CAF updates (and PID)

ND-GAr Weekly Meeting
11.07.2023

Francisco Martínez López
f.martinezlopez@qmul.ac.uk

What are CAFs?

- **C**ommon **A**nalysis **F**iles are the high level representation of art (FD/ND-GAr) or edep-sim (other NDs) truth and reconstructed outputs useful for analysis.
  - They contain one StandardRecord object per event (== neutrino interaction) with enough information to perform a variety of physics analyses.
  - These files are the inputs to analysis frameworks like CAFAna or MaCh3.
- Existing (TDR era) ND CAFs were produced using parametric reco.
  - The current state of maturity of GArSoft makes possible including proper reconstruction information in the ND analysis samples.
- (ND) CAFs are created using the ND_CAFMaker tool, common to all NDs.
  - It retrieves the truth information from the GENIE event record and uses a collection of "reco branch filler" modules to parse the reco information of the different detectors.

CAF status

- Jeremy's hierarchical CAFs proposal has been finally merged (20/06/2023).
  - The next step is to update the CAFMakers (both FD and ND) to fill the CAFs with the new format.
  - People are already working on the ND-LAr/2x2 parts, that'll make things easier by the time we start filling in the ND-GAr part.
- For ND-GAr the biggest change is the addition of the new common reco branch, supposed to contain the highest level reco information for each detector (more on the next slides).
- The detector-specific branch, with the track and ECAL cluster information, remains almost unchanged.
  - Perhaps we can add more information there if necessary, e.g. MuID clusters.

CAF common reco branch

- Inside `SRCommonRecoBranch/SRInteractionBranch` we will need to create new a vector of `SRInteraction` objects to place the interactions from GArSoft reconstruction (could be more than one as now there is one `StandardRecord` object per trigger).

- Each `SRInteraction` contains the reconstructed vertex position, the hypotheses for the neutrino direction, flavour and energy and a collection of reconstructed particles.

```
class SRInteraction
  {
    public:
      /// Reconstructed vertex location (if any)
      SRVector3D vtx;

      /// Hypotheses for this interaction's parent particle direction
      SRDirectionBranch dir;

      /// Hypotheses for this interaction's neutrino identity
      SRNeutrinoHypothesisBranch nuhyp;

      /// Hypotheses for this interaction's neutrino energy
      SRNeutrinoEnergyBranch Enu;

      /// Collections of reconstructed particles
      SRRecoParticlesBranch part;
  };
```

CAF common reco branch

- Inside `SRCommonRecoBranch/SRInteractionBranch` we will need to create new a vector of `SRInteraction` objects to place the interactions from GArSoft reconstruction (could be more than one as now there is one `StandardRecord` object per trigger).

- The `SRRecoParticleBranch` contains vectors of `SRRecoParticle` objects, each with PDG, energy, momentum and position information.

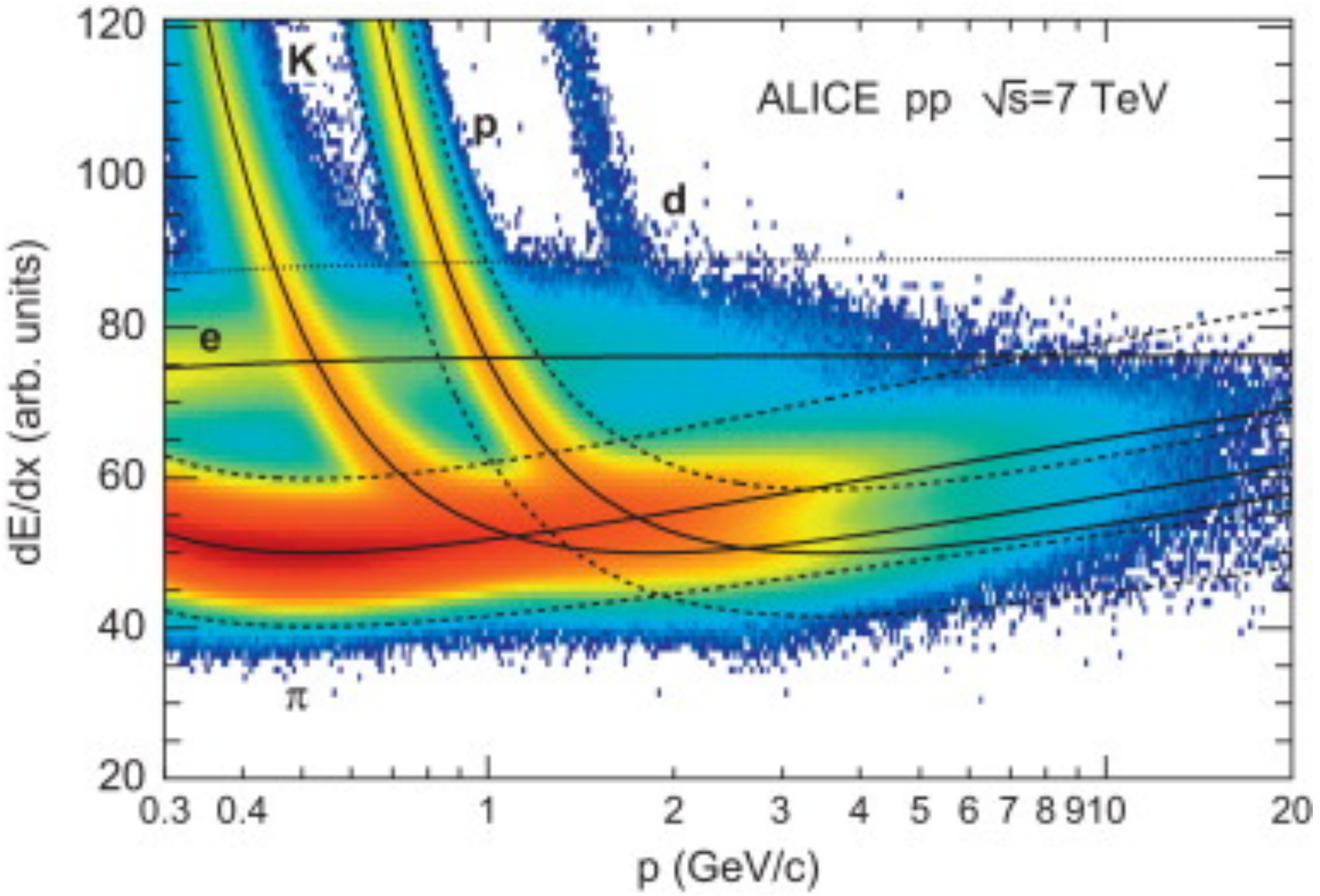
```
class SRRecoParticle
  {
    public:
      bool        primary  = false;                ///< Is this reco particle a "primary" one (i.e. emanates directly from the reconstructed vertex)?

      int         pdg      = 0;                    ///< PDG code inferred for this particle.
      int         tgtA     = 0;                    ///< Atomic number of nucleus this particle was reconstructed in (useful for, e.g., SAND)

      float       score    = NaN;                  ///< PID score for this particle, if relevant

      float       E        = NaN;                  ///< Reconstructed energy for this particle
      PartEMethod E_method = PartEMethod::kUnknownMethod;   ///< Method used to determine energy for the particle
      SRVector3D  p;                               ///< Reconstructed momentum for this particle

      SRVector3D  start;                           ///< Reconstructed start point of this particle
      SRVector3D  end;                             ///< Reconstructed end point of this particle, if that makes sense

      bool        contained = false;

      TrueParticleID truth;                        ///< Associated SRTrueParticle, if relevant (use SRTruthBranch::Particle() with this ID to grab it)
  };
```

