

CNN-based EM / Track ID

P. Płoński, D. Stefan, R. Sulej

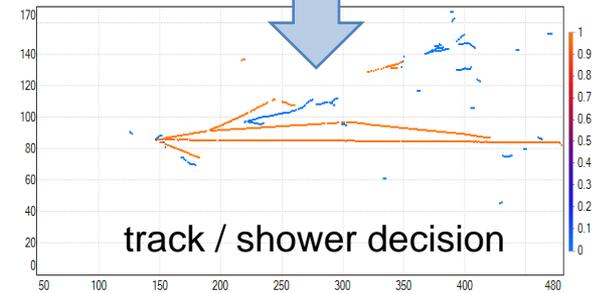
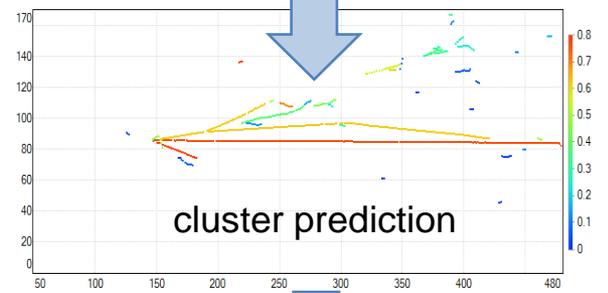
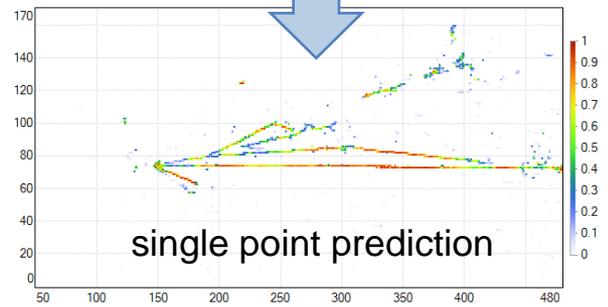
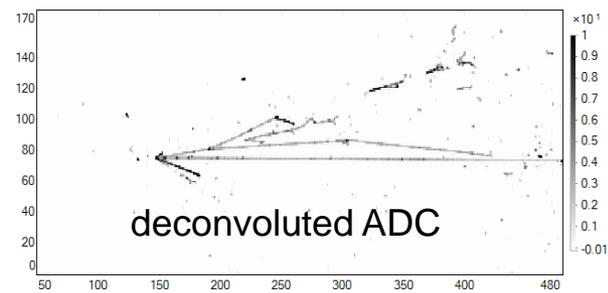
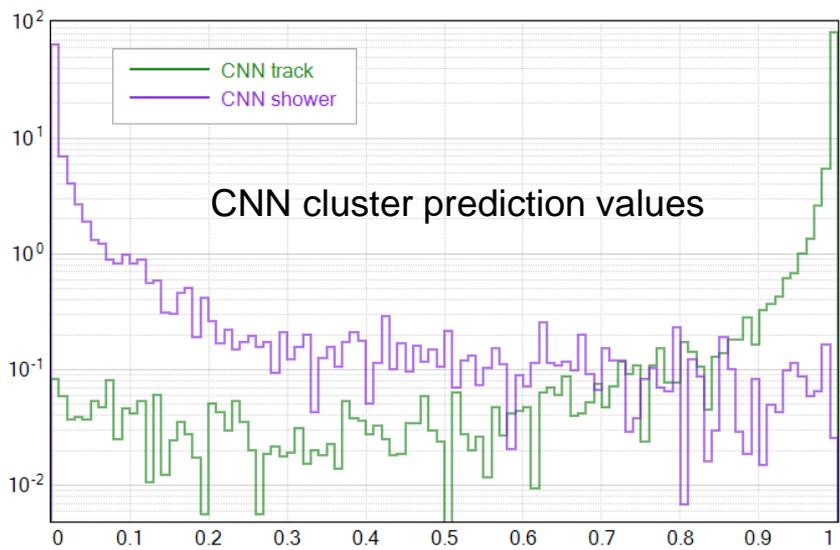
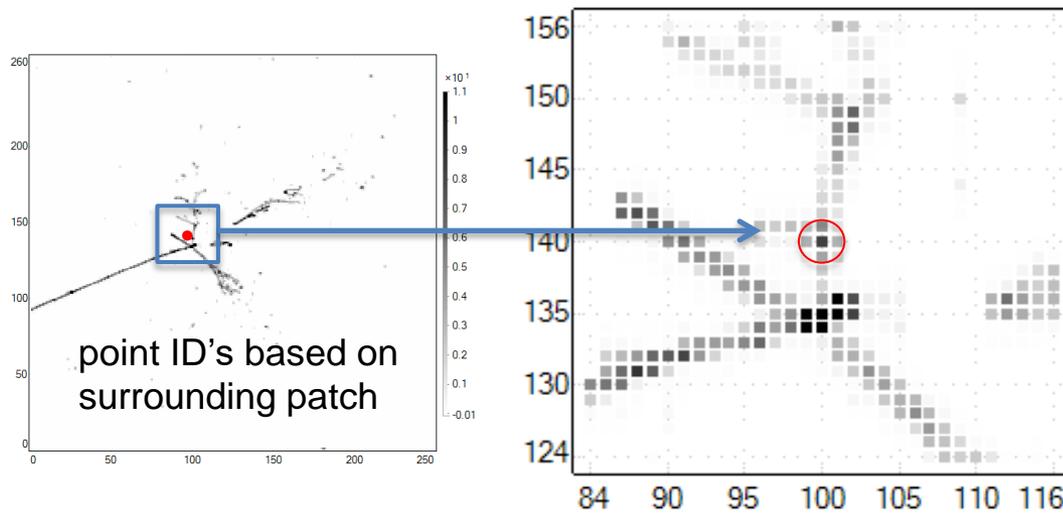


- EM vs. track-like cluster ID, basic idea
- update on implementations in LArSoft
- target developments for the next weeks

