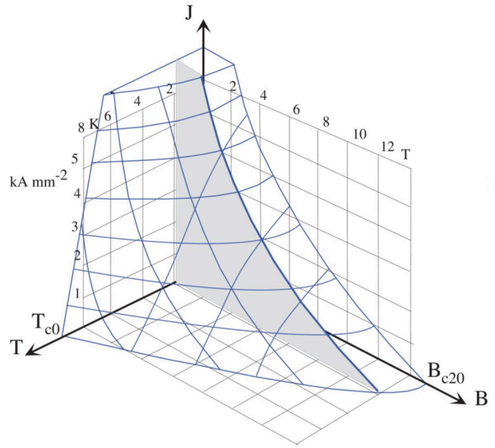
Data Analysis and ML Efforts for Quench Diagnostics

Maira Khan
Jan 13, 2025

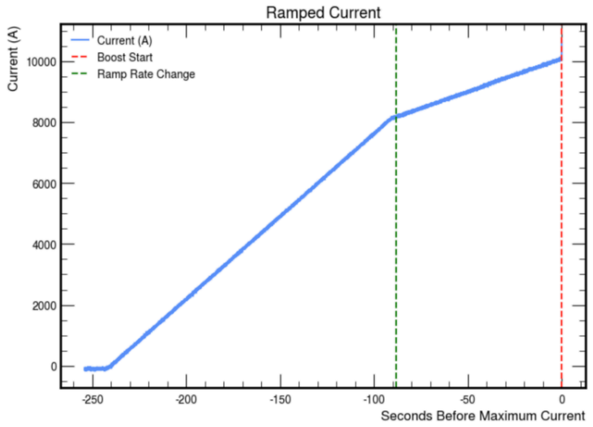
Quench Detection

Superconducting (SC) magnets are critical to the operation of accelerator experiments, and their continuous operation is necessary

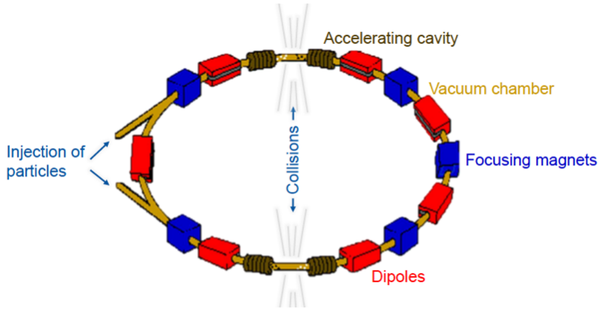
- SC Magnets must operate at high current to produce the required field to control beam trajectory.
- To allow magnets to operate at this high current magnets must be “trained” by iteratively ramping current upwards to reach maximum current until the magnet *quenches*
- A magnet ***quench*** occurs when the SC magnet goes from its superconducting state to a resistive state, exceeding bounds of the critical surface
- Quenches are costly, damaging, so we would like to flag precursors before the quench occurs.