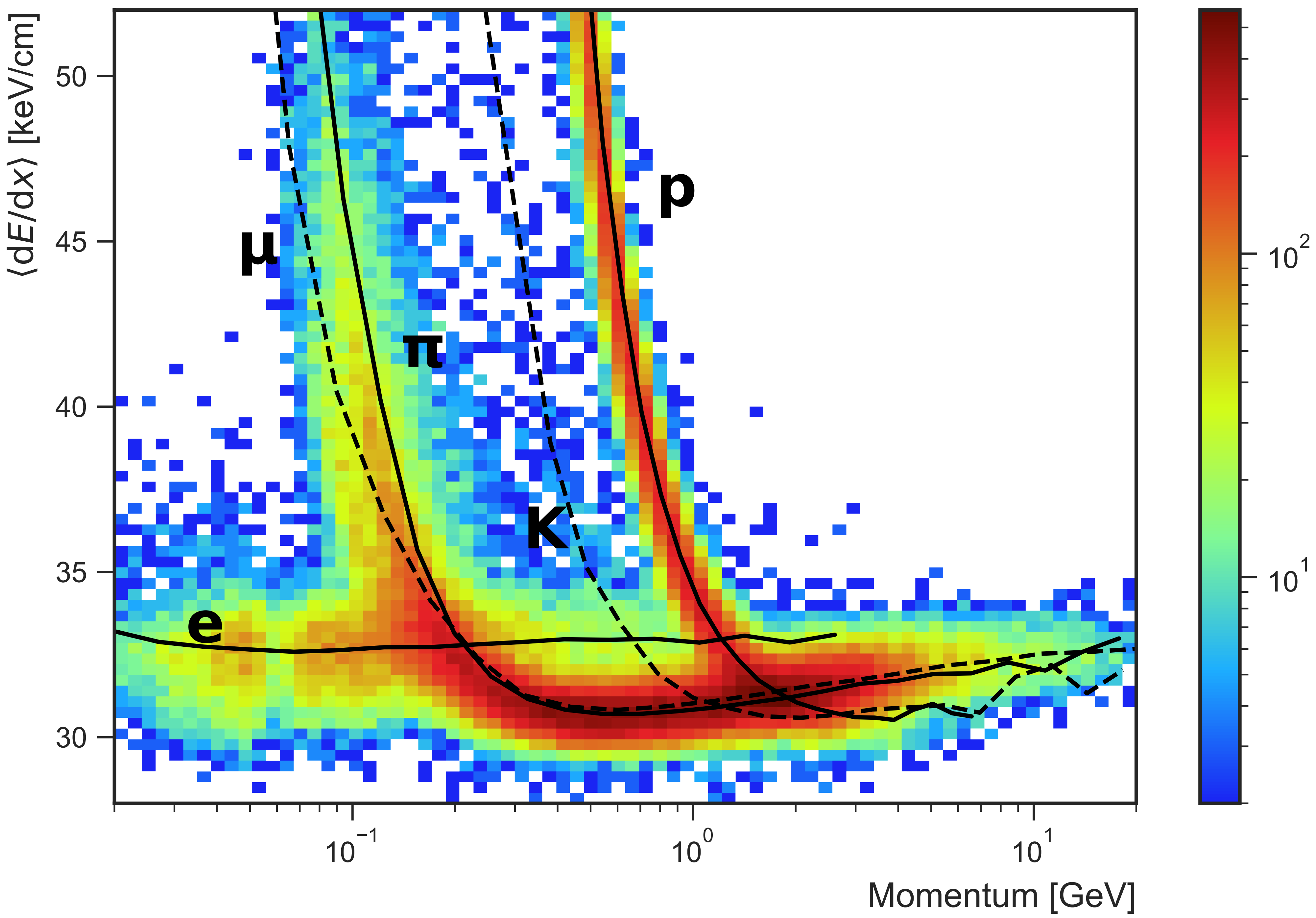
CAF common reco branch

- Inside `SRCommonRecoBranch/SRInteractionBranch` we will need to create new a vector of `SRInteraction` objects to place the interactions from GArSoft reconstruction (could be more than one as now there is one `StandardRecord` object per trigger).
- Each `SRInteraction` contains the reconstructed vertex position, the hypotheses for the neutrino direction, flavour and energy and a collection of reconstructed particles.
- The `SRRecoParticleBranch` contains vectors of `SRRecoParticle` objects, each with PDG, energy, momentum and position information.
