CTD and HEP.TrkX report

Steve Farrell

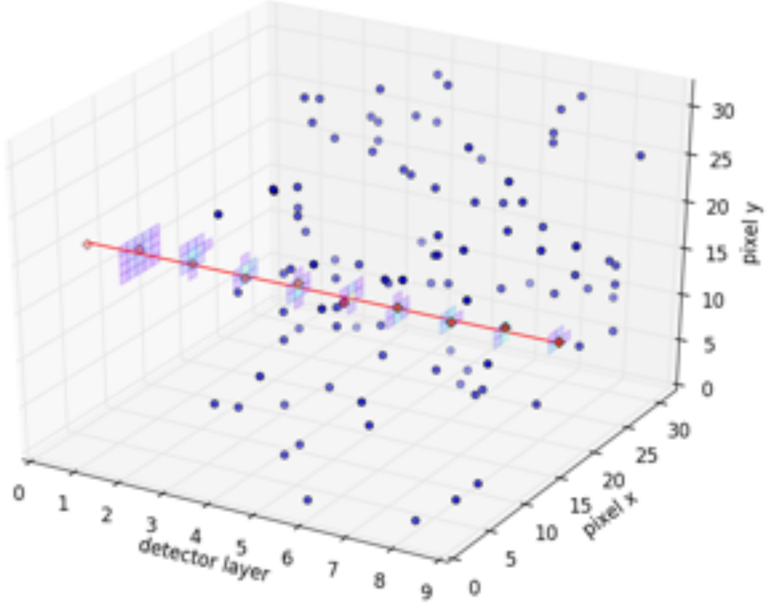
Mantissa-HEP Meeting

Mar 27, 2017

Contents

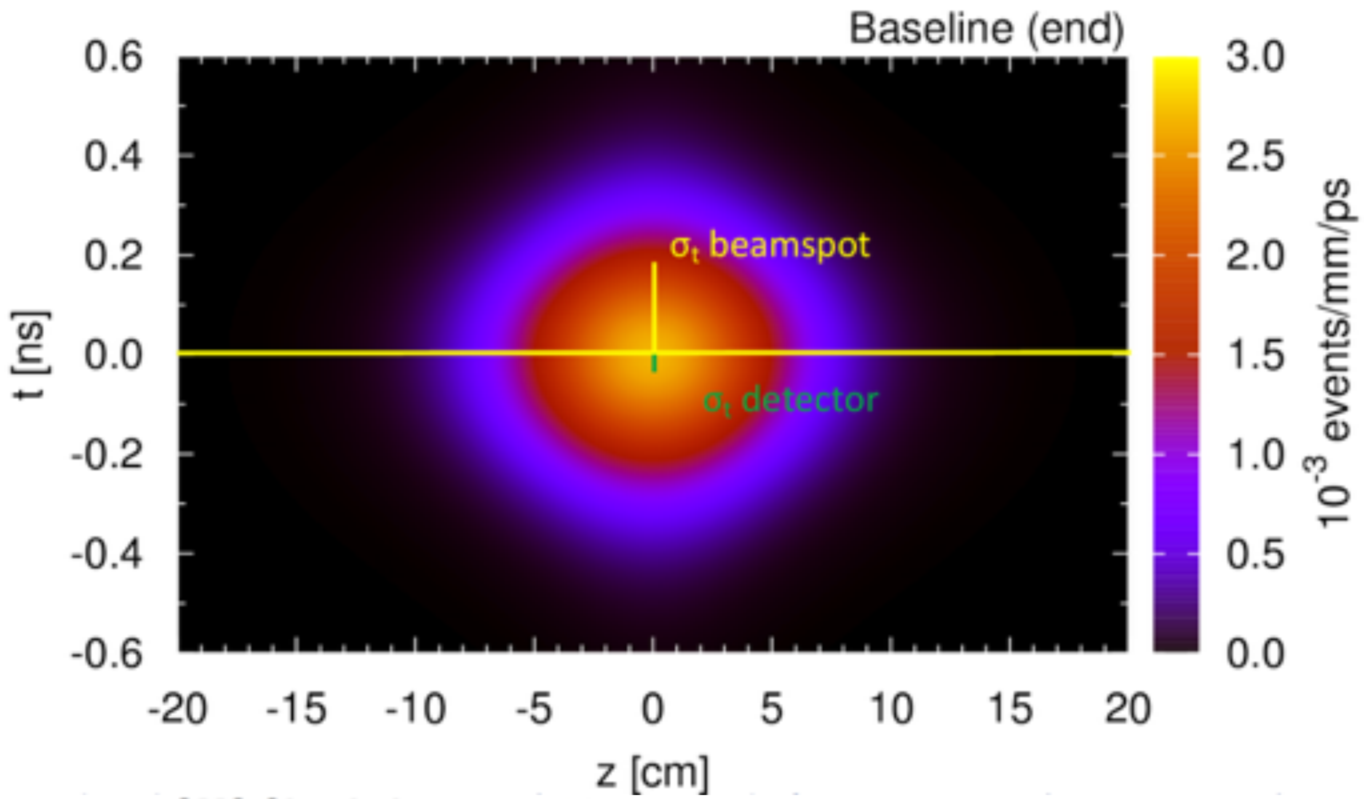
- Some (ML) highlights from the Connecting The Dots - Intelligent Trackers Workshop at LAL/Orsay

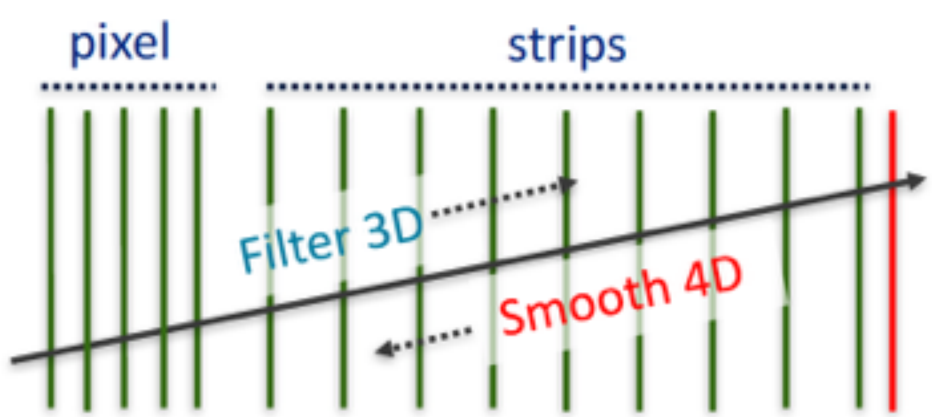
- and some HEP.TrkX stuff

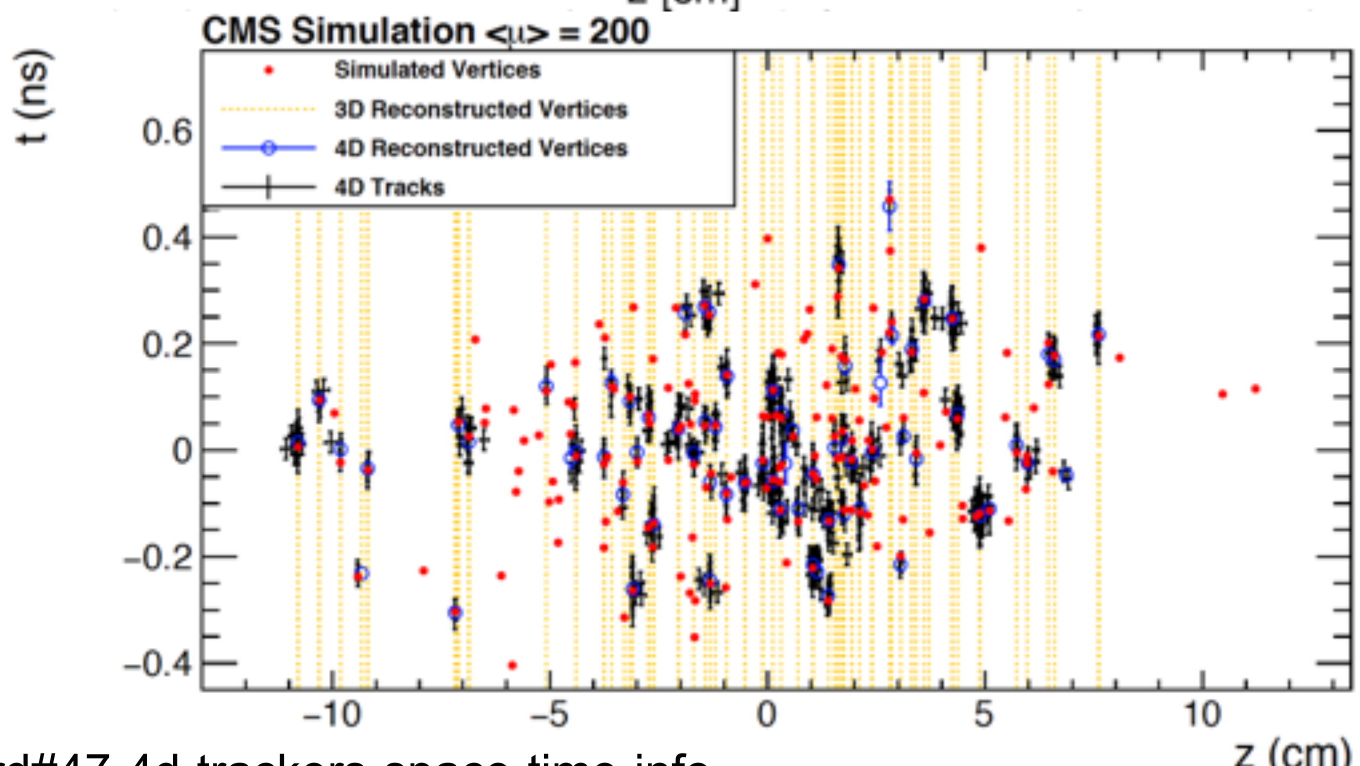


4D tracking

- At HL-LHC and future colliders, collision vertices densely distributed in space and time
  - hard to disentangle in space alone
- Special timing detectors can help
  - hardware options under study
- Even just a single timing layer at outer edge of tracker can give considerable performance benefits





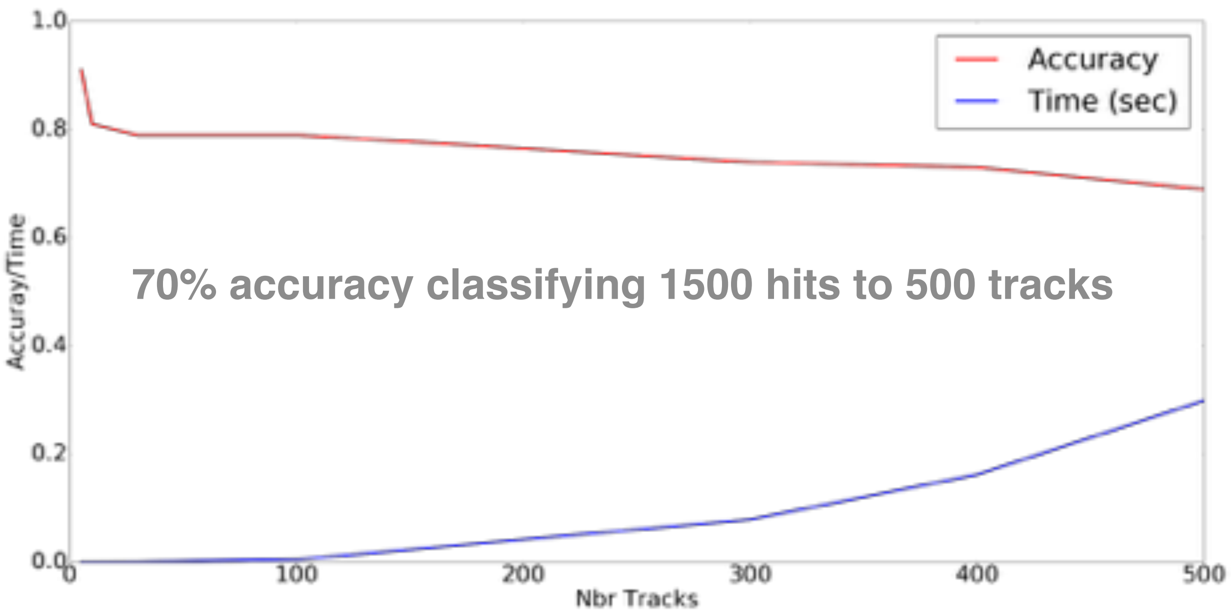


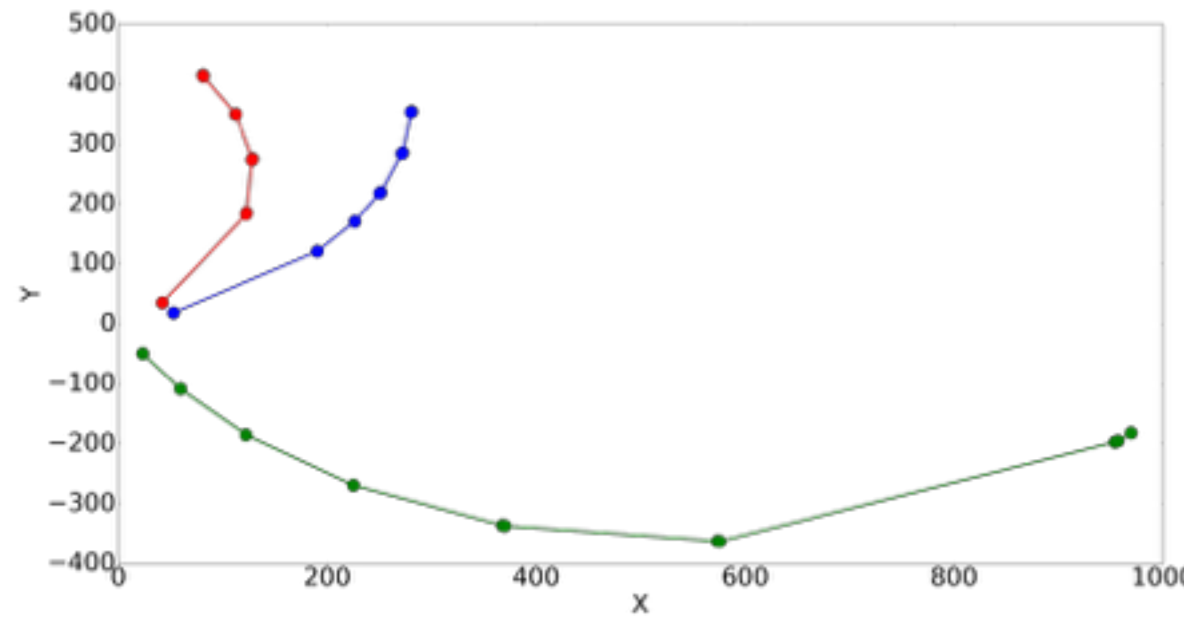
- Future trackers may be fully 4D

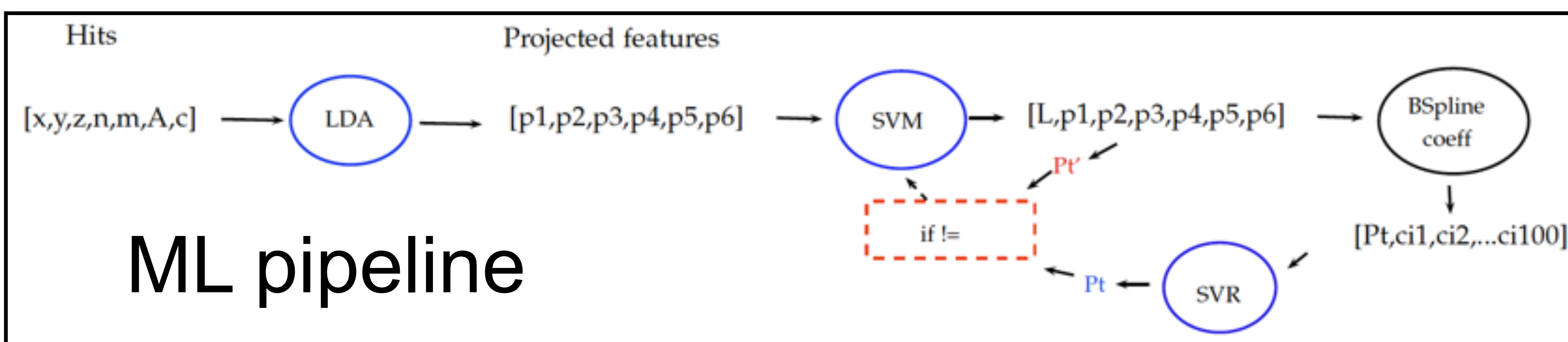
http://indico.cern.ch/event/577003/other-view?view=standard#47-4d-trackers-space-time-info

Non-parametric functional regression for track reconstruction

- Uses clustered hits in 3D space with additional cluster features
- Apply LDA to reduce dimensionality by one
- Use SVM to cluster hits into tracks
- Use support vector *regression* to get kinematics from parametric curves





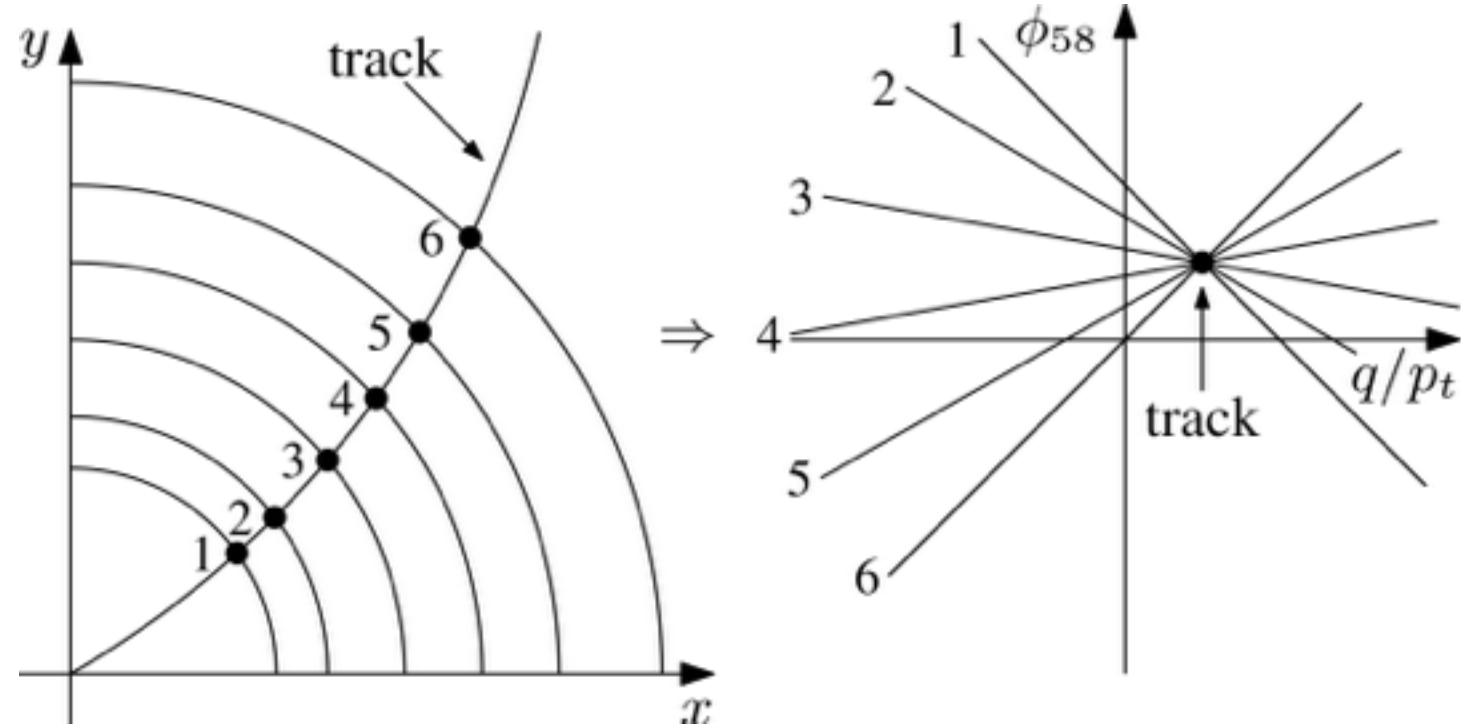


http://indico.cern.ch/event/577003/other-view?view=standard#13-young-scientist-forum-funct

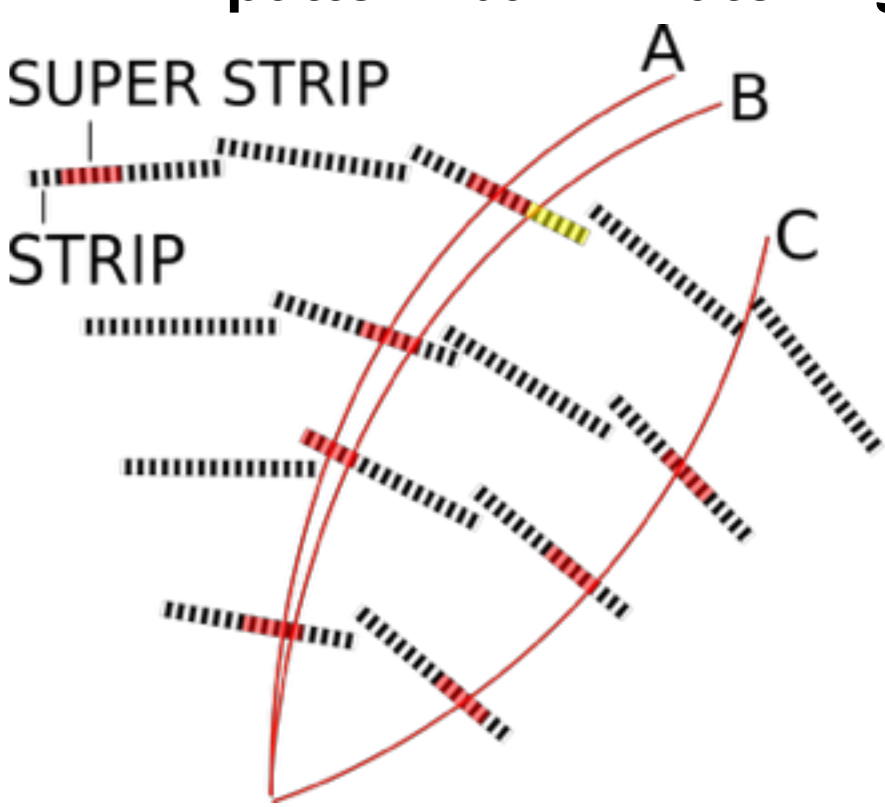
Fast pattern recognition for track triggers

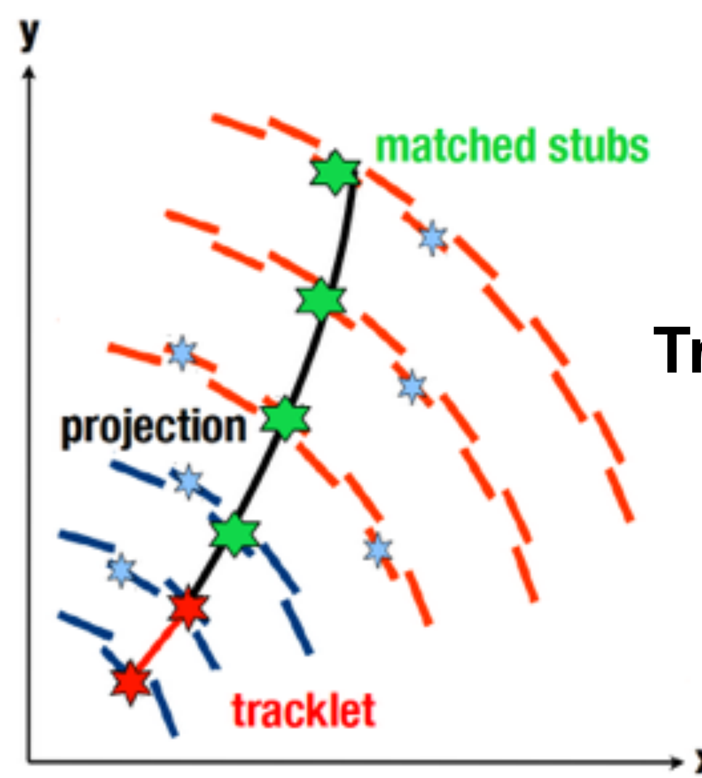
- ATLAS and CMS exploring low-latency hardware and algorithms
  - ~4 μs latencies!
- FPGA + Hough Transform
- FPGA + Associative Memory
- FPGA + Tracklet approach