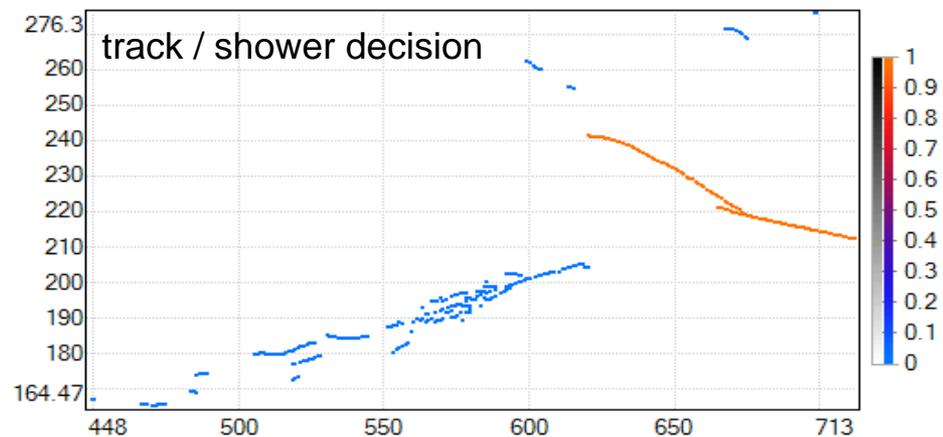
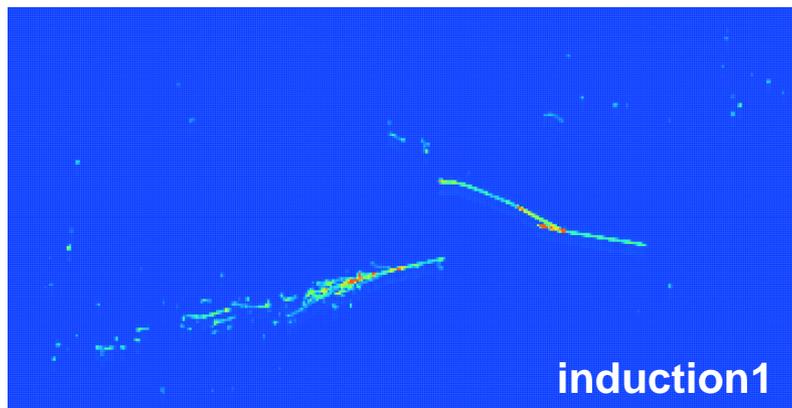
Many slides reused from previous meetings, sorry for those who have already seen this!

EM-like / track-like cluster identification flow:

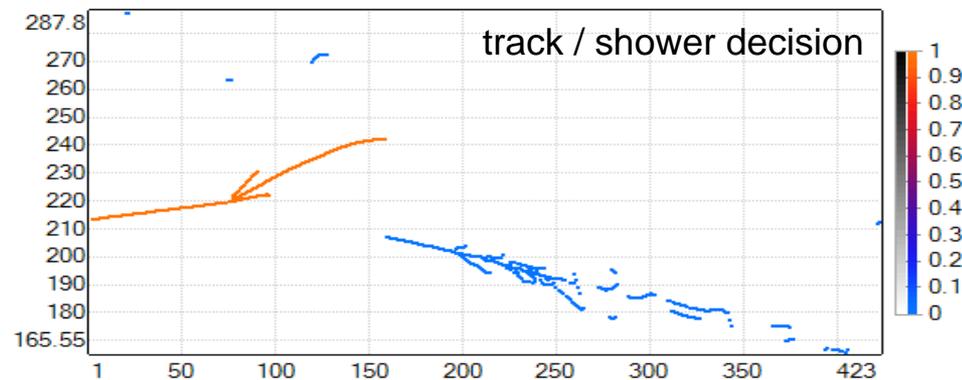
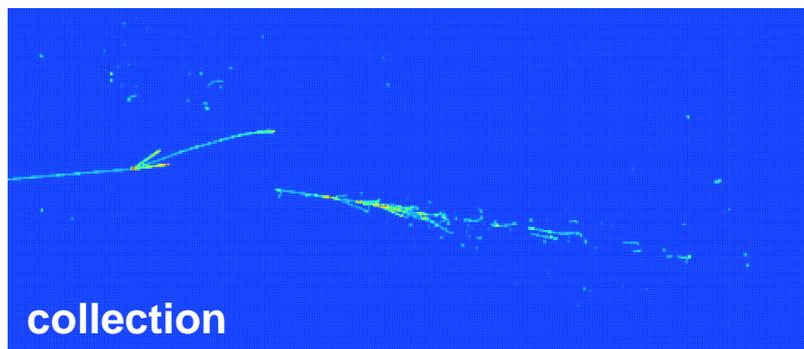
(as usual, old CNN on these pictures!)



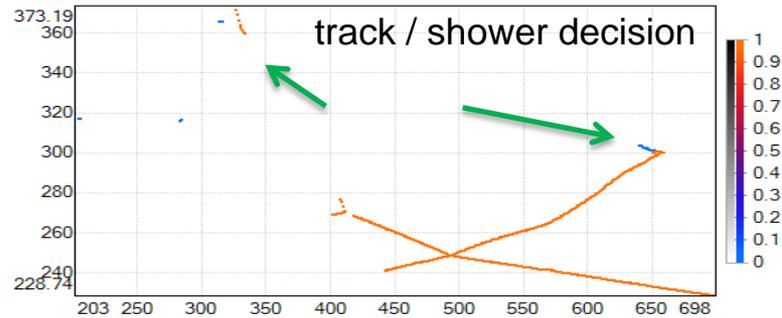
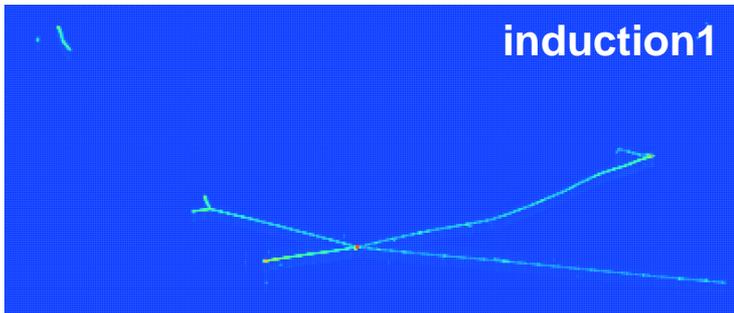
CNN results as of today: π^+ 2 GeV/c in protoDUNE SP



