Pattern recognition on 2D ADC image level

P. Płoński, D. Stefan, R. Sulej

Other slides for the reference

DUNE Collaboration meeting:
https://indico.fnal.gov/getFile.py/access?contribId=54&sessionId=19&resId=0&materialId=slides&confId=10612
LArSoft Coordination meeting:
https://indico.fnal.gov/getFile.py/access?contribId=4&resId=0&materialId=slides&confId=12278
FD sim/reco meeting:
https://indico.fnal.gov/getFile.py/access?contribId=0&resId=0&materialId=slides&confId=12330

Here:

- Short intro on the approach
- EM vs. track-like cluster ID
- next steps: vertex identification, higher level DNN structures

Why: compare the following list with yesterday's talks.

- **Track-like vs. EM shower distinction – a task for itself**
- track vs EM shower: affecting *everything* in topology reconstruction
- **detection of decay points**
- **vertex & kink finding/classification** in test-beam, primary vertex finding in ν events
- electron candidate selection (and other high-level reco tasks)
- ROI selection in noisy environment (where hit finding becomes difficult)
- enable dedicated hit fitters for various classes of ADC regions
- …

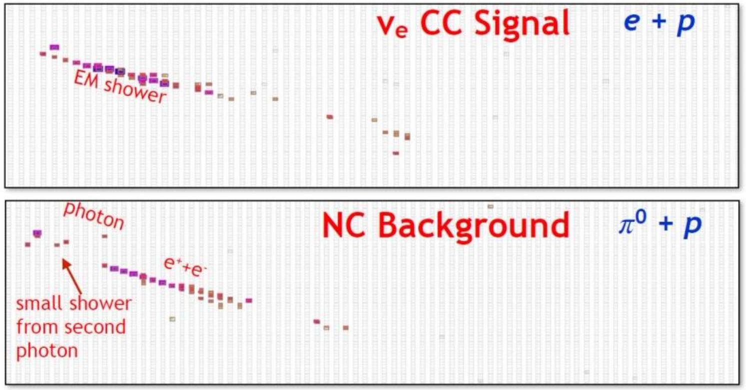
- hit-based pattern recognition and downstream reconstruction up to the neutrino ineraction classification struggle with these tasks.
- hits have advantages, but also information is reduced, hit finders / fitters efficiency is limited, can be confused by the noises

Deep Neural Nets options for ν detectors:

1. Use *full event „images"*: make the classification of the event / regression of the energy, etc.
2. Go *step by step*, define simple tasks that can advise „standard" reconstruction: can understand event parts, can be useful quickly (even if we target more overall reconstruction with DNN) ← **we go for this path**

DNN Option 1. Example from the NOvA slides at Art Users Meeting, FNAL, June 17

https://indico.fnal.gov/conferenceOtherViews.py?view=standard&confId=12068, talk by A. Radovic



- „imaging” detector, 2D projections, just like LArTPC
- similat to our „90 deg rotated” design option: beam perpendicular to slices (planes of scint. bars)
- resolution: **6 cm x 3.9 cm**
- ν energy peak: **2 GeV**
- event classification from 2x 2D overal images
- my note: DNN architecture with a late connection between 2D views (I know this is not esy)

ν_e selection efficiency: **35%** → **49%** (at the same purity: ~87%, if I understood)

„previous” (= LEM?)

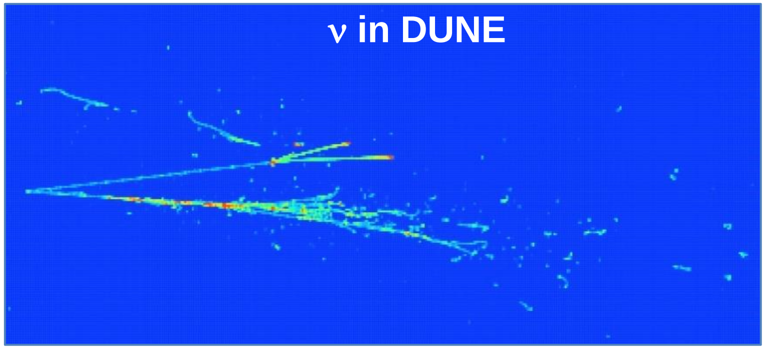
CNN

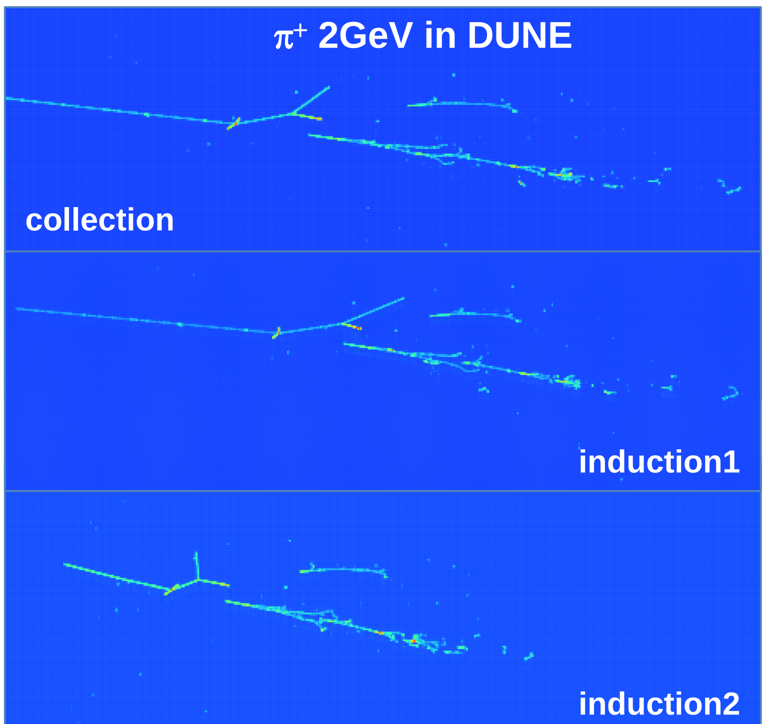
From the next slides in NOvA talk:

„Why not try to identify small reconstructed objects like prongs?
Or take advantage of semantic segmentation to classify *every cell* in a NOvA event.”

→ this is going to be ~ DNN Option 2, which also we are exploring

…and with LArTPC imaging:





- resolution: 5mm x 0.4 mm
- 2x or 3x 2D projections, but slightly different config. from NOvA: beam parallel to time-slices (at least „baseline" opt.)
- each time slice seen by every 2D projection
  → exploited e.g. by 3D imaging
- ν energy peak: 4 GeV

Requirement:

- ν_e selection efficiency: **~90%**
  (we want also purity much higher… 9x% !!!)

In LArTPC data we have:

- much more information to handle, larger variety of event topologies
- detiled vertex features (cascade displacement): powerfull discriminant, if large-scale event compounds correctly recognized

Example of present (DUNE MCC6) result of 2D pattern recognition:

high energy ν CC | **low energy ν CC**

ADC

pattern reco #1

pattern reco #2

- Large and small scale shower features important:
  → energy reconstruction and interaction classification

- EM tends to include hadronic parts

…or

- EM blindly treated as tracks

➔ both result with poor 3D

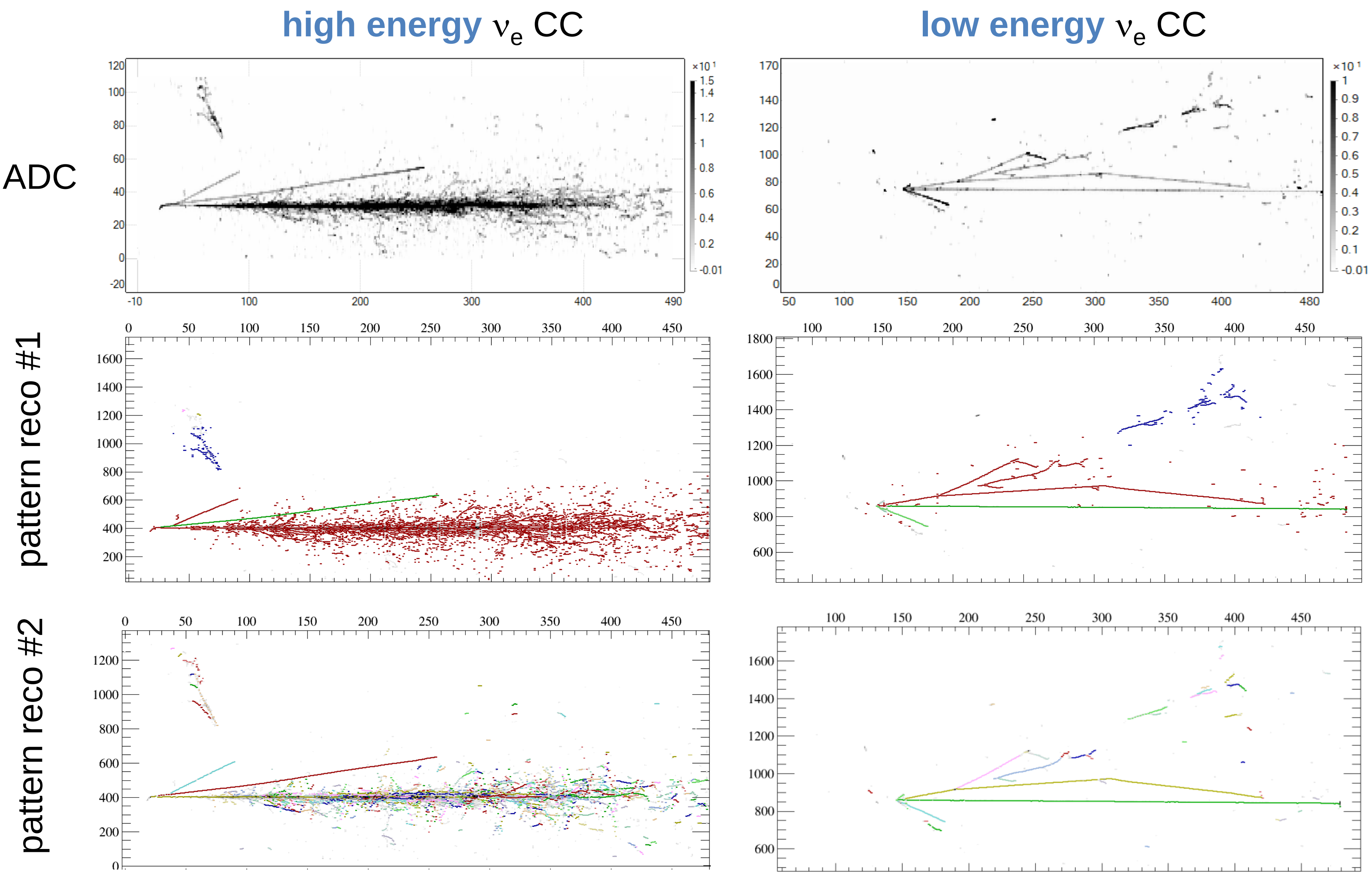
- Same problems in small scales around the primary vertex