Hough Transform



AM pattern bank matching

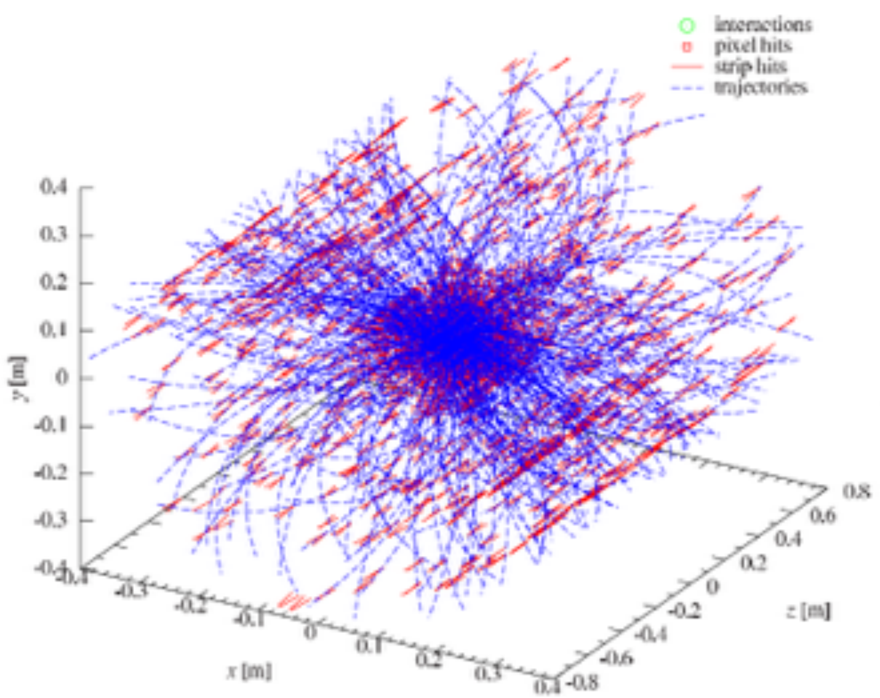


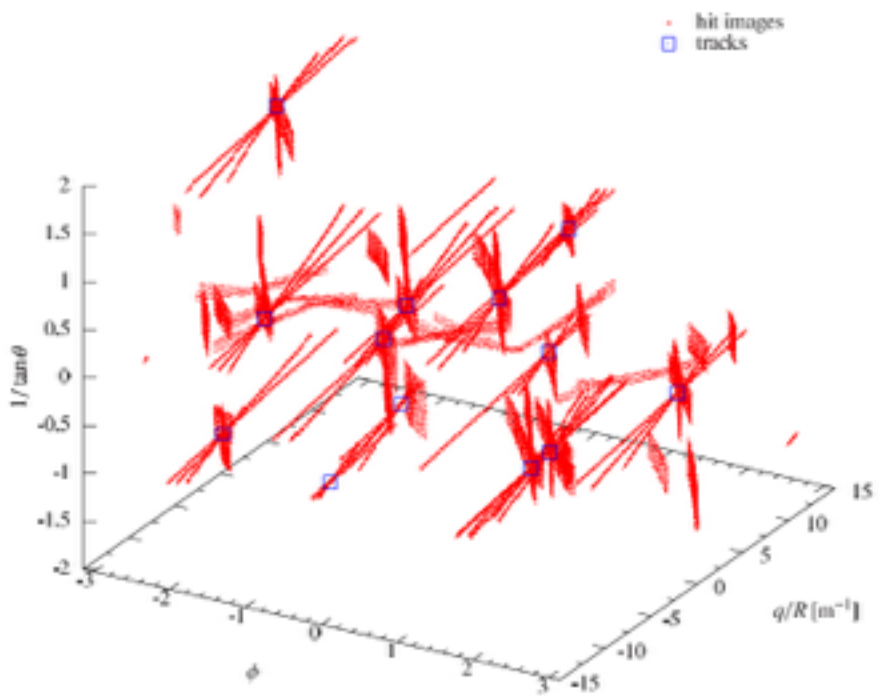


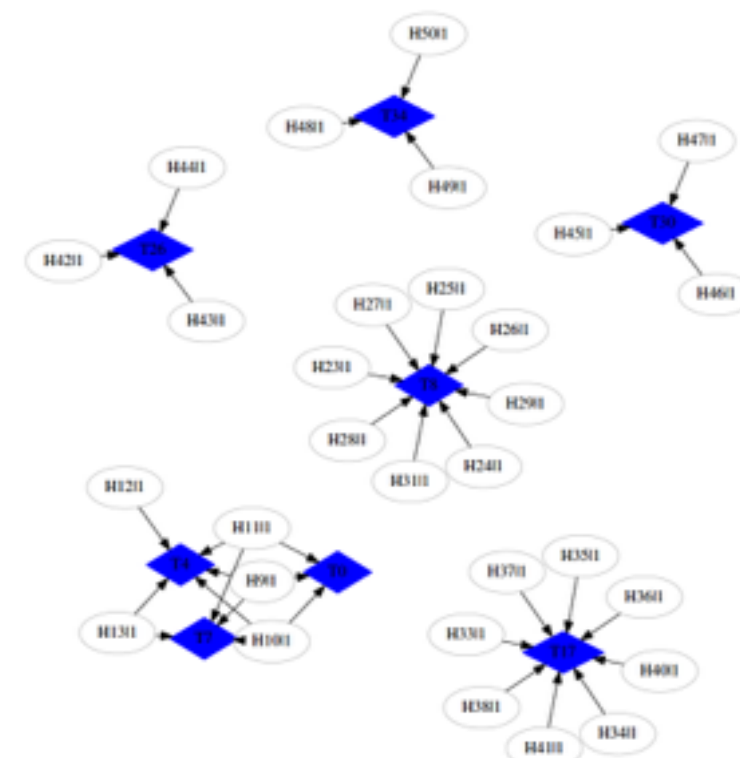
Tracklet extrapolation

Combination of data analysis techniques for efficient track reconstruction in high multiplicity events

- Cool ideas for high reconstruction efficiency, even for low PT





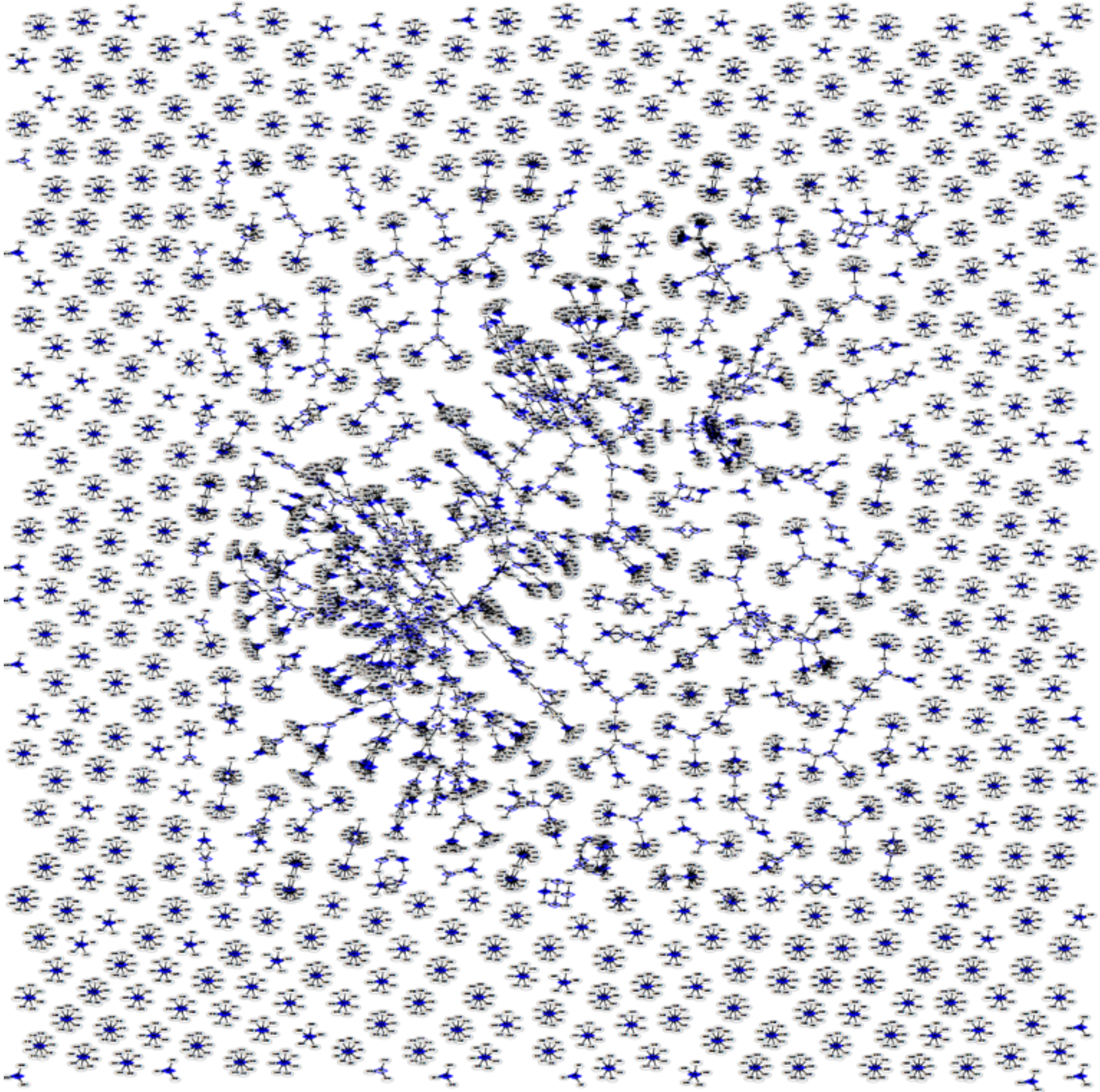


- Hough transform + template fit, then search a bipartite graph of candidates

Ferenc Siklér

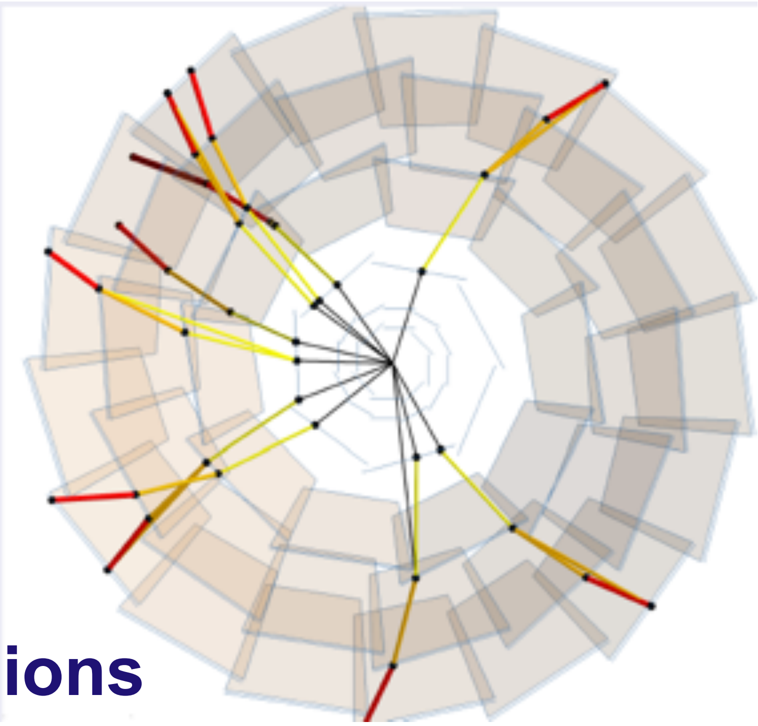
http://indico.cern.ch/event/577003/other-view?view=standard#5-combination-of-various-data

40 p-p

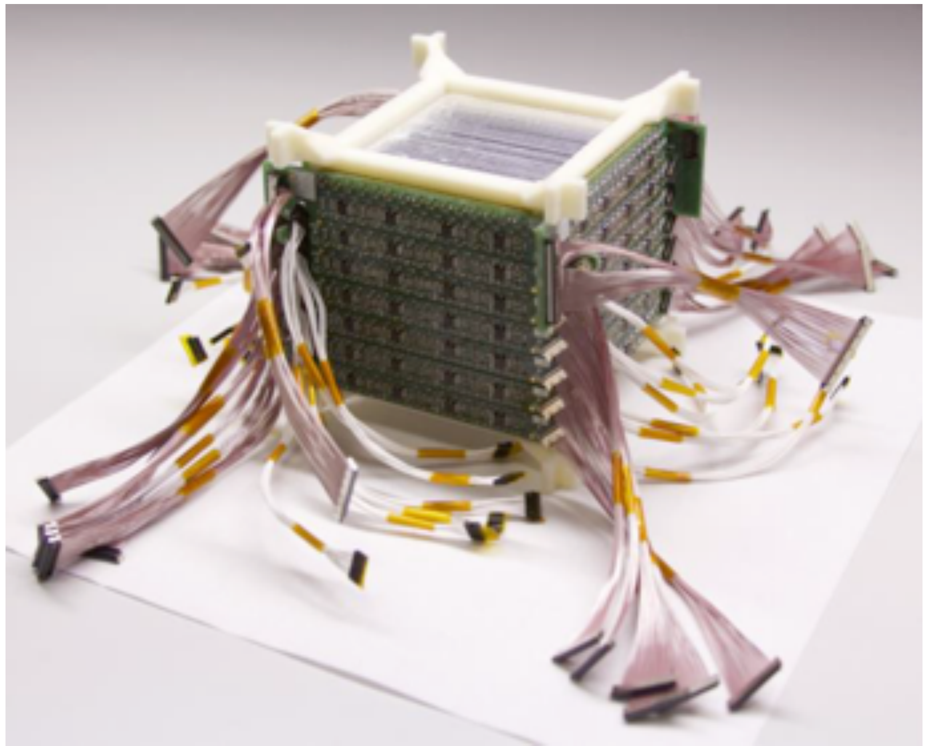


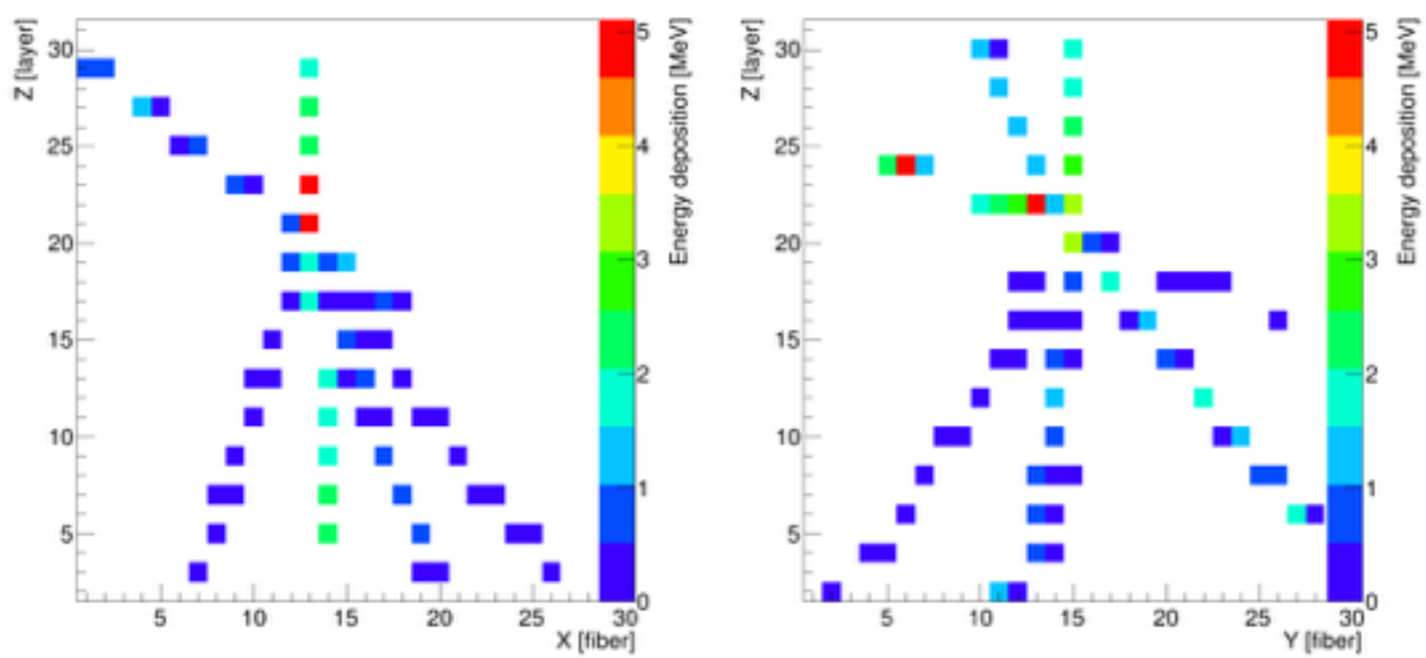
More interesting applications

- **Belle II tracking**
  - Cellular automaton connects candidates
  - Hopfield network resolves overlaps



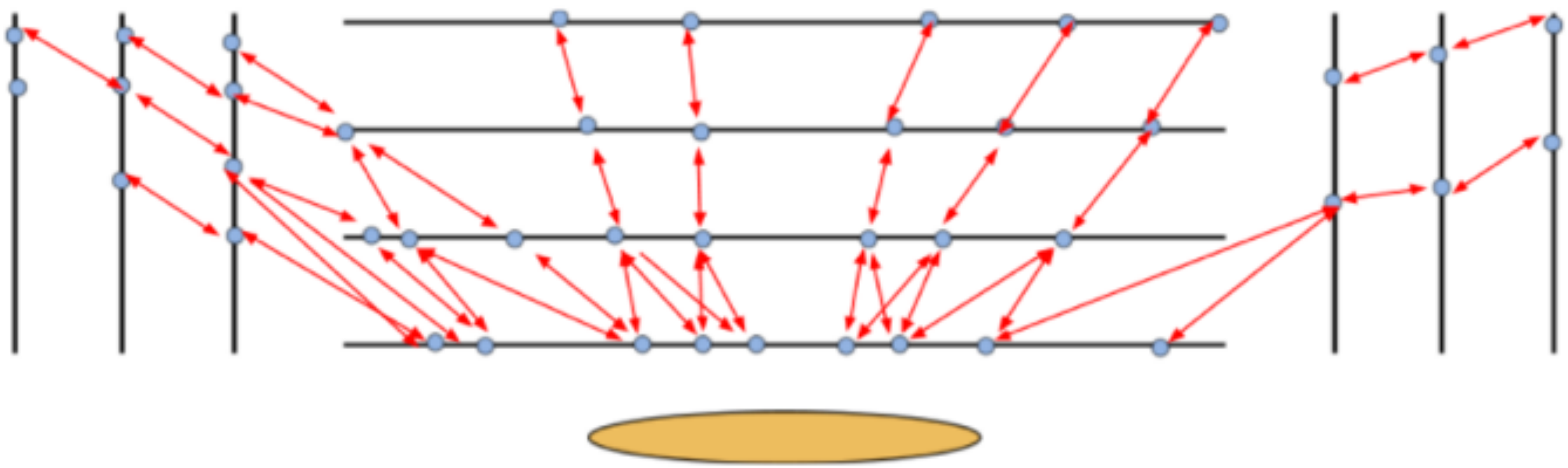
- **A multi-purpose particle detector for space missions**
  - Bayesian particle filter or MCMC likelihood for precision
  - Exploring HT and NNs for fast online analysis



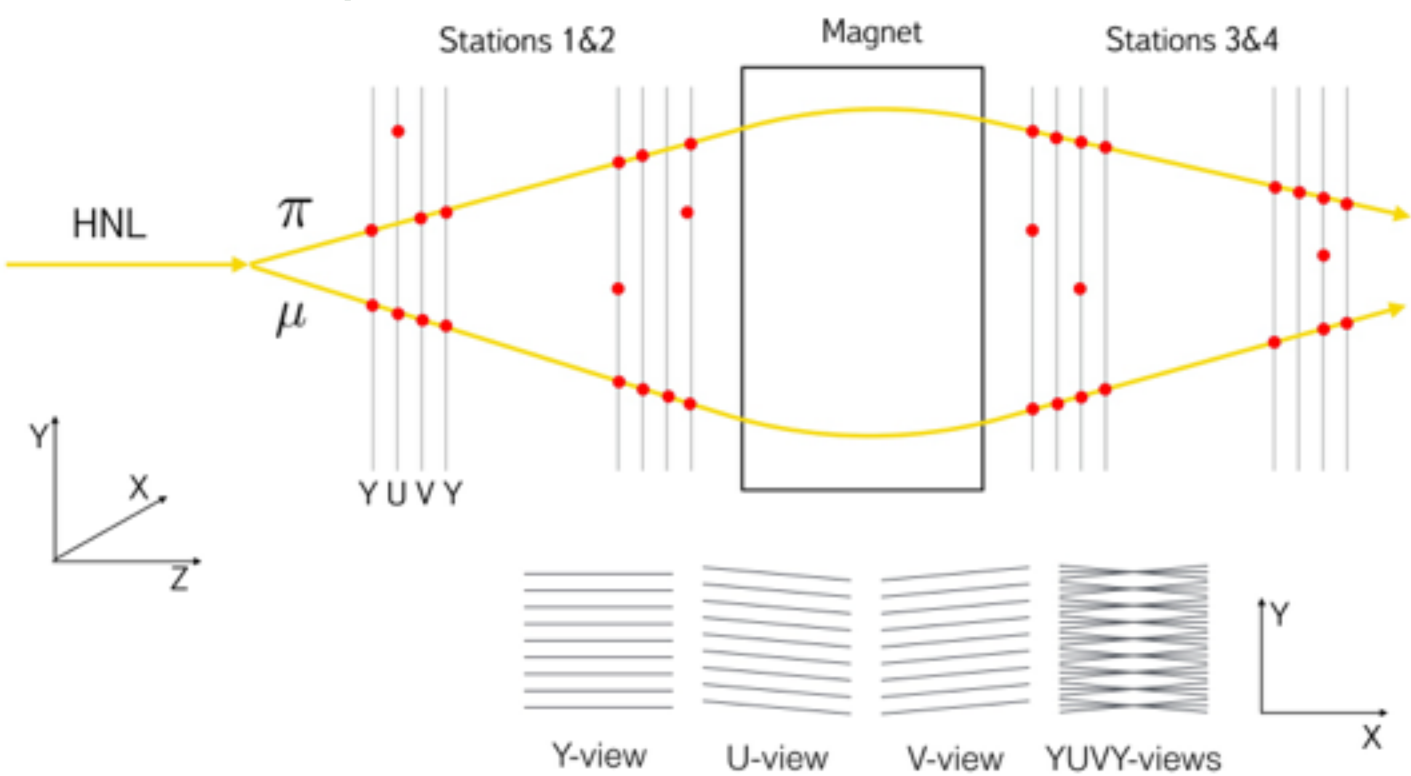


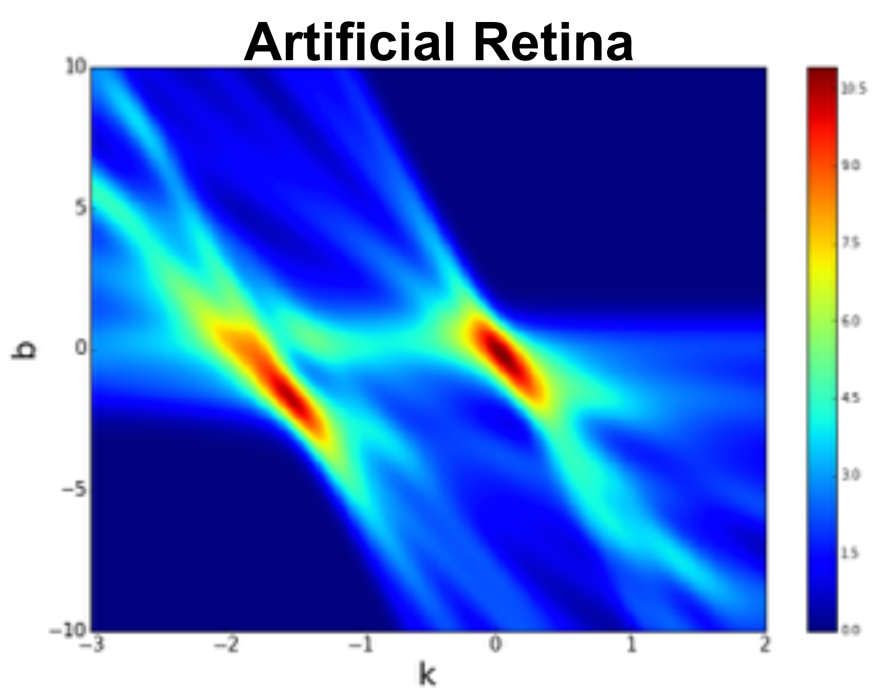
More interesting applications

- **Cellular automaton for CMS track seeding**



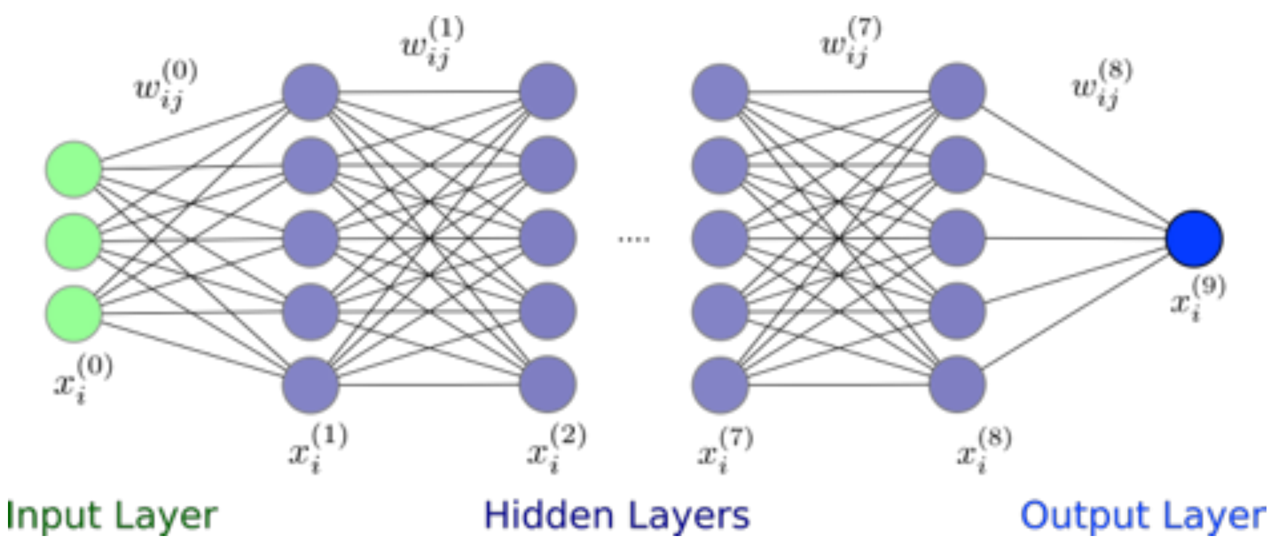
- **SHiP Spectrometer tracker**
  - Compared RANSAC, HT, and Artificial Retina

