Machine Learning with the MIP Timing Detector

Margaret Lazarovits

Learning Machine Learning

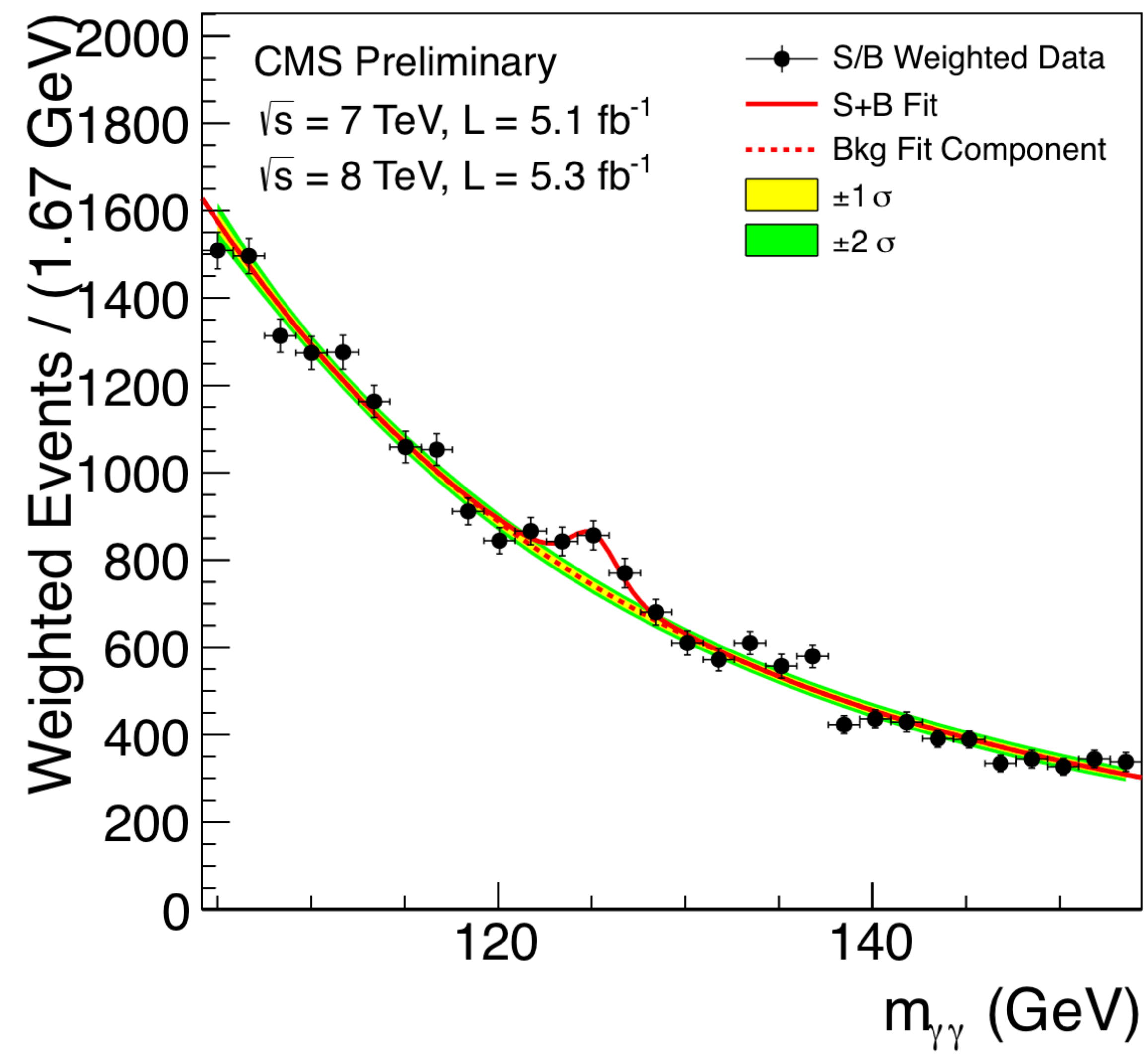
February 28, 2019

Outline

1. Particle Physics Overview
   - What are we looking for?
   - How are we looking for it?
2. Detector Physics
   - Compact Muon Solenoid (CMS)
   - HL-LHC Upgrade & Issues
3. What is the MTD?
   - Basics
   - Design
4. Current ML Status
   - Neural Network
5. Next Steps

What are we looking for?

- Standard Model Physics - completing the picture
  - Mass Hierarchy
  - Matter/Antimatter
  - Unifying forces
- New Physics - Beyond Standard Model (BSM)
  - SUSY
  - Higgs Boson
  - Dark Matter
- Cosmology
  - Recreating early universe conditions
  - Heavy ion collisions
  - Cosmic rays



Higgs🅱oson discovery in diphoton channel (CERN, 2012)

How are we looking for it?