- Huge effort in hit-based algorithms… try to use it, not waist it

Our personal opinions:

- …this can be improved with a more complete information from 2D ADC
- …more fun in designing the network architecture than if-else-then-repeat on hit configurations

Example of present (DUNE MCC6) result of 2D pattern recognition with hits:



First block: Track-like or EM activity?

➔ A *„basic task”*…

➔ At the same time: one of the **highest priority needs** for the standard reconstruction.

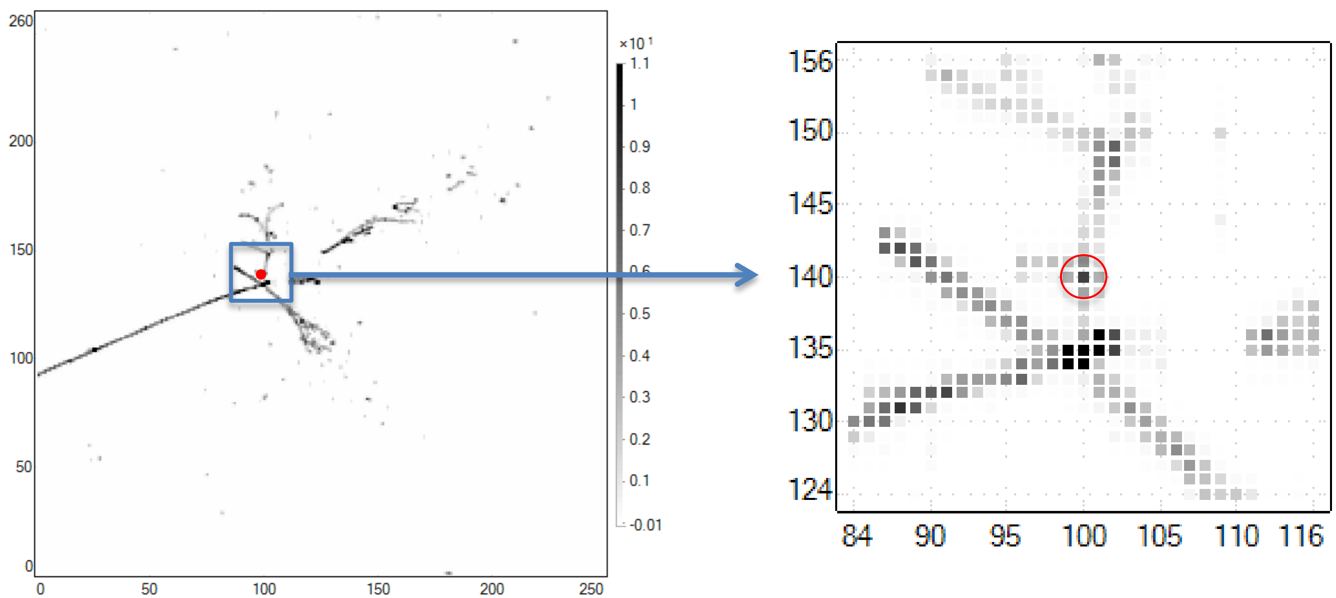
➔ Important for physics: ν_e selectors for FD, EM vs hadronic part in protoDUNE, FD, …

Very high resolution of LArTPC → start with downsampled images due to data volumes:

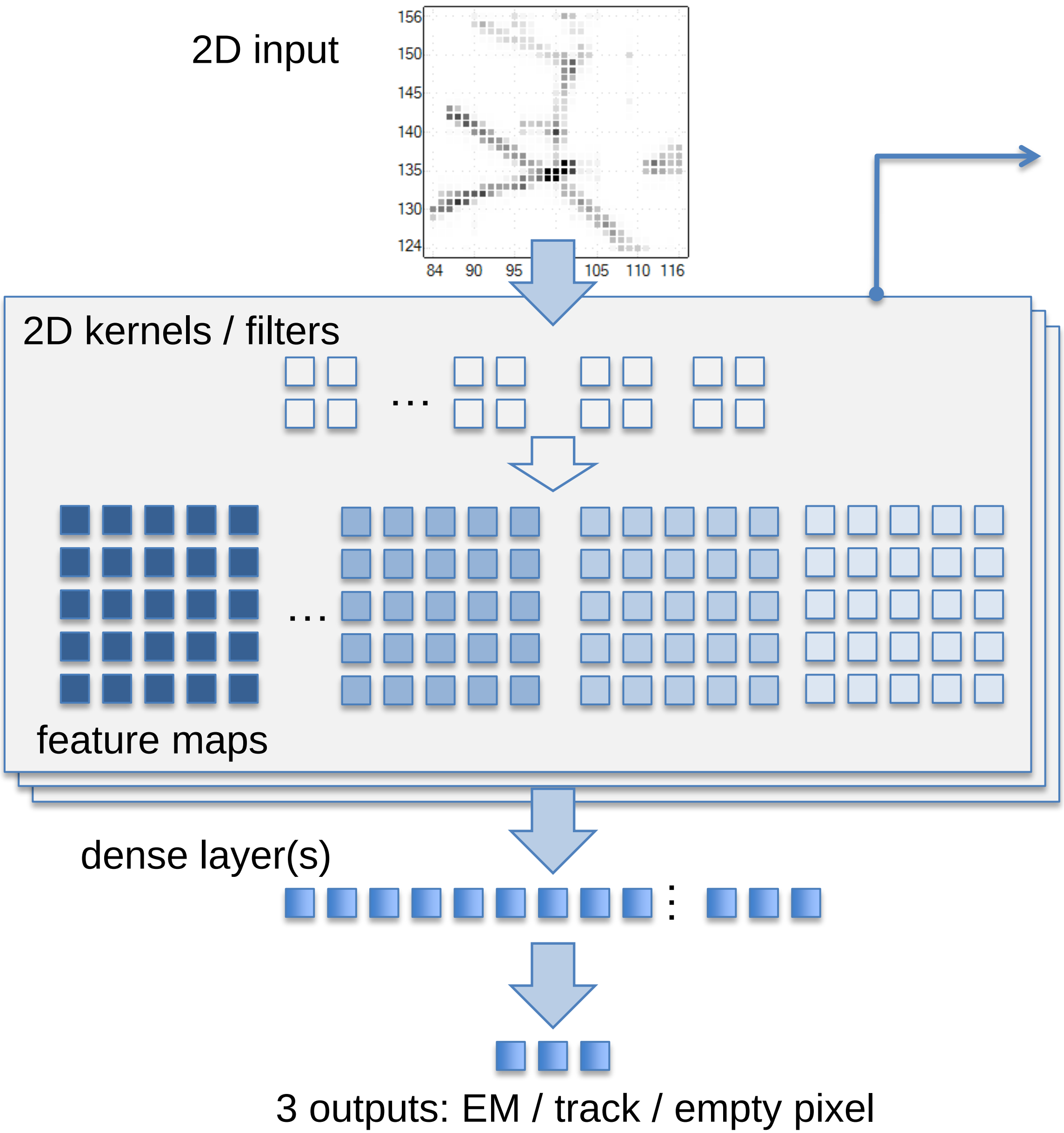
- downsample in drift direction (x10) down to the resolution ~wire pitch …and we know x5 will do better
- use max ADC, not average → avoid fading small signals (max-pooling, rethought and survives)