Critical surface of a magnet



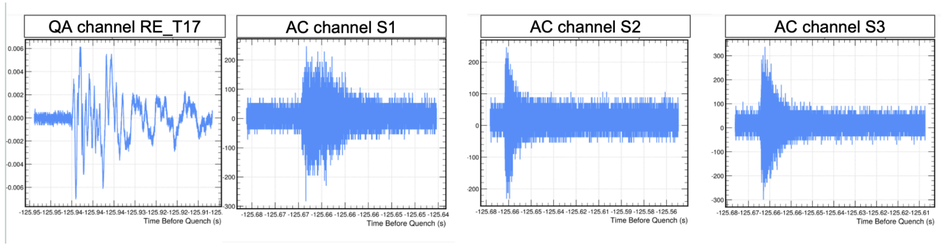
Current ramping linearly to quench



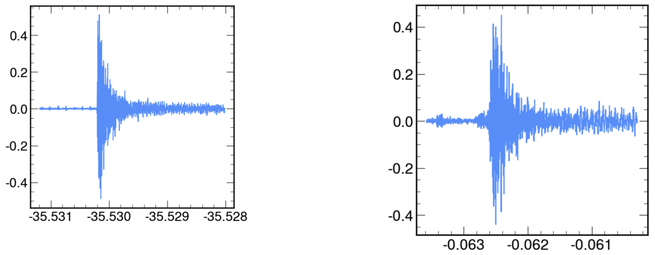
$$B = \mu_0 \frac{NI}{l}.$$

Large fields control beam trajectory

Our Goal: Building a Reliable ML Based Quench Detection Algorithm



Sample Event MBHSM03 Ramp 3



Sample Events MQXSFS1d

Quench

$$u \propto I^4$$

Current and strain energy density increases, often scaling with event density

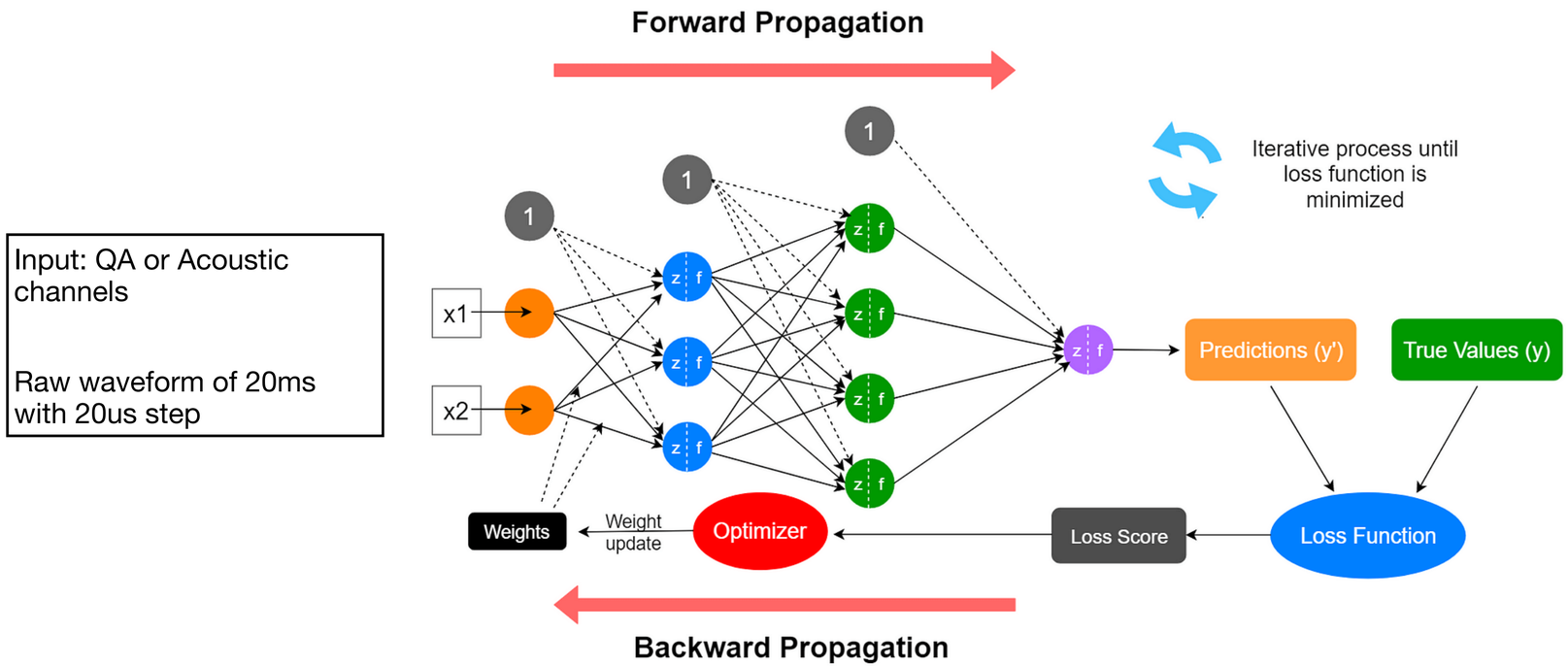
Events closer to the quench could be more “energetic” than events occurring during training. We would like to capture these differences

Ideal trigger is on an O(ms) scale

~250ms of quench

What does an ML Model Need to “Diagnose a Quench”

First.. A Brief Introduction on ML Models



1. Inputs which contain some parameters (treated I.I.D from a distribution) are fed into a neural network

2. These inputs are propagated through the network. A weight matrix is trained to optimize some objective via a loss function. Weights are iteratively updated. Post training, inputs are passed through this weight matrix to return a value. This could be how anomalous an input is, or show us whether it is in one input class or another

To build the simplest network possible: we can classify snippets of time series and tag if they are implicated in a quench or not. For this, we need a notion of TRUTH from our data.

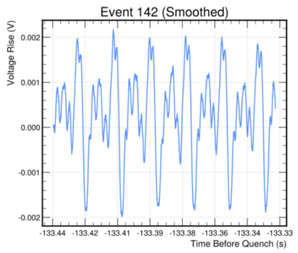
Two Strategies for ML Based Approaches

- We can either build a supervised dataset, maximize the likelihood of triggering as close to the quench as possible using a training set of potential precursors or simply train an unsupervised model to recognize based on the history of previous events and their configurations

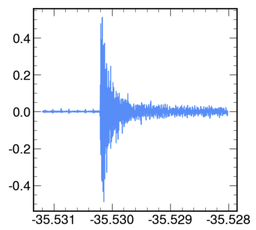
- The supervised approach makes the assumption that there *exists* such an event to trigger in the first place. We are working on time of arrival embeddings.