All OK

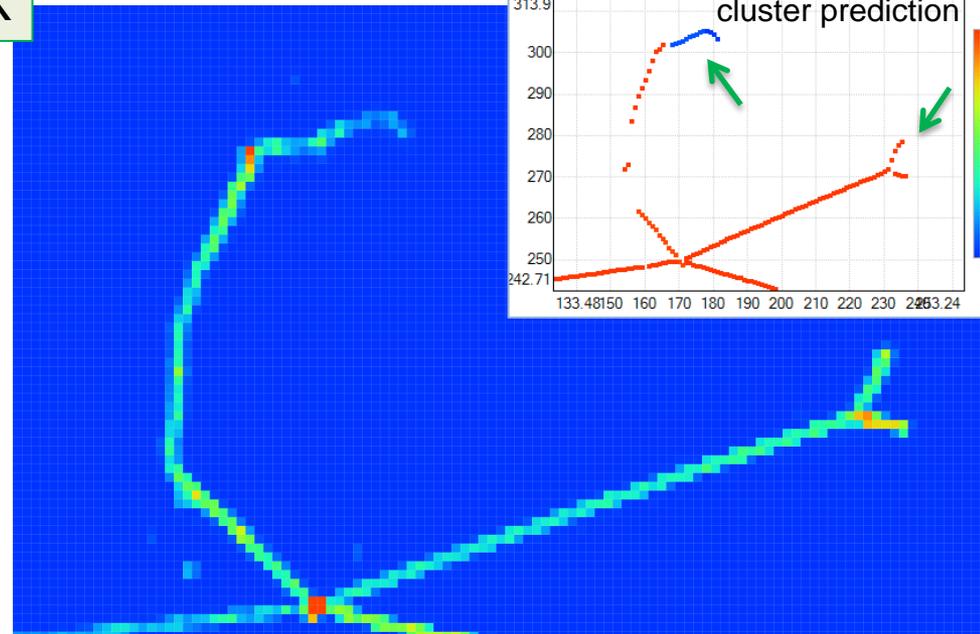
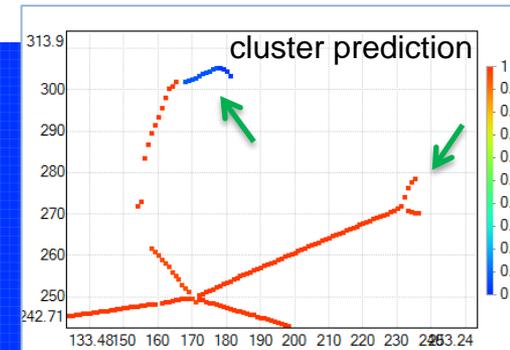
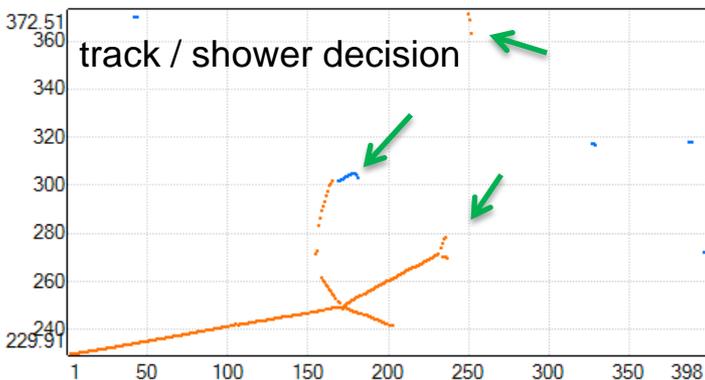