Classify interesting point using surrounding patch (today still 32x32):

- large enough to capture the context
- small enough to handle training data …and we already know it should be bigger



Convolutional Neural Network (CNN)



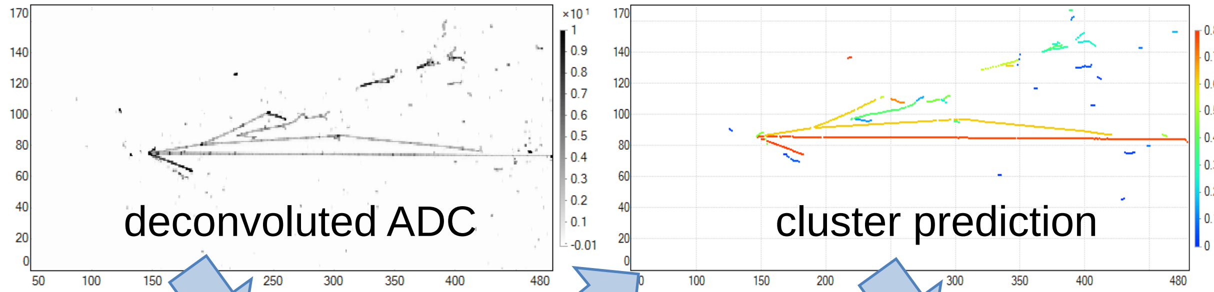
- makes use of 2D meaning at input
- convolutional layers (many):
  - ***very flexible configuration of architectures*** replaces blind full interconnection of MLP nets;
  - hand-adopted to particular task
- fully interconnected (*dense*) layers just before the output
  - huge number of connections here..
- not less computational complexity…
- but proven to learn difficult tasks, not solved otherwise
- one can look at 2D kernels and feature maps to understand what features are being used

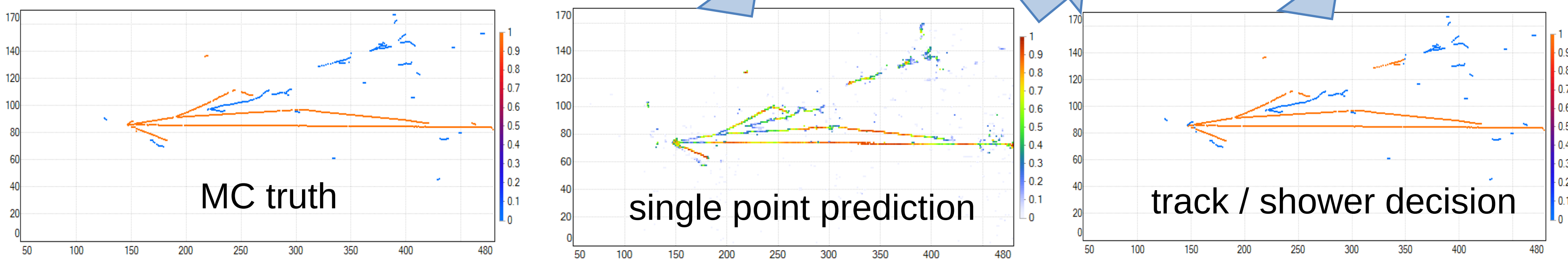
- MLP used as well to have „baseline” results

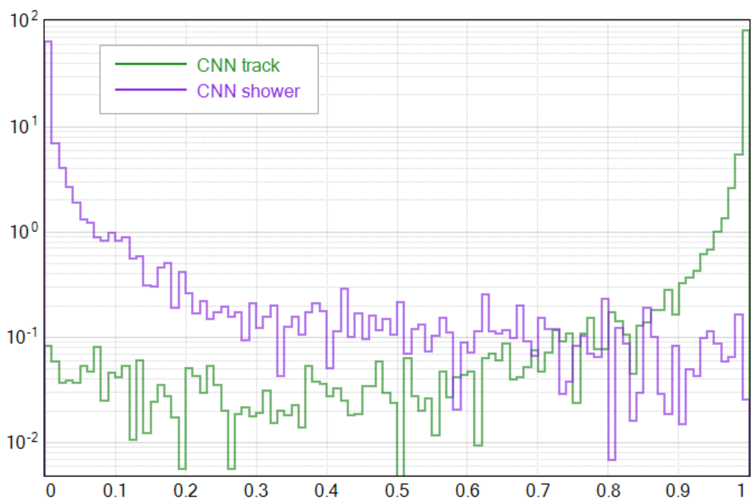
MLP/CNN trained on π^+ in protoDUNE, applied also to ν_e in DUNE FD

EM-like / track-like cluster identification flow:

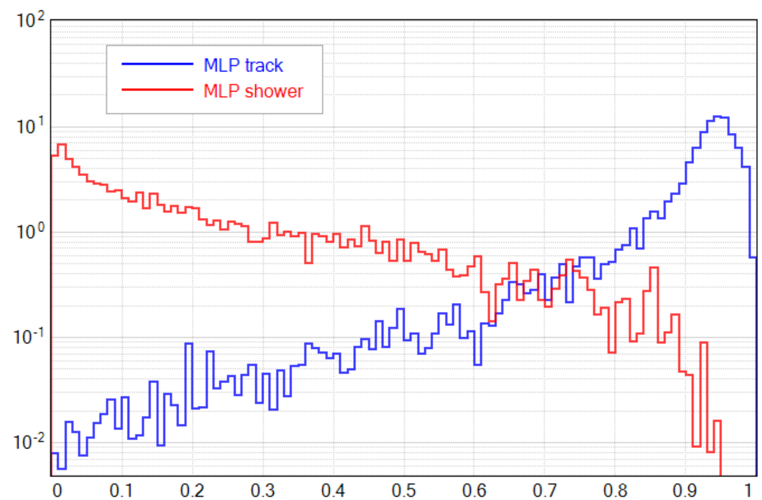
(not the recent CNN on these pictures!)







CNN cluster prediction values

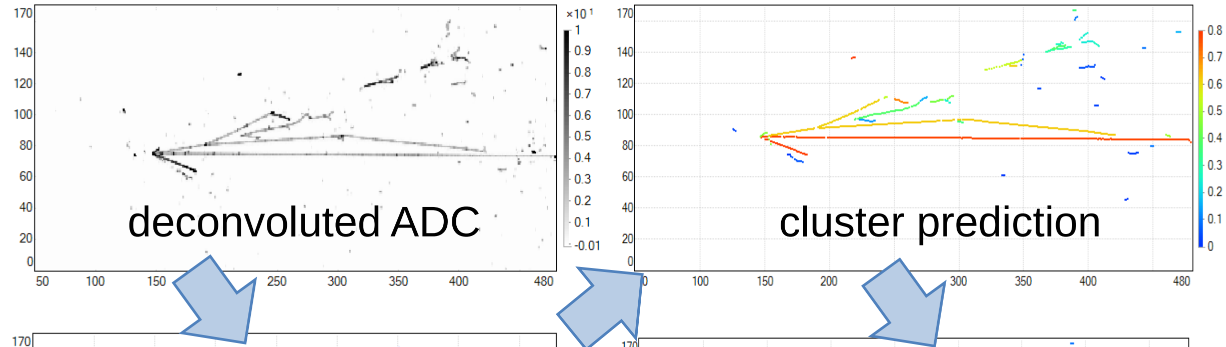


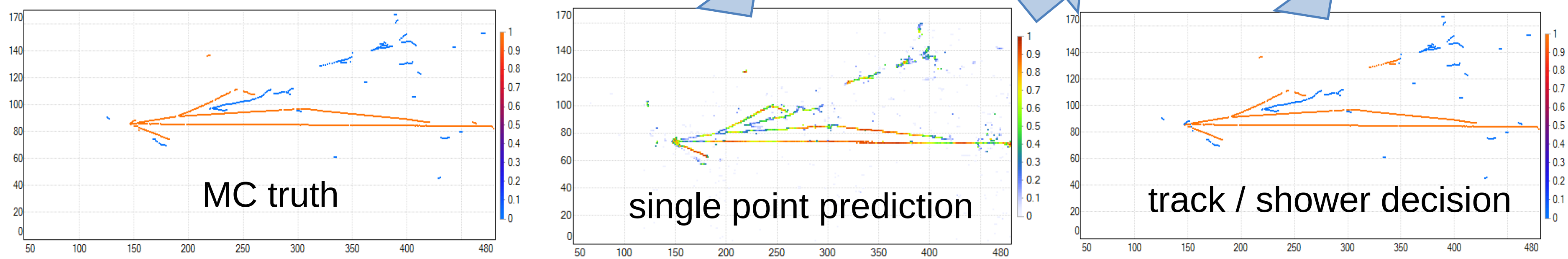
MLP cluster prediction values

MLP/CNN trained on π^+ in protoDUNE, applied also to ν_e in DUNE FD

EM-like / track-like cluster identification flow:

(not the recent CNN on these pictures!)





