Convolutional Neural Networks for Particle Tracking

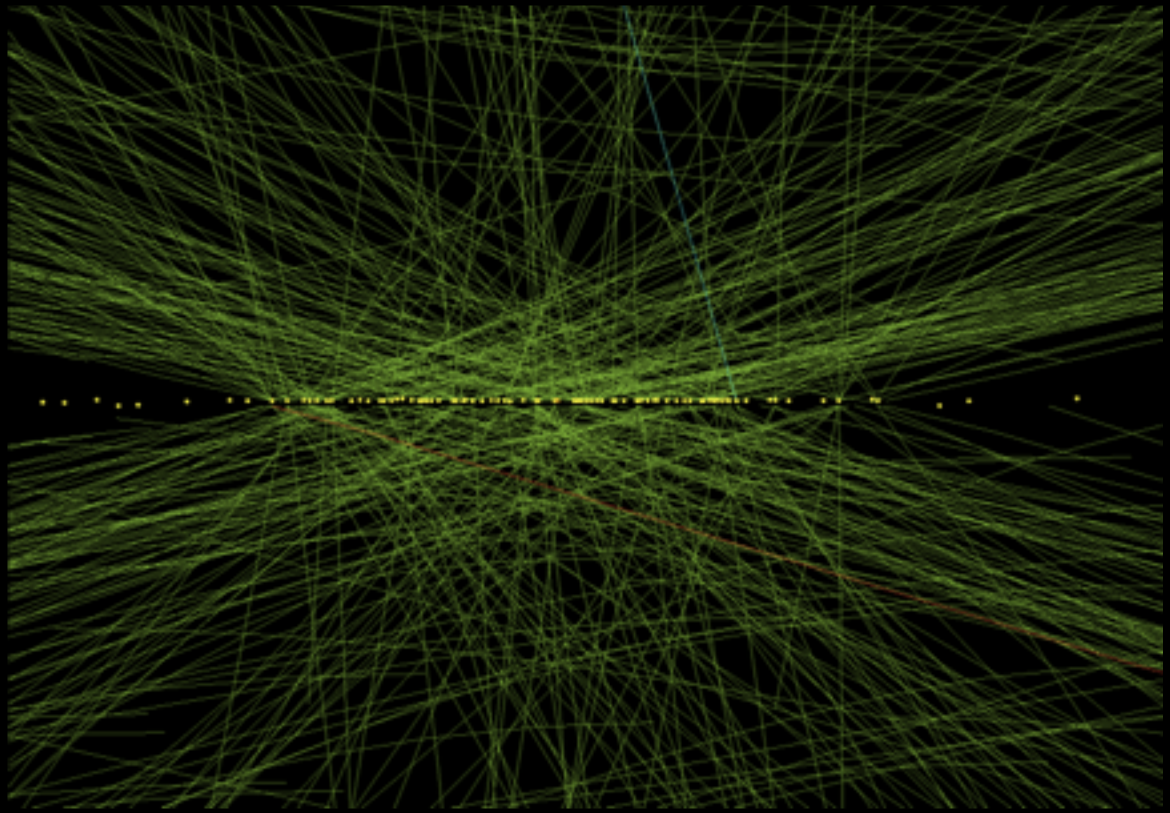
Steve Farrell
for the HEP.TrkX project

May 8, 2017
DS@HEP, FNAL

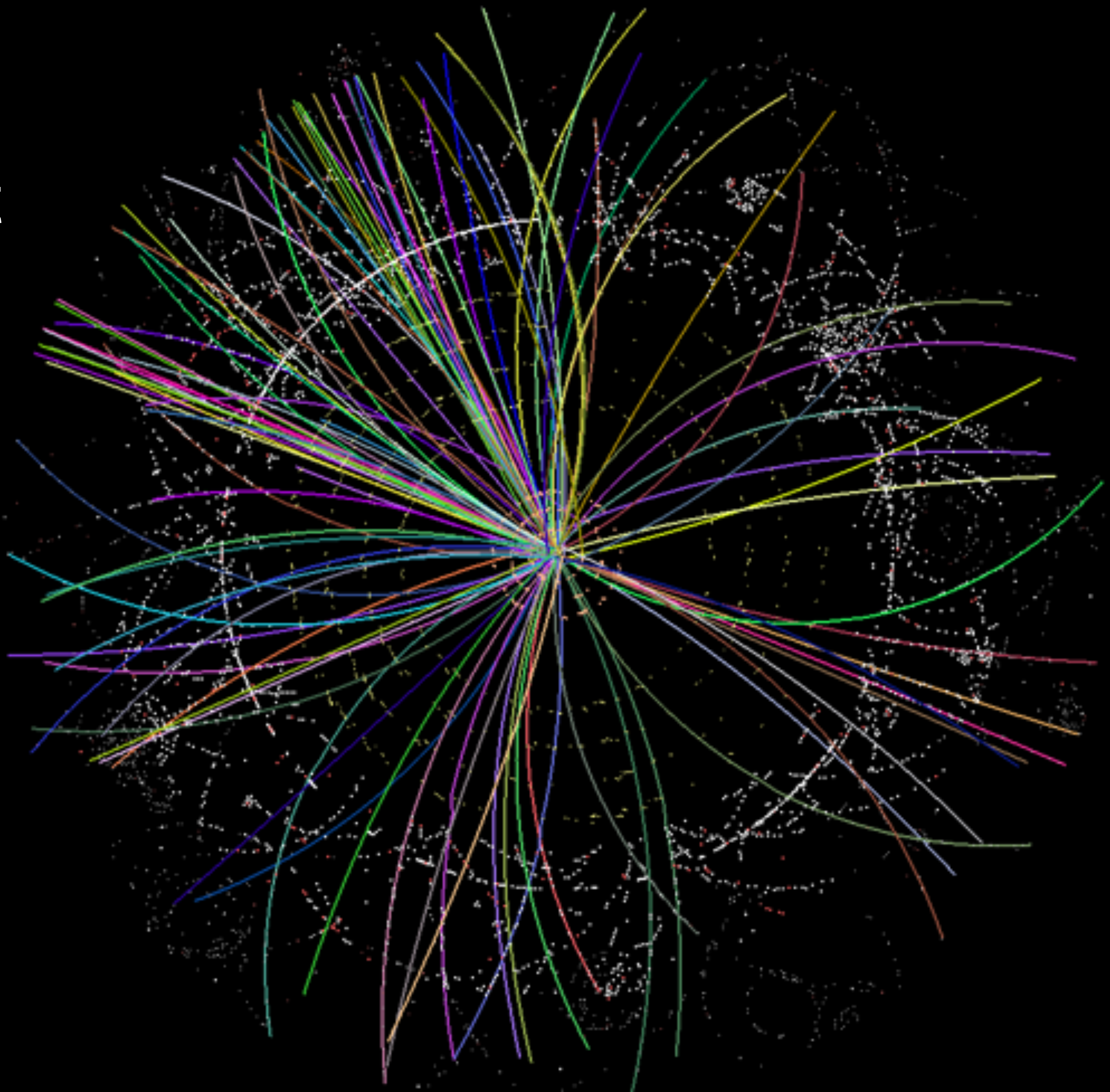
Berkeley Lab
Ernest Orlando Lawrence Berkeley National Laboratory
Caltech
Fermilab

Particle tracking at the LHC

- An interesting and challenging pattern recognition problem
- A very important piece of event reconstruction!

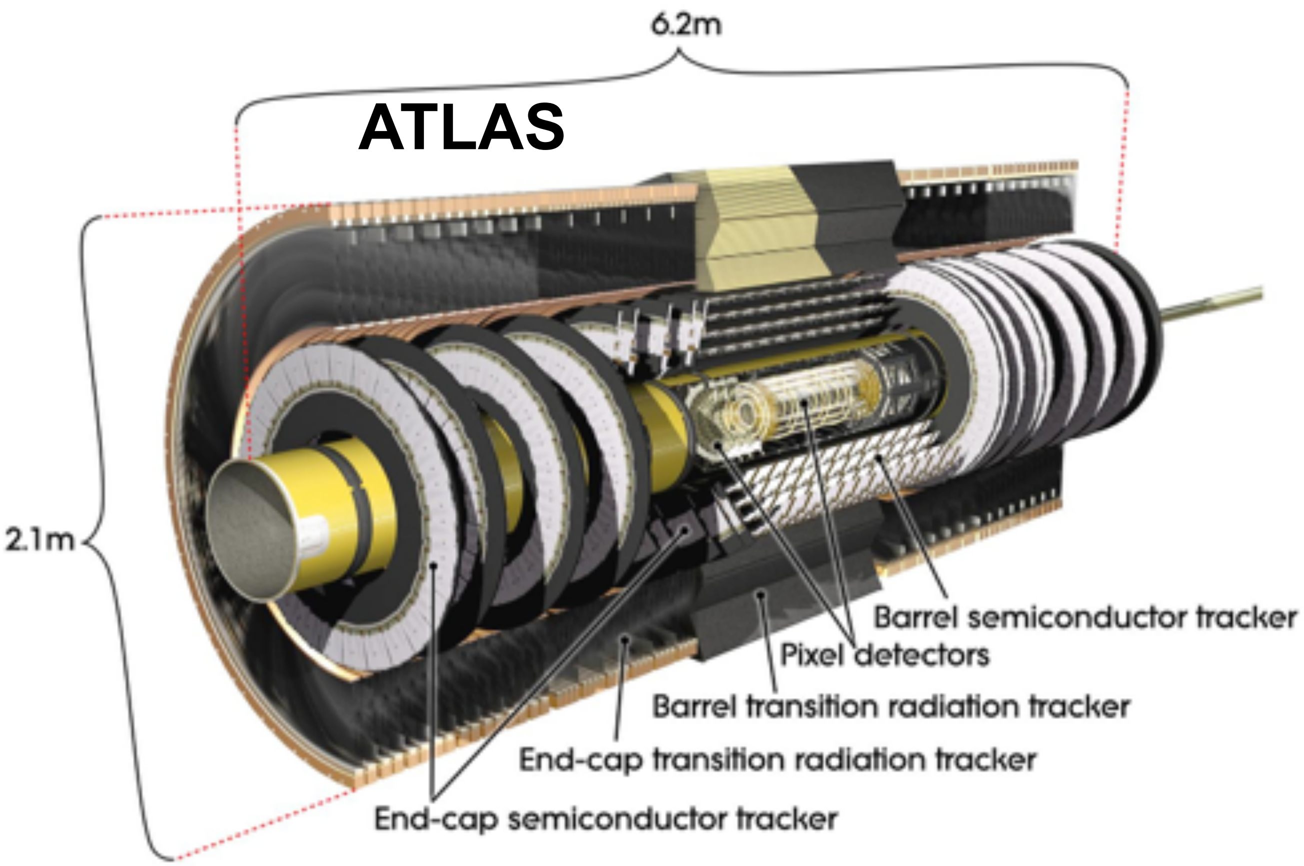


Up to 200 interactions per bunch crossing

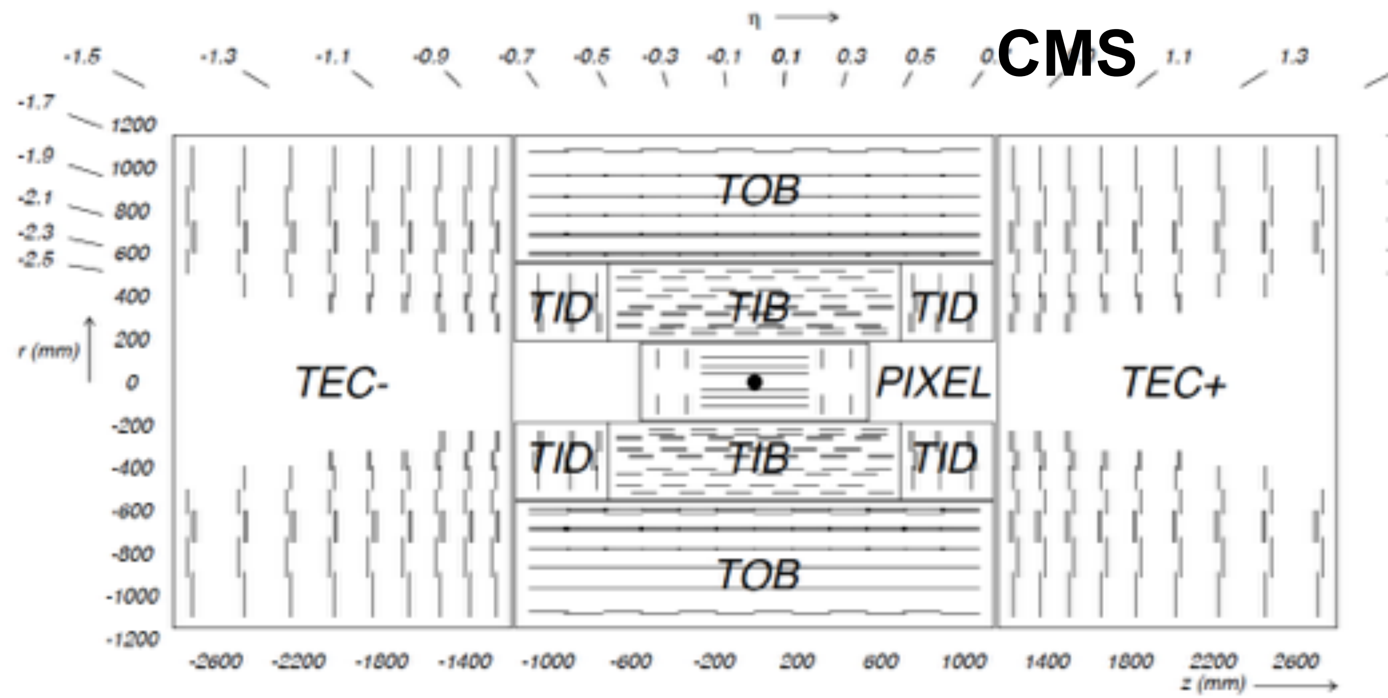


Thousands of charge particle tracks

ATLAS and CMS tracking detectors



http://atlas.cern/discover/detector/inner-detector



http://iopscience.iop.org/article/10.1088/1748-0221/3/08/S08004

- Cylindrical detectors composed of pixel, strip, or TRT layers to detect passage of charged particles
- Both undergoing evolution for HL-LHC
- O(100M) readout channels!