Supervised Model

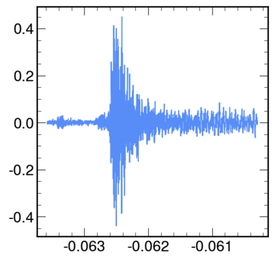
Automatically tag and classify events



Noise



Training Event



Precursor

Unsupervised Model

Model takes statistical inputs, produces some anomaly value and we design a trigger algorithm based on these outputs (shown at MDP 2024)

Building an Event Tagging System

- An automated data curation and selection system was developed to generate the following classes of data: (1) electronics/thermal noise (2) events that *do not* directly cause a quench and (3) events likely implicated in the quench

- By definition, we should define (3) from the positive differential offset point from the earliest voltage taps

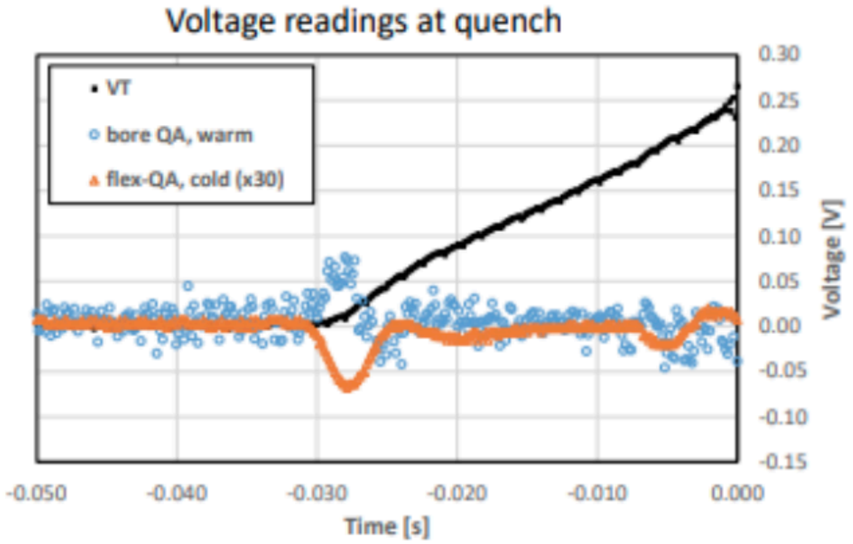
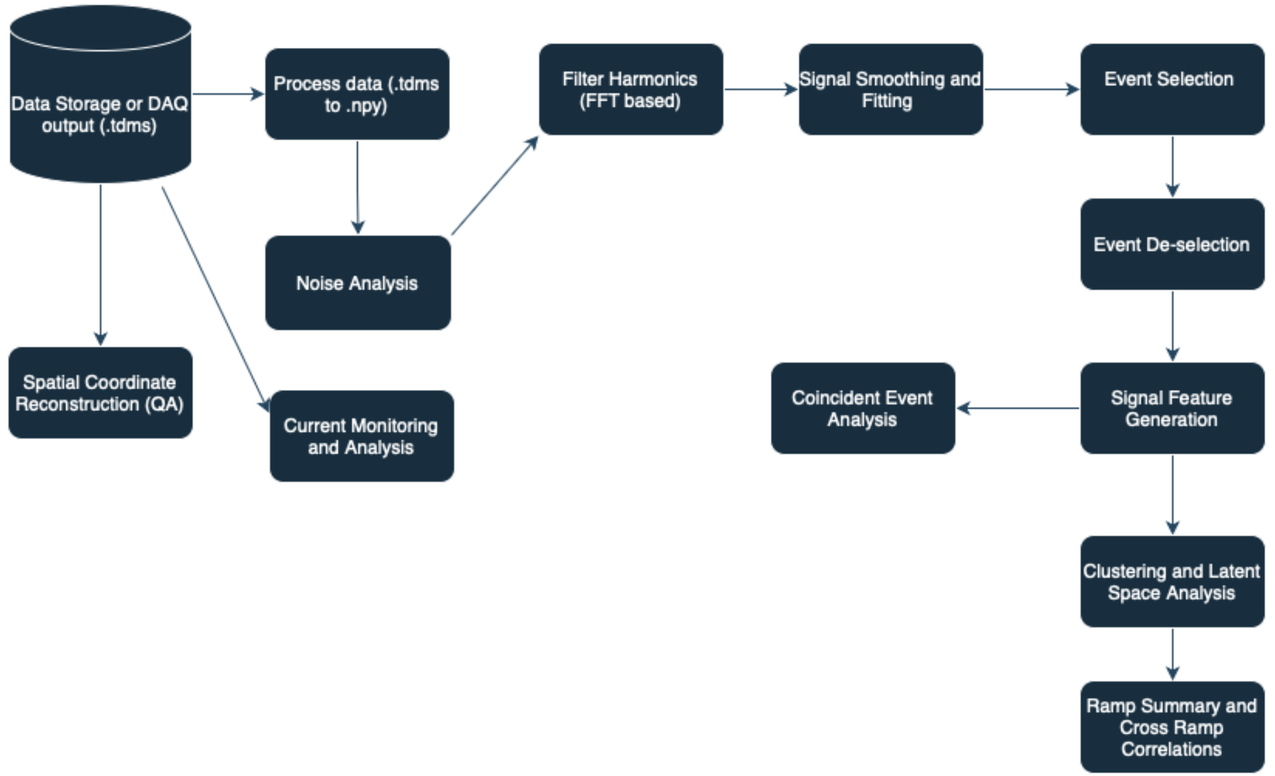
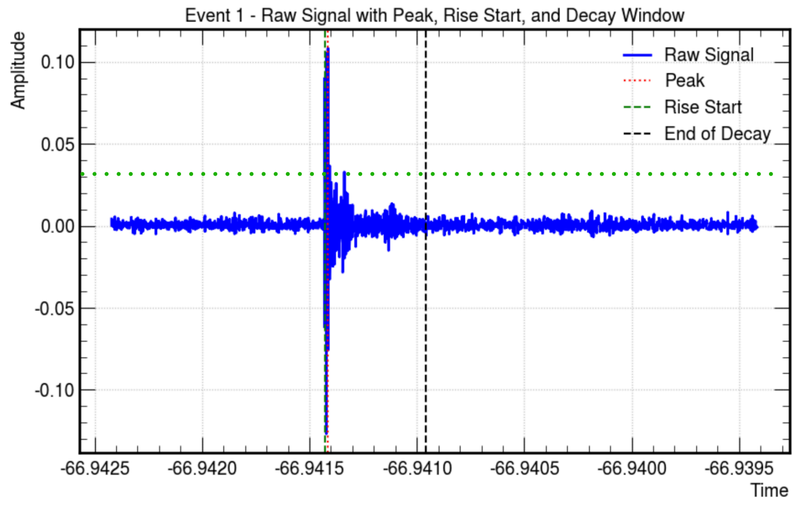