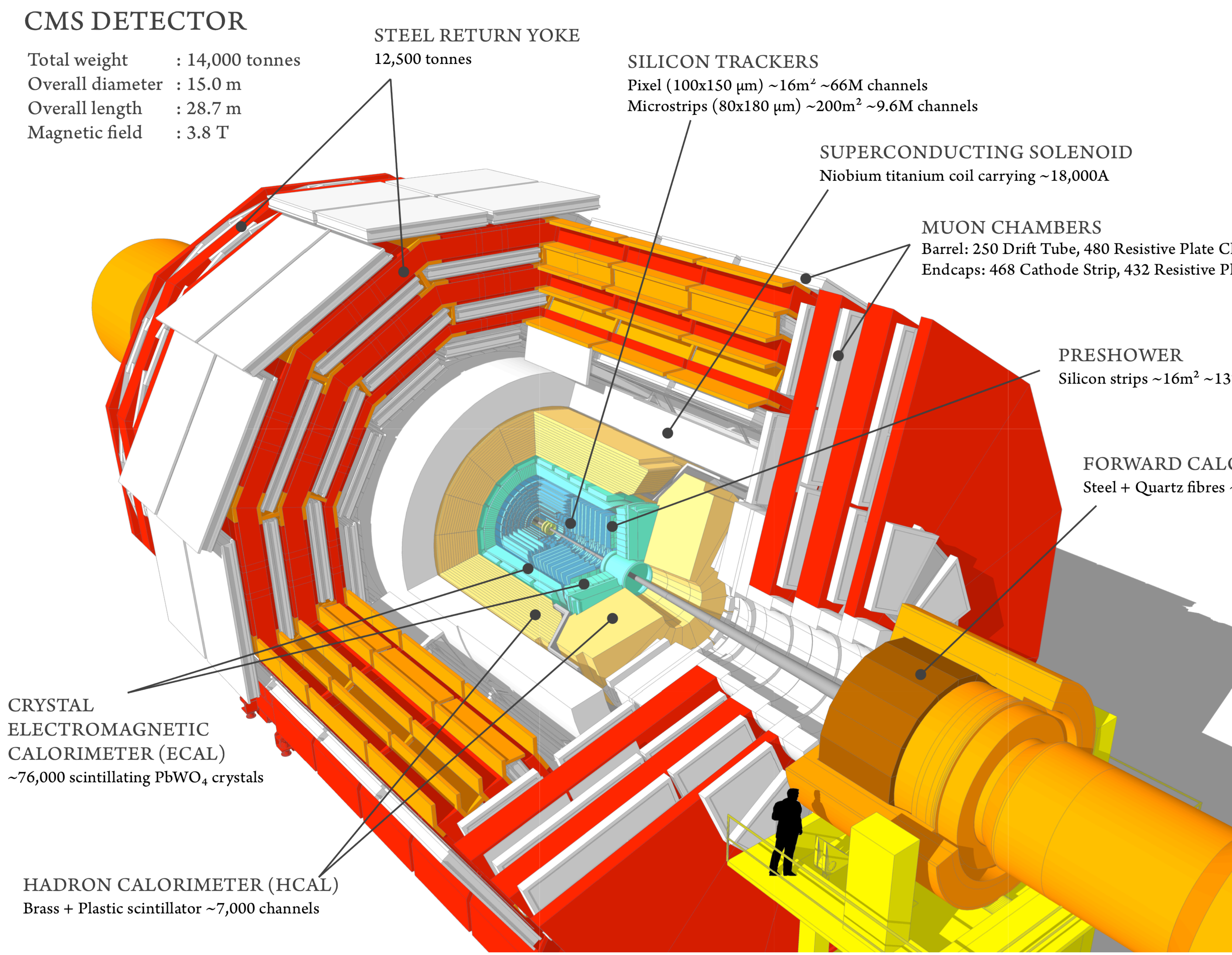
Accelerators

- SLAC - Stanford Linear Accelerator Center
- ILC - International Linear Collider
- Fermilab - Tevatron (RIP 1983 - 2016)
- LHC - Large Hadron Collider