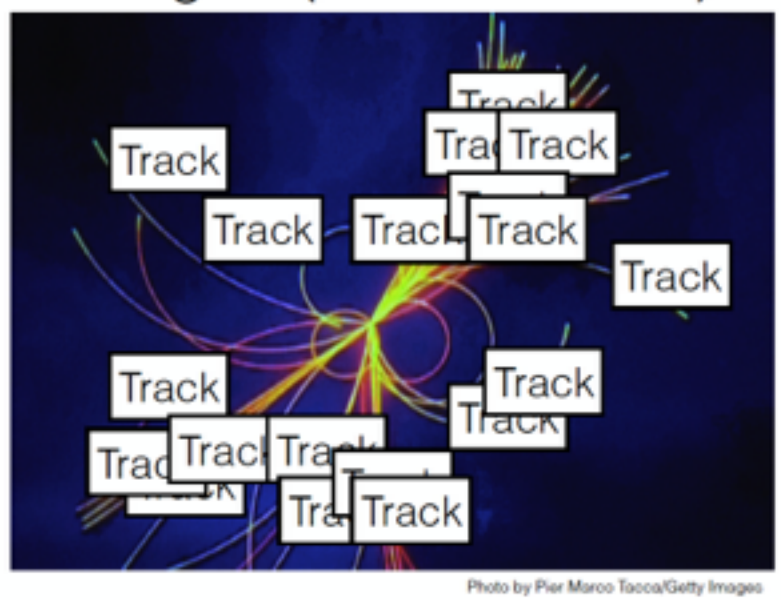
The situation today

- **Current tracking algorithms have been used very successfully in HEP/LHC experiments**
    - Good efficiency and modeling with acceptable throughput/ latency
- **However, they don't scale so well to HL-LHC conditions**
    - Thousands of charged particles, $O(10^5)$ 3D spacepoints, while algorithms scale worse than quadratic
- Thus, it's worthwhile to try and think "outside the box"; i.e., consider ***Deep Learning algorithms***
    - Relatively unexplored area of research
    - Might be able to reduce computational cost or at least increase parallelization
    - Might see major improvements

Some deep learning inspirations

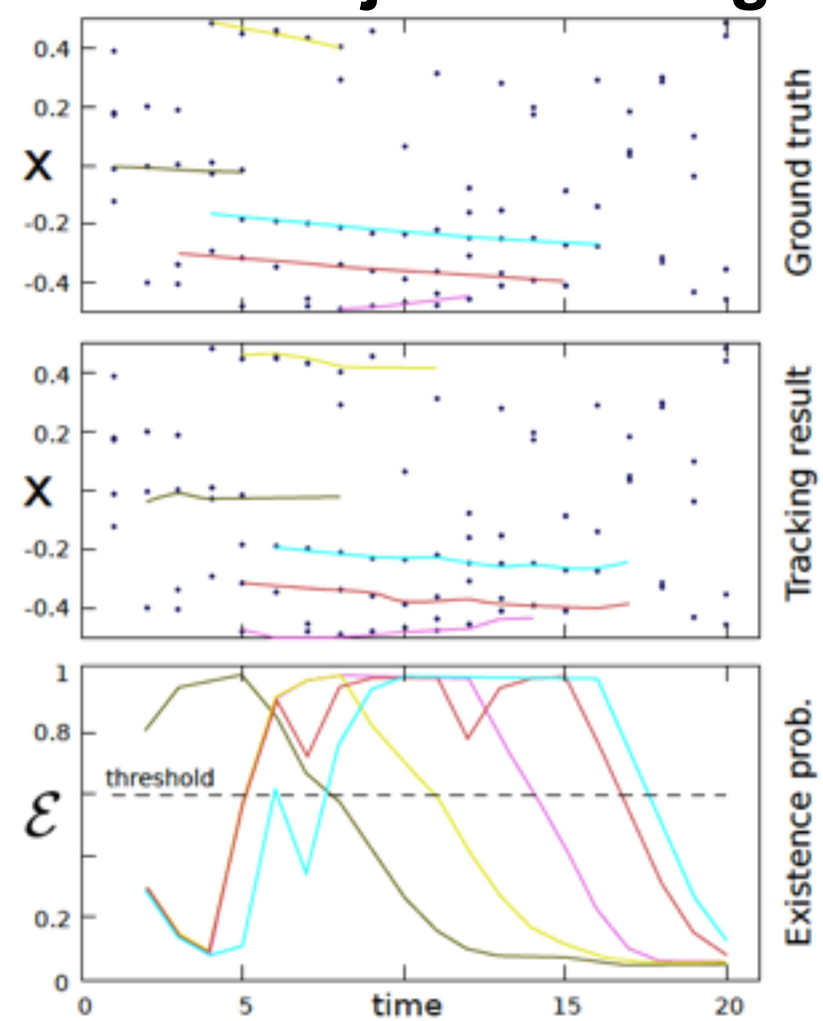
Image segmentation

Our goal (more or less…):



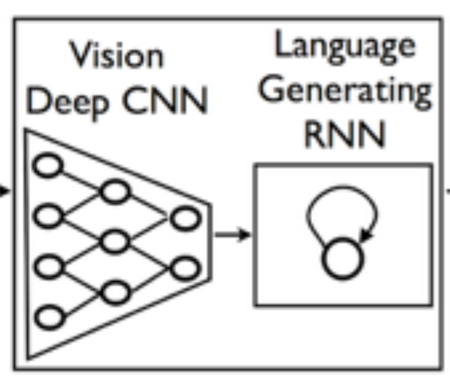
https://arxiv.org/abs/1604.02135

Online object tracking



https://arxiv.org/abs/1604.03635

Image captioning

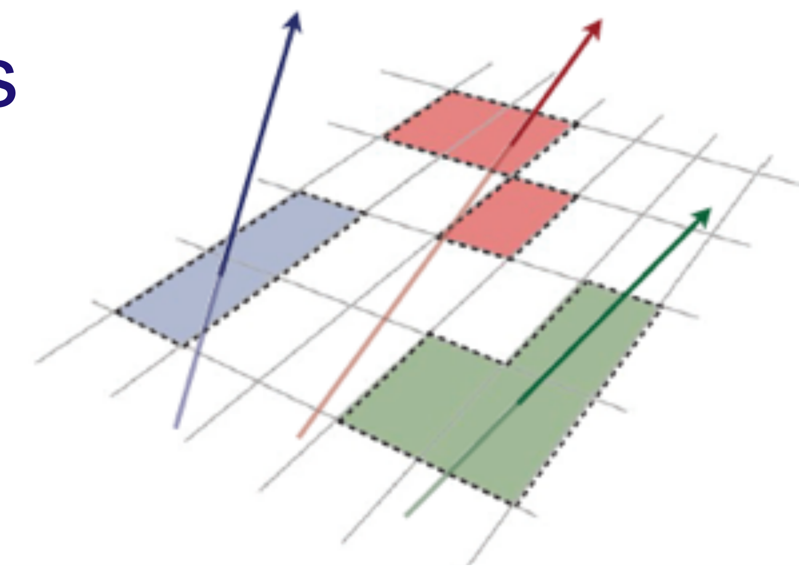


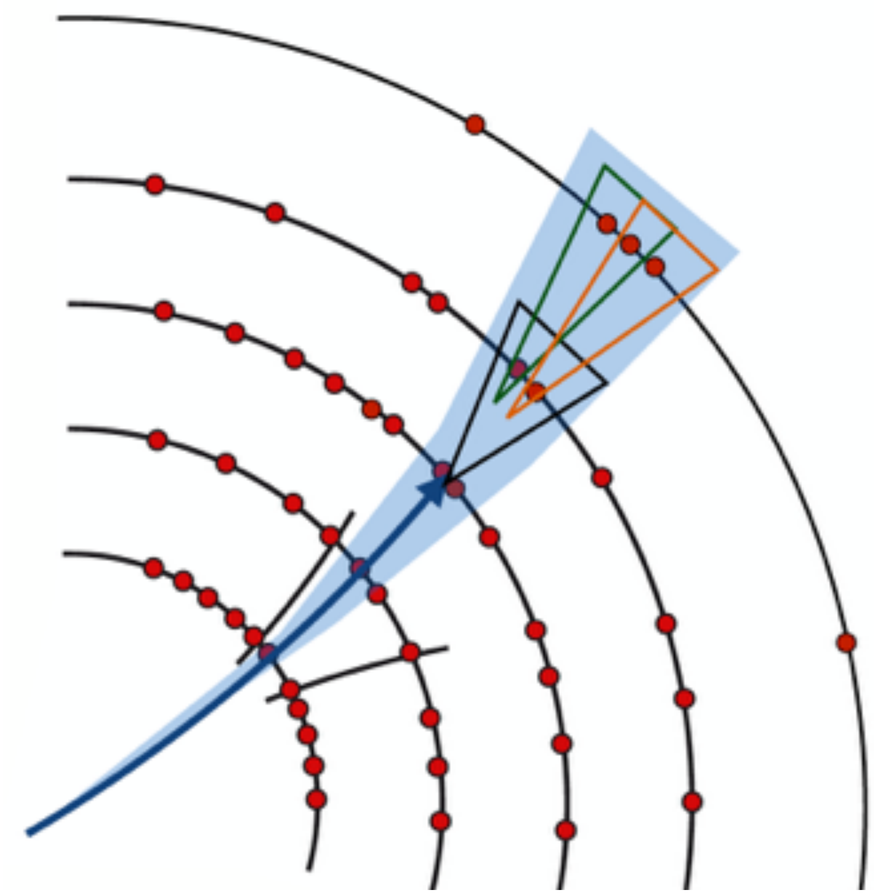
A group of people shopping at an outdoor market.

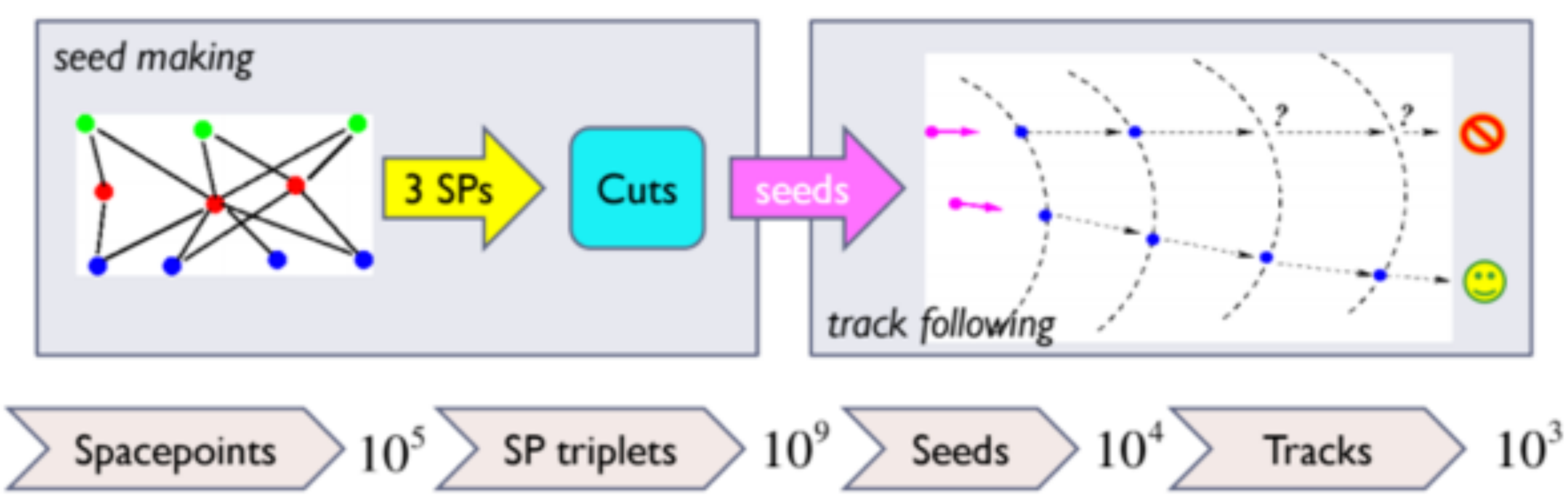
There are many vegetables at the fruit stand.

Current algorithmic approach (ATLAS, CMS)

- Divide the problem into sequential steps
    1. Cluster hits into 3D spacepoints
    2. Build triplet “seeds”
    3. Build tracks with combinatorial Kalman Filter
    4. Resolve ambiguities and fit tracks







Credit: Andy Salzburger

Alternative approaches include Hough transform, Cellular Automaton, RANSAC, etc.

Where to begin?

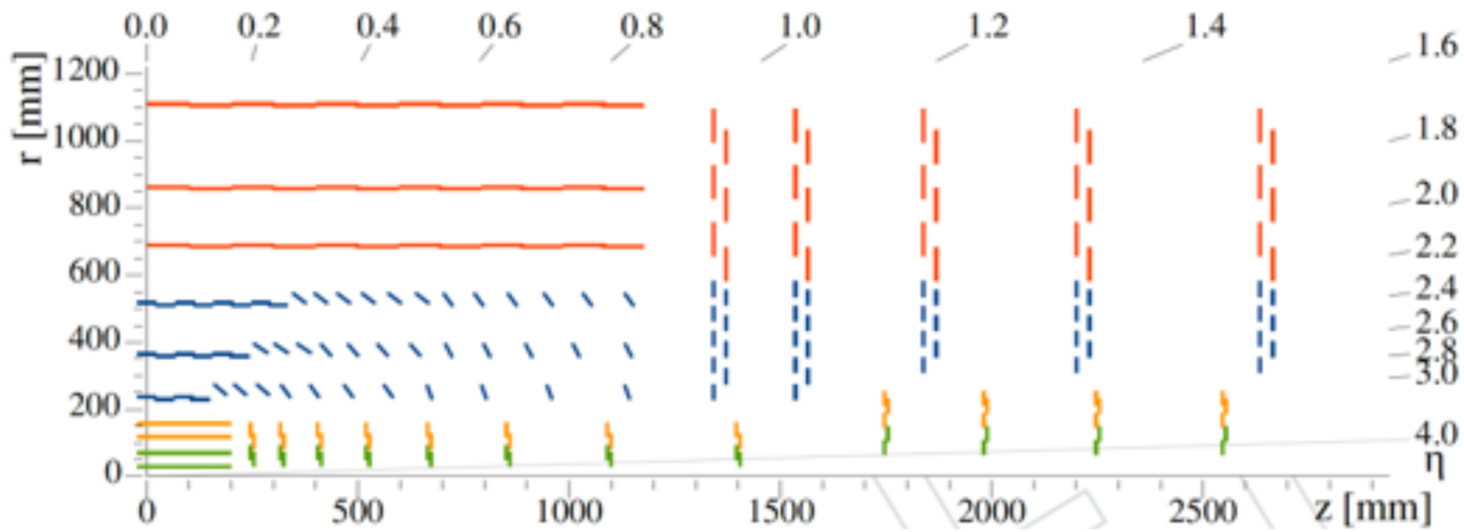
- **What could ML be applied to?**
  - hit clustering
  - seed finding
  - single-track hit assignment
  - multiple-track "clustering"
  - track fitting
  - end to end pixels to tracks
- **How to represent the inputs, outputs (and intermediates)?**
  - discrete vs. continuous space
  - hit assignments vs. physics quantities
  - engineered vs. learned representations

Many options!

Various challenges

- **Data sparsity**
  - Occupancy << 1%
  - Except in dense jets…
- **Data irregularity**
  - Complex geometry
  - Detector inefficiencies, material effects
- **Defining good cost functions**
  - Particularly for multi-track models
  - How to quantify reco efficiency in a differentiable way?
- **Experimental constraints on performance, interpretability**
  - A big deal, for obvious reasons
- **Time and space complexity constraints**
  - Otherwise, what's the point?

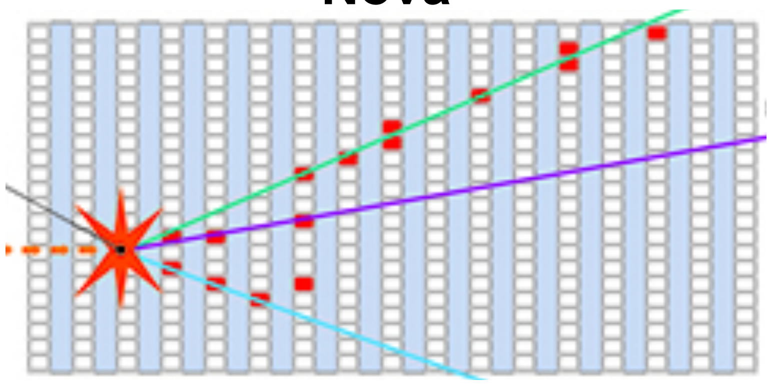
CMS "tilted" proposal for HL-LHC



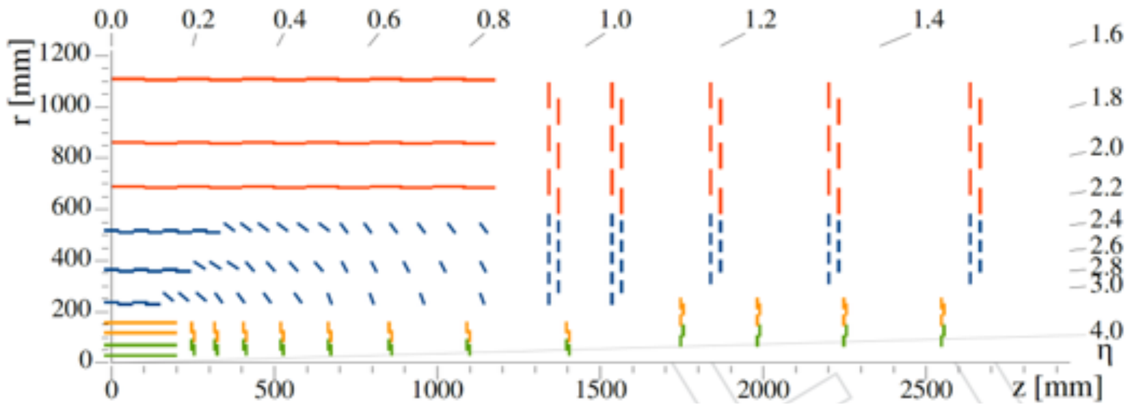
Detector images

- Neutrino experiments may have nice "image" detectors, but it's a bit harder with LHC detectors!

