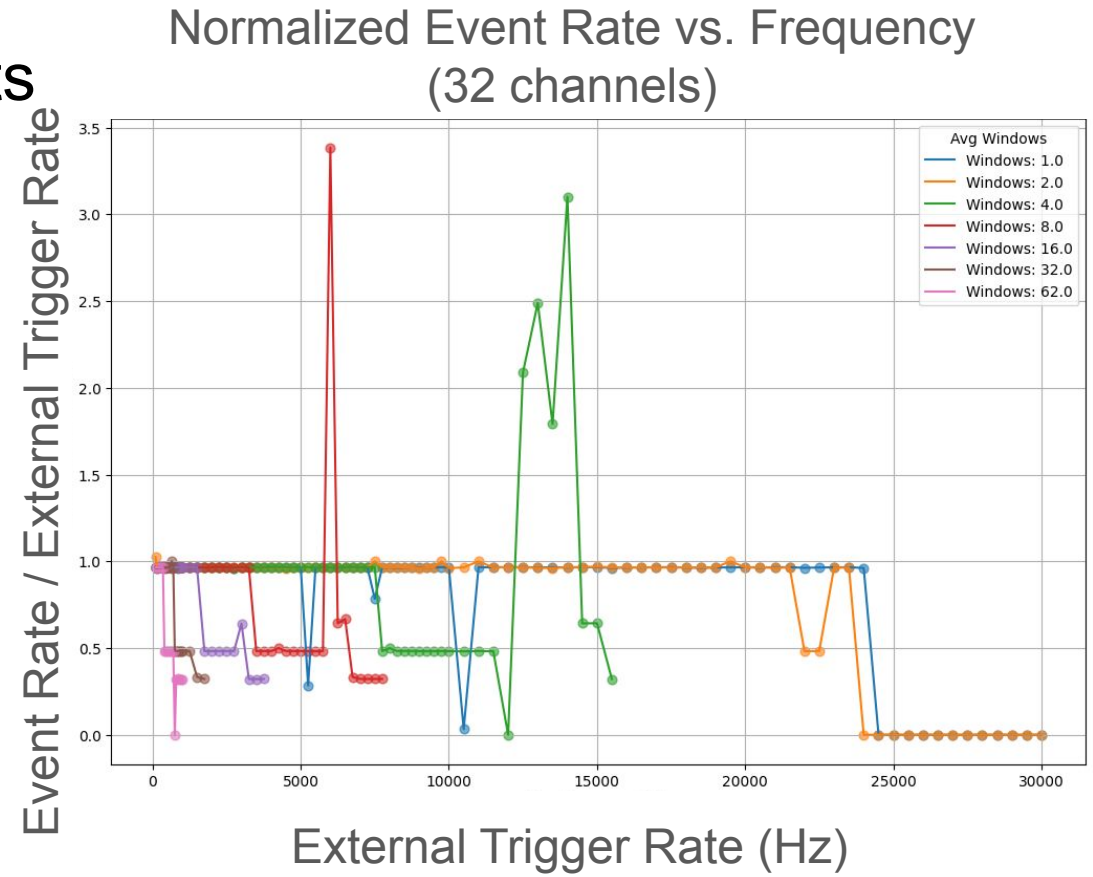
HDSoC DAQ - Rate Tests

- Majority of input parameters → performance as expected
- Outliers where we underperform
  - Expect good performance under 55 MB/s
  - Looking for cause of performance drops



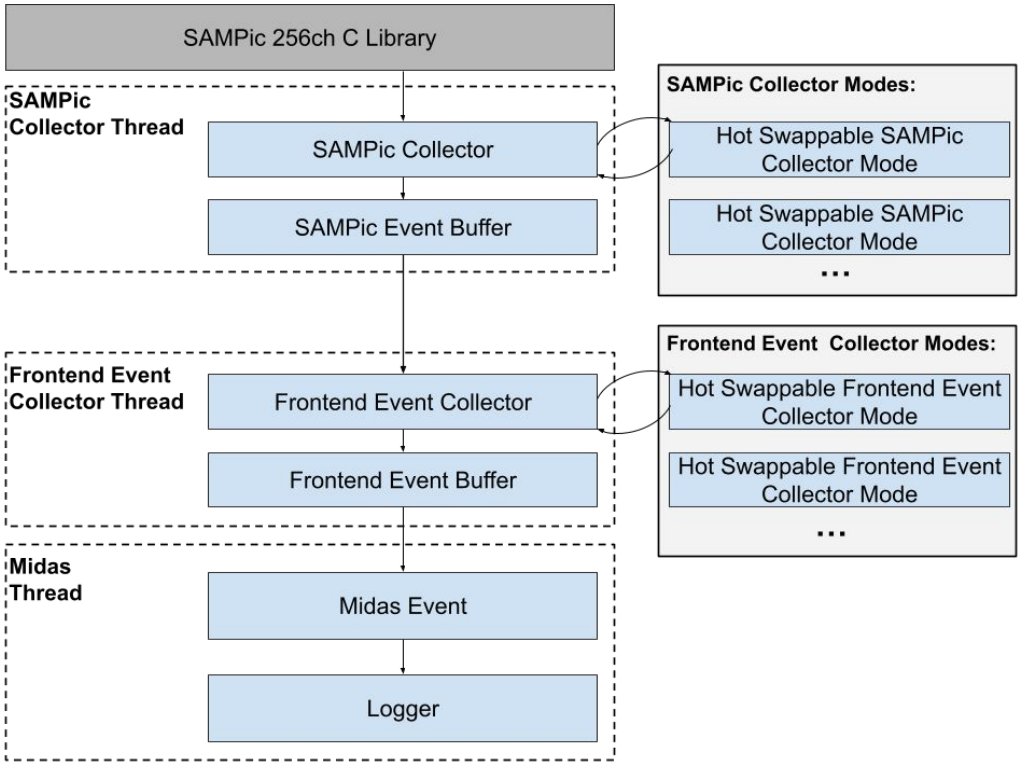
HDSoC DAQ - Rate Tests

- For 32 channels (all active)
- 1 window = 32 12-bit ADC samples
- 1 Gsps
- Can take 32 traces length 64 ns at rates ~20kHz reliably
- Events begin dropping near 55 MB/s threshold



SAMPic DAQ - Status

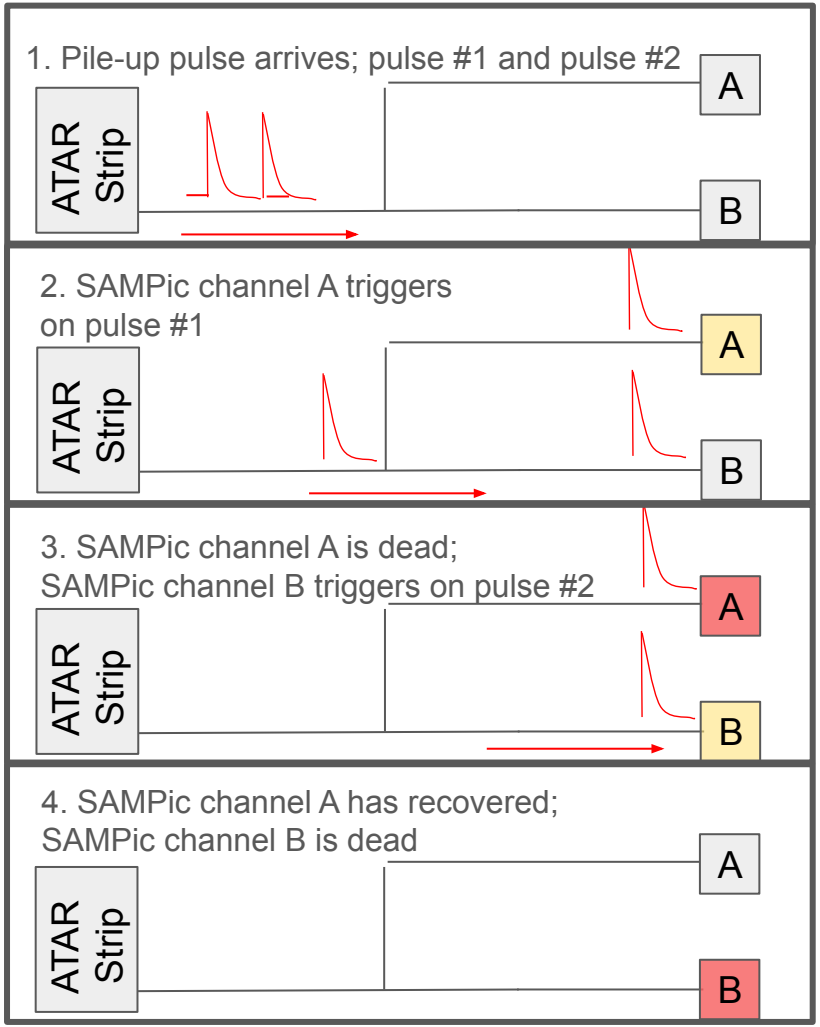
- Working midas integration
  - Converts ODB params → Sampic settings for most settings
  - 1 thread to collect “SAMPic Events”
  - 1 thread to form “SAMPic Events” into “Frontend Events”
    - Frontend events may span multiple sections of “SAMPic Events”
  - 1 thread to handle midas state machine
- Event formation logic is hot swappable for ease of development



Data flow diagram for SAMPic Data

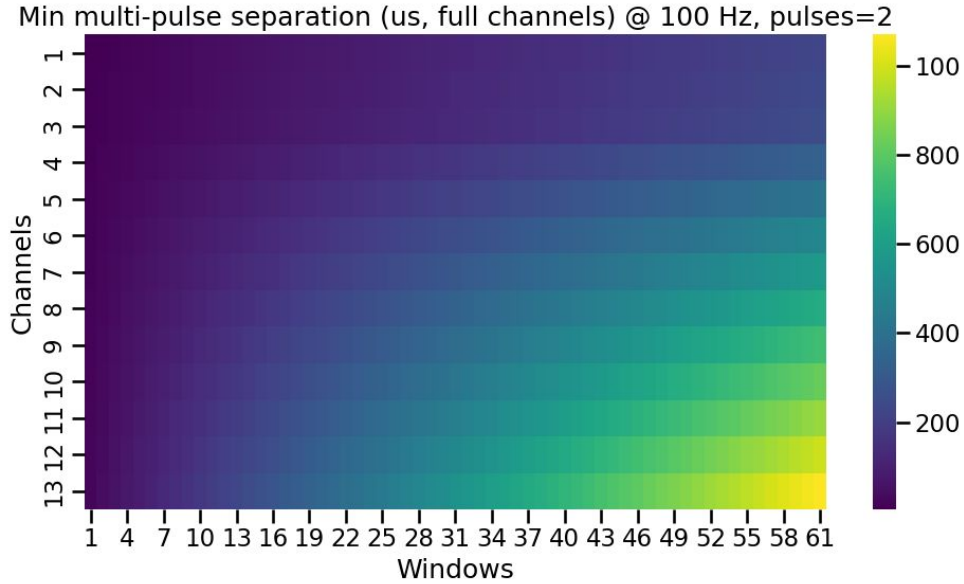
SAMPic DAQ - Ping Pong Mode

- In HDSoC approach, you see all activity on any given channel in the past ~2 μs
- For SAMPic, only short readout (~32ns) then channel is dead (~2 μs)
  - Problem: We become blind to pile-up
  - Solution: Ping pong mode
- Ping ping mode:
  - Copy input across 2 SAMPic channels A and B
  - Channel B can still trigger during Channel A deadtime
- For heavy pile-up we may need ping pong mode for across 4 channels
  - 4800 ATAR channels→ 9600 SAMPic channels (ping pong)
  - 4800 ATAR channels→ 19200 SAMPic channels (double ping pong)

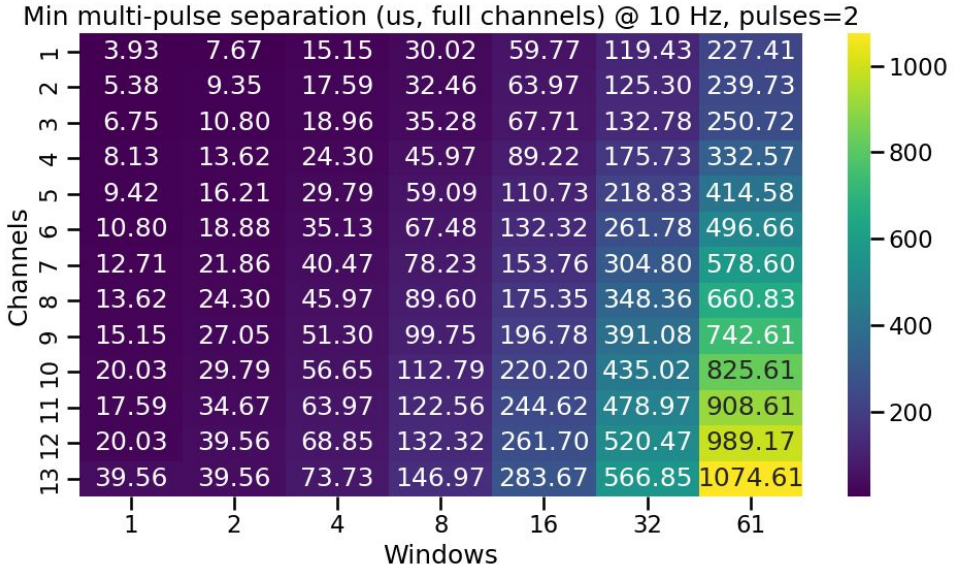


HDSoC Deadtime Scan

Parameter Space (793 combinations):
Channels = [1,2,...,13]
Windows = [1,2,3,...61]



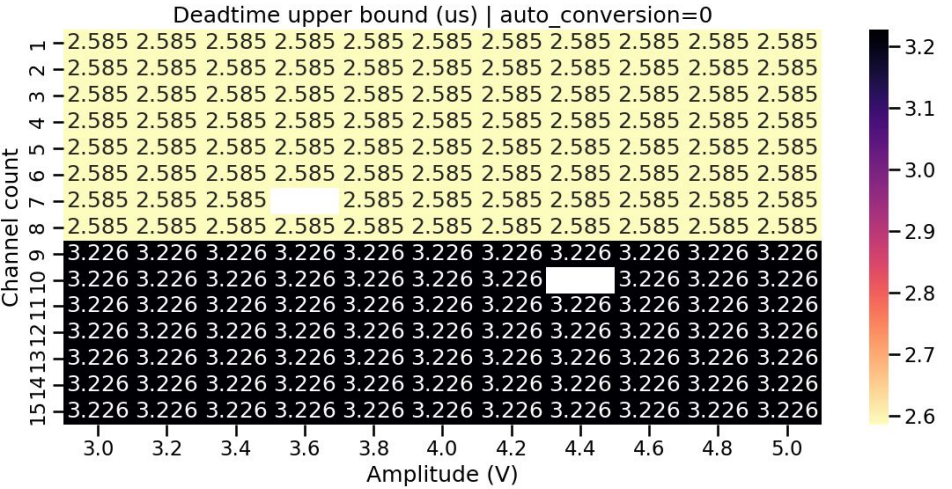
Parameter Space (91 combinations):
Channels = [1,2,...,13]
Windows = [1,2,4,8,16,32,61]

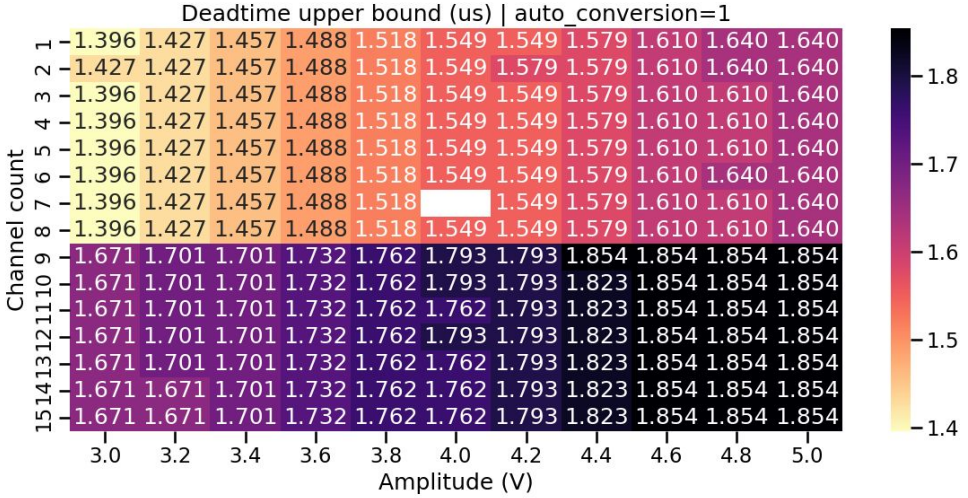


SAMPic Deadtime Scan

- Some runs failed (likely a communication error), the scan was programmed to move on in this case
- There's some scaling between Lecroy input voltage and SAMPic read voltage (ex. 5V lecroy input digitizes as ~0.5V pulse on SAMPic)
- Unknown Lecroy channel specific behavior causes difference in deadtime
  - Channel count 1-8 is just lecroy channel A
  - Channel count 9-15 includes both lecroy channel A and B

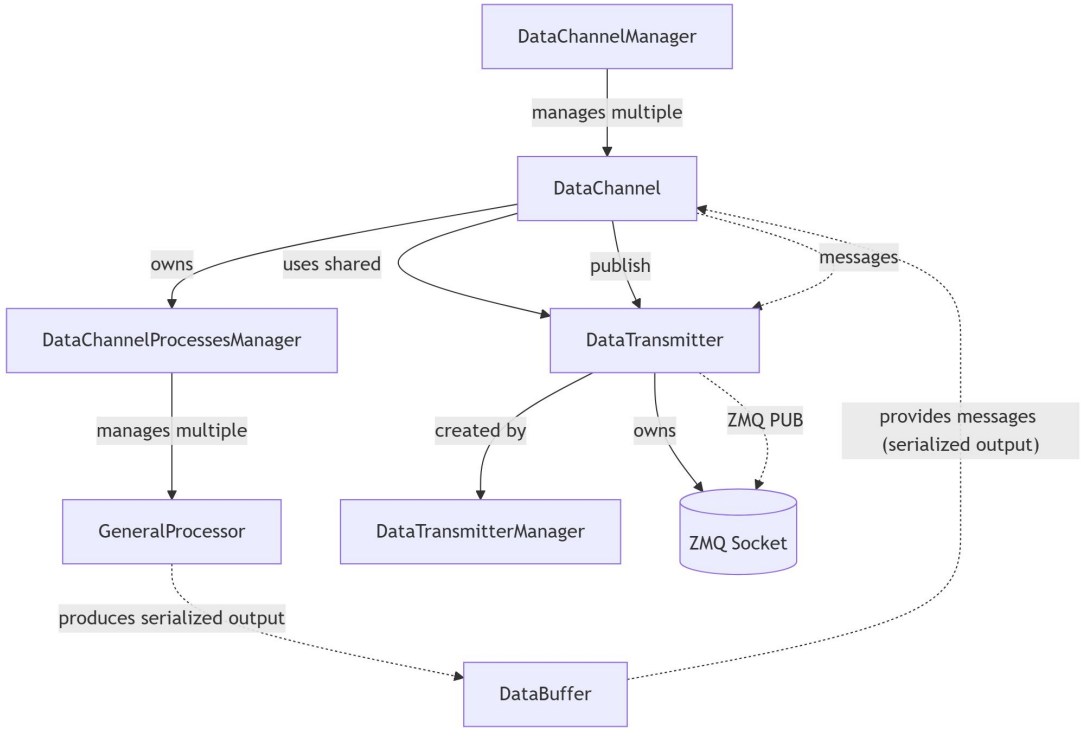
Parameter Space (330 combinations):
Channels = [1,2,...,13,14,15]
Amplitude = [3.0,3.2, … 5.0]
Auto_conversion = [False, True]





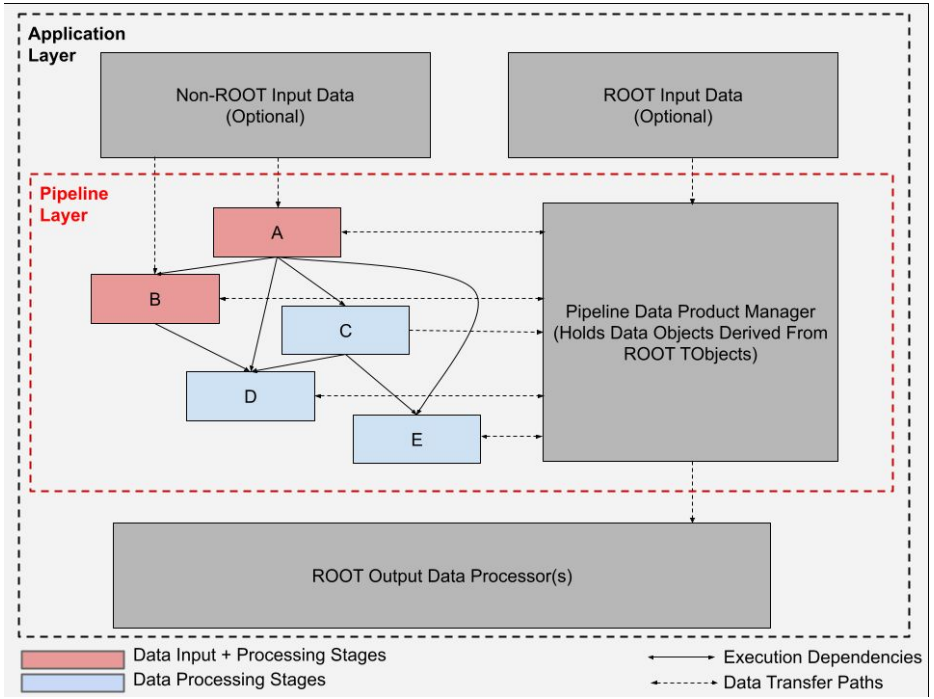
ZeroMQ Publisher Application

- Configurable software to publish data over ZeroMQ
- Main branched configured to use midas
  - Not technically necessary, data source can be anything
  - Uses my midas receiver library
- Uses C++ Package Manager (CPM) to clone (most) dependencies from gitub and build them
  - Less work setting up environments for user



Supplementary Software - Analysis Pipeline Framework

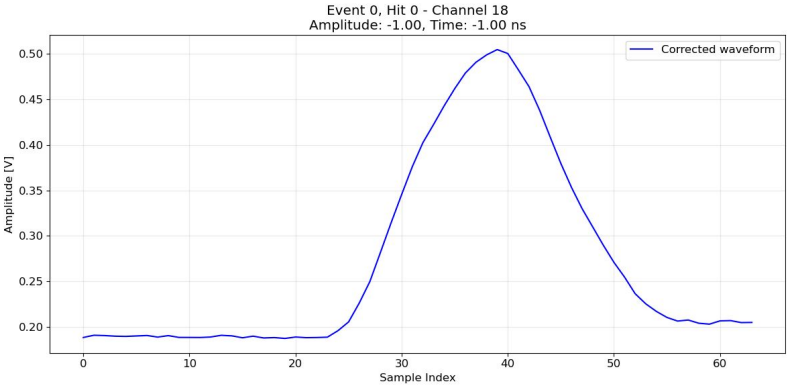
- Lightweight, configurable pipeline framework for ROOT data
  - Inspired by Gaudi, but simpler and faster to deploy
  - Built on ROOT, leveraging:
    - Plugin system for modularity
    - Reflection for flexible configuration
    - Custom data products
- “Backbone” of apps:
  - ZMQ publisher (part of DQM framework)
  - Midas file unpacker app
- Documentation available
  - Wiki
  - Repo



Example data flow diagram for an analysis pipeline

Supplementary Software - Unpacker Application

- Application to convert midas files to ROOT Trees
  - Currently on works for HDSoC and SAMPic MIDAS files
  - Adaptable to event structure changes
- Adding new unpackers simplified
  - Boils down to adding new plugins in the analysis pipeline framework



Example Digitized SAMPic Waveform

Example HDSoC data ROOT tree formed by the application

Simulation Software Development

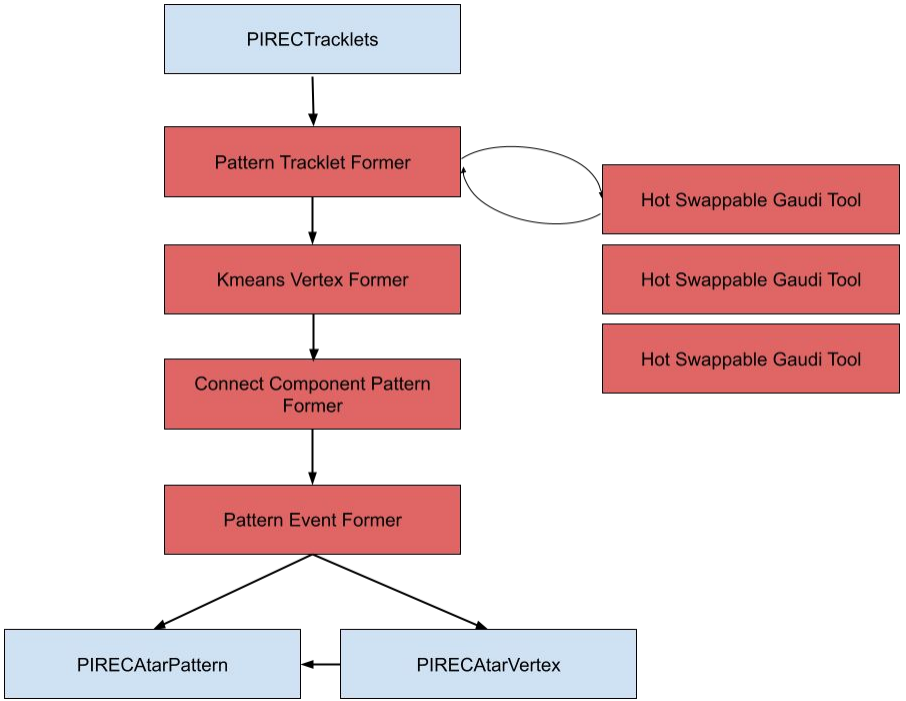
Citations: ATAR Pattern Finding Presentation, Sean Foster (Slides 5-6)
https://indico.psi.ch/event/17600/contributions/59200/attachments/31908/63829/Pattern%20Finding%20-%20CM%20Oct%202025.pdf

Vertex Finding Algorithm

- Use PID to select pion, muon, and positron tracklets
- Start with the pion tracklets
  - For each pion tracklet, search for an overlap with any of the muon and positron tracklets (candidate tracklets); search is in each orientation (XZ and YZ) separately
  - For each candidate, an overlap type is determined using the tracklet's endpoints
  - From the candidates, the "best" (closest) overlap is selected
- Overlap types
  - Distance between endpoints is below a threshold
  - Extrapolated distance to an endpoint is below a threshold
  - Missing hits in this orientation, so no overlap possible
  - No overlap
  - … ●
- A vertex is formed if a valid overlap is found in each orientation
- Move onto the muon tracklet
  - For each muon tracklet, search for any overlaps with positron tracklets
  - Overlaps (and vertices) are determined in the same way as for the pion tracklet
- Form single-endpoint vertices for all remaining endpoints
  - Algorithm expects all endpoints to be assigned a vertex
  - All remaining endpoints are made into their own vertices

Pattern Finder Implementation into Simulation Framework

- Pattern finder leverages Gaudi Tools to organize standard utilities
  - Allows for easily hotswapping “stages” to benchmark different approaches against each other
- Pattern finding playground for development in python



Another Pattern Finding Approach

- Define “expected” vertex types
  - Ex: pi->e, pi->mu
- Each tracklet gets “scored” based on what vertex it belongs to
  - Ex. pi tracklet looks for e or mu tracklet to connect to, whichever is closer is scored higher
- Biased
  - Only finds vertices it’s “told” to look for
- Scoring comparisons need to be tuned
  - Inherently subjective

```python
# vertex_types.py

from algorithms.vertex.vertex_types.vertex_type import VertexType
from algorithms.vertex.vertex_types.scoring.distance_scorer import DistanceScorer


class PionMuonVertex(VertexType):
    def __init__(self):
        super().__init__(
            id="pi+_to_mu+",
            input_particles={211},  # pi+
            output_particles={-13},  # mu+
            scorer=DistanceScorer()
        )


class PionPositronVertex(VertexType):
    def __init__(self):
        super().__init__(
            id="pi+_to_e+",
            input_particles={211},  # pi+
            output_particles={-11},  # e+
            scorer=DistanceScorer()
        )


class MuonPositronVertex(VertexType):
    def __init__(self):
        super().__init__(
            id="mu+_to_e+",
            input_particles={-13},  # mu+
            output_particles={-11},  # e+
            scorer=DistanceScorer()
        )
```

Example list of vertex types

Citations: Status of the full simulation chain and R_emu measurement, Patrick Schwendimann
https://indico.psi.ch/event/17600/contributions/59206/attachments/31950/63841/Schwendimann_FrameworkStatus.pdf

Failure Modes for full reconstruction

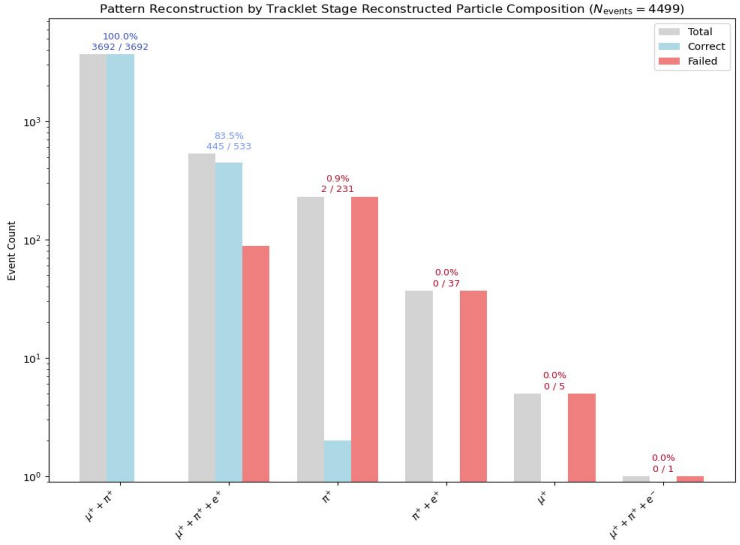
Summary of Failure Modes

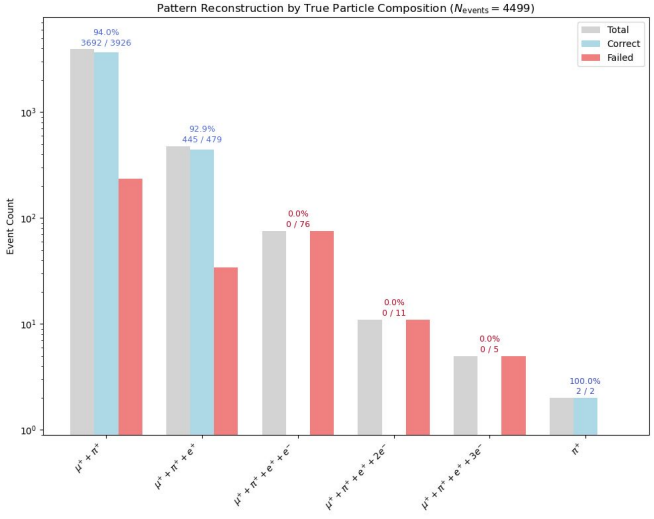
Bias introduced

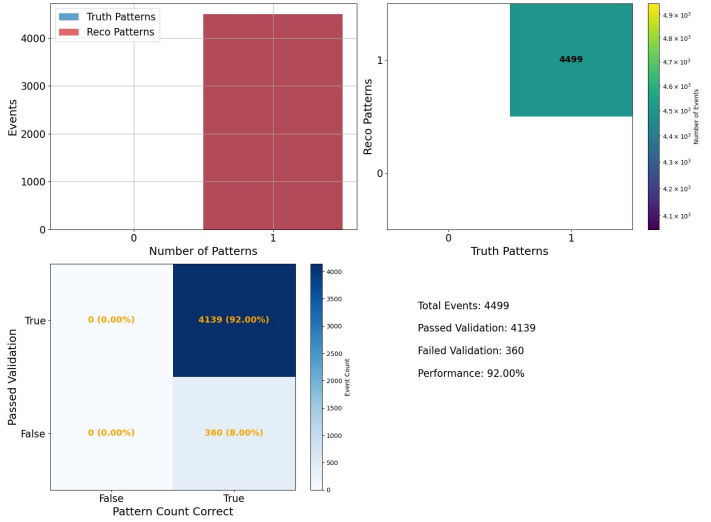
Mode	Pienu	Pimunu	Deviation	
What could go wrong?	Rate %	Rate %	%	
Positron identified as Muon	0.056(3)	0.039(6)	0.018(7)	Particle Identification
Other cases where no positron was identified	0.004(1)	0.006(2)	-0.002(2)	
Pion identified as Positron	0.286(8)	0.285(18)	0.001(2)	
Ambiguous Track, picked wrong solution	1.519(20)	1.102(36)	0.416(41)	Matching Tracks to Patterns
Matching XZ and YZ info to 3D start failed	0.281(8)	0.278(18)	0.003(20)	
Large number of Endpoints, picked the wrong one as start	2.106(24)	2.076(49)	0.030(55)	
Non-Positron hits confuse endpoint finding (e.g. Bhabha)	0.577(12)	0.575(26)	0.002(29)	
Start point is 1mm or more away from where it should be	1.025(16)	0.916(33)	0.109(33)	
Others (Mostly issues related to muon reconstruction)	0.015(2)	0.262(17)	-0.248(17)	
Total	**5.869(40)**	**5.540(81)**	**0.328(91)**	

Example Reconstruction Accuracy (New Approach)

- Performs better
  - Reverse engineered, so it “cheats”
  - Inaccuracy solely due to Tracklet Finding inaccuracy

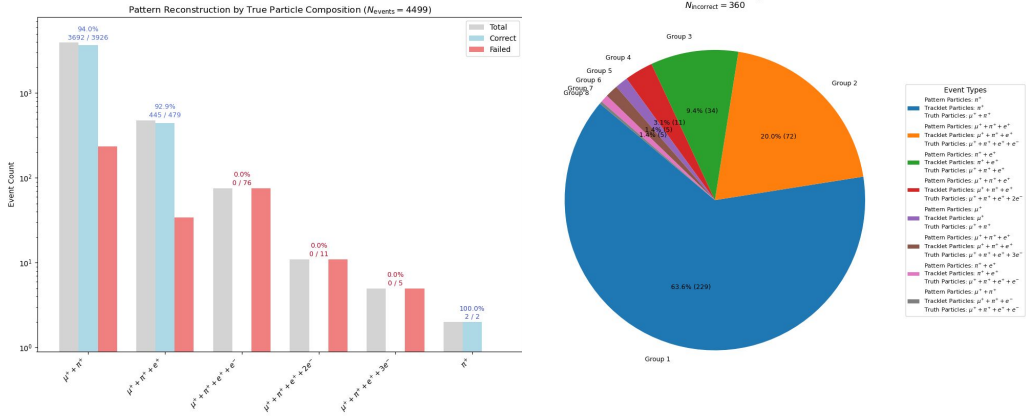




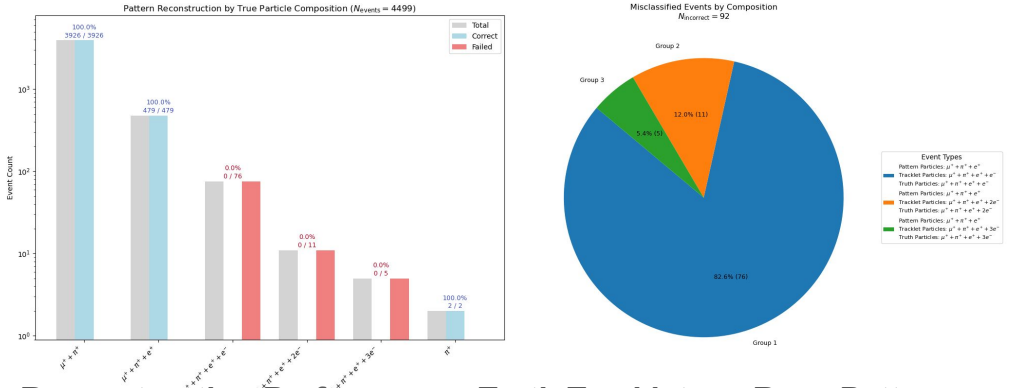


Tracklet finding biggest source of inaccuracy

- Slightly modified algorithm
  - See “new approach” slide
- All inaccuracies are in the tracklet finding stage
  - **Test:** Run pattern reconstruction using reconstructed tracklets vs. truth tracklets
  - **Result:** Pattern reconstruction has perfect performance
- No pileup in this study



Reconstruction Performance: Reco Tracklets -> Reco Patterns



Reconstruction Performance: Truth Tracklets -> Reco Patterns

What is a Neural Network?