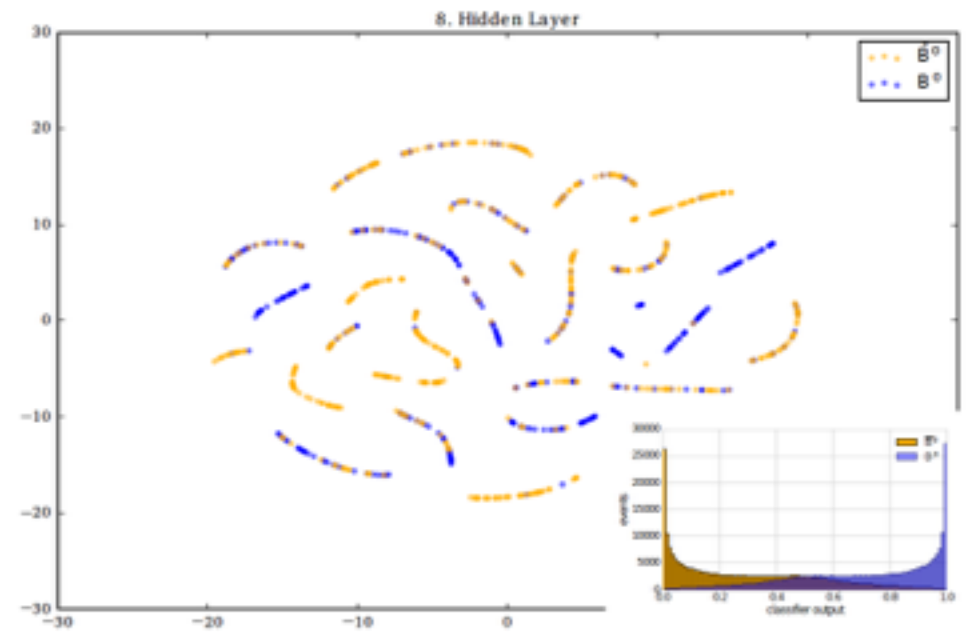




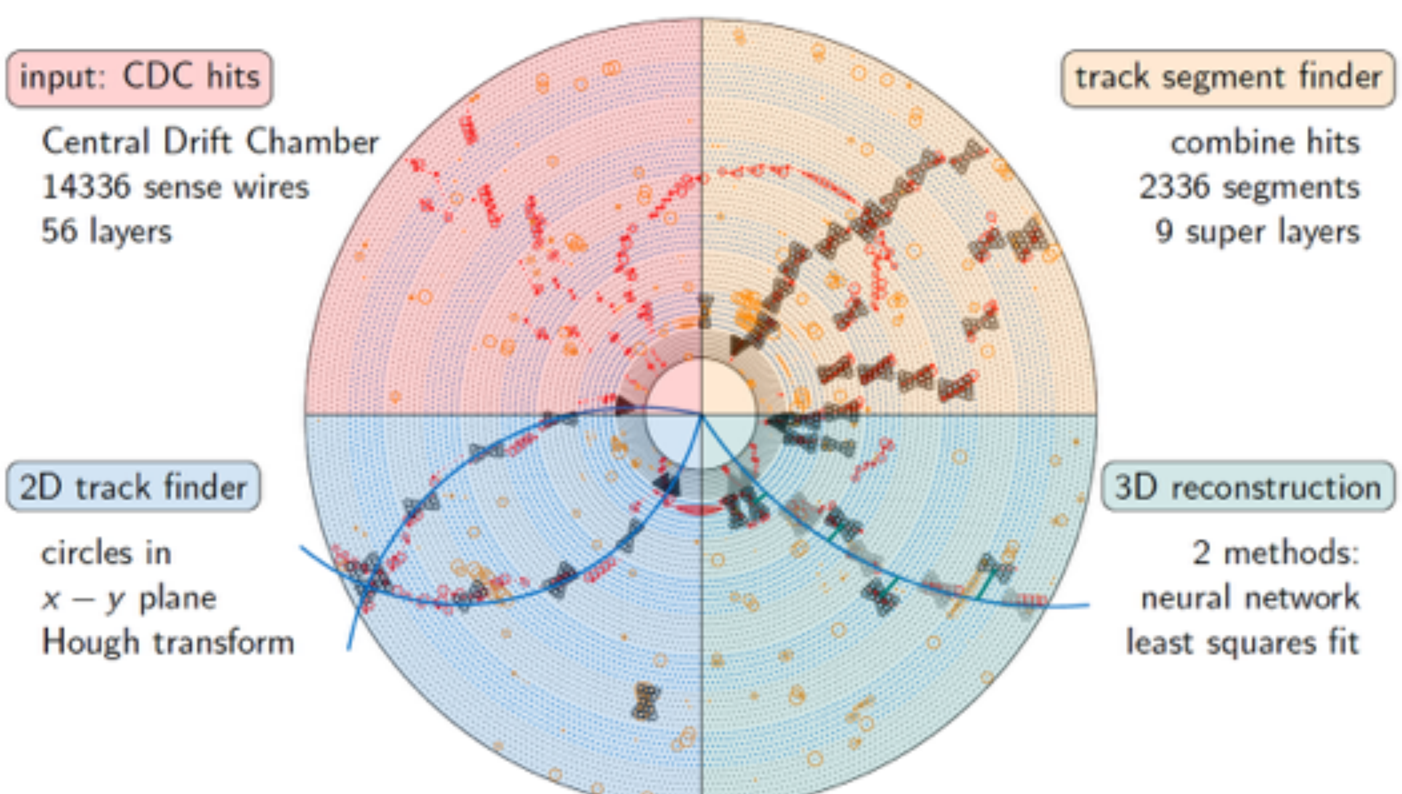
Deep learning applications

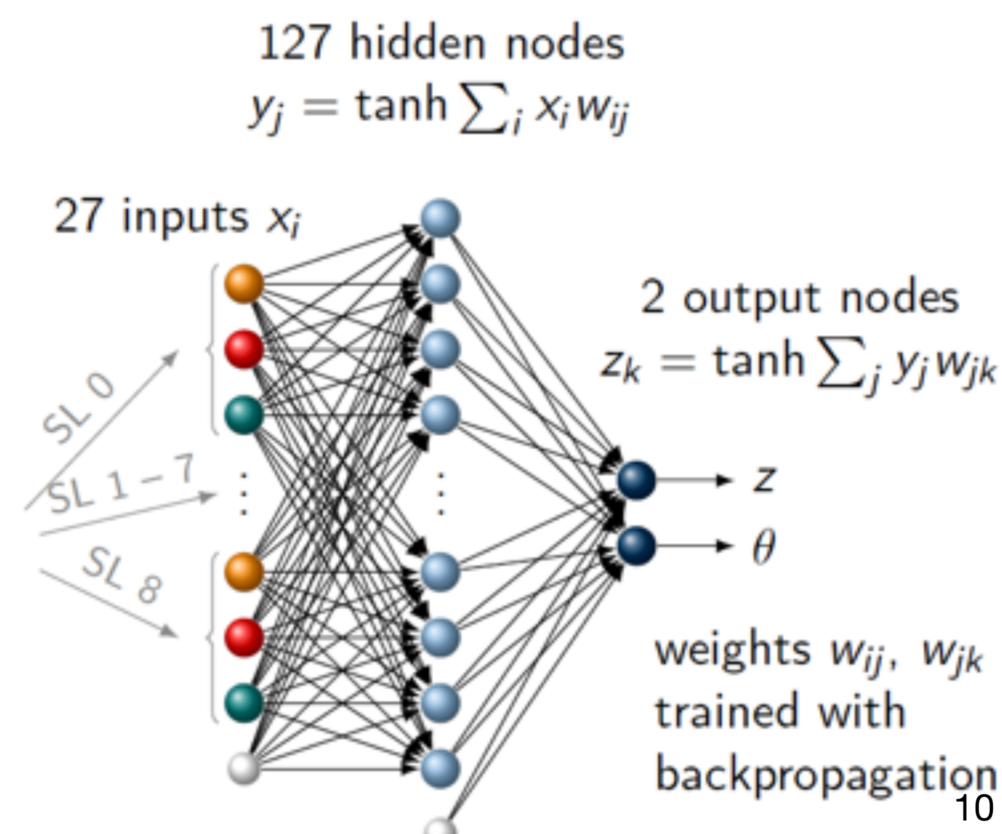
- Flavor tagging with DNNs in Belle II
  - 140 tracking features => 9 layer MLP





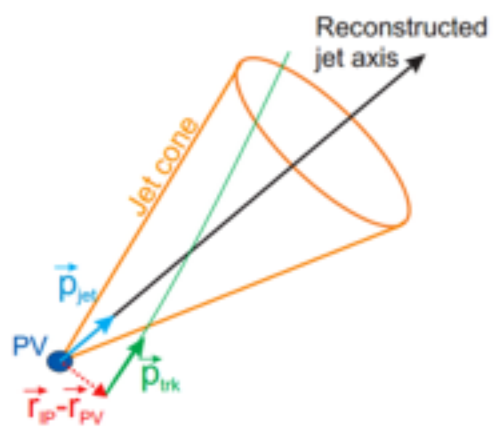
- Vertex reconstruction with NNs at Belle II

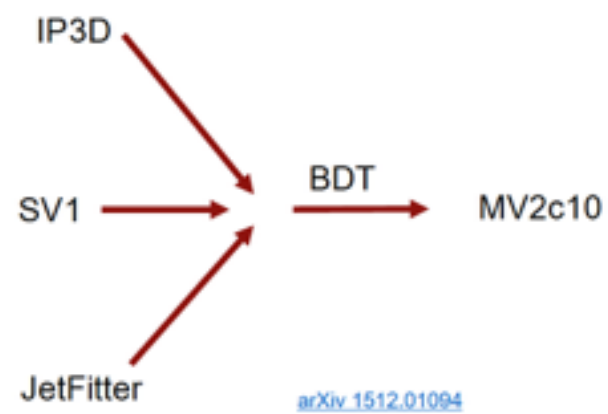


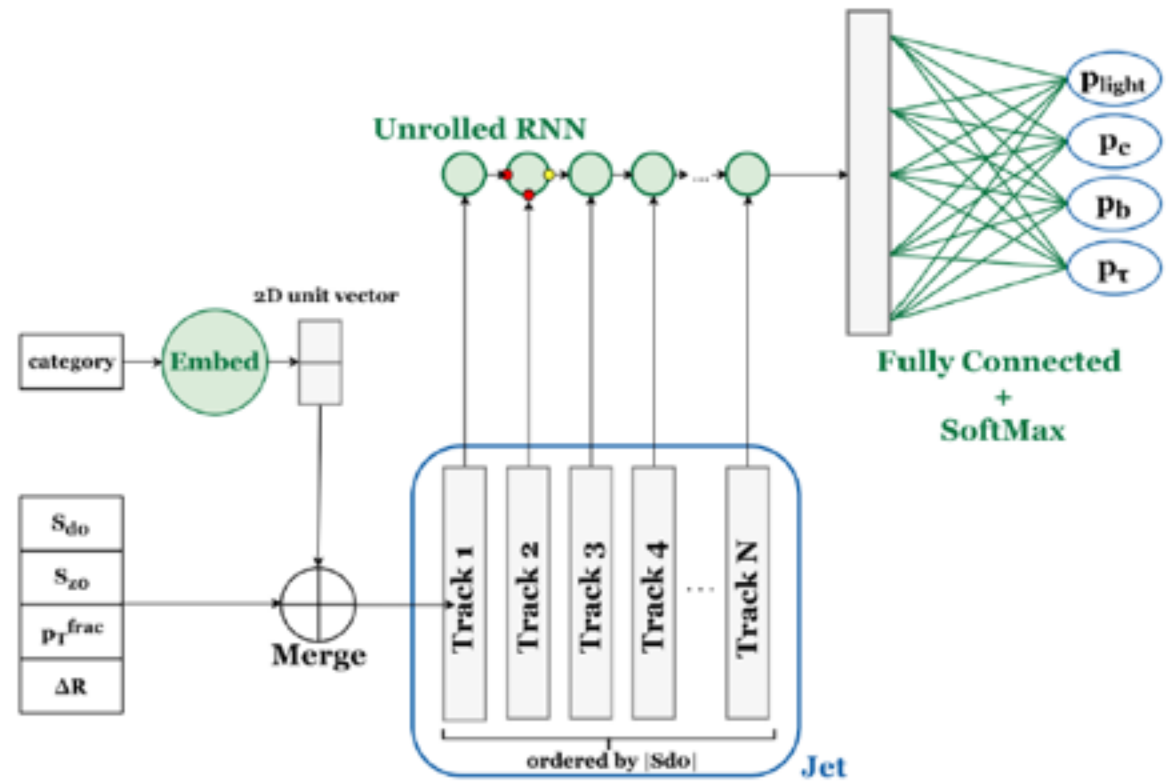


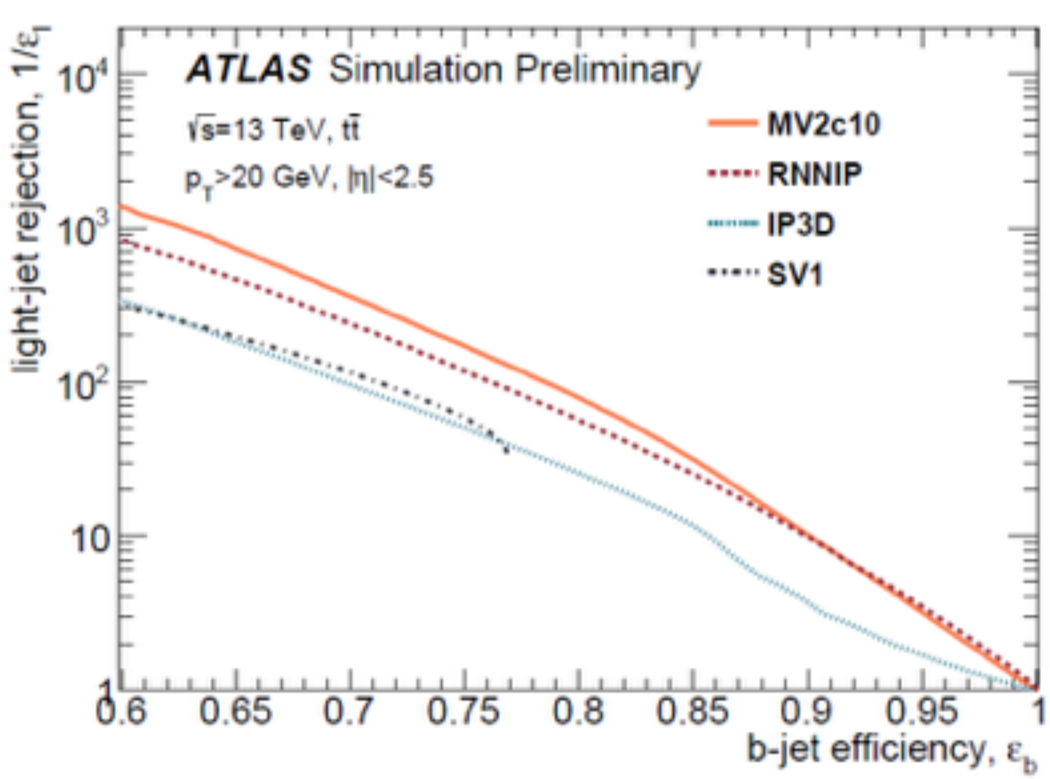
Deep learning applications

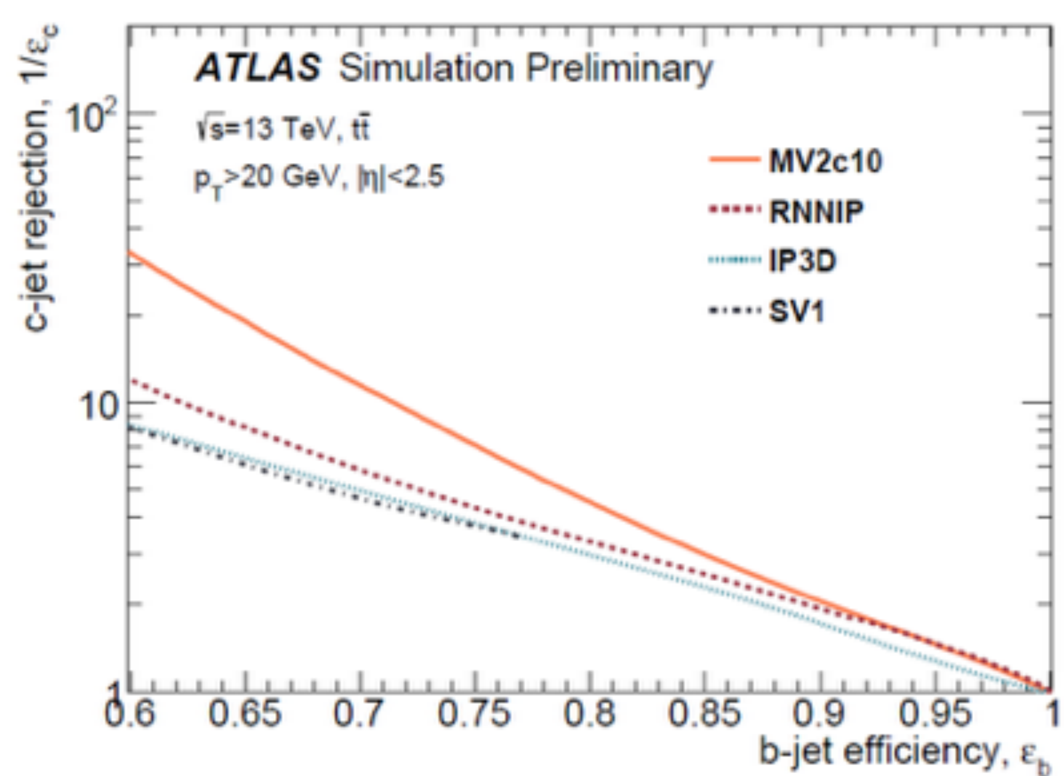
- LSTMs for b-tagging in ATLAS
  - easily beats other “baseline” taggers
  - but doesn’t replace high-level tagger









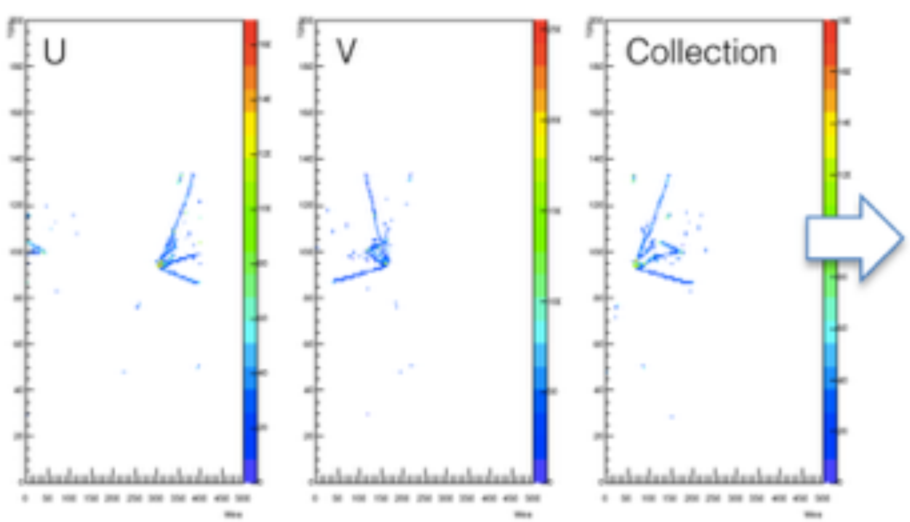


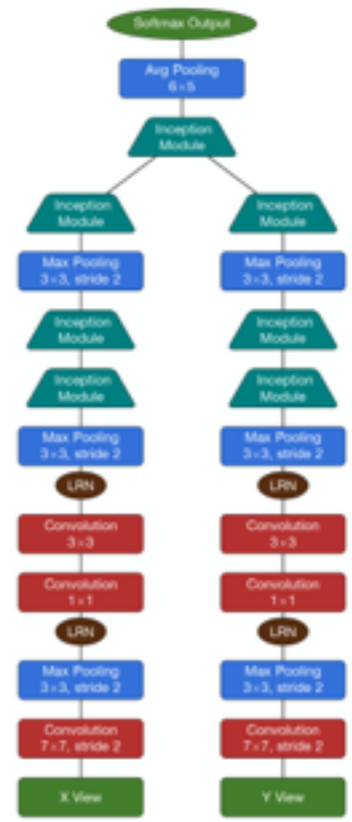
http://indico.cern.ch/event/577003/other-view?view=standard#41-young-scientist-forum-ident

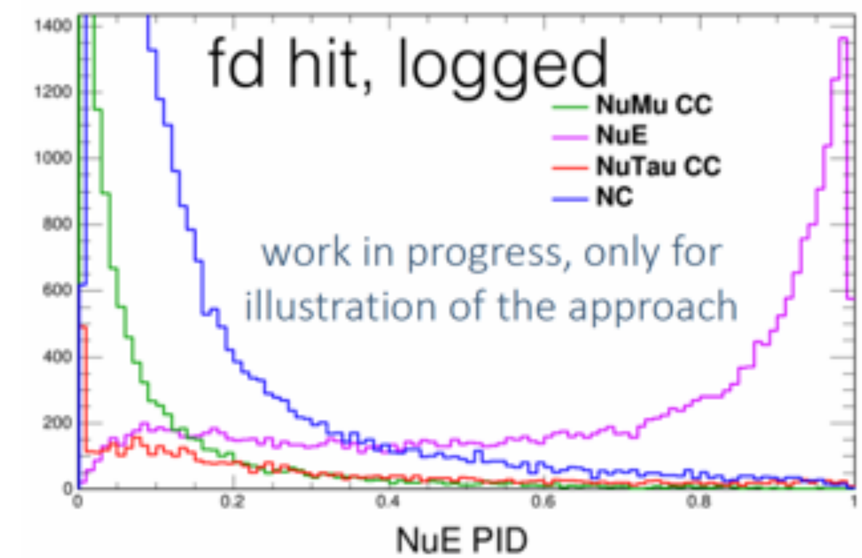
ML for neutrino experiments (e.g. DUNE, NOvA)

- Deep nets for classification
  - Not new, but now used by DUNE

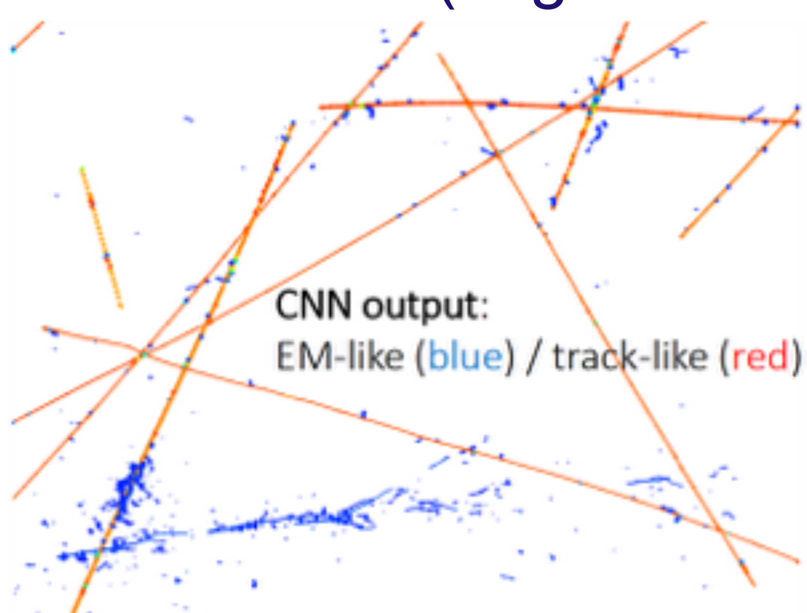
DUNE:

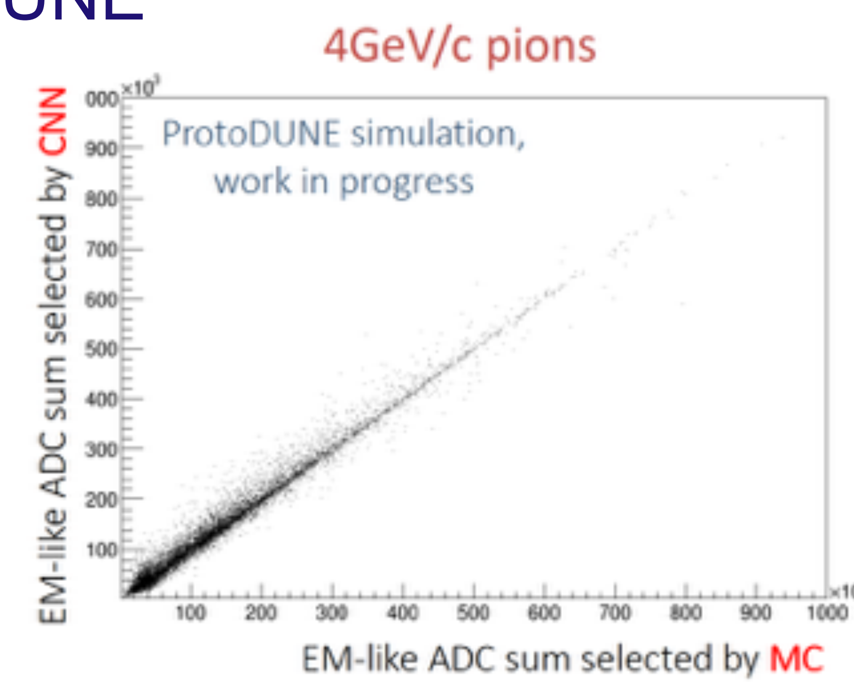






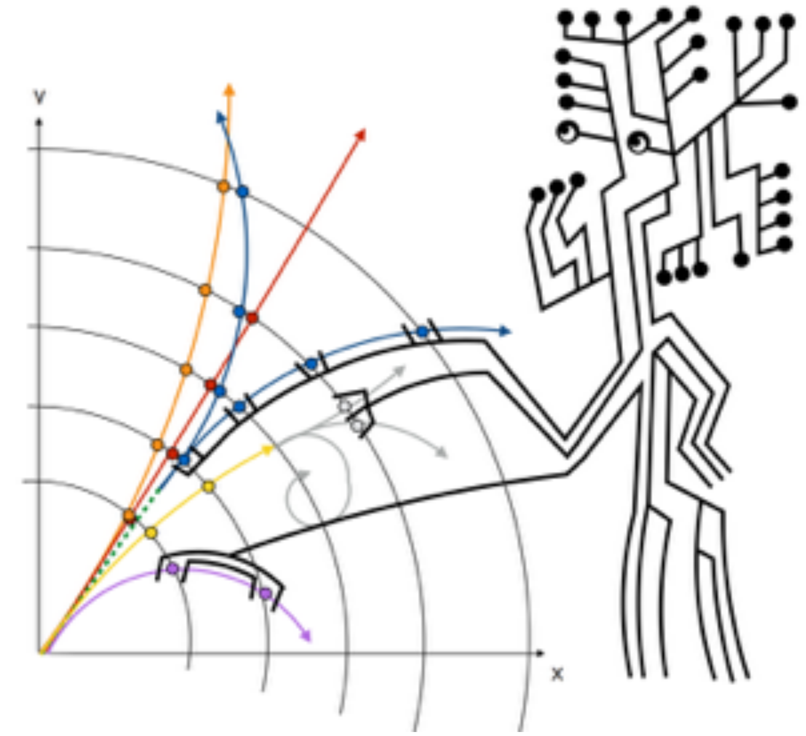
- Pixel-level classification (segmentation) in DUNE





http://indico.cern.ch/event/577003/other-view?view=standard#59-machine-learning-approach-t

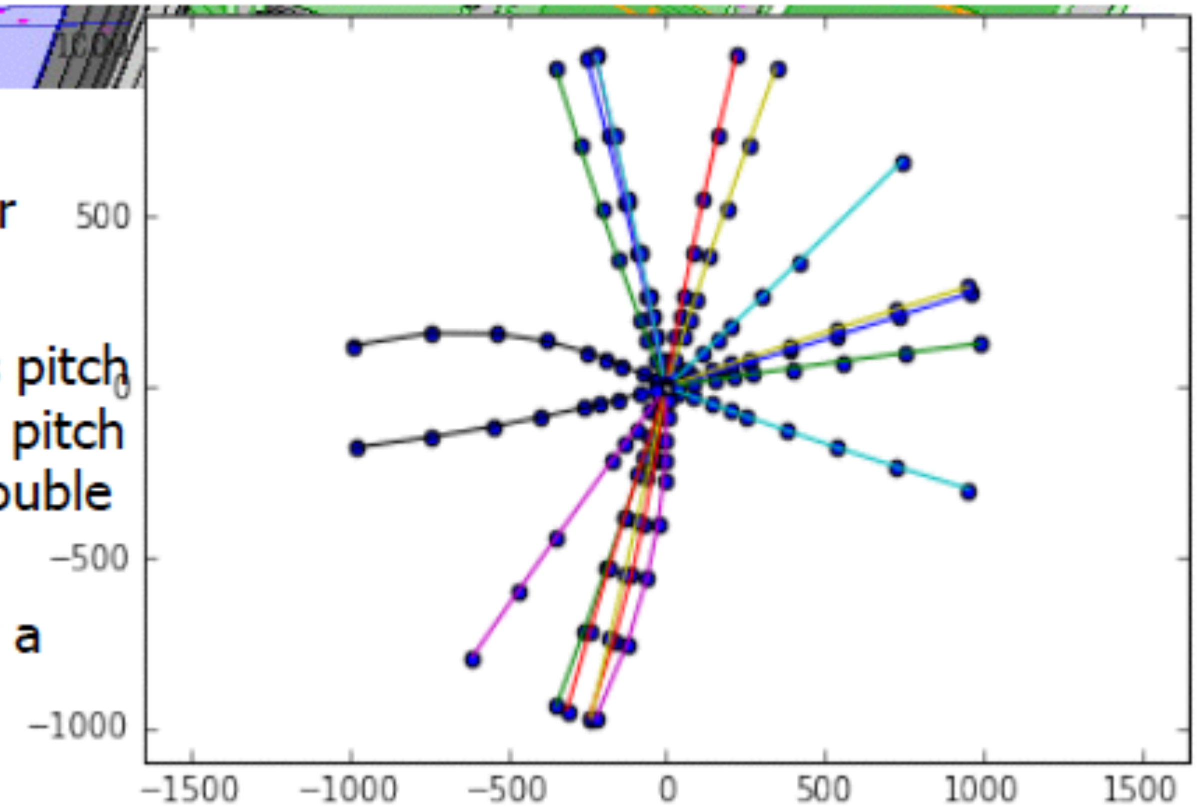
Tracking Machine Learning Challenge



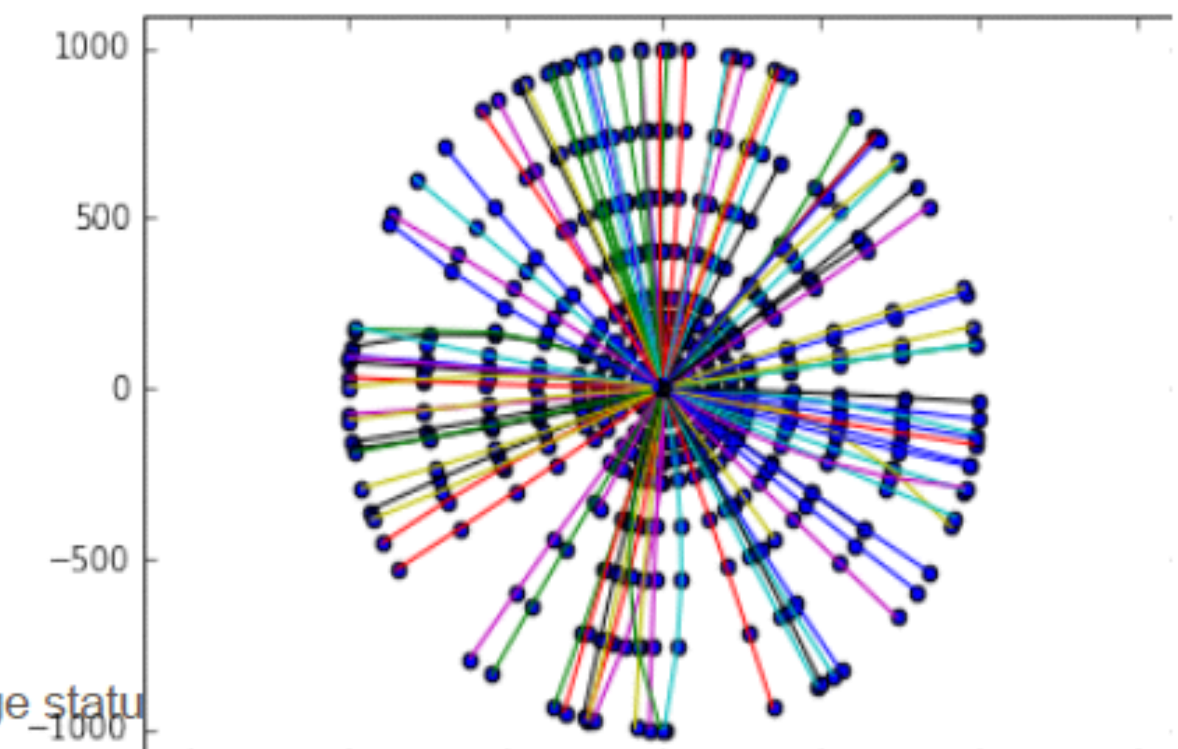
- Born out of CTD2015 at LBL, still in development
- ACTS dataset
  - Generic detector with non-uniform magnetic field, semi-realistic material and detector resolution effects
  - Realistic events (tt + ~200 pileup), ~0.5MB/event
    - => 500GB for 1M events…
- Challengers must cluster the hits together into tracks
  - Figure of merit built from efficiency, fake rate
    - weighted towards high track efficiency for high-momentum tracks
  - Evaluation time will be measured (somehow)
    - probably focused on more in a later challenge
- Still debating platform
  - though proposal submitted for NIPS competition

http://indico.cern.ch/event/577003/other-view?view=standard#50-status-of-track-machine-lea

TrackMLRamp 2D simulation

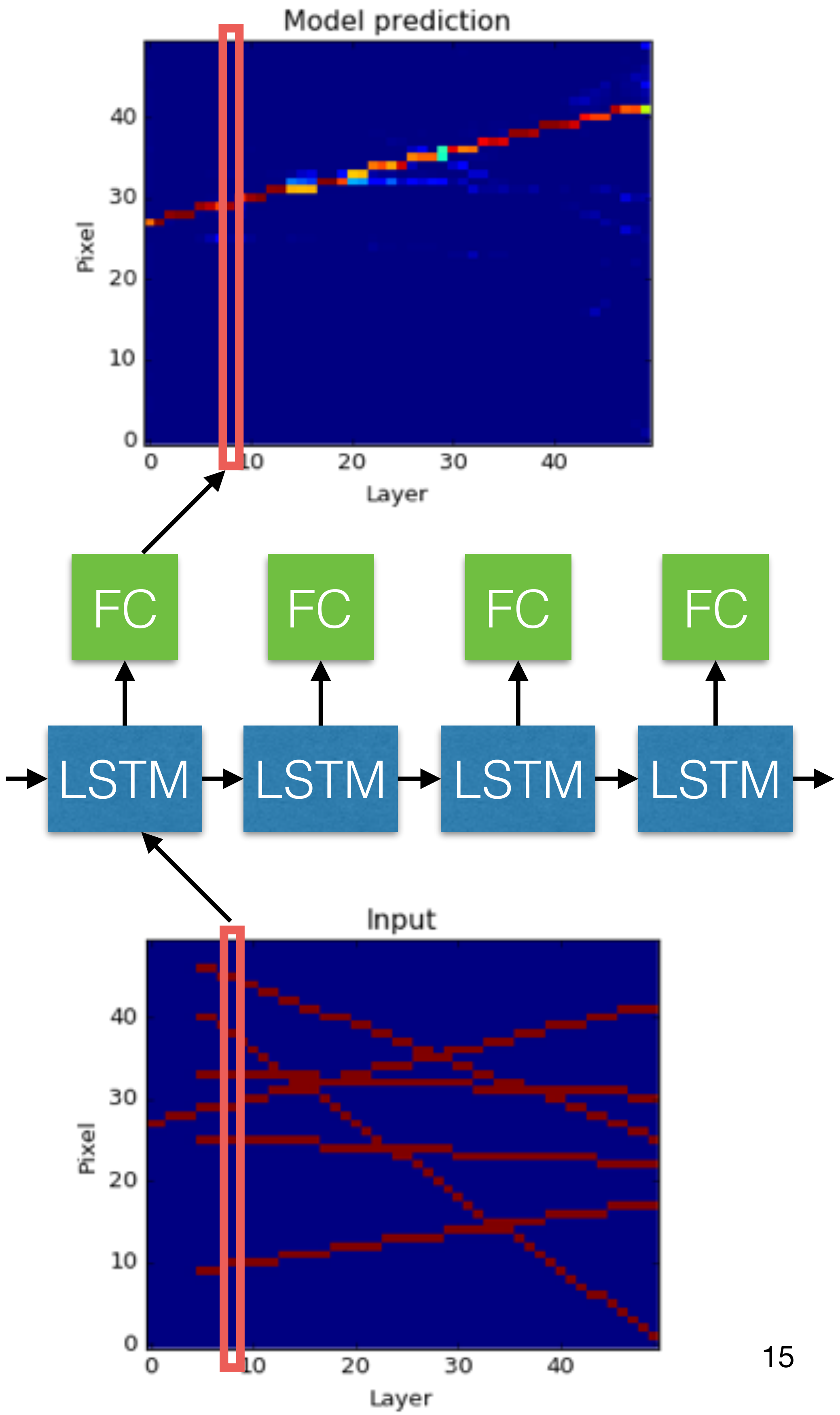


- Written in python (numpy, pandas)
- 2D simulation, detectors are perfectly circular
- Unit mm and MeV
- Use typical HL-LHC detector layout : 5 layers pitch 25um, radii {39,85,155,213,271} , +4 layers pitch 50um radiii {405,562,762,1000} (simulate double layer strip 75um)
- Digital read out : a hit is a "pixel" crossed by a track
- Constant magnetic field 2T
- Multiple scattering 2% radiation length each layer: σ_ϕ=13.6 MeV $\sqrt{(0.02)}$/P (MeV)
- Hit inefficiency 3%
- Particle stopping probability 1% per layer
- Particle gun :
  - uniform phi distribution baseline
  - Poisson ~10 tracks per event
  - Momentum : flat 300 MeV to 10 GeV
  - Origin vertex spread : σ_x=σ_y =2/3. mm
    - Each track has a different vertex



LSTM model for building a track

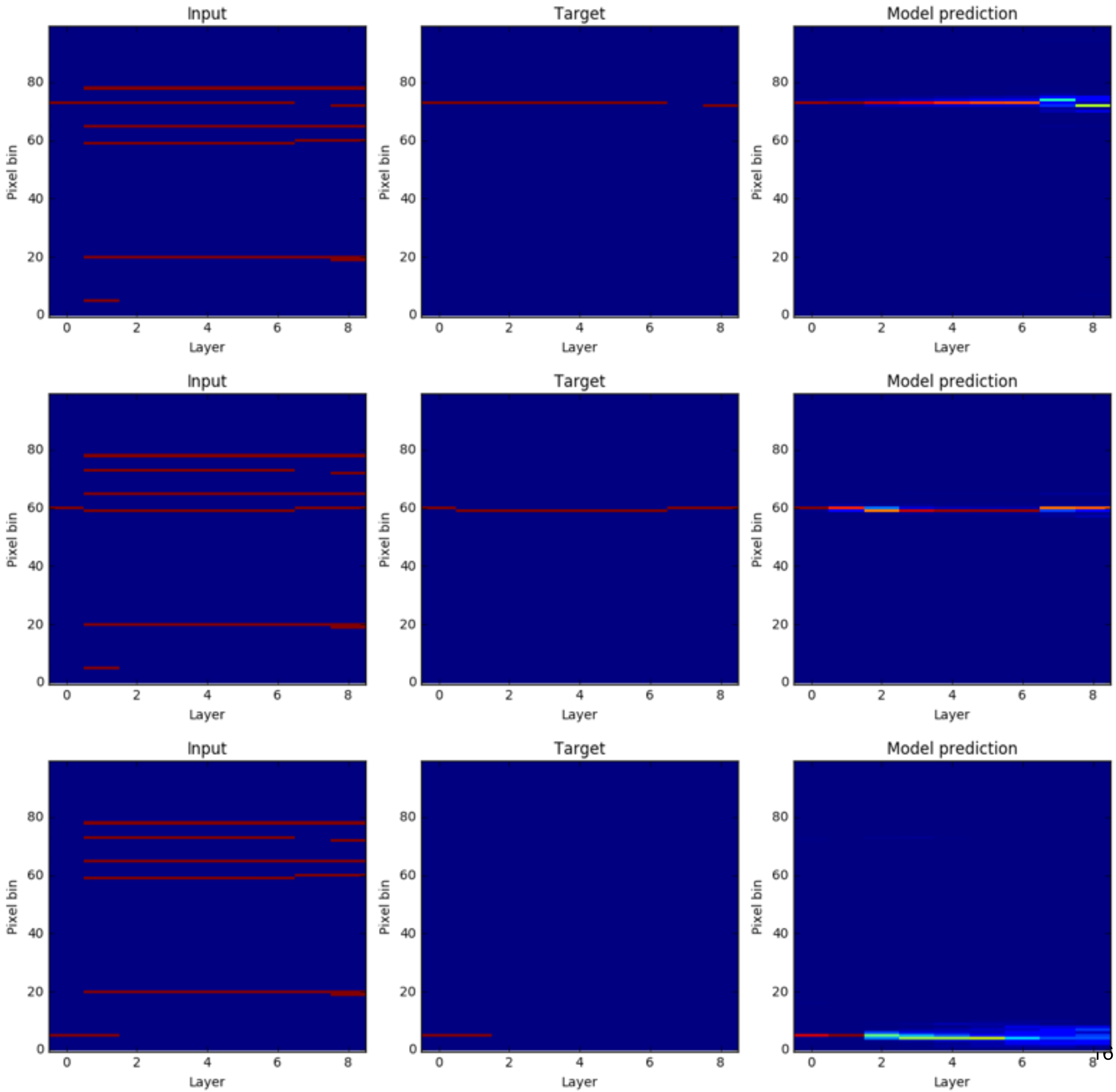
- Try to build a single, *seeded* track from a set of hits with backgrounds
- Detector plane pixel arrays fed into the model one at a time
- The model spits out an array of “scores” for that detector plane
    - Pixel predictions (or hit “classification”)
- The LSTM memory is used to carry the dynamic state estimate, updated at each iteration
- The model may consider multiple candidate paths, but hopefully converges on correct one



LSTM applied to RAMP challenge

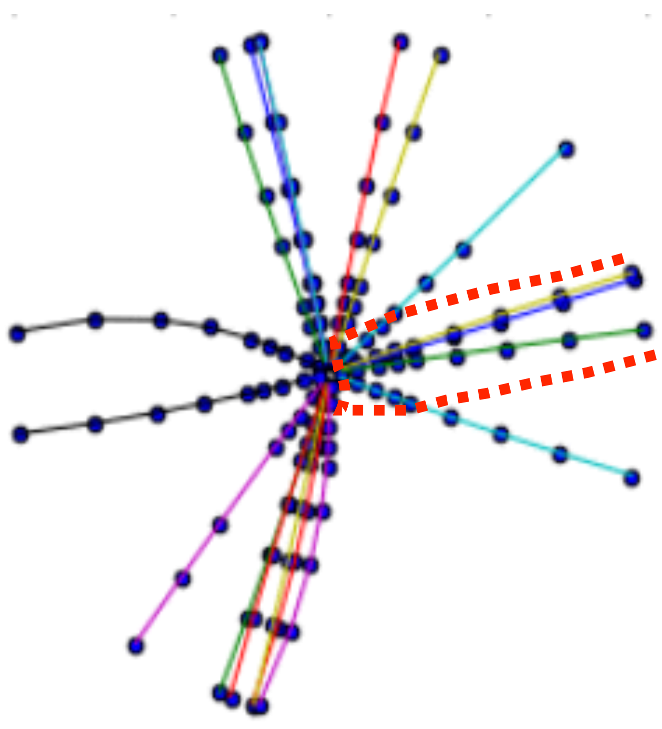
- Rebin phi to 200 bins in each layer
- Use first layer hits as seeds
- Loop over seeds, use LSTM to score hits
- For each hit, take best track assignment as label

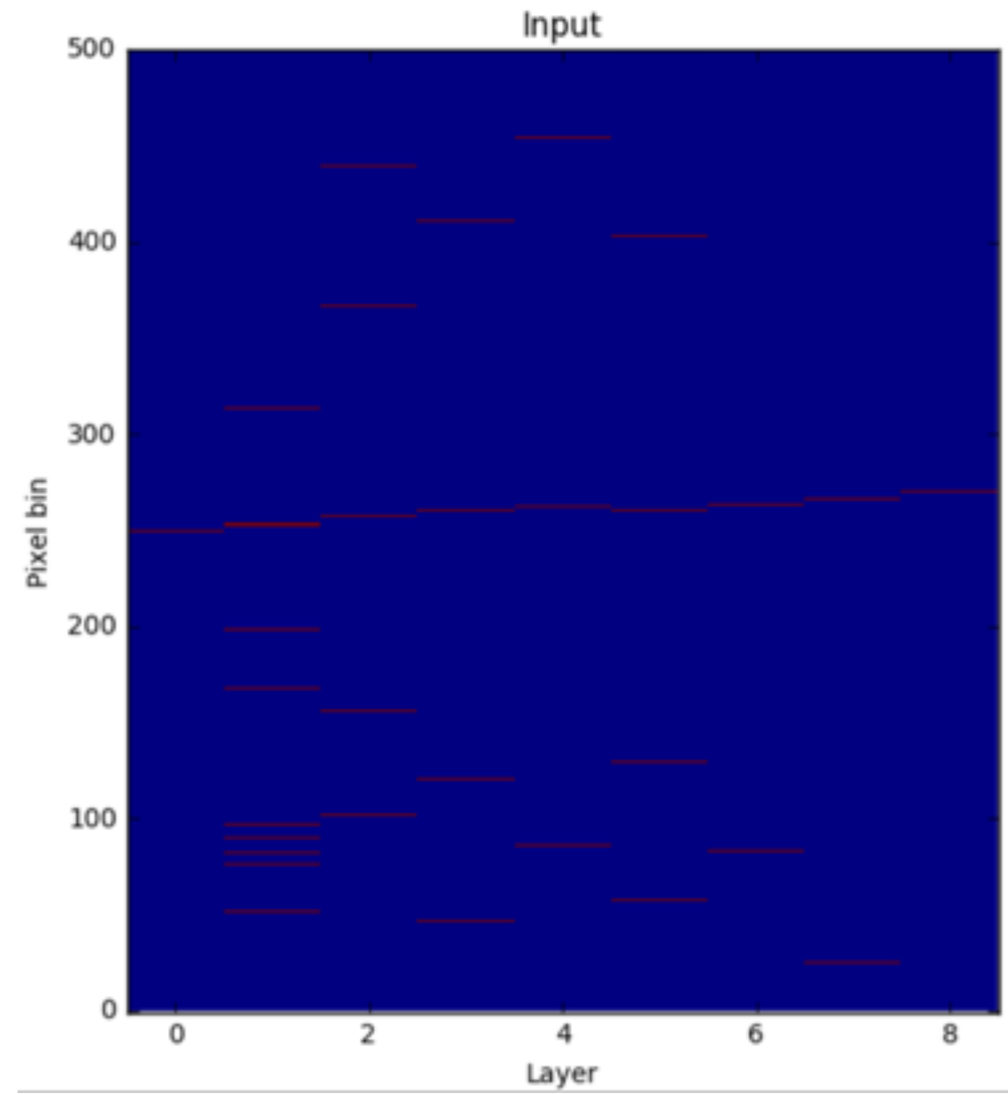
92.1% efficient



A later improvement

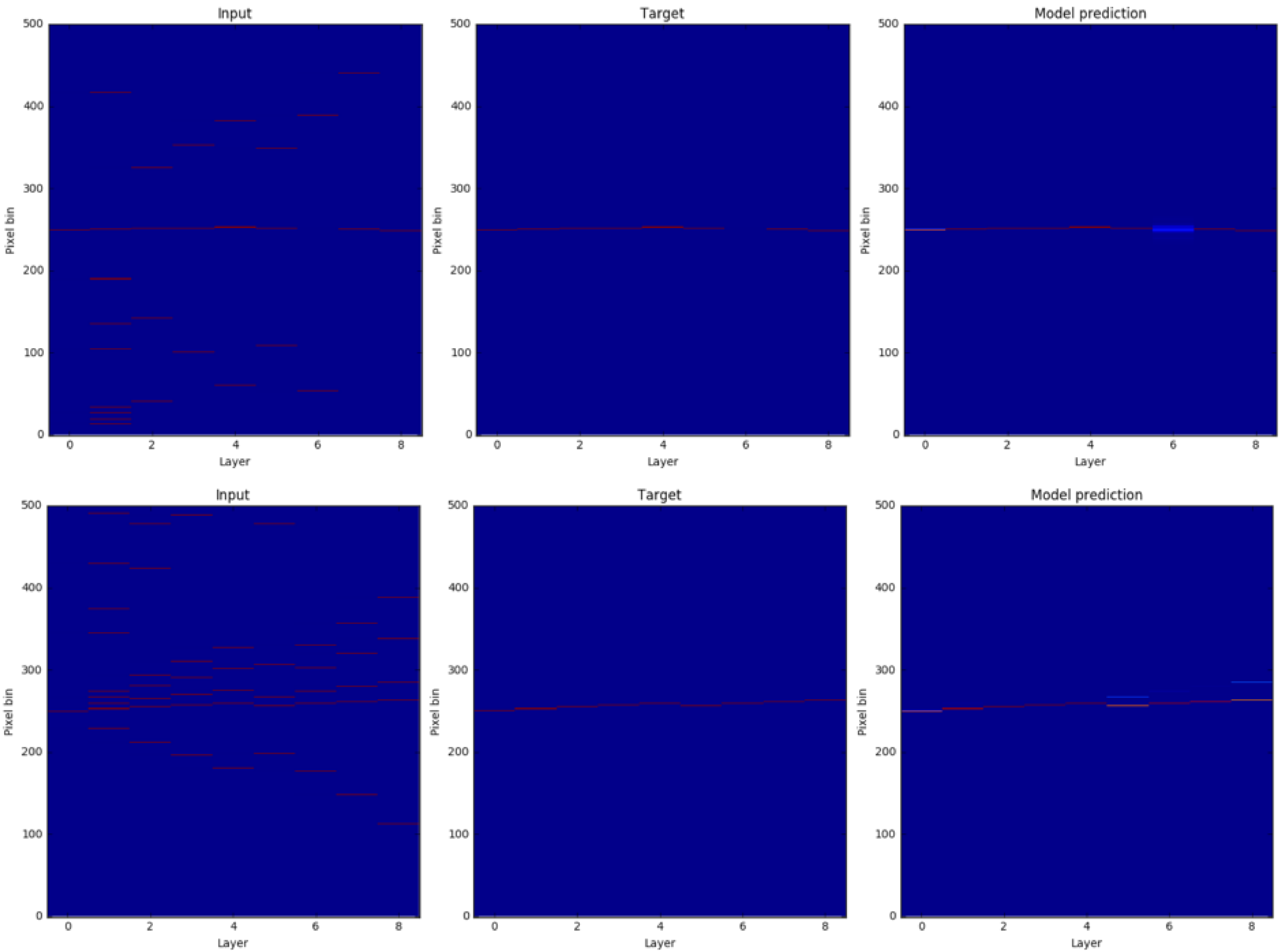
- Take fixed number of pixels per-layer as a window around the track seed