Fig. 4. The second (last) quench in the inner coil: Earliest segment/section of VTs and QAs responding to quench development, quench detection is at 0 s.



Event Tagging and Selection Methods



Threshold

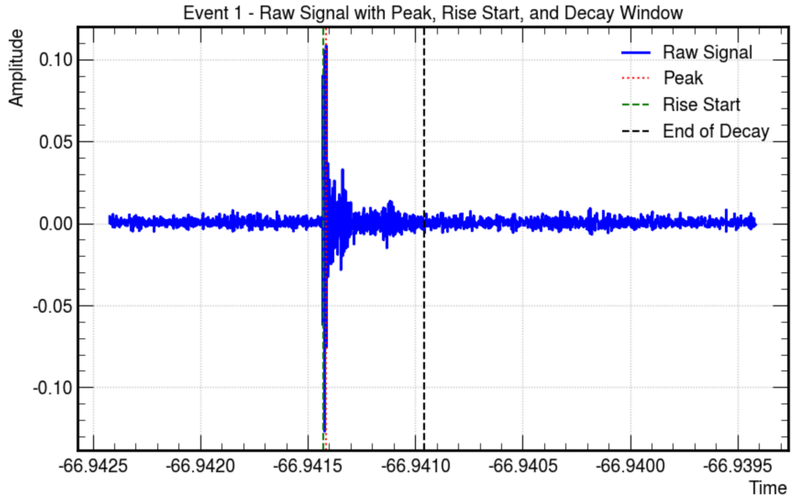
Option 1: Threshold based

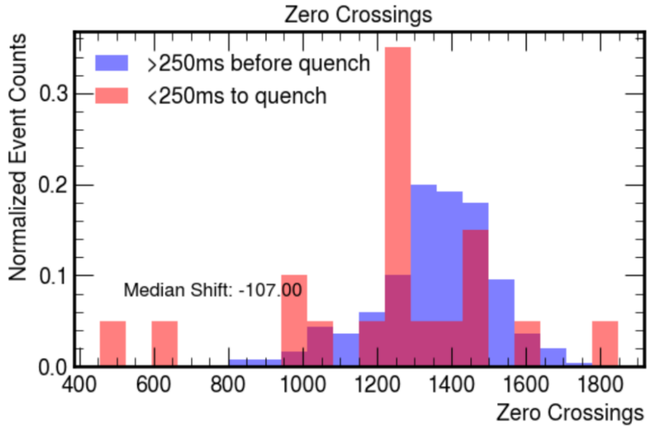
For a normalized signal, we find where the signal exceeds some threshold value and recover 1ms before the peak onset. We iteratively group together the successive peaks above a noise floor until we recover the fully decayed signal.

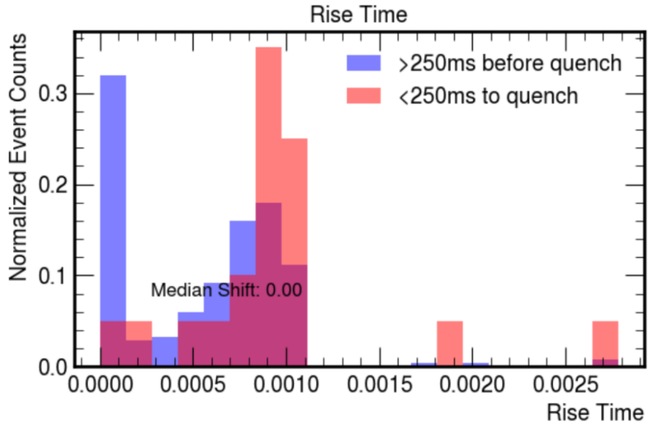
Option 2: Differential based

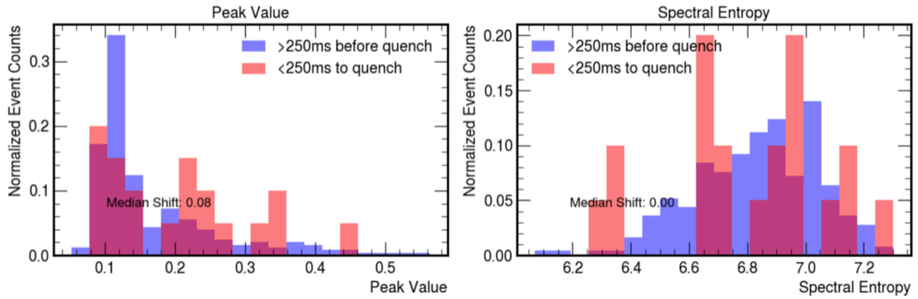
For a normalized signal, we find where the signal exceeds some differential threshold. When the differential returns to that of the noise floor (before peak onset) for a some decay time parameter, we end the event window