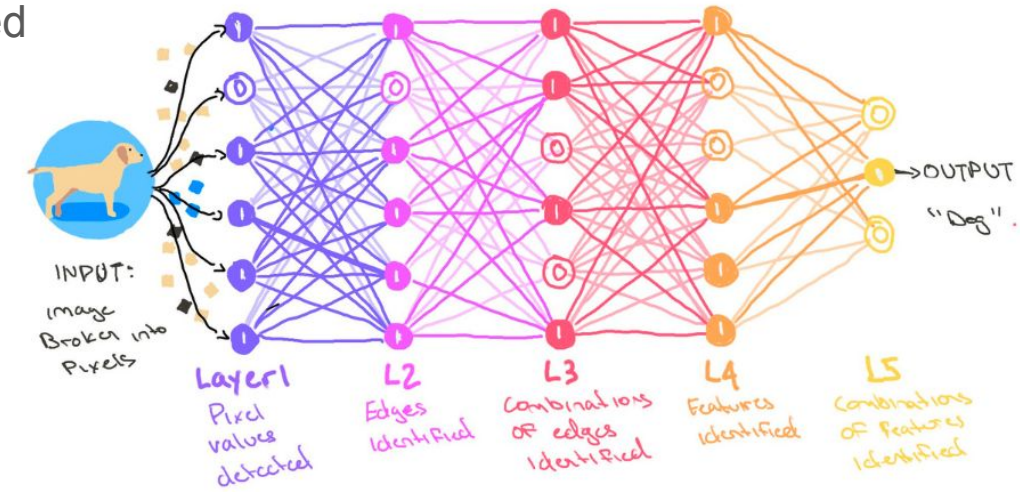
- A neural network is a parameterized function that approximates non-linear mappings through stacked linear transformations and nonlinear activations.
- A composition of function layers

$$f(\cdot;\theta) = f_L \circ f_{L-1} \circ \cdots \circ f_1$$

- Parameters (weights and biases) varied to optimize arbitrary loss function i.e. “learned”

$$f_\ell(\mathbf{x};\theta_\ell) = \sigma_\ell(\mathbf{W}_\ell\mathbf{x} + \mathbf{b}_\ell)$$

$$\text{where } \theta_\ell = \{\mathbf{W}_\ell, \mathbf{b}_\ell\}$$

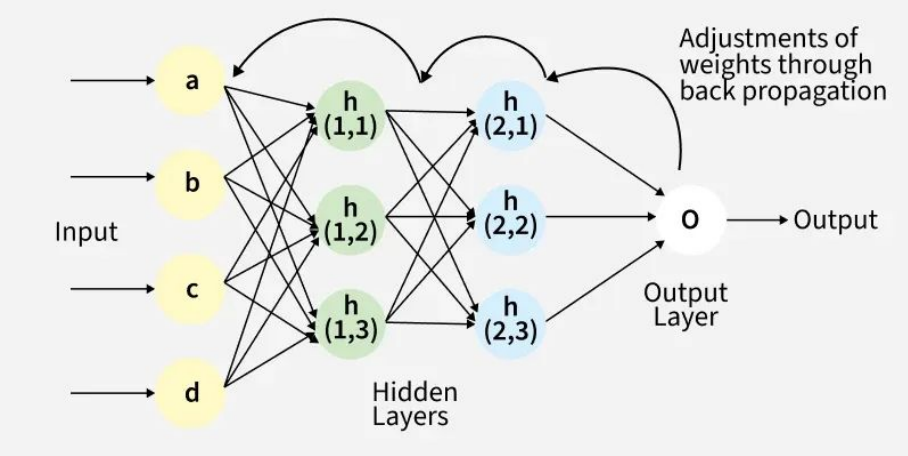


Example annotated multi layered neural network used to classify images of dogs

This is a specific type of neural network called a multi layer perceptron (MLP), the simplest neural network

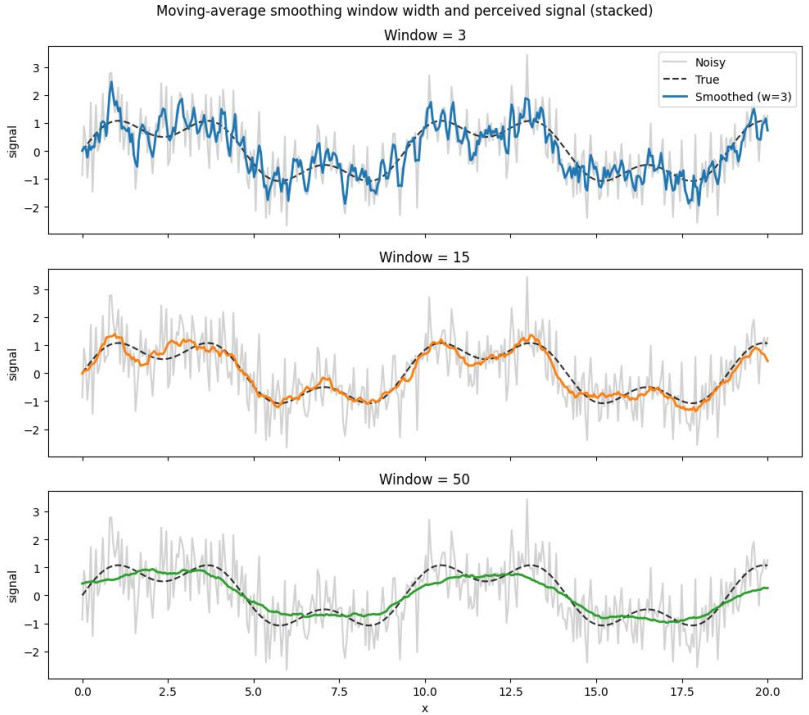
Backpropagation (Overview)

- Much more detail in this article, and wikipedia
- **Backpropagation computes gradients of the loss with respect to all model parameters by applying the chain rule backward through the network**
- Compute the gradient of the loss in the space of weights
    - Optimal weight updates for the next iteration



What are Hyperparameters

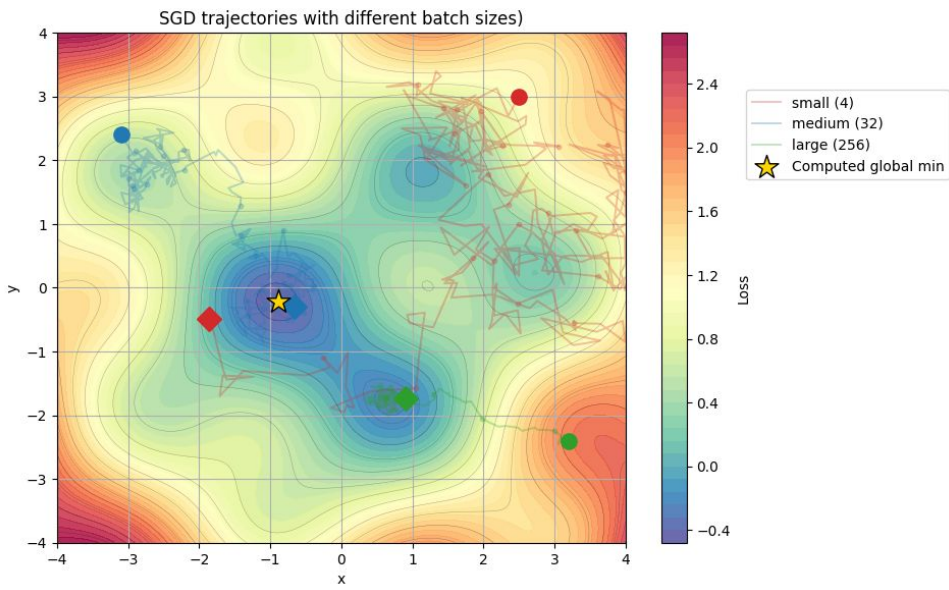
- **“A Hyperparameter is a parameter that can be set in order to define any configurable part of a model's learning process”**
  - These are parameters that are not learned by the model
  - Chosen before the model is trained
- Examples of what they control
  - Optimization dynamics (learning rate, batch size, optimizer)
  - Model capacity (hidden size, depth, heads)
  - Regularization (dropout, weight decay)

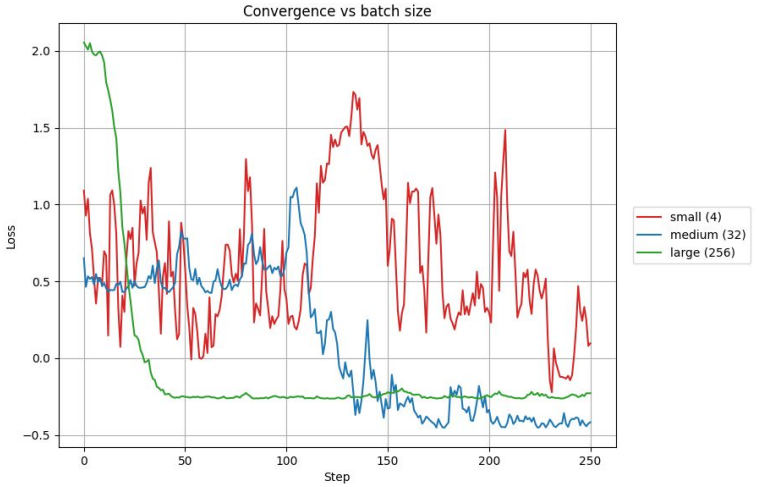


Example: Moving average window size is a hyperparameter. It can be over, or under tuned.

Hyperparameter Example (Batch Size)

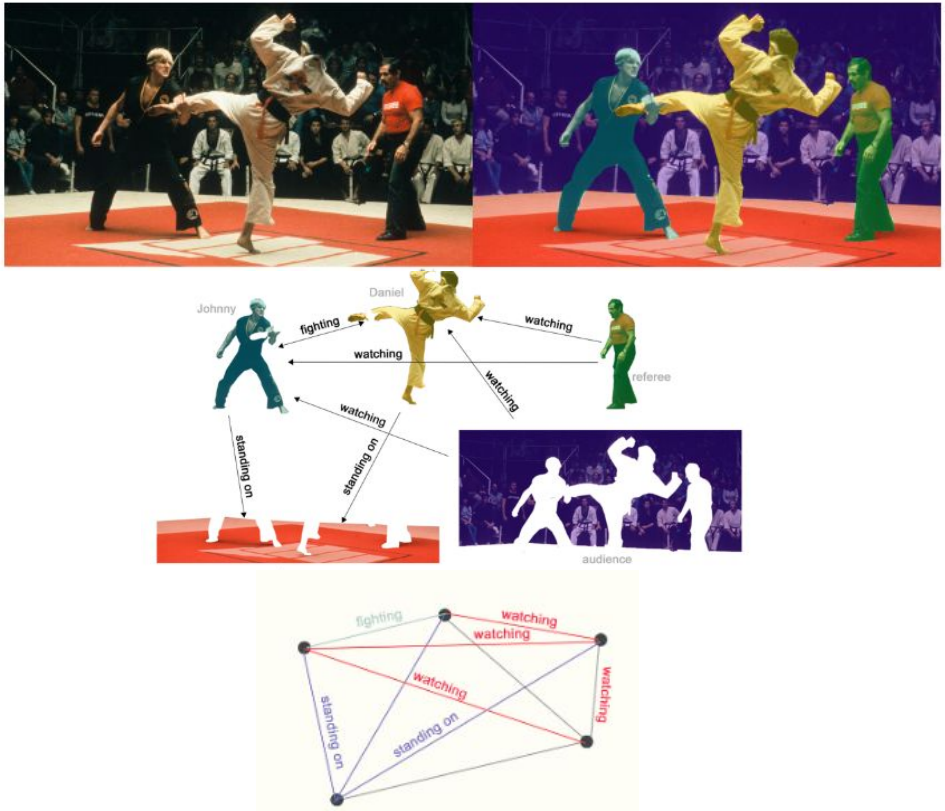
- More detail in this article
- **Batch size is how many training data points are processed before updating the model**
  - Makes predictions on batch, uses error to inform gradient descent algorithm
- Larger batch size
  - More accurate gradient descent
  - Fewer updates per epoch
  - Less noisy gradient
  - Less general, can get stuck in local minima
- Small batch size
  - Less accurate gradient descent
  - More updates per epoch
  - Noiser gradient
  - Can escape local minima





What is a Graph Neural Network (GNN)?

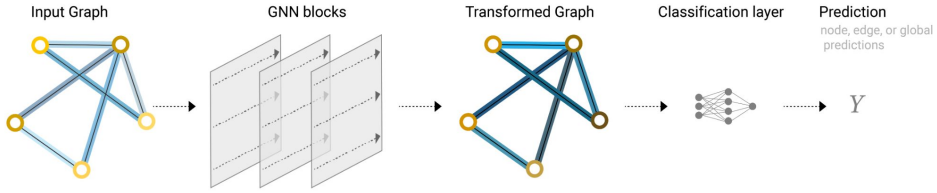
- A graph neural network is a neural network defined on graph-structured data
- Data represented by
  - Nodes (objects, e.g. particle hits in detector)
  - Edges (relationships, e.g. spatial proximity of two hits in detector)
  - Globals (property of group, e.g. total energy)
- Much better explained in this article



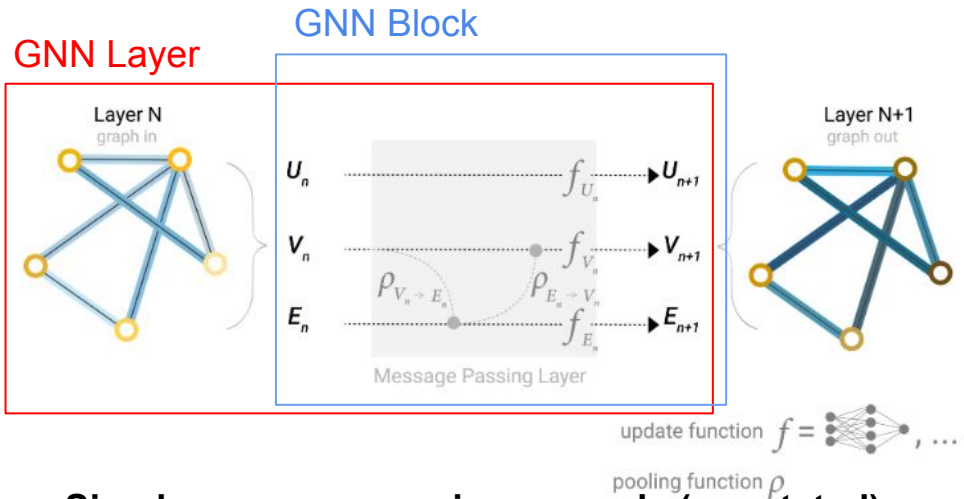
Example of data turned into a graph input

What is a Graph Neural Network (GNN)?

- We input a graph through GNN layers which define *how information flows*
- Example GNN layer strategies are:
  - Independent updates (no connectivity)
    - Apply learned functions to nodes, edges, globals independently
  - Message passing (most common, many versions)
    - Edges gather information from connecting nodes
    - Edges compute a “message” for each neighboring node
    - Nodes gather messages from edges
    - Graph is updated
- GNN Models have 3 basic type of predictions:
  - Node level predictions
  - Edge level predictions
  - Graph level predictions
- Update functions are MLPs



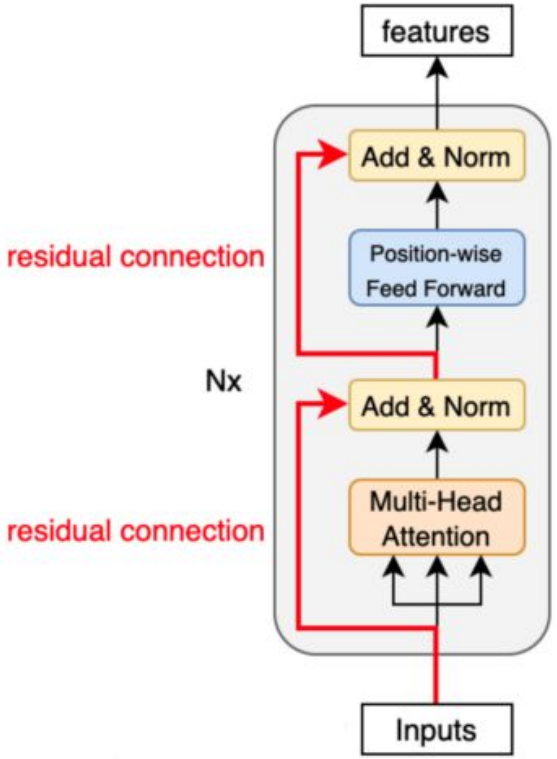
Example of full GNN model



Simple message passing example (annotated)

What is a Transformer Encoder Block?

- A transformer encoder block is a neural network layer that
  - A neural network block that operates on a set of feature vectors
  - Updates each vector using information from related vectors
  - Produces context-aware features of the same shape
  - Can be stacked to build progressively larger context
    - This is because the model computes residuals or “corrections”, not replacements
- In the context of graphs
  - updates node embeddings using information from neighboring nodes and edges
- Analogy:
  - MHSA = “Talk to other nodes”
  - FFN = “Reflect privately”



What is Multi-Head Self-Attention (MHSA)?

- Self-attention lets each element decide which other elements matter and how strongly to combine them
- Multiple heads let the model learn different interaction patterns at the same time

$$x_i' = \sum_{j \in \mathcal{C}(i)} \alpha_{ij} \, v_j$$

$$\alpha_{ij} = \frac{\exp\left(\frac{q_i^\top k_j}{\sqrt{d_k}}\right)}{\sum_{j' \in \mathcal{C}(i)} \exp\left(\frac{q_i^\top k_{j'}}{\sqrt{d_k}}\right)}$$

x_i : node embedding (hit-level features)
q_i : what this node is looking for
k_j : what neighboring nodes offer
v_j : what information neighbors contribute

$$q_i = W_Q x_i, \quad k_j = W_K x_j, \quad v_j = W_V x_j$$

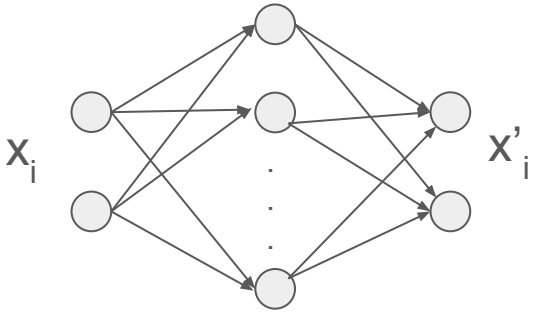
$\mathcal{C}(i)$: set of elements that can interact with i (for graphs: nodes connected to i)

What is a Feed Forward Network (FFN)?

- A position-wise feed-forward network applied to each element independently
- Uses the same two-layer MLP for all elements (shared weights)
- No communication between elements occurs in this layer
- Adds nonlinear feature refinement after self-attention
- Aside:
  - Feed Forward Networks are more general than position wise FFNs

$$x_i' = \mathrm{FFN}(x_i)$$
$$\mathrm{FFN}(x) = W_2\,\sigma(W_1 x + b_1) + b_2$$

x_i : element / node
σ : nonlinear activation (ReLU, GELU)
W_1, W_2 : learned linear layers

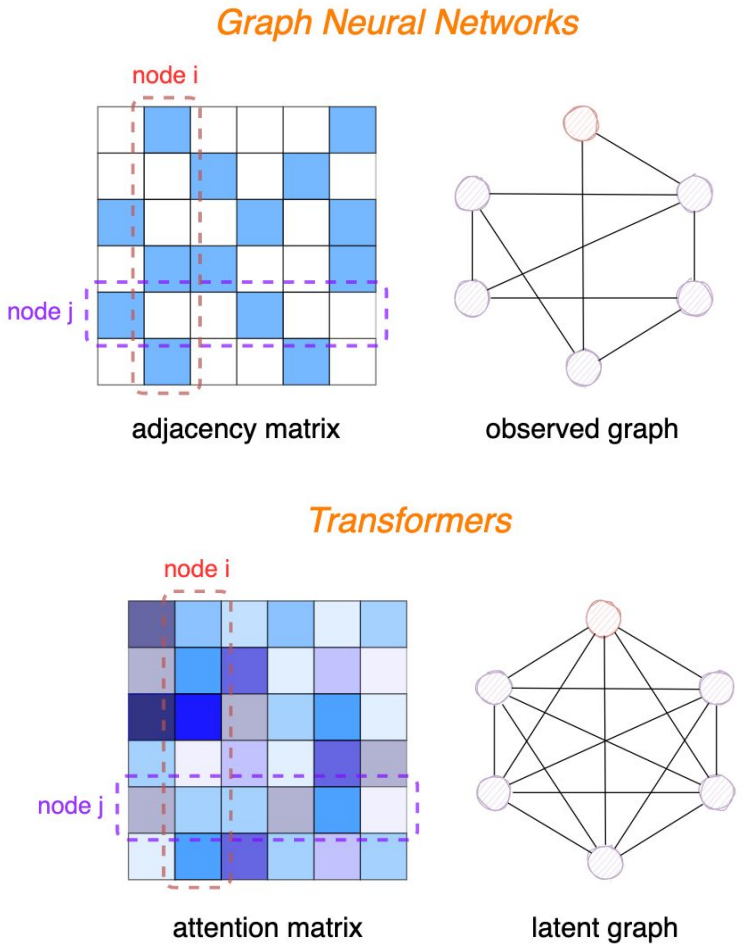


Example FFN for node with 2 features. Same FFN is applied to all nodes independently

Citation:
https://towardsdatascience.com/how-to-build-graph-transformers-with-o
-n-complexity-d507e103d30a/

What is a Graph Transformer?

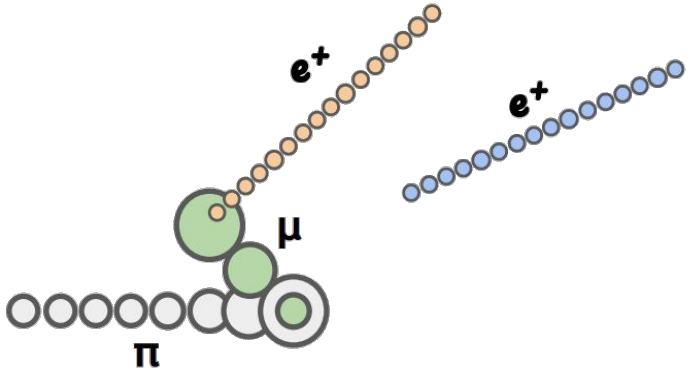
- A graph transformer is a GNN that generalizes message passing using transformer-style attention
- These are among the most expressive modern ML models
  - Can solve a wider class of problems than most models
- GNN
  - Edges define where information flows
  - Message passing same “strength” for all edges
- Graph transformers
  - Attention learns how strongly information flows along those edges.
  - Message passing differs in learned “strength” for different edges



Comparison of GNNs and Transformers in terms of message passing over different structures

G4Pioneer Monte Carlo + Detector Response

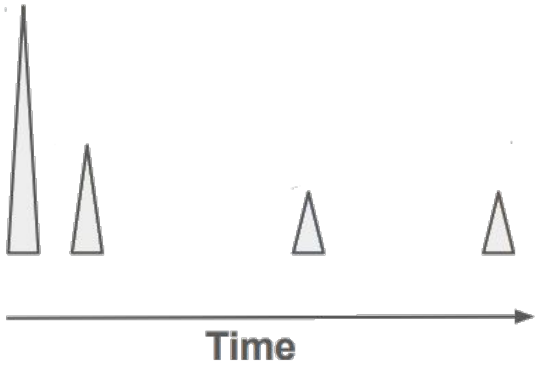
- **G4Pioneer Monte Carlo** simulates the underlying physics
  - Geant4 Fork
- **Detector Response** converts truth into detector-level signals
  - Depends on detector geometry
  - Handles event mixing
- ML training requires both detector-level outputs and truth-level information



Cartoon example of detector response with truth information added to hits

ML Ready Dataset Gaudi Pipeline (Time Group Forming)

- Gaudi pipeline does simple traditional labeling and reconstruction
- **Example time group forming**:
  - Detector response events are “split” into time-based clusters
    - Really, we just label hits with a time group they belong to
  - Most the time one time group == one particle
    - This helps the later ML stages reconcile particle specific features



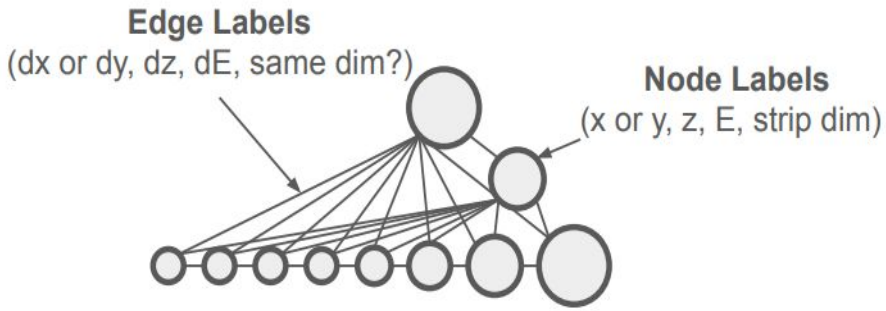
Cartoon example of time group spikes in energy deposition

Representing Information as a Graph

- Our final step before feeding the data into the model is constructing a fully connected graph
- Nodes contain hit specific information
  - Ex. position, deposited energy, etc.
- Edges contain related information between hits
  - Ex. difference in positions
- Global information for event level inputs
  - Ex. Total Energy
- We can append information to these graphs as downstream models make predictions

Graph level feature example:

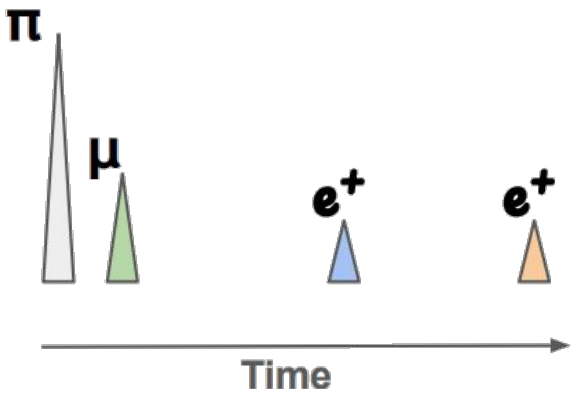
- Total energy



Graph representation of base detector information

Group Classifier

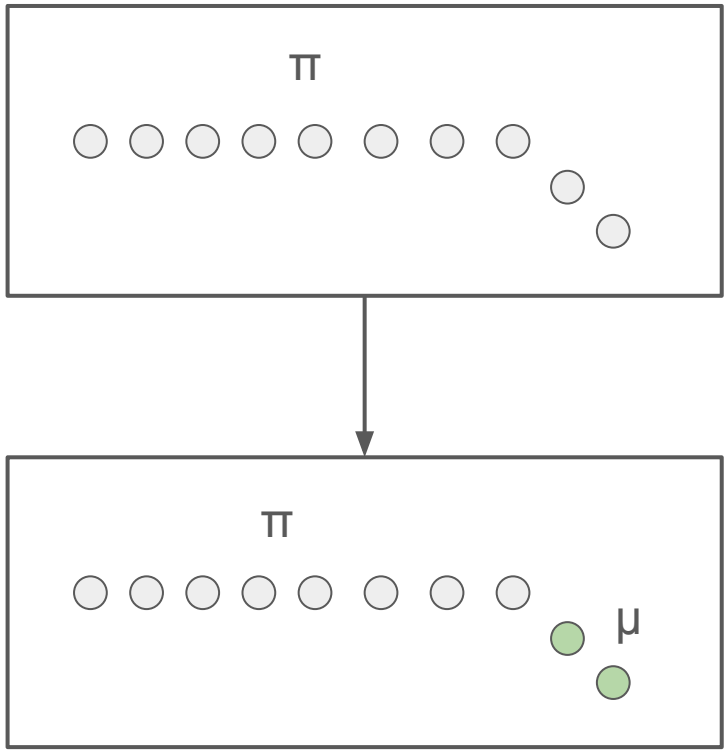
- **Multi-label graph transformer classifier**
  - Subgraph level task
- Inputs (represented as a graph):
  - Detector response event information
  - Time grouping labels
- Targets:
  - Truth level representation of what particles are in the group
  - Particularly: `[pionInGroup, muonInGroup, MIPinGroup]`
- Outputs:
  - Predicted probability that the time group belongs to each class `[prob_pionInGroup, prob_muonInGroup, prob_MIPinGroup]`



Cartoon example of time group spikes in energy deposition, now labeled by group classifier

Group Splitter

- **Multi-label graph transformer classifier**
  - Node level task
- Inputs (represented as a graph):
  - Detector response event information
  - Time grouping labels
  - Predicted time group level particle class
- Targets:
  - Truth level representation of the particle type for each hit
  - Particularly, for each hit in the graph: `[[is_pion, is_muon, is_mip], [is_pion, is_muon, is_mip], …]`
- Outputs:
  - Predicted probability that each hit belongs to each class `[[prob_pion, prob_muon, prob_mip], [prob_pion, prob_muon, prob_mip], …]`



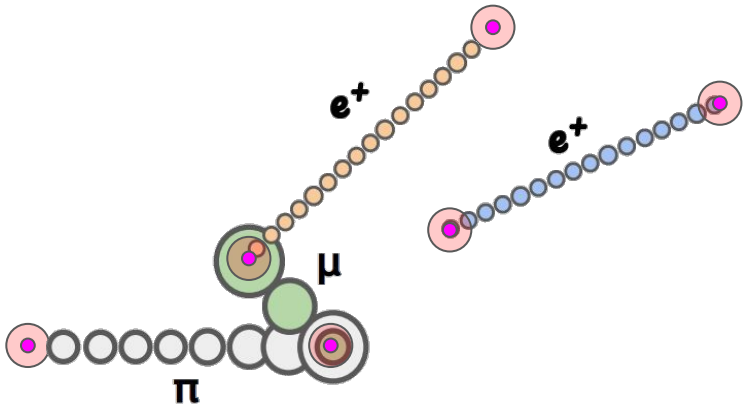
Cartoon example of how a time group may be split into multiple particle classifications

Endpoint Regressor

- **Quantile graph transformer regressor**
- Inputs:
  - Detector response event information represented as a graph
  - Predicted time group level particle class
  - Predicted hit level particle classes
- Targets:
  - Truth level endpoints
  - Particularly: `[[x_start, y_start, z_start], [x_end, y_end, z_end]]`
- Outputs:
  - Predicted endpoints with quantiles `[[[x_start_q1, y_start_q1, z_start_q1], [x_end_q1, y_end_q1, z_end_q1]], ...]`

● = Median endpoint estimate

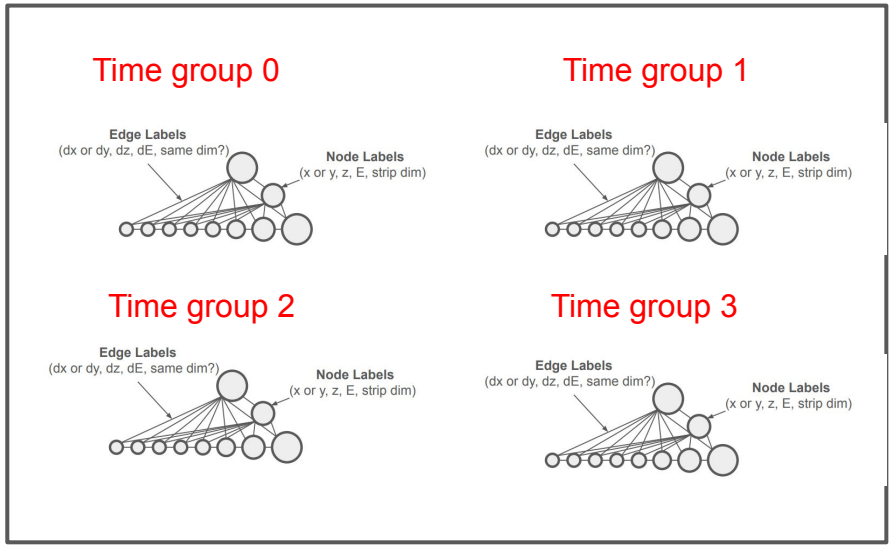
= Error range determined from quantiles



Cartoon example of how we estimate endpoints with uncertainties given by the quantiles

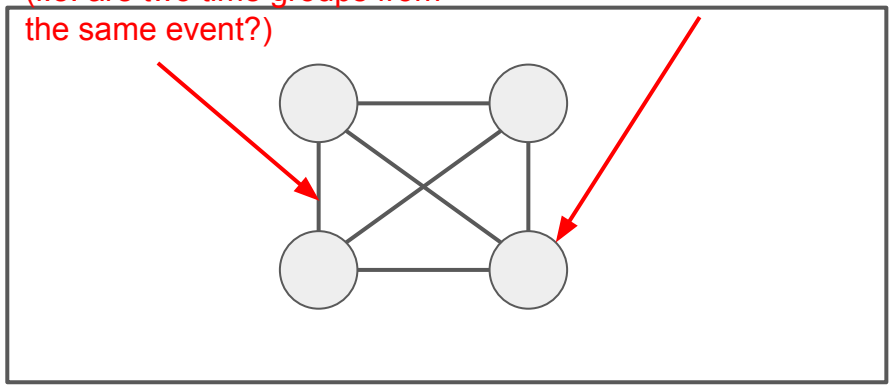
Pattern Finder

- **Affinity graph transformer classifier**
  - Edge level task
- Inputs (represented as a graph):
  - Detector response event information
  - Time grouping labels
  - Predicted time group level particle class
  - Predicted hit level particle classes
- Targets:
  - Agreement between truth level event origins between nodes; i.e. “are these two nodes from the same monte carlo truth event”. Explicitly:
    `target_affinity[i][j] = 1 if origin_id[i] == origin_id[j]`
- Outputs:
  - Predicted “affinities” (0,1) between time groups



Edges:
Learned affinity between groups (i.e. are two time groups from the same event?)