- Use a little higher granularity
  - 50 pixels per phi bin

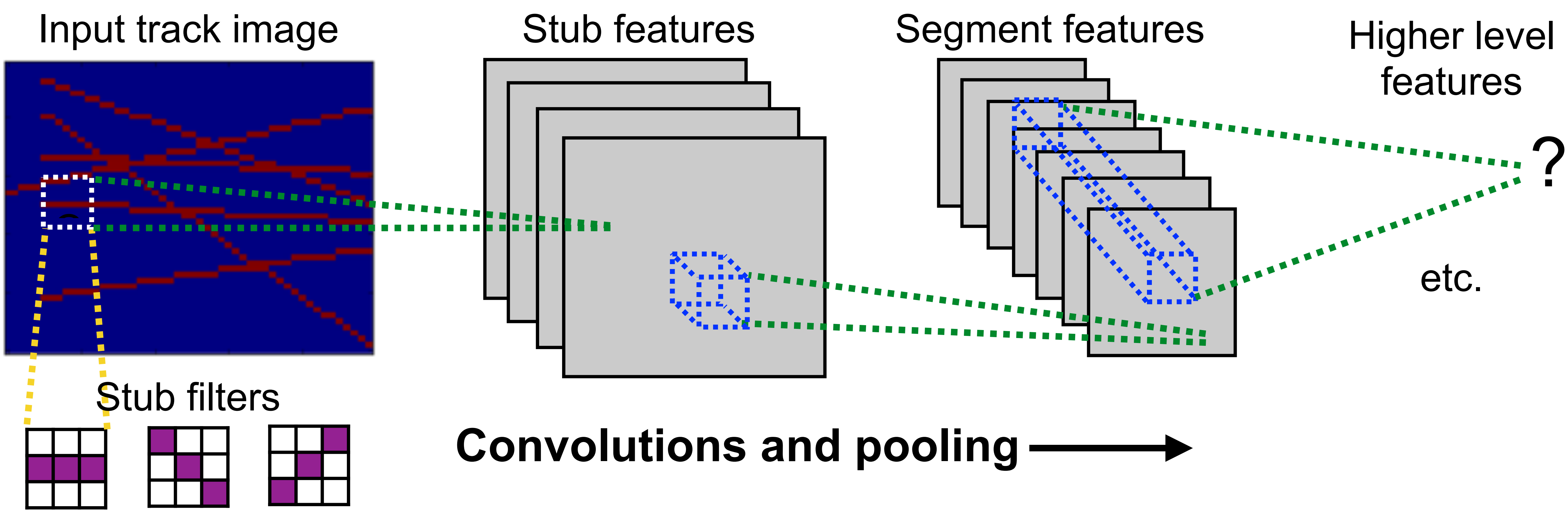
- **94.9% efficient**



Input
Target
Model prediction
Pixel bin
Layer
Input
Target
Model prediction
Pixel bin
Layer

Other HEP.TrkX stuff, some shown at CTD

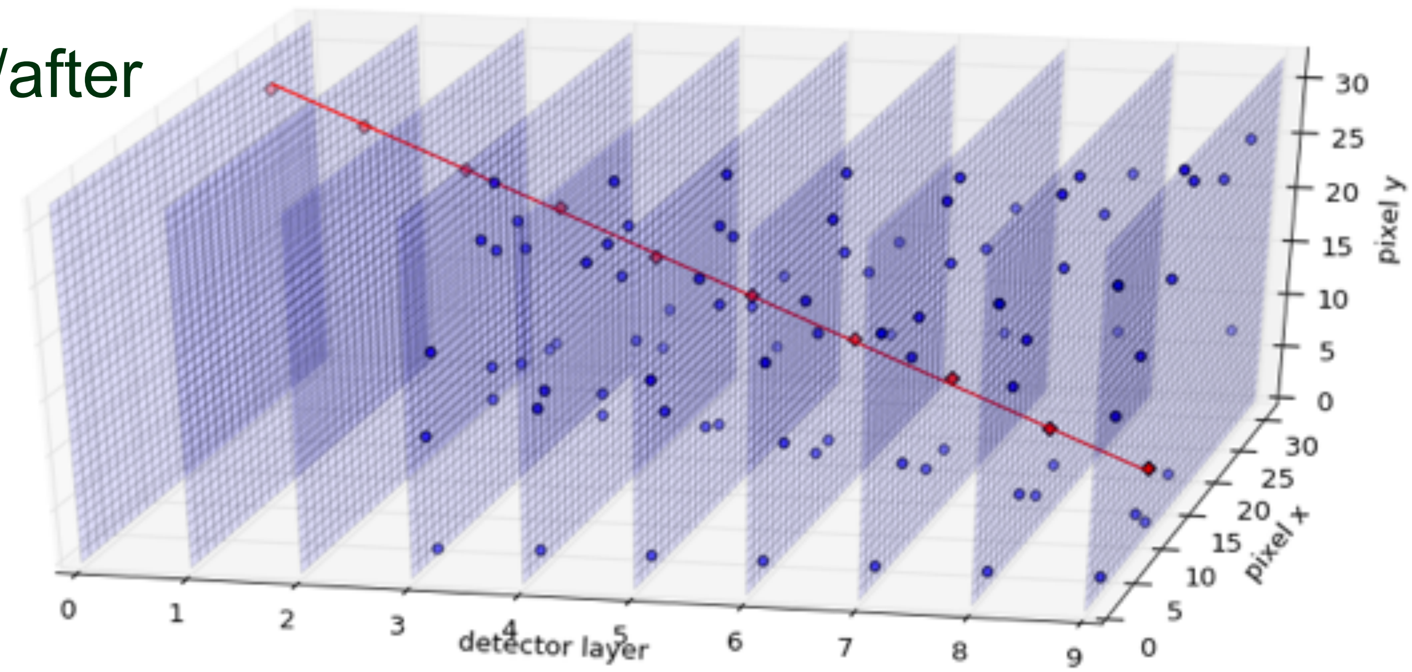
Convolutional networks as track finders



- **Convolutional filters can be thought of as track pattern matchers**
  - Early layers look for track stubs
  - Later layers connect stubs together to build tracks
  - Learned representations are in reality optimized for the data => may be abstract and more compact than brute force pattern bank
- **The learned features can be used in a variety of ways**
  - Extract out track parameters
  - Project back to detector image and classify hits

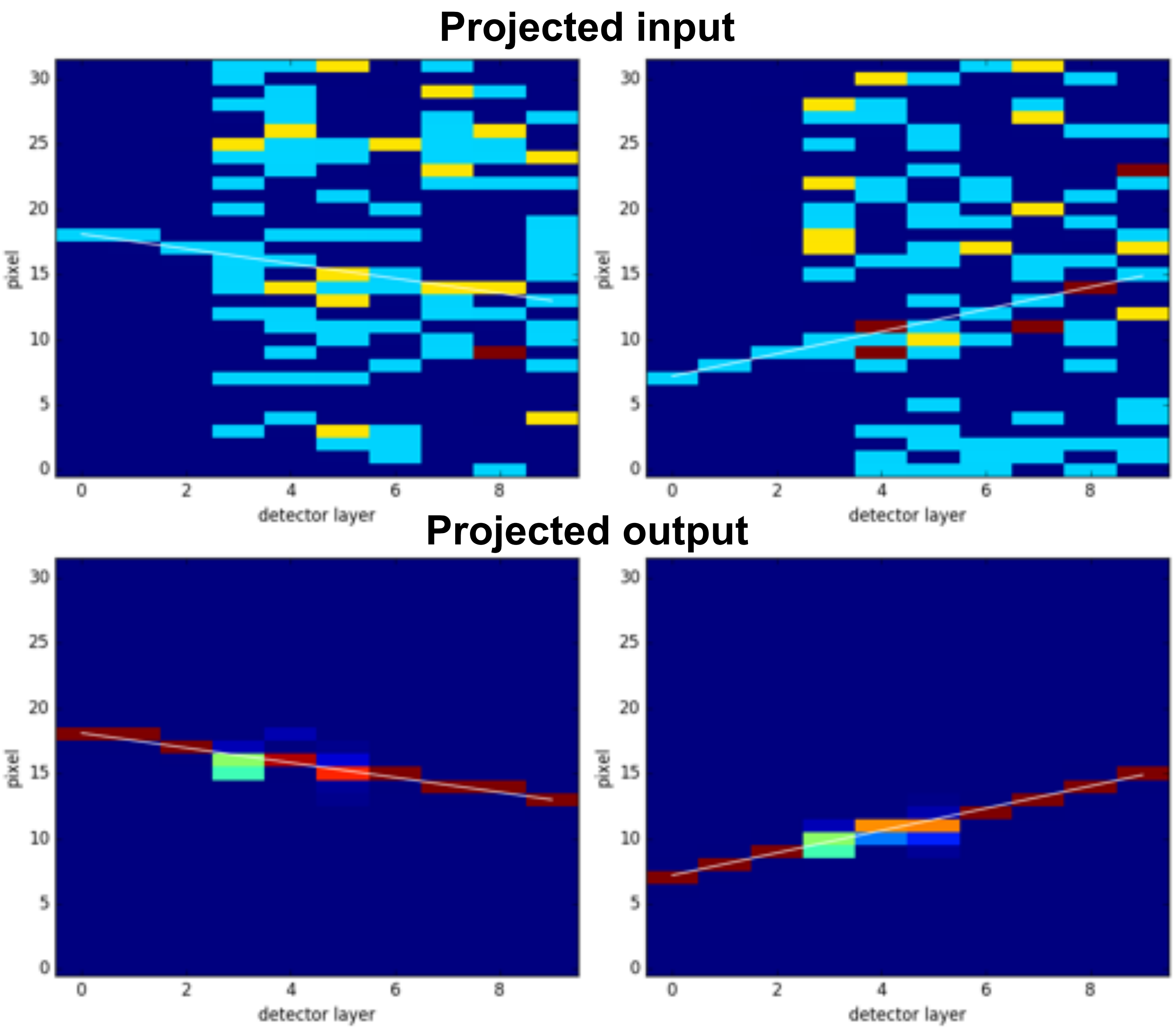
Testing models on 3D toy data

- Deeper LSTM model
  - Adds fully-connected layers before/after the LSTM
- Bi-directional LSTM
  - Adds a second LSTM running over sequence *in reverse*
  - Concatenate the two outputs

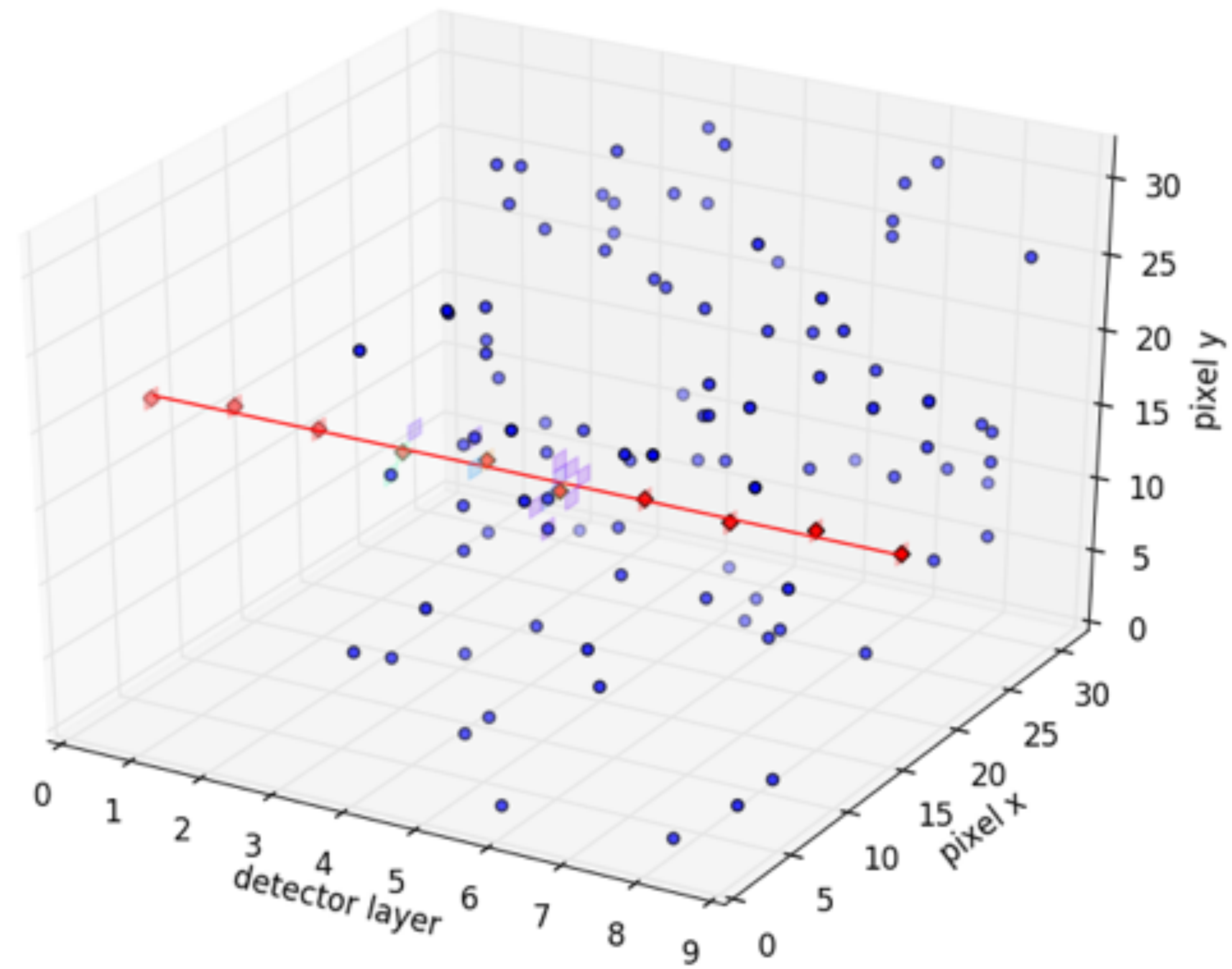


- *Next-layer* LSTM
  - Predict where the hit will be on the *next* detector plane, rather than the current detector plane
  - Basically just an extrapolator, but might be interesting to compare
- 3D convolutional model
  - 10 layers, no downsampling
- 3D conv autoencoder model
  - Uses max-pooling to downsample
  - Decodes with single fully connected layer

LSTM prediction

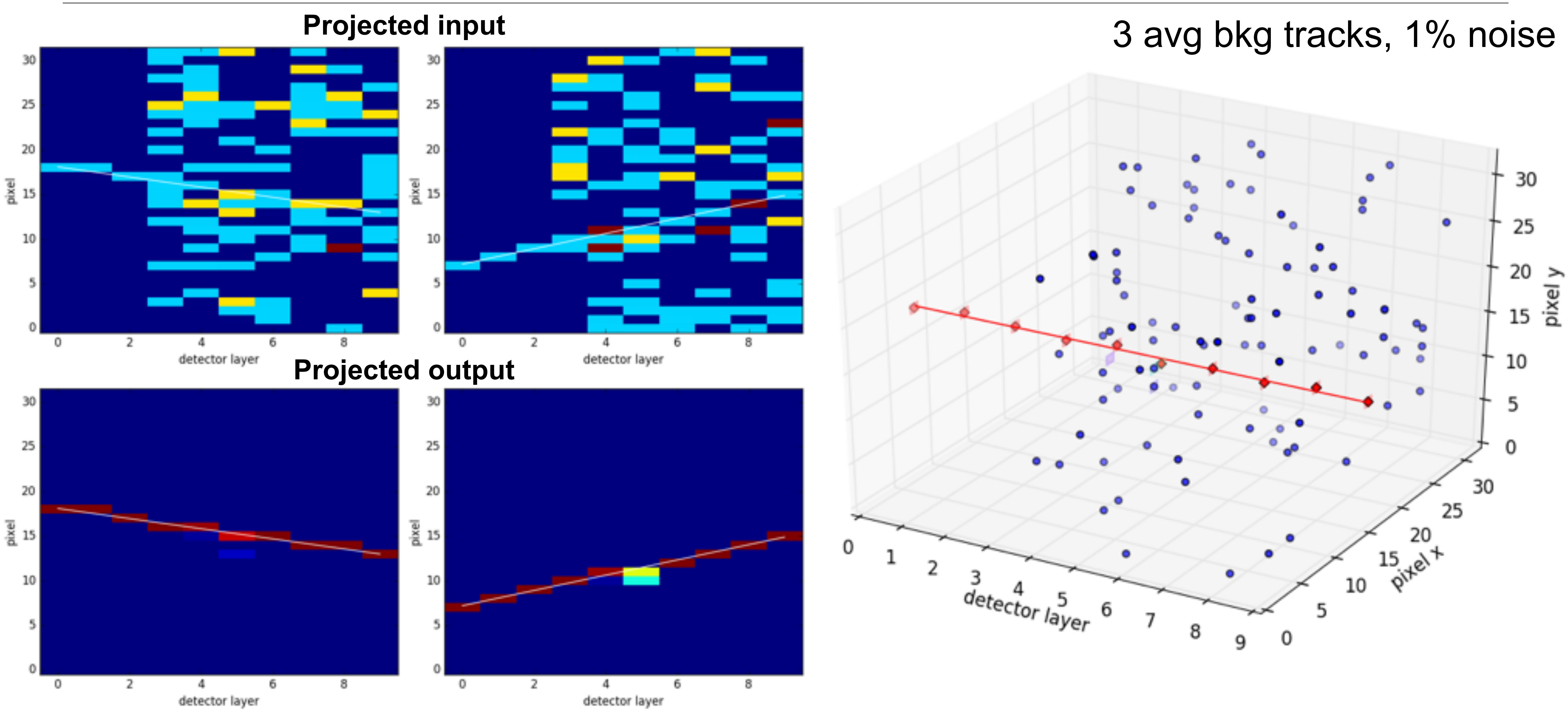


3 avg bkg tracks, 1% noise



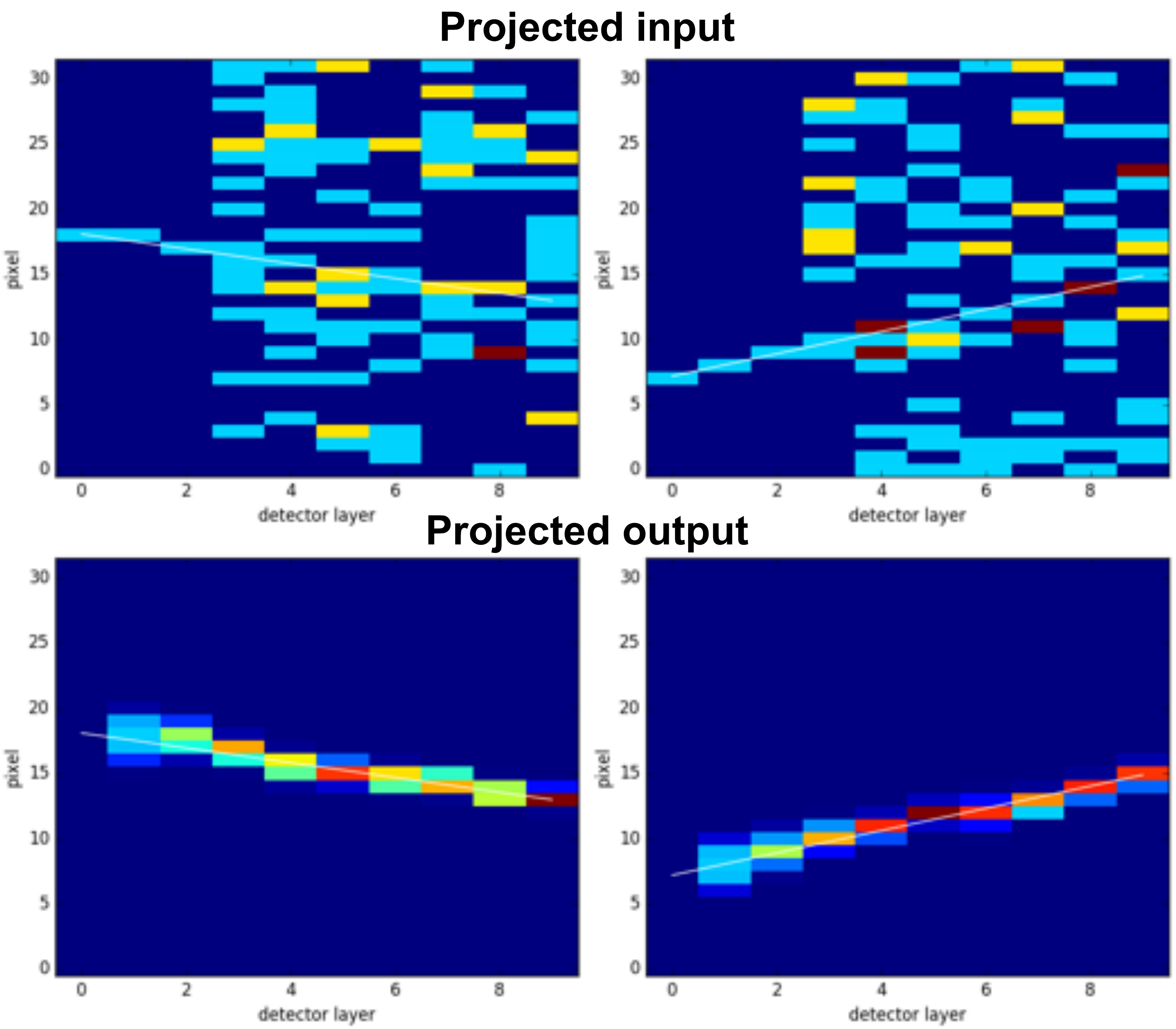
- Sometimes gives predictions that are not smooth
- Occasionally fooled by adjacent hits, though it tends to correct itself