- patch 32x32 (~15cm^2) can see local context of a tested point
- information on a view from a „larger distance” is provided now by the „standard” clustering algorithm, Cluster Crawler:

  → it is very efficient in selecting parts of objects
  → but does not give EM / track ID
  → in a future work may be replaced with a higher-level DNN structure

MLP / CNN results as of today

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

MLP: **92.4%** track / **91.7%** EM correct cluster ID rate – kept for reference

CNN: **96.2%** track / **96.6%** EM correct cluster ID rate, now works for all views

(at the collab. meeting result was ~ 90% / 90%)

usual mistake sources:

- most cases: complicated configurations, especially if on the image boundaries
- there is some orientation dependence: more difficult recognition for particles if direction strictly row or column of pixels
- long track-like electron
- 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

→ more topologies at input: helped

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

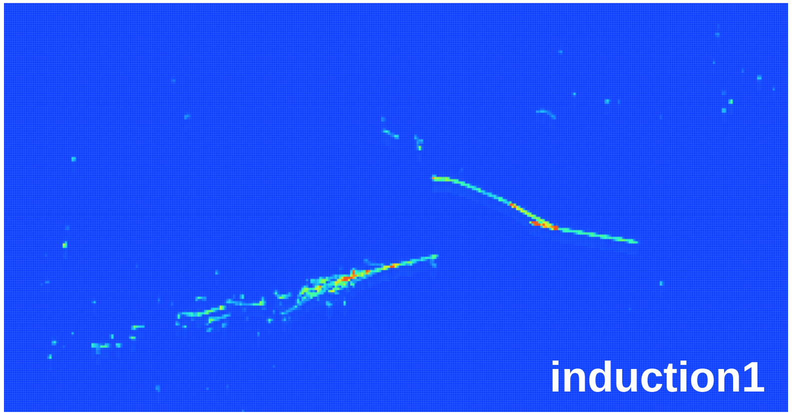
→ next: prepare model with neutrino events (data dumped, ready for the „python" work)

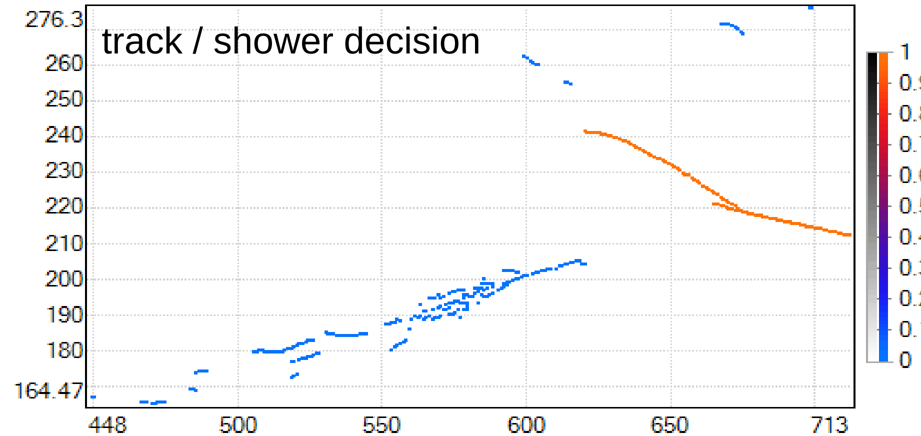
→ MLP: results now kept for the reference only

→ CNN: ***goes to LArSoft develop***, ready to: EM/track ID, combine with 3D tracking, …

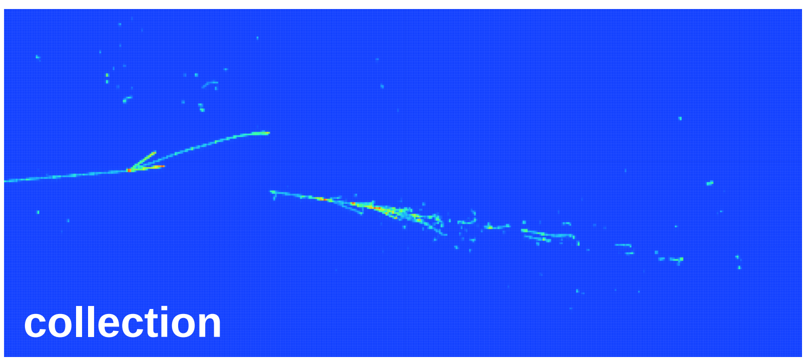
→ larger patch and/or higher reslution in drift, automated search for the best model by Piotr

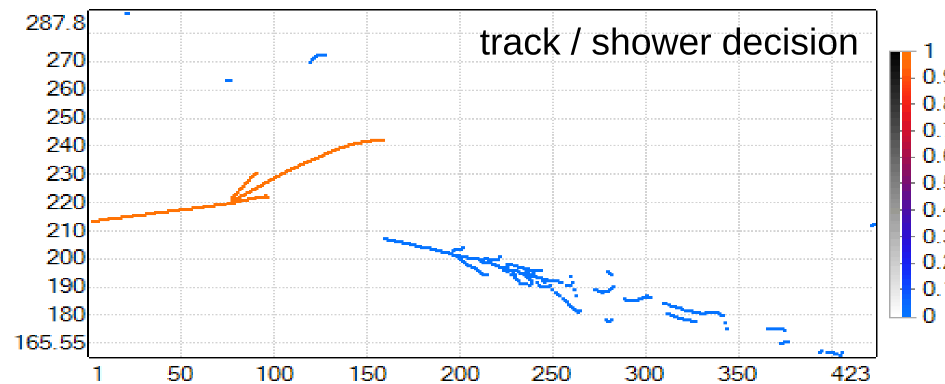
CNN results as of today: π^+ 2 GeV/c in protoDUNE SP



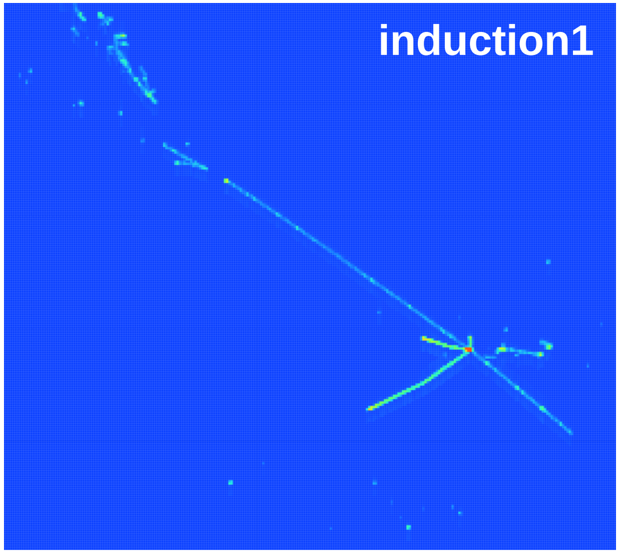


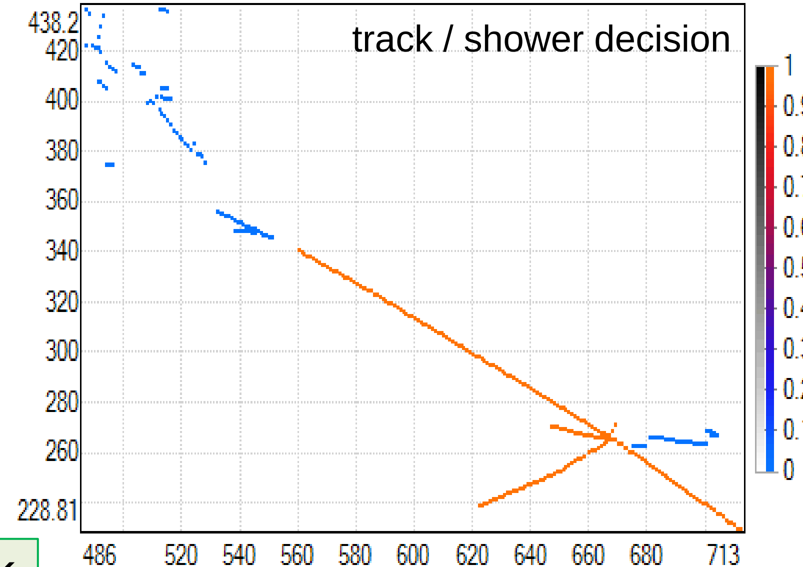
All OK



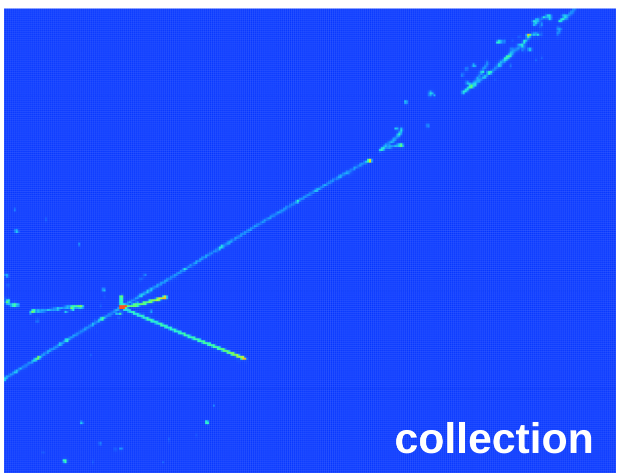


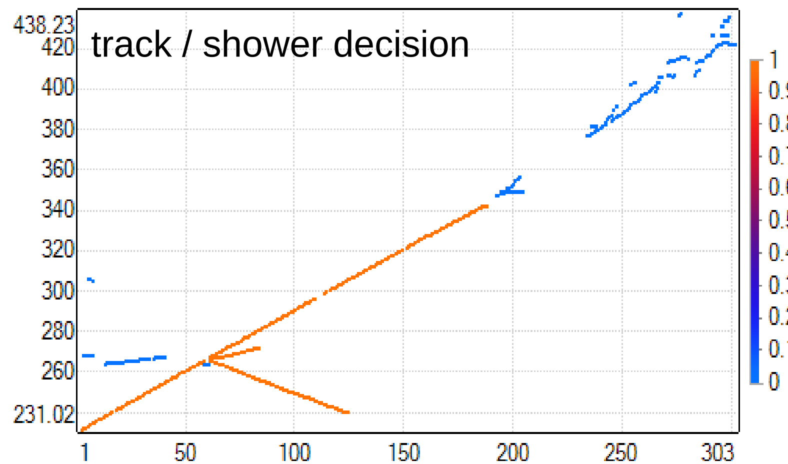
CNN results as of today: π^+ 2 GeV/c in protoDUNE SP