Node labels:
On per time group graph

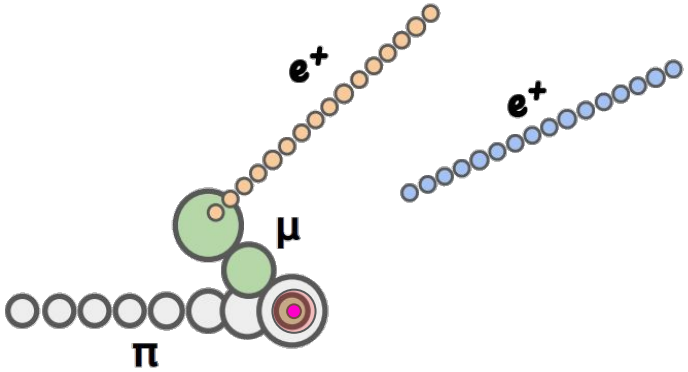


Pion Stop Regressor

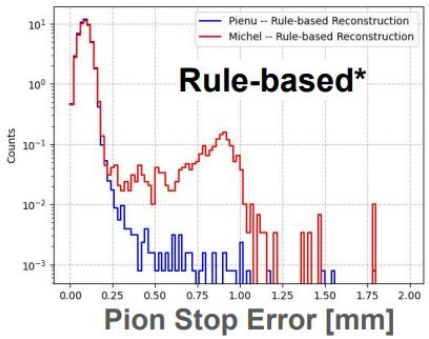
- **Quantile graph transformer regressor**
  - Graph level task
- Inputs (represented as a graph):
  - Detector response event information
  - Time grouping labels
  - Predicted time group level particle class
  - Predicted hit level particle classes
  - Predicted affinities
- Targets:
  - Truth level pion stop
  - Particularly: `[x_stop,y_stop,z_stop]`
- Outputs:
  - Predicted pion stop `[[[x_stop_q1, y_stop_q1, z_stop_q1], …]`
- Notes:
  - Quantiles effectively give uncertainties if chosen to be top 84%, top 50%, and top 16% quantiles = median ± 1 standard deviation

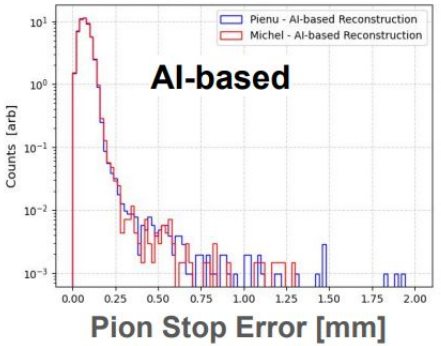
● = Pion stop estimate

= Error range determined from quantiles



Cartoon example of pion stop estimate

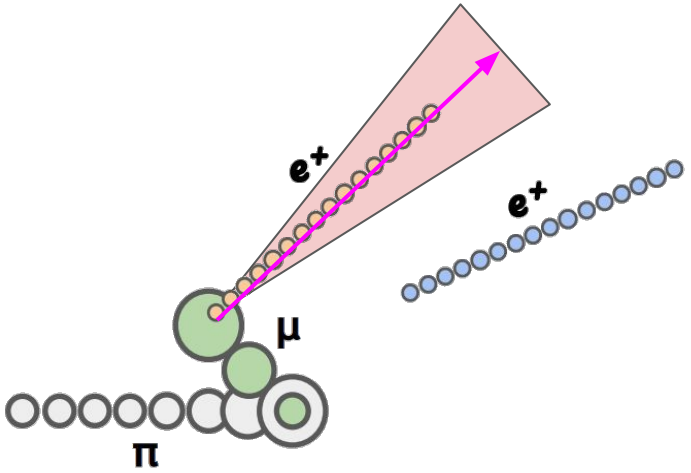




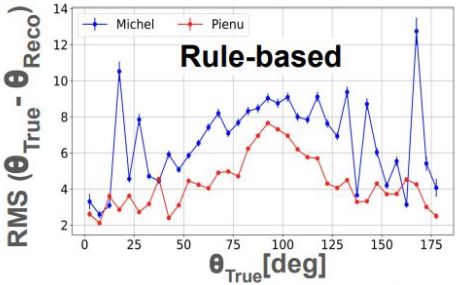
direction in detector coordinate system with uncertainty

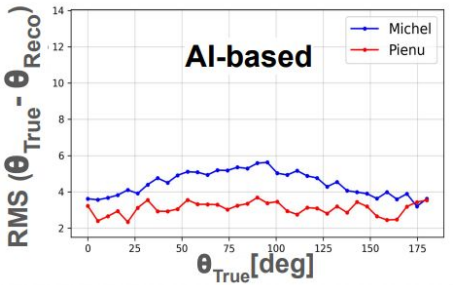
Positron Angle Regressor

- **Quantile graph transformer regressor**
  - Graph level task
- Inputs (represented as a graph):
  - Detector response event information
  - Time grouping labels
  - Predicted time group level particle class
  - Predicted hit level particle classes
  - Predicted affinities
- Targets:
  - Truth level direction vector for positron angle
  - Particularly: `[(sinθcosφ), (sinθsinφ), (cosθ)]`
- Outputs:
  - Predicted direction vector for positron angle `[[(sinθcosφ)_q1, (sinθsinφ)_q1, (cosθ)_q1], …]`



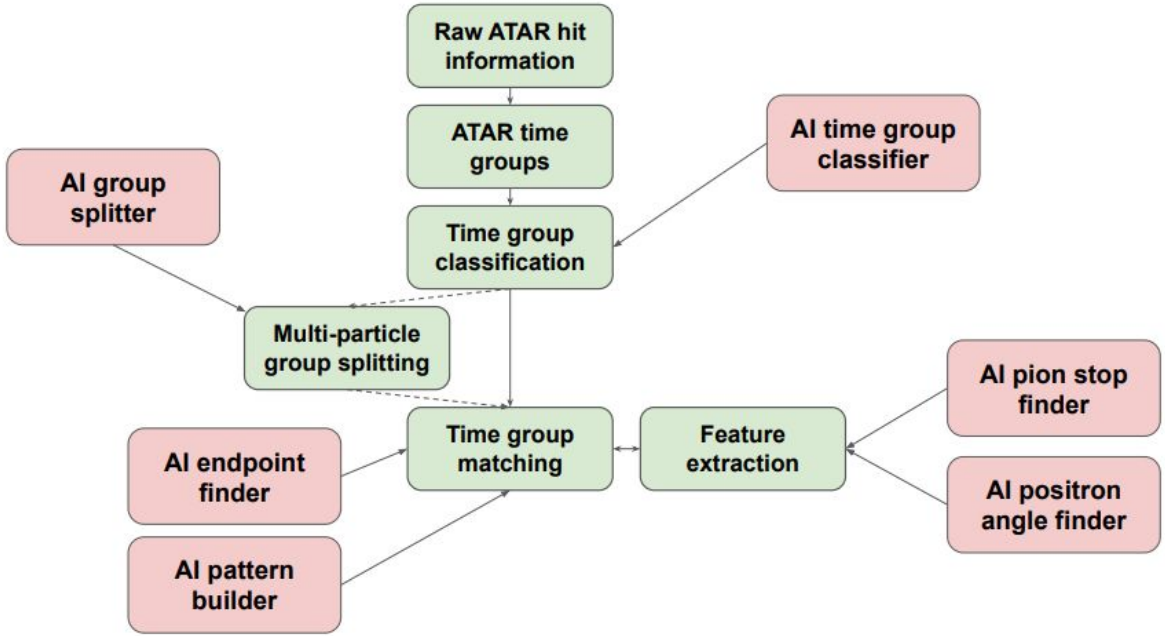
Cartoon example of positron angle estimate





Full Reconstruction Diagram

- Conceptually, all the same stages
- The idea is the ML models are “implementations” of a stage
  - In principle they can be swapped out with a traditional reconstruction (non-machine learning)
  - Allows for benchmarking of different implementations



Why Have Divisions?

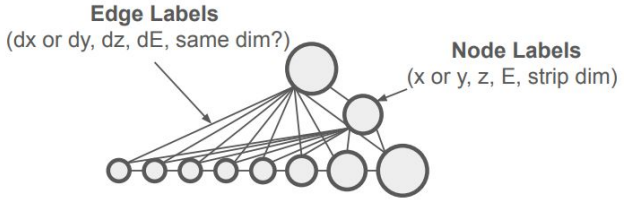
- Each pipeline stage operates on a well-defined “sample”
  - Each division defines a **dataset boundary**
    - A dataset boundary is the point at which the meaning of a row is fixed and written to storage, so downstream stages can rely on it without knowing how it was produced
    - It’s okay to create appending files or masks,
    - If you must mutate data (i.e. change your definition of a “sample” then you create another dataset boundary), then you should create a new dataset boundary or “division”
- If we don’t create clear divisions
  - Hidden coupling between stages can occur
    - I.e. you change one stage, an unrelated stage no longer works
  - Batching becomes unclear
    - Your definition of a batch changes between data boundaries
    - Memory and performance become less predictable
- In short, each division should scale separately

GraphRecord Former

- We don’t store data as their graphs, we store them as the minimal information needed to build the graph

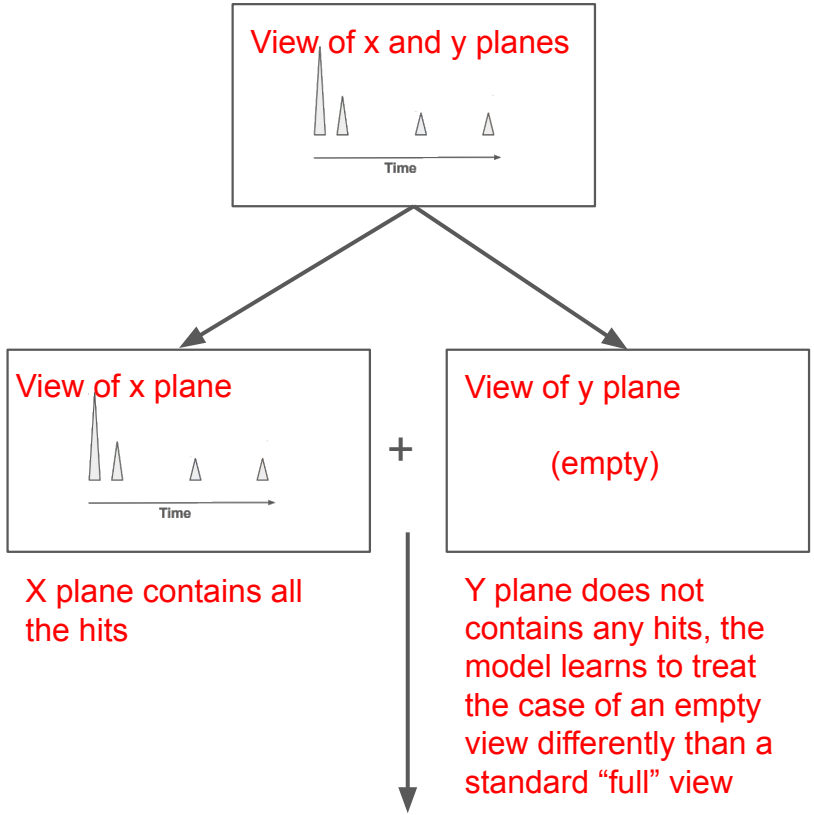
- As a result we have a “data loading stage” before running any model where we construct a graph from the data

```
Columns:
_____________
event_id
theta
phi
pion_stop_x
pion_stop_y
pion_stop_z
total_pion_energy
...
hit_coord→ list < float >
hit_z→ list < float >
hit_energy→ list < float >
hit_pdg_mask→ list < int >
```



Stereoscopic View

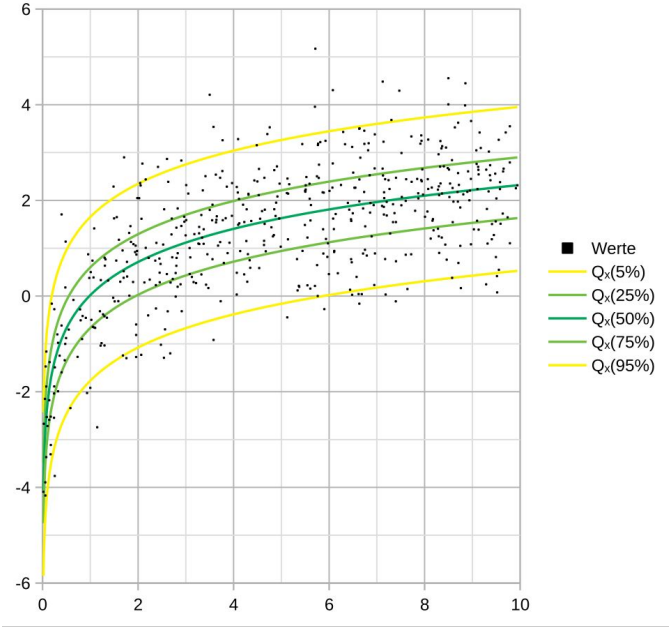
- Before each “Division 1” model “does it’s job”, it first construct a “stereoscopic” view of the data
  - Hits are split by view:
    - 0 = X plane, 1 = Y plane
  - Each view is embedded independently, then fused: [emb_view0, emb_view1, mask0, mask1]
    - mask_i = 1 if view i has hits, else 0
  - **This process is learned for each model**
- Why this helps:
  - Treats view ID as discrete routing
  - Explicitly encodes that intermediate views (e.g. “0.5”) do not exist
- Why we want this stage learnable:
  - weight each view
  - rescale per-view features
  - handle missing or weak views
- Why it’s model-specific
  - Different tasks rely on views differently
    - e.g. missing Y view matters more for endpoint finding than for particle classification



Exaggerated cartoon example of how a stereoscopic view can help in the case of missing views

Quantiles

- For regressions, quantiles are useful to extract uncertainties
- Uses a quantile regression loss function
- We can use quantile regression to have our models target a specific quantiles
  - Usually we select
    - Lower quantile = 16 ~ 1σ
    - Mid quantile = 50 ~ median
    - Upper quantile = 84 ~ +1σ



Wikipedia example of Quantile Regression. Notice how it forms confidence bands around the best fit (or median)

Backpropagation (Explicit, Part I)

- Diagram with some labels to conceptualize the variables

Indices:

$l \equiv$ layer index

$i \equiv$ neuron index in layer l

$j \equiv$ neuron index in layer $l-1$

$k \equiv$ neuron index in layer $l+1$

$t \equiv$ training iteration

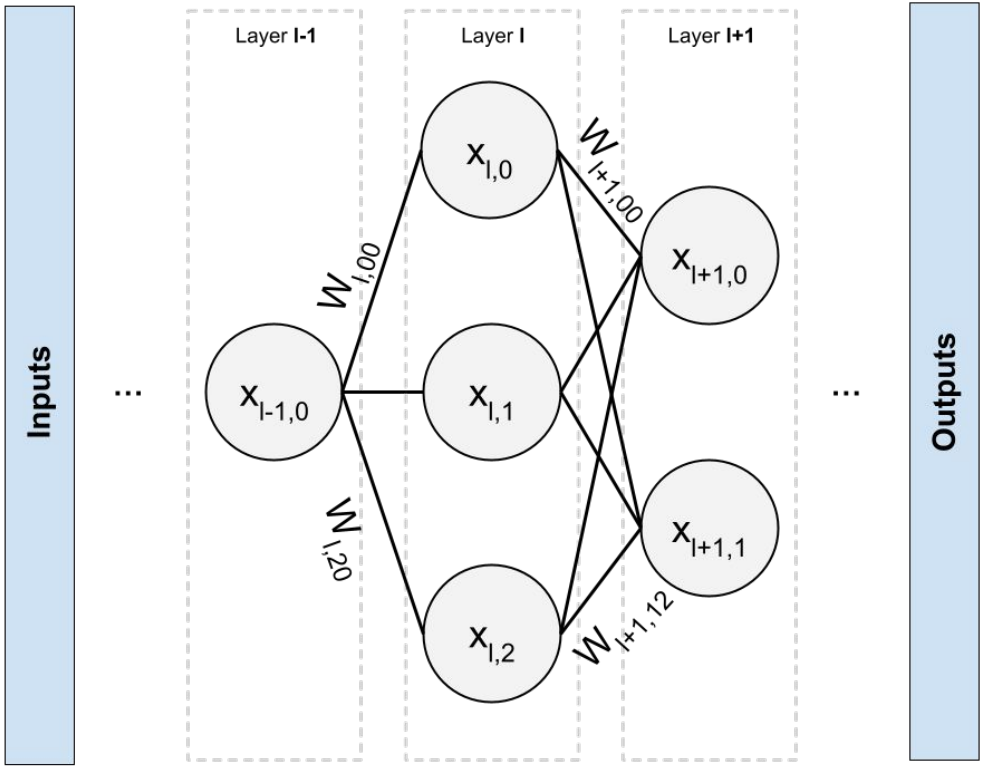
Core quantities:

$L \equiv$ loss (scalar)

$x_{l,i} \equiv$ activation of neuron i in layer l

$W_{l,ij} \equiv$ weight from $(l-1, j)$ to (l, i)

$\alpha \equiv$ learning rate



Backpropagation (Explicit, Part II)

- Equations for how to update weights from back propagation

Indices:

$l \equiv$ layer index
$i \equiv$ neuron index in layer l
$j \equiv$ neuron index in layer $l-1$
$k \equiv$ neuron index in layer $l+1$
$t \equiv$ training iteration

Core quantities:

$L \equiv$ loss (scalar)
$x_{l,i} \equiv$ activation of neuron i in layer l
$W_{l,ij} \equiv$ weight from $(l-1, j)$ to (l, i)
$\alpha \equiv$ learning rate

1. Backpropagate activation gradients:

$$\frac{\partial L}{\partial x_{l,i}} = \sum_k \frac{\partial L}{\partial x_{l+1,k}} \frac{\partial x_{l+1,k}}{\partial x_{l,i}}$$

2. Weight gradient via local chain rule:

$$\frac{\partial L}{\partial W_{l,ij}} = \frac{\partial L}{\partial x_{l,i}} \frac{\partial x_{l,i}}{\partial W_{l,ij}}$$

3. Gradient descent update to weights:

$$W_{l,ij}^{(t+1)} = W_{l,ij}^{(t)} - \alpha \frac{\partial L}{\partial W_{l,ij}}\bigg|_{W=W^{(t)}}$$

Backpropagation (Explicit, Part III)

- The back propagation step is a recursive step

Indices:

$l \equiv$ layer index
$i \equiv$ neuron index in layer l
$j \equiv$ neuron index in layer $l-1$
$k \equiv$ neuron index in layer $l+1$
$t \equiv$ training iteration

Core quantities:

$L \equiv$ loss (scalar)
$x_{l,i} \equiv$ activation of neuron i in layer l
$W_{l,ij} \equiv$ weight from $(l-1, j)$ to (l, i)
$\alpha \equiv$ learning rate

The backpropagation step:

$$\frac{\partial L}{\partial x_{l,i}} = \sum_k \frac{\partial L}{\partial x_{l+1,k}} \frac{\partial x_{l+1,k}}{\partial x_{l,i}}$$

This formula is recursive, so for N layers you have explicitly:

$$\frac{\partial L}{\partial x_{0,i_0}} = \sum_{i_1} \sum_{i_2} \cdots \sum_{i_N} \frac{\partial L}{\partial x_{N,i_N}} \prod_{m=0}^{N-1} \frac{\partial x_{m+1,i_{m+1}}}{\partial x_{m,i_m}}$$

Or more commonly you can write in Jacobian form, which cleans up the notation

$$\frac{\partial L}{\partial x_0} = \prod_{m=0}^{N-1} \left(\frac{\partial x_{m+1}}{\partial x_m} \right)^\top \frac{\partial L}{\partial x_N}$$

Backpropagation (Explicit, Part IV)

- The actual values of $x_{l,i}$ are *architecture dependent*
- In this simple example, you can compute them by defining a bias and activation function (eg. softmax)

We can define $z_{l,i}$ for the raw computed "response" from the previous layer:

$$z_{l+1,k} = \sum_j W_{l+1,kj}\, x_{l,j} + b_{l+1,k}$$

Then we apply some activation, usually to constrain values between 0 and 1 to prevent exponential blowup. This allows to (finally) formally compute all needed partial derivatives:

$$x_{l+1,k} = \phi(z_{l+1,k})$$

$$\frac{\partial x_{l+1,k}}{\partial x_{l,i}} = \phi'(z_{l+1,k})\, W_{l+1,ki}$$

Core quantities:

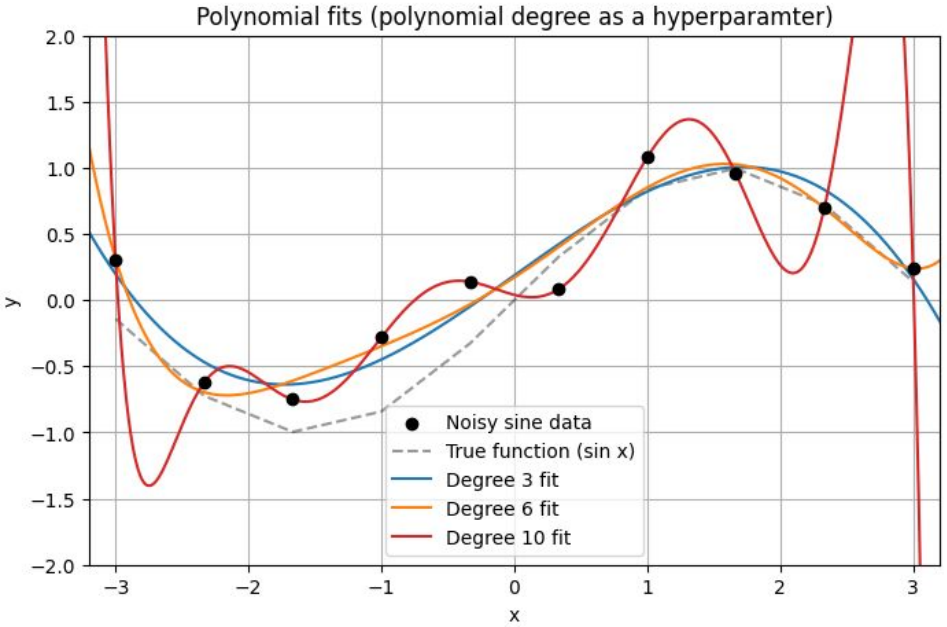
$L \equiv$ loss (scalar)
$x_{l,i} \equiv$ activation of neuron i in layer l
$W_{l,ij} \equiv$ weight from $(l-1, j)$ to (l, i)
$\alpha \equiv$ learning rate
$z_{l,i} \equiv$ pre-activation (raw linear response)
$b_{l,i} \equiv$ bias parameter for neuron (l, i)
$\phi(\cdot) \equiv$ activation function

Indices:

$l \equiv$ layer index
$i \equiv$ neuron index in layer l
$j \equiv$ neuron index in layer $l-1$
$k \equiv$ neuron index in layer $l+1$
$t \equiv$ training iteration

Hyperparameters Need to be Tuned Appropriately

- Improper hyperparameter choices can lead to
  - Overtraining
  - Slower convergence
  - Vanishing or exploding gradients
    - Impossible to find true solution at this point
- Tuning hyperparameters can be hard
  - They interact with each other (often non-linearly)
  - Optimal choice varies with each dataset/model



Example: Fitting data with a polynomial. The degree of the polynomial, d, is a hyperparameter

Hyperparameters (Learning Rate)

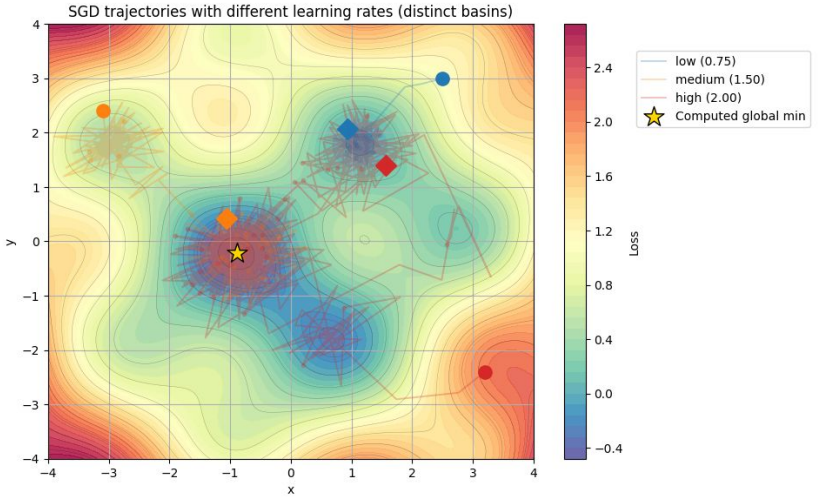
- More detail in this article
- **Learning Rate controls how large each parameter update is in the direction of the gradient**
- Larger learning rate:
  - Bigger, more exploratory steps
  - Less likely to get stuck in local minima
  - Harder to settle precisely at the optimum
- Smaller learning rate:
  - Smaller, more precise steps
  - More likely to get trapped in local minima
  - Better at fine-tuning near the optimum

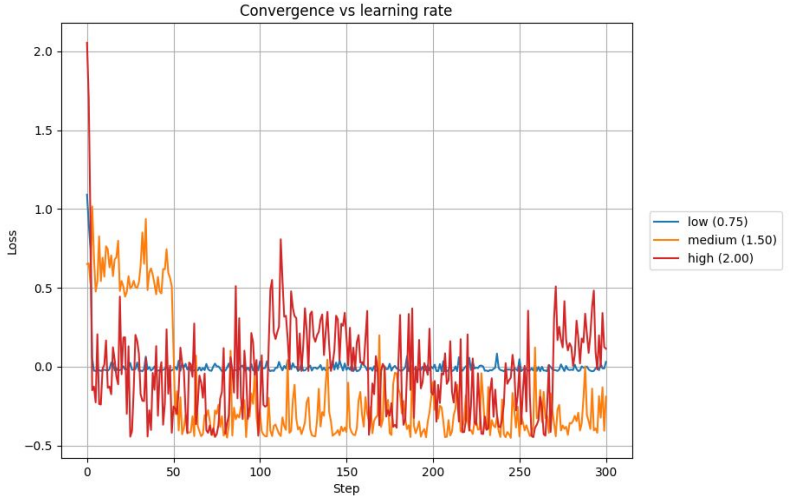
$$w_{\text{new}} = w_{\text{old}} - \alpha \nabla_w L(w)$$

w = model weights

α = learning rate

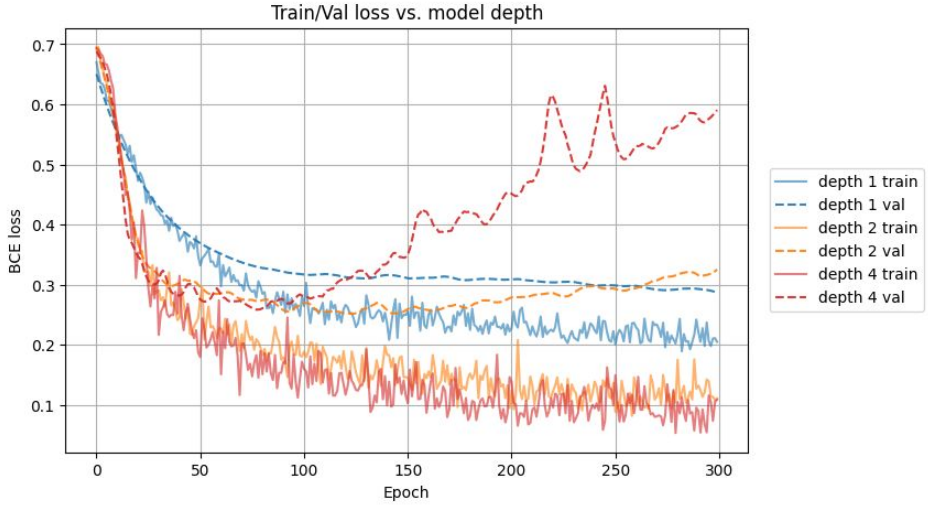
$\nabla_w L(w)$ = gradient of the loss with respect to the weights

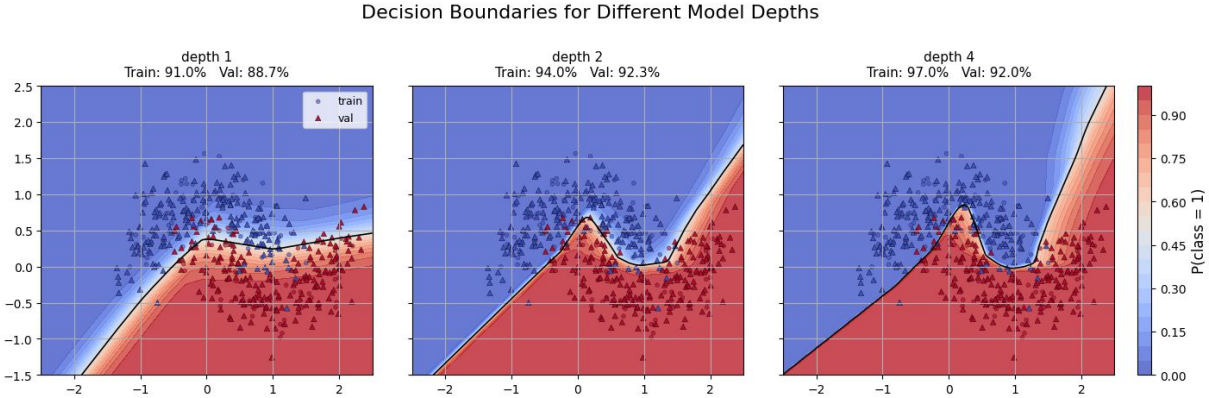




Hyperparameters Example (# of layers)