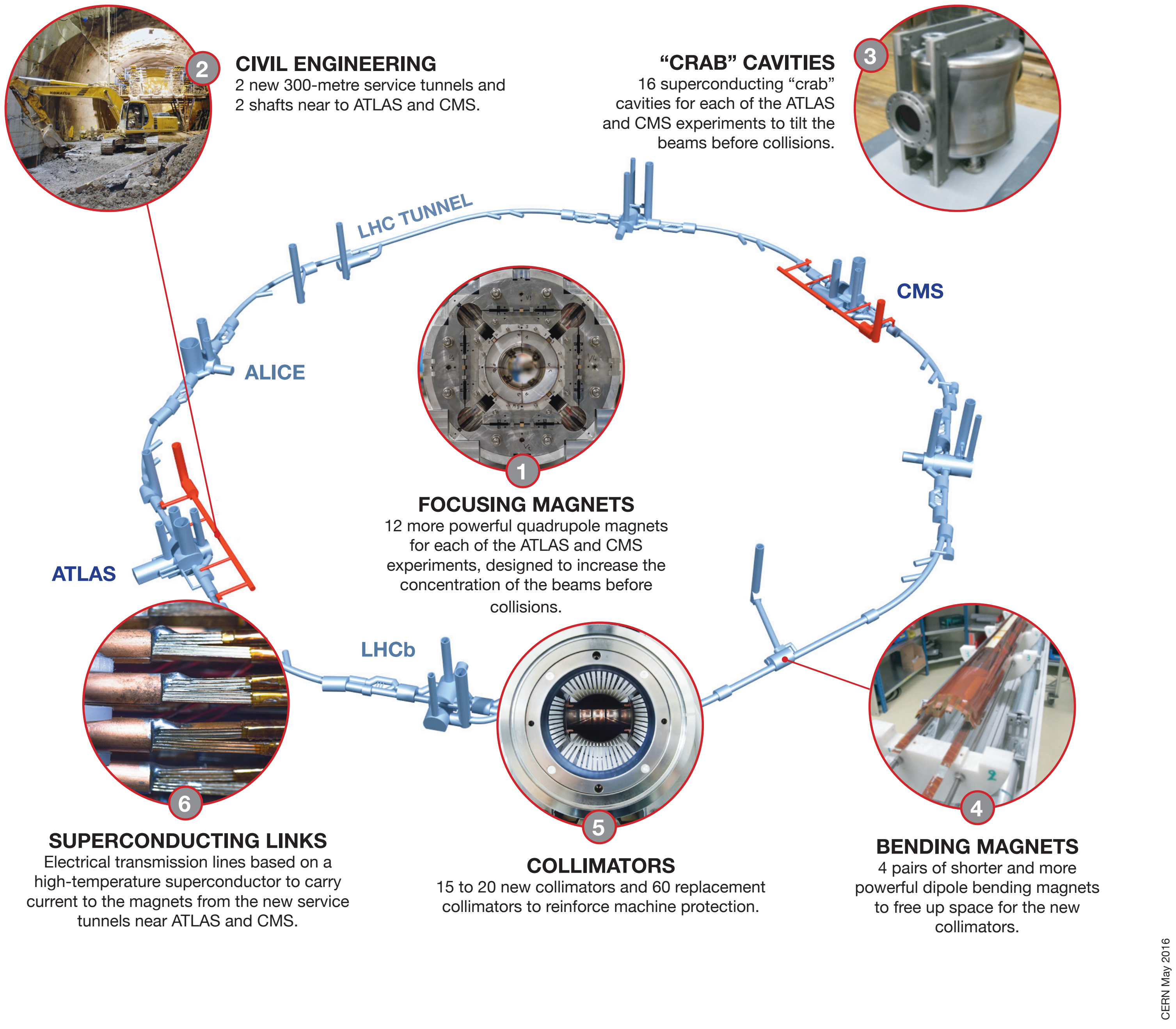
Detectors

- DUNE - Deep Neutrino Underground Experiment
- IceCube
- MINERvA - Main Injector Experiment for v-A
- D0 + CDF
- CMS + ATLAS

Compact Muon Solenoid (CMS)