CNN results as of today: π^+ 2 GeV/c in protoDUNE SP

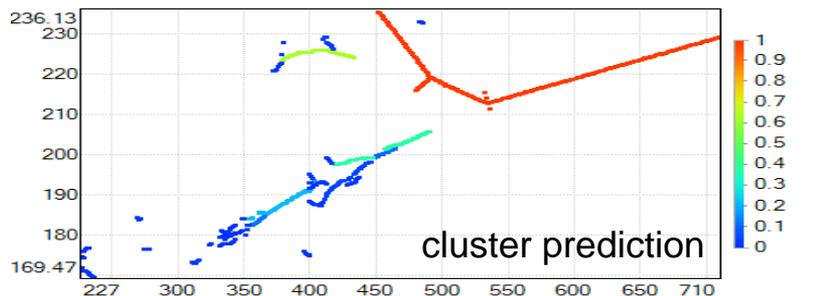
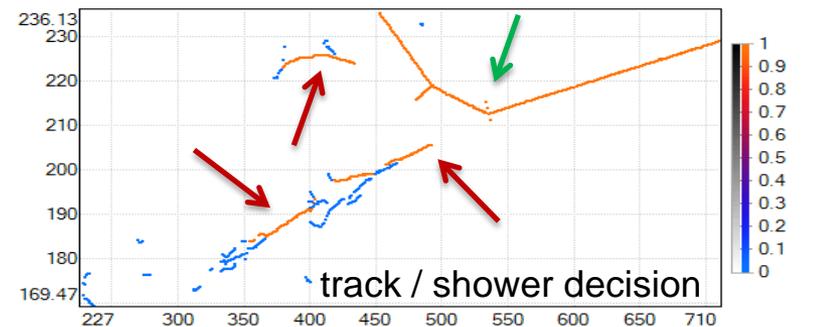
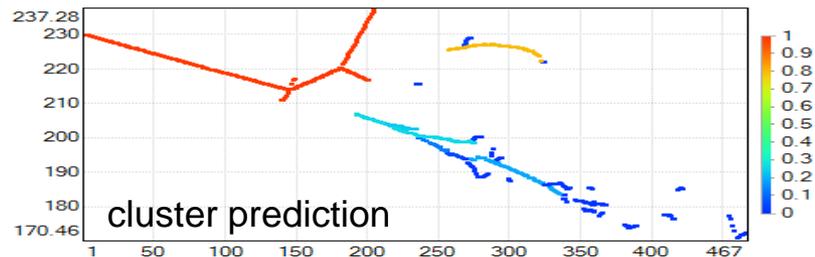
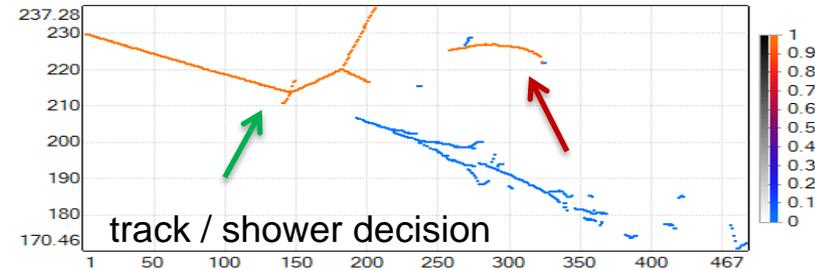
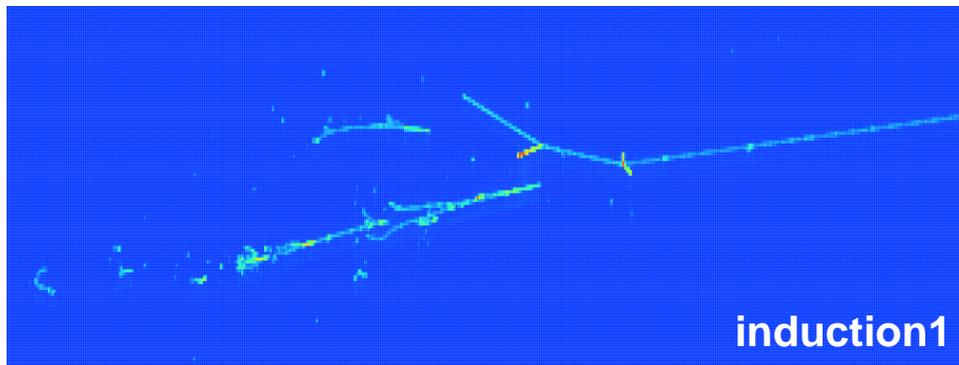
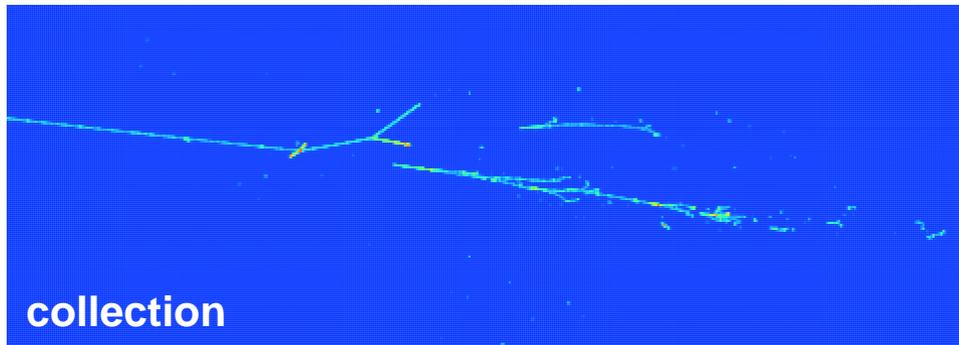


All OK

Even though CNN was not specially tuned for Michel's, the prediction values are pretty „decided”



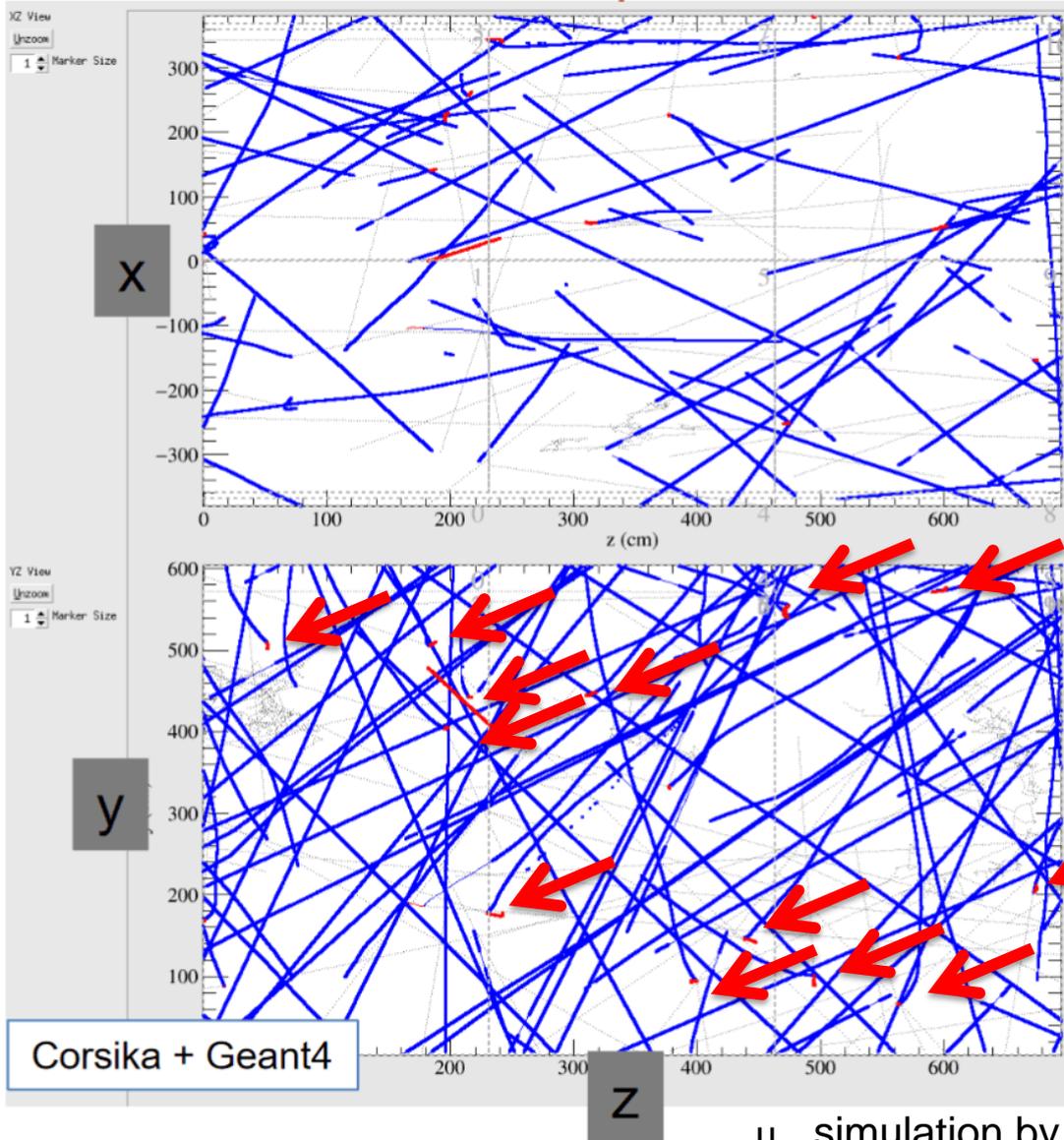
CNN results as of today: π^+ 2 GeV/c in protoDUNE SP