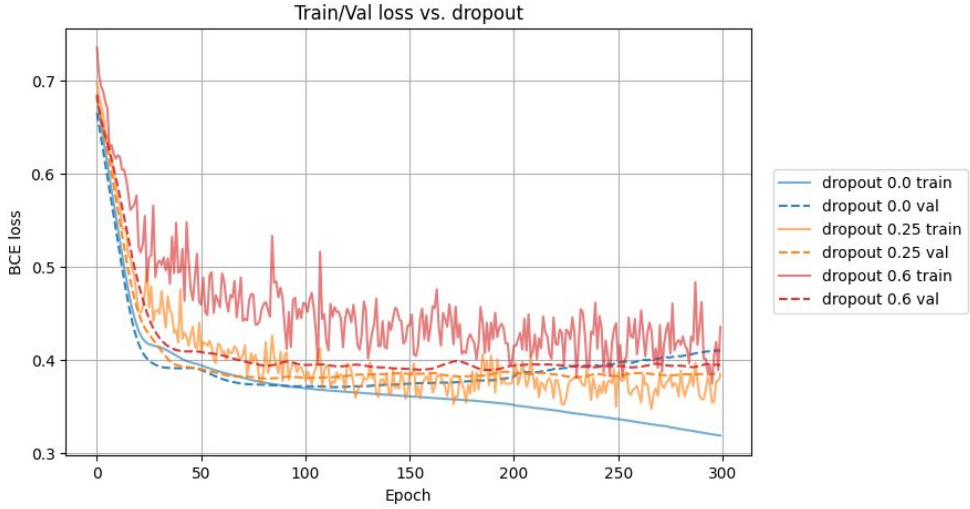
- More detail in this article
- **Hidden dimension sets the size of the model's internal feature representations at each layer.**
- More layers:
  - Learns more "hierarchical"/ "abstract" features
  - Higher risk of overfitting or training instability (higher loss fluctuations)
  - Higher memory and compute cost
- Less layers:
  - Learns "simpler"/"shallow" features
  - Lower risk of overfitting
  - Lower memory/compute cost

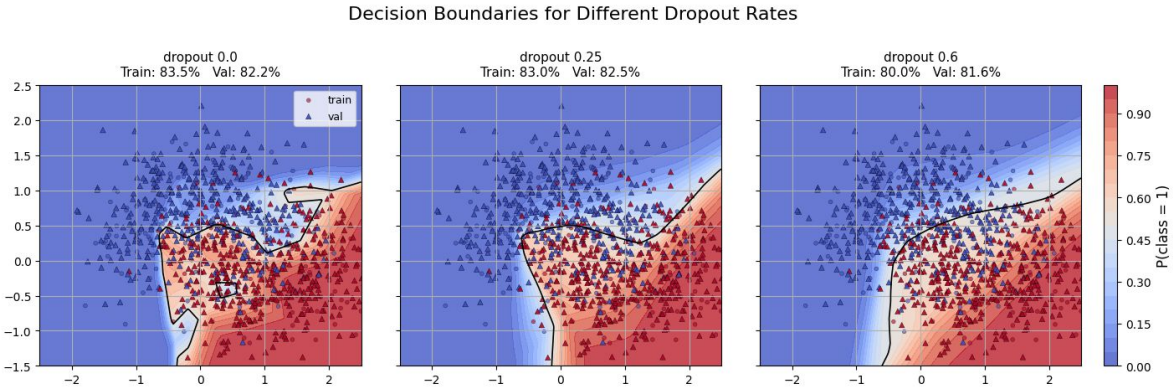




Hyperparameters (Dropout)

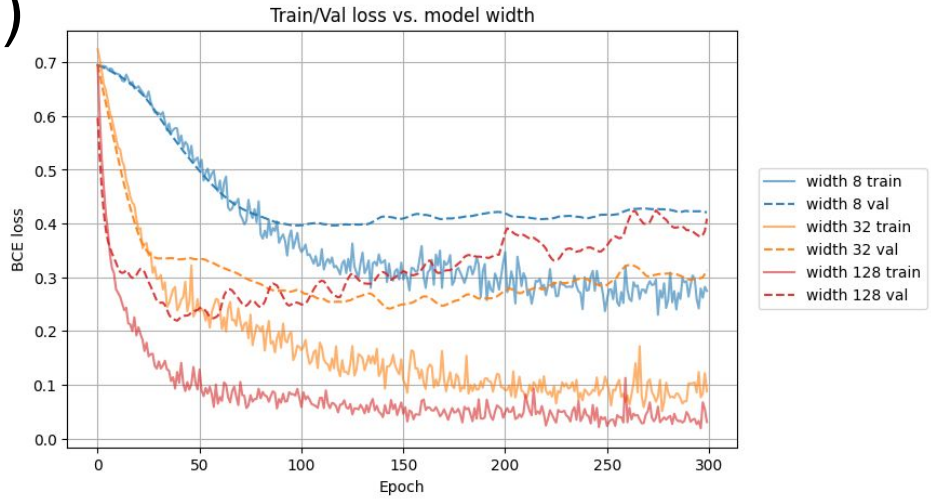
- More detail in this article
- **Dropout randomly disables a fraction of activations during training to reduce overfitting**
- Higher dropout:
  - Better for noisy datasets
  - Less risk of overfitting
  - Higher risk of underfitting
- Lower dropout:
  - Better for less noisy datasets
  - Less risk of underfitting
  - Higher risk of overfitting

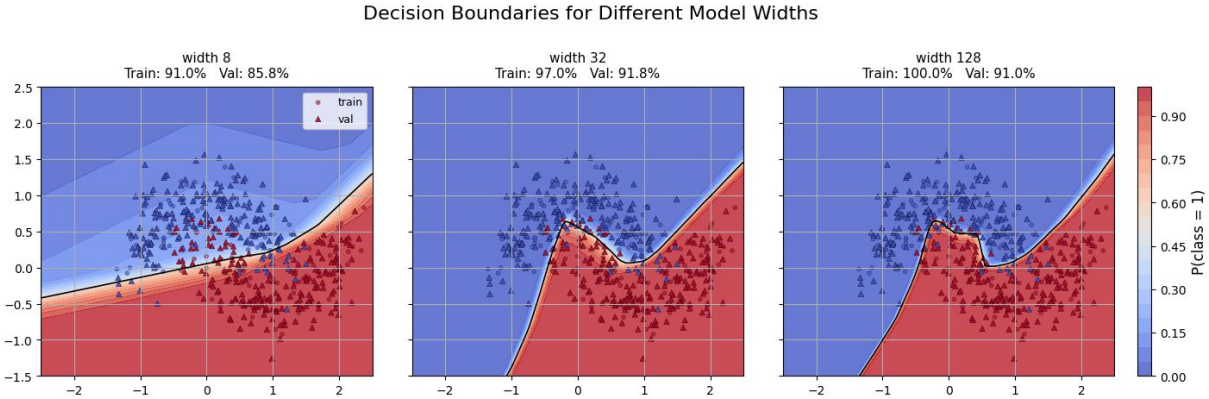




Hyperparameters (Hidden dim)

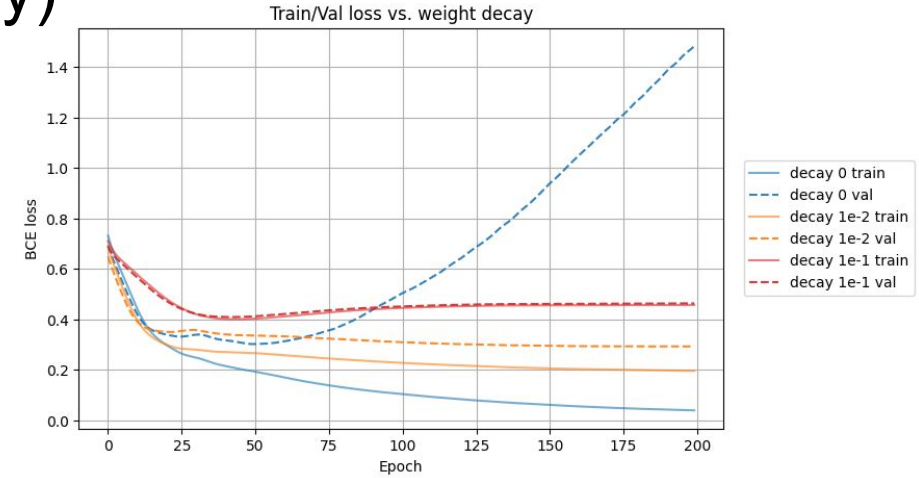
- More detail in this article
- **Hidden dimension sets the size of the model's internal feature representations at each layer.**
- Larger hidden dim:
  - Can represent more complex patterns
  - Higher risk of overfitting
  - Higher memory and compute cost
- Smaller hidden dim:
  - Limited ability to model complex patterns
  - Lower risk of overfitting
  - Lower memory/compute cost

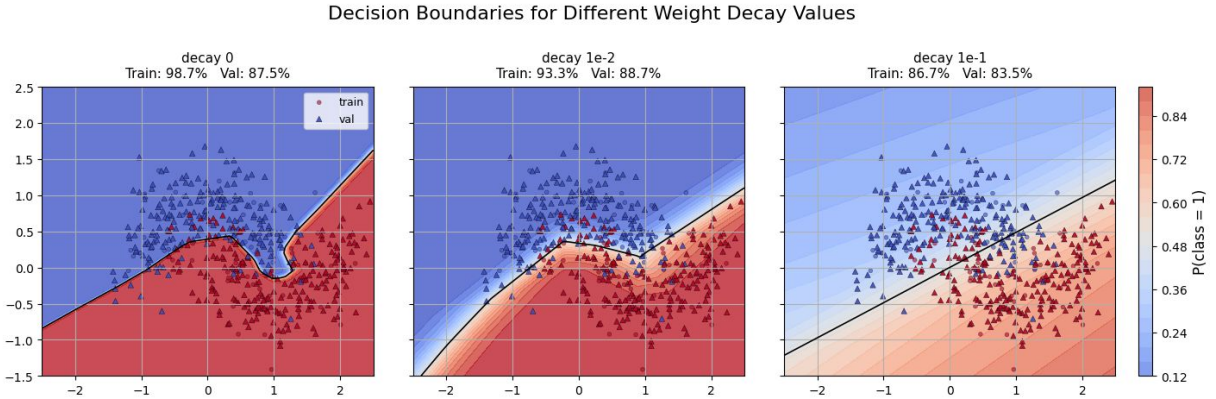




Hyperparameters (weight decay)

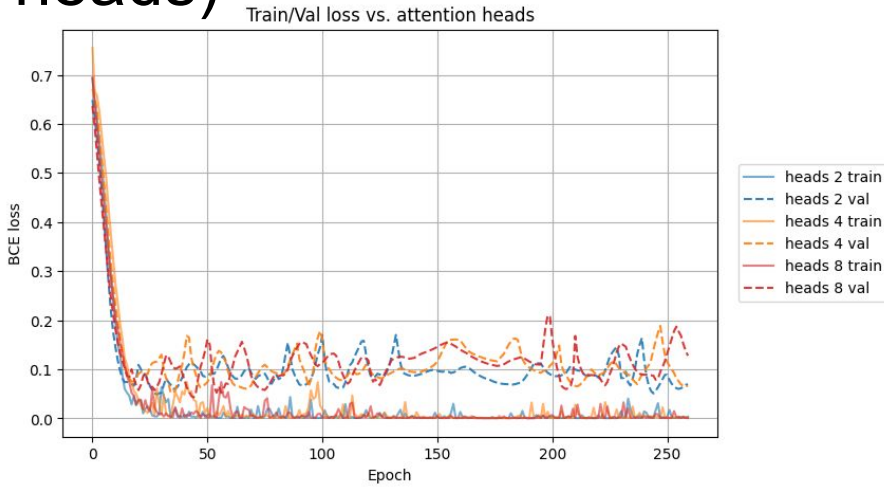
- More detail in this article
- **Weight decay adds a penalty on large weights to the loss, encouraging simpler models. Similar to dropout.**
- Higher weight Decay:
  - Better for noisy datasets, stronger regularization
  - Lower risk of overfitting
  - Higher risk of underfitting
- Lower weight decay:
  - Better for less noisy datasets, less regularization
  - Lower risk of underfitting
  - Higher risk of overfitting

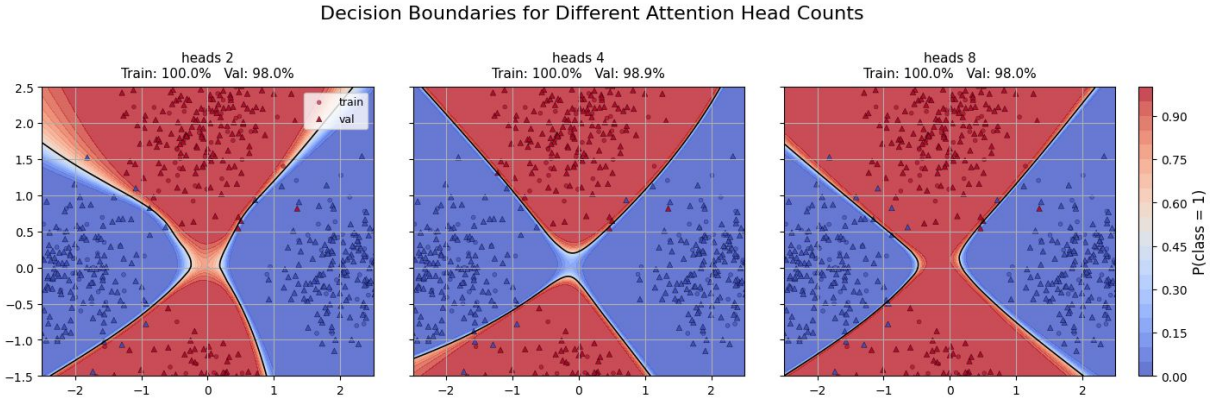




Hyperparameters (# of attention heads)

- More detail in this article
- **The number of attention heads controls how many independent subspaces the model attends to in parallel within each attention layer**
- More attention heads:
  - Learns multiple attention patterns in parallel
  - Higher memory/compute cost
  - Diminishing returns if per-head dimension is too small
  - Can overfit and destabilize training
- Fewer attention heads:
  - Learns fewer attention patterns
  - Lower memory/compute cost
  - May miss distinct relationships
  - Can underfit if data is very heterogeneous



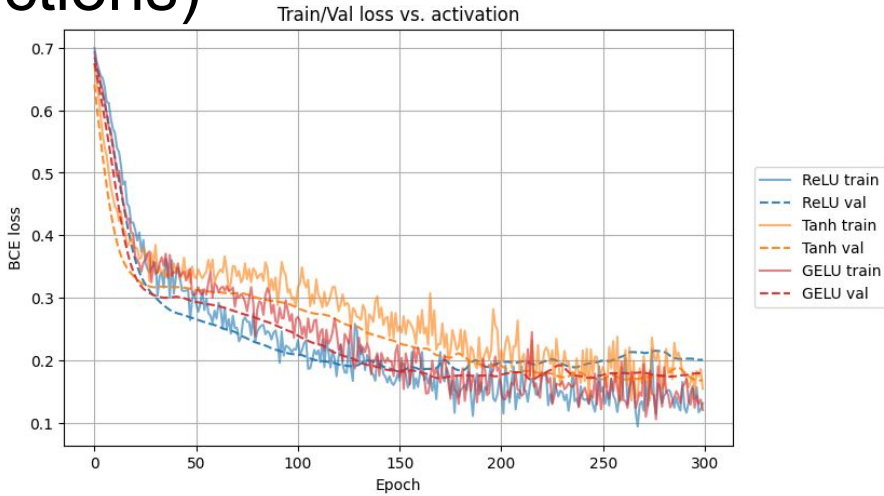


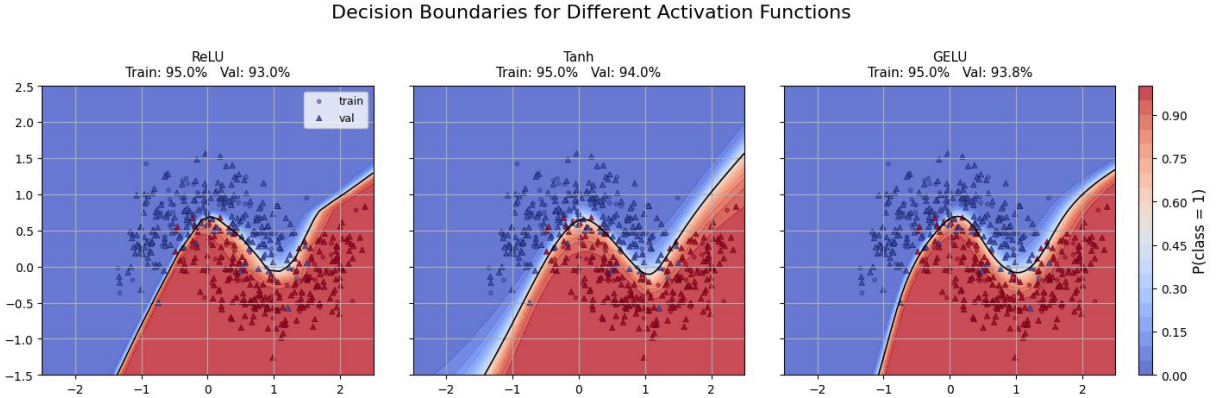
Hyperparameters (activation functions)

- More detail in this article
- **The activation function controls the nonlinearity applied at each layer**

Examples:

- ReLU
  - Simple, piecewise-linear activation
  - Fast training; can form sharp, noisy boundaries
- Tanh
  - Smooth, bounded activation
  - Produces smoother boundaries; may saturate
- GELU
  - Smooth, probabilistic gating
  - Balances smoothness and expressiveness



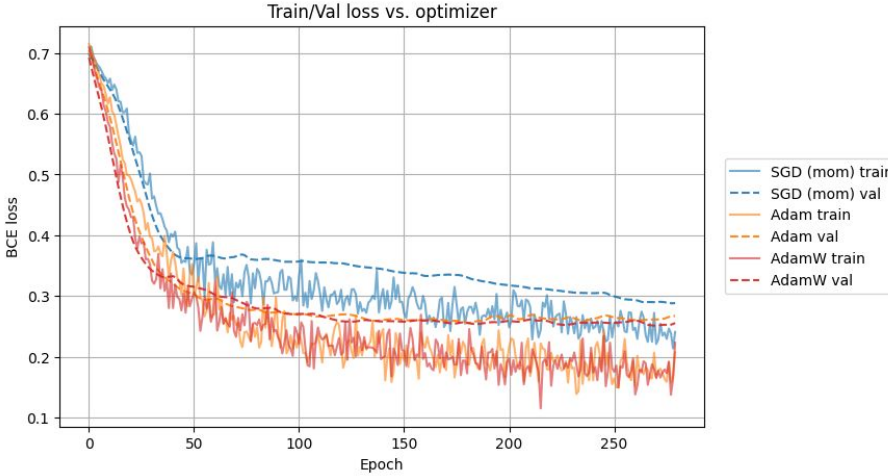


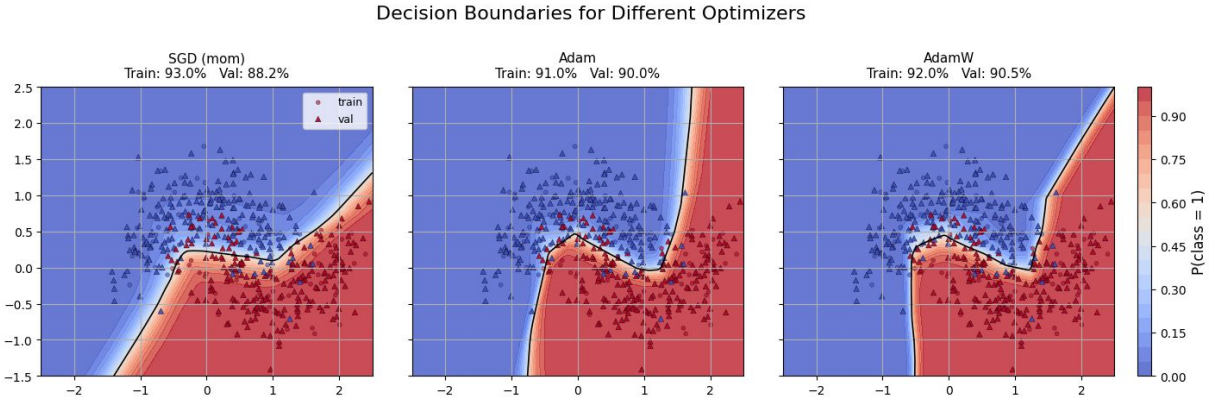
Hyperparameters (optimizers)

- More detail in this article
- **Optimizers control how gradients are transformed into parameter updates during training**

Examples:

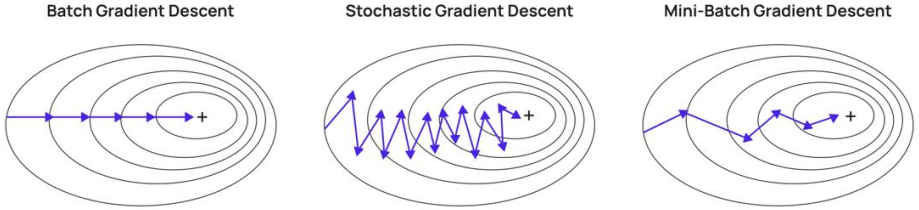
- Stochastic Gradient Descent (SGD) with momentum
  - Simple, stable updates
  - Often performs well on unseen data, not just the training set
  - Sensitive to learning rate and scaling
- Adaptive Moment Estimation (Adam)
  - Adaptive learning rates per parameter
  - Fast convergence, less sensitive to hyperparameter choice
  - Can overfit or generalize worse than SGD
- AdamW
  - Adam with decoupled weight decay
  - More reliable regularization behavior
  - Standard choice for modern deep models

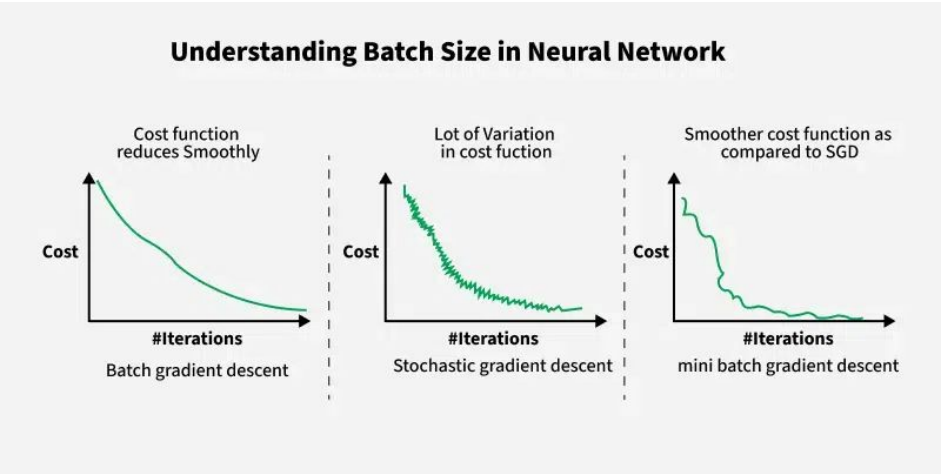




Hyperparameters (Gradient Descents)

- Much more detail in this article
- Batch Gradient Descent (BGD):
  - Uses the entire dataset in one iteration. The batch size is equal to the total number of training samples.
- Mini-Batch Gradient Descent (MBGD):
  - Uses a predefined number of samples from the dataset for each update. This method lies between batch and stochastic gradient descent (SGD).
- Stochastic Gradient Descent (SGD):
  - Processes only one sample at a time for each update, making the batch size equal to 1.





Hyperparameters (why are more layers harder to train)

- More layers → more loss instability
  - Due to back propagation
  - The size of your step becomes more “untrainable” the more layers you add
- Cases:
  - λ ≈ 1
    - Gradients stable as model layers increase
  - λ < 1
    - Gradients vanish as model layers increase
    - No more learning occurs!
  - λ > 1
    - Gradients blow up as model layers increase
    - Cannot converge on solution
- One goal of model architecture choice is to get λ ≈ 1

Backprop signal:
$$\frac{\partial L}{\partial x_0} = \prod_{m=0}^{N-1} \left(\frac{\partial x_{m+1}}{\partial x_m} \right)^\top \frac{\partial L}{\partial x_N}$$

Layer Jacobian:
$$x_{m+1} = \phi(W_{m+1} x_m + b_{m+1})$$
$$\frac{\partial x_{m+1}}{\partial x_m} = D_{m+1} W_{m+1}$$
$$\text{where} \quad D_{m+1} = \text{diag}(\phi'(W_{m+1} x_m + b_{m+1}))$$

Substitute:
$$\frac{\partial L}{\partial x_0} = \left(\prod_{m=0}^{N-1} W_{m+1}^\top D_{m+1} \right) \frac{\partial L}{\partial x_N}$$

Take norms:
$$\left\| \frac{\partial L}{\partial x_0} \right\| \leq \left\| \frac{\partial L}{\partial x_N} \right\| \prod_{m=0}^{N-1} \|W_{m+1}\| \, \|D_{m+1}\|$$

$$\implies E\left[\left\| \frac{\partial L}{\partial x_0} \right\| \right] \approx \left\| \frac{\partial L}{\partial x_N} \right\| \left(E[\|W\|] \, E[\|D\|] \right)^N$$

$$\text{Let} \quad \lambda \equiv E[\|W\|] \, E[\|D\|]$$

Note: See paper on arxiv

What is a Tree-Structured Parzen Estimator (TPE) (Part I)

- Given a (possibly stochastic) black box function you want to minimize (ex. model loss)

$$y = f(\theta)$$

- Hyperparams ≡ θ
- Define the following:

$$\gamma \equiv \text{"good"/"bad" fraction (say, 0.2)}$$
$$y^* \text{ is chosen such that } P(y < y^*) = \gamma$$
$$\text{so } y^* \text{ is in, say, the best } 20\% \text{ of losses}$$
$$\max(y^* - y, 0) \equiv \text{improvment}$$

- Then, for any given theta we can define *expected improvement*

$$\mathrm{EI}(\theta) \equiv \mathbb{E}[\max(y^* - y, 0) \mid \theta] = \int_{-\infty}^{y^*} (y^* - y)\, p(y \mid \theta)\, dy$$

- Goal: Maximize expected improvement

Note: See paper on arxiv

What is a Tree-Structured Parzen Estimator (TPE) (Part II)

- For a 1 dimensional y, it turns out to be easier to work with $p(\theta \mid y)$, we can invert using Bayes' rule:

$$p(y \mid \theta) = \frac{p(\theta \mid y)\, p(y)}{p(\theta)}$$

- And substitute

$$\mathrm{EI}(\theta) \propto \int_{-\infty}^{y^*} (y^* - y)\, \frac{p(\theta \mid y)}{p(\theta)}\, p(y)\, dy$$

- Since we don't know every value of y this becomes an impossible task. We must make an approximation by dividing into "good" and "bad" distributions

$$p(\theta \mid y) \approx \begin{cases} p(\theta \mid y < y^*) \equiv \ell(\theta), & y < y^* \text{ (good)} \\ p(\theta \mid y \geq y^*) \equiv g(\theta), & y \geq y^* \text{ (bad)} \end{cases}$$

$$p(\theta) \approx \ell(\theta) \int_{y<y^*} p(y)\, dy + g(\theta) \int_{y \geq y^*} p(y)\, dy = \gamma l(\theta) + (1-\gamma) g(\theta)$$

Note: See paper on arxiv

What is a Tree-Structured Parzen Estimator (TPE) (Part III)

- The integral, by construction, only cares about the “good” region, so the integral simplifies to

$$\mathrm{EI}(\theta) \propto \frac{\ell(\theta)}{p(\theta)} \int_{-\infty}^{y^*} (y^* - y)\, p(y)\, dy$$

- But the integral is now constant in theta! So we have:

$$\mathrm{EI}(\theta) \propto \frac{\ell(\theta)}{p(\theta)}$$

- Where $p(\theta) = \gamma\, \ell(\theta) + (1 - \gamma)\, g(\theta)$ so we can write:

$$\mathrm{EI}(\theta) \propto \frac{\ell(\theta)}{\gamma\, \ell(\theta) + (1 - \gamma)\, g(\theta)}$$

Note: See paper on arxiv

What is a Tree-Structured Parzen Estimator (TPE) (Part IV)

- But γ is fixed, so

$$\begin{aligned}\arg\max_{\theta} \mathrm{EI}(\theta) &= \arg\max_{\theta} \frac{\ell(\theta)}{\gamma\,\ell(\theta) + (1-\gamma)\,g(\theta)} \\ &= \arg\max_{\theta} \frac{\ell(\theta)/g(\theta)}{\gamma\,\ell(\theta)/g(\theta) + (1-\gamma)} \\ &= \arg\max_{\theta} \frac{\ell(\theta)}{g(\theta)}\end{aligned}$$

- Where the final step is because x/(bx+c) is monotonic in x for x >0 ,let x = l/g

- In other words, we just need to find $\arg\max_{\theta} \frac{\ell(\theta)}{g(\theta)}$ which is doable via algorithm!

Note: See paper on arxiv

What is a Tree-Structured Parzen Estimator (TPE) (Part V)

- Now to define the algorithm, first we observe T (~10) trials randomly:

$$\textbf{Observed trials:} \quad \mathcal{D} = \{(\theta^{(i)}, y^{(i)})\}_{i=1}^{T}$$
$$\textbf{Quantile threshold:} \quad P(y < y^*) = \gamma$$
$$\mathcal{D}_{\text{good}} = \{\theta^{(i)} : y^{(i)} < y^*\}$$
$$\mathcal{D}_{\text{bad}} = \{\theta^{(i)} : y^{(i)} \geq y^*\}$$

- From this data, we want to build:

$$\ell(\theta) \equiv p(\theta \mid y < y^*)$$
$$g(\theta) \equiv p(\theta \mid y \geq y^*)$$

- So we define

$$\theta \equiv (\theta_1, \theta_2, \ldots, \theta_d)$$

- And assume:

$$\textbf{Independence assumption:} \quad p(\theta \mid y < y^*) \approx \prod_{k=1}^{d} p(\theta_k \mid y < y^*)$$

Note: See paper on arxiv

What is a Tree-Structured Parzen Estimator (TPE) (Part VI)

- This allows us to write:

$$\ell(\theta) \approx \prod_{k=1}^{d} \hat{p}_{\text{good}}(\theta_k)$$

$$g(\theta) \approx \prod_{k=1}^{d} \hat{p}_{\text{bad}}(\theta_k)$$

- We can fit to our samples to get a continuous distribution spaces for each param

$$\hat{p}_{\text{good}}(\theta_k) \leftarrow \text{fit to } \{\theta_k^{(i)} : \theta^{(i)} \in \mathcal{D}_{\text{good}}\}$$

$$\hat{p}_{\text{bad}}(\theta_k) \leftarrow \text{fit to } \{\theta_k^{(i)} : \theta^{(i)} \in \mathcal{D}_{\text{bad}}\}$$

- Then we sample from the “good” spaces for M candidates index by m

$$\tilde{\theta}_{m,k} \sim \hat{p}_{\text{good}}(\theta_k), \qquad k = 1, \ldots, d$$

- And finally, choose our next theta, add it to the data set, and repeat

$$\theta^{(t)} = \arg\max_{\tilde{\theta}_m} \frac{\ell(\tilde{\theta}_m)}{g(\tilde{\theta}_m)} = \prod_{k=1}^{d} \frac{\hat{p}_{\text{good}}(\tilde{\theta}_{m,k})}{\hat{p}_{\text{bad}}(\tilde{\theta}_{m,k})}$$

Note: See paper on arxiv

What is a Tree-Structured Parzen Estimator (TPE) (Part VII)

- How do we obtain our fits from the data?
- Uses Kernel Density Estimation (KDE)
  - The actual fit is done when determining each “gaussian kernel” (or sigma)
  - Likelihood-optimal is expensive ~O(n^2d)
    - n = # samples
    - d = # hyperparameters
  - Optuna uses the “heuristic” version, which requires choosing some value for c.

Given samples: $\{\theta_k^{(i)}\}_{i=1}^n, \qquad \theta_k^{(i)} \in \mathbb{R}$

Kernel density estimate:

$$\hat{p}(\theta_k) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{K}\left(\frac{\theta_k - \theta_k^{(i)}}{h_k}\right)$$

Gaussian kernel:

$$\mathcal{K}(u) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{u^2}{2}\right)$$

Bandwidth (likelihood-optimal):

$$h_k^\star = \arg\max_h \sum_{i=1}^{n} \log \hat{p}_{-i}(\theta_k^{(i)} \mid h)$$

Bandwidth (heuristic used in practice):

$$h_k = c \cdot \mathrm{std}\left(\{\theta_k^{(i)}\}_{i=1}^n\right), \qquad c > 0$$

Example Graph Transformer Layer

Example: Graph Transformer Layer (Neighborhood Self-Attention)

For each node v in the graph:

$$
\begin{aligned}
q_v &= W_Q h_v, \\
k_u &= W_K h_u, \\
v_u &= W_V h_u, \\
&\forall u \in \mathcal{N}(v)
\end{aligned}
$$

$$
\alpha_{vu} = \mathrm{softmax}_{u \in \mathcal{N}(v)} \left(\frac{q_v^\top k_u}{\sqrt{d}} + f(e_{vu}) \right)
$$

$$
h'_v = \sum_{u \in \mathcal{N}(v)} \alpha_{vu} \, v_u
$$

where:

h_v is the node embedding of node v,

$\mathcal{N}(v)$ is the set of neighbors of node v,

e_{vu} are edge features between nodes v and u,

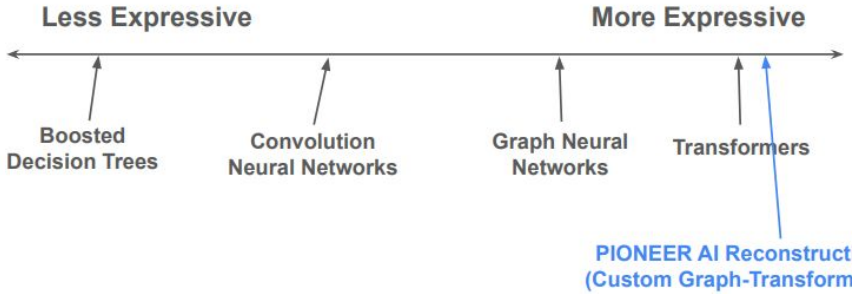
$f(\cdot)$ is a learned function producing an attention bias from edge features.