- We need to think about how to fill the blanks.
  - What do we have available in GArSoft to estimate the direction/energy/type of the neutrino?
  - What methods can we use to reconstruct the energy or the PDG of the particles?

dEdx truncated mean

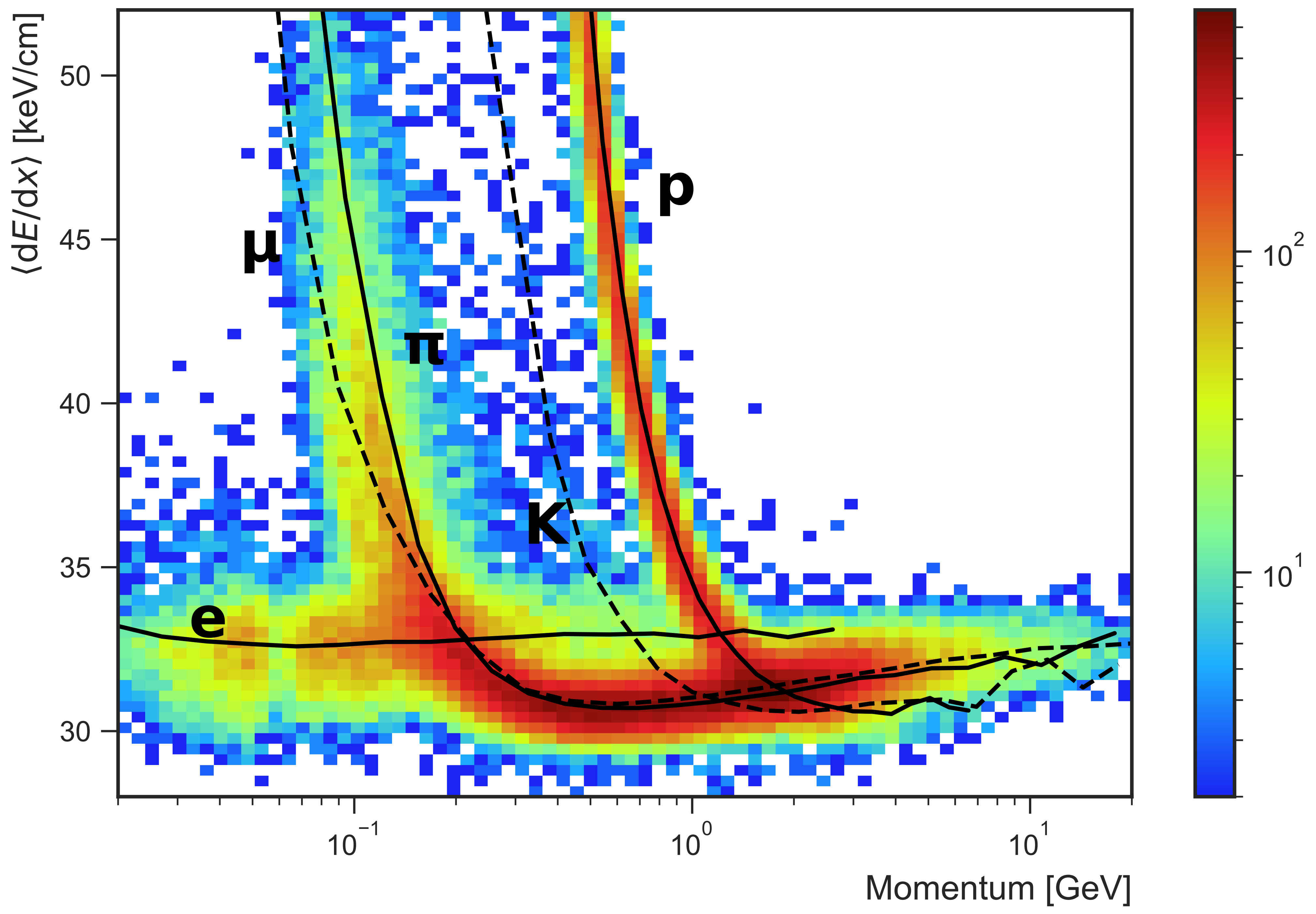
- Using dEdx vs residual range is problematic, as finding the correct track end impacts the PID score considerably (see last CM talk).
- Follow same method as in ALICE, take truncated mean of the 60% lowest energy TPC clusters for each track.