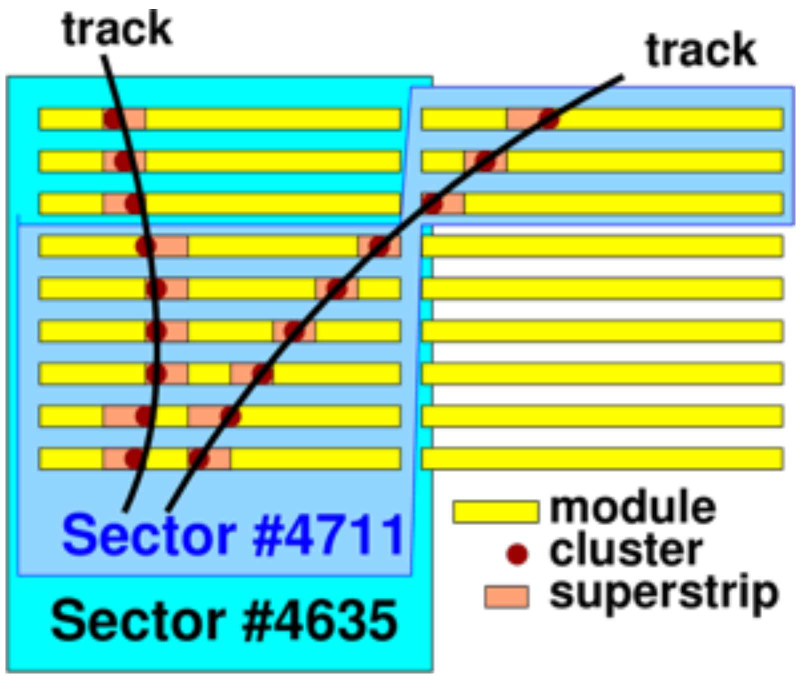
Nova



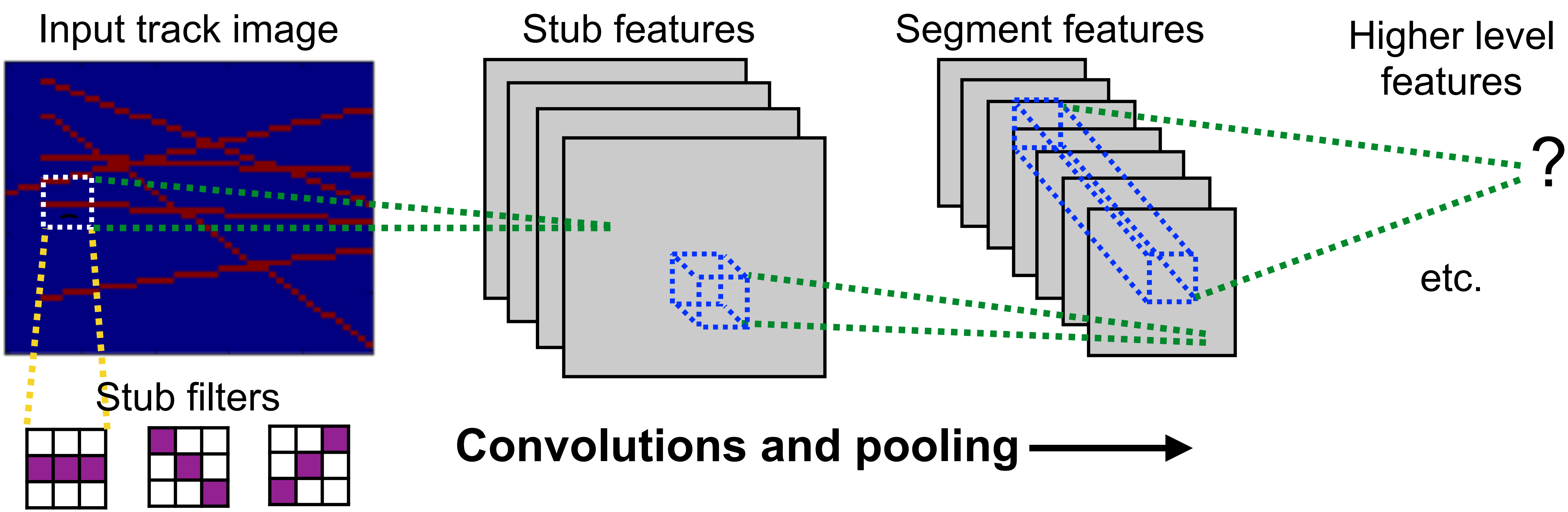
CMS "tilted" proposal



  - Maybe we can unroll + flatten the barrel layers
  - …but size increases with each detector layer
- Raw data is extremely high dimensional ($O(10^8)$ channels!)
  - Maybe we can coarsen it (like AM methods)
  - Smart down-sampling needed
    - CV techniques are good at this



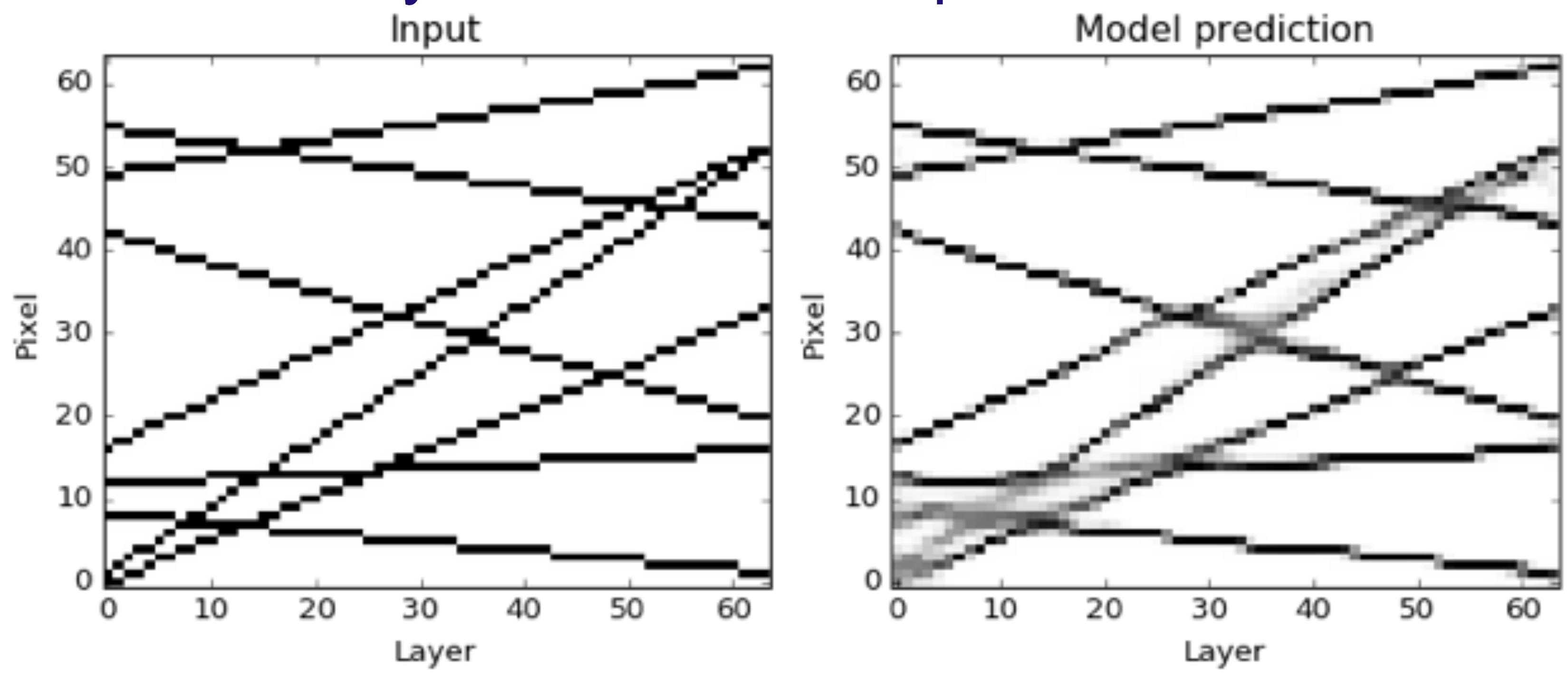
Convolutional networks as track finders



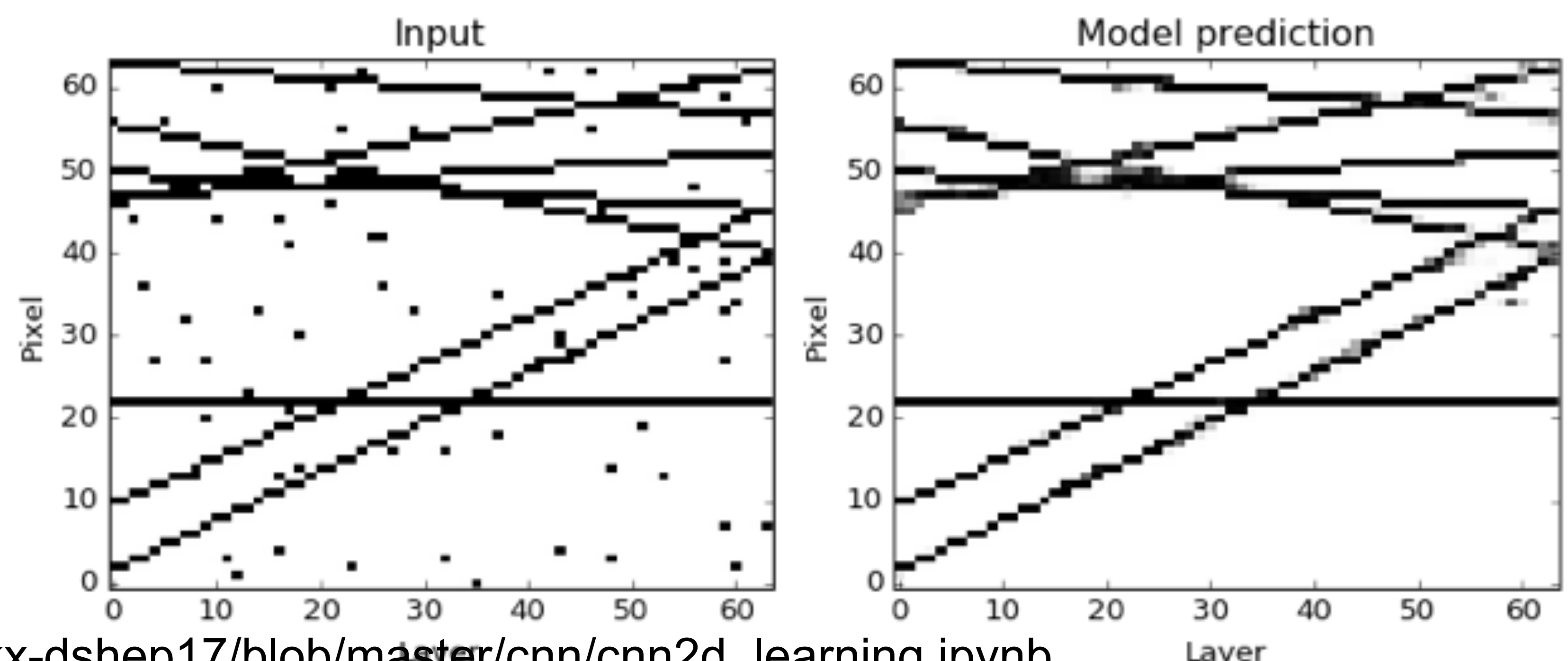
- **Convolutional filters can be thought of as track pattern matchers**
  - Early layers look for track stubs
  - Later layers connect stubs together to build tracks
  - Learned representations are in reality optimized for the data => may be abstract and more compact than brute force pattern bank
- **The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

What can CNNs learn about tracks?

- **Convolutional auto-encoder**: can it learn a smaller-dimensional representation that allows it to fully reconstruct its inputs?
  - Decently well



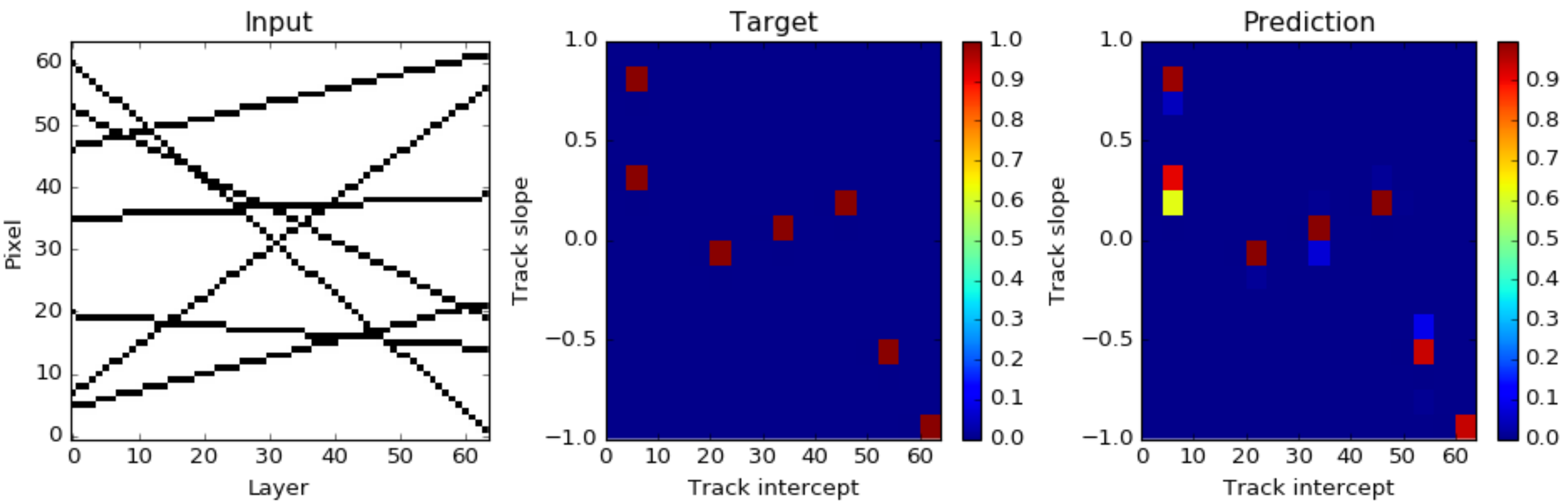
- **De-noising**: can it clean out noise hits?
  - Seems so



https://github.com/HEPTrkX/heptrkx-dshep17/blob/master/cnn/cnn2d_learning.ipynb

What can CNNs learn about tracks?

- **Track parameter estimation**: can it predict the tracks' parameters?
  - Some inspiration from Hough Transform: binned parameter space with peaks at the correct values
  - By converting regression problem into discrete classification problem, can handle variable number of tracks with relatively simple CNN architecture



  - Might be an interesting approach, but it has limitations
    - doesn't map params onto the hits like Hough
    - precision comes at cost of dimensionality