This update is applied independently to all nodes and repeated at each layer.

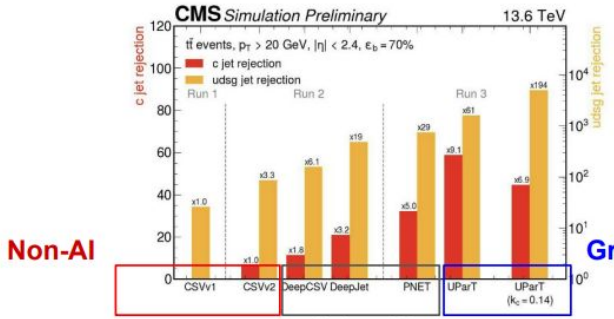
Why use Graph Transformers?

- These are among the most expressive modern ML models
  - Can solve a wider class of problems than most models
- [Fairly easy to construct with torch geometric]
- Attention based transformers allow models to learn how event information is related, as opposed to having to be more explicitly “told” in a traditional GNN approach

AI-based reconstruction:



- The UParT clustering model, built on a graph-transformer architecture, is the most successful CMS jet reconstruction algorithm to date (10.22323/1.476.0992)



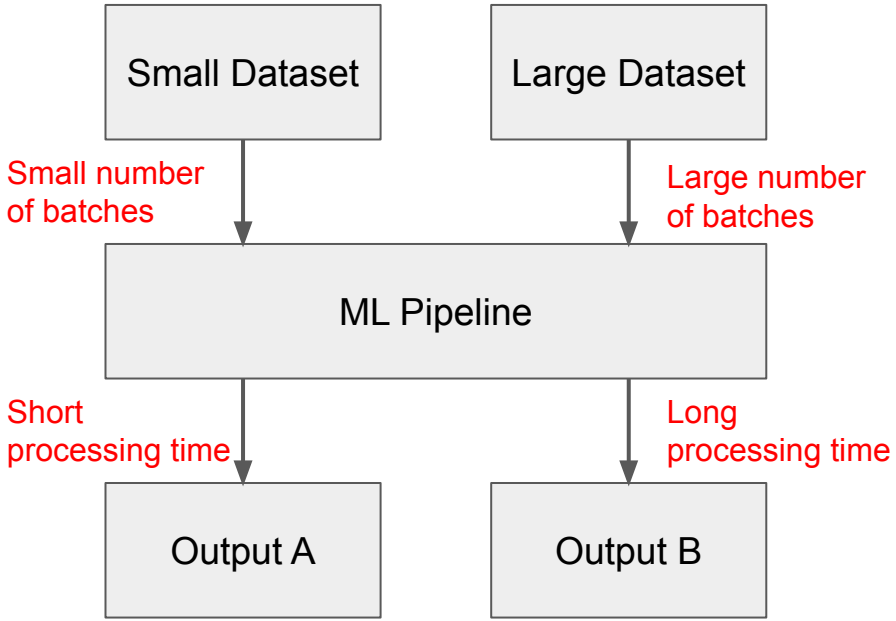
Breaking down “Scalability”

I choose to break down scalability into three categories:

1. Data
    a. ML pipeline does not change behavior as data set grows
        i. Still scales in execution time
2. Compute
    a. ML pipeline does not change behavior as compute power is changed
        i. Still scales in execution time
        ii. Ex. Pipeline runs on developer’s laptop or a CPU/GPU cluster
3. Codebase
    a. ML pipeline does not (greatly) change behavior as complexity grows
        i. New models, stages, or data products require additional code, not (major) code rewrites
        ii. Iteration speed does not (greatly) degrade with system size (i.e. keep things modular!)

What Do We Mean by “Scaling” (Data)

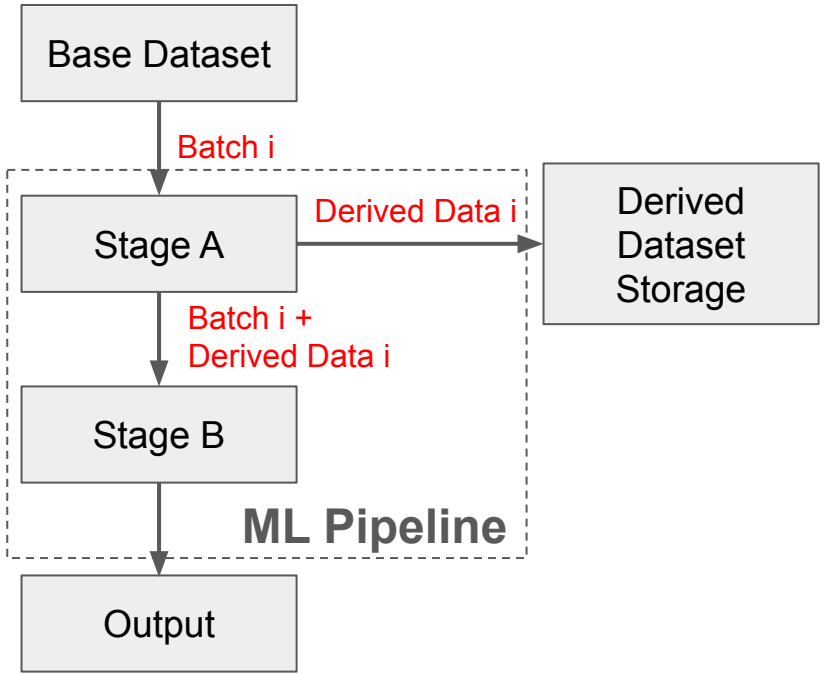
- **Data volume should be able to grow without changing how the pipeline behaves**
- Adding the following should not cause pipeline behavior changes:
  - More events
  - More derived data products
    - predictions, masks, regressions, etc.
  - More passes over the same data
  - Larger event representations
- Implications:
  - Memory usage must not grow with dataset size



Ideal Behavior for Different Sized Datasets

Techniques to Ensure Scalability (Data)

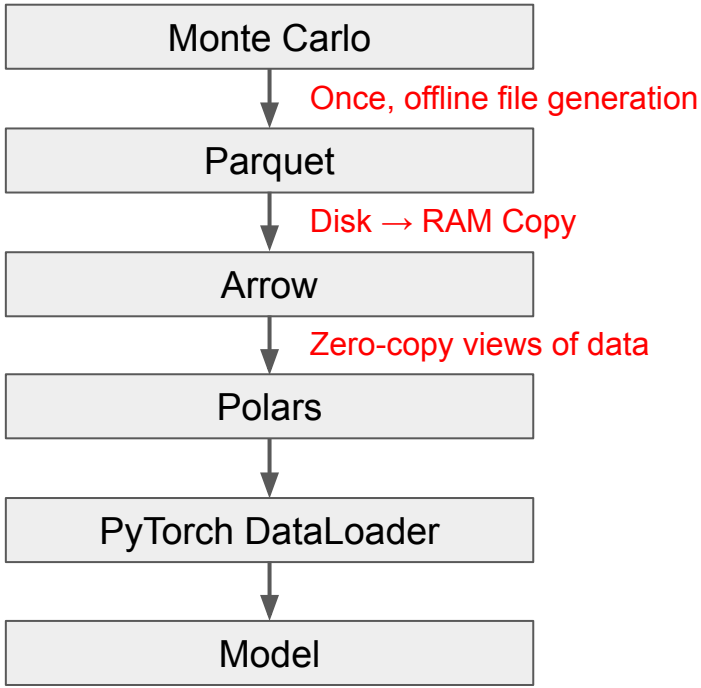
- Stream data in bounded batches
  - RAM usage should not scale with data set size
  - Allows RAM usage to be “tunable”
- Base data should be immutable
  - No modifications or extensions to ML pipeline input data files
- Separate derived data products
  - Predictions, masks, and regressions are produced as independent datasets
- Reference data by IDs, not by object
  - When passing data modules, use file paths or map IDs not in memory collections
  - Similar to why we use pointers in C++



Simple Example Pipeline Including Derived Datasets

Technologies for Scalability (Data)

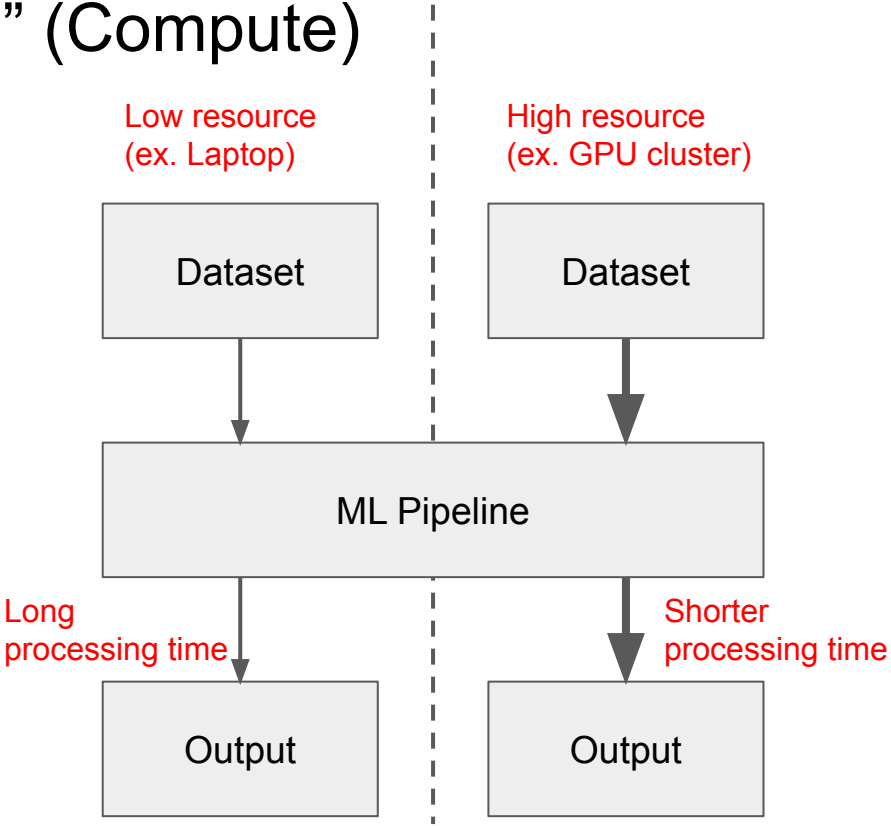
- Apache Parquet (columnar, on disk)
  - ML-native working datasets, append-only, shardable (aka “batchable”)
  - Can key rows by event ID, enabling batch-scoped joins for derived data products
- Apache Arrow
  - Single copy from disk into RAM, then zero-copy views all the way to Torch
- Polars
  - Dataset shaping: filtering, joins, feature derivation
  - Lazy, columnar execution on Arrow-backed data
  - Prepares batchable views for PyTorch ingestion
- PyTorch tensors
  - Execution format for models
- Pytorch DataLoader
  - Handles batching, shuffling, parallel loading, prefetching, enforcing memory bounds



Flow of Data From Monte Carlo to a Model in the ML pipeline

What Do We Mean by “Scaling” (Compute)

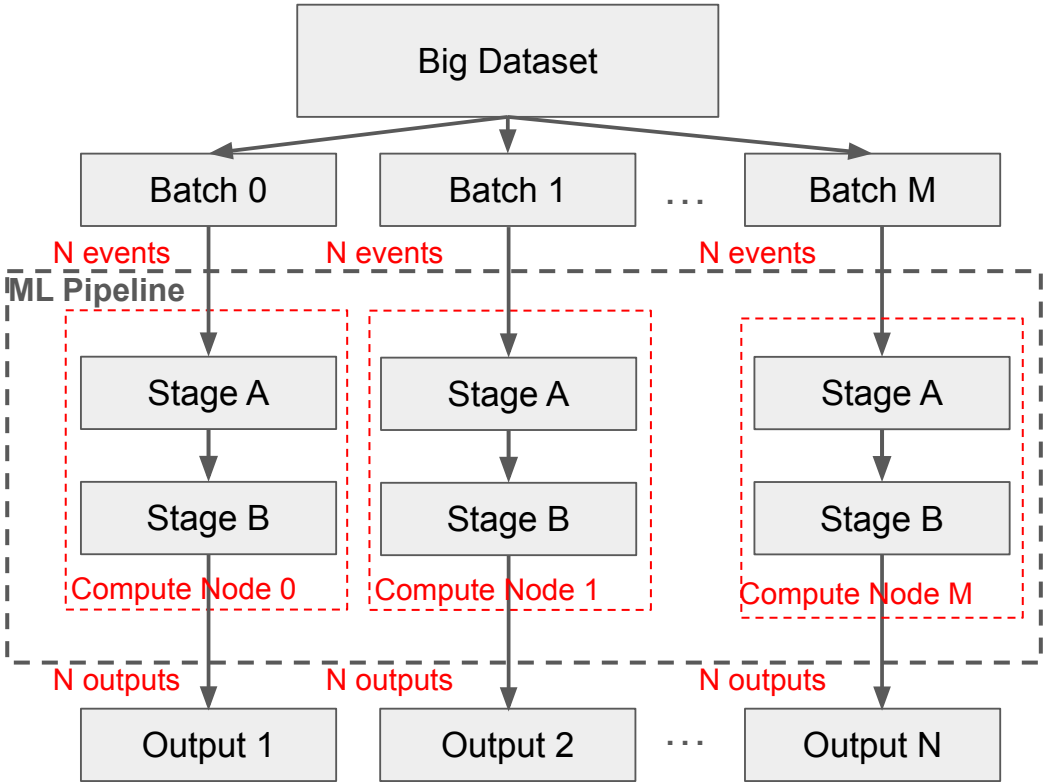
- **Computation resources should be able to grow without changing how the pipeline behaves**
- Adding the following should not cause pipeline behavior changes:
  - Number of GPUs
  - Number of CPU cores/threads
  - Node count
  - Memory Capacity
- Implications:
  - Algorithms must be agnostic to resources
    - Caveat: PyTorch and other frameworks may change their behavior for different devices/device counts for efficiency



Ideal Behavior for Different Amounts of Computing Resources

Techniques to Ensure Scalability (Compute)

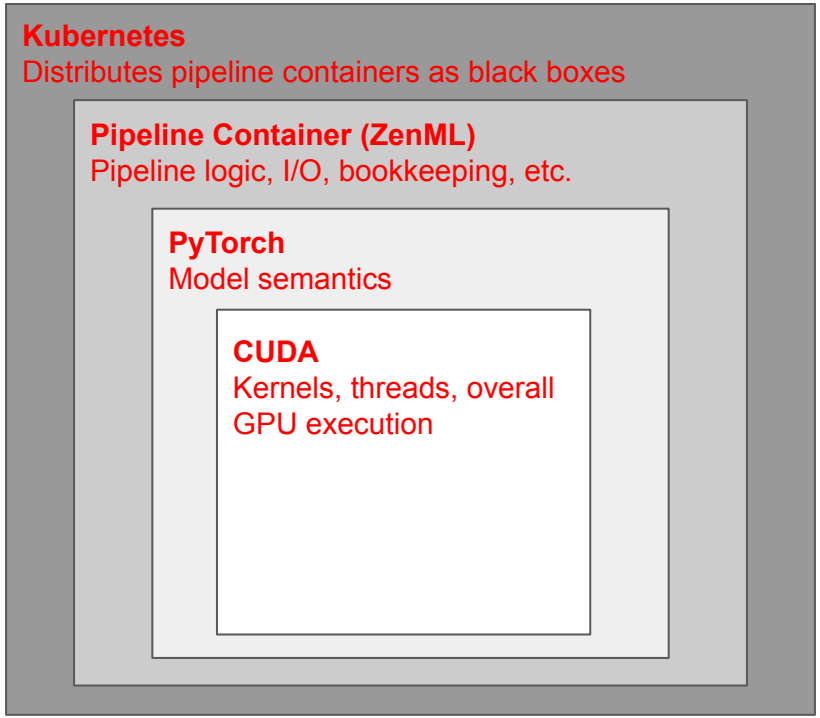
- Make algorithms resource-agnostic
  - No logic branches based on GPU count, core count, node count, or memory size, etc.
- Make algorithms easily parallelizable
  - Decompose computation into independent, composable units
  - Avoid global state dependencies in pipeline stages
  - Avoid “synchronization points”; i.e. points where models must make inferences on whole datasets



Parallelized View of Simplified Pipeline

Technologies for Scalability (Compute)

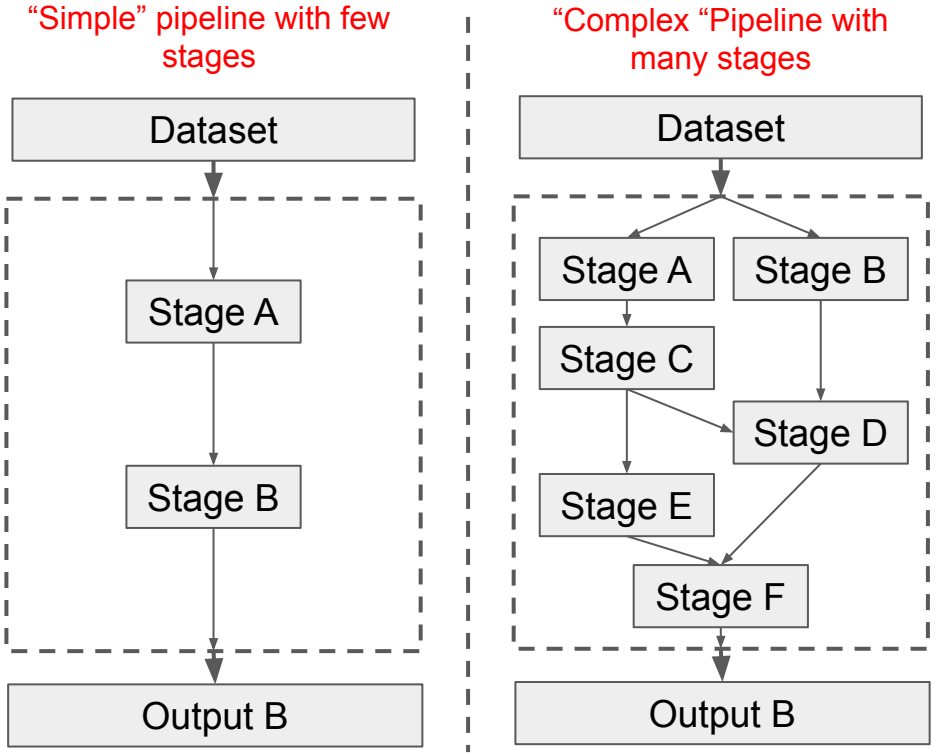
- PyTorch
  - Standard framework for modern ML models
  - Resource-agnostic execution model
- CUDA
  - Operates purely at the level of memory, kernels, and execution, independent of algorithm or pipeline semantics
  - Provides an API for launching and coordinating large numbers of parallel threads on GPUs
- Kubernetes
  - Schedules identical pipeline executions onto available compute nodes
  - Scales how many pipelines run concurrently, not what they do
  - Handles retry logic and resource limits



Simplified “Scope” Of Technologies
Outer Technologies Manage Inner Technologies

What Do We Mean by “Scaling” (Codebase)

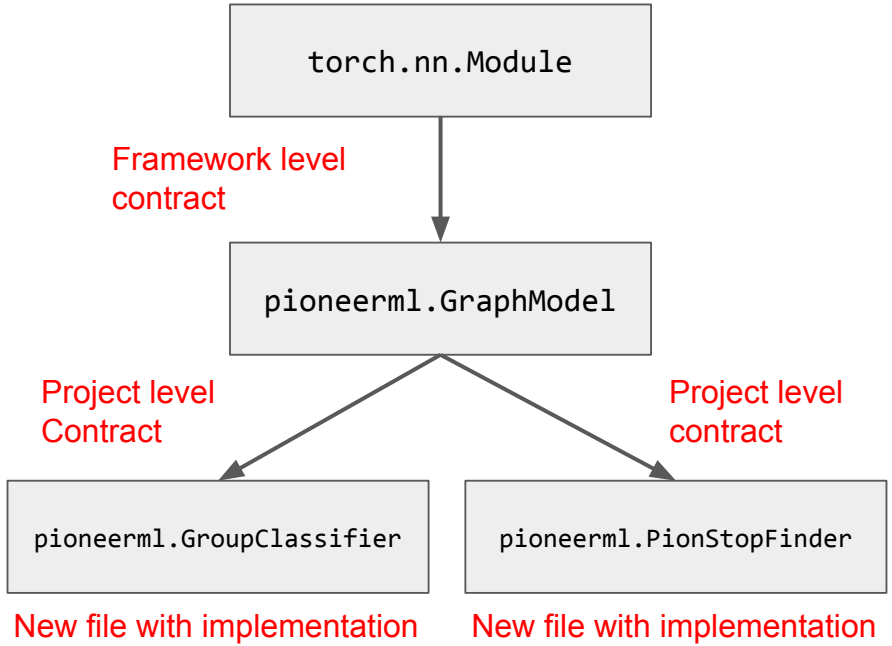
- **Codebase complexity should be able to grow without changing how the system behaves**
- Adding the following should not cause system wide behavioral changes:
  - More pipeline stages
  - More models / algorithms
  - More configuration options
- Implications:
  - New functionality should be added by extensions, not modification
  - System behavior should be locally understood (modularity)
  - Existing code should not require global refactoring to evolve



The “Global” Structure Should Not Change As Complexity Increases

Techniques to Ensure Scalability (Codebase)

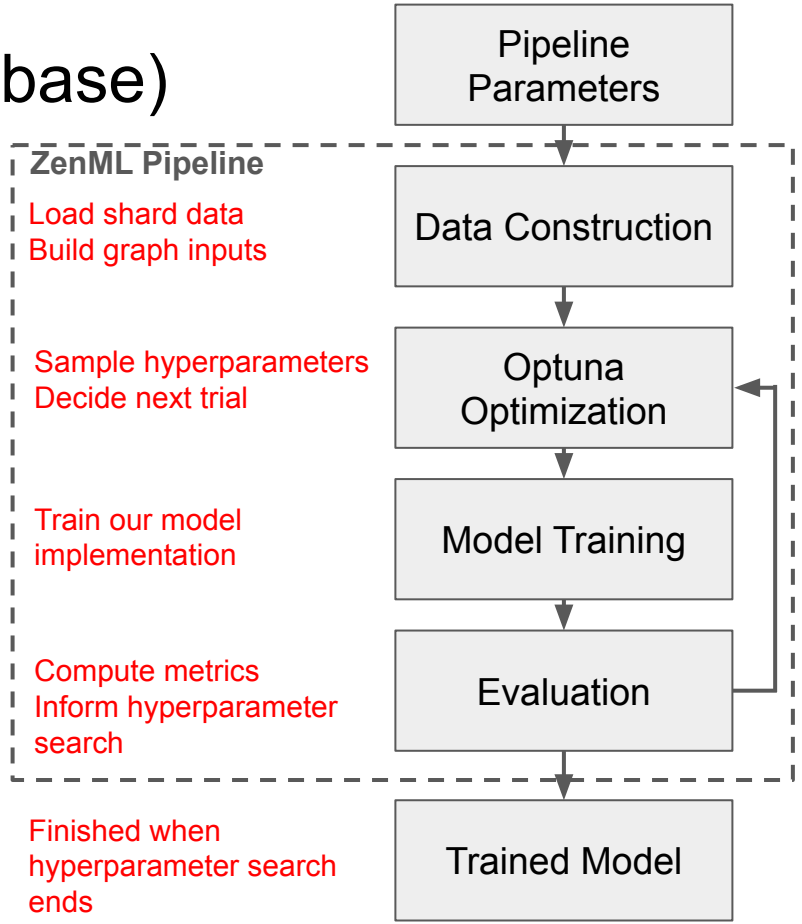
- Use abstraction where appropriate
  - Define stable base interfaces / abstract classes
  - Add new functionality by extending, not rewriting
- Avoid global coupling
  - Modules depend only on explicit inputs
  - Each module manages its own immutable local state
- Isolate responsibilities to achieve modularity
  - Each module has a single, well-defined responsibility
  - Changes remain local to the owning module



Simple Example of Abstraction For Adding New Models

Technologies for Scalability (Codebase)

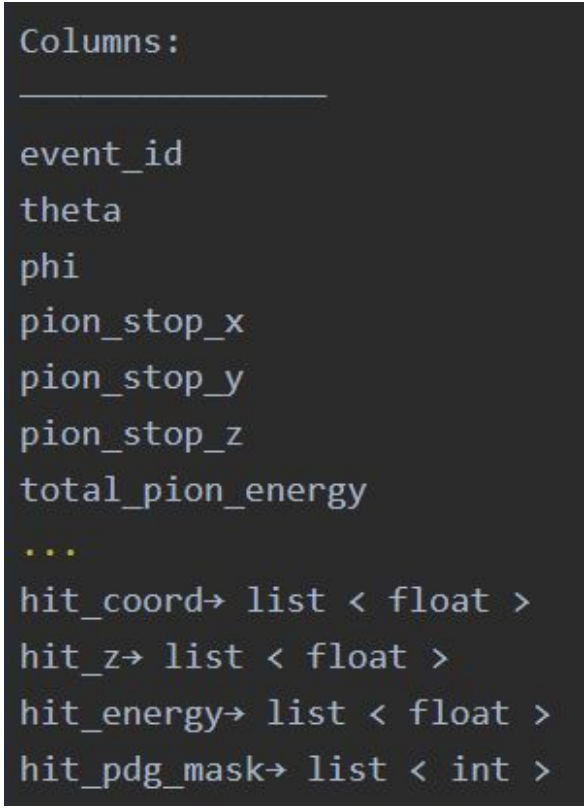
- PyTorch
  - Provides a standard base abstraction classes for many model types (ex. `nn.Module`)
  - Enforces a consistent model interface (ex. `forward` method)
- ZenML
  - Encodes pipelines as composable, declarative units and orchestrates execution
  - Allows pipelines to grow by adding or reordering steps; easy to add new pipelines
  - Manages pipeline state, artifacts, and execution metadata outside user code
- Optuna
  - Isolates hyperparameter search code
  - Enables experimentation without modifying core implementations
  - Really a package for black box searching, by default uses Tree-Structured Parzen Estimator (TPE)



Simplified Example Pipeline for Training Models

What is Apache Parquet?

- **Apache Parquet is a columnar, on-disk data format that is widely used in ML workloads**
- What parquet does
  - Columnar storage → read only the columns (features) your model needs
    - Allows one to efficiently assign a subset of columns as inputs and another subset of columns as targets
  - Supports nested and variable-length fields
  - Data schema is embedded in the file, not inferred by code (ex. Not like numpy, where code defines `dtype` parameter)
- Why this matters for scalability
  - Efficient I/O for large datasets through compression and encoding
  - New models using different subsets of the data becomes trivial



Example time_groups.parquet file structure

What is Batching?

- **Batching means processing a fixed-size subset of events at a time, rather than the full dataset**
- What batching does
  - Groups individual events into batches of size N
  - Each batch is processed independently
  - Batches are discarded after use
- Why this matters for scalability
  - Memory usage is bounded by batch size (for single batch process)
  - Dataset size does not affect RAM usage
  - Enables streaming over arbitrarily large datasets

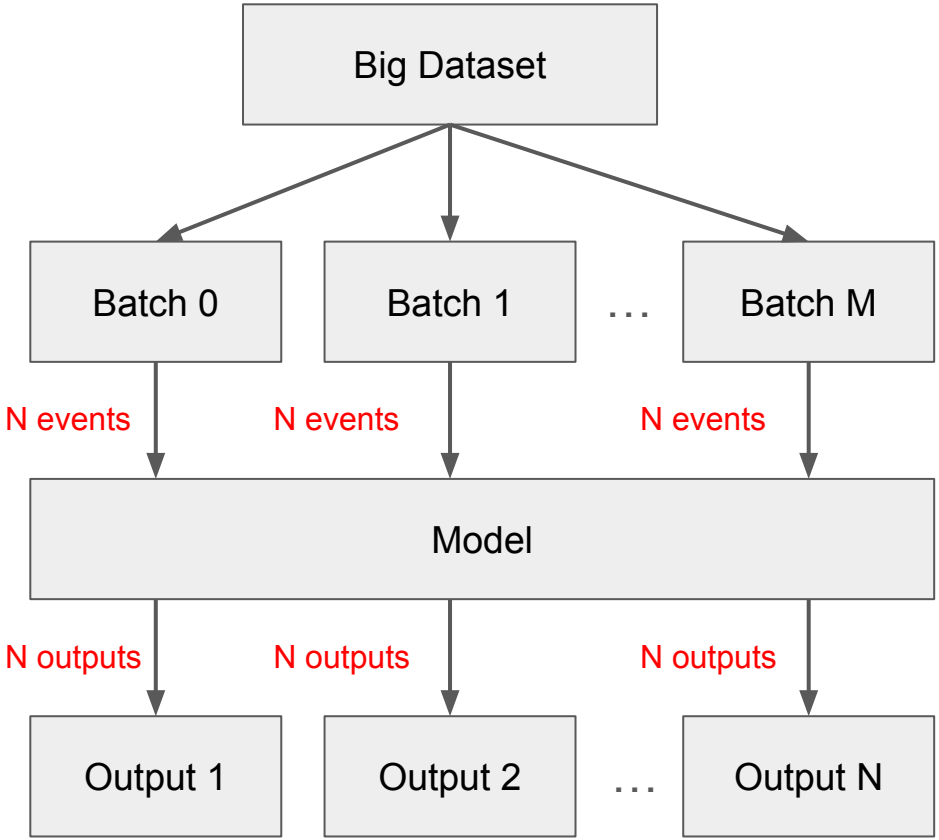
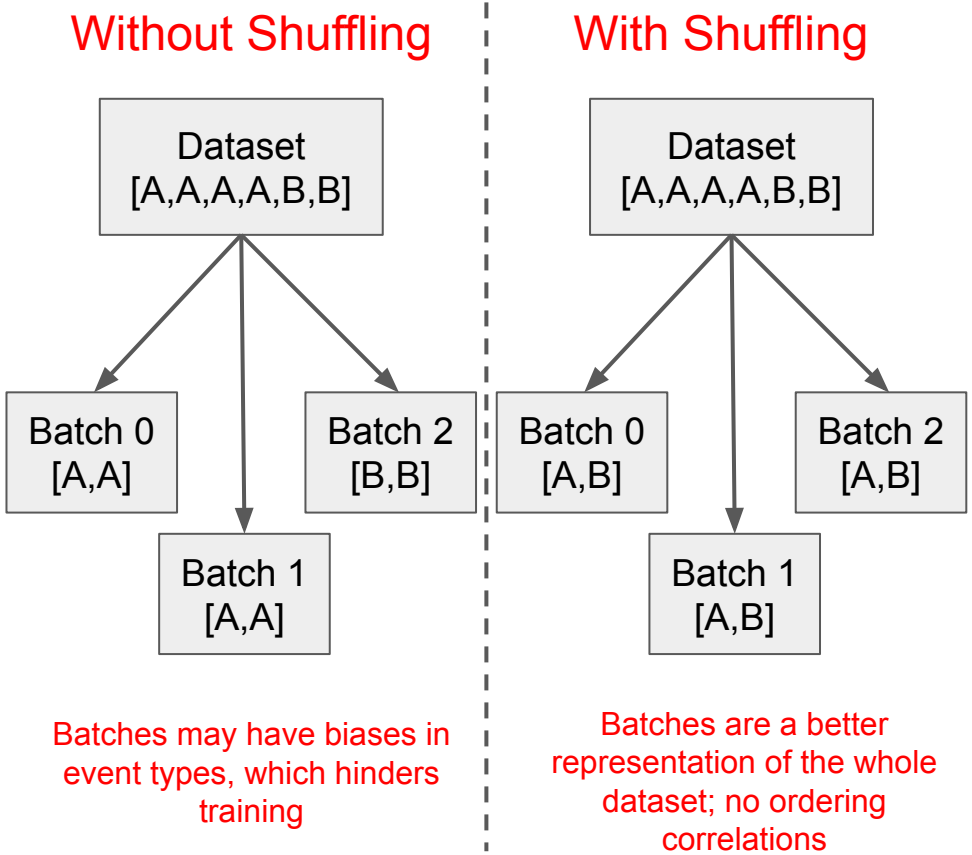


Diagram that Shows how Batching Splits Data

What is Shuffling?