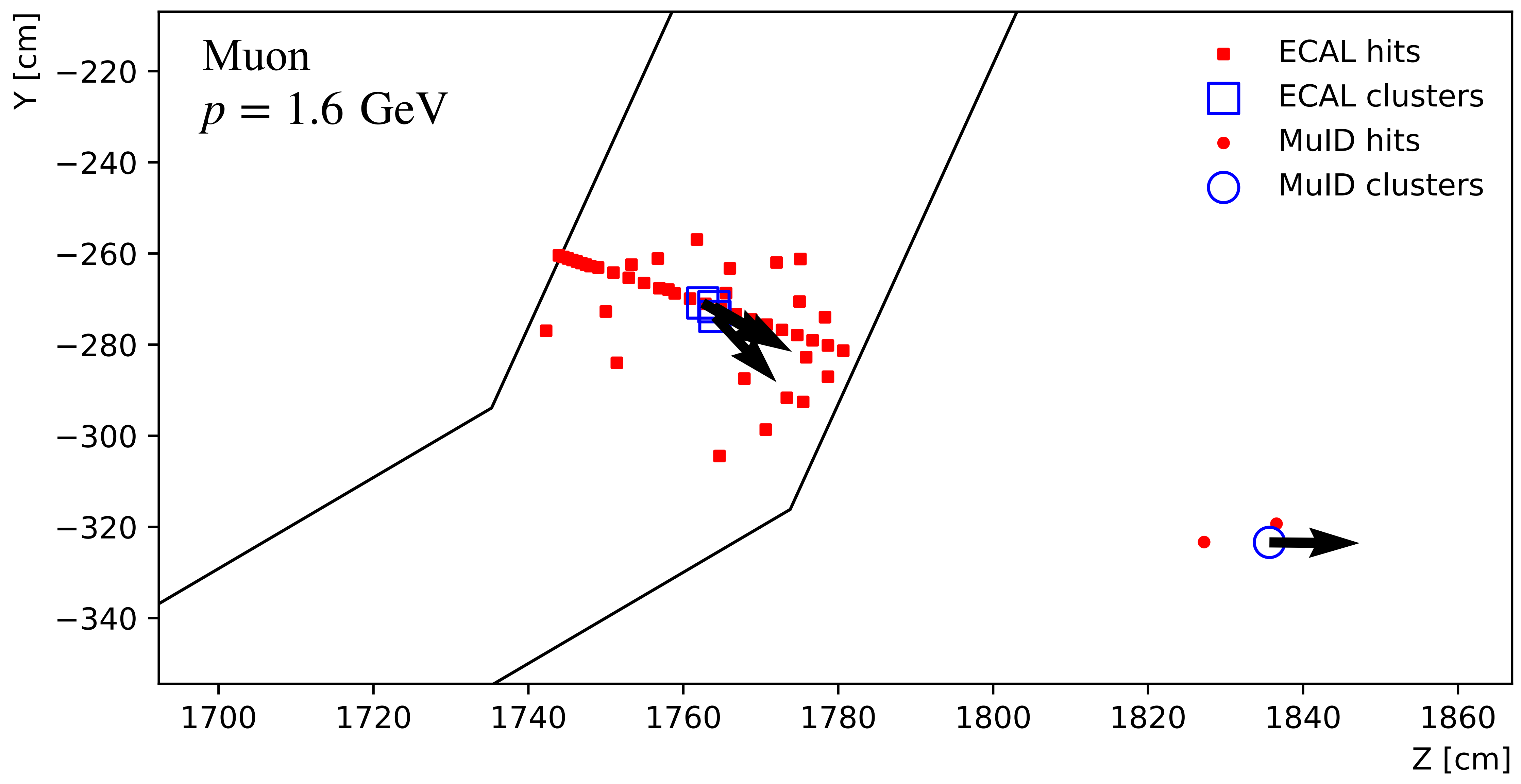
dEdx truncated mean

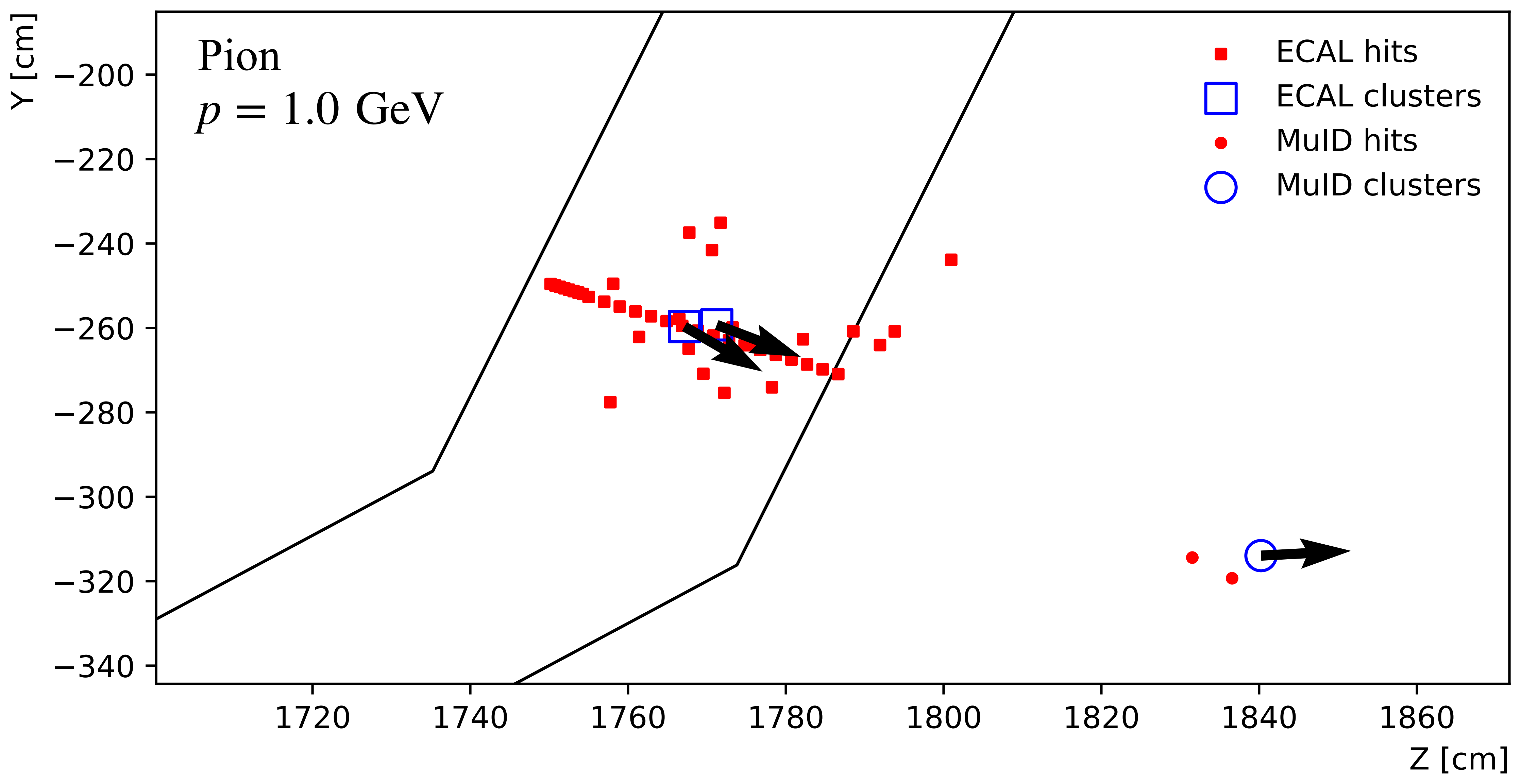
- Using dEdx vs residual range is problematic, as finding the correct track end impacts the PID score considerably (see last CM talk).
- Follow same method as in ALICE, take truncated mean of the 60% lowest energy TPC clusters for each track.
- Doing this allows us to handle also non-contained tracks.
- Further study is needed to optimise and understand the separation power of this method in our case.
- I started implementing this in GArSoft, more updates soon.



