Caltech

The **HEP.TrkX** Project:

Deep Learning for Particle Tracking

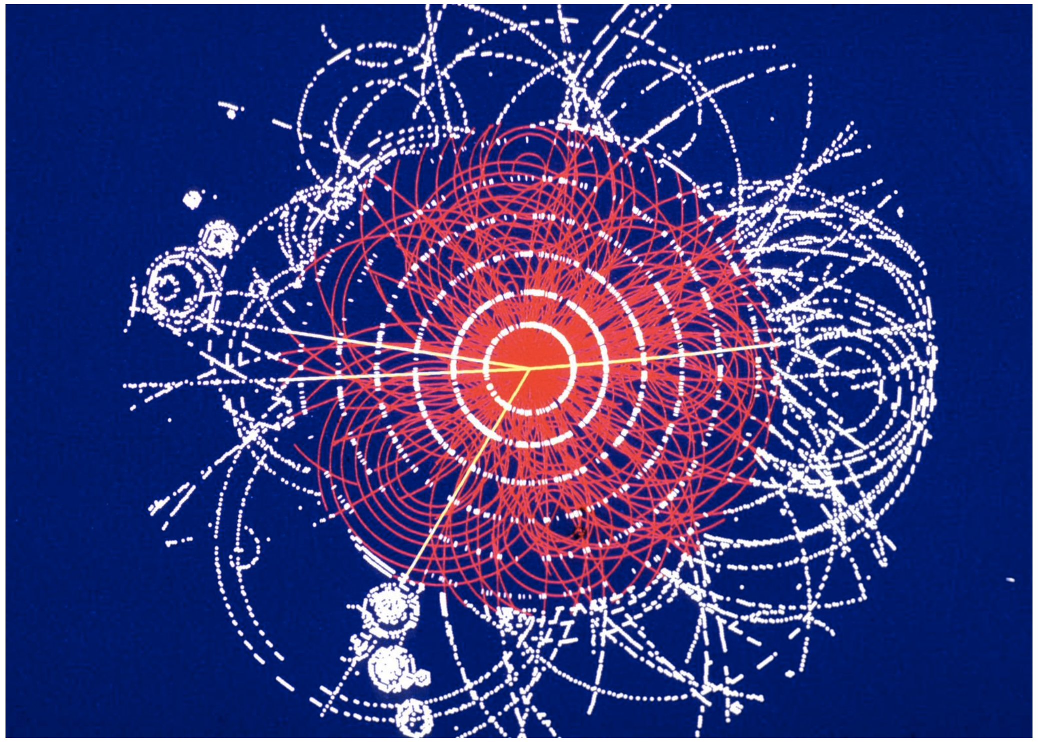


Image: CERN

Dustin Anderson

for the HEP.TrkX Collaboration

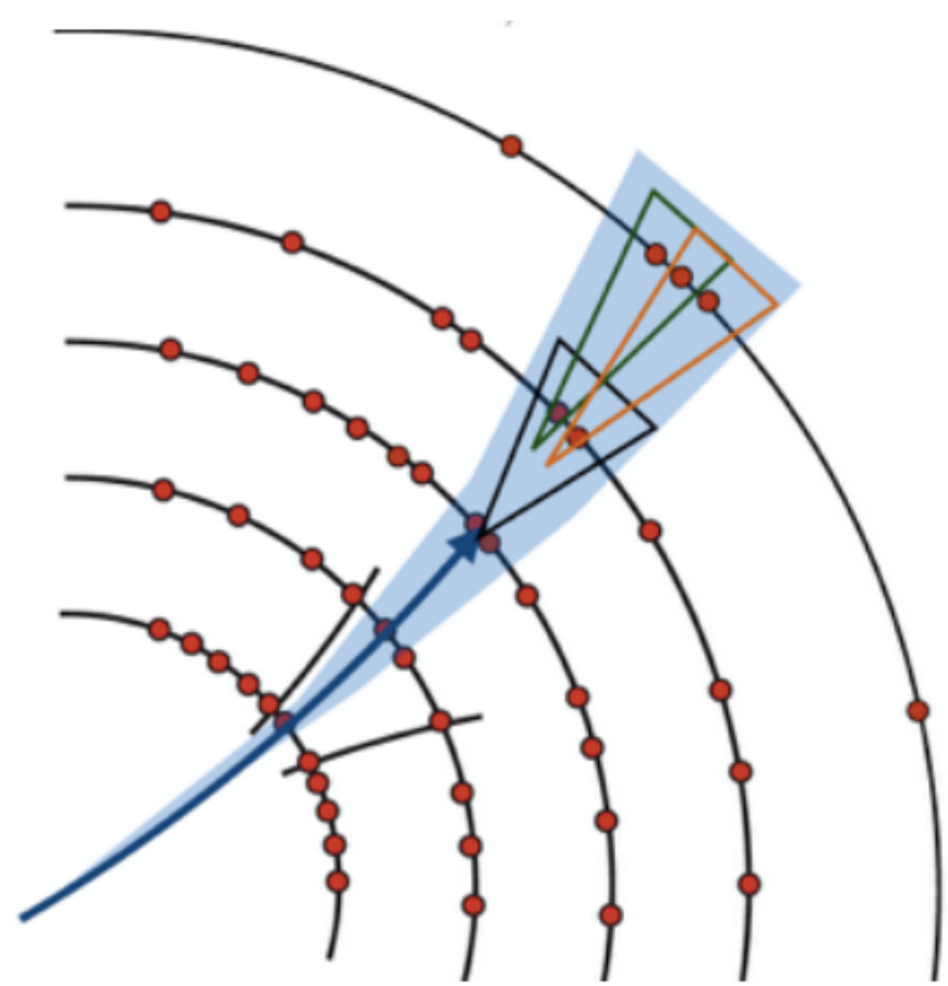
BERKELEY LAB

IML Workshop

22 March 2017

Tracking at the LHC

- LHC particle tracking algorithms have seen great success in Runs I and II. They use a two-part strategy:
  - **Track seeding** using combinatorial search
    - ➡ Complexity: $O(N^3)$
  - **Track candidate formation** using Kalman filter
    - ➡ Complexity: $O(N^2)$ - $O(N^3)$



Credit: Andy Salzburger

Tracking at the HL-LHC

- High-Luminosity upgrade of the LHC:
  - ➡ Luminosity **x 10**
  - ➡ Number of track hits **x 10**
- Up to 200 collision events per bunch crossing in CMS and ATLAS detectors