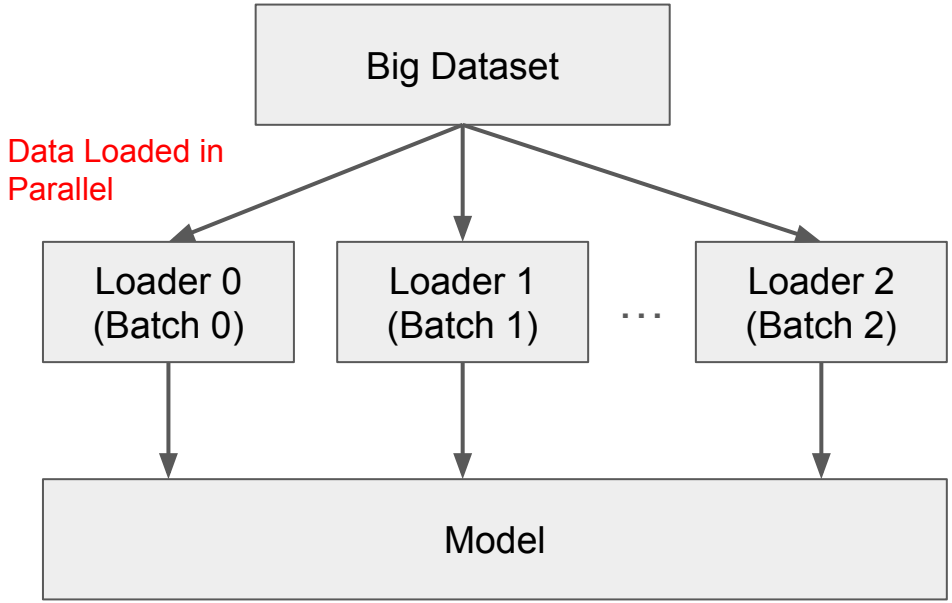
- **Shuffling changes the order in which events are seen, without changing the data itself**
- What shuffling does
  - Randomizes event order before forming batches
  - Ensures batches contain a mix of events
  - Changes between epochs (or passes over the data)
- Why this matters for scalability
  - Prevents bias from data ordering
  - Improves statistical independence between batches
  - Allows repeated passes over large datasets without correlation artifacts



Simplified Example of Shuffling vs. Not Shuffling a Dataset with Distinct Event Types A and B

What is Parallel Loading?

- **Parallel loading means loading multiple batches concurrently, using multiple workers**
- What parallel loading does
  - Multiple workers read and prepare batches simultaneously
  - The model always has a batch ready to process
  - Data loading is decoupled from model execution
- Why this matters for scalability
  - Helps prevent the model from waiting on disk I/O
  - Improves hardware utilization (especially GPUs)

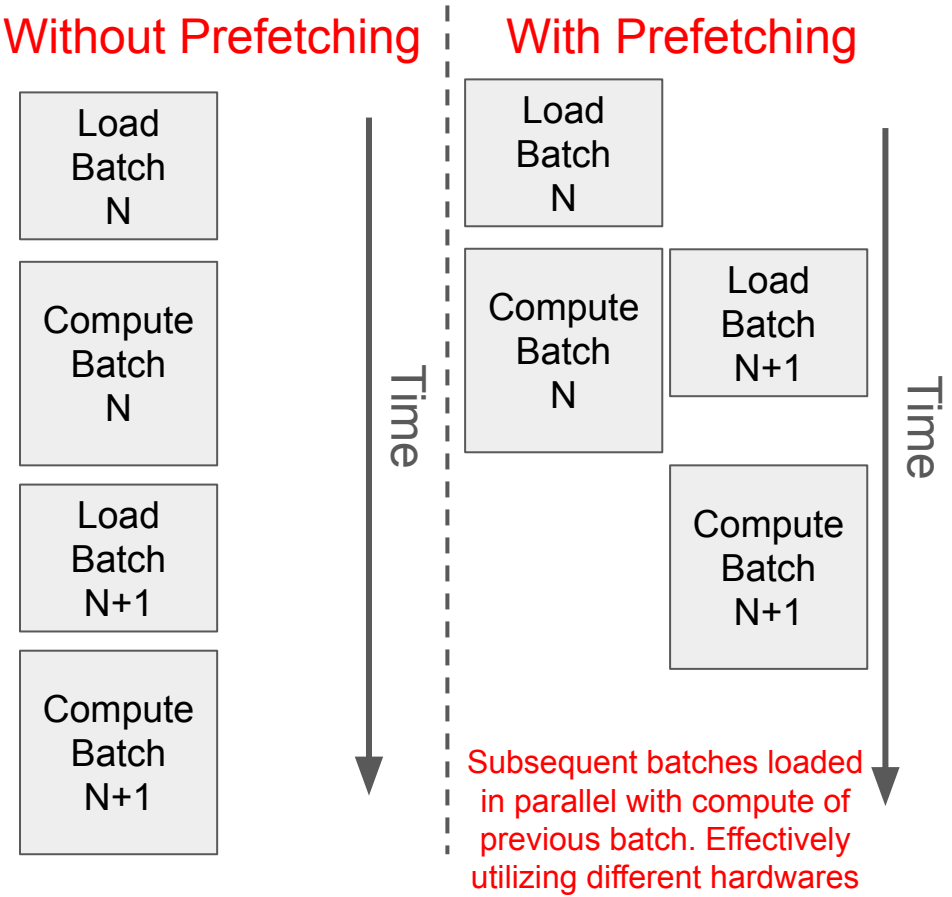


Note: Parallel loading does **not** necessarily mean parallel model execution

Parallel Data Loading Diagram

What is Prefetching?

- **Prefetching means loading future batches while the current batch is being processed**
- What prefetching does
  - While the model computes on batch N the next batch (N+1) is loaded in the background
  - When computation finishes, the next batch is already ready
- Why this matters for scalability
  - Helps prevent the model from waiting on disk I/O



Simplified Prefetching Example Diagram

What is Enforcing Memory Bounds?

- **Enforcing memory bounds means placing a hard limit on how much data can be in memory at once**
- What enforcing memory bounds does
  - Maximum in-flight data size is fixed
  - Batch creation is throttled when memory is full
  - Makes memory usage predictable and stable
- Why this matters for scalability
  - Dataset size does not affect RAM usage
  - Enables streaming over arbitrarily large datasets

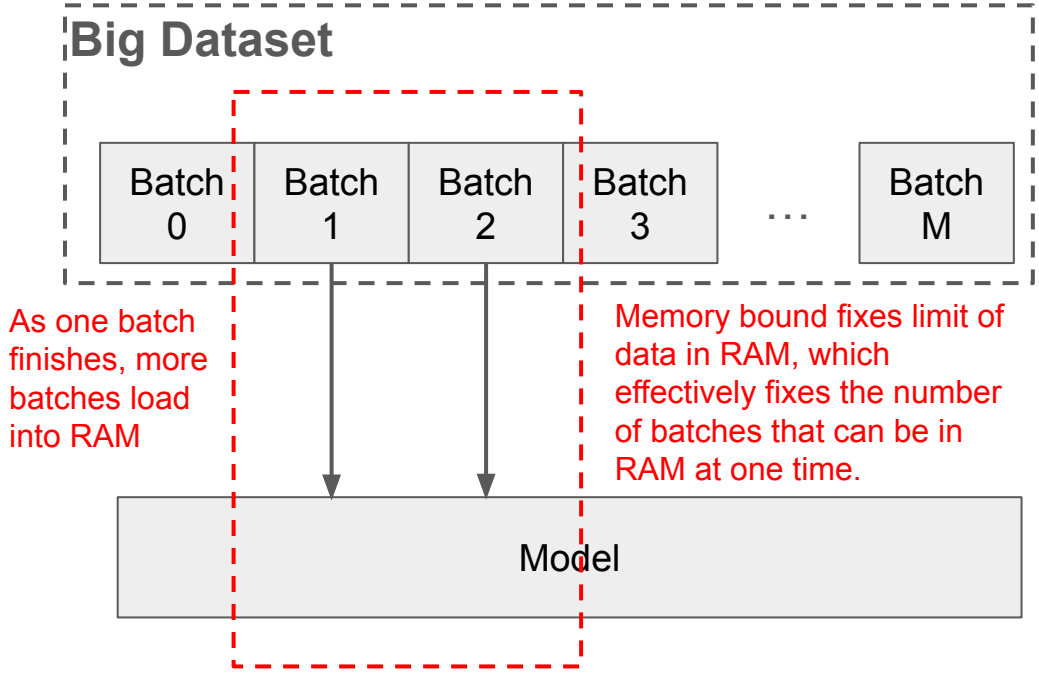
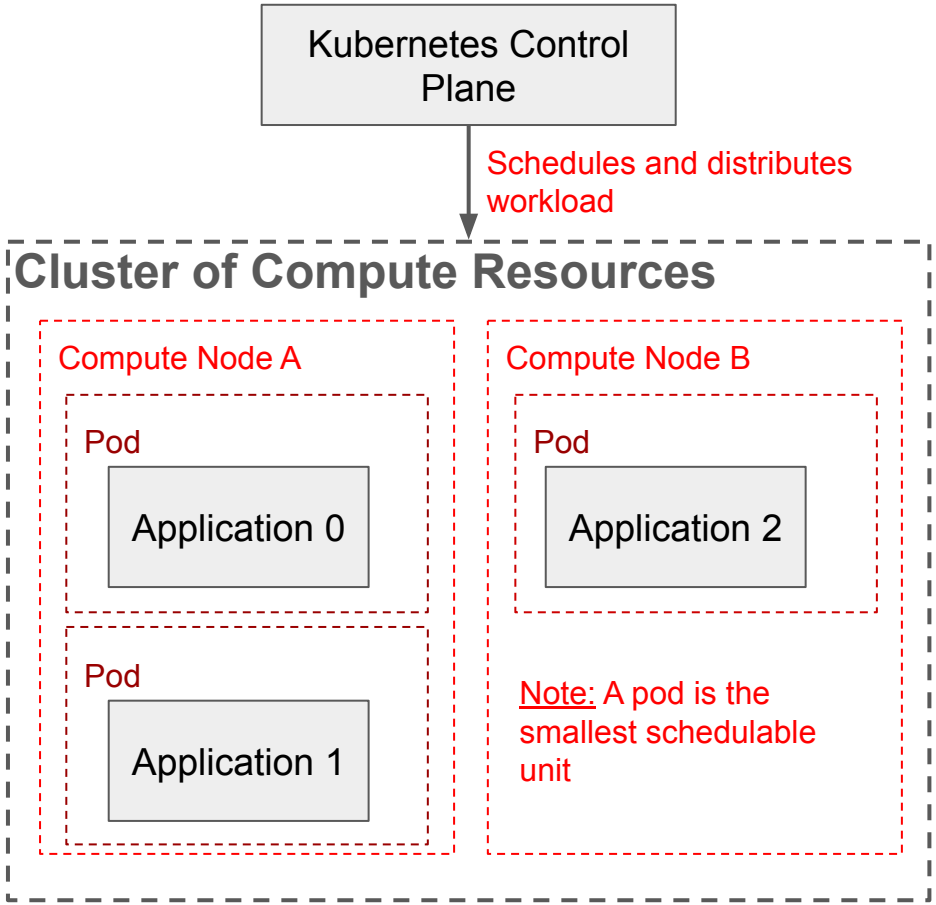


Diagram Showing How Memory Bounds Enforce a Limited Number of Batches Loaded in RAM

What is Kubernetes (K8s)?

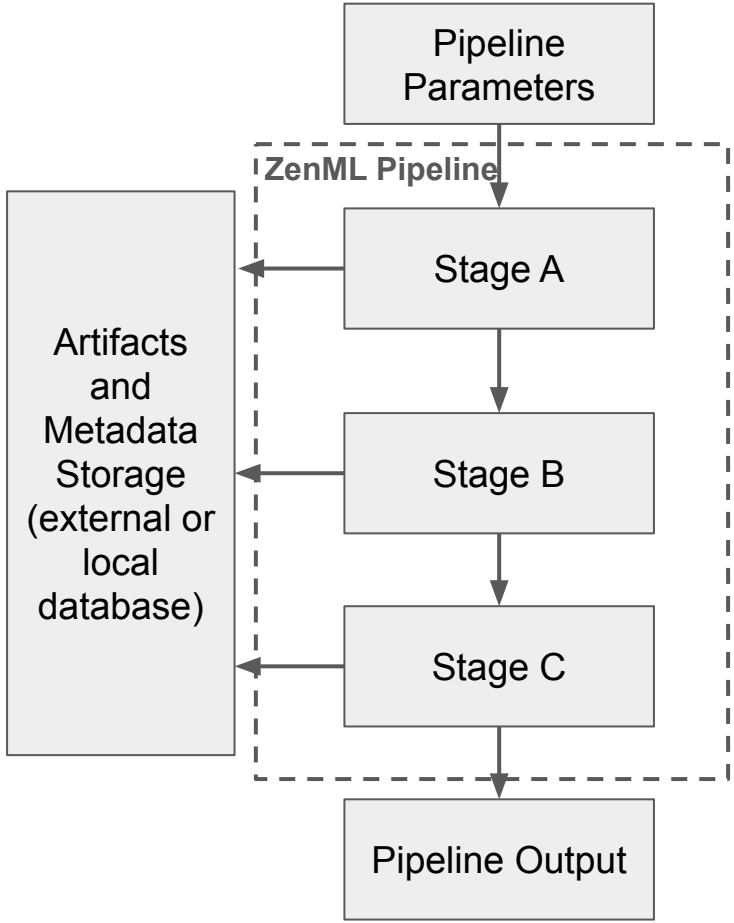
- **A system for running and managing containerized workloads across shared compute resources**
- What Kubernetes does
  - Schedules containers onto available compute nodes
  - Enforces resource limits (CPU, GPU, memory) per container
  - Isolates workloads from one another
  - Handles restarts and retries on failure
- Why this matters for scalability
  - Enables concurrent execution of many independent workloads
  - Prevents resource contention between workloads



Simplified Diagram of How Kubernetes Orchestrates Applications Across a Compute Cluster

What is ZenML?

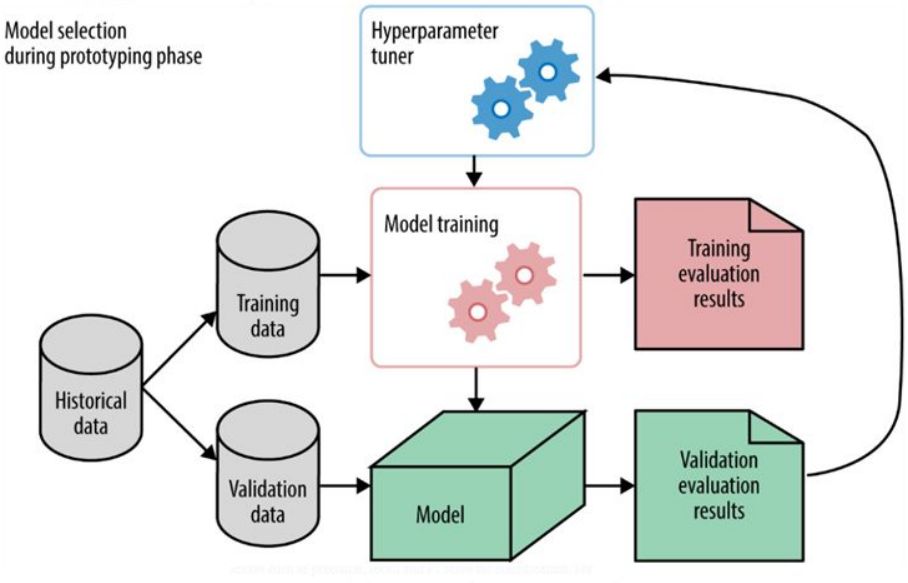
- **A framework for defining, orchestrating, and executing machine-learning pipelines with explicit steps, artifacts, and metadata**
- What ZenML does
  - Encodes workflows as composable, declarative pipelines
  - Orchestrates execution order and dependencies
  - Integrates with execution backends (local, containers, clusters) without changing user code
- Why this matters for scalability
  - Pipelines grow by adding or rearranging stages, not rewriting logic
  - Enables reproducibility, versioning, and parallel development
  - State and artifacts are managed outside user code



Example ZenML pipeline diagram

What is Optuna?

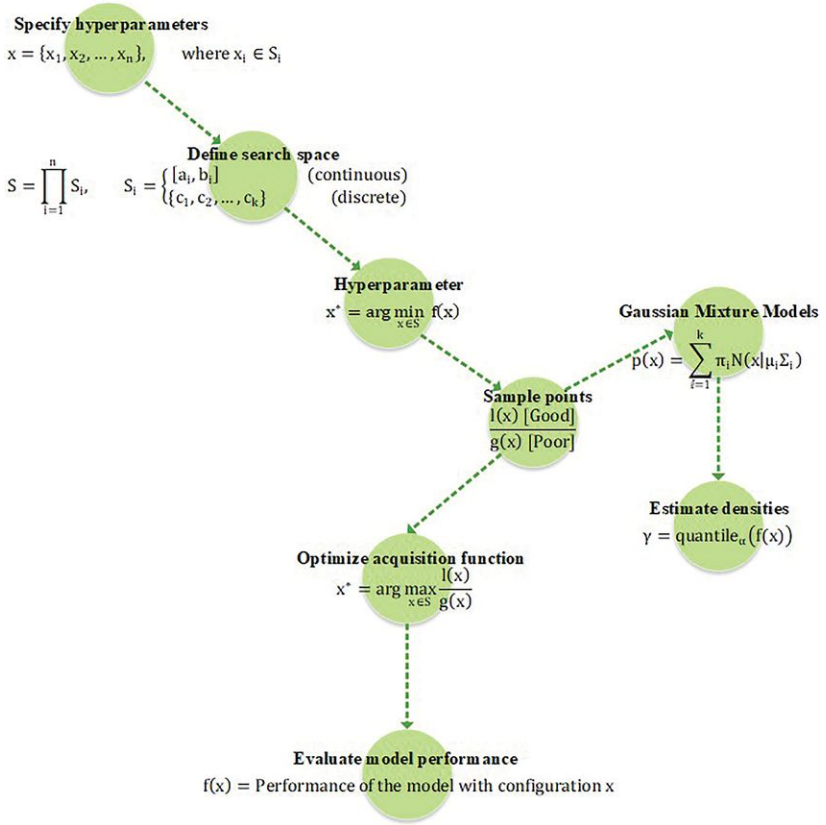
- Many hyper parameters means manual tuning is bad
  - Too slow
  - Suboptimal tuning means you spend more resources (time, computing power, etc.) training unused models
- Optuna is a python package that solves this problem
  - Framework for optimization black box objective functions
  - Technically not an ML package, but rather a package that supports many optimized sampling strategies



Example Optuna workflow diagram

How Optuna Works

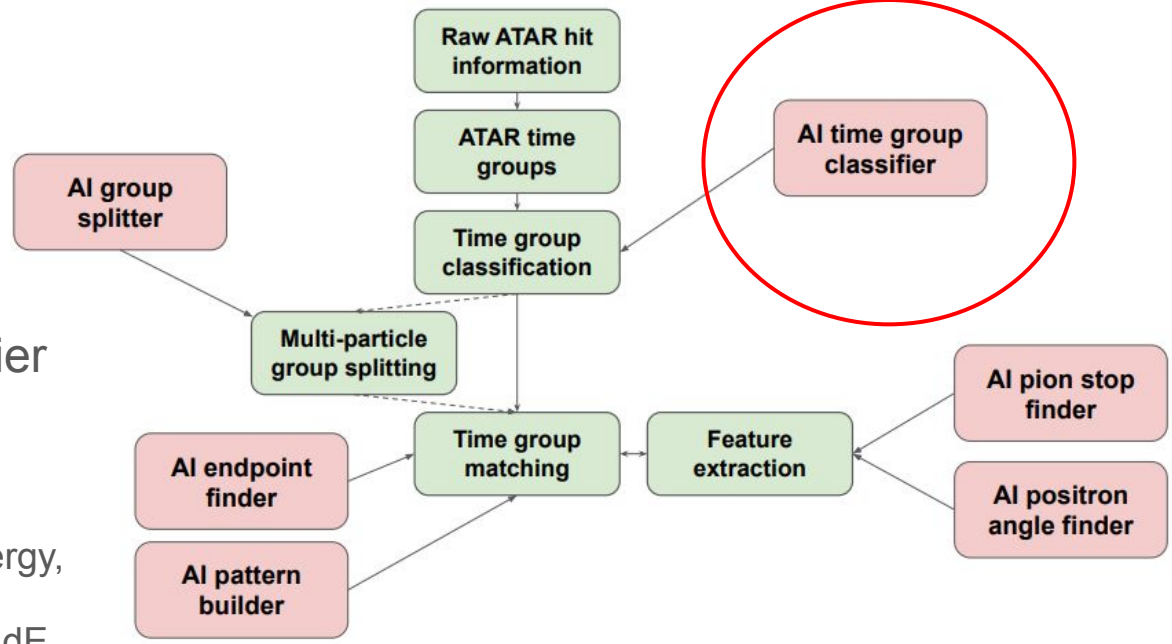
- Optuna supports many sampling algorithms, examples:
  - Grid search
  - Random search
  - Gaussian process-based Bayesian optimization
- For single object functions, the default for Optuna is Tree-Structured Parzen Estimator (TPE)

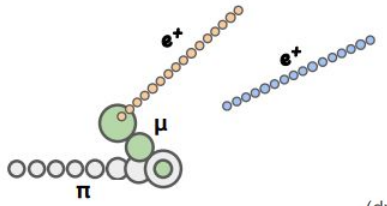


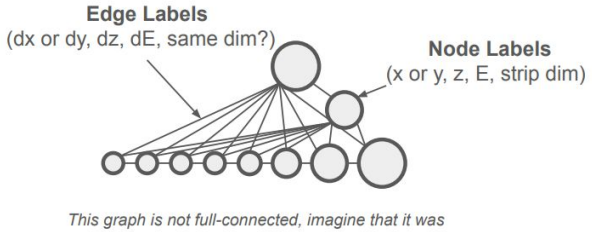
TPE flow diagram (in a nutshell)

Time Group Classifier

- I'll mostly be talking about Omar's "time group classifier model"
- Input:
  - Time grouped hit graph
    - Nodes: [x (or y), z, energy, view, group_energy]
    - Edges: [dx (or dy), dz, dE, same_view]
  - Groups split by time (could potentially contain multiple particle types)
- Output
  - Class labels: [muon, pion, mip]
- Much better explained in [Omar's presentation]

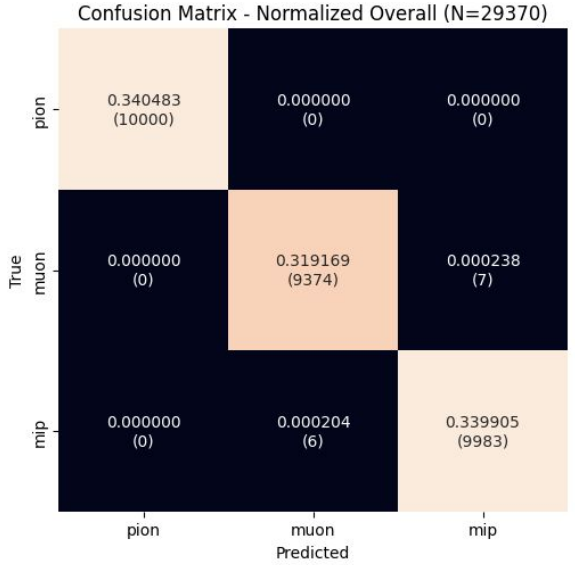


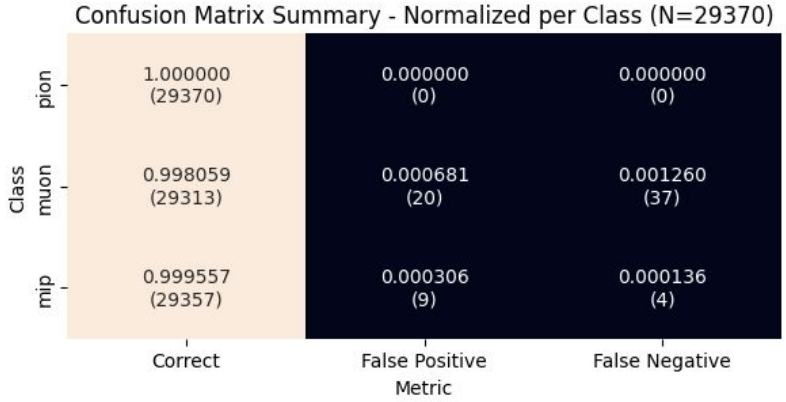




Performance of Model

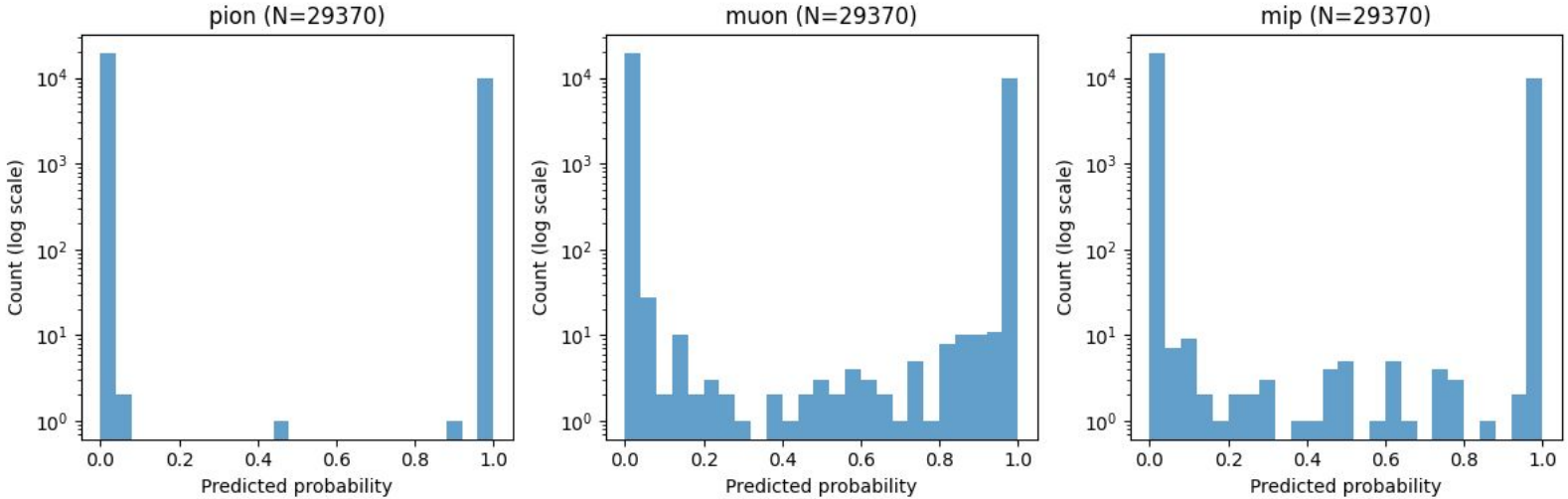
- Closely matches performance of Omar's work
- Differences from Omar's work
  - Used my machine to train
  - Did a hyperparameter search using Optuna
  - Incorporated into ZenML framework for creating pipelines in python
- Unsure how this compares to the traditional reco values(?)
- Unsure exact parameters in Omar's data set





Performance of Model

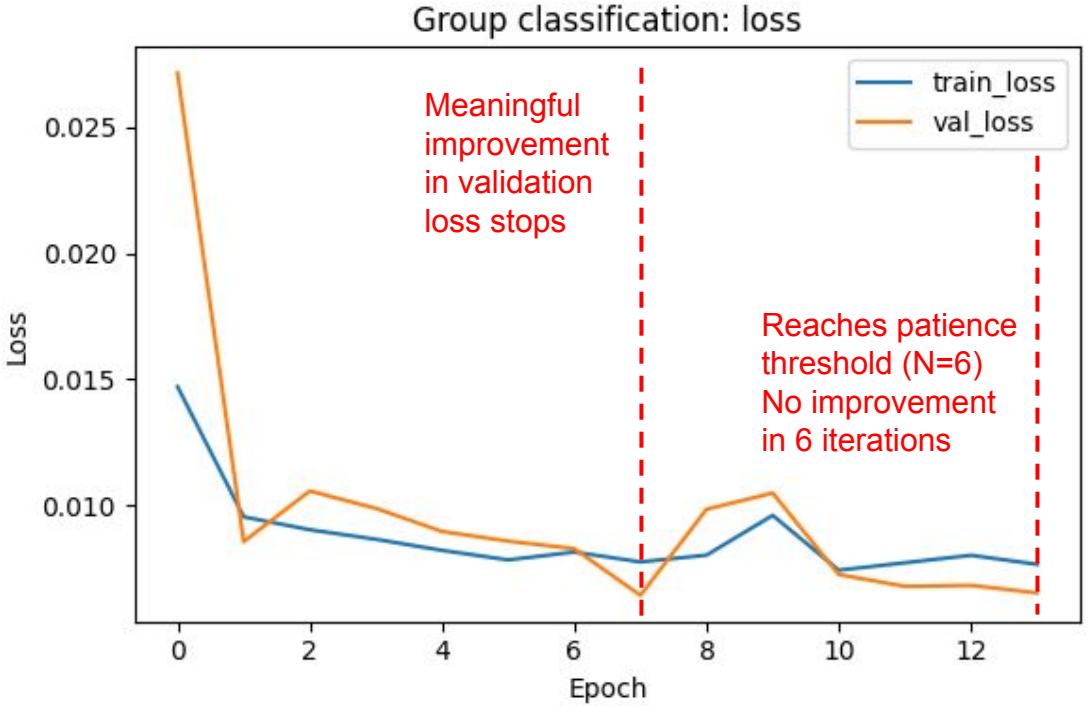
- Histograms below shows the models “sureness” of each class
- Want to see large peaks at 0 and 1
  - 0 → this is definitely not this class
  - 1 → this is definitely this class
- The muon groups are the biggest struggle



Preventing Overtraining (Early Stopping)

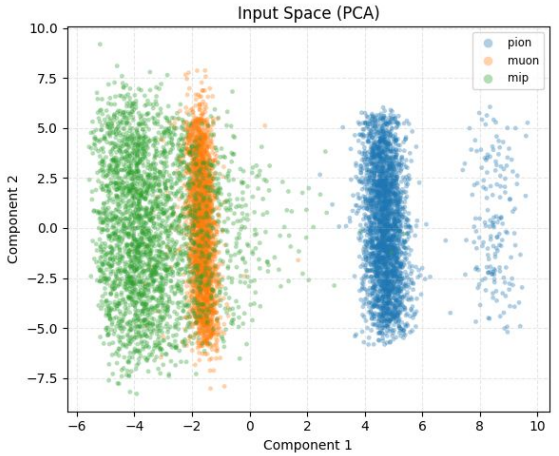
- Can set early stopping when training loss curves
  - If no meaningful (> δ) improvement within N epochs, stop training
  - N := “patience” usually set to ~5% of expected training epochs
- Potential improvements:
  - Loss curve smoothing
  - “Noise” estimations, only continue training if smoothed improvement > noise of previous iterations
- See documentation

*NOTE: This particular model is overtrained because N set to 6 (too high)



Preventing Overtraining (PCA)

- Need a way to visualize many dimensions
- Principal Component Analysis (PCA) good candidate for this
- Compare clusters in embedding space vs. input space
  - Input space shows data driven groupings
  - Embedding space shows learned groupings
  - # of groupings should match!

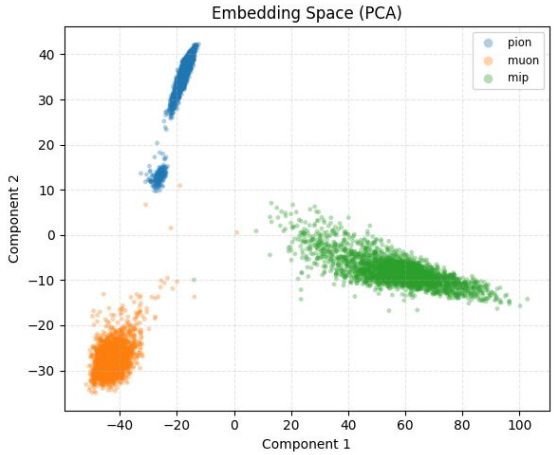


Average vector of all nodes in graph: [coord, z_pos, energy, view, group_energy]

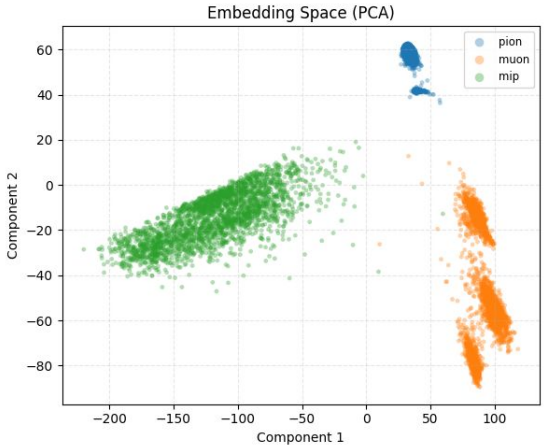
5D → 2D

Well-trained

Over-trained



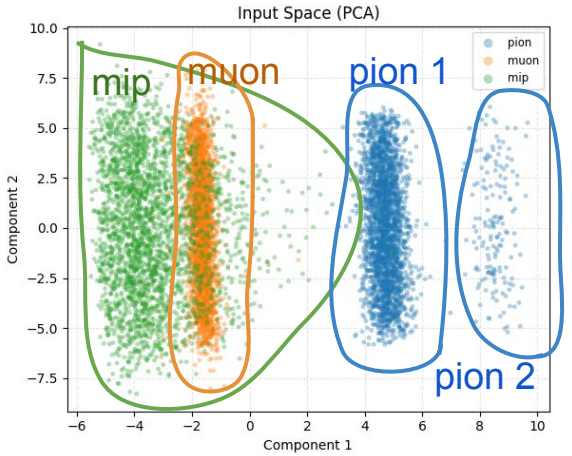
512D → 2D



512D → 2D

Preventing Overtraining (PCA)

- Need a way to visualize many dimensions
- Principal Component Analysis (PCA) good candidate for this
  - Choose to project down to d = 2 dimensions this way
- Compare clusters in embedding space vs. input space
  - Input space shows data driven groupings
  - Embedding space shows learned groupings
  - # of groupings should match!