HL-LHC Upgrade

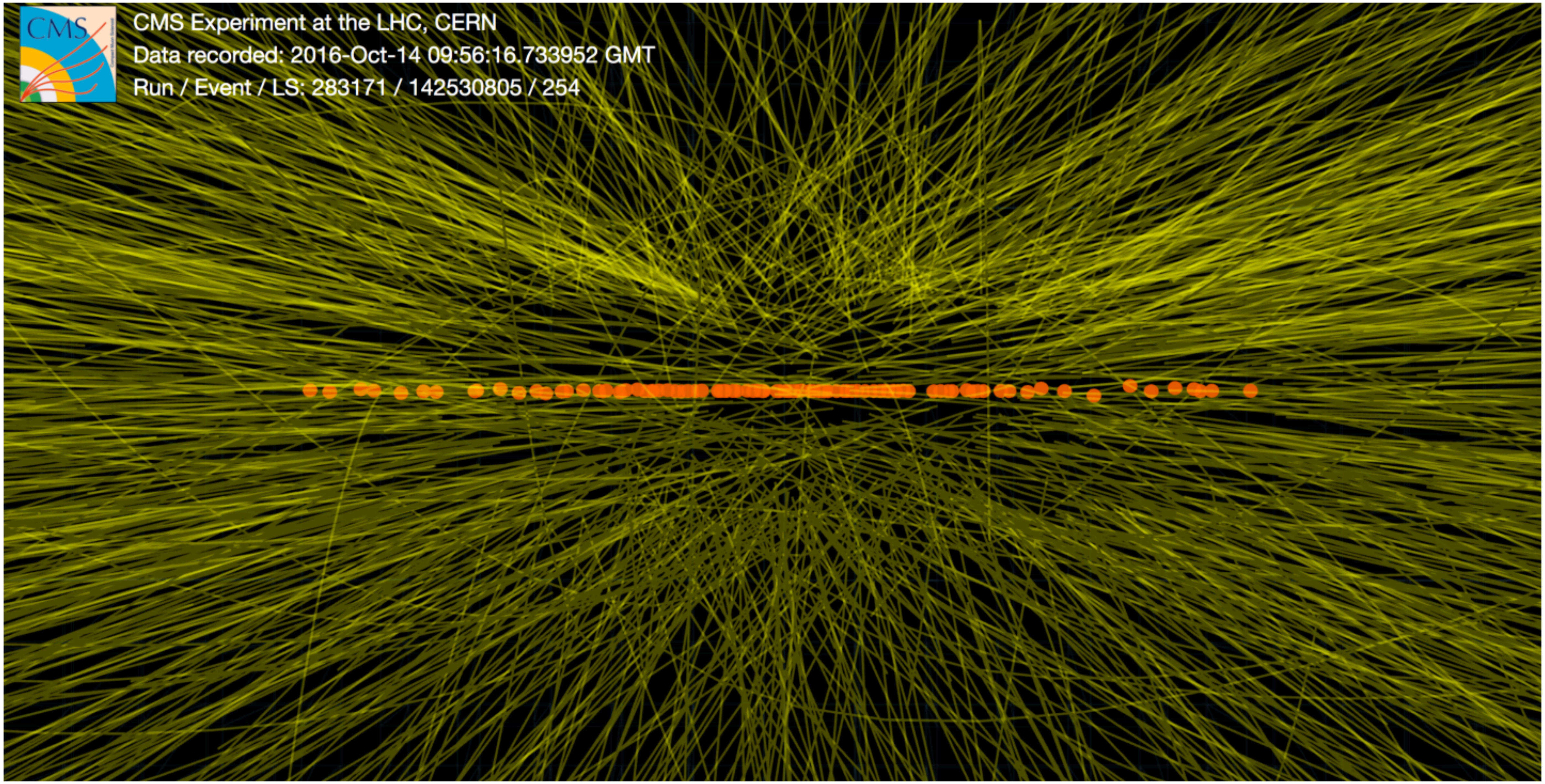


High Luminosity LHC

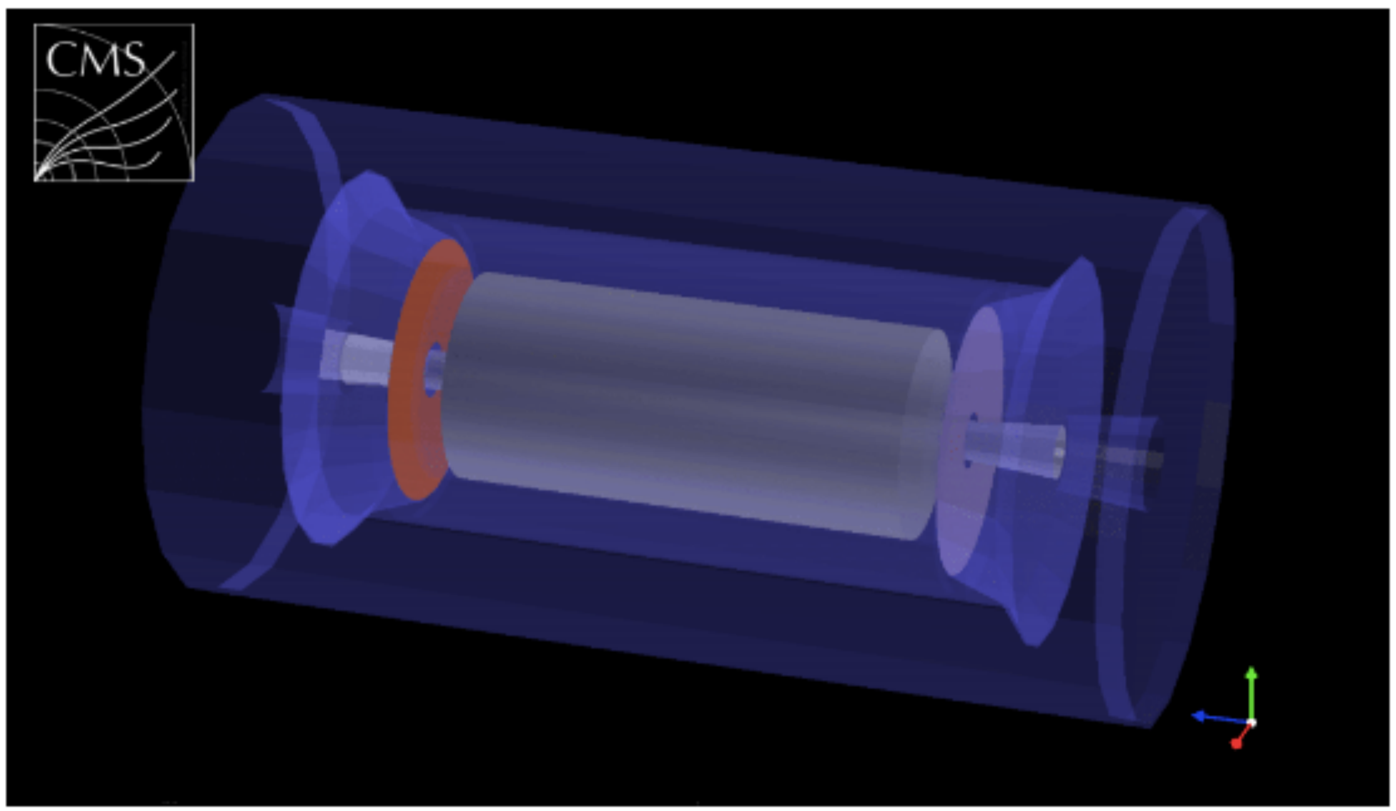
- 10x more data
- Operational around 2026