https://github.com/HEPTrkX/heptrkx-dshep17/blob/master/cnn/cnn2d_learning.ipynb

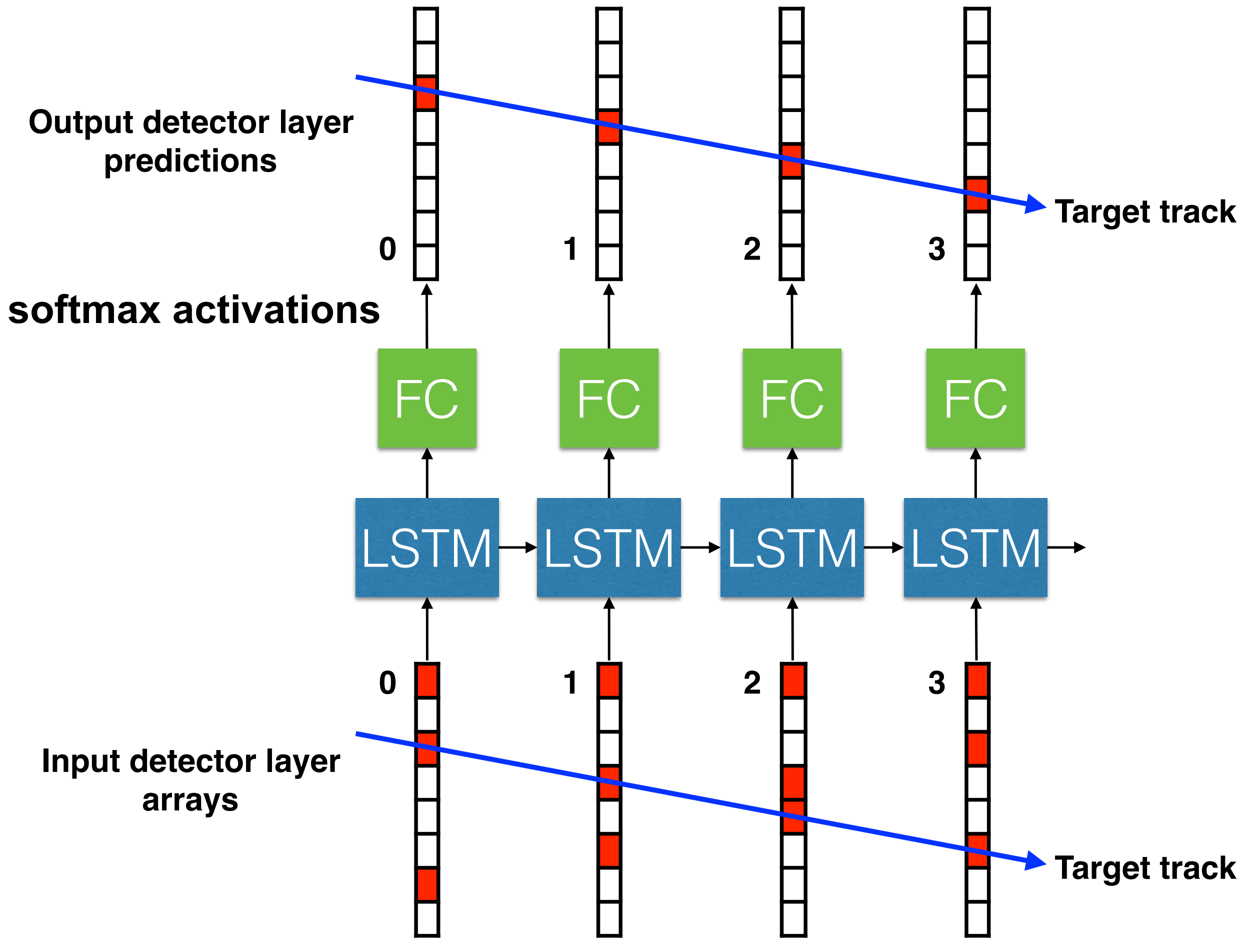
Ongoing HEP.TrkX studies

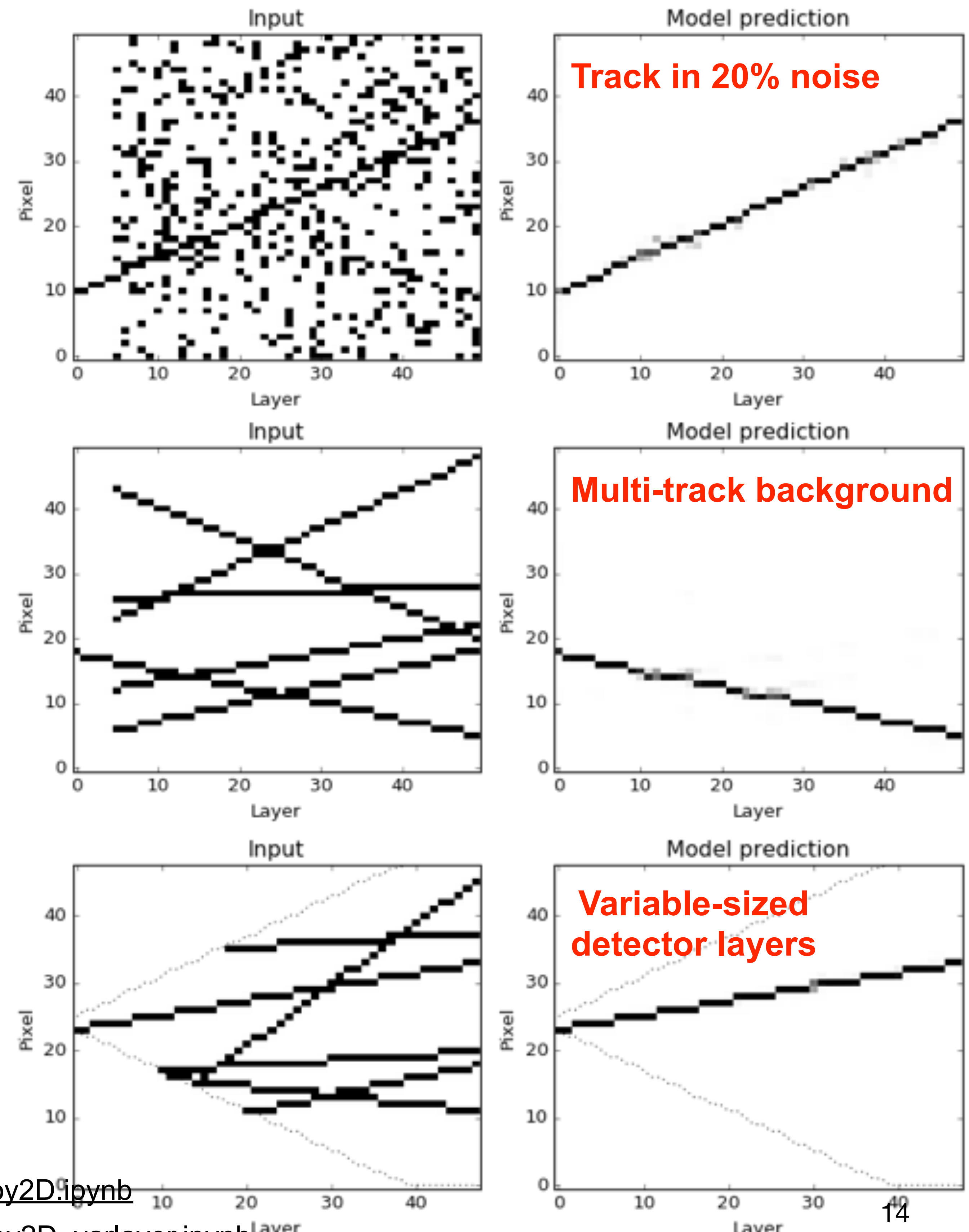
- **About the project**
  - https://heptrkx.github.io/
  - Pilot project funded by DOE ASCR and COMP HEP
  - Part of HEP CCE
  - People:

LBL: Me, Mayur Mudigonda, Prabhat, Paolo
Caltech: Dustin Anderson, Jean-Roch Vlimant, Josh Bendavid, Maria Spiropoulou, Stephan Zheng
FNAL: Aristeidis Tsaris, Giuseppe Cerati, Jim Kowalkowski, Lindsey Gray, Panagiotis Spentzouris

- **Exploratory work on toy datasets**
  - Hit classification for seeded tracks with LSTMs and CNNs
  - End-to-end track parameter estimation with CNN + LSTM
  - and some others

Hit classification with LSTMs in 2D





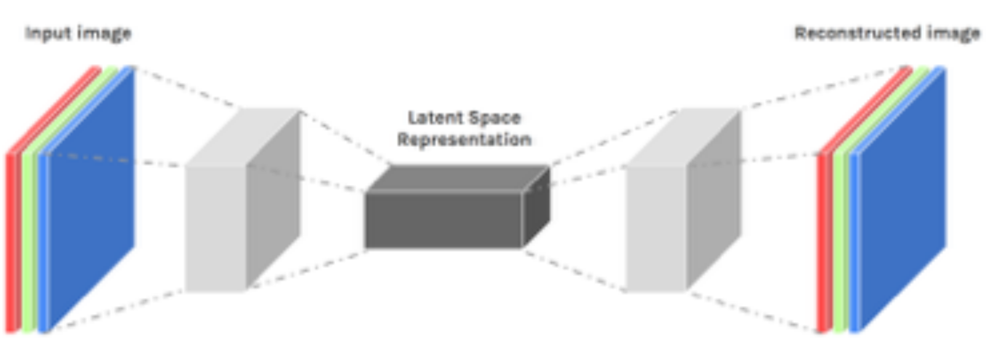
- Seeded track inputs, pixel score outputs per detector layer
- Works decently well
- Can be extended to multiple input seeds and output channels

https://github.com/HEPTrkX/heptrkx-ctd/blob/master/hit_classification/lstm_toy2D.ipynb

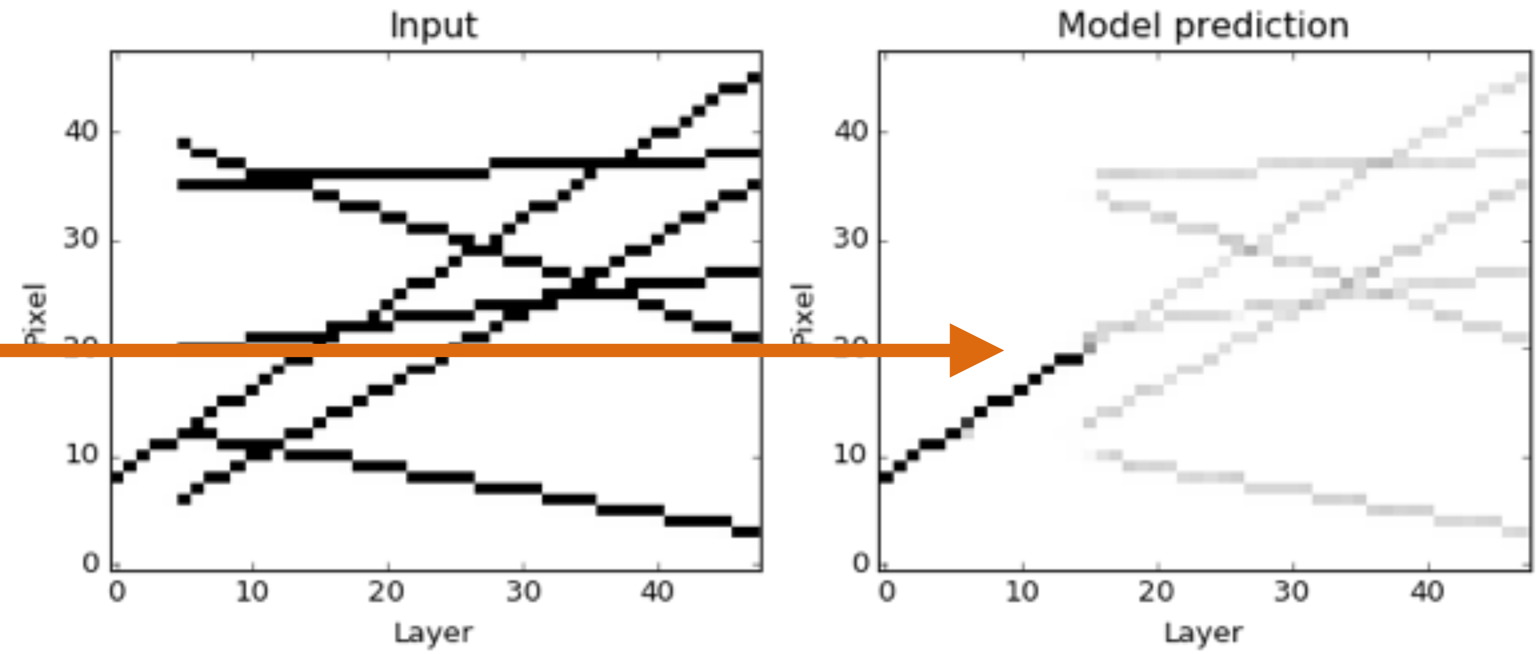
https://github.com/HEPTrkX/heptrkx-ctd/blob/master/hit_classification/lstm_toy2D_varlayer.ipynb

Hit classification with CNNs in 2D

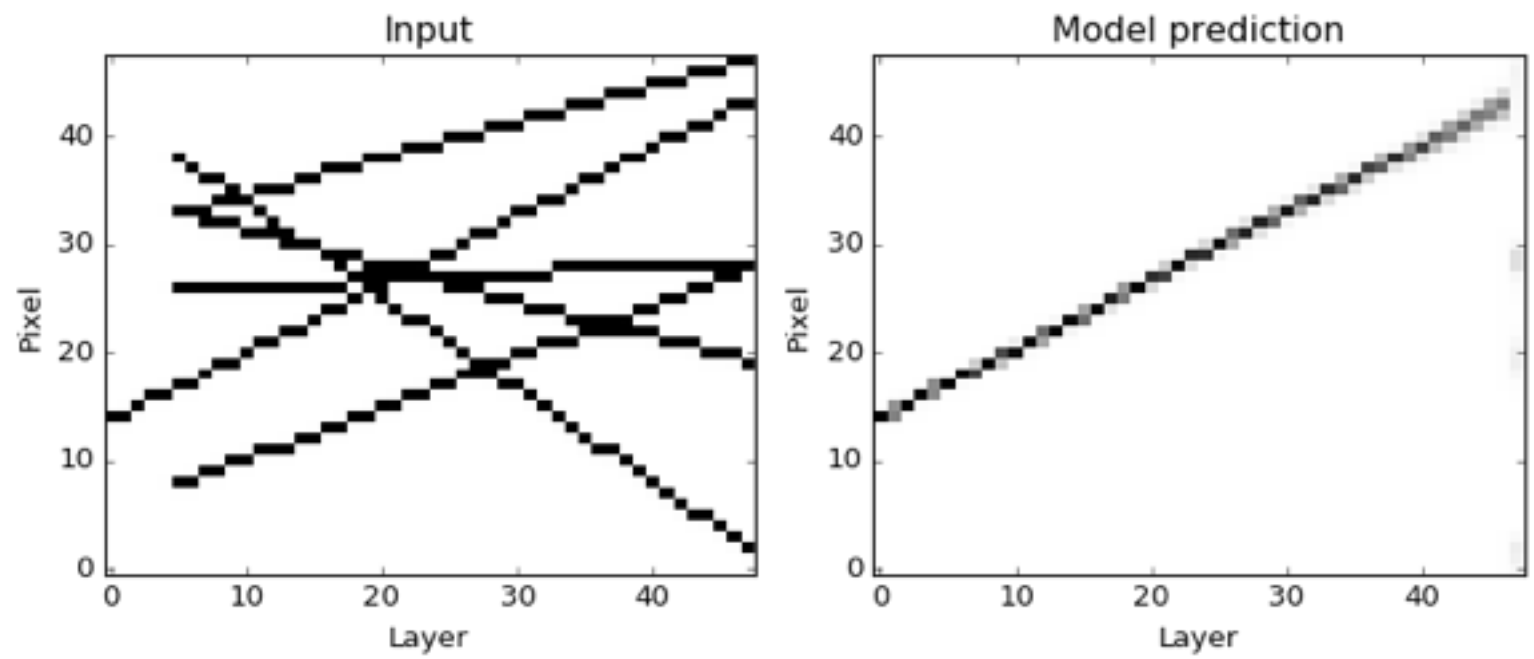
- CNNs can also extrapolate and find tracks
- Extrapolation reach may be limited without downsampling
- Autoencoder architecture allows to extrapolate farther



Trained with 10 conv layers, no down-sampling

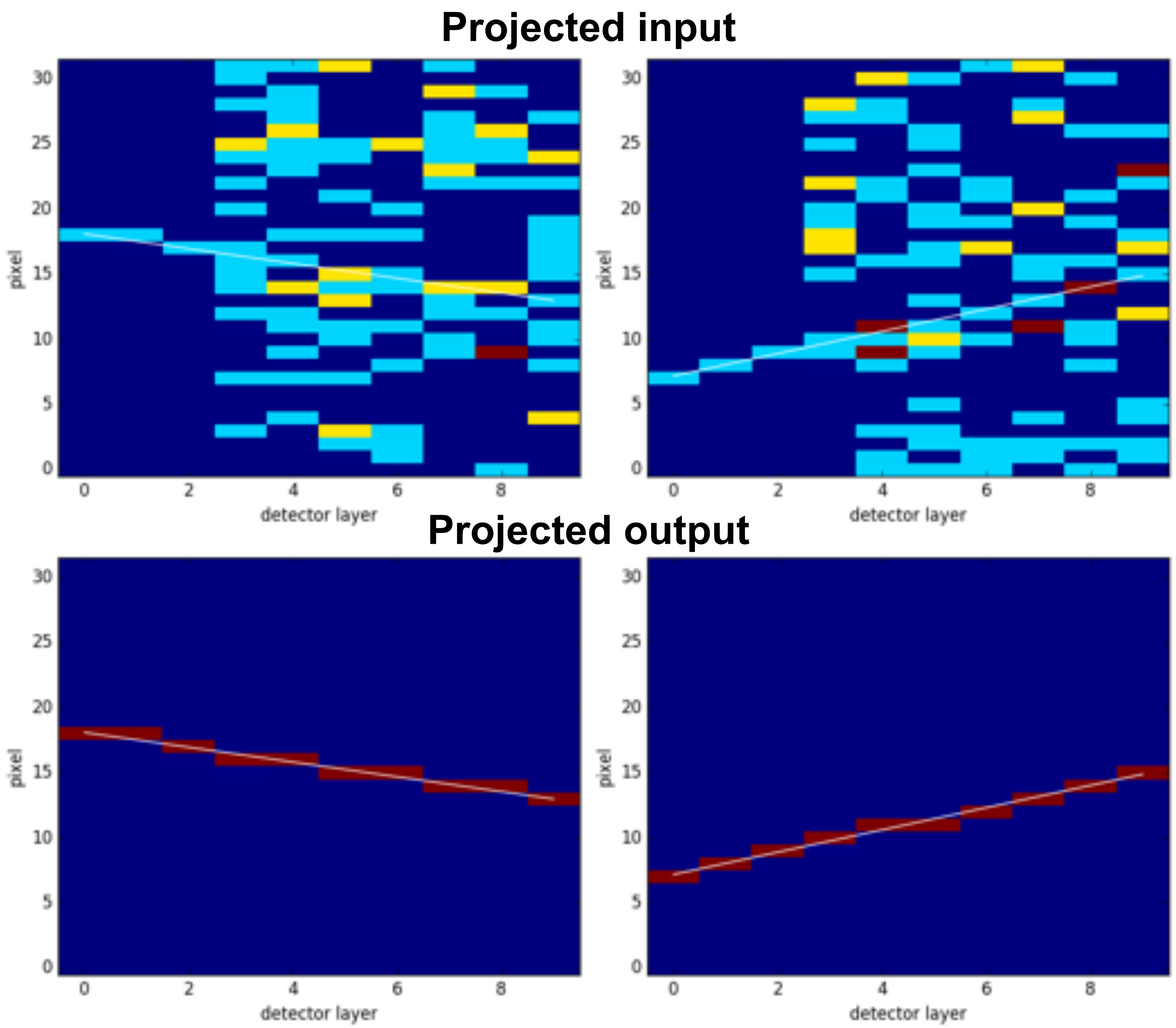


9-layer convolutional “autoencoder"