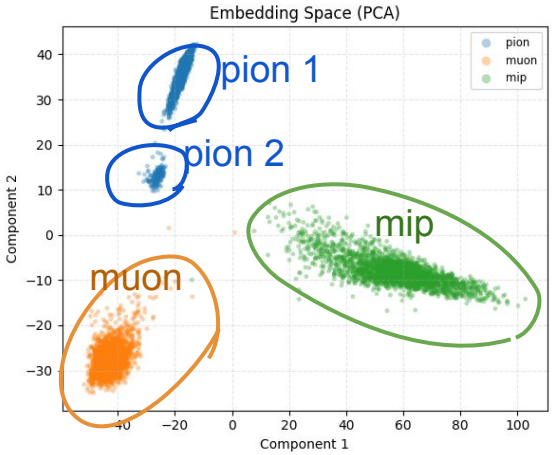


Average vector of all nodes in graph: [coord, z_pos, energy, view, group_energy]

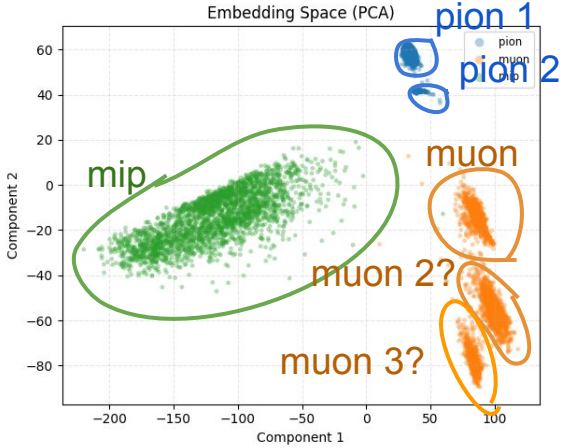
5D → 2D

Well-trained

Over-trained



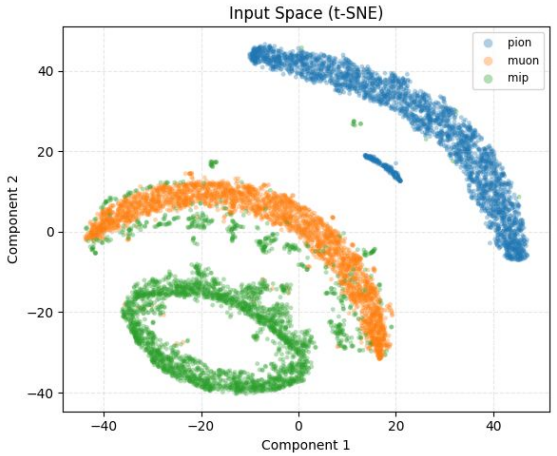
512D → 2D



512D → 2D

Preventing Overtraining (t-SNE)

- Problem with PCA
  - Global linear may not preserve local neighborhoods
    - May not show groups!
- t-distributed stochastic neighbor embedding (t-SNE) is designed to preserve local clusters
  - Maps N dims → d dims
    - We choose d = 2 for visualization ease
- Same ideas as PCA
  - compare input space and embedding space

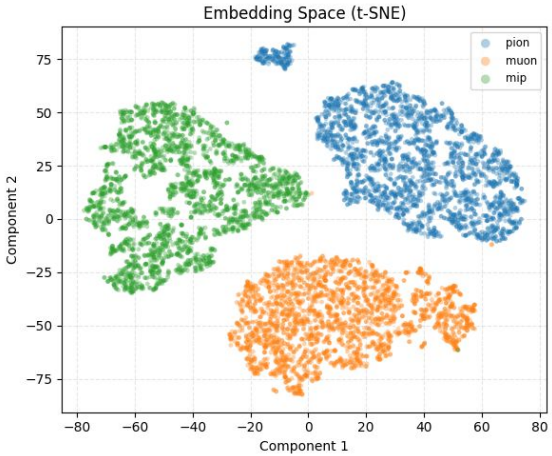


Average vector of all nodes in graph: [coord, z_pos, energy, view, group_energy]

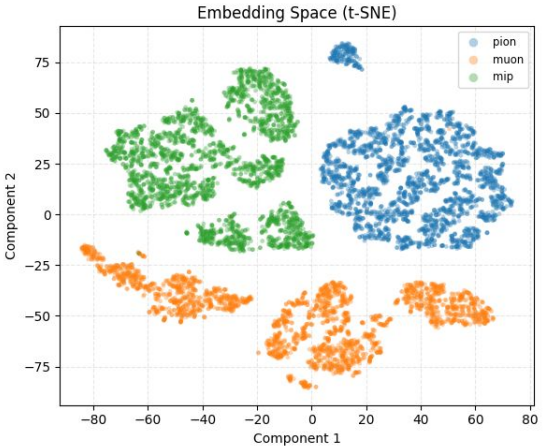
5D → 2D

Well-trained

Over-trained



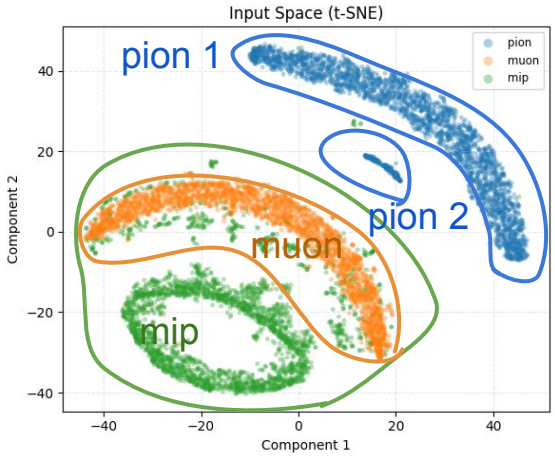
512D → 2D



512D → 2D

Preventing Overtraining (t-SNE)

- Problem with PCA
  - Global linear may not preserve local neighborhoods
    - May not show groups!
- [t-distributed stochastic neighbor embedding](#) (t-SNE) is designed to preserve local clusters
  - Maps N dims → d dims
    - We choose d = 2 for visualization ease
- Same ideas as PCA
  - compare input space and embedding space

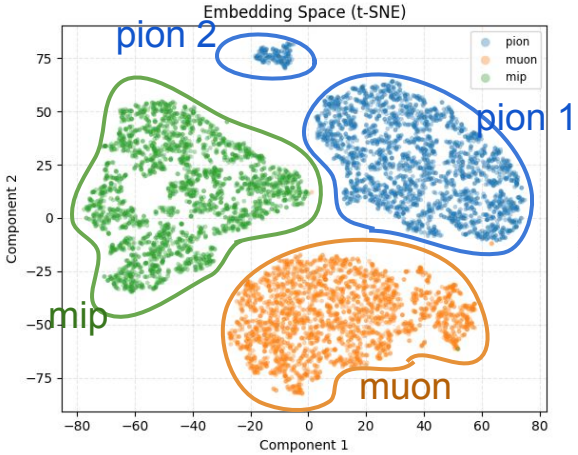


Average vector of all nodes in graph: [coord, z_pos, energy, view, group_energy]

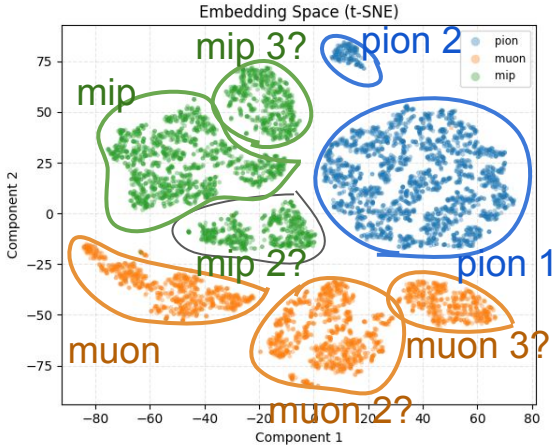
5D → 2D

Well-trained

Over-trained



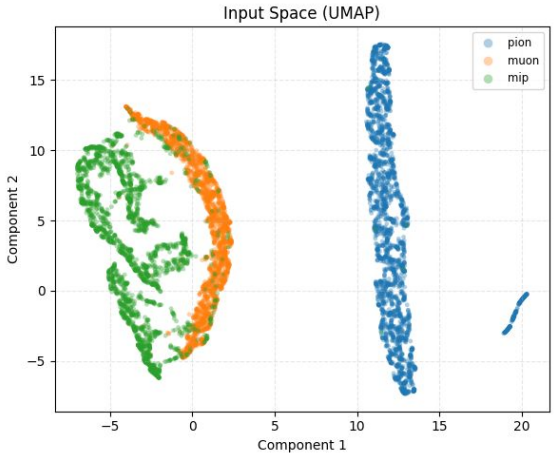
512D → 2D



512D → 2D

Preventing Overtraining (UMAP)

- Problem with t-SNE
  - Depends on a “Perplexity”
    - Parameter, ~how many neighbors a point can have
  - May artificially split or group clusters
- Uniform Manifold Approximation and Projection (UMAP) is designed to preserve the underlying manifold structure
  - Tries to preserve both local neighborhoods *and* their global relationships
  - Maps N dims → d dims
    - We choose d = 2 for visualization ease
- Computational expensive
  - Use as a “tie breaker” if PCA and t-SNE disagree

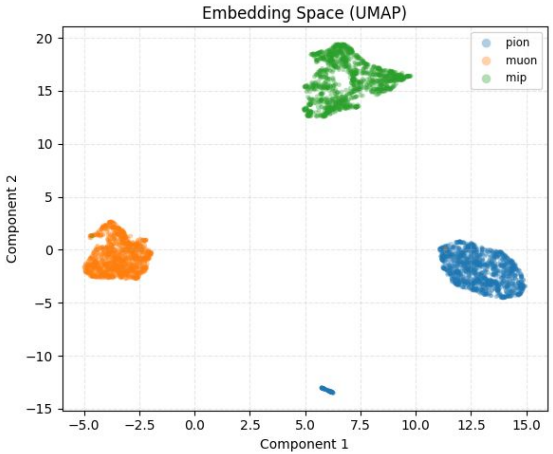


Average vector of all nodes in graph: [coord, z_pos, energy, view, group_energy]

5D → 2D

Well-trained

Over-trained



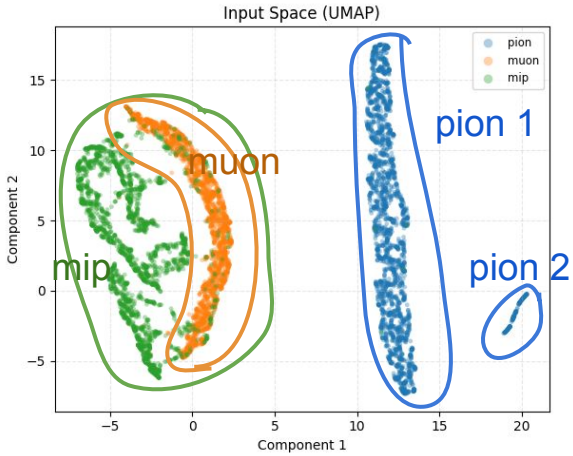
512D → 2D

Sorry!
I don’t have an example for this!

512D → 2D

Preventing Overtraining (UMAP)

- Problem with t-SNE
  - Depends on a “Perplexity”
    - Parameter, ~how many neighbors a point can have
  - May artificially split or group clusters
- Uniform Manifold Approximation and Projection (UMAP) is designed to preserve the underlying manifold structure
  - Tries to preserve both local neighborhoods *and* their global relationships
  - Maps N dims → d dims
    - We choose d = 2 for visualization ease
- Computational expensive
  - Use as a “tie breaker” if PCA and t-SNE disagree

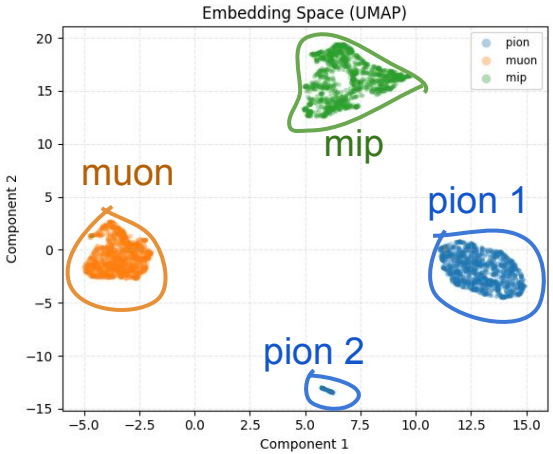


Average vector of all nodes in graph: [coord, z_pos, energy, view, group_energy]

5D → 2D

Well-trained

Over-trained



512D → 2D

Sorry!
I don’t have an example for this!

512D → 2D

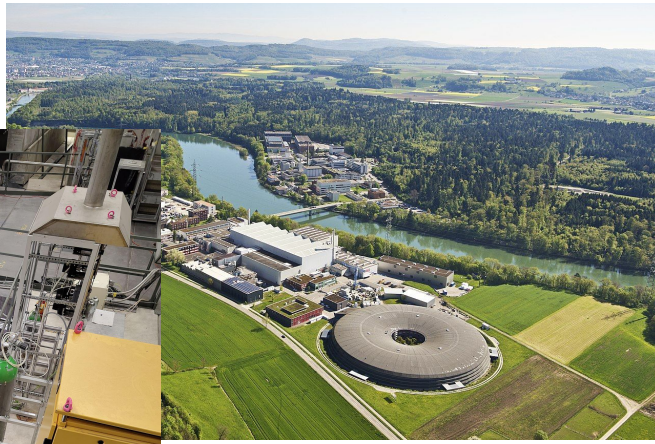
j.carlton@uky.edu

PSI Data

2025 PSI Test Beam

Overview

- PIONEER LYSO Tapered Array calorimeter test
  - August 18 - September 1, 2025
- Made measurements using LYSO scintillator crystals
  - 2023 Rectilinear
    - Rectangle shaped segment, does not match our calorimeter geometry
  - 2025 Tapered Array
    - “Soccer ball” segment shaped, true segment of our calorimeter geometry
- Proved tapered LYSO can be used for PIONEER’s calorimeter by measuring:
  - **Energy resolution**
  - Timing resolution
  - Spatial resolution

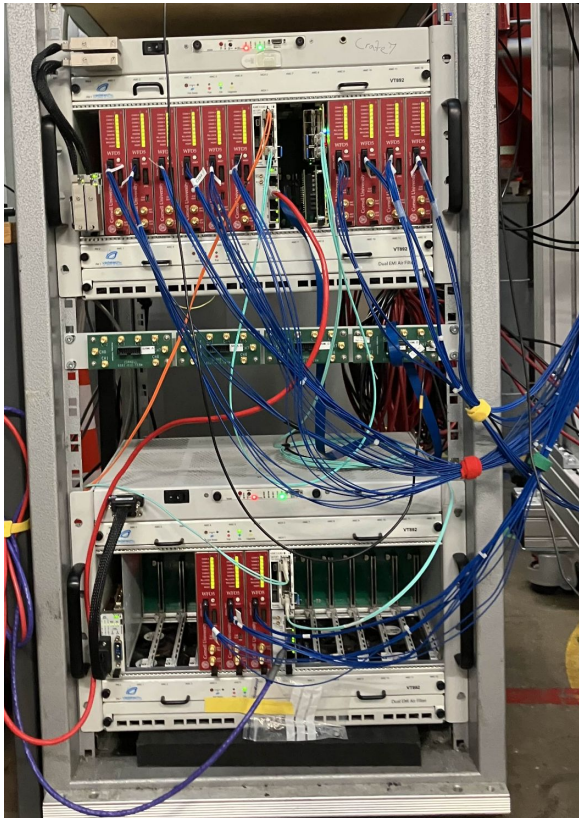


g-2 DAQ modified - Usage for LYSO testbeam

- Iteration on previous LYSO test beam

Run	Crates	Channels Used	Event Rate (Hz)	Data Rate (MB/s)
2023 PSI LYSO Test Beam	1	~50	~400	~30
2025 PSI LYSO Test Beam	2	~60	~2500	~200

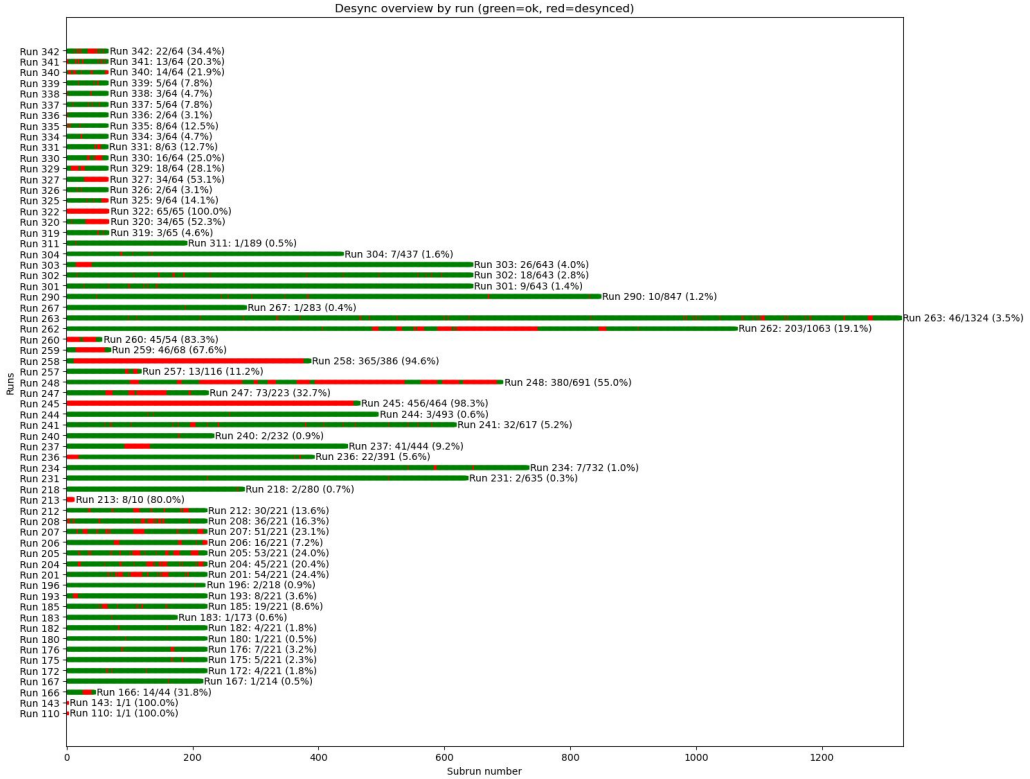
- Rate limited by HDD write speed ~250 MB/s
  - For small scale experiments, could be solved by writing directly to SSDs (>1GB/s write speed)
    - Limited by SSD space (≤8TB for consumer electronics)
- Desyncs between crate data caused (mostly) recoverable issues
  - ~2% of events across all data runs are desynced



Two crate setup used at PSI LYSO Testbeam

g-2 DAQ modified - Desync Issues

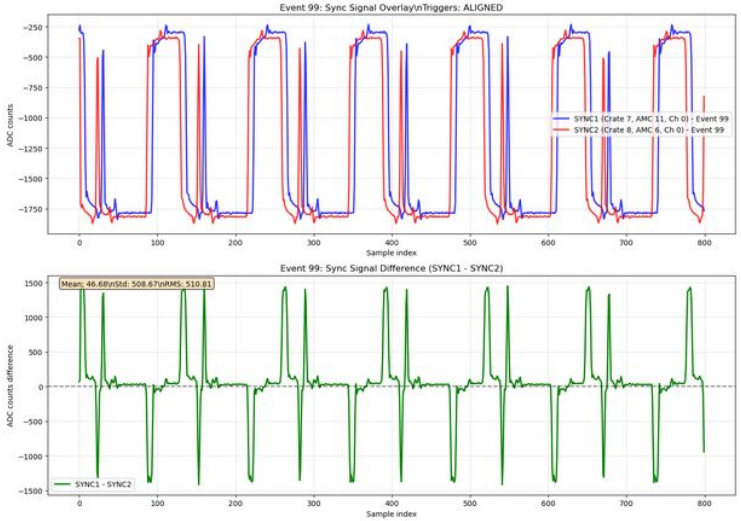
- Desync issues should be fixed
  - Replicated the issue on the UKy teststand
    - Caused by internal ring buffer overflows (“GPU_BUFFER”)
  - Fixed software bug preventing trigger throttling from working properly
    - Trigger throttles apply back pressure to prevent ring buffer overflows
- For the testbeam we used a “band-aid” solution
  - Event builder stops run on desync detected
  - Run is restarted immediately
  - Gives the crates time to recover
  - Downsides: splits data runs, likely not scalable for many crates



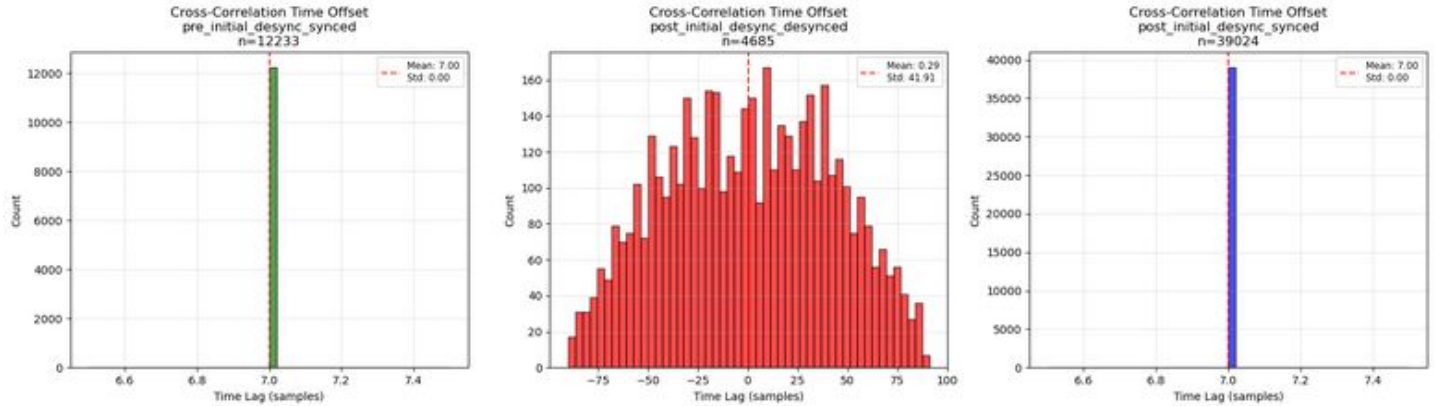
All desynced subruns in 2025 PSI LYSO Testbeam

Study on desynced data

- Elog available
- Clock signal fanned out out into both crates
- Showed the DAQ “recovered” from desyncs properly



Clock signal fanned out in both crates for synced event

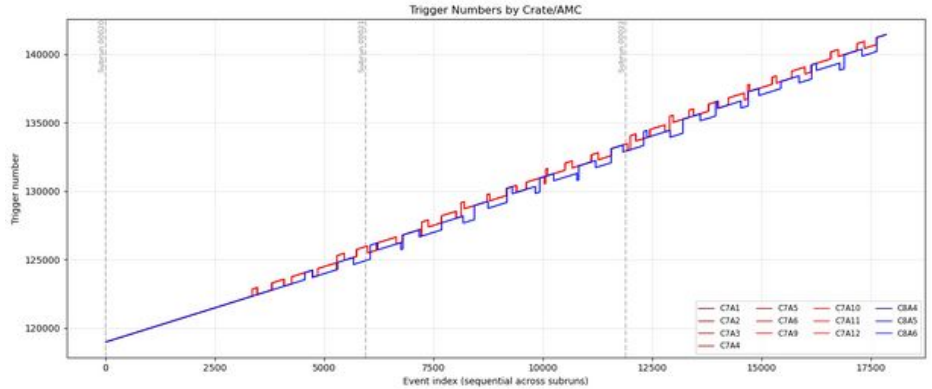


Cross correlated of time offset between both crate clock signals

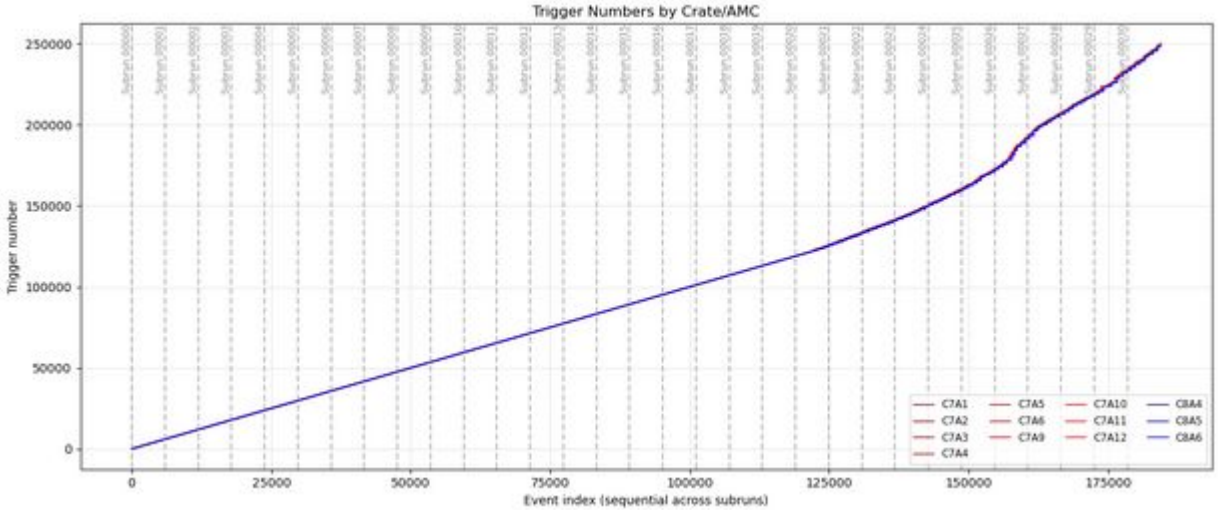
More on desyncs

- Desyncs occurred in multiples of the GPU buffer size (512)

- Saw periods of “instability” and periods of recovery



Trigger index for each module across one subrun



Trigger index for each module across subruns

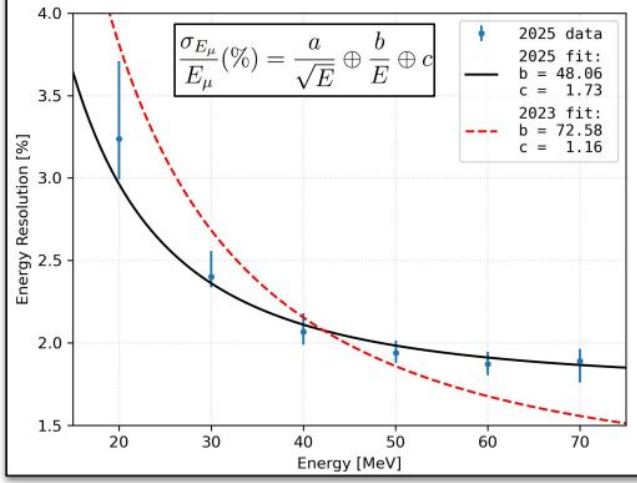
Energy Resolution

Energy resolution:

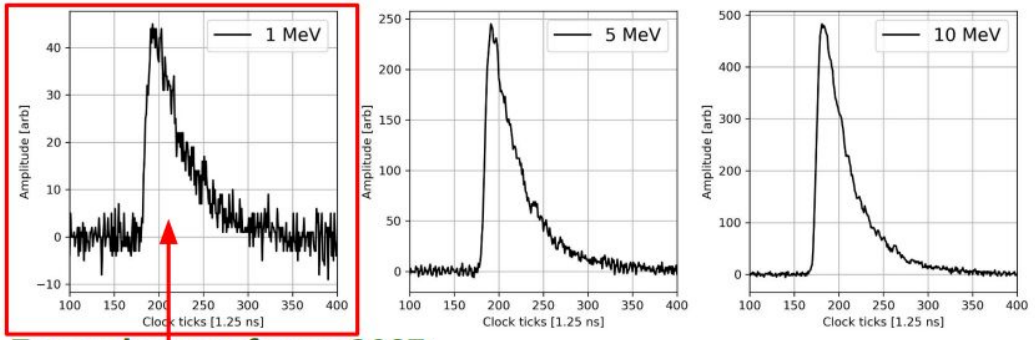
- < 2 % above 50 MeV

Comparison to 2023 testbeam:

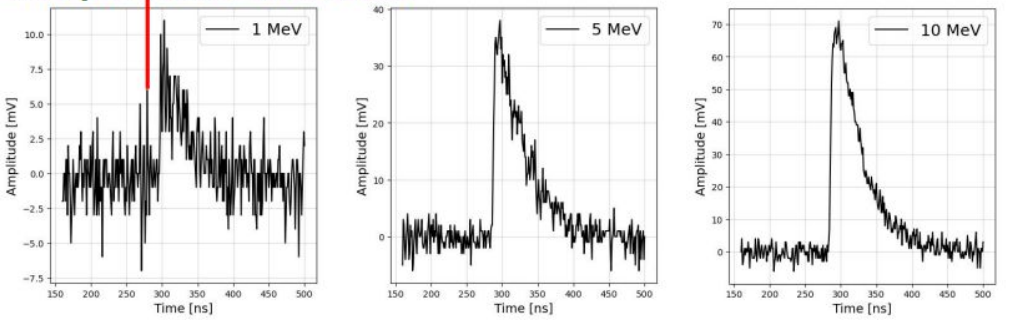
- Lower noise with new PMTs
- More leakage



Example waveforms 2025:



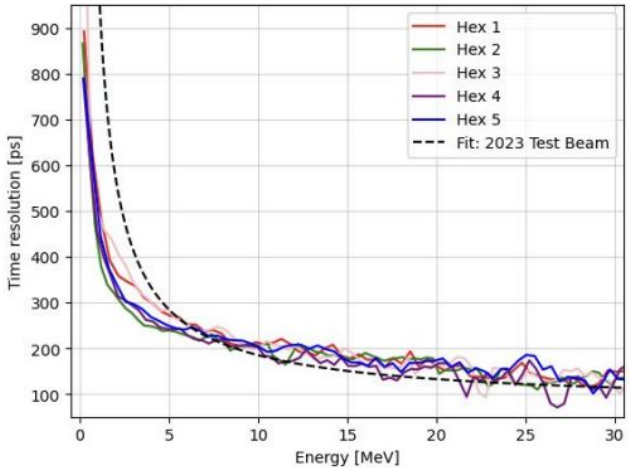
Example waveforms 2023:



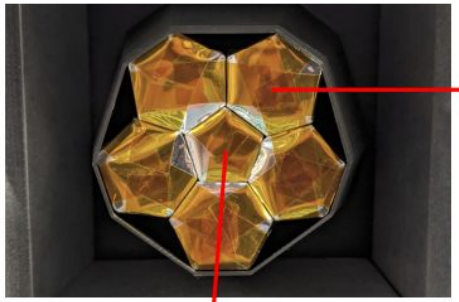
Timing Resolution

Time resolution:

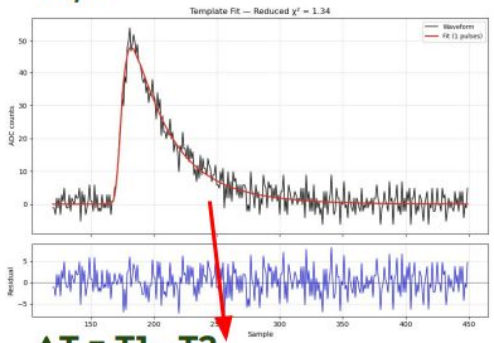
- Similar 100 ps time resolution above 30 MeV
- Sub 1 ns time resolution down to 511 keV



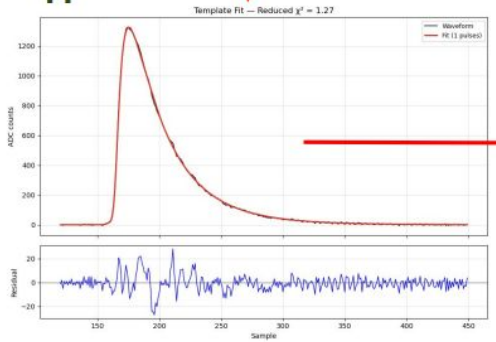
Time resolution measurement:



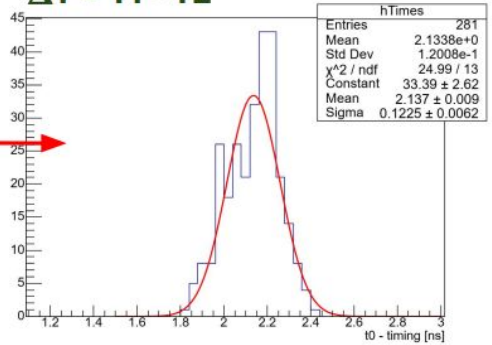
T2, E



T1



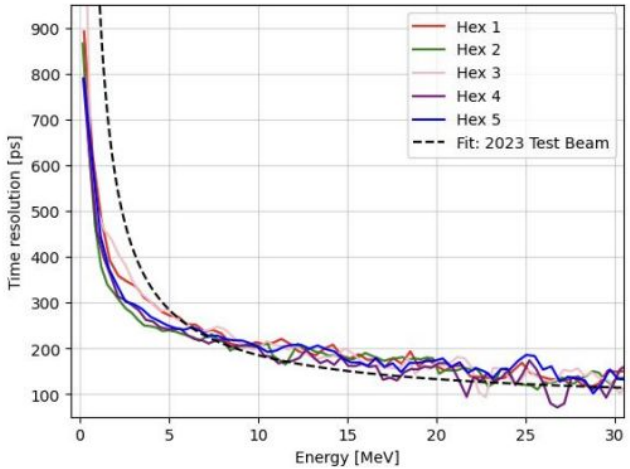
ΔT = T1 - T2



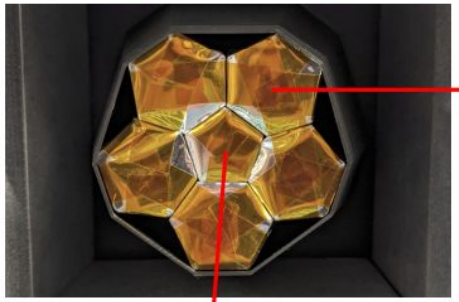
Timing Resolution

Time resolution:

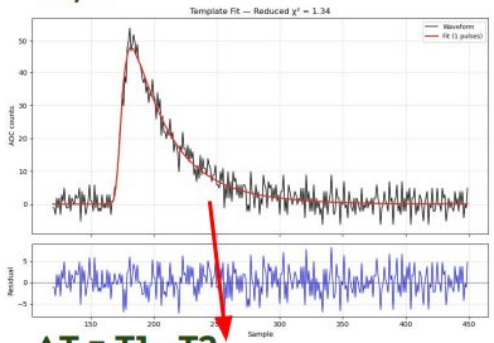
- Similar 100 ps time resolution above 30 MeV
- Sub 1 ns time resolution down to 511 keV



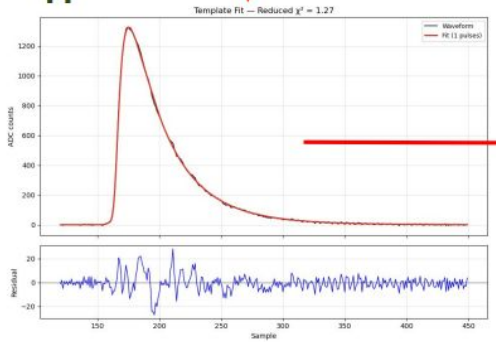
Time resolution measurement:



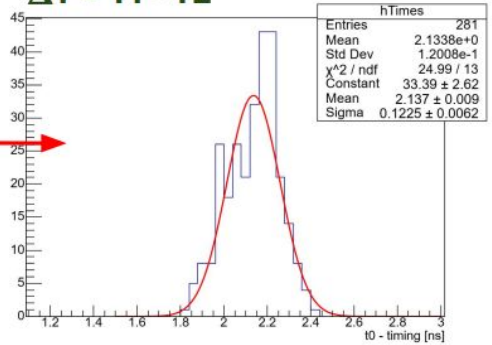
T2, E



T1



ΔT = T1 - T2



Position Resolution

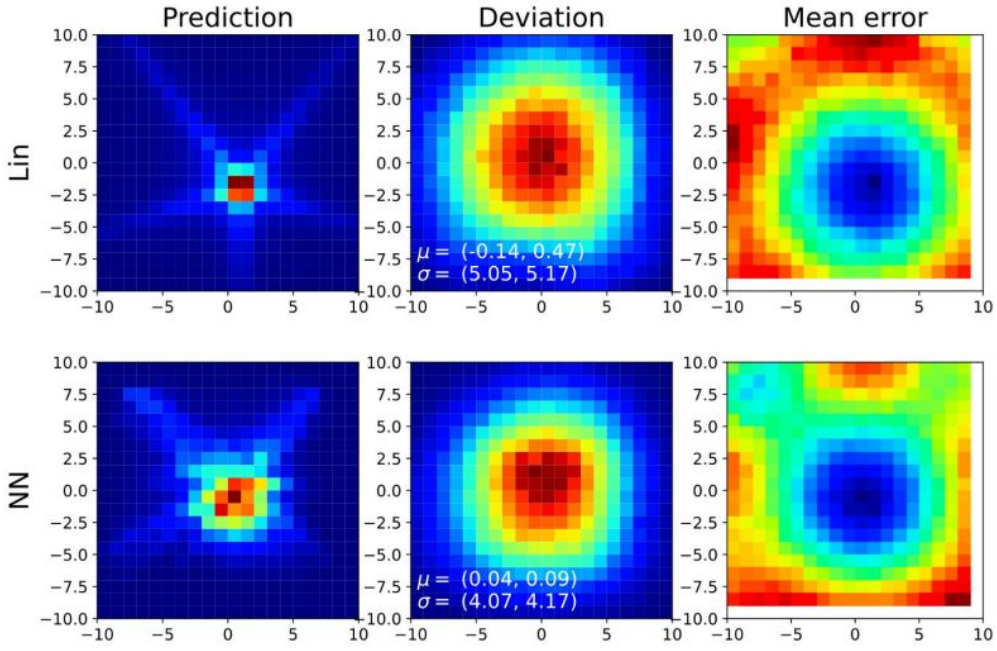
Position resolution:

- Energy weighted mean:
  ➔ ~ 5 mm resolution
- Neural network:
  ➔ ~ 4 mm resolution

➔ Result can be used to crosscheck simulations

➔ Simulations will extrapolate result to full calorimeter geometry

Position resolution measurement:

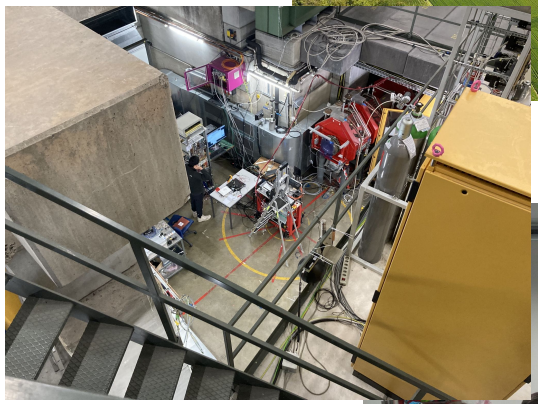


2023 PSI Test Beam

Overview

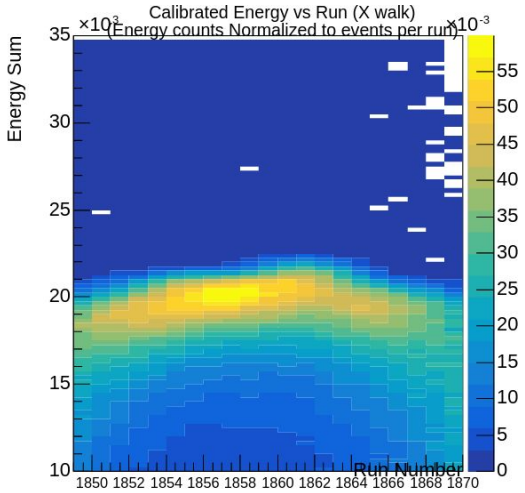
- PIONEER LYSO Calorimeter test
  - November 15 - 29, 2023

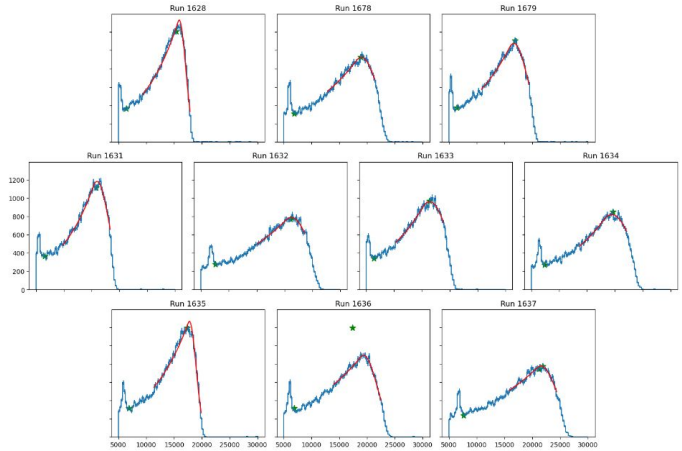
- Made measurements using LYSO scintillator crystals to determine if they are an adequate candidate for PIONEER's calorimeter
  - **Energy resolution**
  - Timing resolution
  - Spatial resolution



Contributions

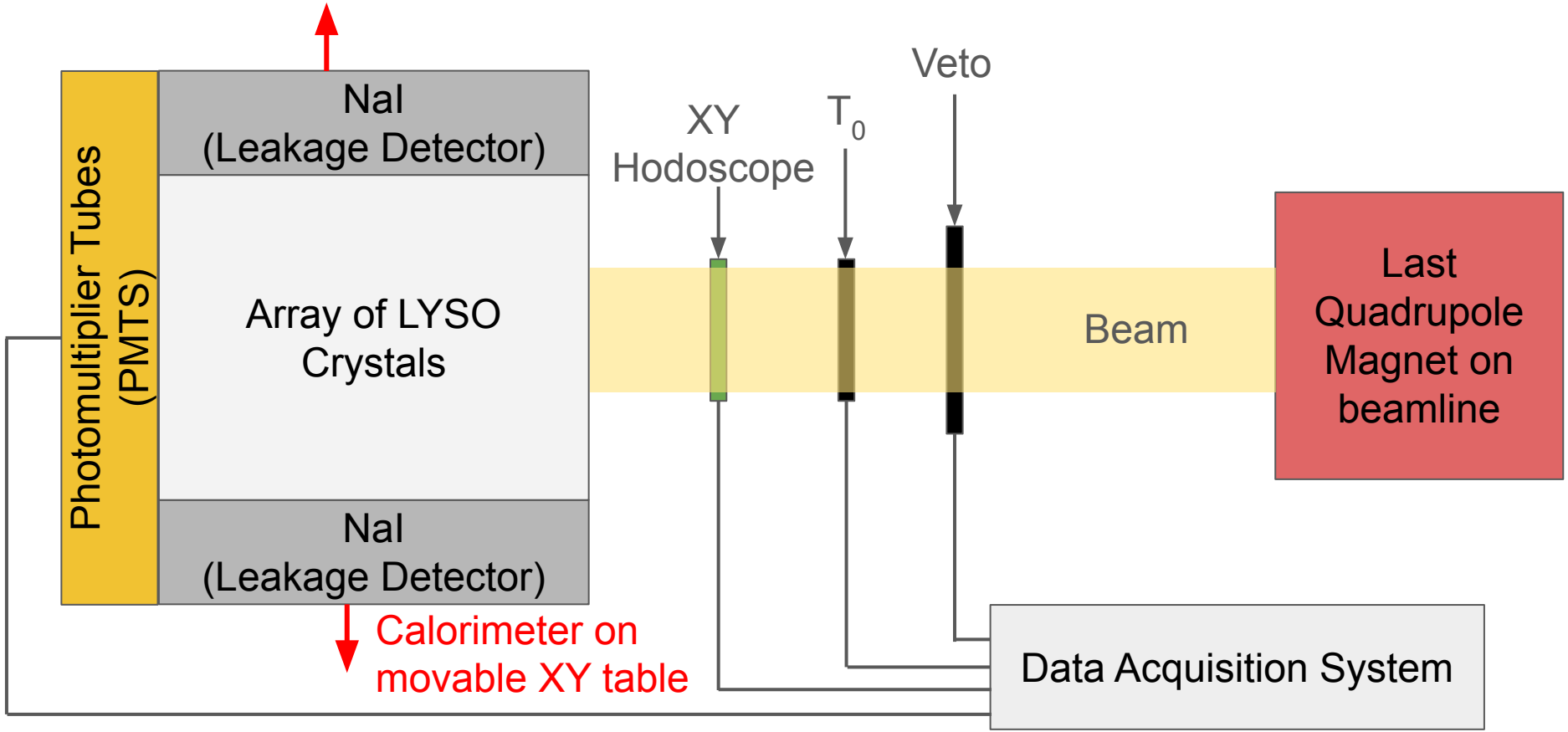
- Repurposing g-2 DAQ Software
- Flexible Pipeline for Data Quality Monitor
- Beamtime “Live” DAQ Maintenance
- Onsite preliminary data analysis



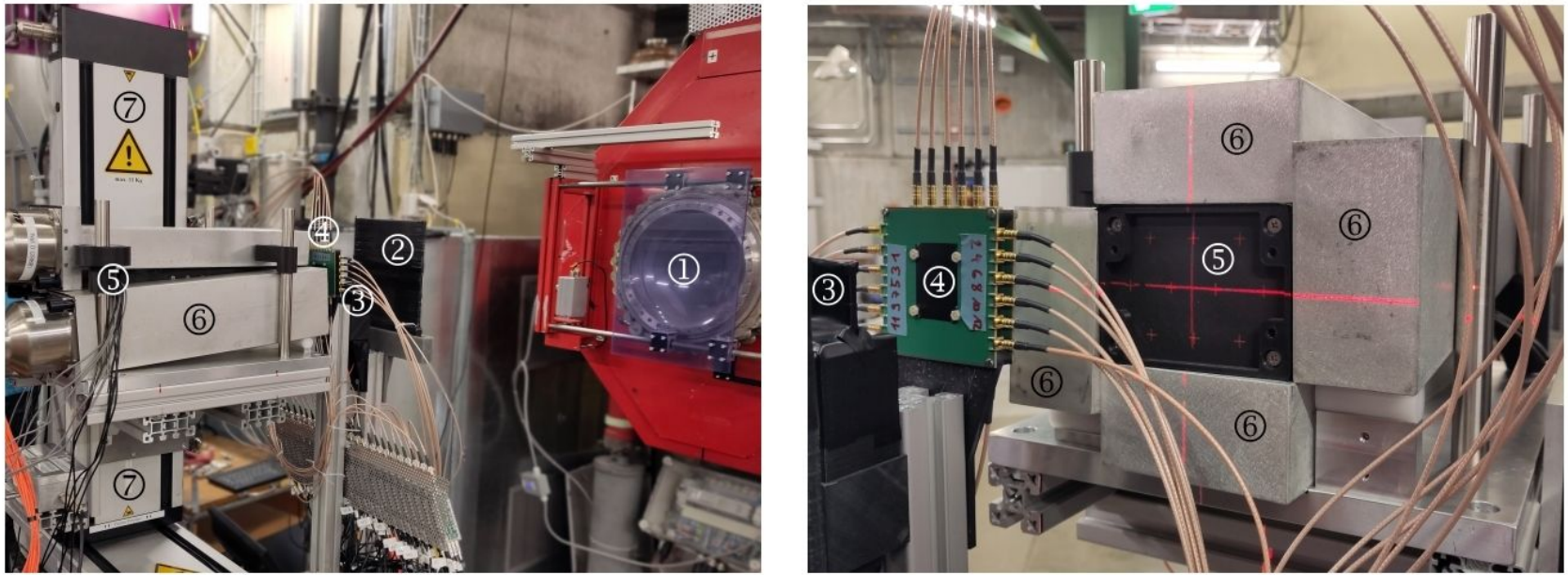


Examples of preliminary analysis work done at PSI

Experiment Diagram - Conceptual Picture



Experiment Diagram - Labeled Picture



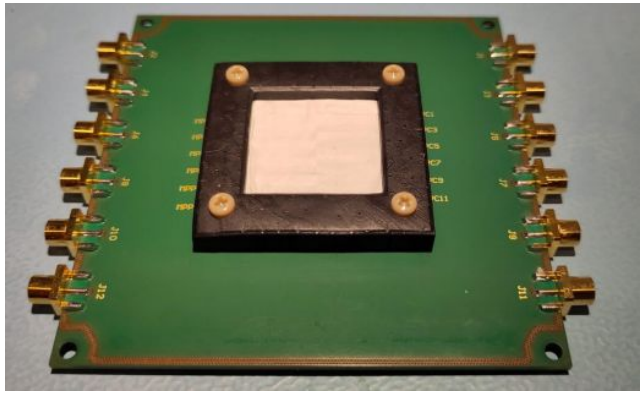
(a) (b)

A full view of the detector setup during the PSI test beam (a) and a close-up of the calorimeter front-face during laser alignment (b). Positrons from the last quadrupole magnet ① pass through the VETO counter ②, T0 ③, and beam hodoscope ④ before depositing energy in the LYSO array ⑤. The LYSO crystals, along with the surrounding NaI detectors ⑥, are mounted on a movable XY table ⑦.

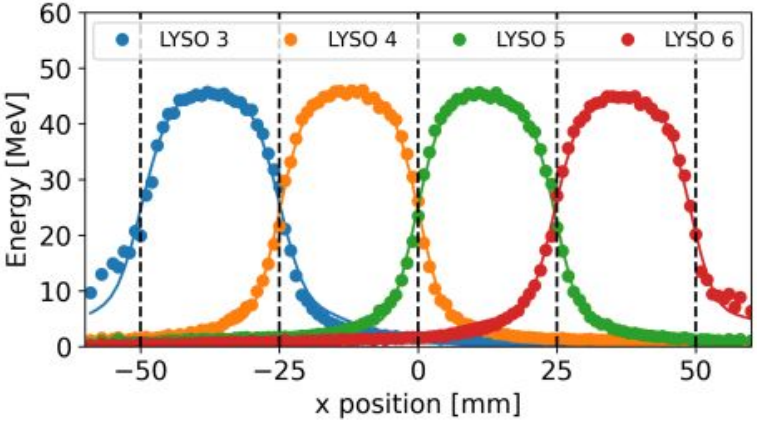
j.carlton@uky.edu

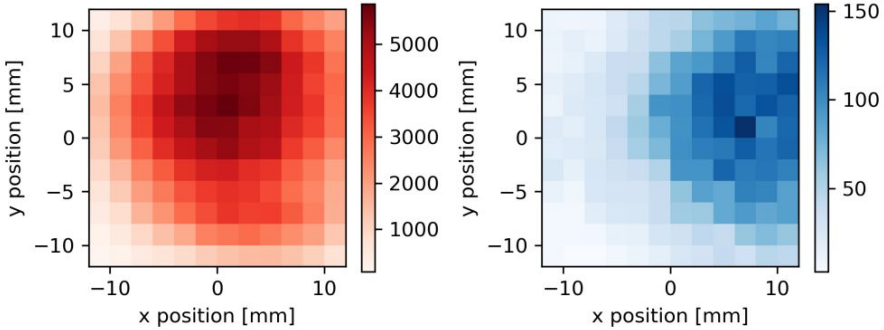
Hodoscope

- 2 Layers of 12 scintillator strips
  - Layers offset by 90 degrees
- 1 mm x 1 mm “pixels” created by strip intersections
  - Allows for finer positioning data



1 Hodoscope layer, 12 SiPMs connecting to 12 BC404 plastic scintillator 2mm wide bars

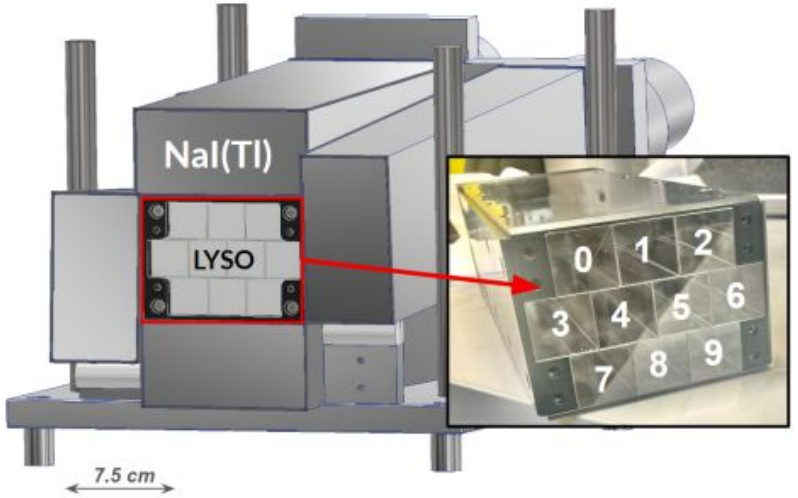




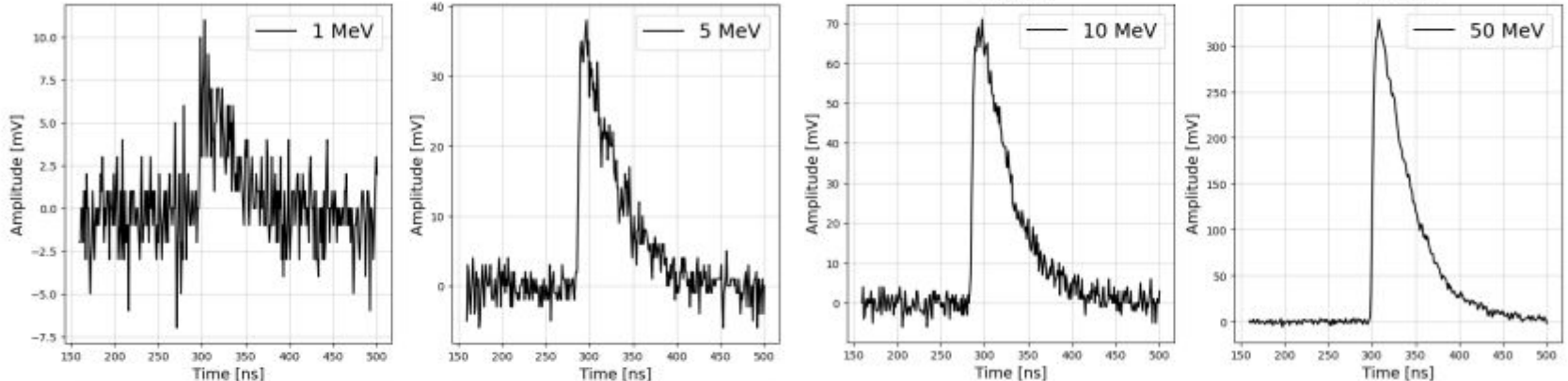
Beam Profile: Red - positrons, Blue - muons

LYSO Calorimeter

- Constructed from an array of 10 LYSO crystals
  - NaI for leakage detection
- X_0 = 1.14 cm
- R_M = 2.07 cm

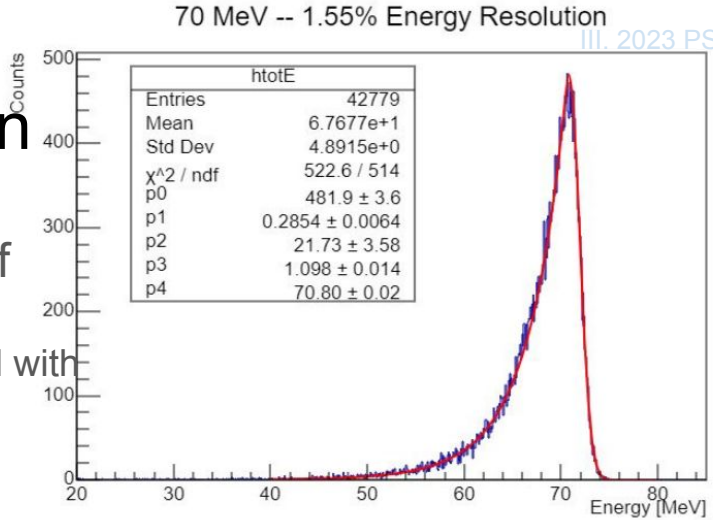


Front-facing image of LYSO calorimeter



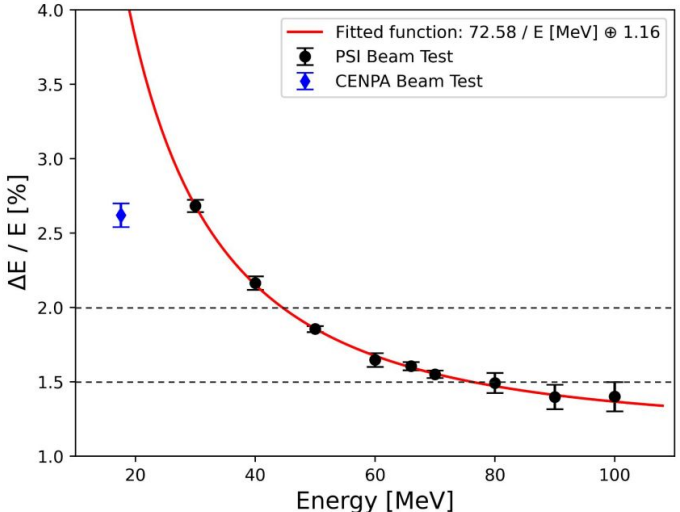
Results - Energy Resolution

- Measured an energy resolution of **ΔE/E = 1.55 ± 0.05%**
  - Published as 1.80, recently improved with better integration strategy
  - 70 MeV ≈ e energy in $\pi \rightarrow e\nu_e$



- Over two times better than reported results for previous generation LYSO crystals

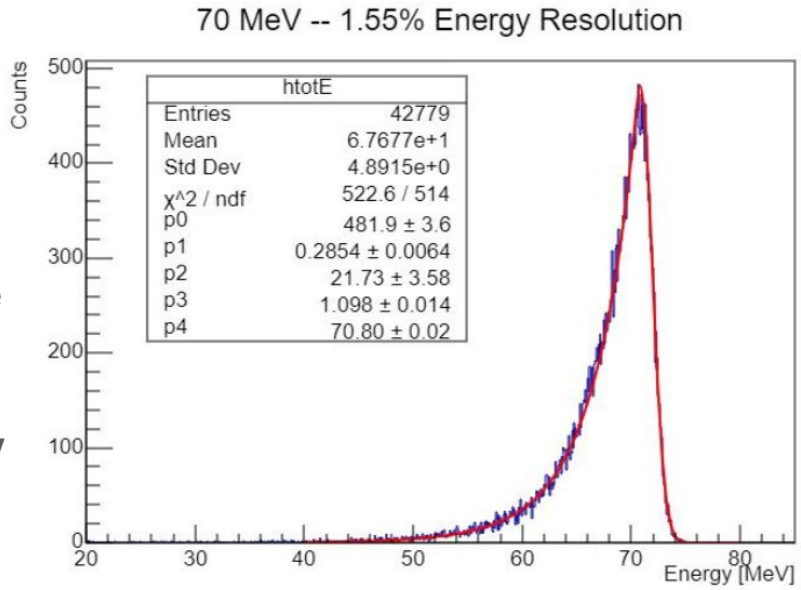
- Similar to liquid xenon energy resolution



j.carlton@uky.edu

Energy Resolution Definition

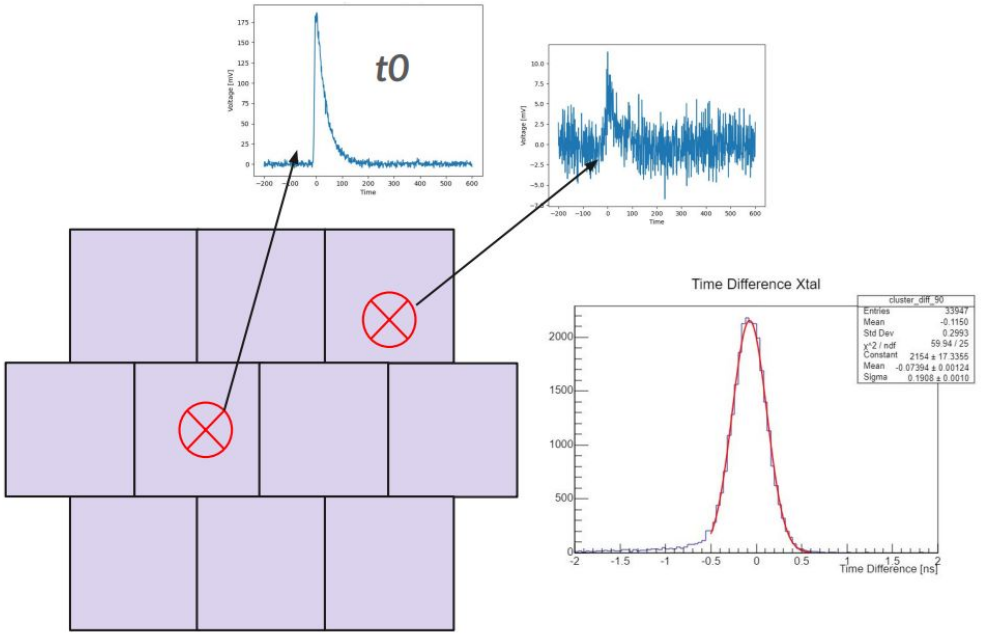
- Energy resolution = ΔE/E
  - E is the peak energy
  - ΔE is the width of the peak
- Gaussian fit around the peak
  - a “crystal ball” fit is used here
  - Gaussian around center, x^{-n} on “sides” where n is a parameter
  - Gaussian parameter σ used for ΔE
- In this case, p4 = **E = 70.80 ± 0.02 MeV**
- p3 = **ΔE = 1.098 ± 0.014 MeV**
- ΔE/E = 0.0155 = 1.55%



j.carlton@uky.edu

Timing Resolution Definition

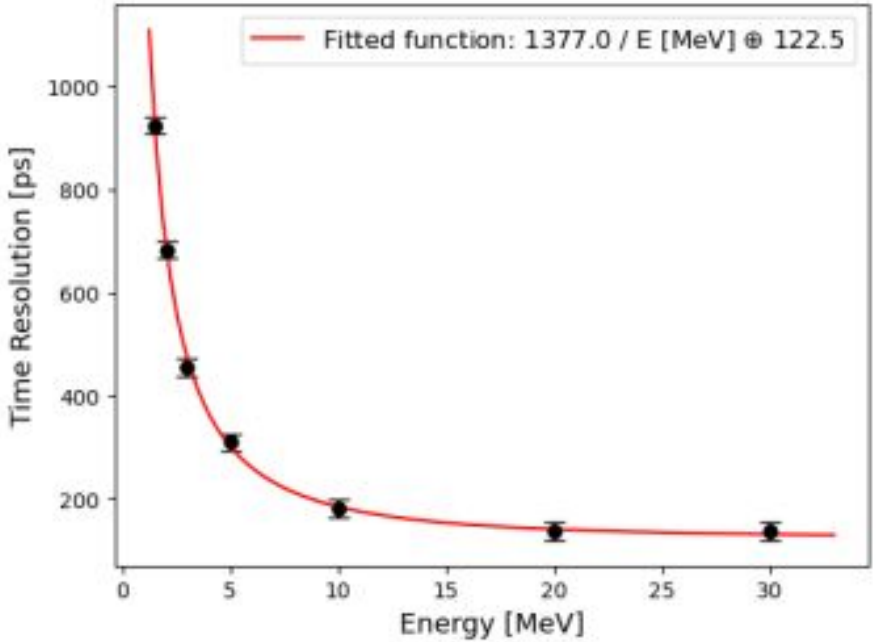
- Use the strongest signal in an event as reference signal.
  - t_0 = time of peak
- In the same event find all crystal peaks t_i
  - Only use peaks above some energy threshold
- $\Delta t = t_0 - t_i$
  - The width of a gaussian fit to a histogram of all such measurements gives the timing resolution



j.carlton@uky.edu

Results - Timing Resolution

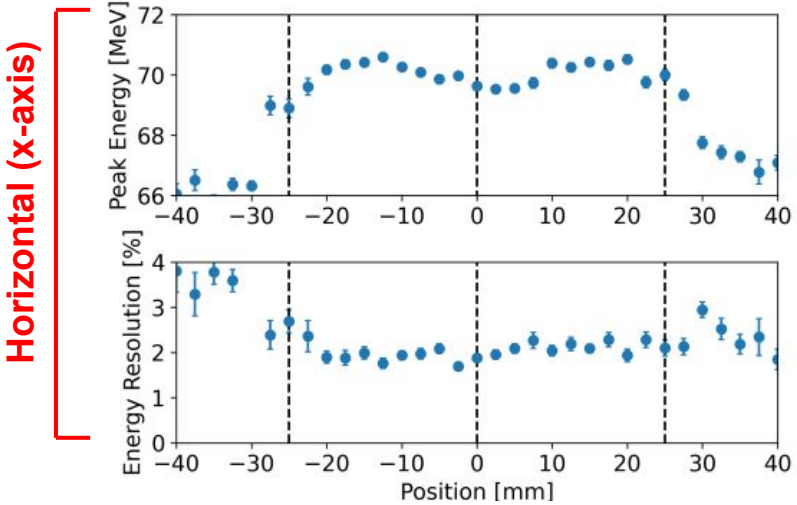
- Timing resolution for 70 MeV events expected to be about 122.5 ps
- This measurement was largely influenced by noise from incorrect high voltage during test beam
  - Using a system of synchronized LEDs, clean, simultaneous signals were generated at UW
  - Improved timing resolution to about 60 ps
  - About that same as LXe

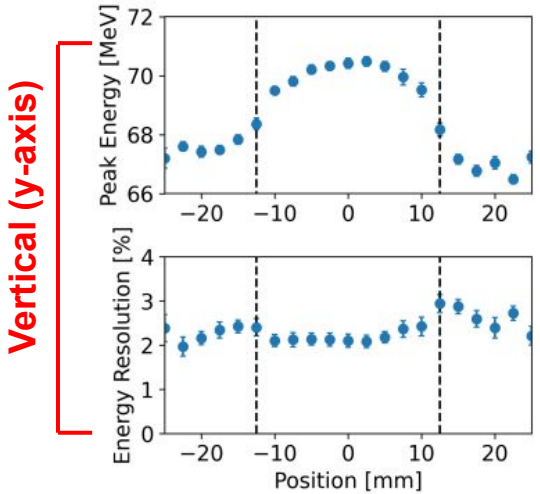


j.carlton@uky.edu

Results - Energy Resolution

- Energy resolution is uniform near the center of the lyso array
- Towards the edges the energy resolution decreases due to leakage
  - In this case, into the NaI array





Graduation Planned Timeline

PIONEER Demonstrator (Now after planned defense date)

- "Full" experiment demonstrator
  - **PSI shutdown delayed, demonstrator moved back to take advantage**
- Prototypes for all detectors
  - Small number of ATAR Layers (16 layers)
  - Small spherical segment of tapered LYSO crystals (12 crystals)
  - Some spherical "shell" segment of tracker
- DAQ handles event construction