The most confused one...

- many *near-the-threshold* prediction values
 - need simple work on threshold optimization – can be good to get familiar with the tools
- may be also limited by patch size/resolution

Cosmic muons in protoDUNE



Stopping muons are important for protoDUNE's calibration (various analyses are coming).

Many stoppers per each event!

Here the uBoone approach is very reasonable:

RNN for stopper localization.

μ simulation by Elisabeth, picture from Dorota's slides

CNN as of today

Cluster classification (ClusterCrawler as input, decision made of hit classification)

CNN: **96.2% track / 96.6% EM** correct cluster ID rate (2GeV/c π^+ in protoDUNE)

- module for tagging clusters and unclustered hits pushed yesterday

usual mistake sources:

- most cases: complicated configurations, especially if on the image boundaries
- some orientation dependence: more difficult recognition for particles if direction strictly row or column of pixels → less downsampling may help here
- long track-like electrons
- too small patch (important context not seen) / low drift resolution (electron features downsampled)
- sometimes clustering makes its own mistake and merges two objects of different ID...
- *seems resolved now*: short hadron near cascade / vertex

→ large training set: >5M patches, many topologies: no overtraining at all!

→ trained on collection and induction views together (can do dedicated models, but prefer single one until there is well simulated difference between views)

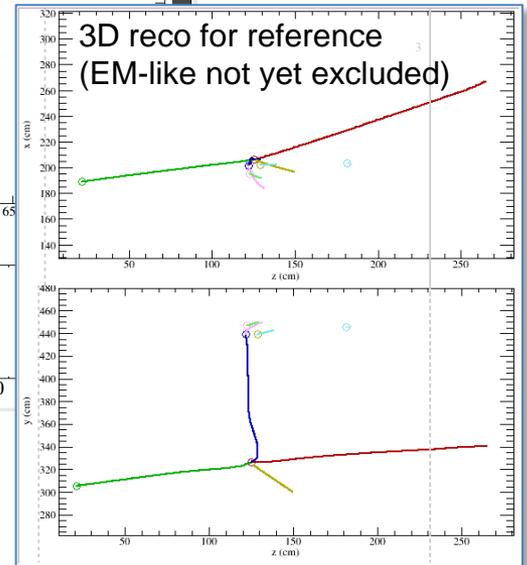
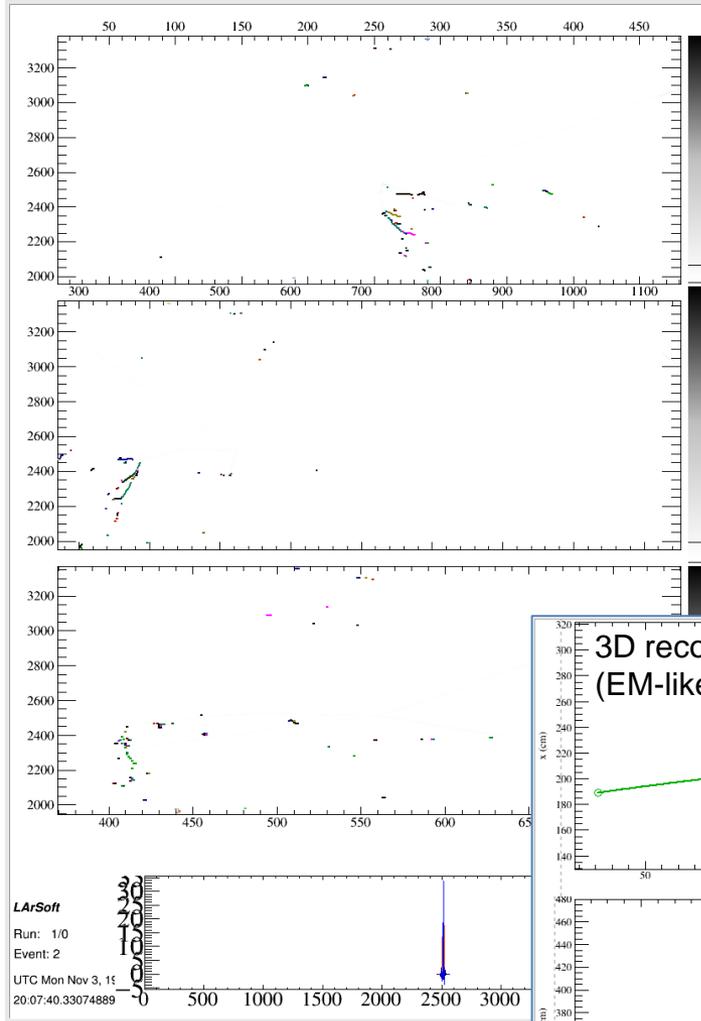
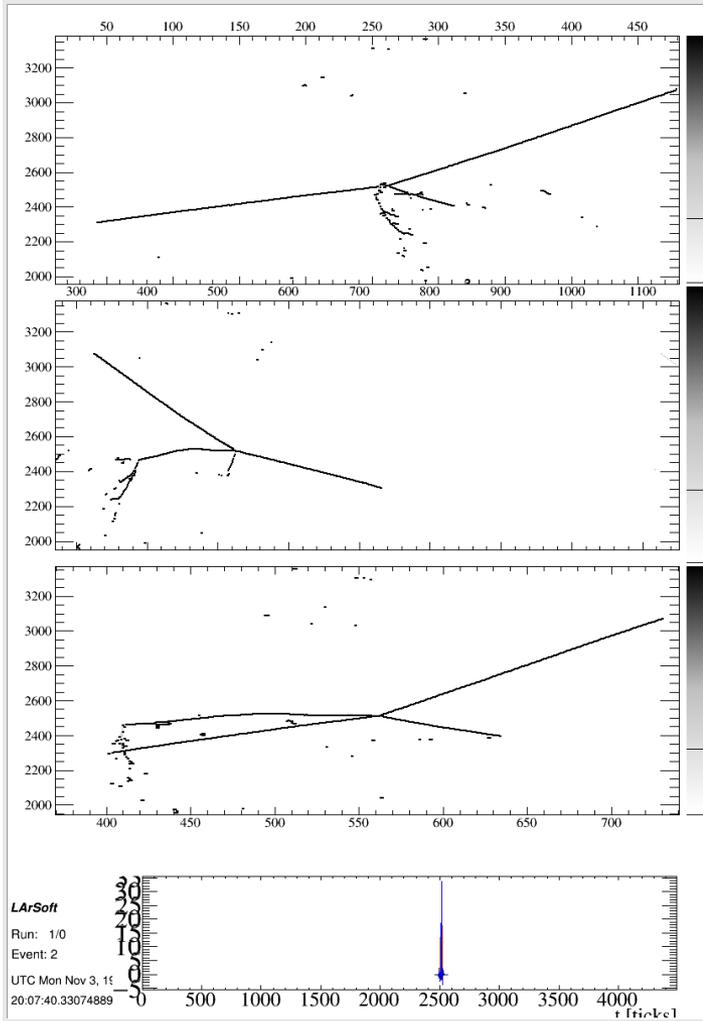
→ next: increase resolution in drift direction, increase noise, [try applying on real data](#)