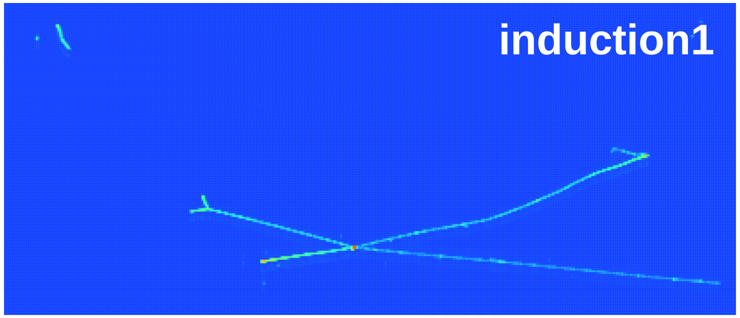


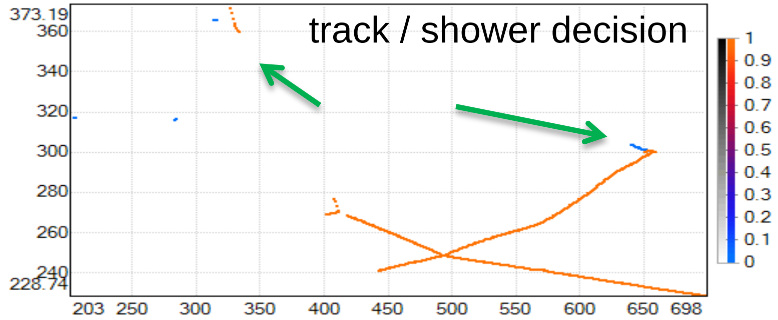
All OK



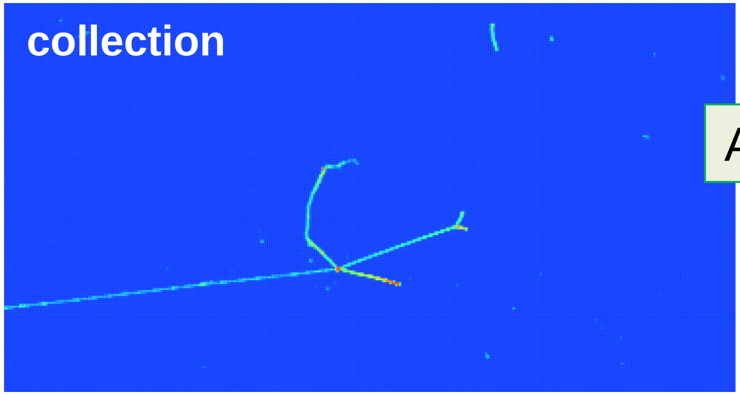


CNN results as of today: π^+ 2 GeV/c in protoDUNE SP

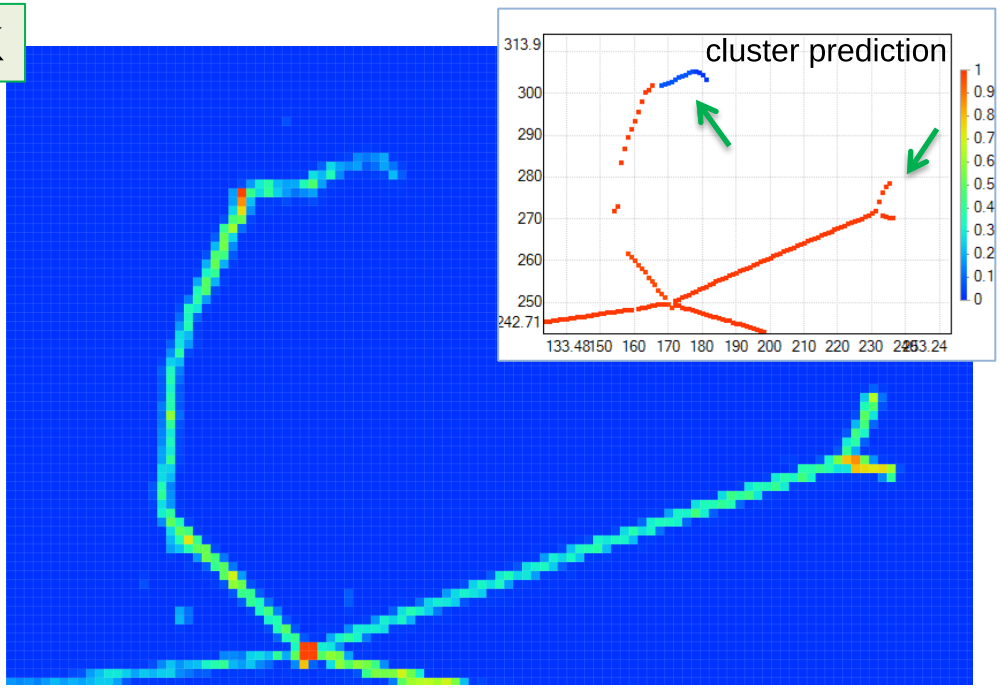


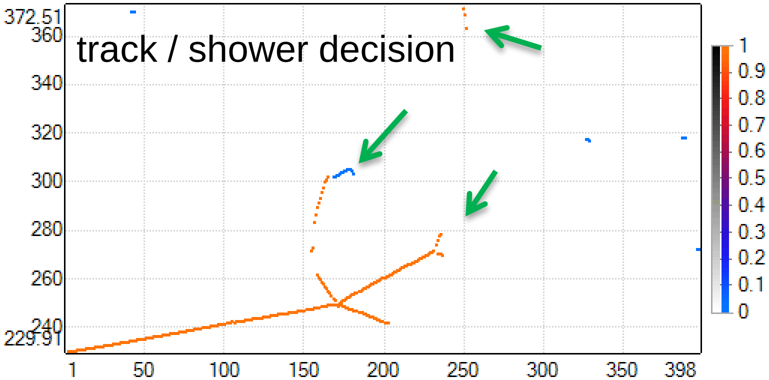


Even though CNN was not specially tuned for Michel's, the prediction values are pretty „decided"