- new, innovative technology

HL-LHC Issues

Pileup



MIP Timing Detector



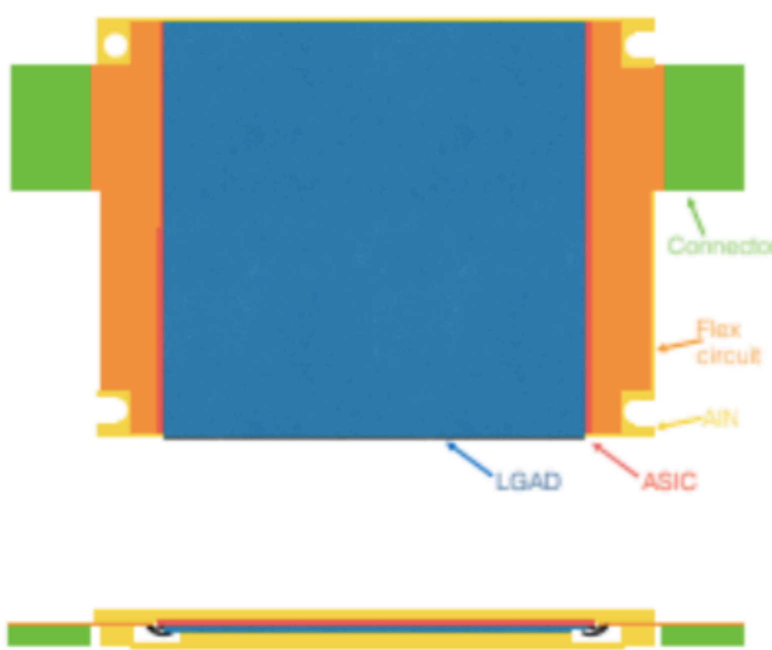
CMSSW generated visualization of MTD encaps (orange) and barrel (gray)