Muons/pions in the ECAL

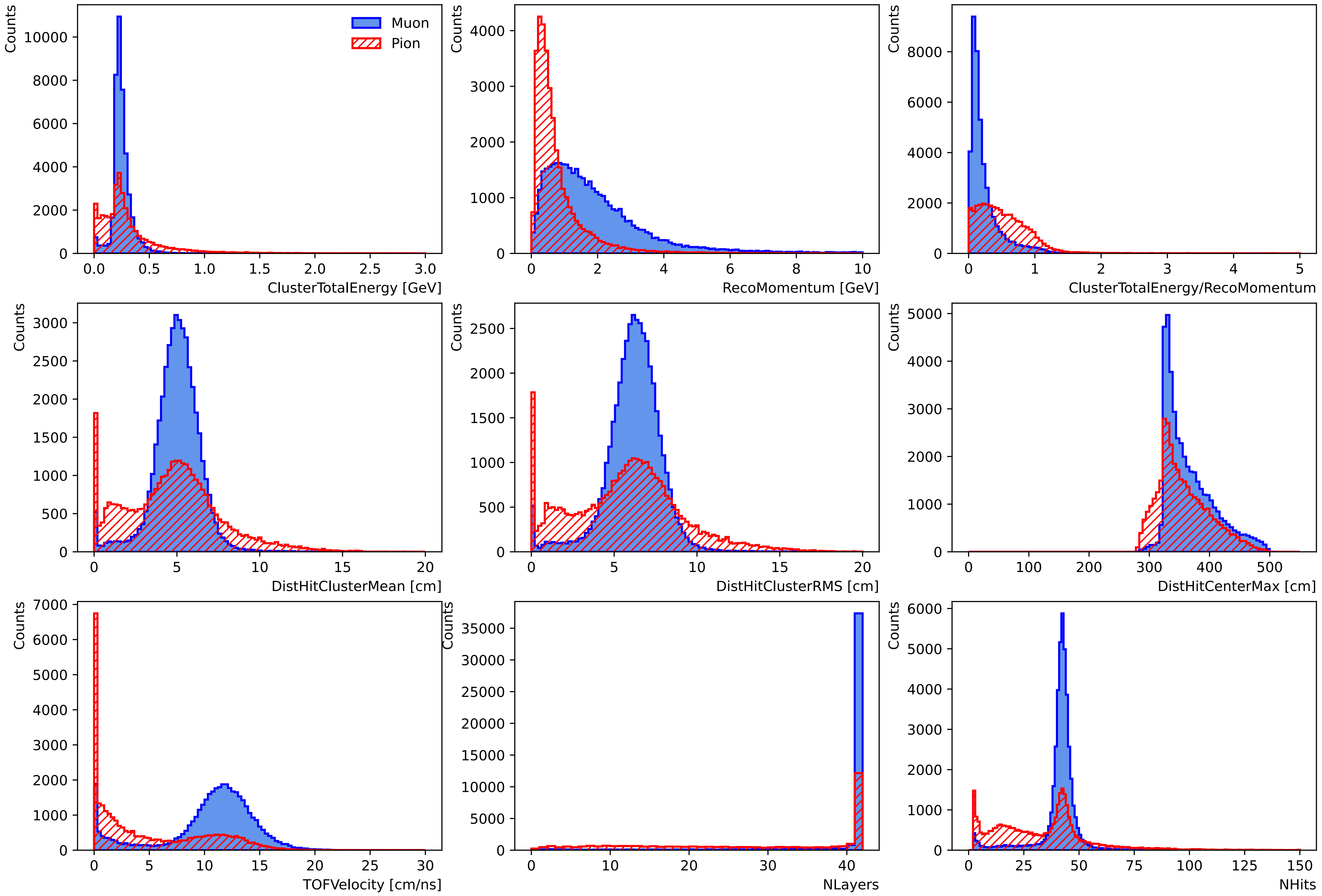
- Using the truncated dEdx to separate muons and pions is not an option, the curves overlap in the relevant momentum range.
- I tried to extract some relevant features from the ECAL for muons and pions using a FHC sample, to use them as the input of a BDT.
- Ideally we should also include information from the MuID (work in progress).