Bidirectional LSTM prediction

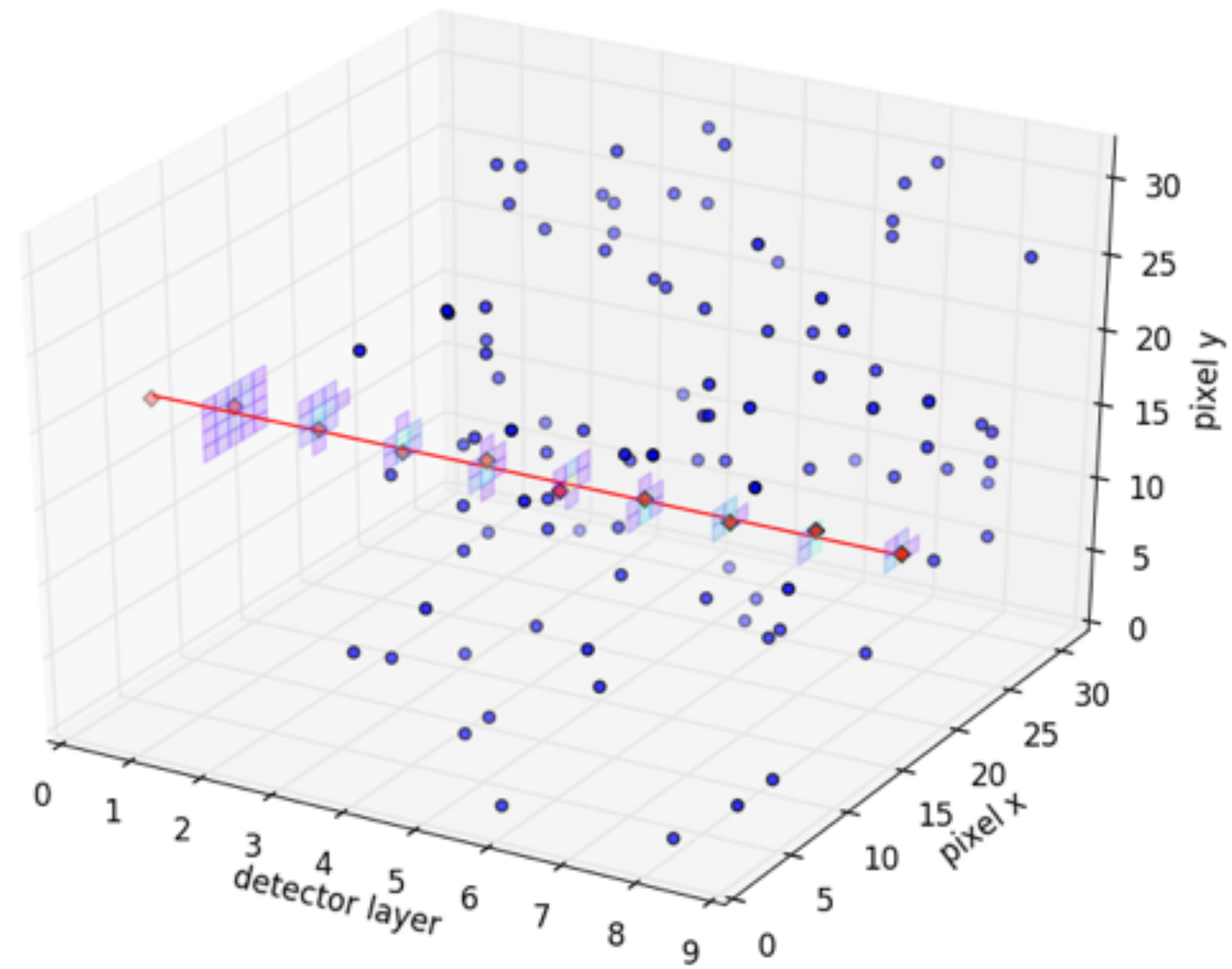


- Very precise predictions
  - can see into the future, which presumably helps
- still has few rare artifacts

Next-layer LSTM prediction

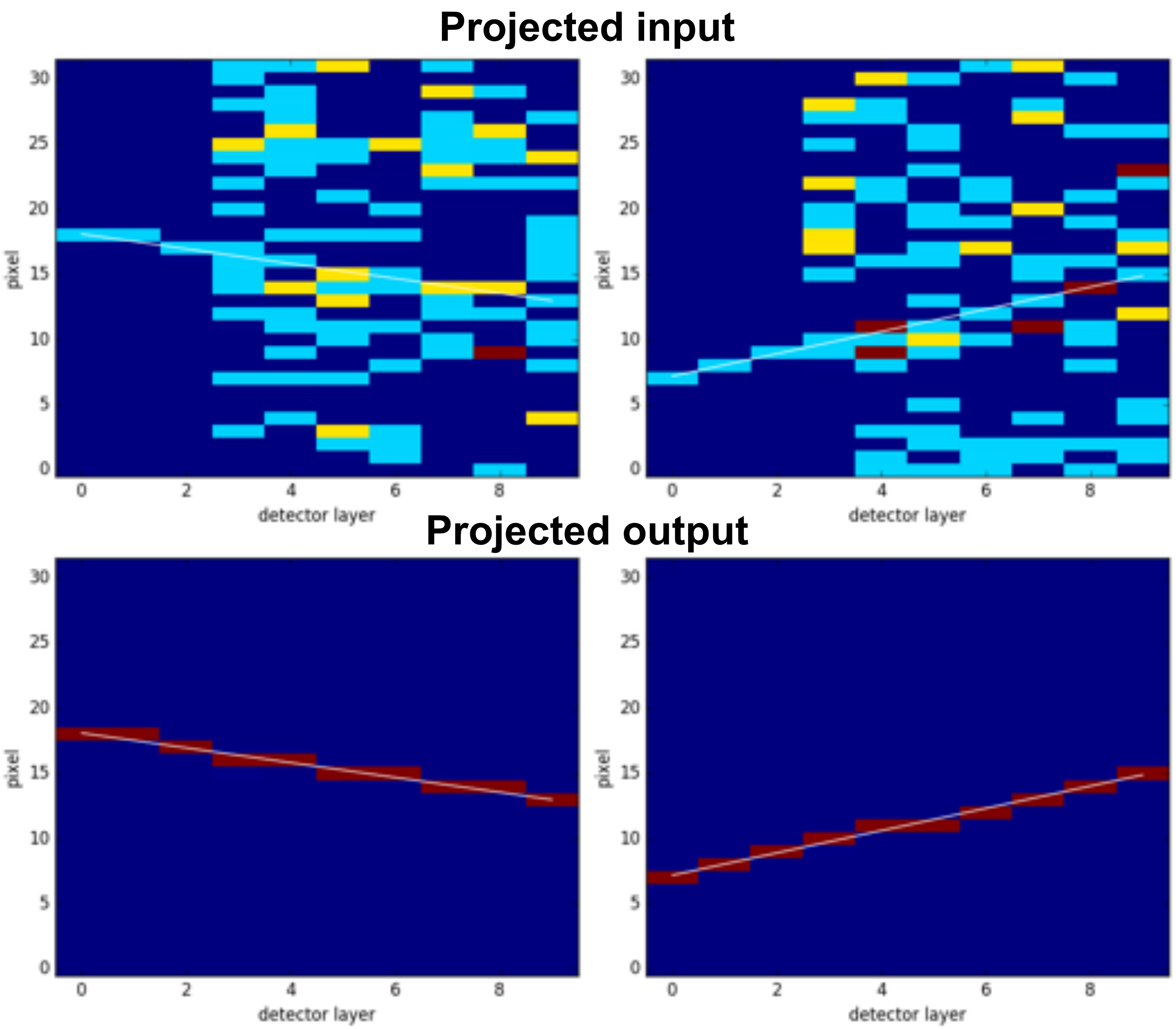


3 avg bkg tracks, 1% noise

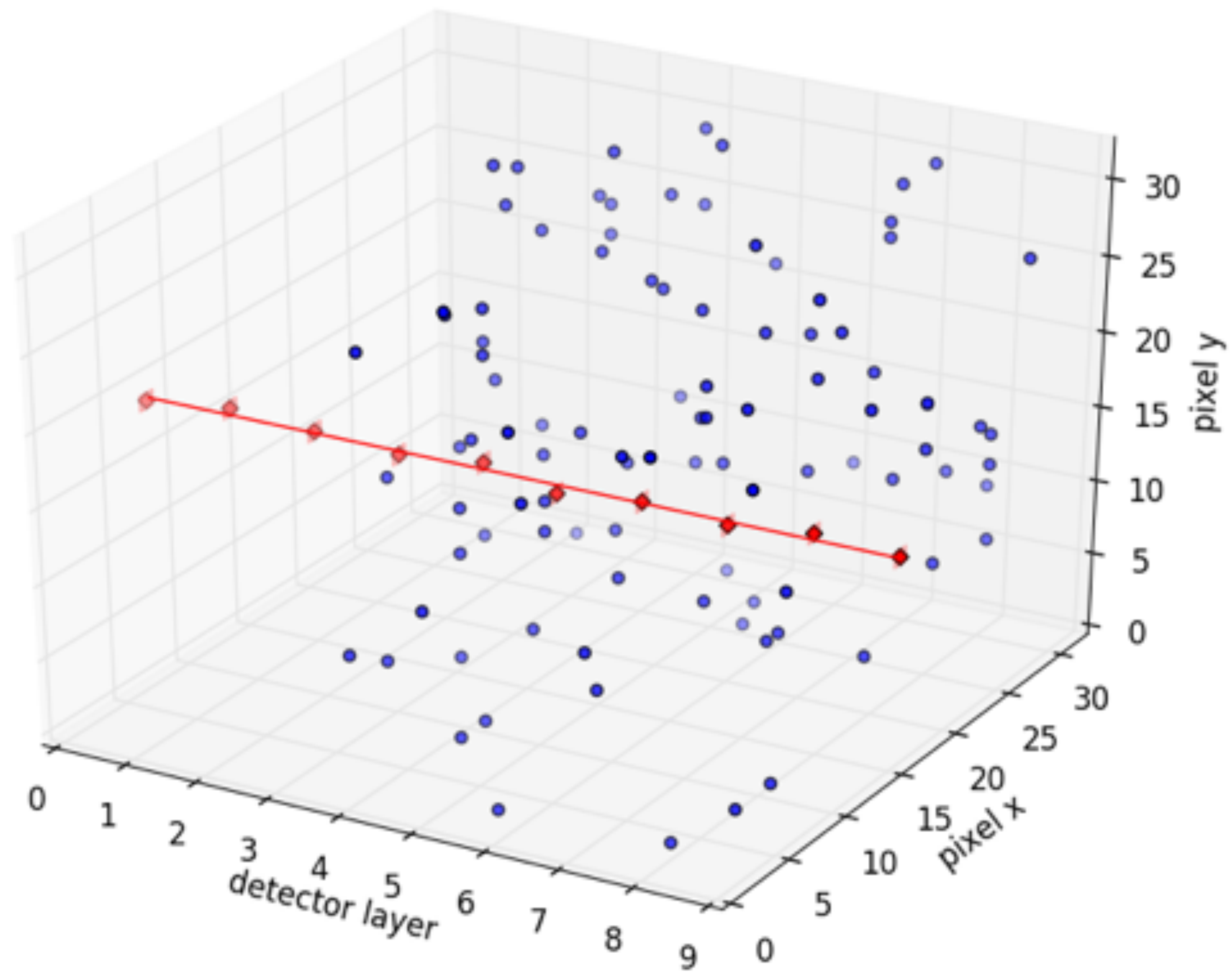


- Next-layer model gives predictions that are less precise but smoother and more accurate
  - Mostly unaffected by nearby stray hits
- With this detector occupancy, they are the best at classifying hits
  - but this may change with higher occupancy

ConvNN prediction

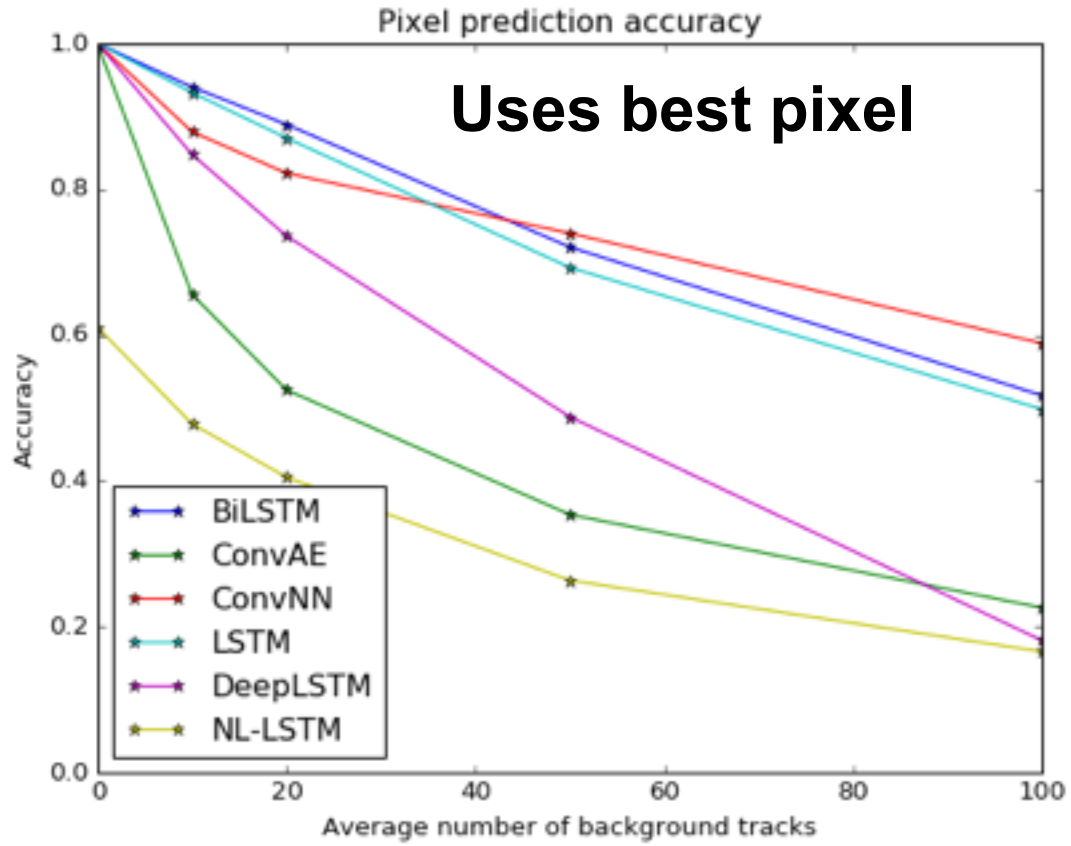


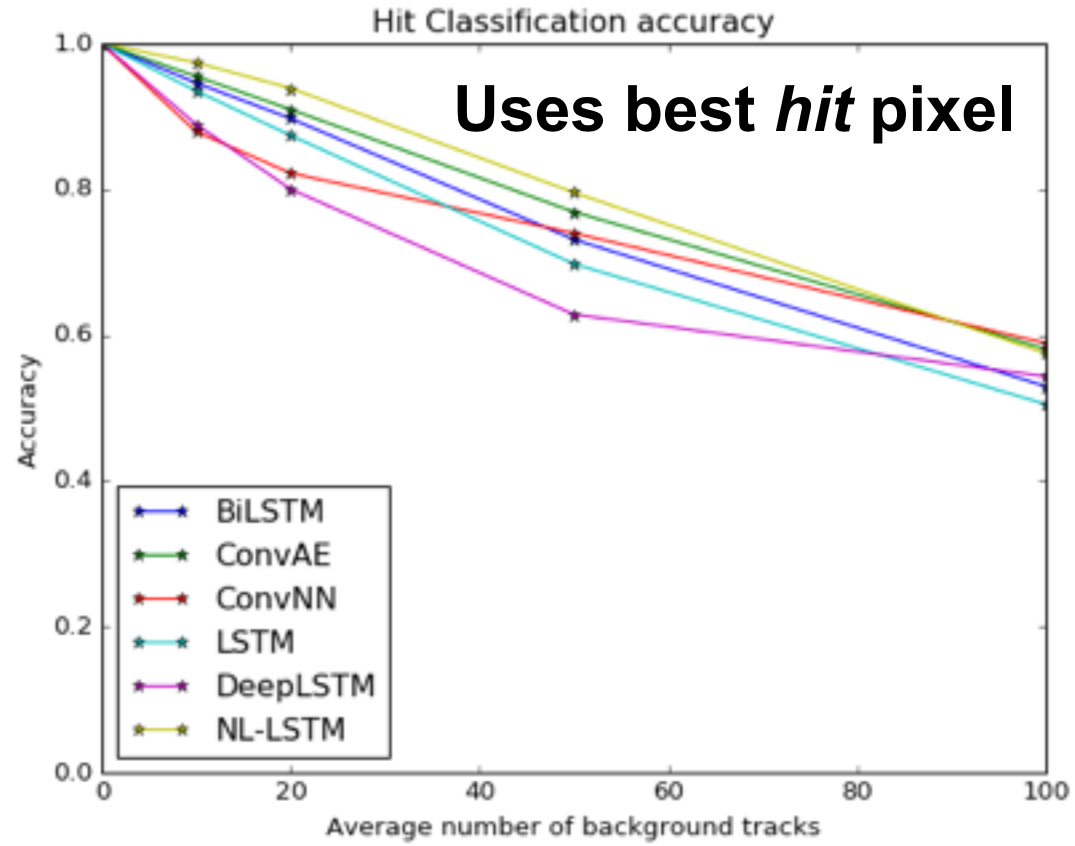
3 avg bkg tracks, 1% noise

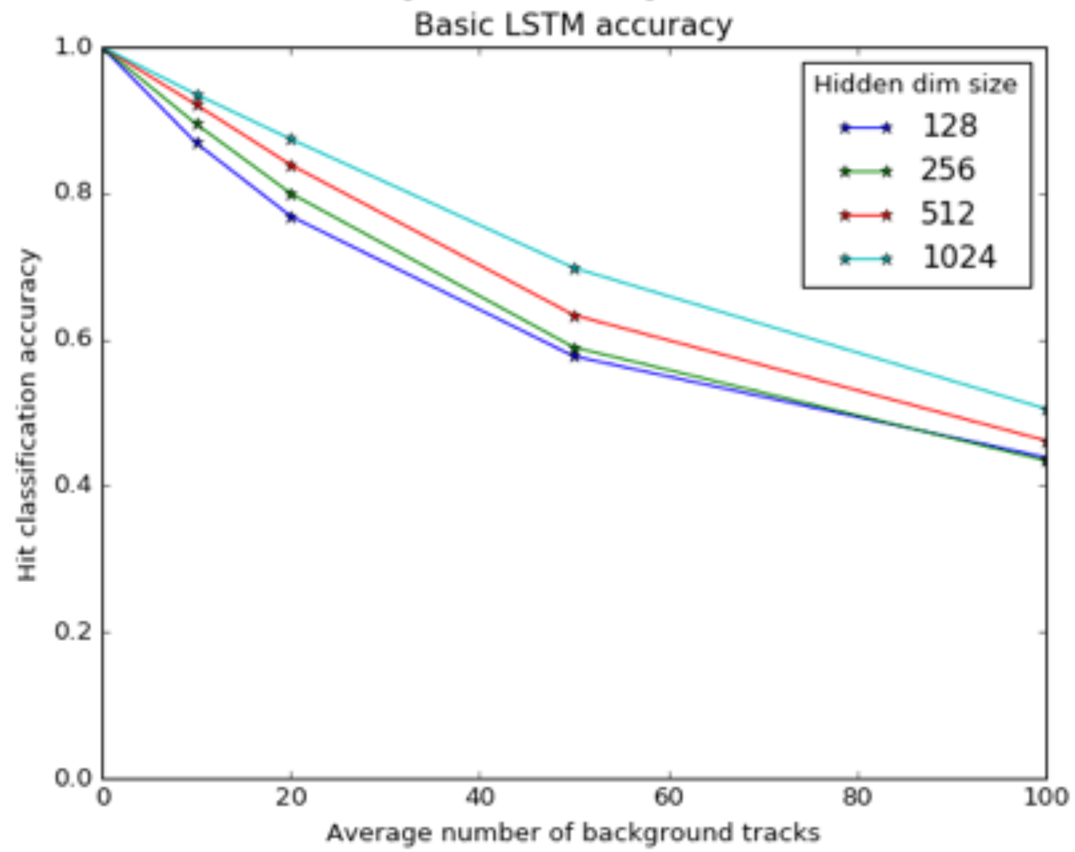


- Simple conv net is clean and precise in this case

Architecture comparisons





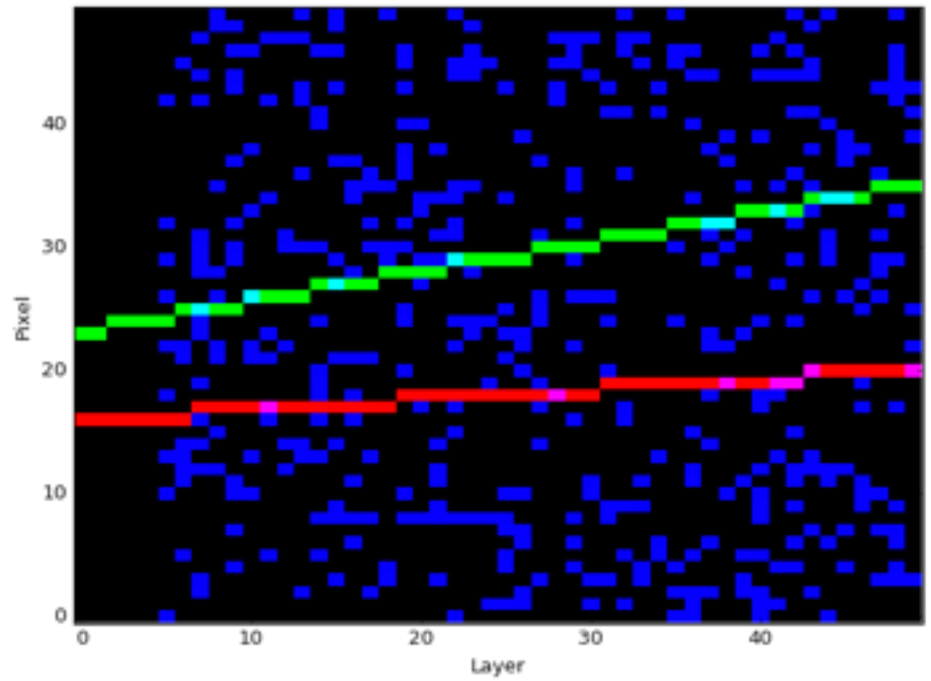


- Models' performance tanks with increasing track multiplicity
  - ConvNN scales the best
- Interesting tradeoffs between the architectures

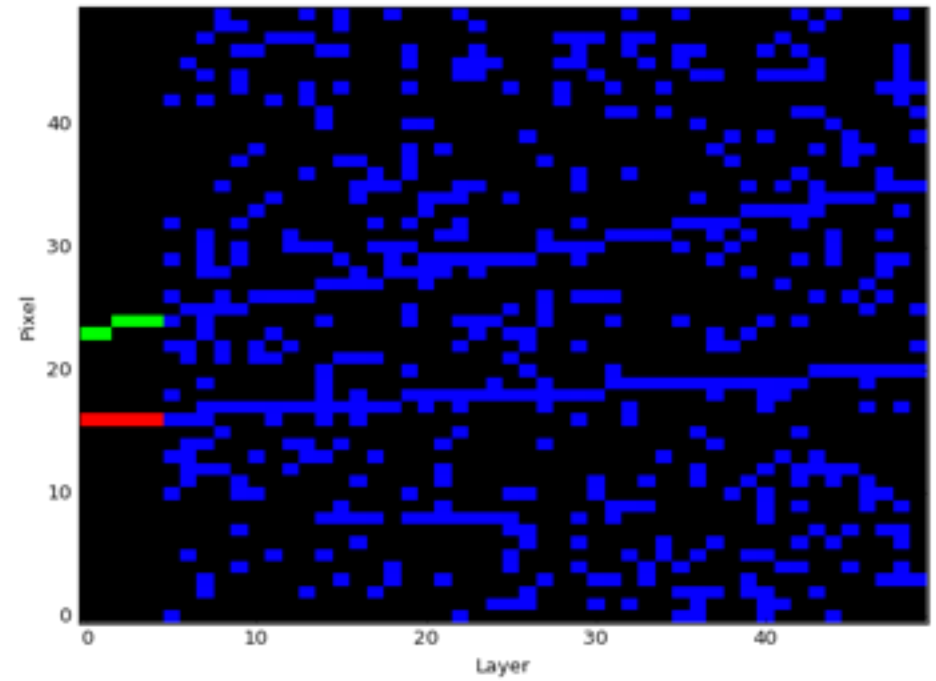
Towards multi-track tracking

- Attempt to extend model for multiple input seeds and multiple output tracks

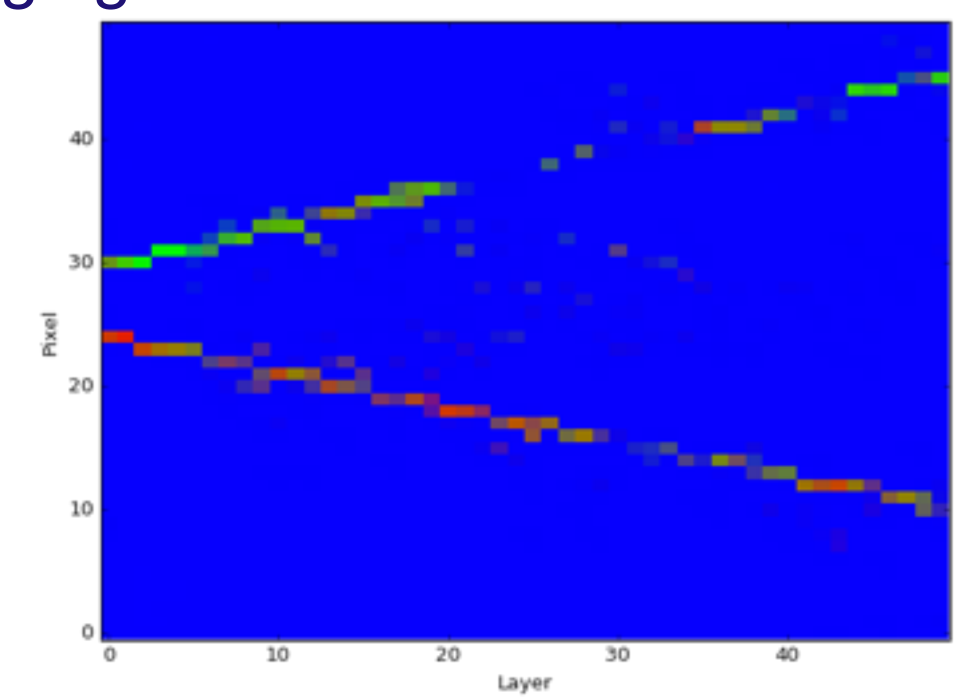
Multi-channel data



Specify seeds in model input

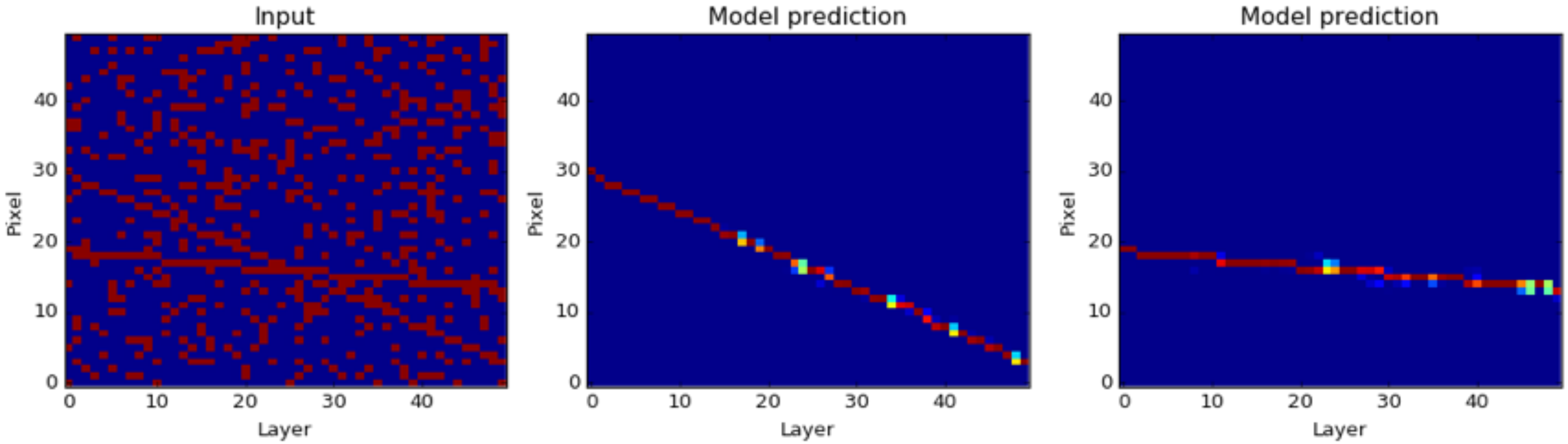


- Every pixel classified by LSTM as belonging to one of the track channels or the unassigned channel
  - Doesn’t train well, but kinda works