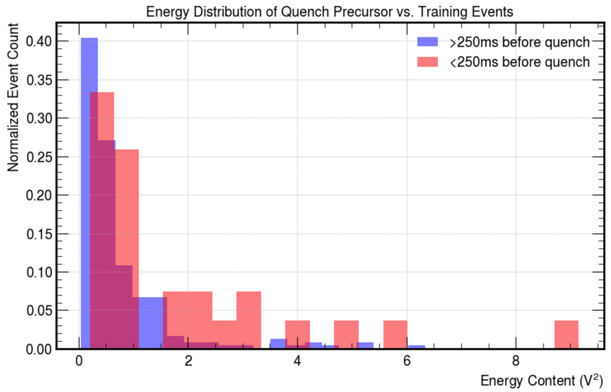
Event Tagger Utility 1a: Understanding Events in Training Using Distributions









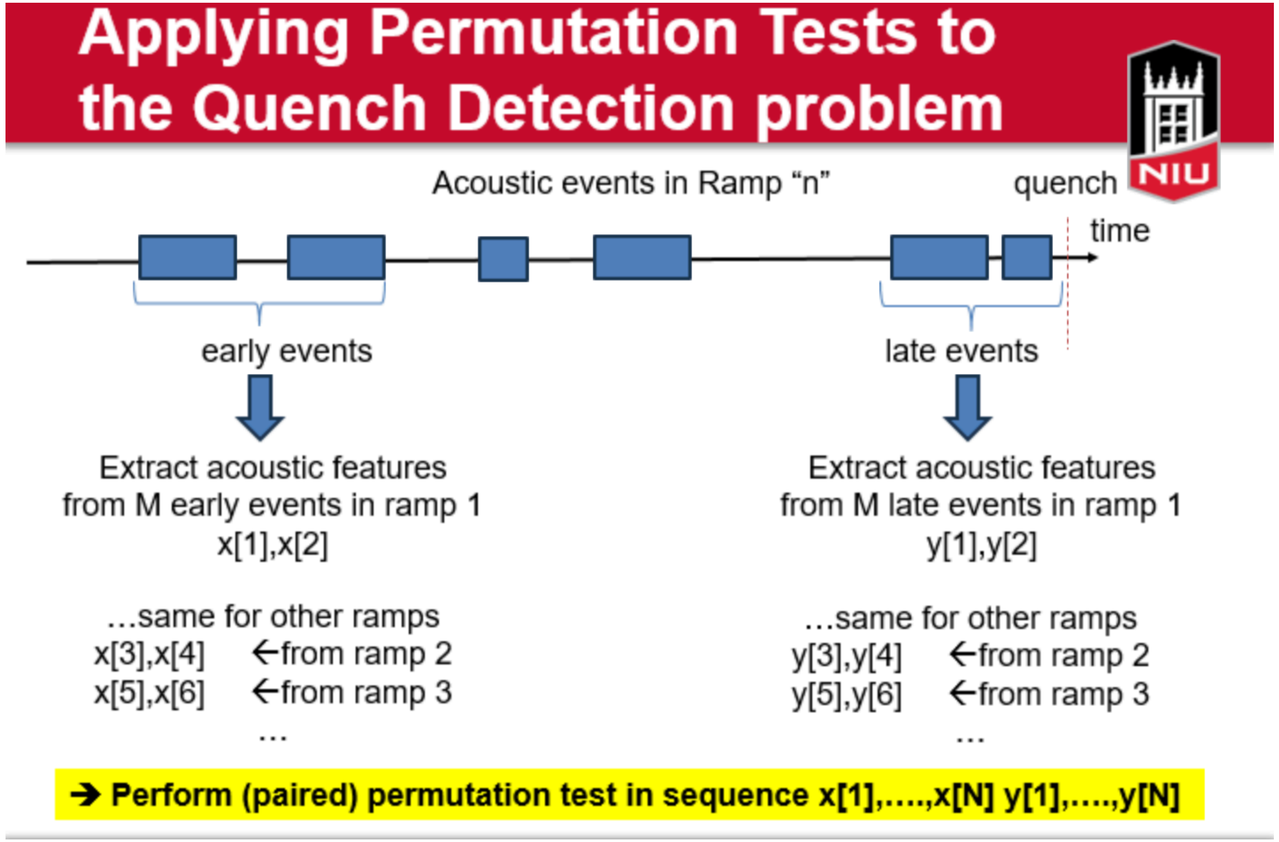


Machine learning models try to “learn” how to fit our inputs (windows of time) to a very high parameter function