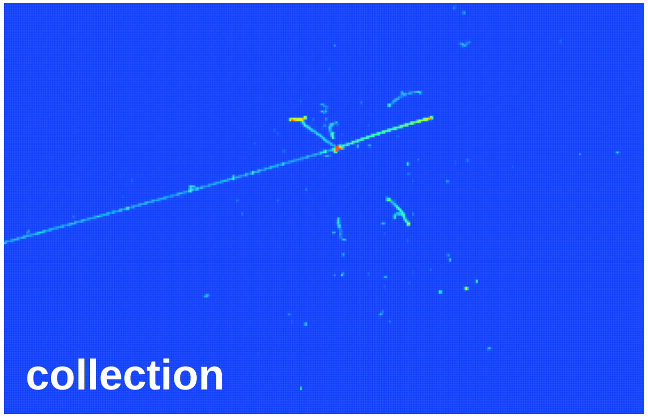


All OK

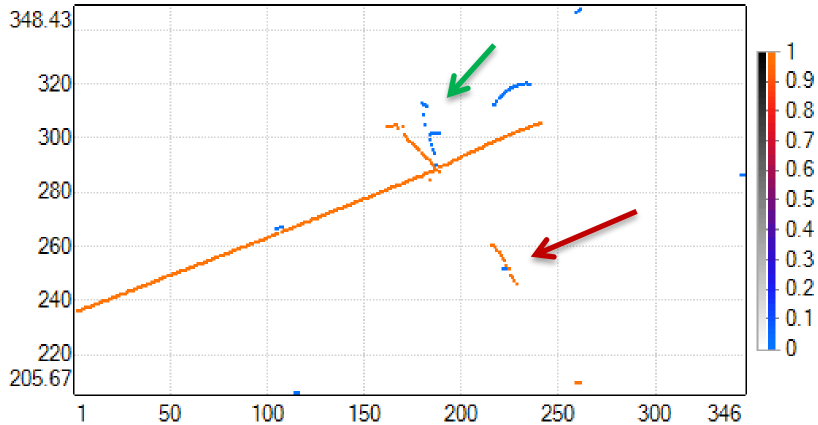




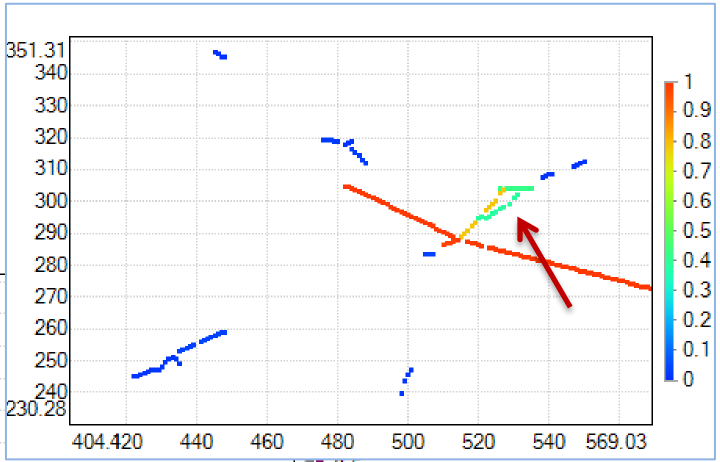
CNN results as of today: π^+ 2 GeV/c in protoDUNE SP