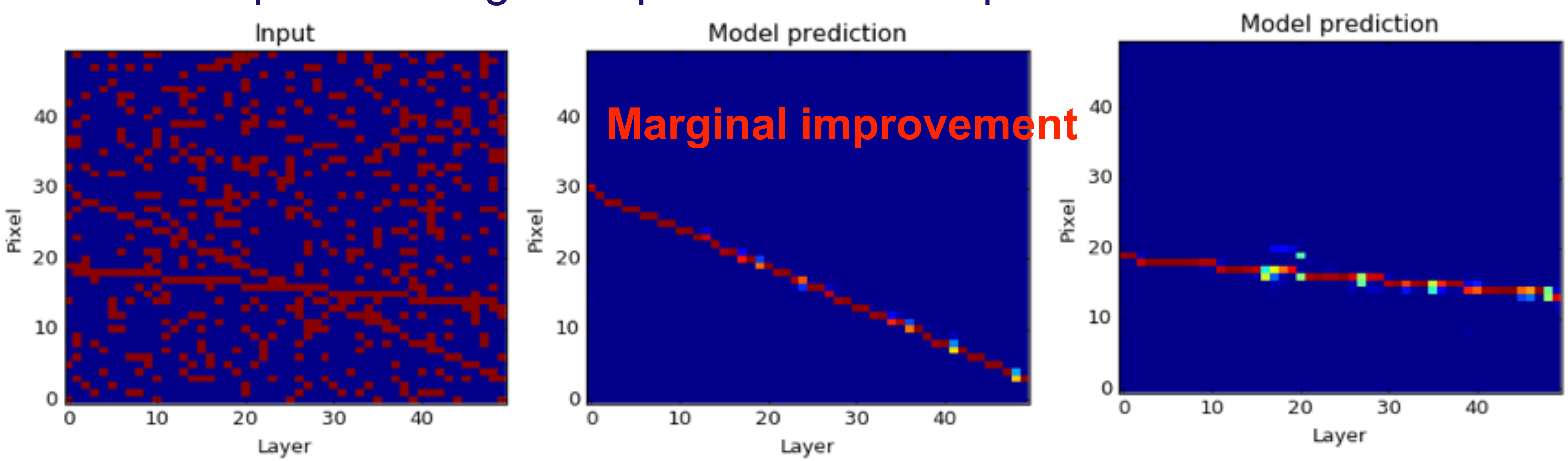


Towards multi-track tracking

- Calculate probability scores per-track, per-layer, as was done before

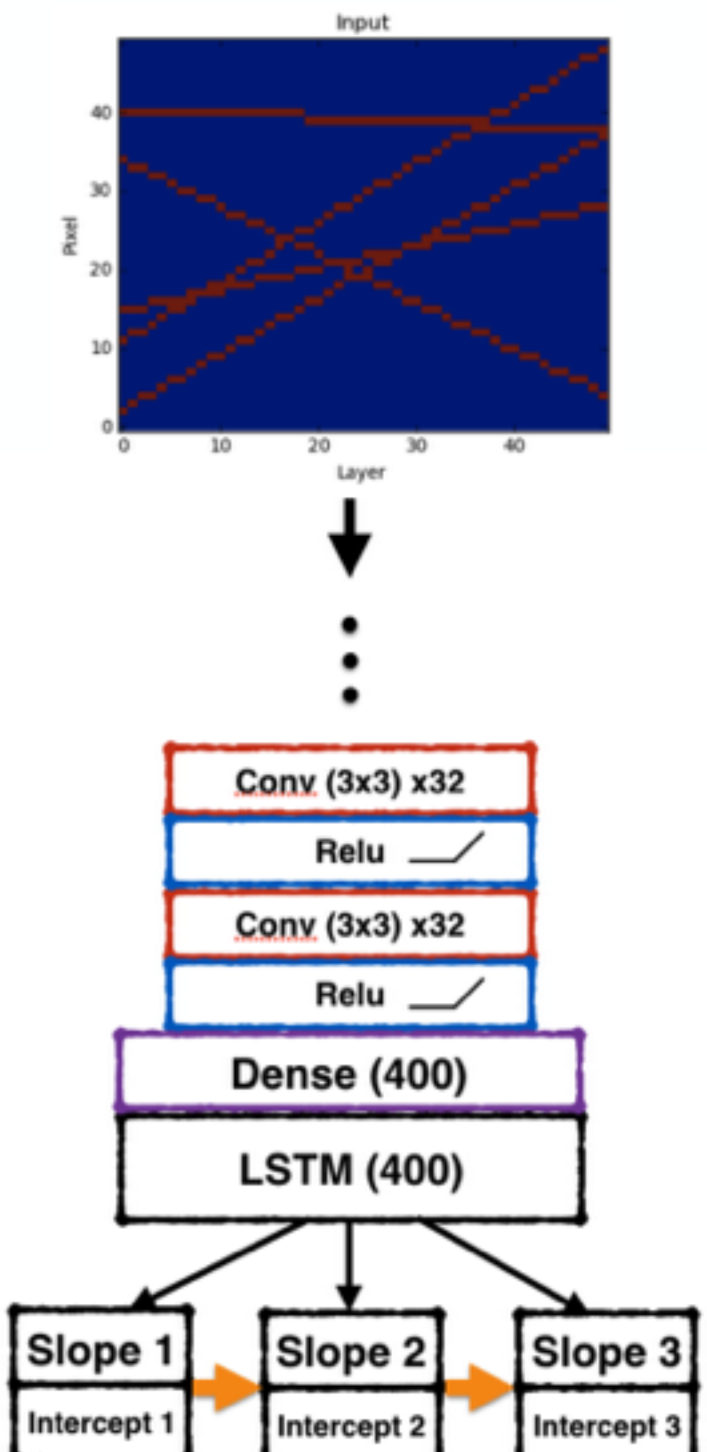


- Allow the LSTM to process the data multiple times, combining previous iteration’s output with original input to refine the prediction

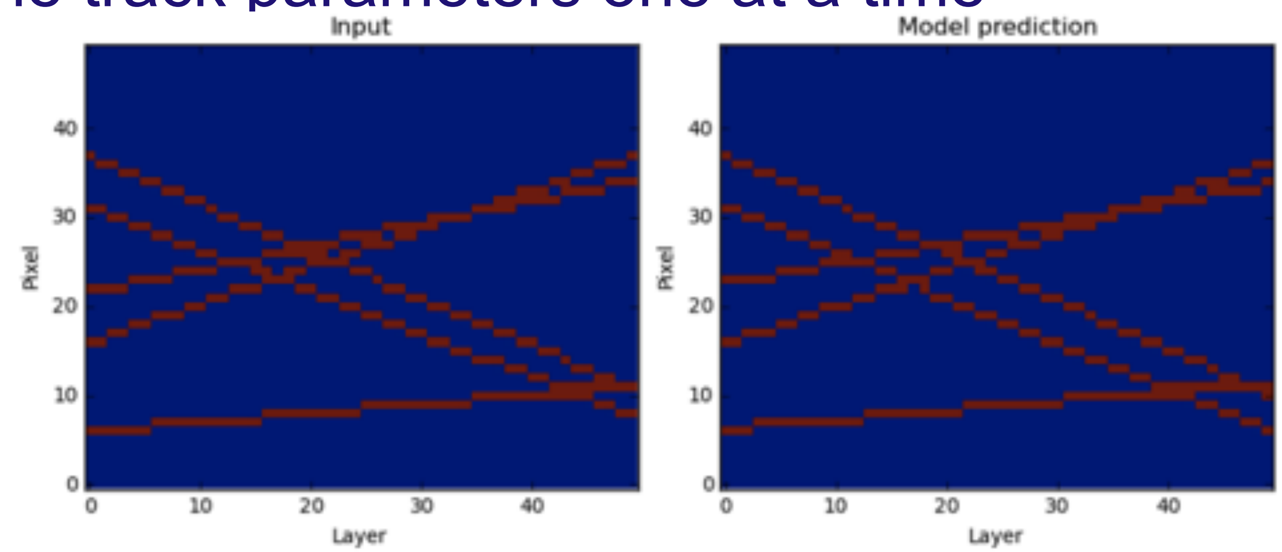


Marginal improvement

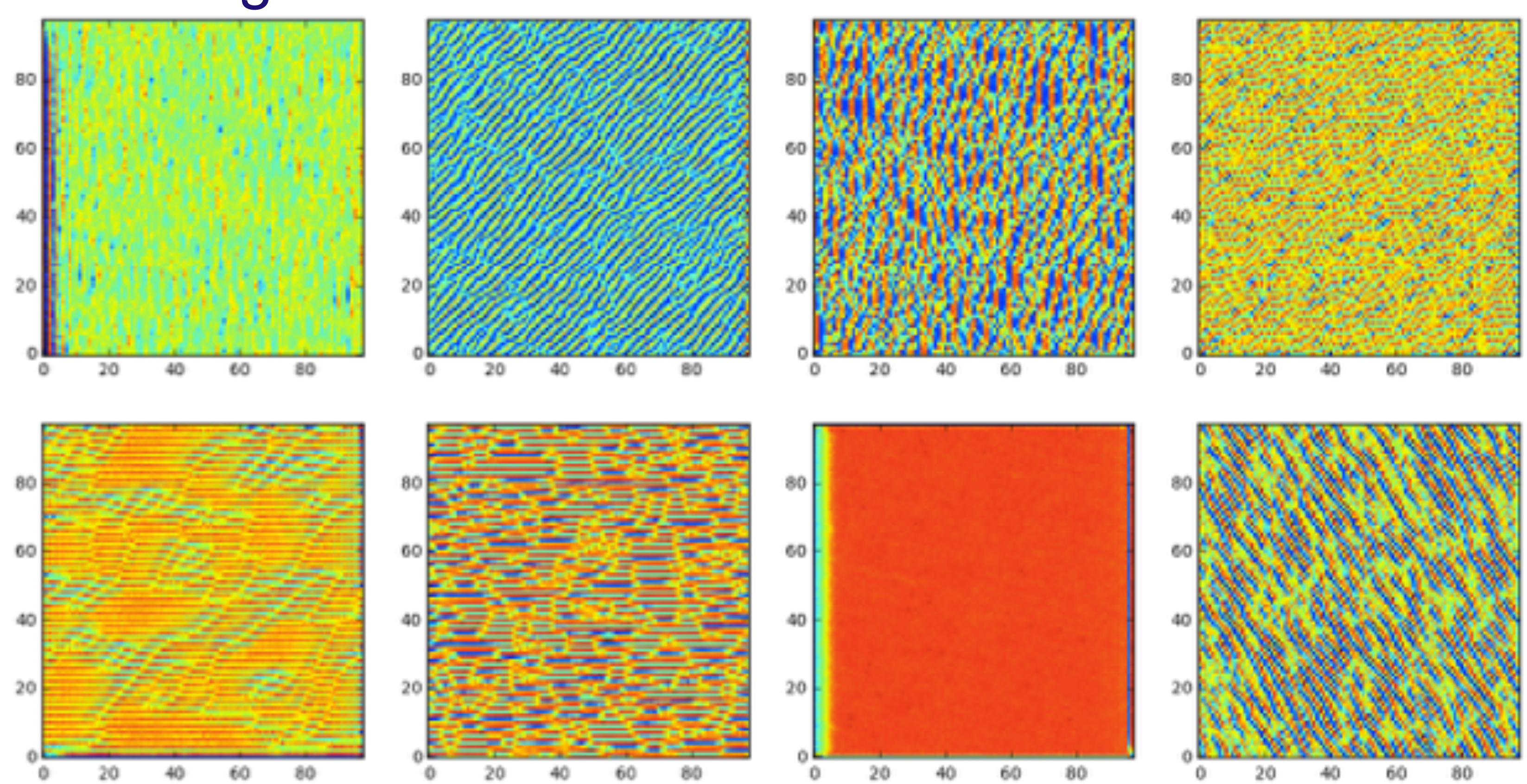
Track image captioning => hits to track params



- CNN extracts feature representation of detector image
- LSTM spits out the track parameters one at a time
- It actually works!

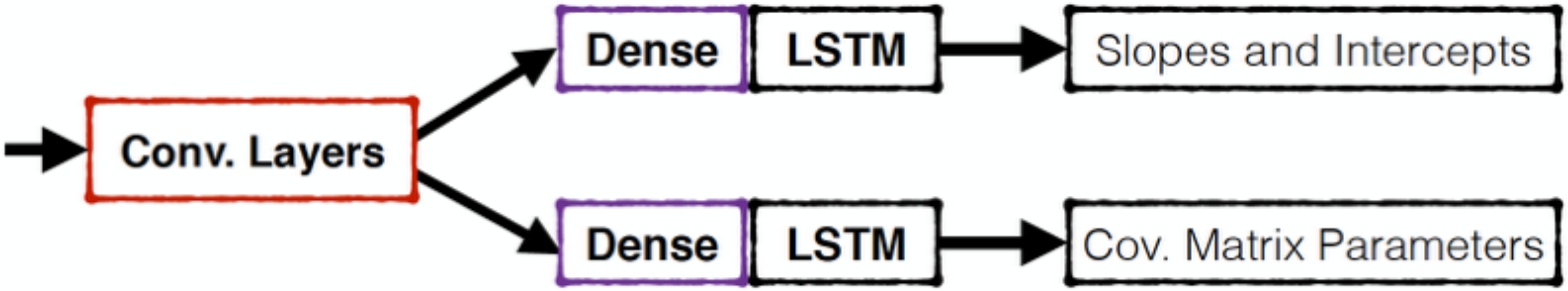


- Find images that maximize filters:



Estimating uncertainties on parameters

- In addition to the track parameters, we would need the covariance
- How do we extend the model to spit out reasonable uncertainties?
  - Add additional output to model for the covariance matrix:



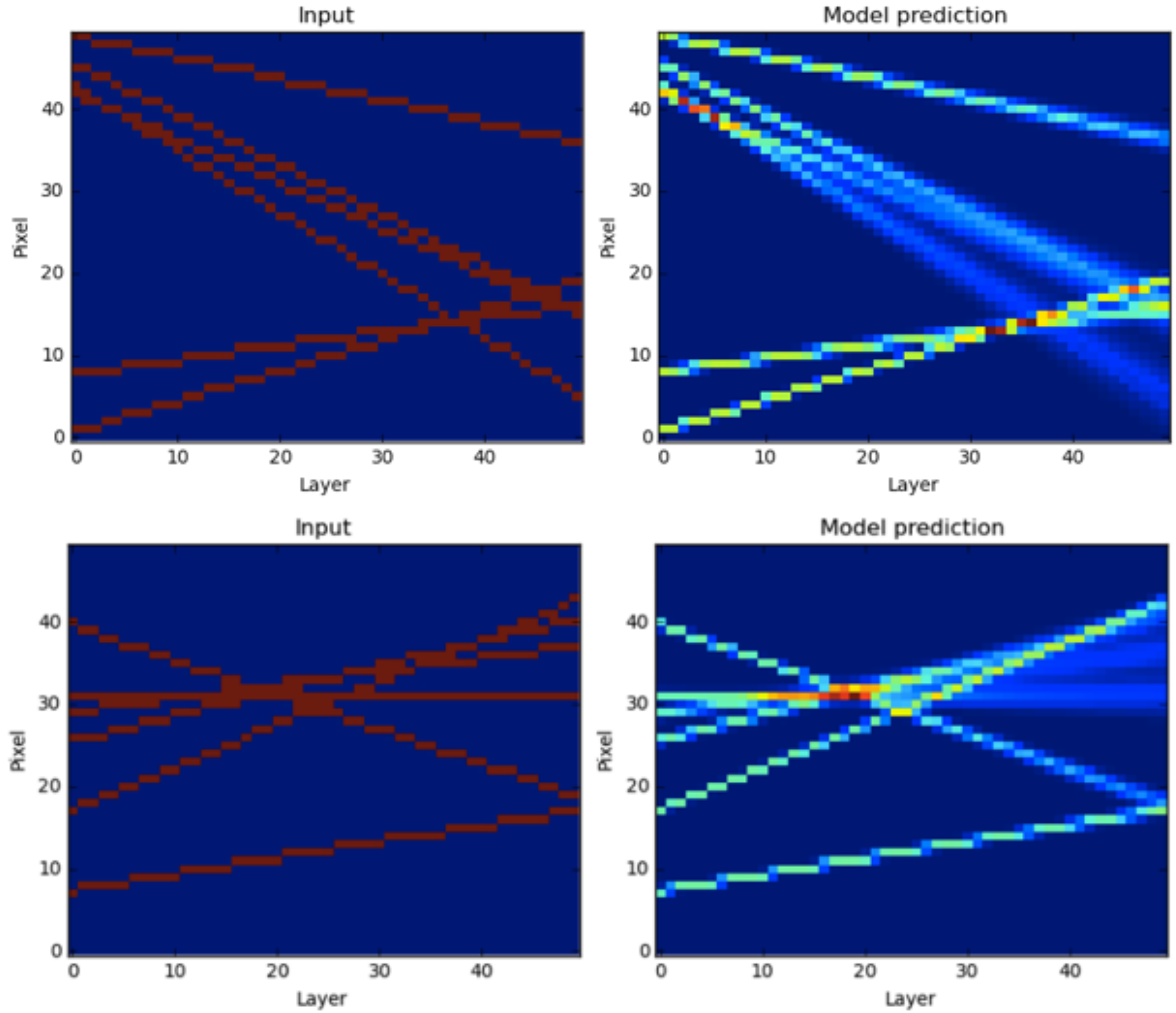
  - Replace mean-squared-error loss function with a log gaussian likelihood:

$$L(\boldsymbol{x}, \boldsymbol{y}) = \log|\boldsymbol{\Sigma}| + (\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x}))^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x}))$$

Minimize this during training

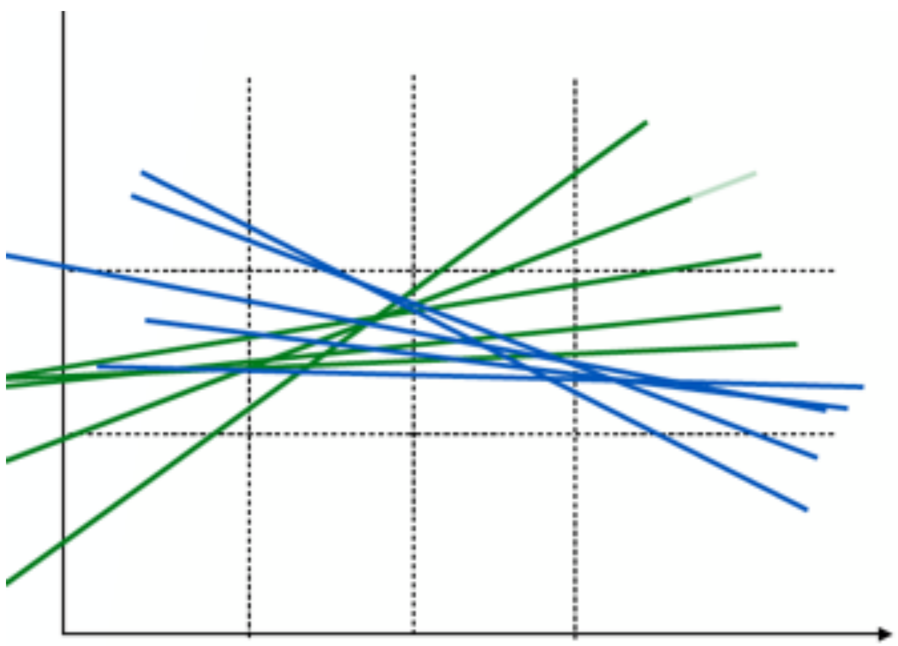
Visualizing predictions with uncertainty

- Drawn by sampling many times from the nominal predictions and uncertainties

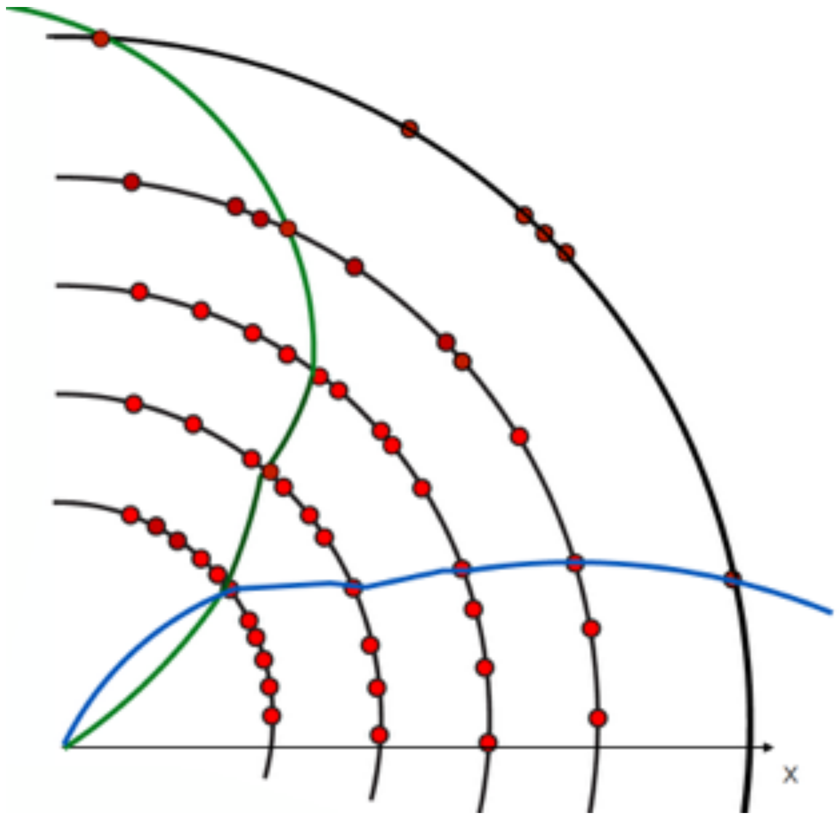


Other ideas - data transforms

- Hough Transform breaks down in LHC-like data due to process noise and high occupancy