By isolating the regions of higher signal amplitude closer to quench, as well as events that occur during training we can analyze these signal snippets as samples from a distribution

This may inform what inputs we might put into the model to better trigger on events closer to the quench

Event Tagger Utility 1b: Building Statistically Significant Differences



With collaboration of NIU professor B. Fonseca and his student B. Roehrig, we have been able to come up with a methodology to understand if there are statistically significant differences between events closer to the quench

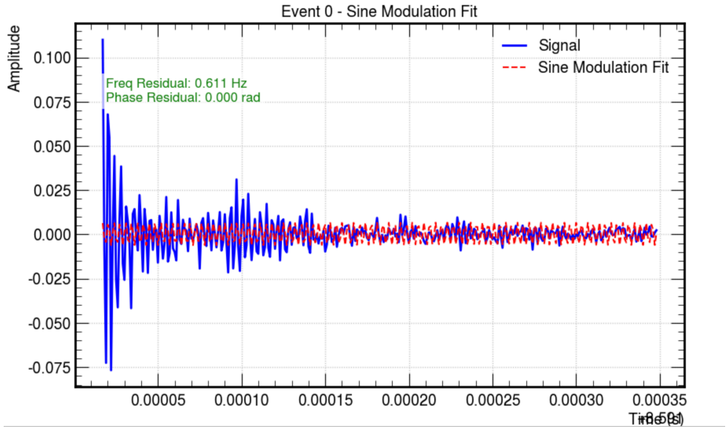
Initial p values show that at higher amplitude events, there is are statistically significant differences between training events and quench precursors using some frequency based parameters

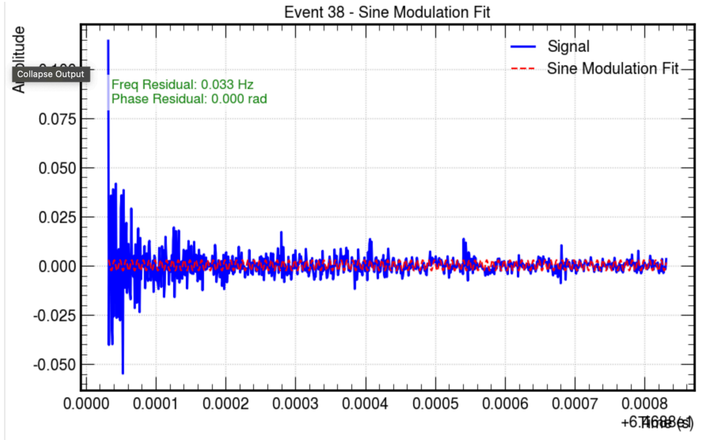
thresh	p-values from 5 tests					
0.65	0.0001	0	0.0001	0	0	
0.55	0.0002	0	0.0001	0	0.0001	
0.45	0.0002	0.0002	0.0003	0.0004	0.00009	
0.35	0.0002	0.0001	0	0	0.0001	
0.25	0.0008	0.001	0.0004	0.0004	0.001	
0.15	0.04483	0.04777	0.0496	0.0491	0.479	
0.05	0.1309	0.1391	0.1353	0.14	0.1365	

thresh = 0.25					
winLen	p-values from 5 tests				
1000	0.0006	0.0011	0.0005	0.0008	0.0003
875	0.0001	0.0001	0.0002	0.0001	0.0003
750	0.0001	0	0.0001	0.0002	0.0001
625	0.0051	0.0052	0.0062	0.0049	0.005
500	0.0054	0.0063	0.0052	0.005	0.0063
375	0.0086	0.0082	0.0103	0.0093	0.0086
250	0.0025	0.0021	0.0029	0.0027	0.0028

Event Tagger Utility 2: Building Data Driven Simulations

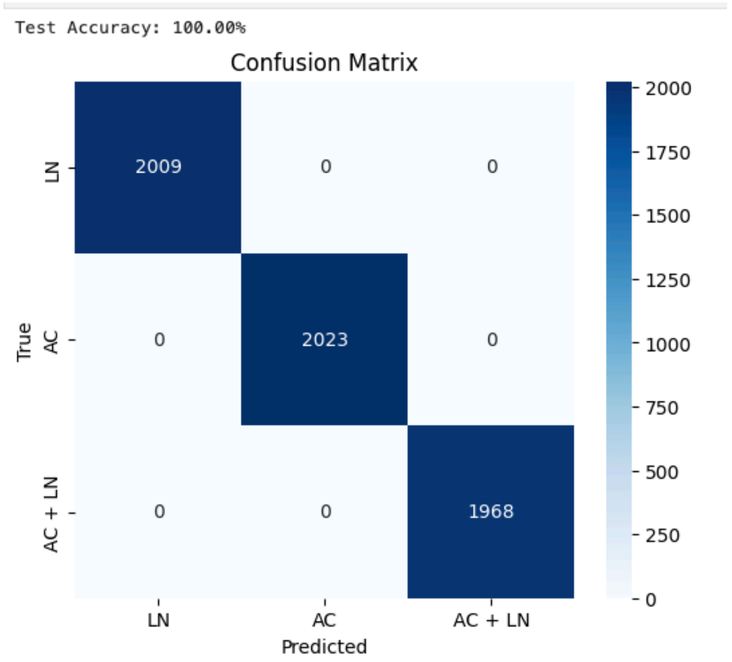
- From these isolated events, we can also build the following simulations in a data driven way
  - Acoustic event simulations
    - Fit to dominant frequency component (very naive)
    - Apply fit to $x(t) = A \cdot e^{-\alpha t} \cdot \cos(2\pi f t + \phi)$
  - QA
    - unipolar and bipolar pulsed simulation in progress
  - Noise simulation
    - Mask events
    - Calculate and discretize PSD
    - Generate uniformly distribution random phases [0, 2pi]
    - Scale amplitudes
    - Generate complex FFT spectrum
    - Take iFFT