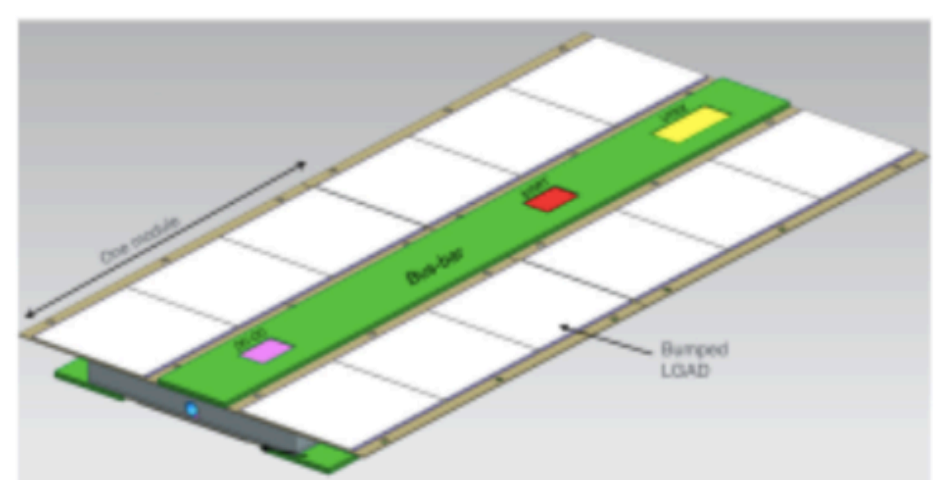
- Detects MIPs (Minimum Ionizing Particles) in space as well as time
- Precision Timing (~30 ps)
- Improve track + vertex reconstruction
- Improve missing pT resolution
- <u>Reduced pileup rate</u>

What is the MTD?

Endcap technology

- Silicon low gain avalanche detector (LGAD) on the bottom
- ASIC in the middle
- Flex circuit
- Surrounded by aluminum compound