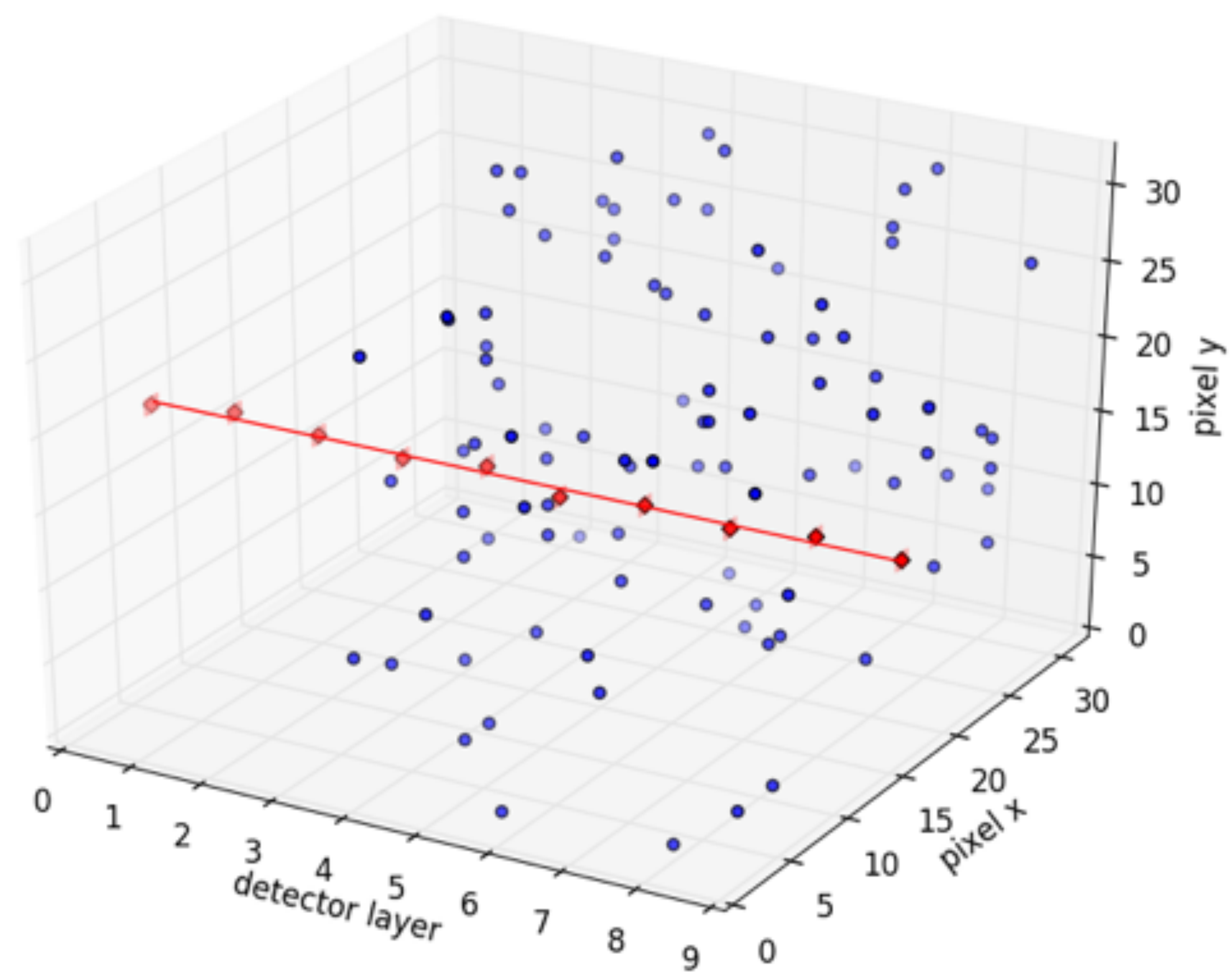


https://github.com/HEPTrkX/heptrkx-ctd/blob/master/hit_classification/cnn_toy2D.ipynb

Hit classification with CNNs in 3D

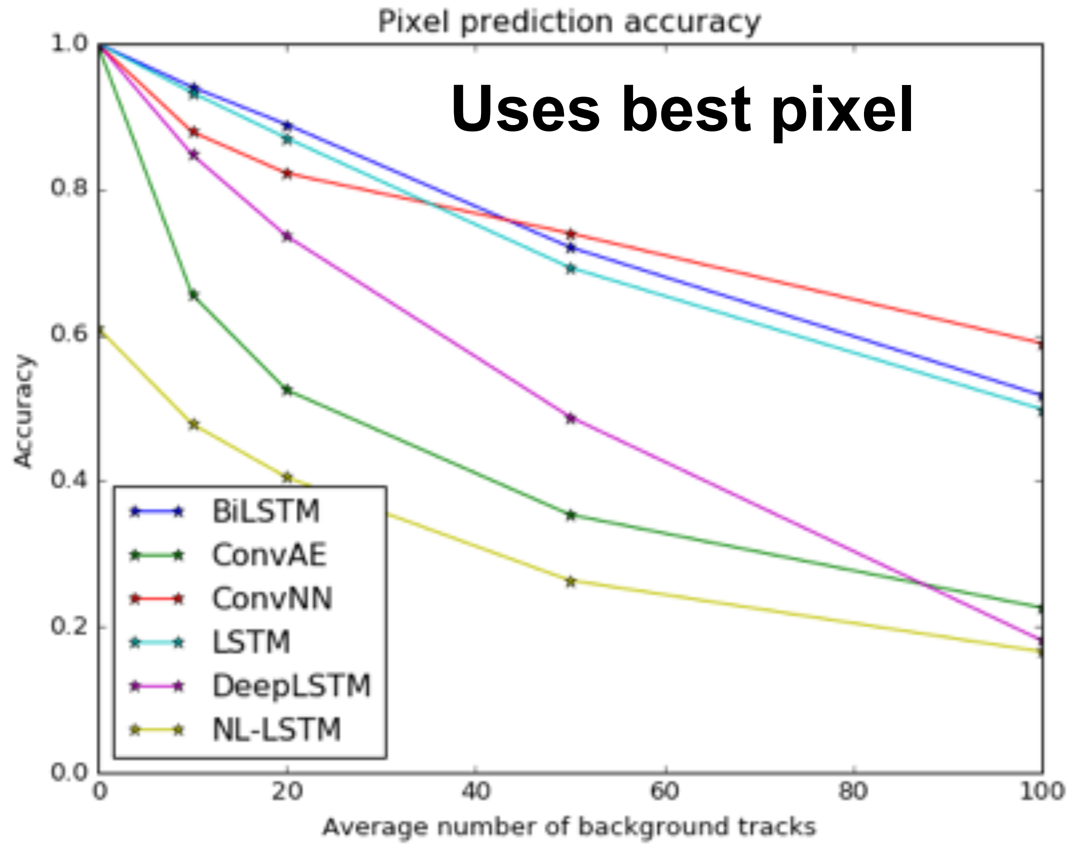


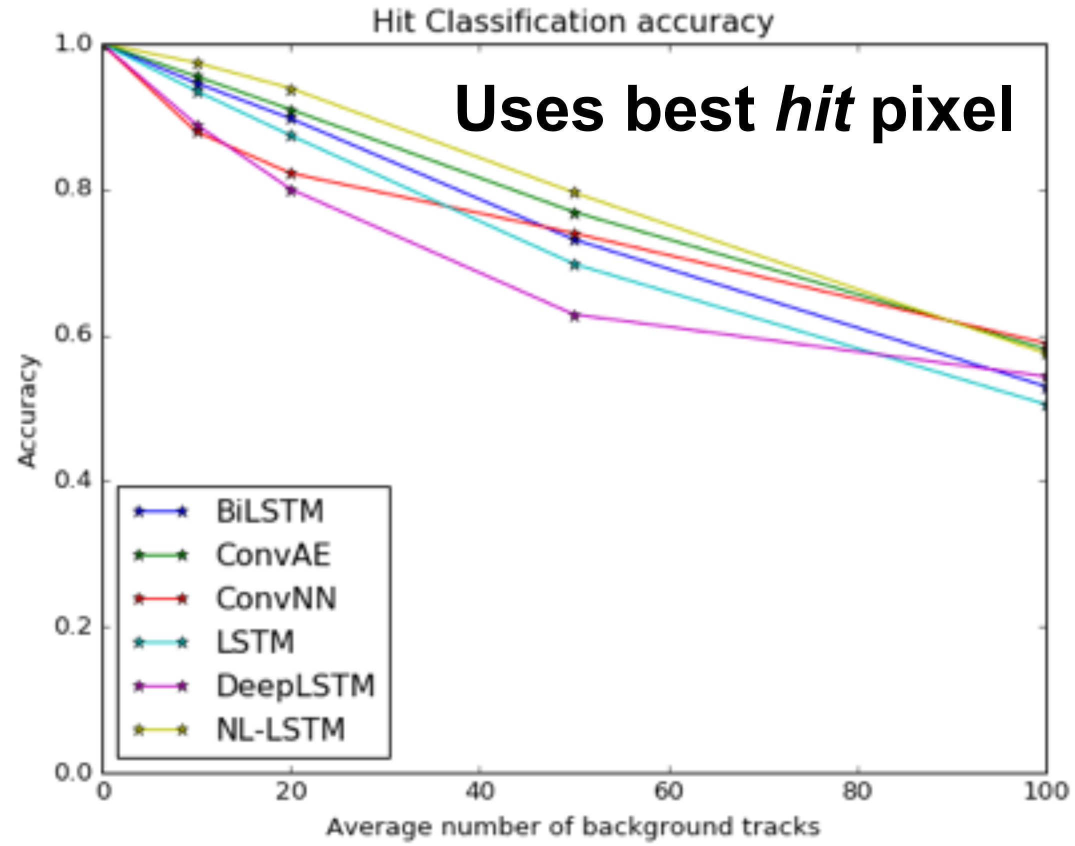
3 avg bkg tracks, 1% noise



- Basic CNN model with 10 layers and 3x3x3 filters
- Gives nice clean, precise predictions

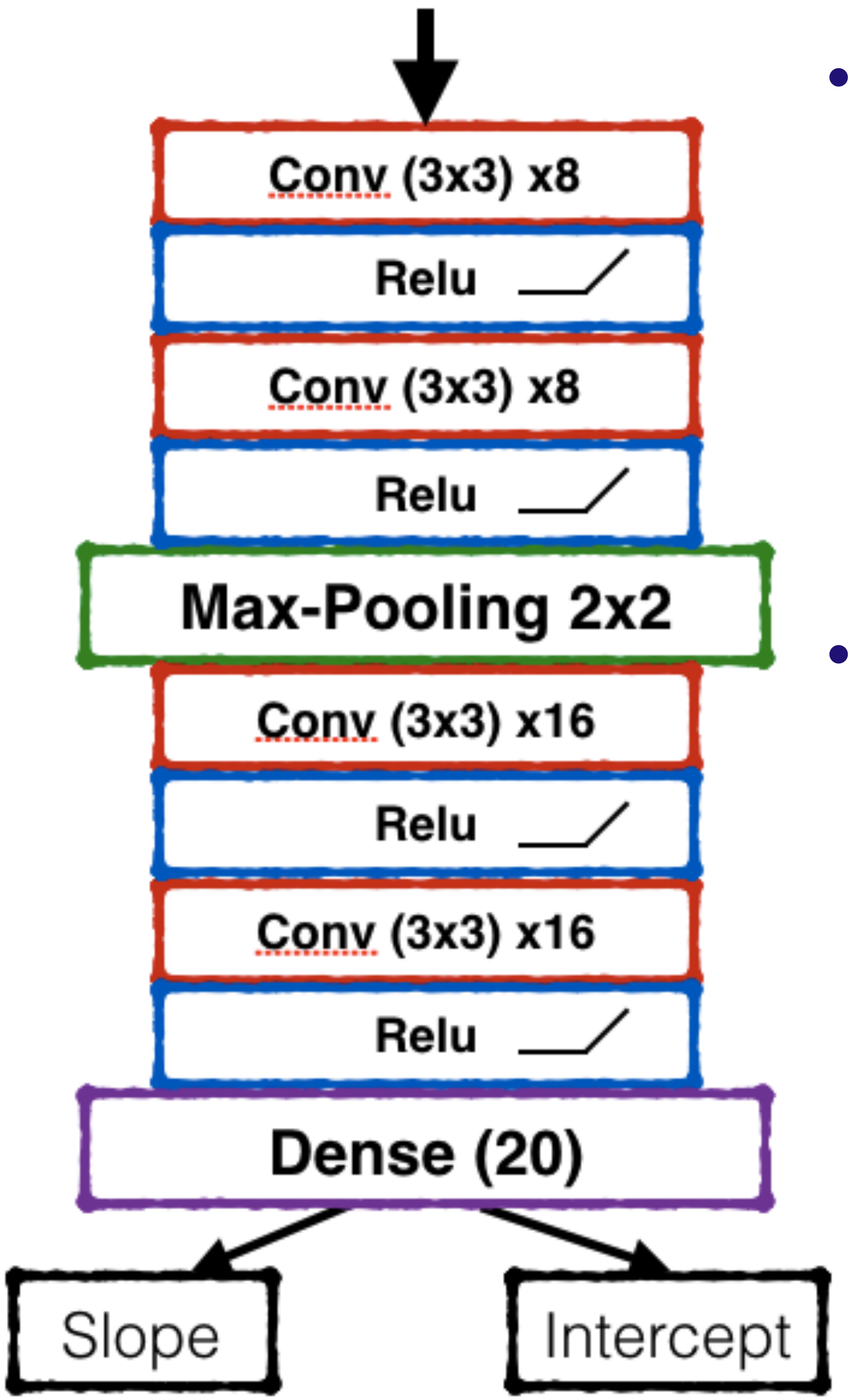
Architecture comparisons



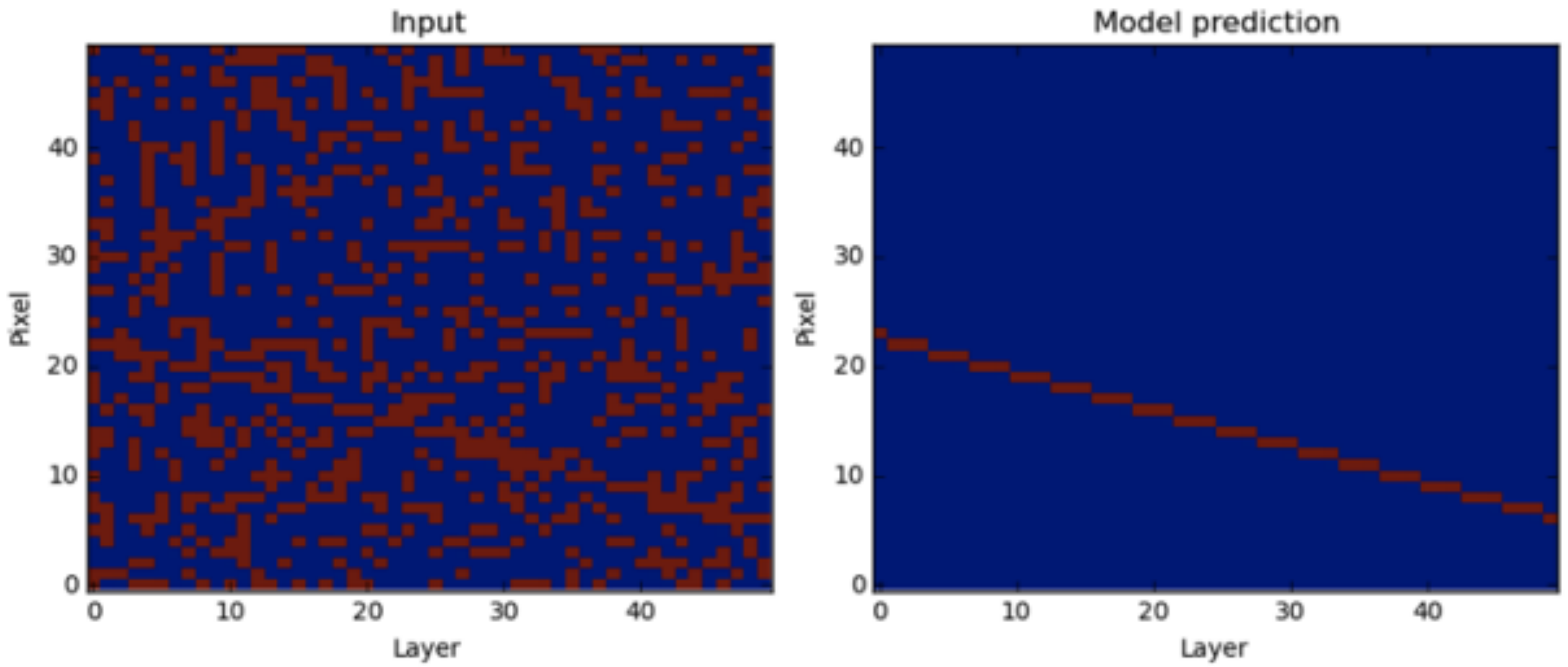


- Both LSTMs and CNNs do well at classifying hits for reasonable occupancy
- Models’ performance degrades with increasing track multiplicity
- CNNs seem to scale well to high track multiplicity