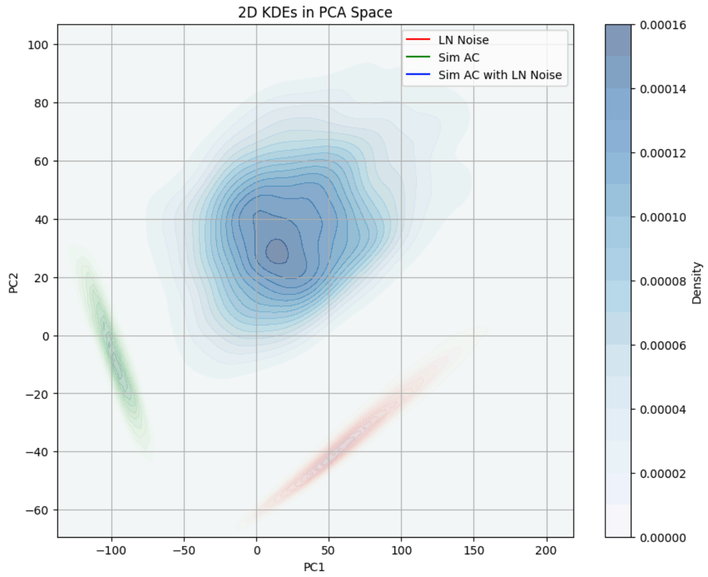


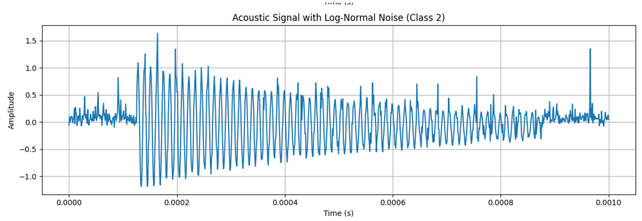


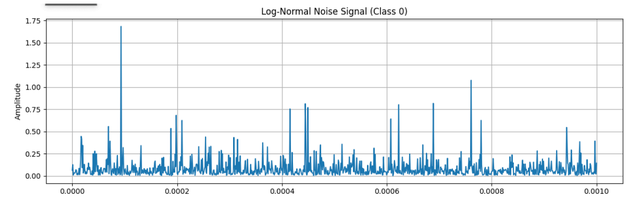
Building a Notion of Distance on Simulated Data

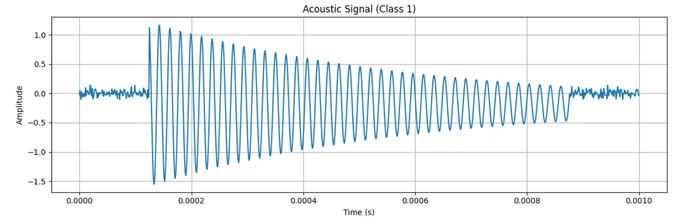
- To start to build this supervised learning pipeline, a very preliminary toy dataset was build
  - 3 Classes:
    - Toy simulated acoustic events with some baseline white noise floor
    - Toy simulated acoustic events with log normal noise
    - Log normal and white noise











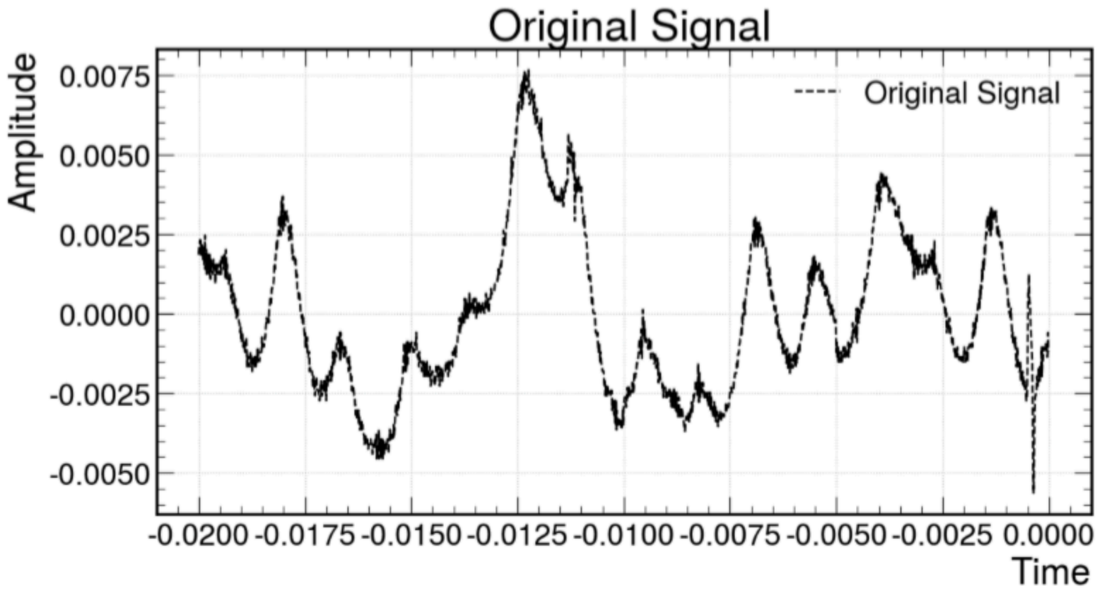
Performance is perfect, given it is an easy problem on very much toy data, but the utility is in building a notion of separation in the model for these events.