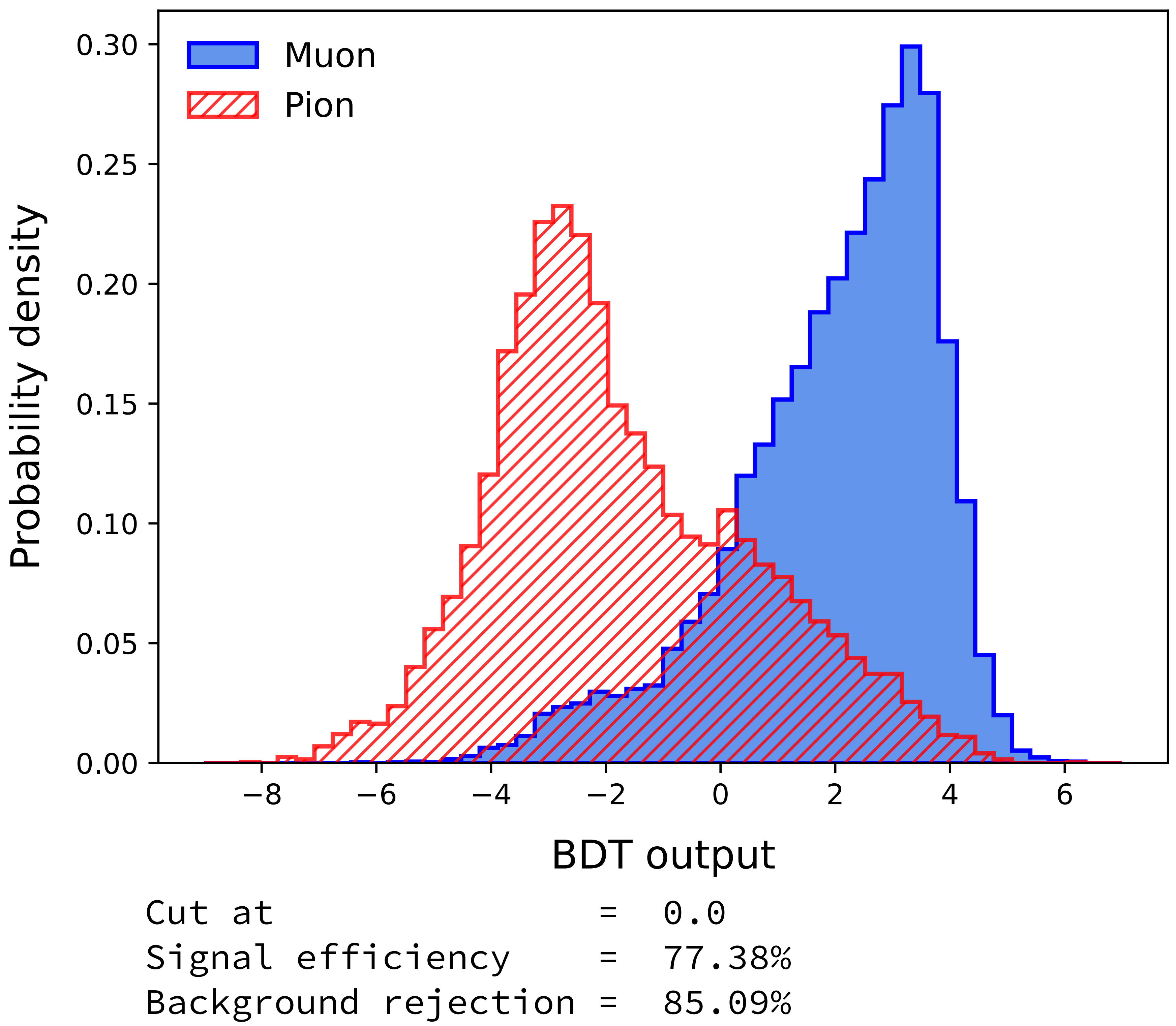


Muons/pions in the ECAL

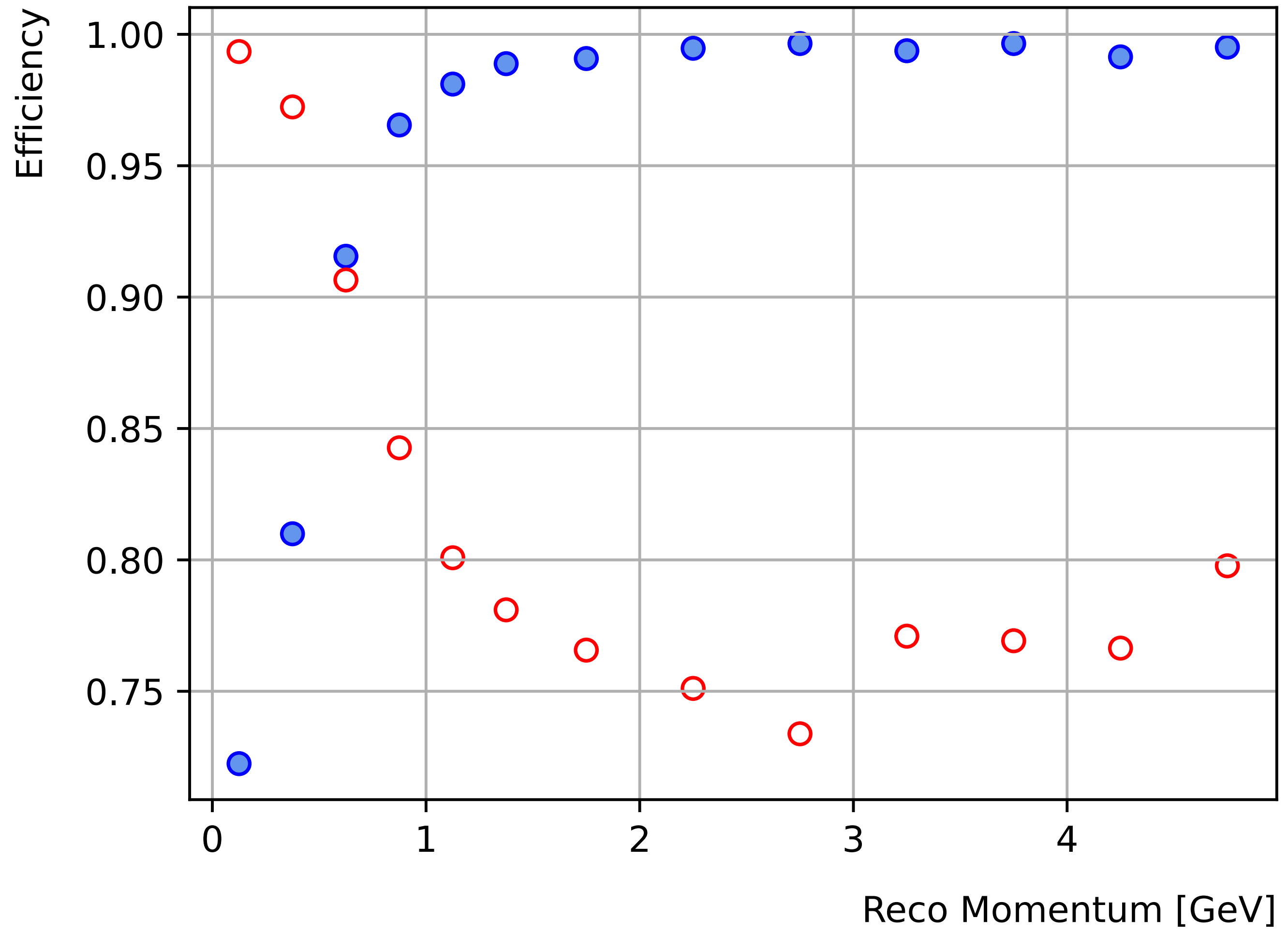
- As we can have more than one cluster associated to a reco track I chose a set of variables defined for each track:
  - ECAL total energy.
  - Mean and RMS distance between hits and cluster main axis.
  - Max distance between hits and TPC centre.
  - Velocity from ToF.
  - Number of ECAL layers with hits.
  - Total number of ECAL hits.