Track parameter estimation

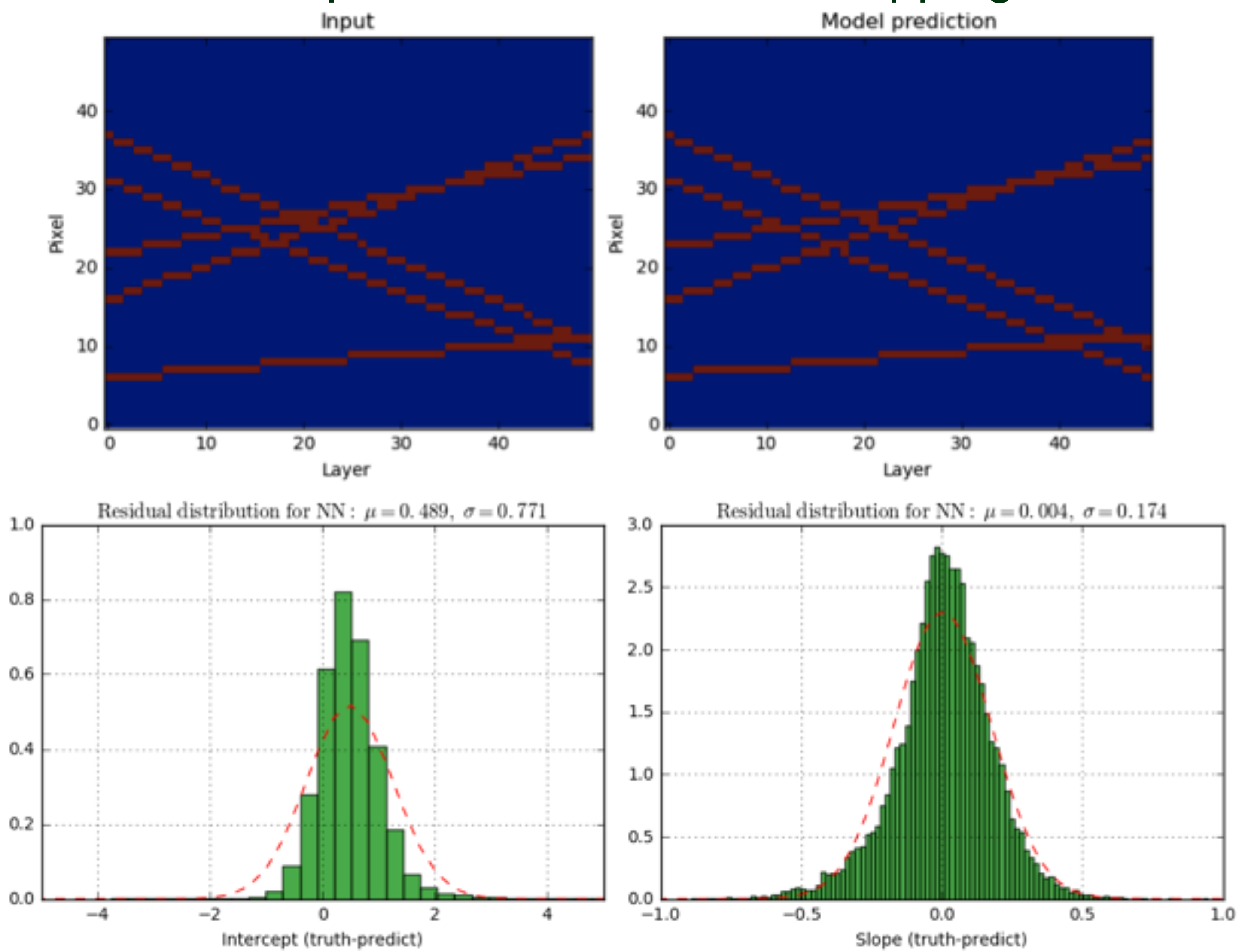


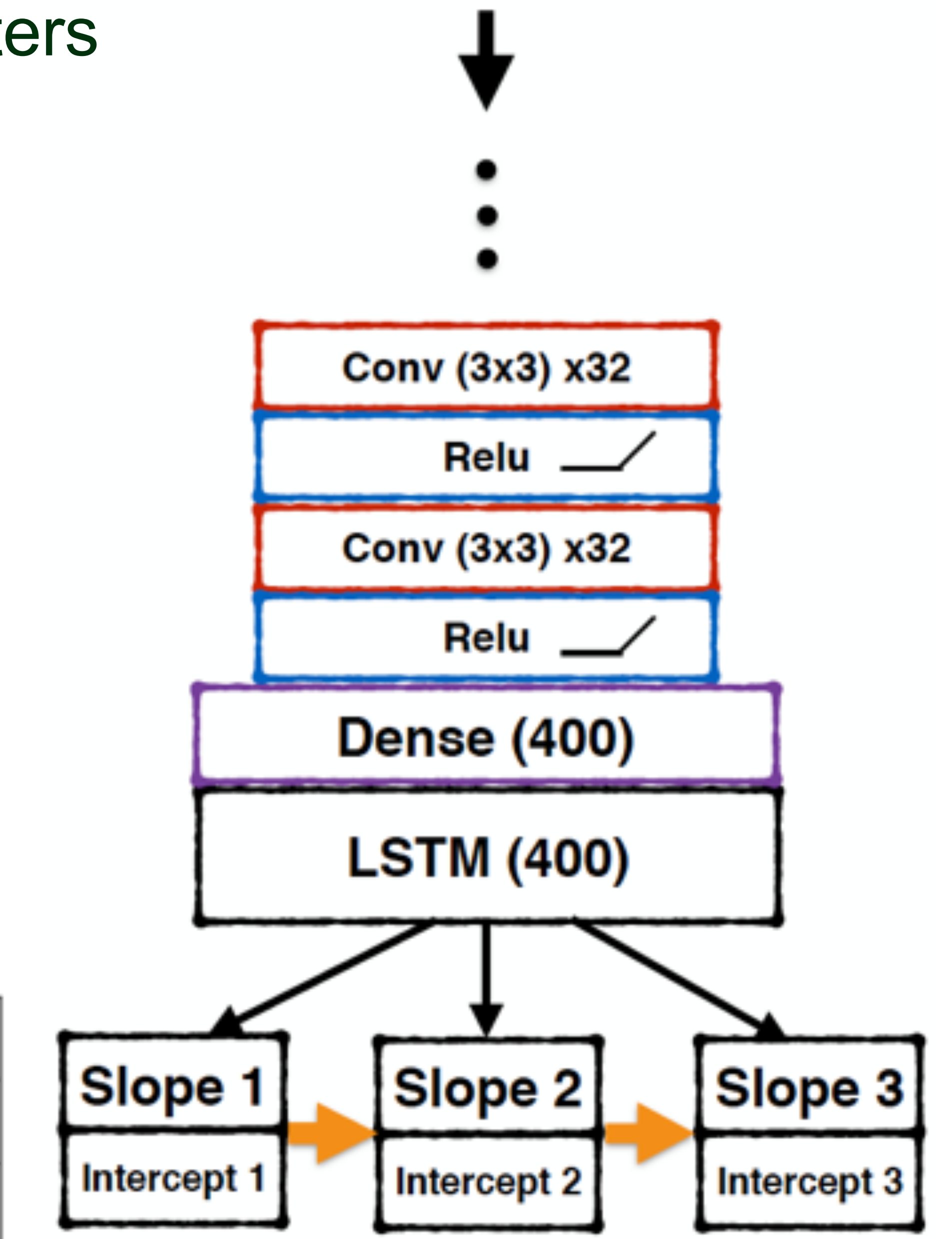
- Use a basic CNN with downsampling and regression head to estimate a track's parameters
  - could be an auxiliary target to guide training, or potentially useful as the final output of tracking!
- Identifying straight line params in heavy noise:



Extending to variable number of tracks

- Attach an LSTM to a CNN to emit parameters for a variable number of tracks!
  - The LSTM generates the sequence of parameters
  - Requires an ordering the model can learn
  - Should provide some kind of stopping criteria

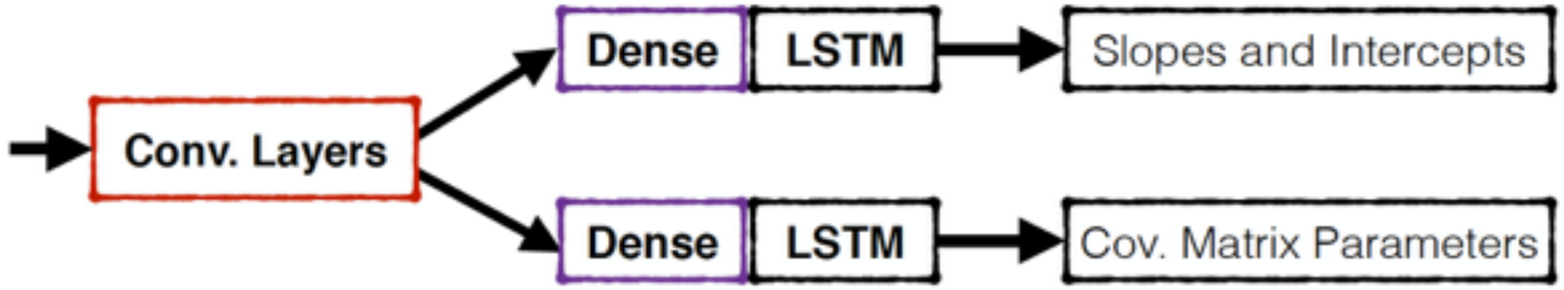




[Work of Dustin Anderson]

Estimating uncertainties on parameters

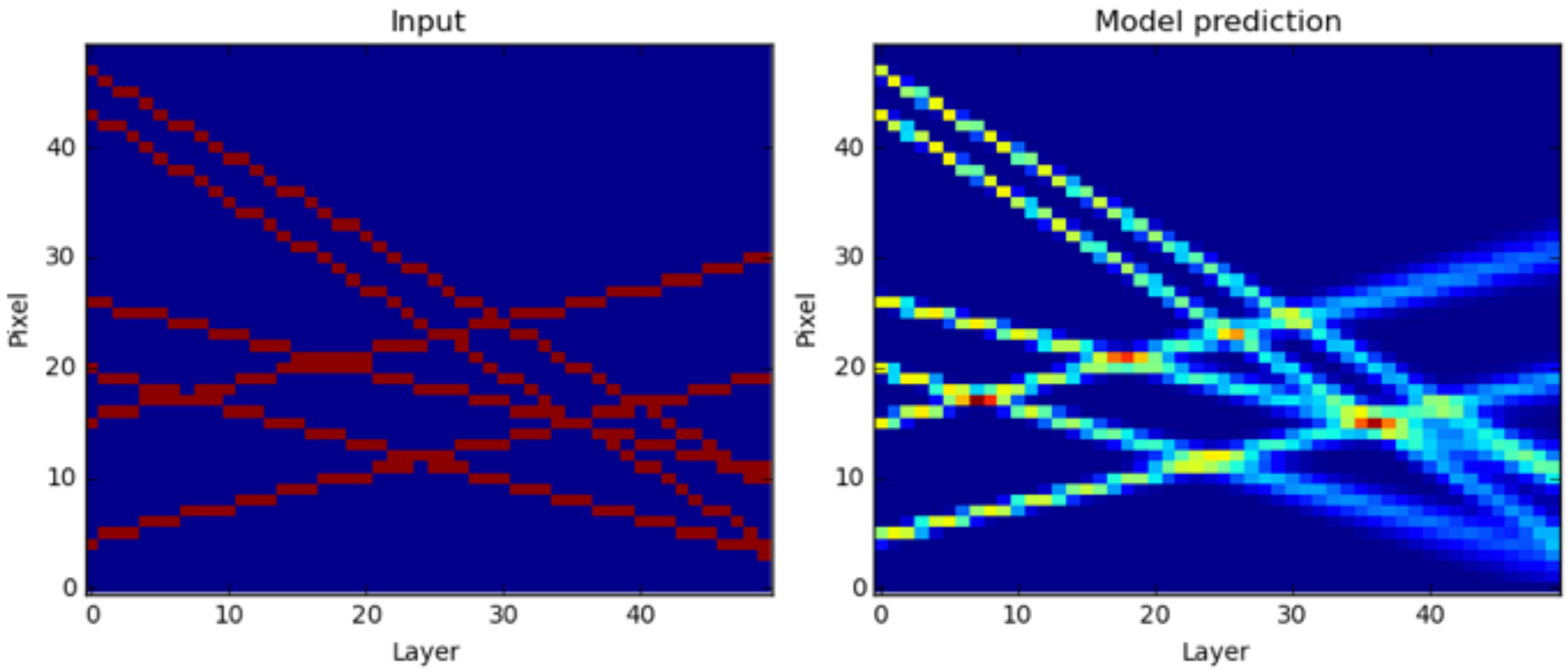
- Train the model to also estimate the uncertainties by adding additional targets:



- Train using a log gaussian likelihood loss:

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

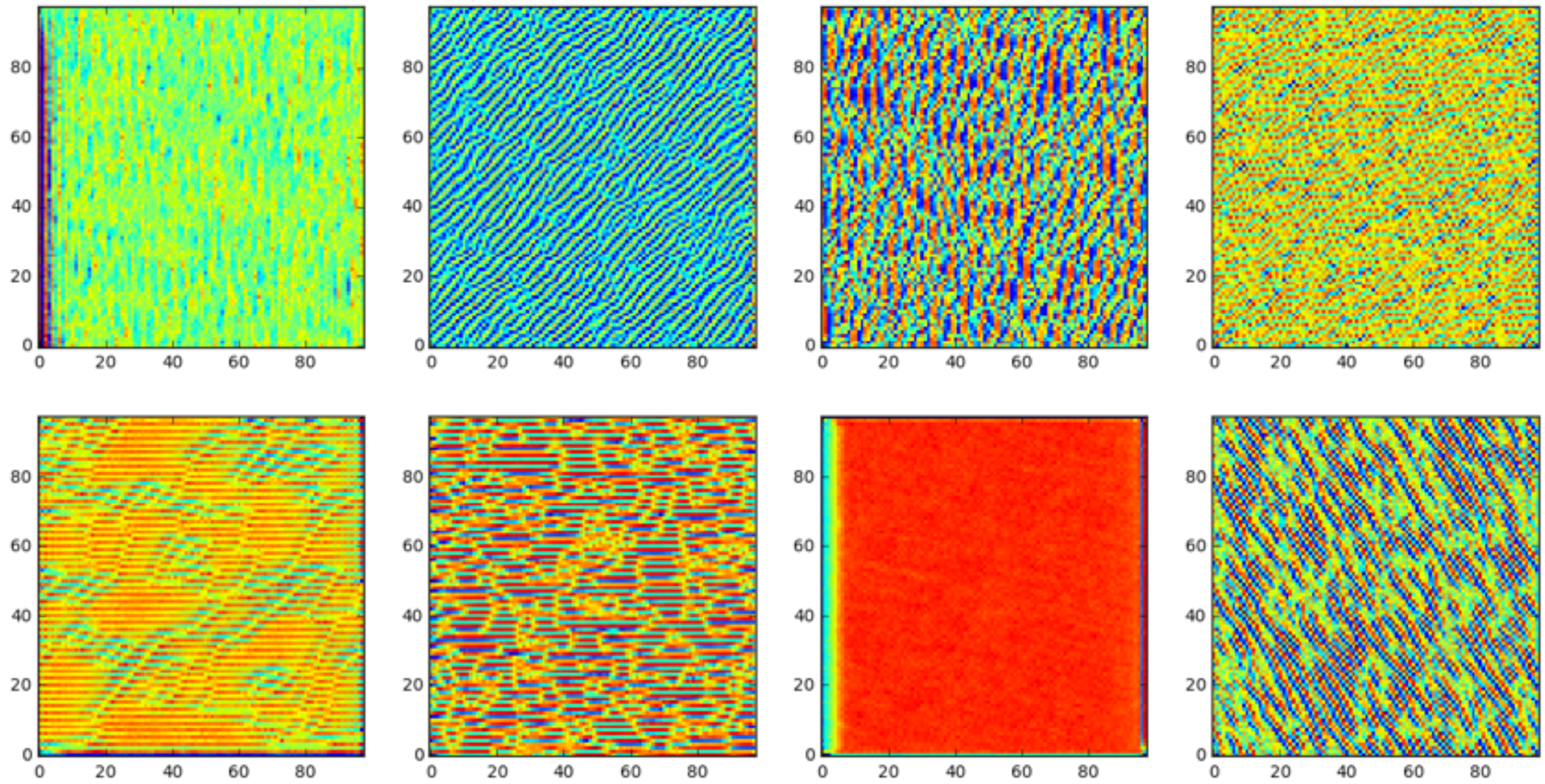
- and *voila!*



[Work of Dustin Anderson]

Visualizing CNN features

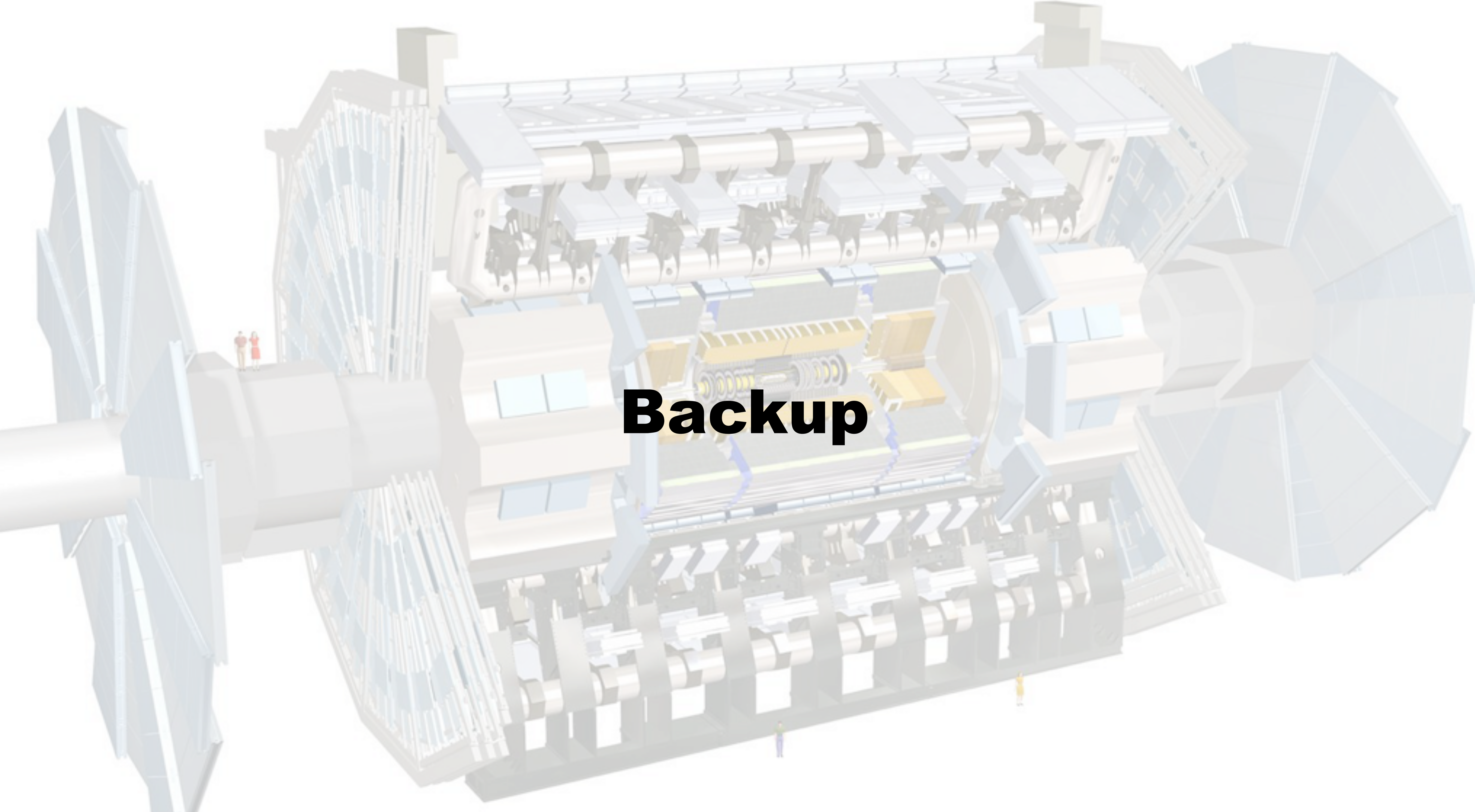
- We can visualize what the CNN is learning by finding images which maximize a particular filter's activation
- Here are the 2nd layer filters of the CNN+LSTM track parameter model



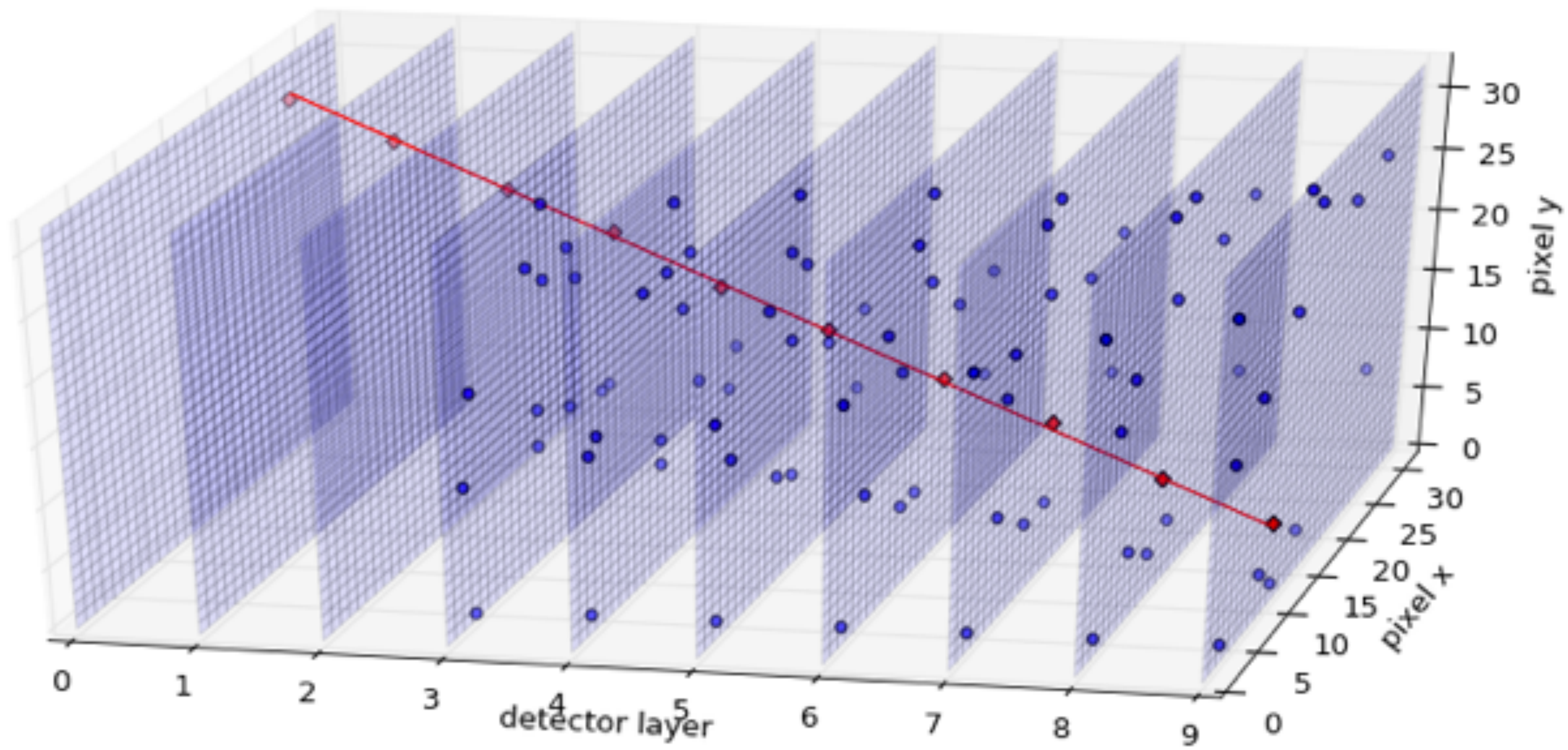
Conclusion

- There is some hope that deep learning techniques could be useful for particle tracking
  - Powerful non-linear modeling capabilities
  - Learned representations > engineered features
  - Easy parallelization
- It’s not yet known if computer vision techniques like CNNs offer the most promise, but they have some nice features
  - They can learn useful things about the data and seem versatile
  - Some successes seen with highly simple toy datasets
- Where do we go from here?
  - Try to apply these ideas to realistically complex data
  - Continue thinking up new approaches

Backup



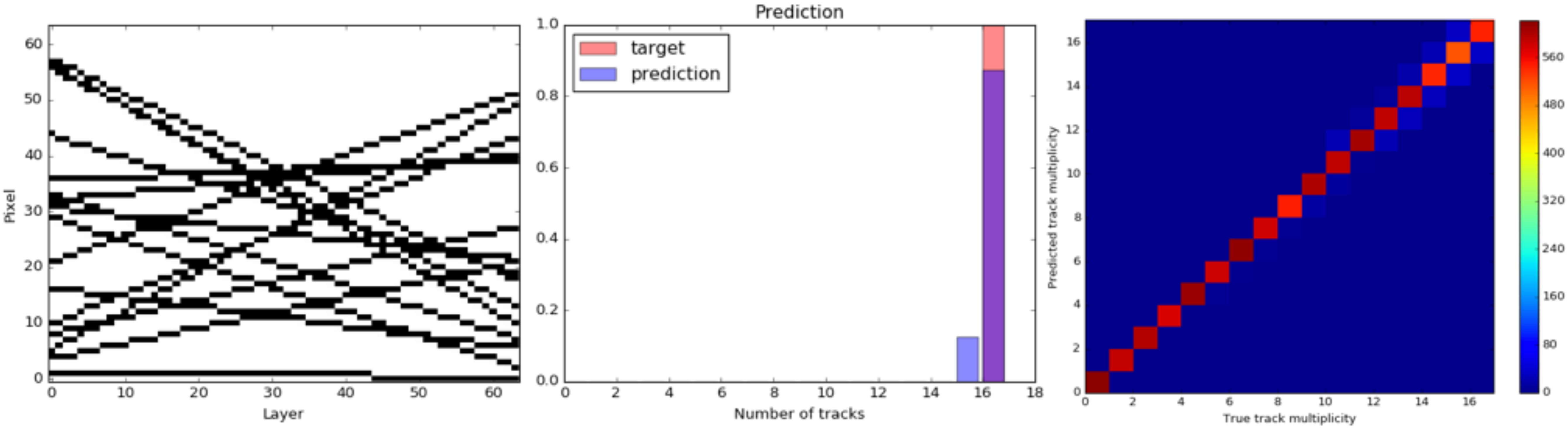
3D toy detector data



- Starting to get a little more "realistic"
  - 10 detector planes, 32x32 pixels each
  - Number of background tracks sampled from Poisson
  - With/without random noise hits
- Adapting my existing models to this data is mostly straightforward
  - Flatten each plane for the LSTM models
  - Use 3D convolution

What can CNNs learn about tracks?

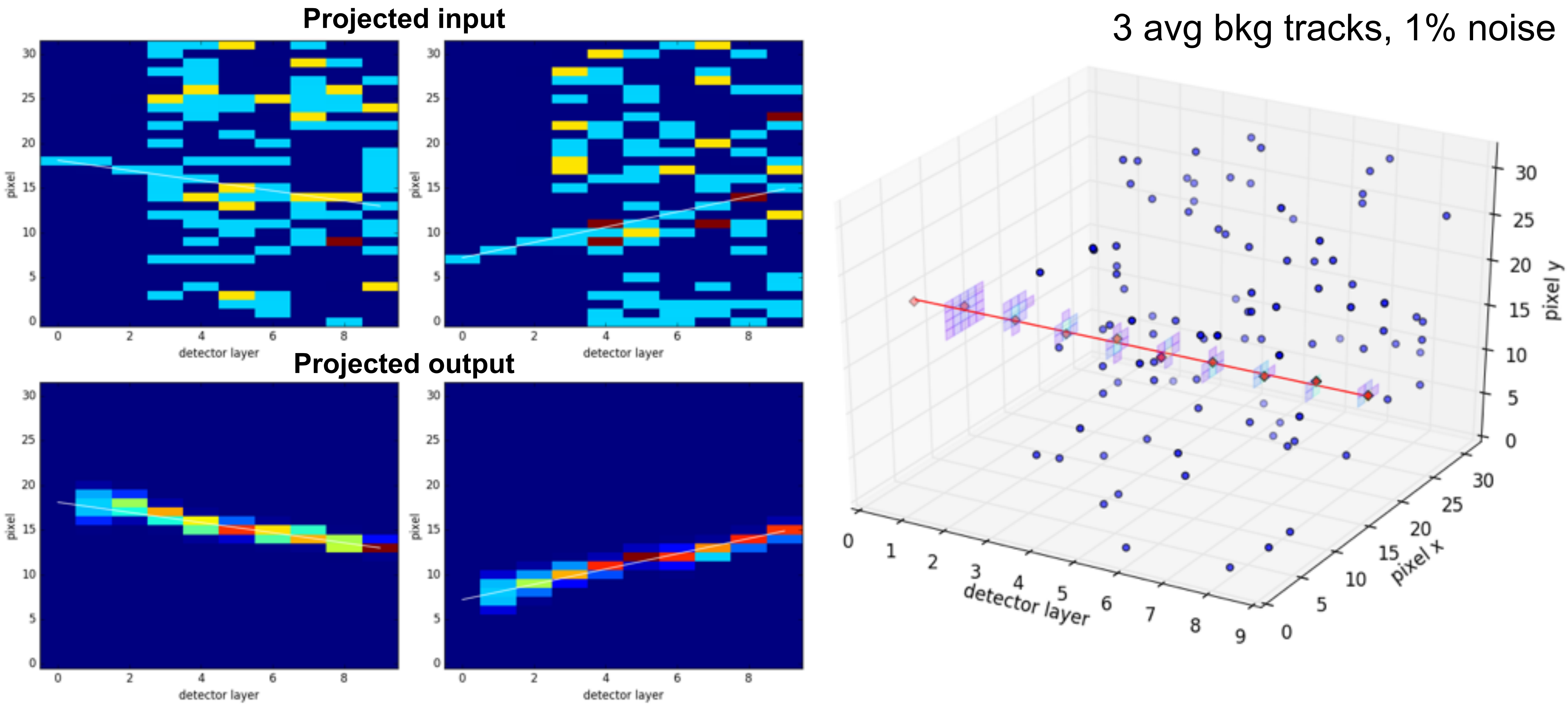
- **Track counting**: can it predict how many tracks are in an event?
  - can be framed as a regression problem, but here I framed it as a *classification* problem