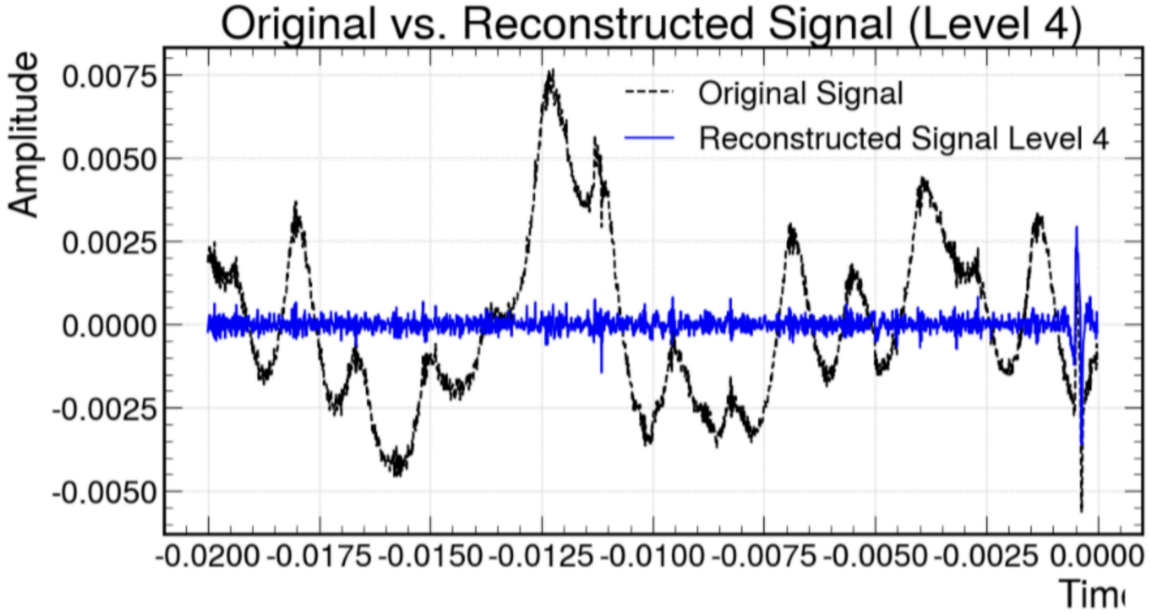
We can re-use these learned distances to make better decisions on our real data (to come soon).

HTS Studies

- HTS Magnets are currently under investigation and FNAL, though the current threshold based systems cannot reveal signals from below the noise floor.

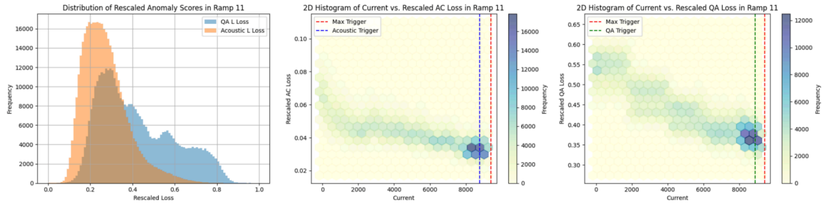
- Initial investigations to using wavelet coefficients for reconstructing signals below the noise floor have been made, and we aim to potentially integrate these features to our ML based pipeline





Project Thrusts Moving Forward

Unsupervised Machine Learning Methods



- Need a complete study of unsupervised methods, with trigger optimizations to establish an unsupervised baseline
- Potential use of learned wavelet embeddings to isolate signals in HTS magnets for trigger under power supply

Supervised Machine Learning

- Must validate existing simulation pipeline and iteratively train classification model with more real data
- Current simulation validation tests must be completed using real data runs
- Consistent metrics should be recorded to note the “distance” between noise, precursors and training events
- Tests must be complete performing inference on real “tagged data”
- We must add relevant geometric/locational information to add both cumulative and localized parameters

Semi-Supervised Machine Learning

- Must use existing learned distances from supervised setting in a real time setting to see if our priors from simulation translate to real data