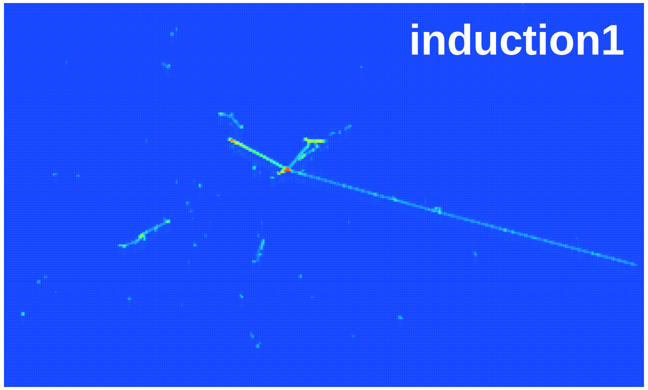
track / shower decision



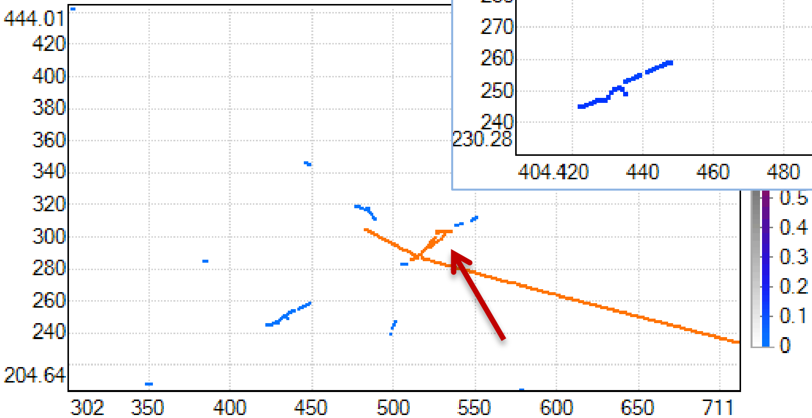
cluster prediction



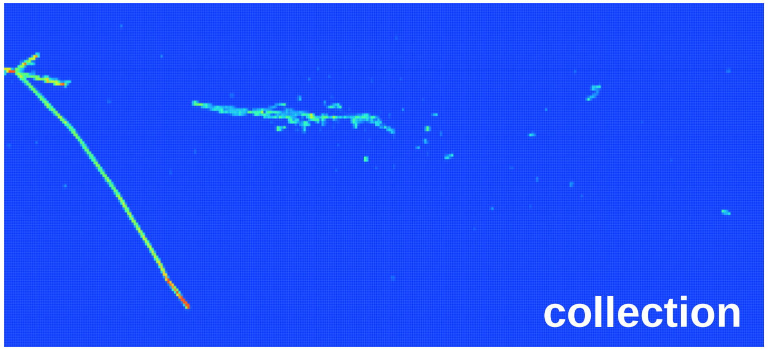
- decision threshold may be wrong

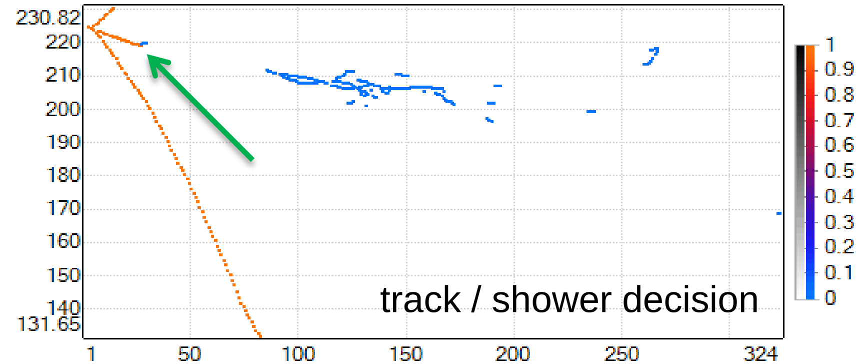


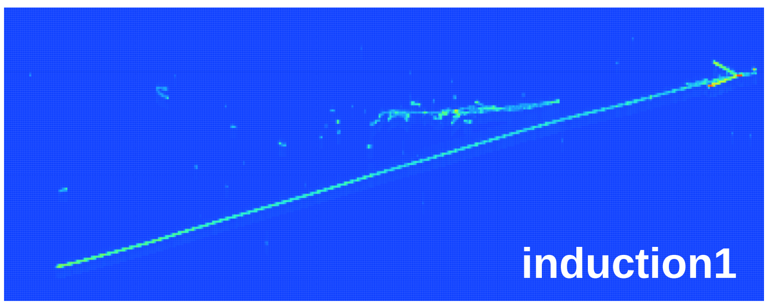
track / shower decision

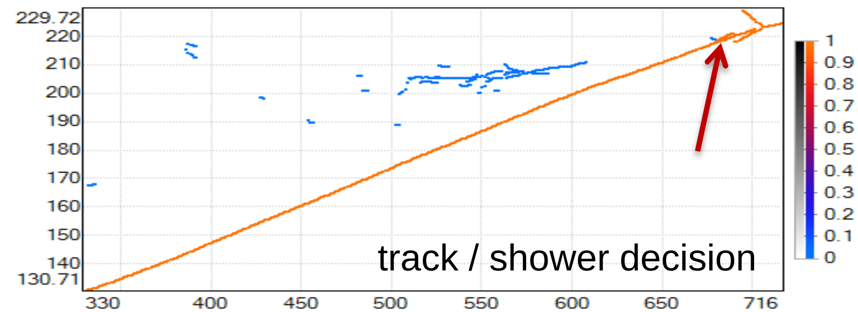


CNN results as of today: π^+ 2 GeV/c in protoDUNE SP

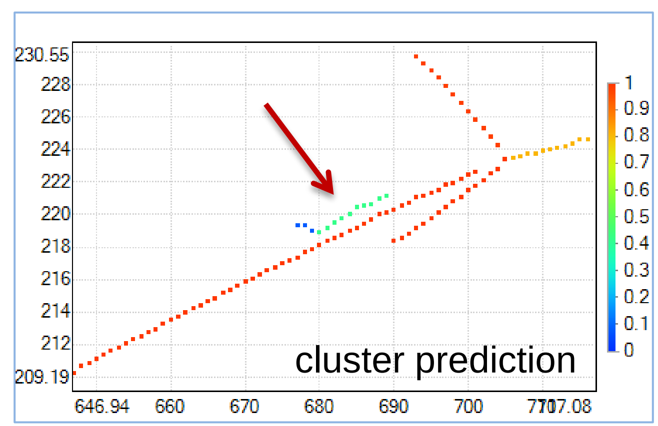




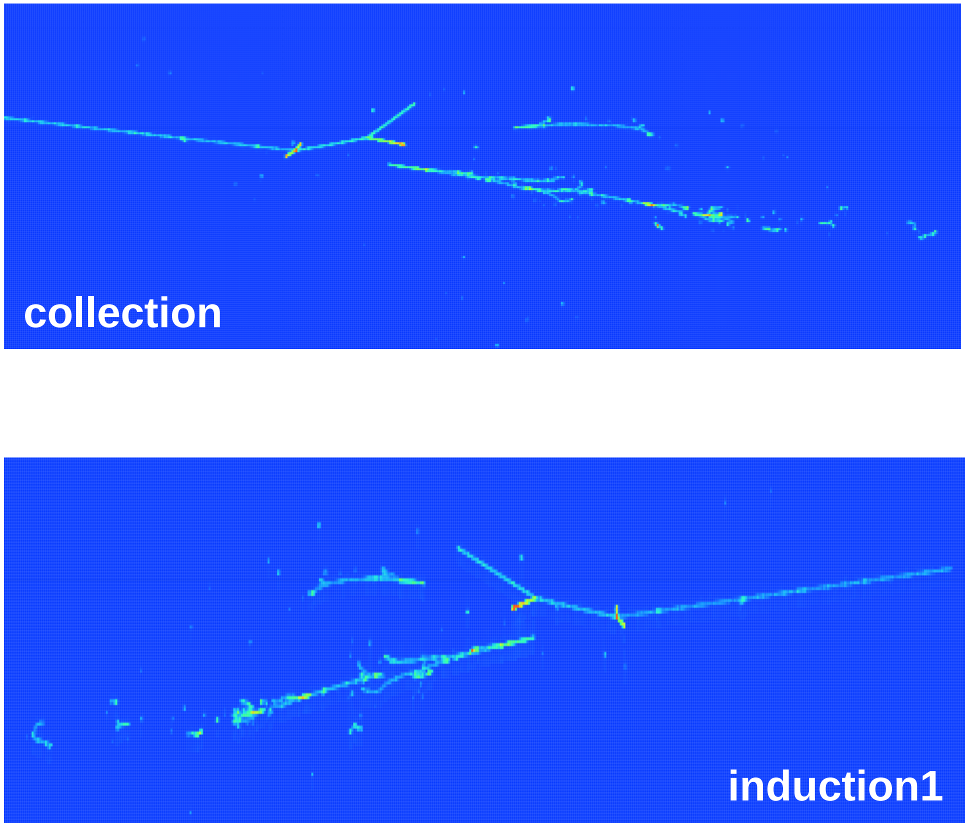


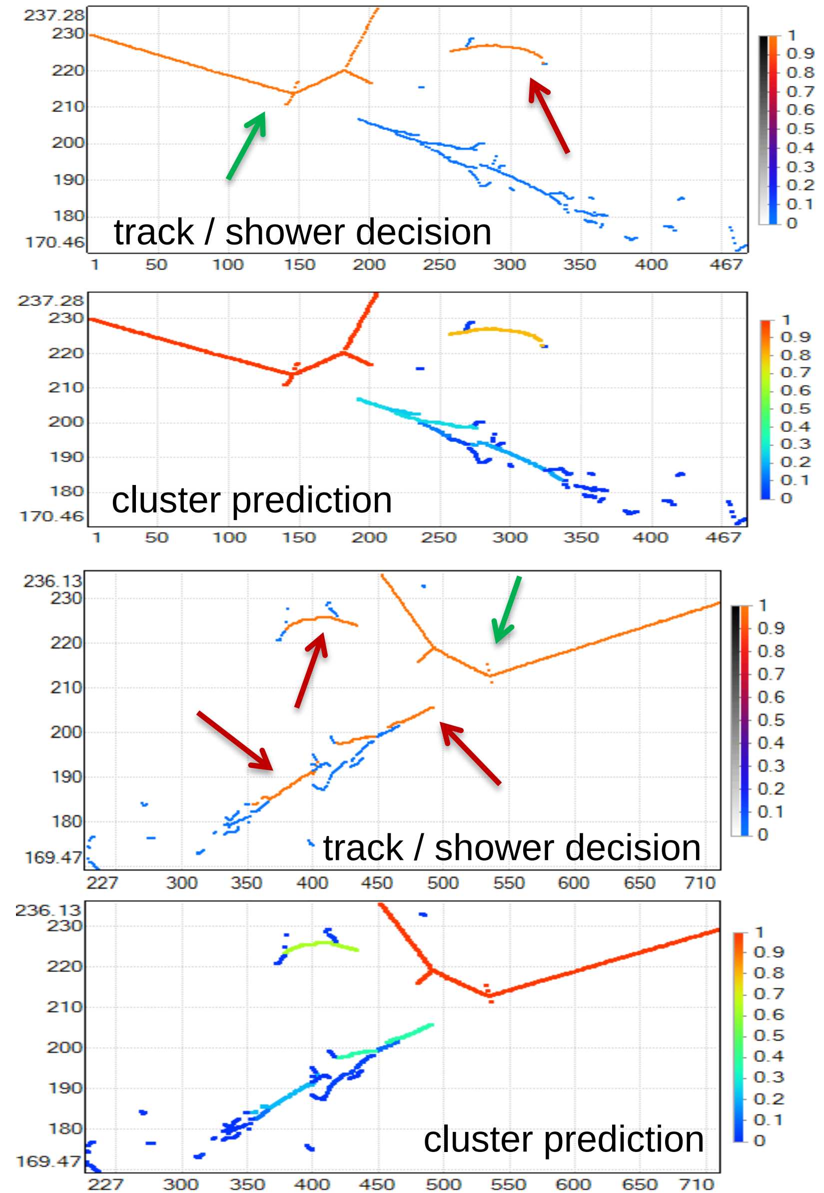


- again: decision threshold may be wrong
- tiny electron well found at the end of pi!



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



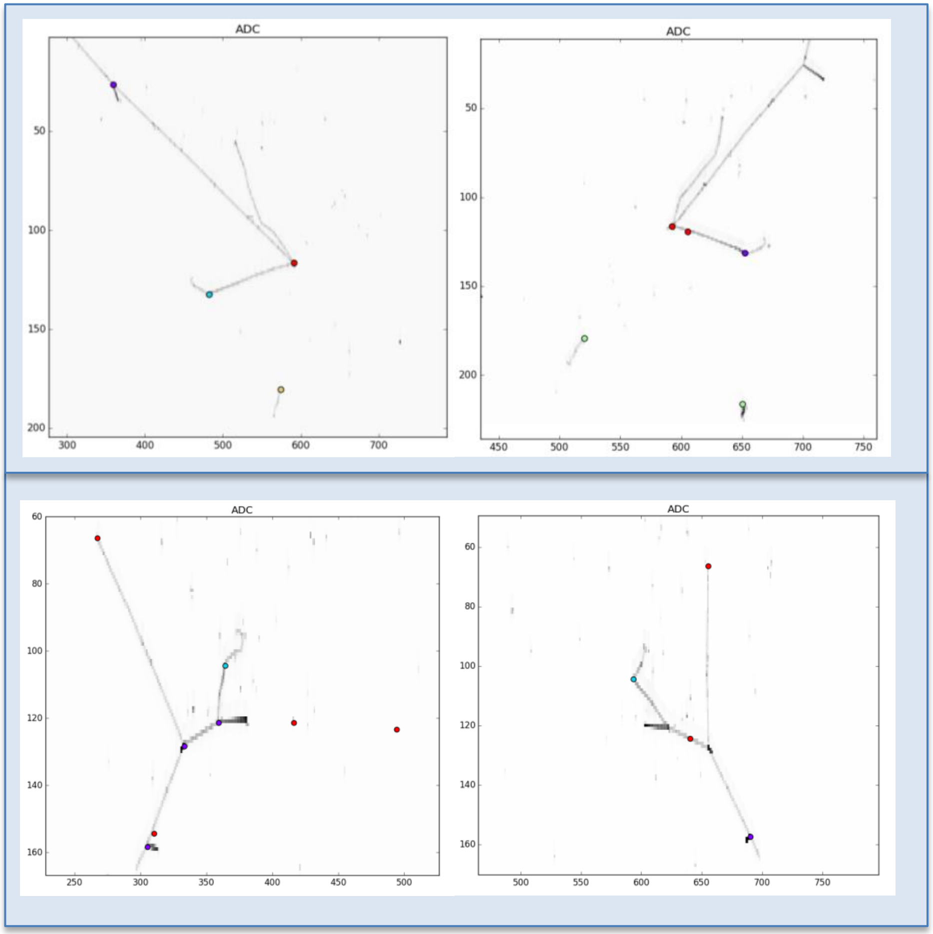


The most confused one…

- many on-the-border prediction values
- may be limited by size/resolution

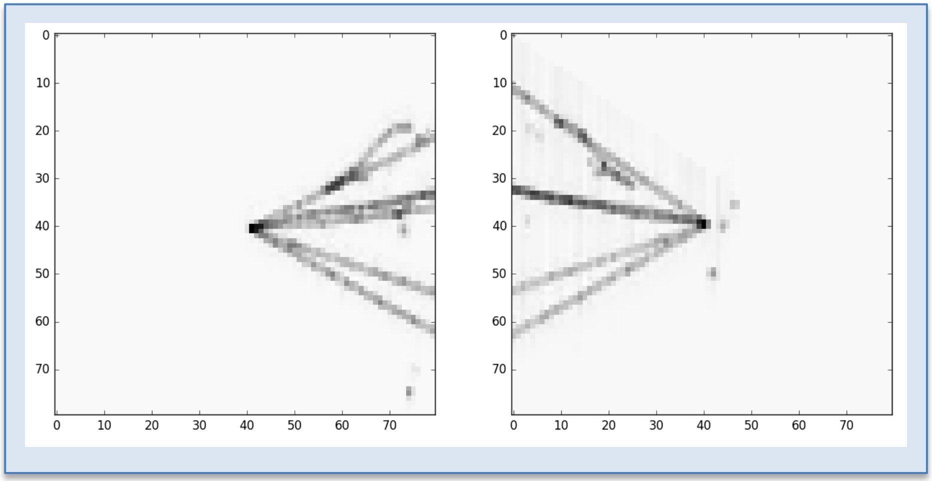
Next blocks

- <u>Vertex identification</u>
  - support tracking with interaction/decay finding
  - select EM shower starting points (not trivial in low energy)