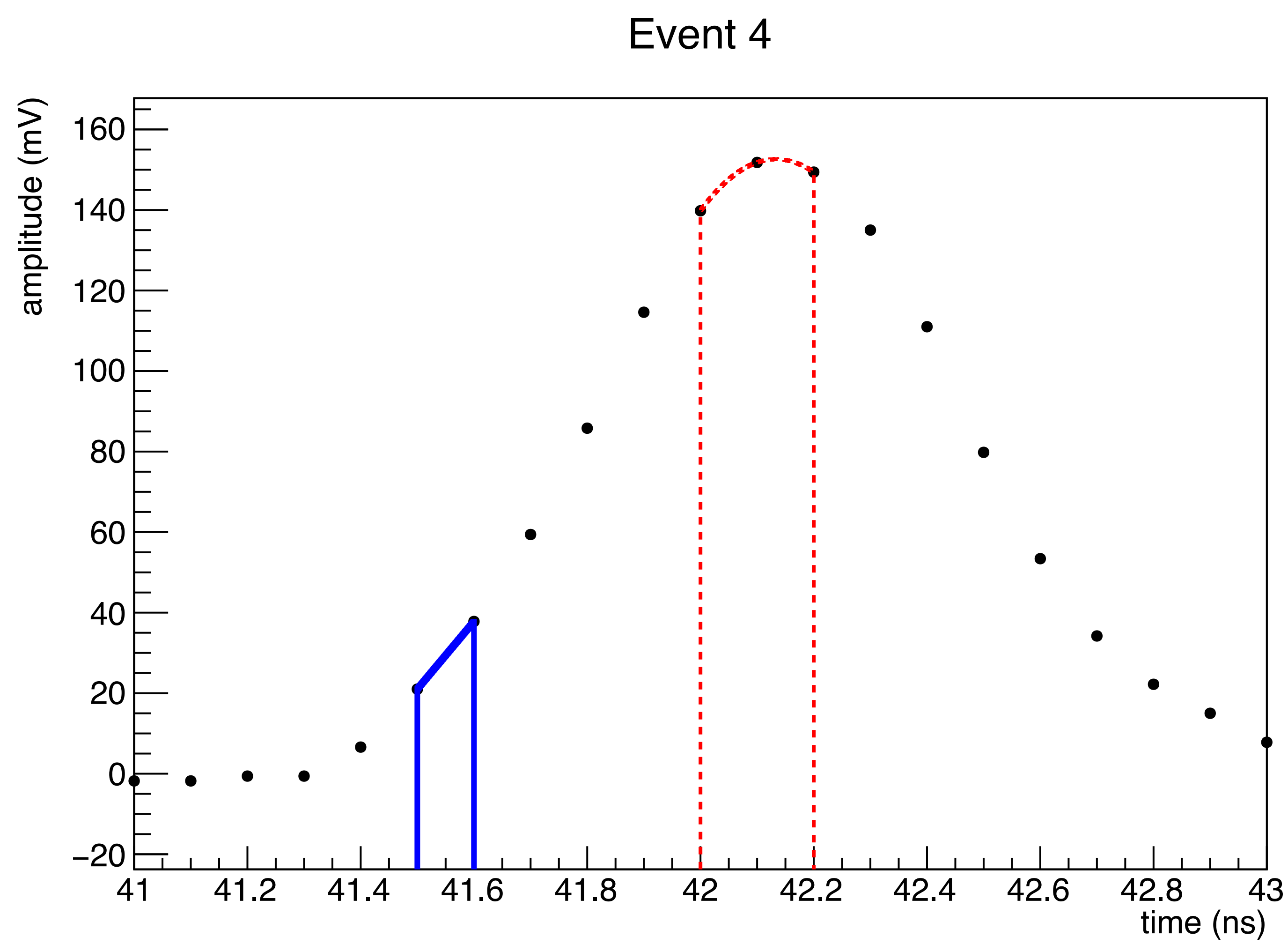


How does the MTD measure time?

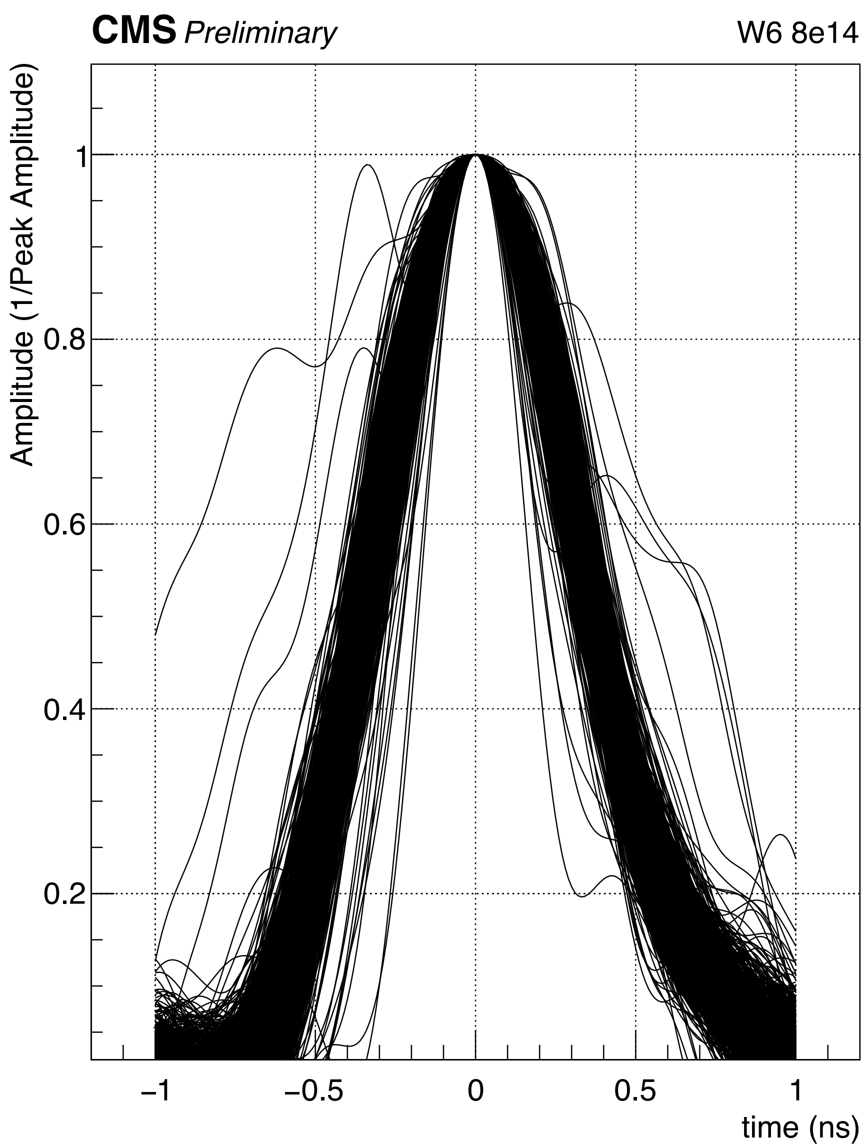
- Project at hand - could we potentially use neural networks?