→ next (waiting as well): vertex classification, similar approach, [priority: \$\mu \rightarrow e\$ decay points, what else?](#)

all hits



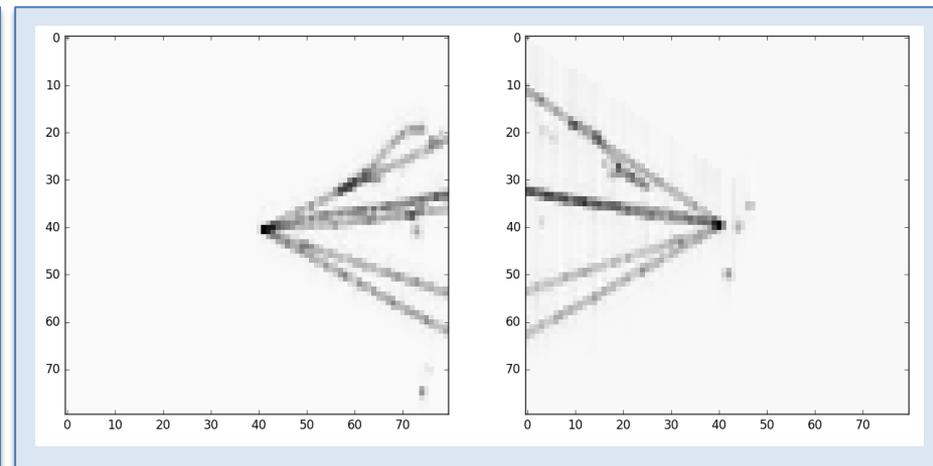
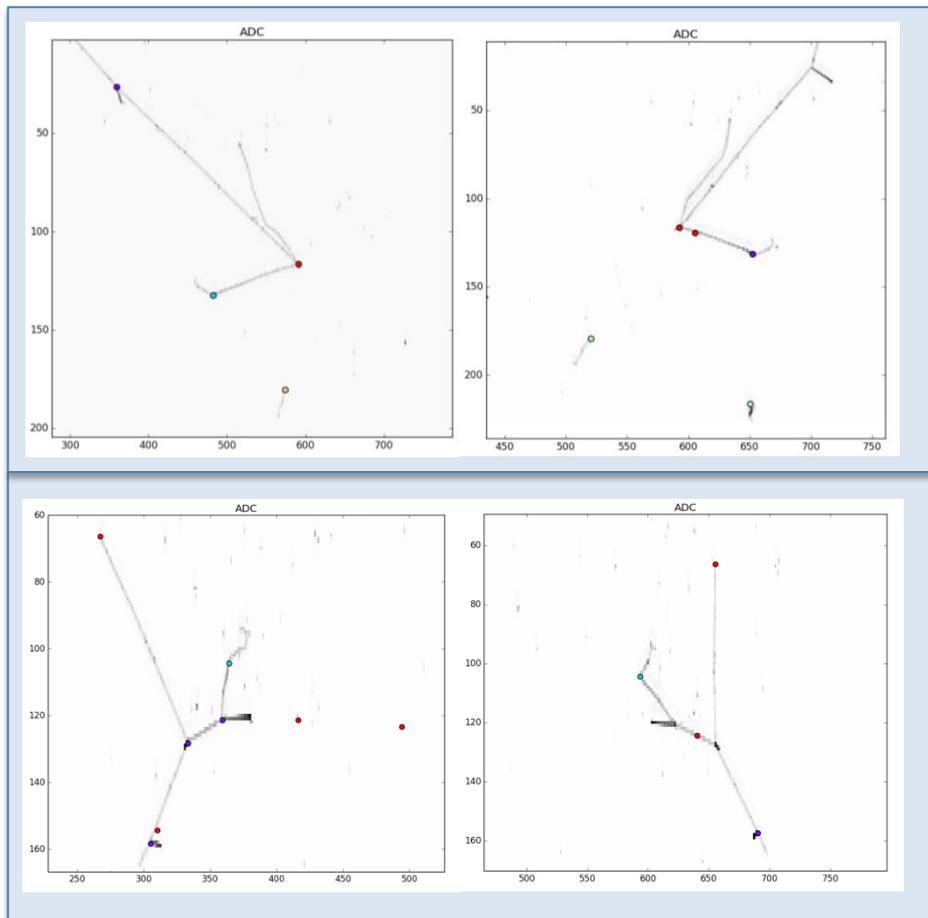
EM-like parts