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

- <u>Neutrino classification</u>
  - force classifier to be focused on the vertex features
  - **try to be sensitive to „gap” in full neutrino event**



- need more events to build training set (only 1 training image pair/triplet per 1 event)
- more complex (interesting) architectures
- uses larger patch arount the vertex and less downsampled drift
- need to adjust image building to better contain event
- more careful when producing data files to avoid really huge volumes

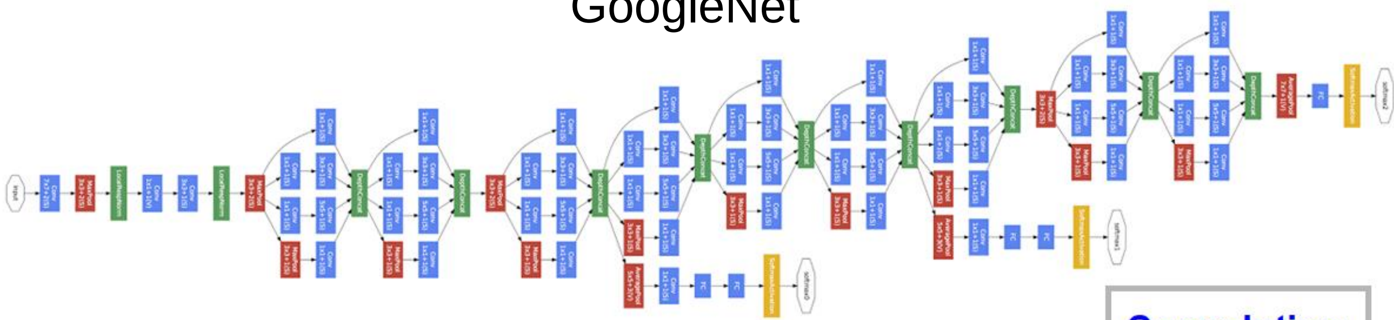
What else should be done at the „low level" of small patches?

- **Use 3D imaging to select corresponding points in 2D patches in all views**
  - → no need to go to high-level reconstruction to make use of full information
  - → easy to be conservative and use single views if 3D finds ambiguity
- **Use noise from real data empty events combined with MC particles**
  - → proove that noise patterns can be rejected

This are short tasks and expertise in experiment frameworks is minimal!

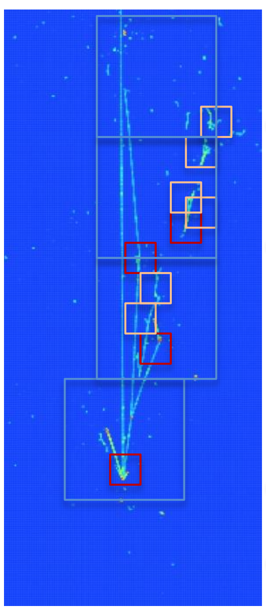
Goal: put blocks together

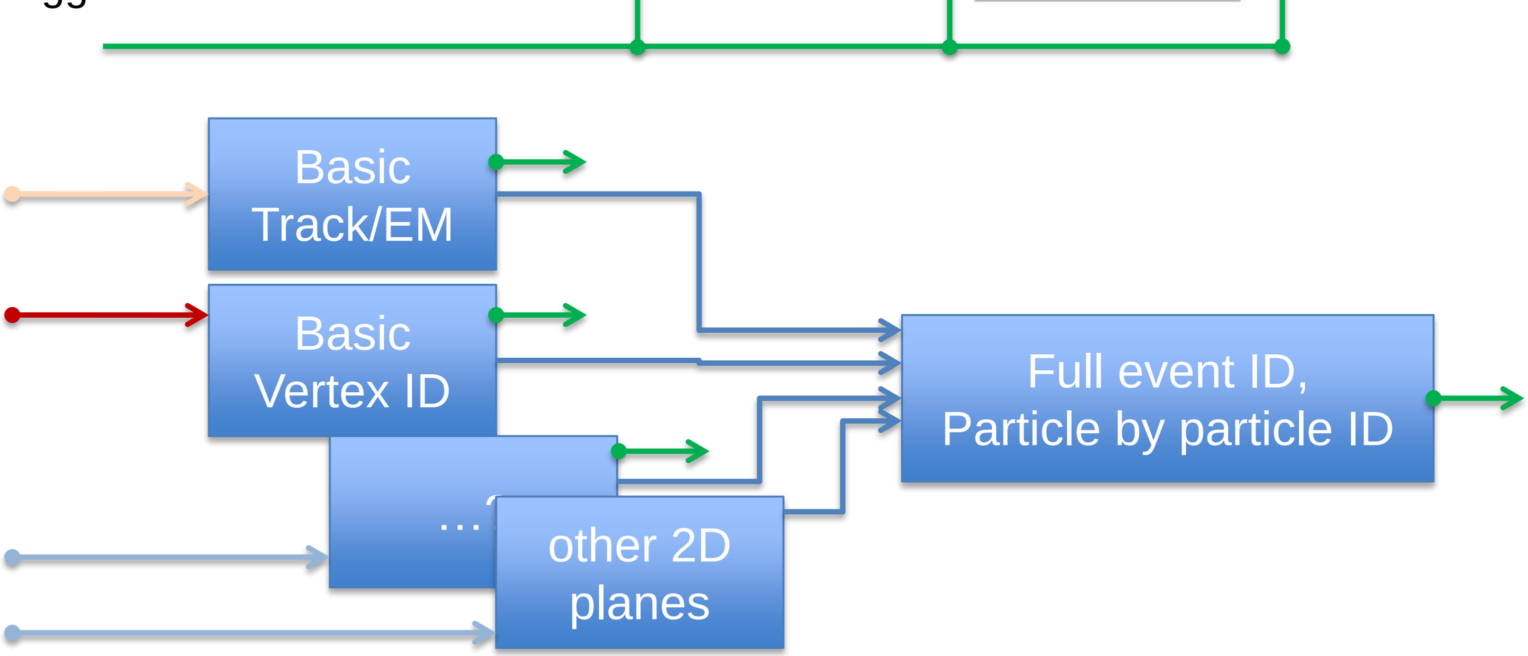
GoogleNet



Convolution
Pooling
Softmax
Other

- there are many tricks in DNN designing
- e.g. one can „apply training” in several points of a bigger structure





Summary

- Merging this first tool with LArSoft develop.

- Single model made for Collection / Induction planes
  - if separate models better, will provide such functionality
  - **optimal models/architectures will come from Piotr's scans**
  - **correct ID rates: 96.2%** track / **96.6%** EM

- Didn't manage for today with using EM/track ID in tracking codes, only efficiency testing modules show how to classify a cluster or single point…

- Work on vertex identification started, includes neutrino event classification
  - Would like to make it sensitive to the electron-vertex gap and test on 3/5mm pitch

- Deep learning is a serious chapter in LArTPC's. We're discussing a good base for development support in LArSoft.
  - Keras: actually no new dependencies needed in LArSoft – but will move to more efficient implementation of inference mode when Tensorflow included in UPS
    - we're targeting more deep manipulations in optimizers → Keras easier for that
  - NOvA uses Caffe: same possibilities, different box, more all-in-one-ready-to-use, this may work as well for LAr people and is not excluded in a future

Backup

CNN / MLP machinery inside & outside LArSoft (1)

Use ***Keras*** as a primary toolkit for CNN training, MLP's made with *NetMaker*

- need training data out of LArSoft: part of preparatory work in LArSoft and part in Python
- CNN model prepared in Python (Amir's GPUs used), model & weights dumped to plain text
- MLP model done on Dorota's super-laptop, model & weights dumped to xml file
- prototypes ready → massive search for optimal models on ***mljar*** by Piotr

Models applied in LArSoft

- simple C++ code to load and run Keras models, similarly for models form NetMaker
- interface classes to hide the model origin and run everything in the same fashion

- *Tensorflow* to be added to LArSoft ups → then a good way to calculate CNN output

→ have look at *larreco/RecoAlg/ImagePatternAlgs/Keras*:

- simple code to run Keras models
- we are using it with our ideas for CNN in LArTPC, but it enables running any model, so 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

CNN / MLP 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 sheme remains, just a better model can be inserted.
  - can provide small (5-6MB size) MLP model in code directory to be able to run code (or not if absolutely forbidden)
  - final CNN models for various tasks and detector configurations should go to dune_pardata

The job configuration for modules

```
#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
 }
}
```

- *pointid* here is making the training data files (that are further processed in python scripts)
- *testeff* applies MLP or CNN to clusters
- please, contact us if need help on running

things to be set up

```
physics.analyzers.testeff.PointIdAlg.NNetModelFile:      "/home/robert/fnal/v5/mlp/mlp_3class_4k_9.xml"
#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]  # multiple TPC an views can be dumped
physics.analyzers.pointid.SelectedView:   [0]
physics.analyzers.pointid.OutTextFilePath: "/home/robert/fnal/v5/cnn/raw_data"
```