  - seemingly not a very difficult task for a deep NN

https://github.com/HEPTrkX/heptrkx-dshep17/blob/master/cnn/cnn2d_learning.ipynb

Next-layer LSTM prediction



- 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

The HEP.TrkX project

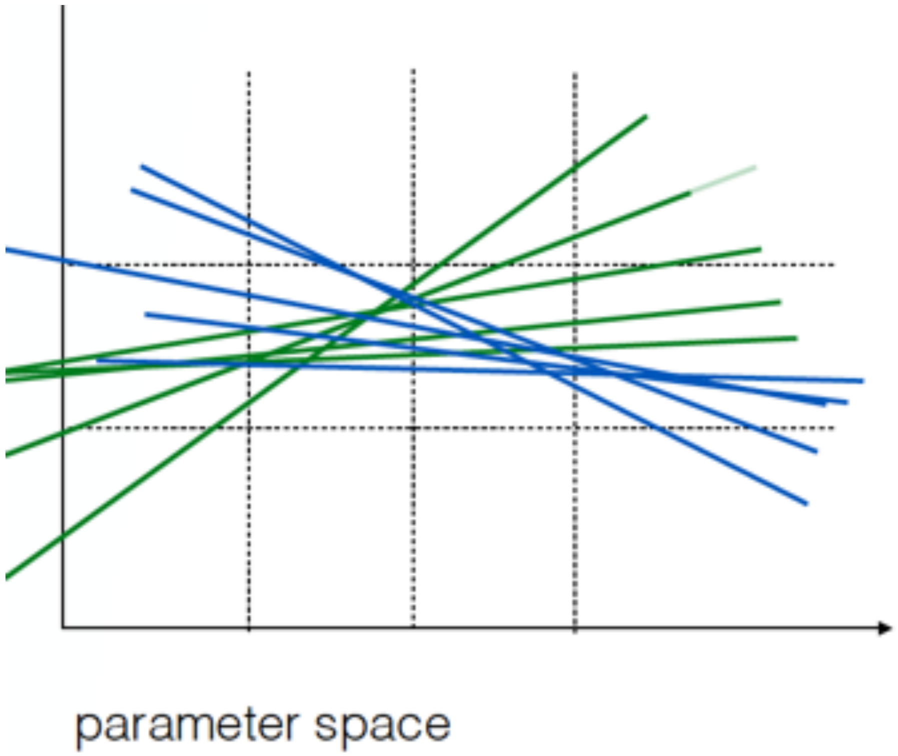
- **A 1-year pilot project to develop ML algorithms for HEP tracking**
  - Funded by DOE ASCR and COMP HEP, part of HEP CCE
  - Collaboration between ATLAS, CMS, LAr folks from LBL, Caltech, and FNAL

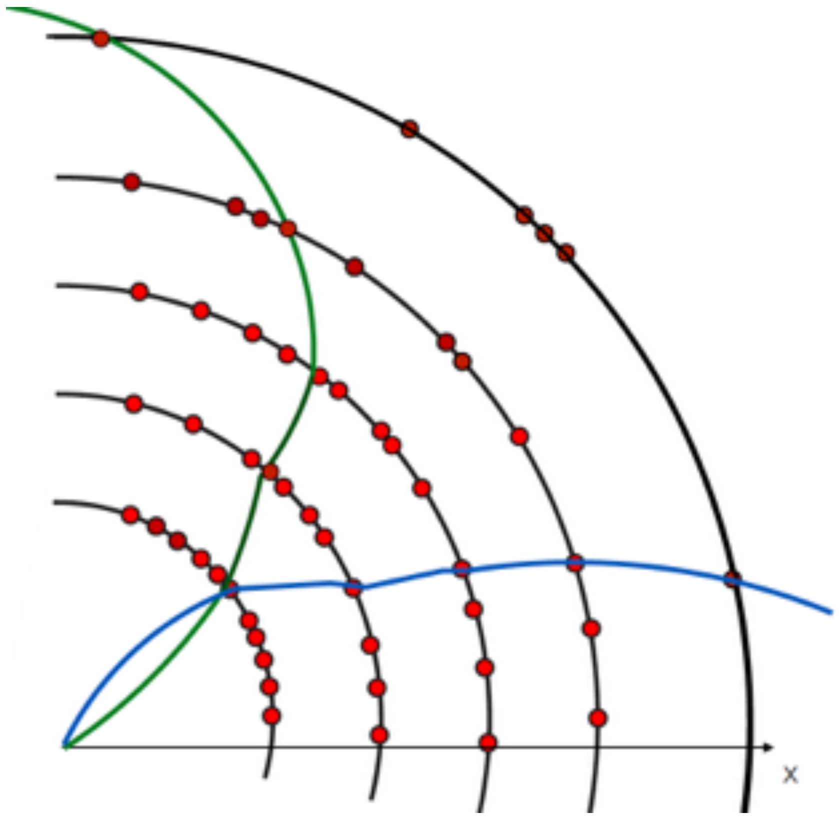
    LBL: Me, Mayur Mudigonda, Prabhat, Paolo
    Caltech: Dustin Anderson, Jean-Roch Vlimant, Josh Bendavid, Maria Spiropoulou, Stephan Zheng
    FNAL: Aristeidis Tsaris, Giuseppe Cerati, Jim Kowalkowski, Lindsey Gray, Panagiotis Spentzouris

- **Some goals**
  - Explore the broad space of ideas on simplified tracking problems
  - Develop a toolkit of promising ideas
    - ideas that work (physics constraints)
    - ideas that *scale* (computing constraints)
- **The work is in an *exploratory phase***
  - Testing ideas in a breadth-first fashion
  - Very much a work-in-progress

Other ideas - data transforms

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

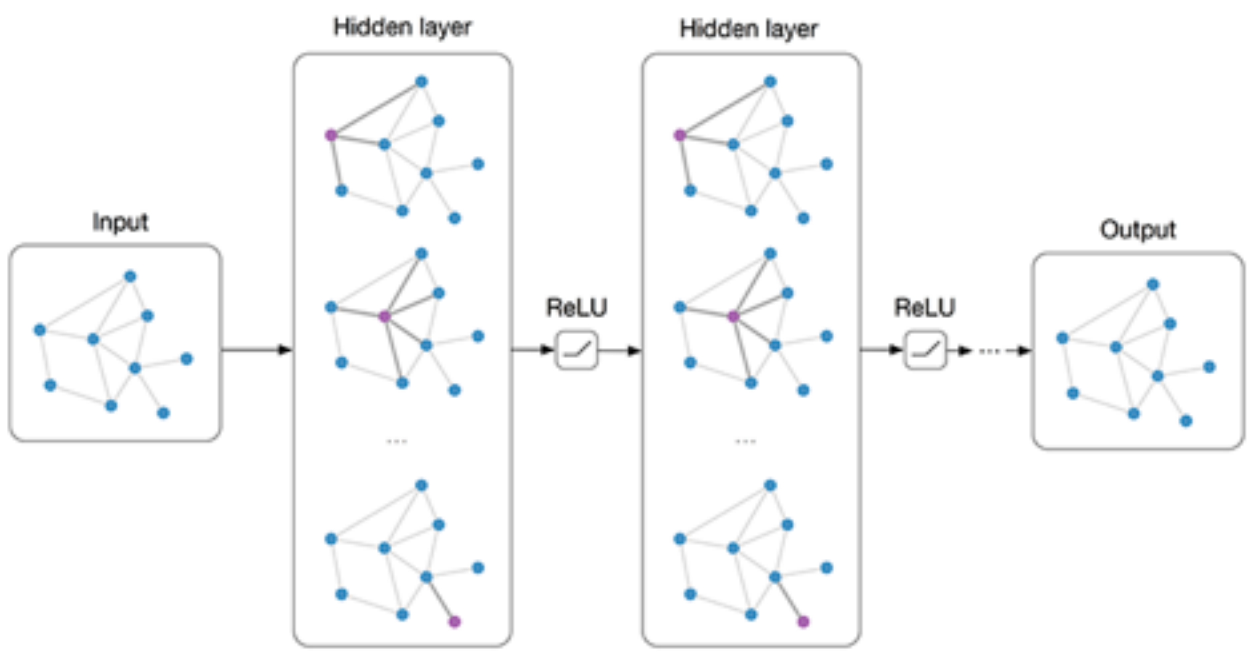




- But what if a deep network could *learn* a mapping to group together hits that belong to the same track?
  - You don’t need to impose a specific representation
  - The model could take event context into account

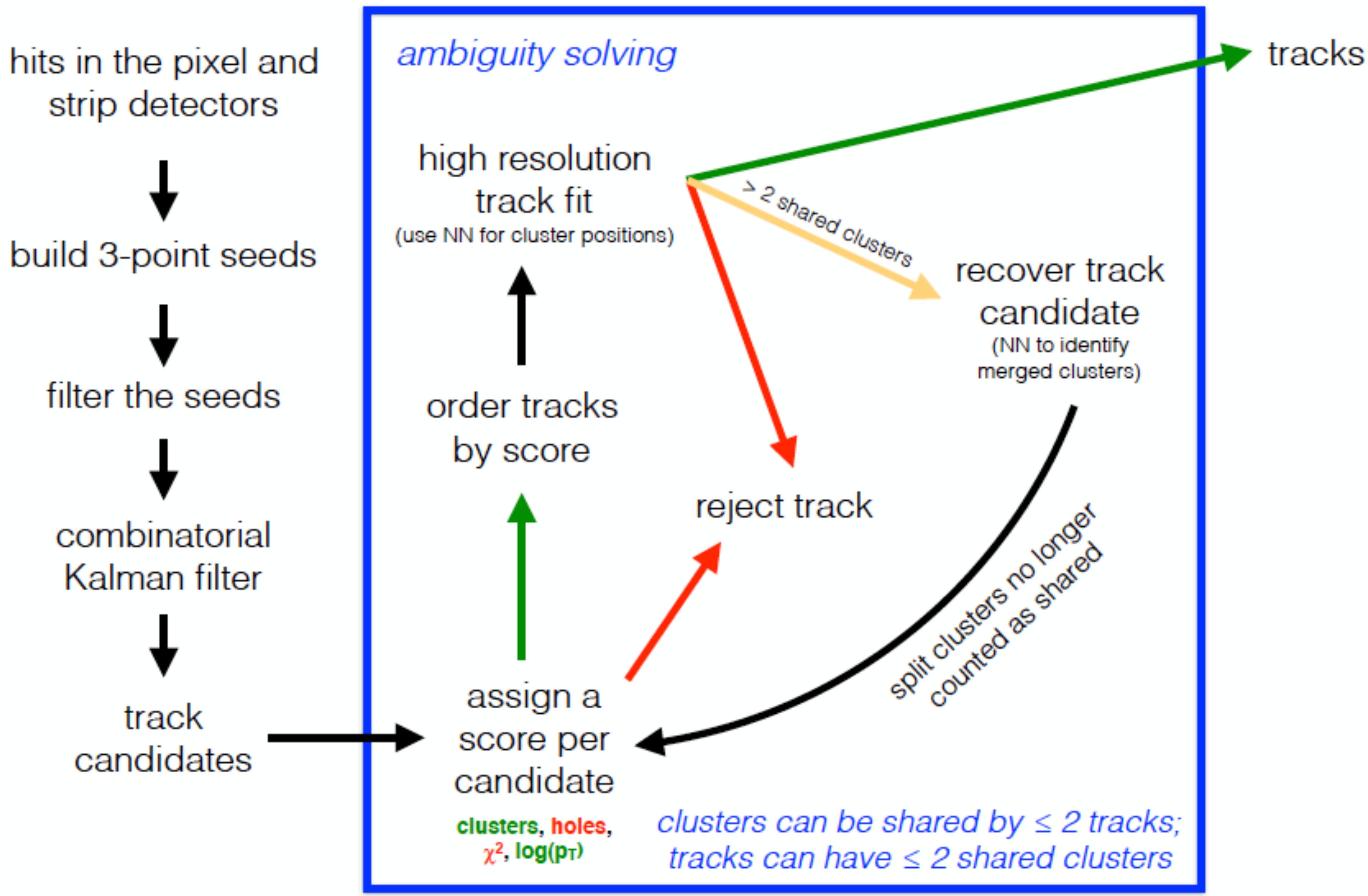
Other ideas - graph convolutions

- 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
- 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

ATLAS tracking in dense environments



Stolen from Ben Nachman’s TPM presentation:
https://indico.physics.lbl.gov/indico/event/433/

Model architectures - ConvNN

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 10, 32, 32)	0	
reshape_1 (Reshape)	(None, 1, 10, 32, 32)	0	input_1[0][0]
convolution3d_1 (Convolution3D)	(None, 8, 10, 32, 32)	224	reshape_1[0][0]
convolution3d_2 (Convolution3D)	(None, 8, 10, 32, 32)	1736	convolution3d_1[0][0]
convolution3d_3 (Convolution3D)	(None, 8, 10, 32, 32)	1736	convolution3d_2[0][0]
convolution3d_4 (Convolution3D)	(None, 8, 10, 32, 32)	1736	convolution3d_3[0][0]
convolution3d_5 (Convolution3D)	(None, 8, 10, 32, 32)	1736	convolution3d_4[0][0]
convolution3d_6 (Convolution3D)	(None, 8, 10, 32, 32)	1736	convolution3d_5[0][0]
convolution3d_7 (Convolution3D)	(None, 8, 10, 32, 32)	1736	convolution3d_6[0][0]
convolution3d_8 (Convolution3D)	(None, 8, 10, 32, 32)	1736	convolution3d_7[0][0]
convolution3d_9 (Convolution3D)	(None, 8, 10, 32, 32)	1736	convolution3d_8[0][0]
convolution3d_10 (Convolution3D)	(None, 8, 10, 32, 32)	1736	convolution3d_9[0][0]
convolution3d_11 (Convolution3D)	(None, 1, 10, 32, 32)	217	convolution3d_10[0][0]
reshape_2 (Reshape)	(None, 10, 1024)	0	convolution3d_11[0][0]
timedistributed_1 (TimeDistribute	(None, 10, 1024)	0	reshape_2[0][0]

Total params: 16065

Model architectures - Conv autoencoder

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 10, 32, 32)	0	
reshape_1 (Reshape)	(None, 1, 10, 32, 32)	0	input_1[0][0]
convolution3d_1 (Convolution3D)	(None, 8, 10, 32, 32)	224	reshape_1[0][0]
convolution3d_2 (Convolution3D)	(None, 8, 10, 32, 32)	1736	convolution3d_1[0][0]
maxpooling3d_1 (MaxPooling3D)	(None, 8, 10, 16, 16)	0	convolution3d_2[0][0]
dropout_1 (Dropout)	(None, 8, 10, 16, 16)	0	maxpooling3d_1[0][0]
convolution3d_3 (Convolution3D)	(None, 16, 10, 16, 16)	3472	dropout_1[0][0]
convolution3d_4 (Convolution3D)	(None, 16, 10, 16, 16)	6928	convolution3d_3[0][0]
maxpooling3d_2 (MaxPooling3D)	(None, 16, 10, 8, 8)	0	convolution3d_4[0][0]
dropout_2 (Dropout)	(None, 16, 10, 8, 8)	0	maxpooling3d_2[0][0]
convolution3d_5 (Convolution3D)	(None, 32, 10, 8, 8)	13856	dropout_2[0][0]
maxpooling3d_3 (MaxPooling3D)	(None, 32, 10, 4, 4)	0	convolution3d_5[0][0]
dropout_3 (Dropout)	(None, 32, 10, 4, 4)	0	maxpooling3d_3[0][0]
convolution3d_6 (Convolution3D)	(None, 64, 10, 4, 4)	55360	dropout_3[0][0]
maxpooling3d_4 (MaxPooling3D)	(None, 64, 10, 2, 2)	0	convolution3d_6[0][0]
dropout_4 (Dropout)	(None, 64, 10, 2, 2)	0	maxpooling3d_4[0][0]
convolution3d_7 (Convolution3D)	(None, 96, 10, 2, 2)	73824	dropout_4[0][0]
maxpooling3d_5 (MaxPooling3D)	(None, 96, 10, 1, 1)	0	convolution3d_7[0][0]
dropout_5 (Dropout)	(None, 96, 10, 1, 1)	0	maxpooling3d_5[0][0]
convolution3d_8 (Convolution3D)	(None, 128, 10, 1, 1)	36992	dropout_5[0][0]
permute_1 (Permute)	(None, 10, 128, 1, 1)	0	convolution3d_8[0][0]
reshape_2 (Reshape)	(None, 10, 128)	0	permute_1[0][0]
timedistributed_1 (TimeDistribute	(None, 10, 1024)	132096	reshape_2[0][0]

Total params: 324488

Model architectures - LSTM

```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
input_1 (InputLayer)             (None, 9, 1024)       0
____________________________________________________________________________________________________
lstm_1 (LSTM)                    (None, 9, 1024)       8392704     input_1[0][0]
____________________________________________________________________________________________________
timedistributed_1 (TimeDistribute(None, 9, 1024)       1049600     lstm_1[0][0]
====================================================================================================
Total params: 9442304
____________________________________________________________________________________________________
```