EmTrackClusterId module

Vertex classification: similar „basic block” to EM/track ID

- Vertex identification
 - support tracking with **interaction/decay finding**
 - **select EM shower starting points** (not trivial in low energy)
- Neutrino interaction classification
 - force classifier to be focused on the vertex features
 - **try to be sensitive to the „gap”** in full neutrino events



- need more events to build training set (only 1 training image pair/triplet per 1 event)
- more complex (than very simple) CNN may be needed
- uses larger patch around the vertex and less downsampled drift
- more careful when producing data files to avoid really huge volumes...

data preparation code being validated (still some vtx missed, threshold to be tuned for reasonable visibility criteria, ...)



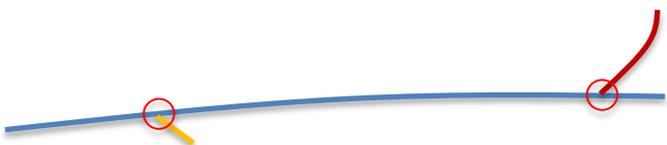
What kinds of interactions (or decay) finding should be the priority?

Vertex on the track reconstructed (or missed) by PMA

Vertex reco in PMA is based on individual 3D tracks. Vertices and tracks are associated (and all together is associated to hierarchy of PFParticles). Result depends (to some extent) on clustering and hits. Direct use of ADC may help to recover a lot...



- particle interacts, daughters reconstructed, interaction vertex found: all OK
- was there any kink missed along the primary?



- particle interacts, one of daughters co-linear with primary, clustered and reconstructed as single track
- can CNN be trained to identify such vertex?
- can it distinguish interaction from a delta ray vertex?
- if just a single track made: run through the trajectory and find any missed vertex? what efficiency is possible?



- $\mu \rightarrow e$ and stopping $\pi \rightarrow \mu \rightarrow e$ nearly identical
- decay vertex often not precisely located (muon includes electron or *vice versa*)

CNN machinery inside & outside LArSoft (1)

Use [Keras](#) as a primary toolkit for CNN training

- training data out of LArSoft: part of preparatory work in LArSoft and part in Python scripts
- CNN model prepared in Python, model & weights dumped to plain text („small” model = ~100MB)

Models applied in LArSoft

- simple C++ code to load and run Keras models from LArSoft modules
- *Tensorflow* to be added to LArSoft ups → then a good way to calculate CNN output, → this is rather long term plan: need to work out a good, generic interface (expect progress after Sept.)

→ Have a look at *larreco/RecoAlg/ImagePatternAlgs/Keras*:

- simple code to run Keras models
- we are using it with our ideas for CNN in LArTPC, but you can experiment by yourself
- [if some architecture configuration missing – we can add it, such changes are not breaking any higher-level code already using keras2cpp](#)
- basic code wrapped in an algorithm class and applied in a couple of modules → you may use it at any low/high level

→ *EmTrackClusterId* module in *larreco/RecoAlg/ImagePatternAlgs*:

- **input:** clusters and single hits; **output:** EM-like clusters (incl. 1-hit clusters)

CNN machinery inside & outside LArSoft (2)

Base algorithms for data preparation

- *larreco/RecoAlg/ImagePatternAlgs/PointIdAlg* (will add other algorithms as needed)
 - *DataProviderAlg*: caches **downsampled matrix of ADC**, functionality for making **2D patches** or flat vectors around wire/drift point
 - *TrainingDataAlg*: prepares **map of PDG codes** and interaction **vertex flags** corresponding to ADC matrix
 - *PointIdAlg*: reads-in network model, calculate network **output** for any **wire/drift coordinates**, or accumulated output for a **vector of hits (cluster)**
 - if more functionality is needed at this level (e.g. different patch size in wire and drift directions): should not break modules

Small, dedicated modules for each application (*larreco/RecoAlg/ImagePatternAlgs*)

- *PointIdTrainingData* & *PointIdTrainingNuevent* modules: **dump training data** (ADC / PDG / vertex maps), can select view and TPC, can look for neutrino interaction in fiducial volume (so the interaction vertex and needed part of the event is well seen)
- *PointIdEffTest* module: this one is testing efficiency and shows **how to apply network** to check if it is EM activity or track-like cluster
- Network model is the exchangeable part at the level of modules: processing scheme remains, just a better model can be inserted.
 - final CNN models for various tasks and detector configurations should go to *experimentXY_pardata*

```
#include "services_dune.fcl"
#include "caldata_dune.fcl"
#include "imagepatternalgs.fcl"
```

```
process_name: PointId
```

```
services:
{
  TFileService: { fileName: "reco_hist.root" }
  MemoryTracker: {}
  TimeTracker: {}
  RandomNumberGenerator: {}
  message: @local::dune_message_services_prod_debug
  FileCatalogMetadata: @local::art_file_catalog_mc
                    @table::protodune_services
                    @table::protodune_simulation_services
}
```

```
source:
{
  module_type: RootInput
  maxEvents: -1
}
```

```
physics:
{
  analyzers:
  {
    pointid: @local::standard_pointidtrainingdata
    testeff: @local::standard_pointidefftest
  }
}
```