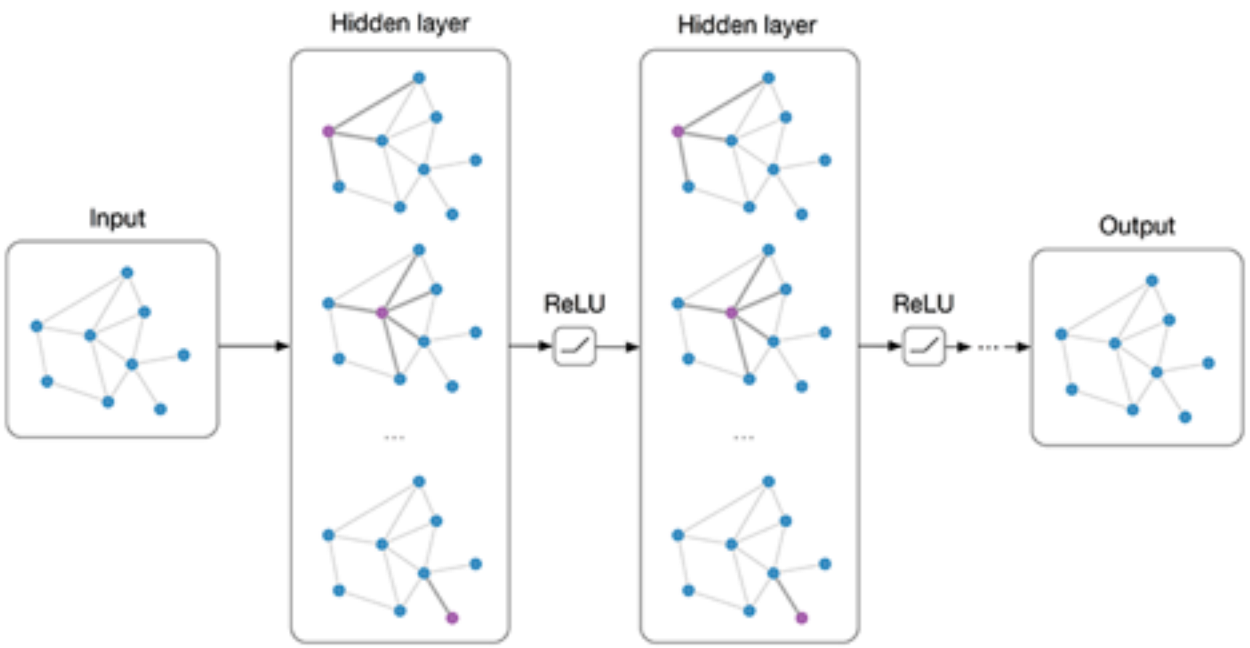
parameter space

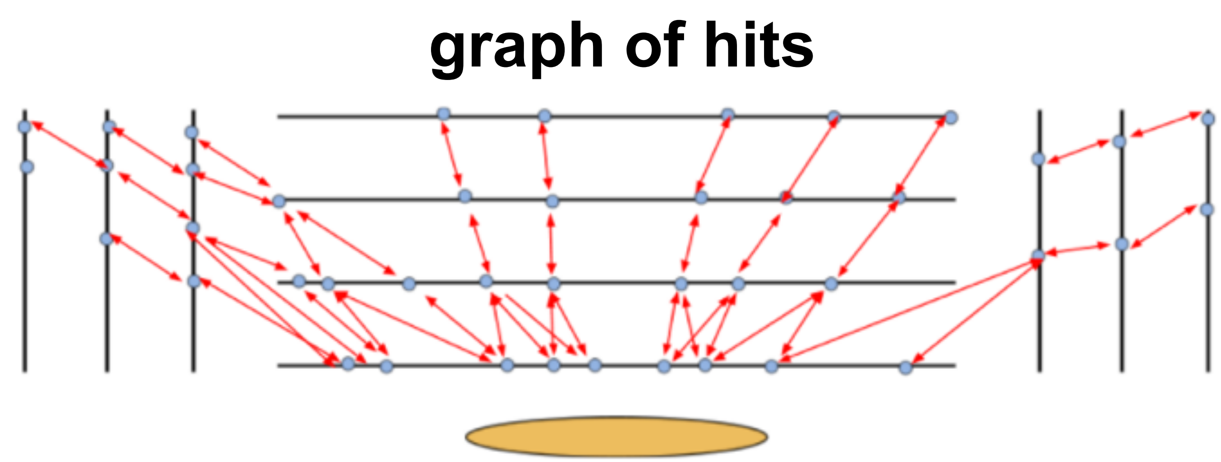


- But what if a deep network could *learn* a mapping to group together hits that belong to the same track?
  - You don't need to impose a specific representation
  - The model could take event context into account

Other ideas - graph convolutions

- Graph convolutions operate on graph-structured data, taking into account distance metrics
  - https://tkipf.github.io/graph-convolutional-networks/

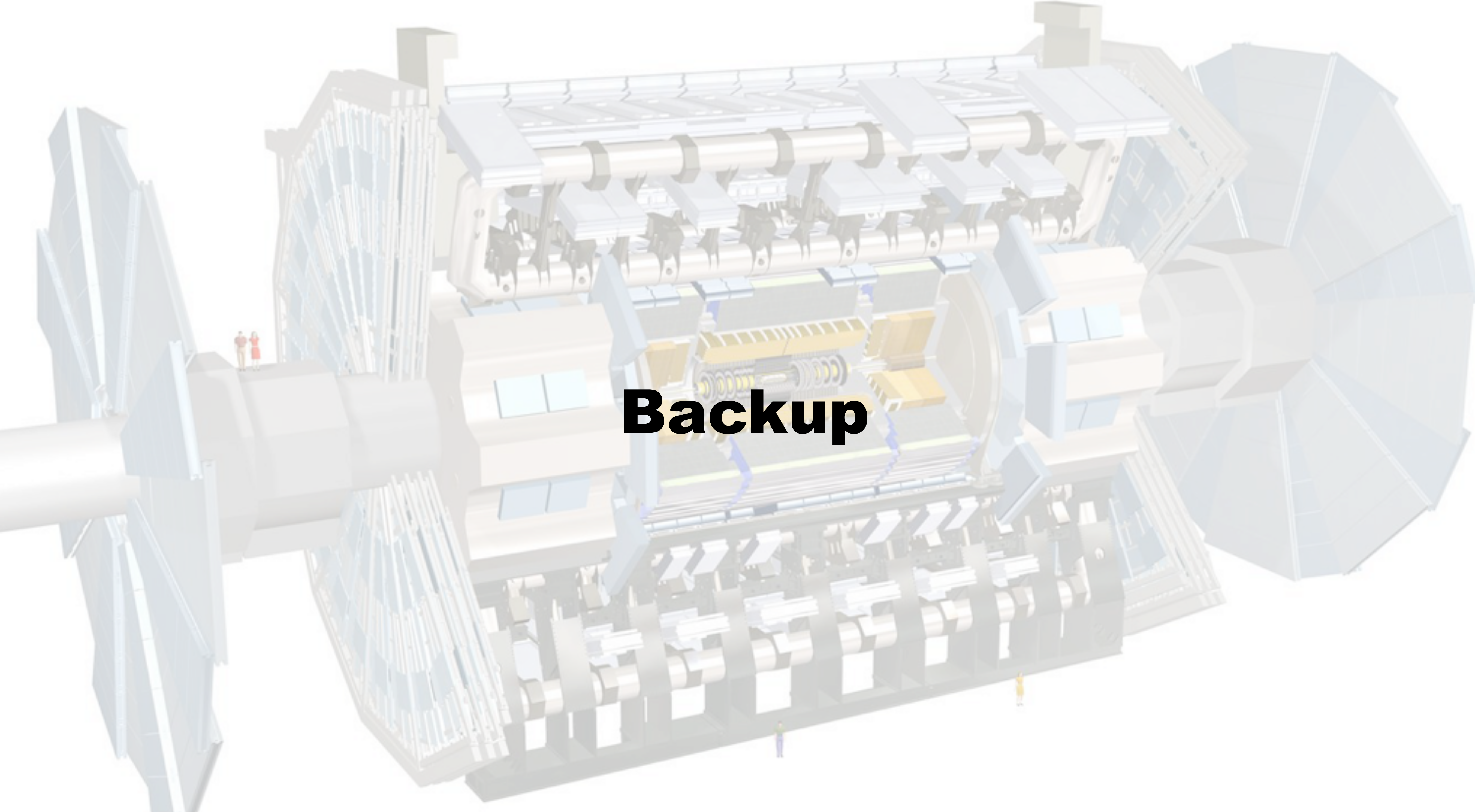




- Connections between ~plausible hits on detector layers can form the graph
  - Handles sparsity naturally
  - Scales naturally with occupancy
  - Handles irregular geometry layout (easily?)
- I haven't dedicated much thought to this yet, but it may be versatile enough to do the kinds of things I've already demonstrated

What's next for HEP.TrkX?

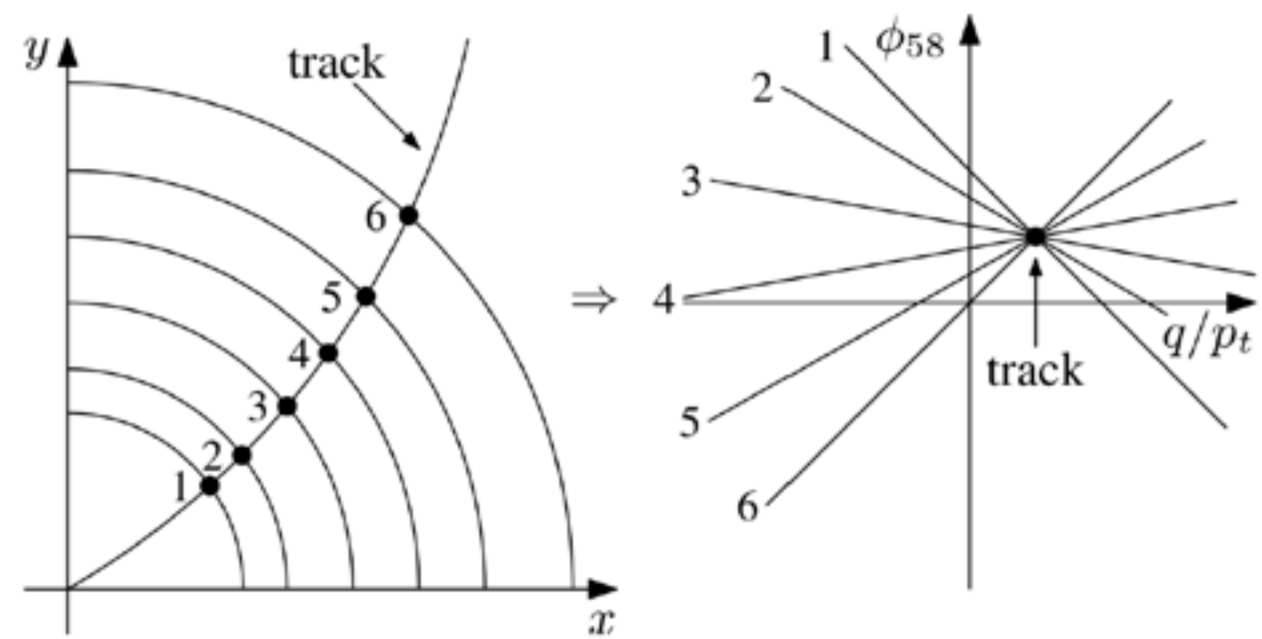
- We need to start focusing on 1-2 things that look promising, study them in depth, and compare to reasonable baselines
- Possibilities:
  - discrete detector track finding methods with realistic data, compare to existing KF
  - continuous hit space track finding (graphs?)
  - seeding => a slightly simpler problem..? Potential for high impact!
- Targets:
  - DS@HEP, FNAL, May 8-12
  - ACAT, Aug 21-25
  - ML conferences



Backup

Hough Transform
Algorithm

1. Calculate ϕ_{58} (ϕ at r_{58}) for each $^{q}/_{p_T}$.
2. Fill the stub into appropriate cells in a 32x64 array in $^{q}/_{p_T}$ x ϕ_{58}
3. Ignore $^{q}/_{p_T}$ values inconsistent with a stub's bend information (rough p_T estimate).
4. Define cells with stubs in at least 4 or 5 layers as track candidates.
   i. 4 layer threshold used to cope with barrel-endcap transition region or dead layers



Algorithm's simplicity → good for FPGAs

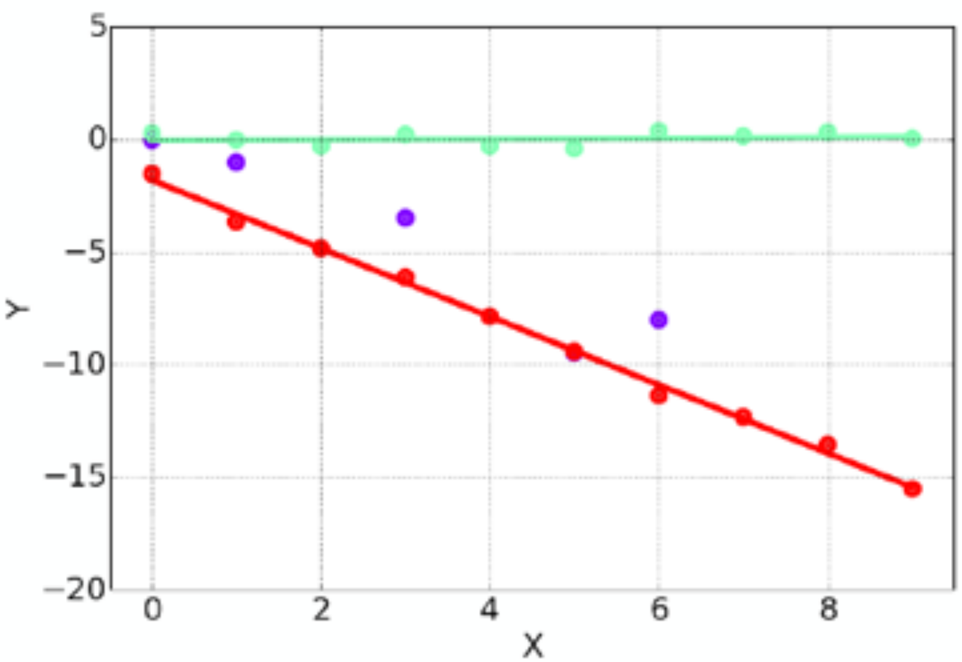
Artificial Retina

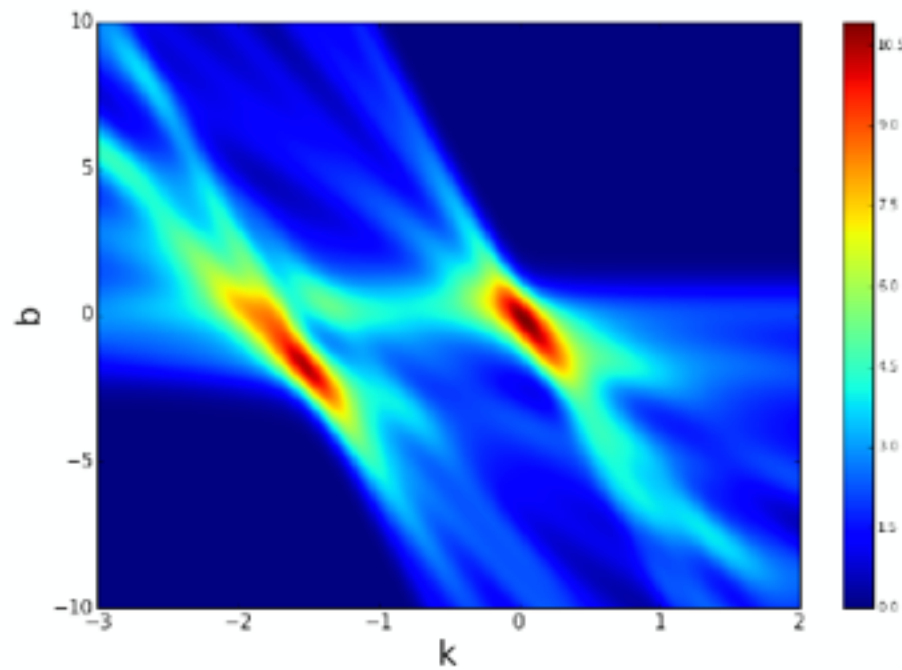
The artificial retina function is defined as:

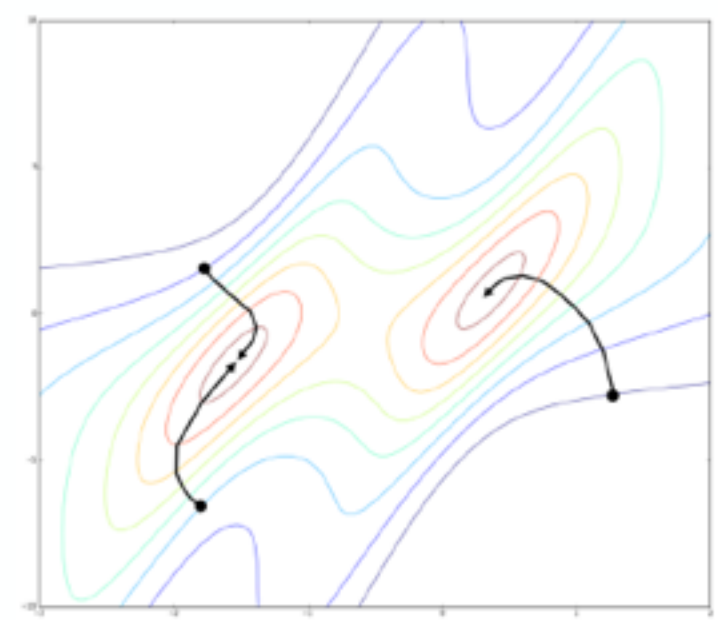
$$R(\theta) = \sum_i e^{-\frac{\rho^2(\theta, x_i)}{\sigma^2}}$$

where $\rho(\theta, x_i)$ is distance between the i-th hit and a track with parameters θ .

For 2D tracks: $\rho(\theta, x_i) = y_i - (kx_i + b)$ $\theta = [k, b]$

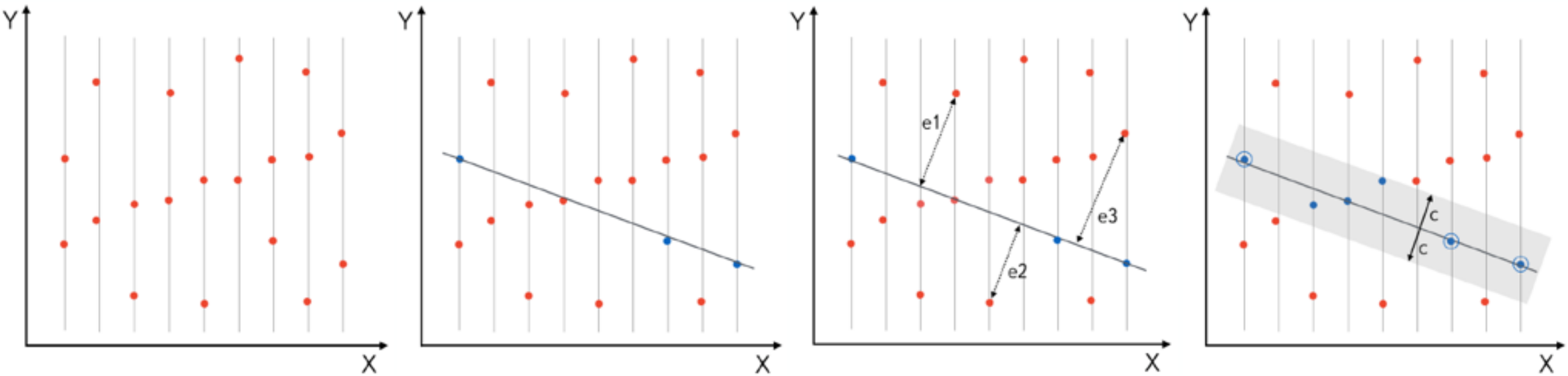






7

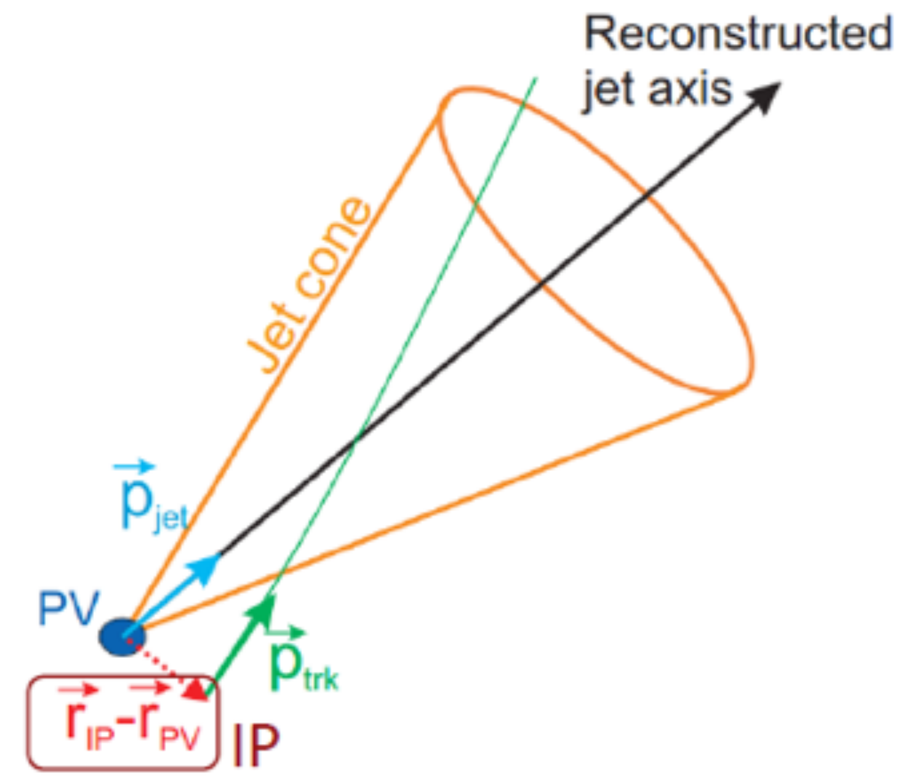
RANSAC



Searching for one track:
1. The RANSAC selects a random subset of the hits.
2. The linear model is fitted using this subset.
3. The error of the data with respect to the fitted model is calculated.
4. The number of inlier candidates is calculated.
5. Steps 1-4 are repeated until the maximum number of iterations.
6. A model with maximum number of inliers is returned.

5

- High level b-tagging algorithm (MV2c10) @ ATLAS combines baseline taggers utilizing information of different aspects of b-hadron decay



- IP3D

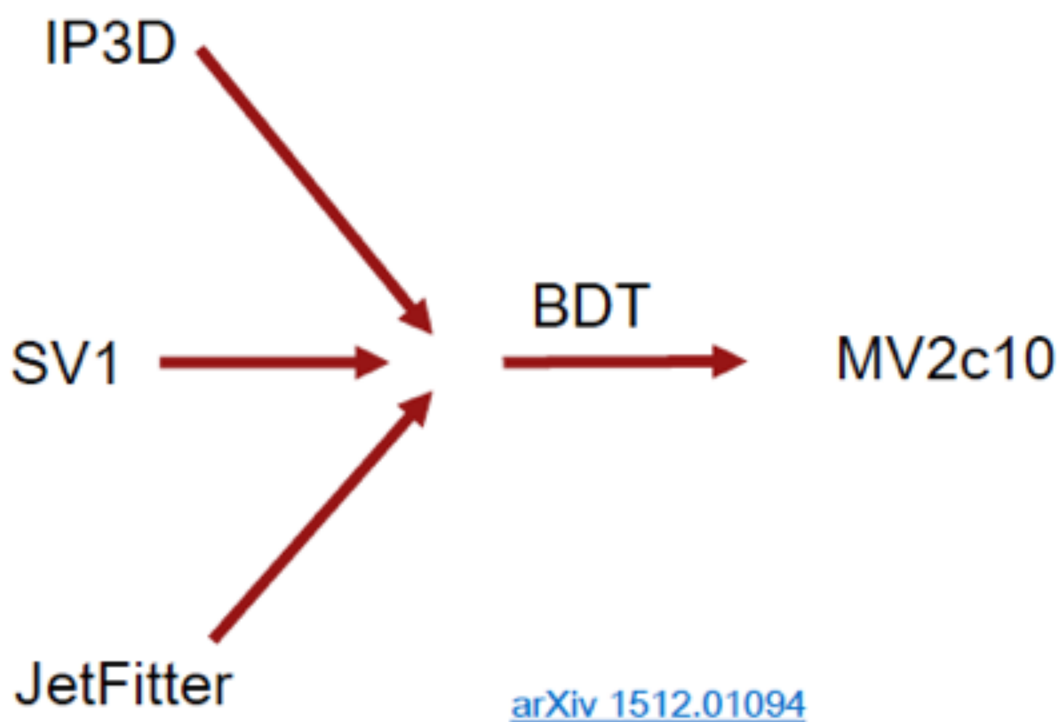
  likelihood-ratio based on the signed transverse and longitudinal impact parameters of tracks associated to jets

- SV1

  Fit secondary vertices with full track covariance matrix; Utilize secondary vertex information, e.g. vertex mass, ratio of vertex energy, number of two track vertices, etc. for tagging

- JetFitter

  A Kalman filter which finds common flight path of b and c hadrons



arXiv 1512.01094