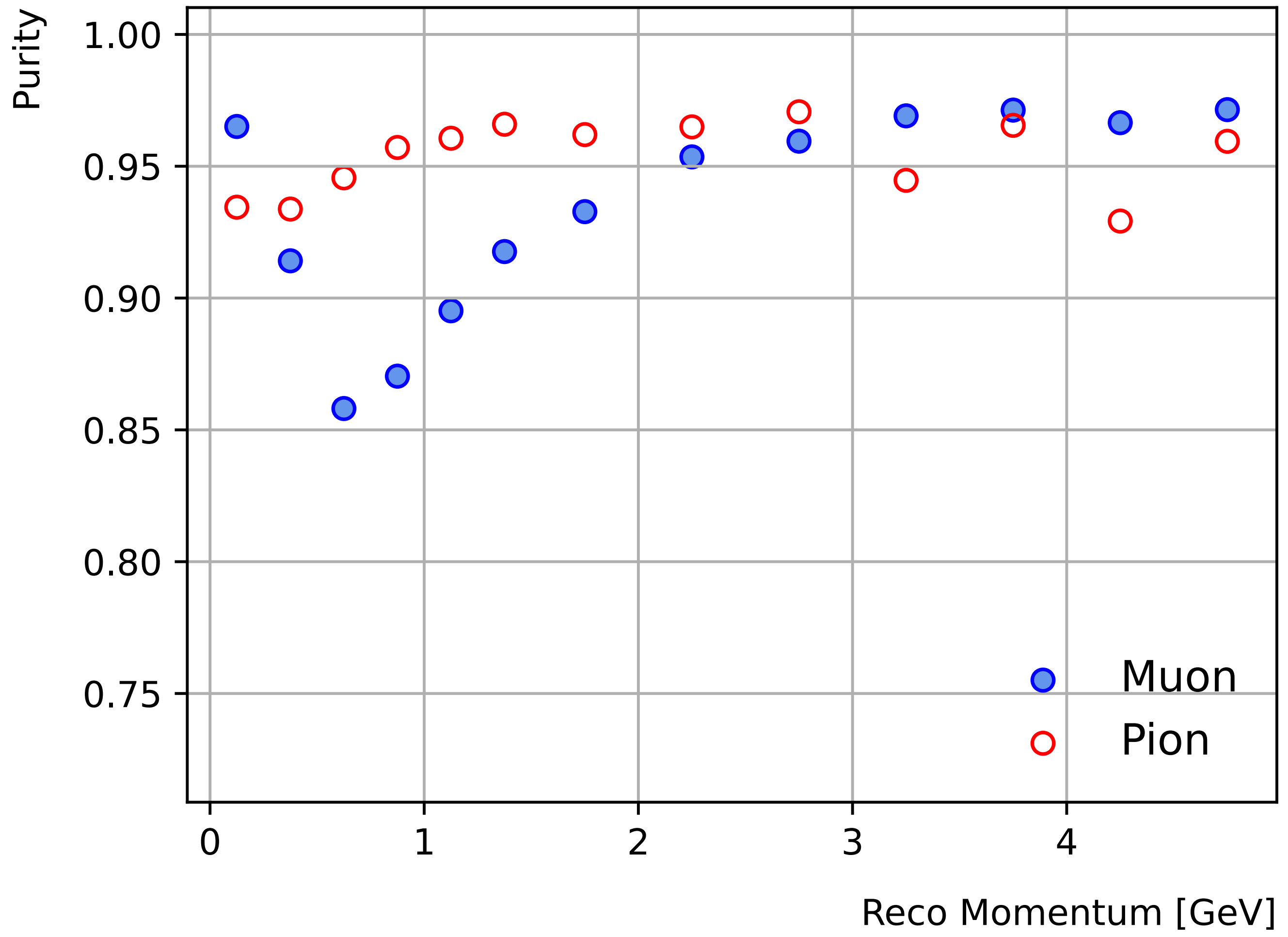


Muons/pions in the ECAL



- Decent separation from ECAL only.
- By adding the MuID I expect to increase the purity at high momentum.





Feature	ClusterEnergy/ TrackMomentum	DistHitCluster mean	DistHitCluster RMS	DistHitCenter max	TOFVelocity	NLayers	NHits
Importance	0.1972	0.0745	0.0707	0.0916	0.3534	0.1454	0.0673

MuID reco hits

- There was an outstanding bug affecting the position of the reconstructed MuID hits.
  - When compared to the simulated hits (pre-digitisation) there was a significant difference between the two.
- The problem was in the segmentation algorithm, need to make PR with the changes.

Conclusion & next steps

- With the new hierarchical StandardRecord already available (v03_00_00) the work of updating the ND CAFMaker can start.
  - Some minor changes are needed to move what ND-GAr had in the previous CAF version to the detector-specific branch.
  - For the common reco branch we still need to address some questions related to the GArSoft reconstruction.
- The truncated mean dEdx method should be implemented as a module part of the reco chain and the proton score stored as a reconstruction data product.
- For the muon/pion separation, we'd need associations between the track and the MuID clusters (or between ECAL clusters and MuID clusters).
- We are still missing the direction/energy/type of the neutrino.