- Previously:

    - Used reference time, interpolated peak, used CFD

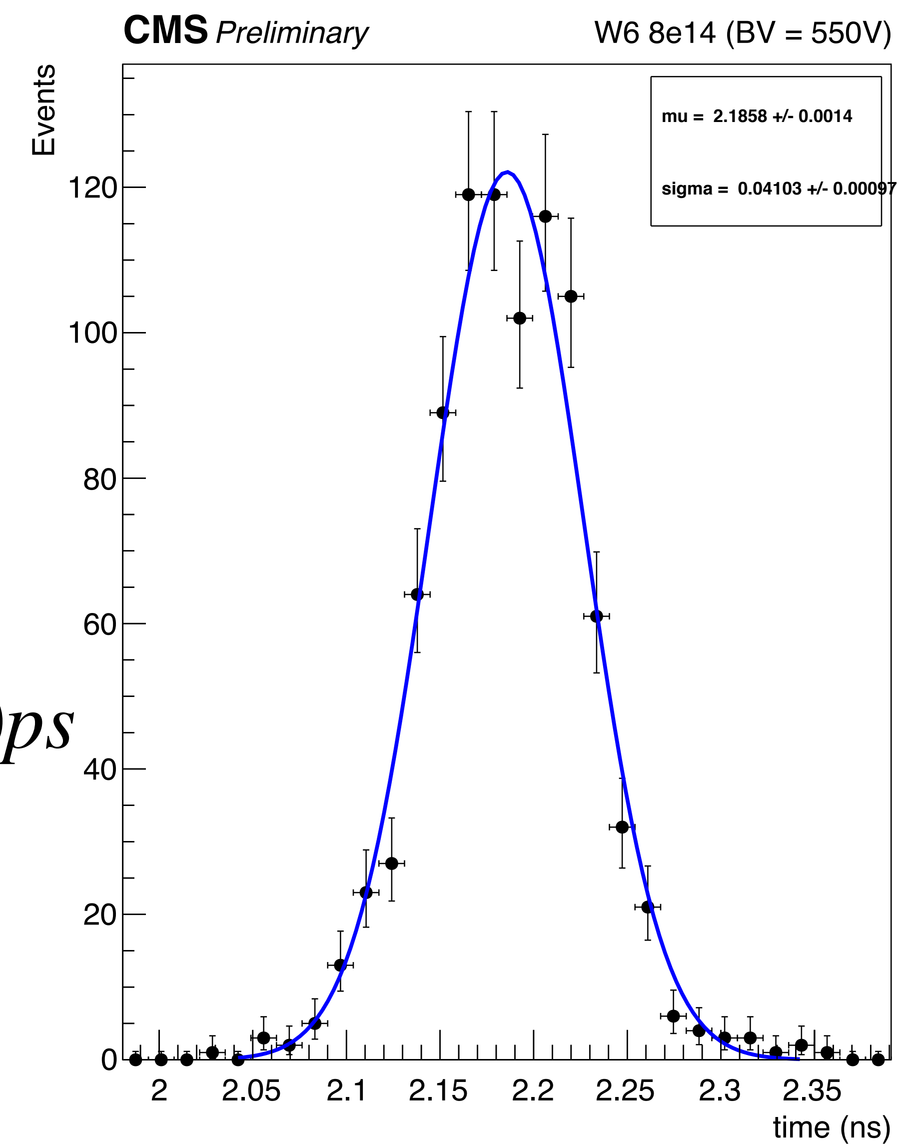


How does the MTD measure time?



- Collect time by CFD for all events to get <u>time resolution</u>

- Goal: $\sigma(t) = \sim 30ps$



Is there a better way?

- Currently using Keras to fit pulse times (inputs) to reference times (labels)
    - Using a moving window to determine if there is a peak (1) or no peak (0)
- Data preprocessing
    - Need to convert vector of sample voltages, sample times and reference times to something more NN friendly
    - Transforms to tensors (matrices) that match shape of input layer

Is there a better way?

Linear stack of layers

128 neurons

```
def baseline_model():
    model = Sequential()
    model.add(Dense(128, input_dim=windows_width, activation='relu'))
#                     v1D(10, 5, activation='relu', input_shape=(len(prep_inputs),1)))
#                     v1D(10, 5, activation='relu'))
#     model.add(MaxPooling1D(3))
#     model.add(Conv1D(160, 10, activation='relu'))
#     model.add(Conv1D(160, 10, activation='relu'))
    #model.add(GlobalAveragePooling1D())
    #model.add(Dropout(0.5))
    #model.add(Dropout(rate = 0.1,seed=100))
    model.add(Dense(128, activation='relu'))
    model.add(Dense(output_dim=maxValue+1, activation='softmax'))
    model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')
    return model
```

Neural Net Architecture

- Dense layers (128 neurons)
- Activation functions
  - Input: relu (rectified linear unit)
  - Output: softmax (for probabilities)
- Loss function: categorical cross entropy
- Optimizer: Adam (adaptive learning rate)

Next Steps

- Improve efficiency of code - slow run time
- Add dropout layers?
- Revise model
  - More hidden layers?
  - Revise activation function in last layer - categorical output (not continuous
- Convolution layers?
- ???