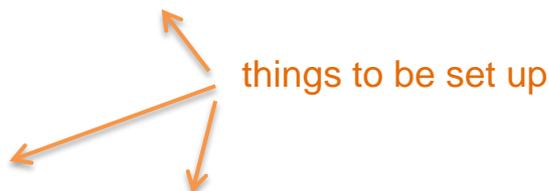
```
reco: []
anadata: [ pointid ]
anatest: [ testeff ]

stream1: [ out1 ]
trigger_paths: [ reco ]
end_paths: [ anatest ]
}
```

```
outputs:
{
  out1:
  {
    module_type: RootOutput
    fileName: "%ifb_%tc_reco.root"
    dataTier: "full-reconstructed"
    compressionLevel: 1
  }
}
```

The job configuration for modules: training data, testing models

- *pointid* here is making the training data files (that are further processed in python scripts)
- *testeff* applies CNN to clusters



```
physics.analyzers.testeff.PointIdAlg.NNetModelFile: "/home/robert/fnal/v5/cnn/small1_sgd_lorate_8k_coll.nnet"
physics.analyzers.testeff.PointIdAlg.PatchSize: 32 # keep it corresponding to what model is expecting
physics.analyzers.testeff.PointIdAlg.DriftWindow: 10 # same note as above
physics.analyzers.testeff.HitsModuleLabel: "linecluster"
physics.analyzers.testeff.ClusterModuleLabel: "linecluster"
physics.analyzers.testeff.View: 2 # select which view is tested
physics.analyzers.testeff.Threshold: 0.4 # threshold for EM / track discrimination (0:EM, 1:track)
physics.analyzers.testeff.SaveHitsFile: false # text file with more detailed output from classification
```

```
physics.analyzers.pointid.TrainingDataAlg.SimulationLabel: "largeant"
physics.analyzers.pointid.TrainingDataAlg.WireLabel: "caldata"
physics.analyzers.pointid.TrainingDataAlg.SaveVtxFlags: true # pdg code is 2 lower bytes, vtx flags are 2 higher
physics.analyzers.pointid.TrainingDataAlg.PatchSize: 32
physics.analyzers.pointid.TrainingDataAlg.DriftWindow: 10
physics.analyzers.pointid.SelectedTPC: [2] # selected TPC and views can be dumped
physics.analyzers.pointid.SelectedView: [0]
physics.analyzers.pointid.OutTextFilePath: "/home/robert/fnal/v5/cnn/raw_data"
```

```
#include "services_dune.fcl"
#include "caldata_dune.fcl"
#include "imagepatternalgs.fcl"
```

```
process_name: PointId
```

```
services:
{
  TFileService: { fileName: "reco_hist.root" }
  MemoryTracker: {}
  TimeTracker: {}
  RandomNumberGenerator: {}
  message: @local::dune_message_services_prod_debug
  FileCatalogMetadata: @local::art_file_catalog_mc
                    @table::protodune_services
                    @table::protodune_simulation_services
}
```

```
source:
{
  module_type: RootInput
  maxEvents: -1
}
```

```
physics:
{
  producers:
{
  emtrackid: @local::standard_emtrackclusterid
}
```

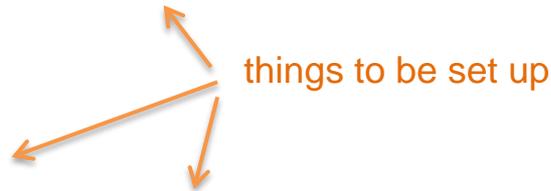
```
reco: [ emtrackid ]
```

```
stream1: [ out1 ]
trigger_paths: [ reco ]
end_paths: [ stream1 ]
}
```

```
outputs:
{
  out1:
{
  module_type: RootOutput
  fileName: "%ifb_%tc_reco.root"
  dataTier: "full-reconstructed"
  compressionLevel: 1
}
```

The job configuration for modules: **production:** tagging clusters and hits

- *emtrackid* applies CNN to clusters and optionally unclustered hits
- the output: new collection of clusters, tagged as EM
- today: only EM clusters are outputted, single EM-like hits are added to the cluster collection as 1-hit clusters, → everything produced by the module was recognized as EM



```
physics.producers.emtrackid.PointIdAlg.NNetModelFile: "/home/robert/fnal/v6/cnn/small1_sgd_lorate_8k_coll_ind.nnet"
physics.producers.emtrackid.PointIdAlg.PatchSize: 32 # keep it corresponding to what model is expecting
physics.producers.emtrackid.PointIdAlg.DriftWindow: 10 # same note as above
physics.producers.emtrackid.HitModuleLabel: "linecluster" # hits used to create clusters (use „," if single hits should not be tagged)
physics.producers.emtrackid.ClusterModuleLabel: "linecluster" # clusters to be tagged
physics.producers.emtrackid.Threshold: 0.3 # threshold for EM / track discrimination (0:EM, 1:track)
physics.producers.emtrackid.Views: [2] # selected views can be processed (or all if the list is empty)
```

Please, contact us for support with applying / training models.

Summary

- All components needed to apply EM/track tagging in place
- CNN model prepared on 2GeV/c π^+ , 5mm wire pitch, should work a broad class of events. Likely need another model for 4mm pitch: one of easy things to start with LArIAT simulations.
- Noise should be studied: how much is acceptable, what „patterns” are in data, is this well modeled with MC?
- Application on data and feedback is needed – this is the real test of the tool.
- Vertex identification / classification is the next thing to run. The same idea of patches. Timescale: Sept. DUNE Collaboration meeting.
- Option in PMA module to match PFParticle hierarchy with test-beam particle to be added: how to consume reco info from upstream detectors?
- **All of these are interesting tools and fresh approach: but for physics results a lot needs to be understood, what are the systematics, how does it work on real data...!**