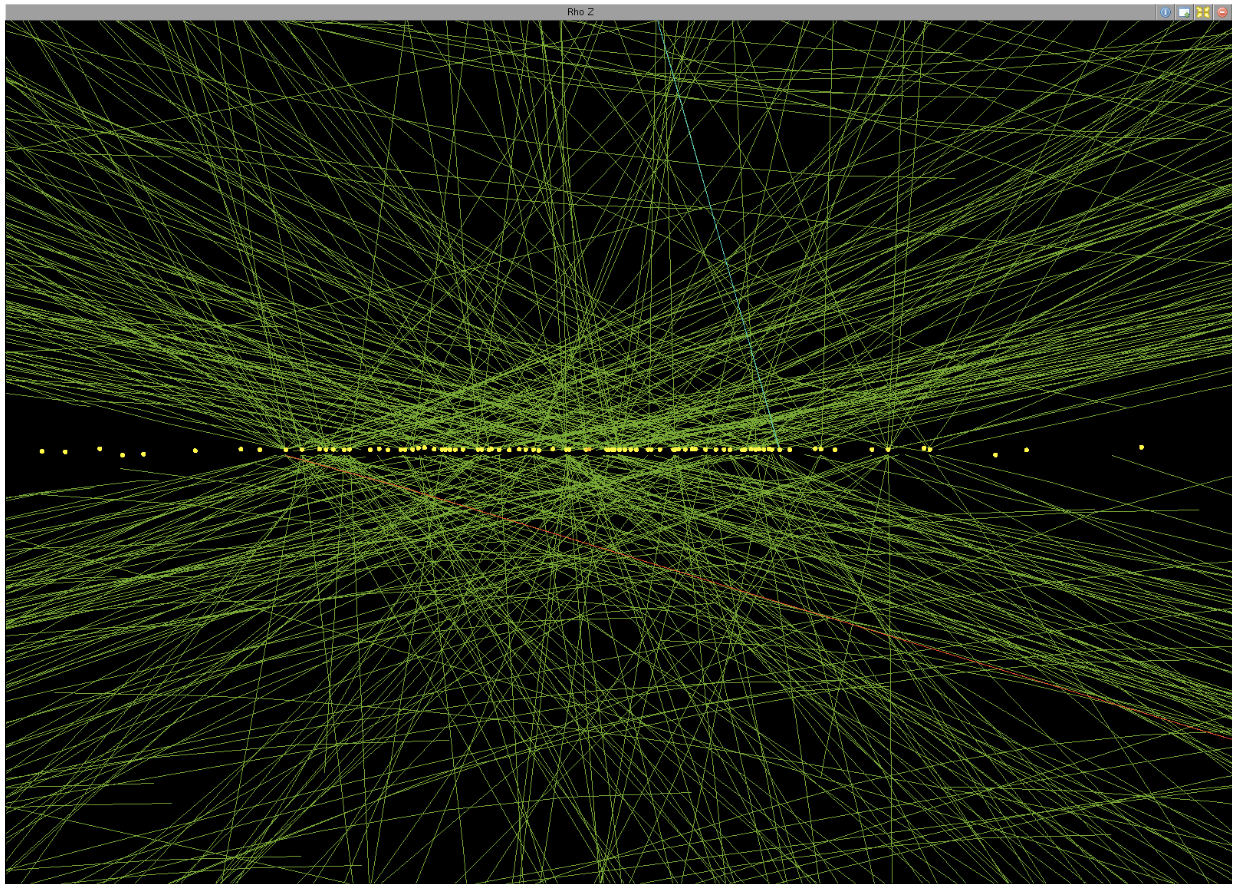
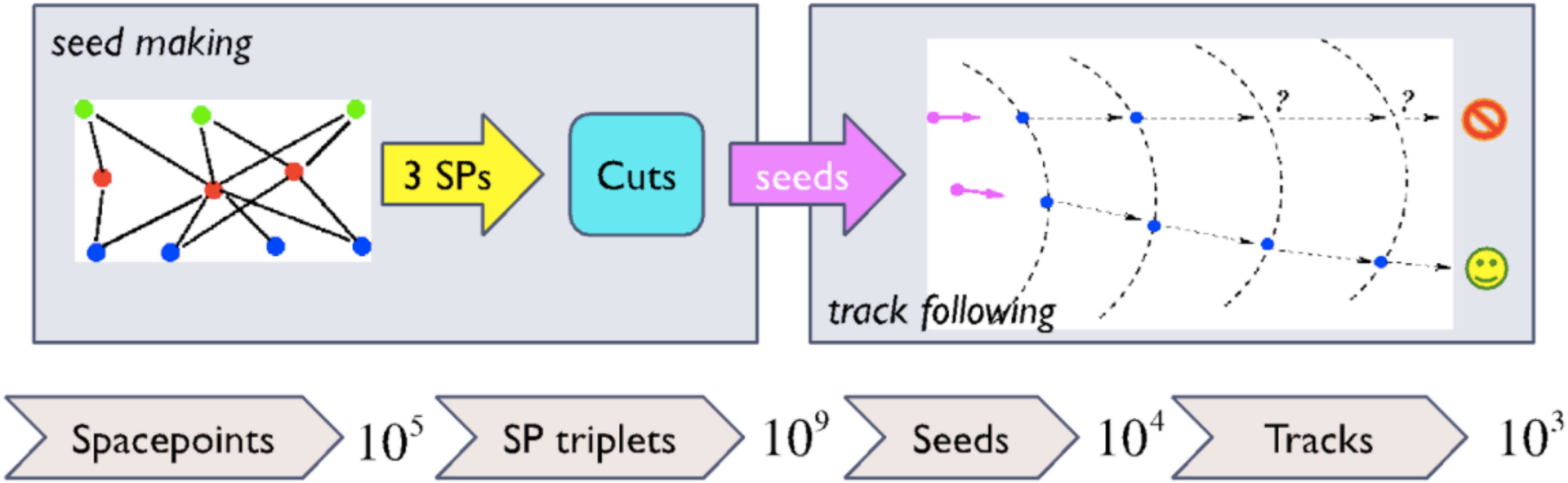


Image: CERN

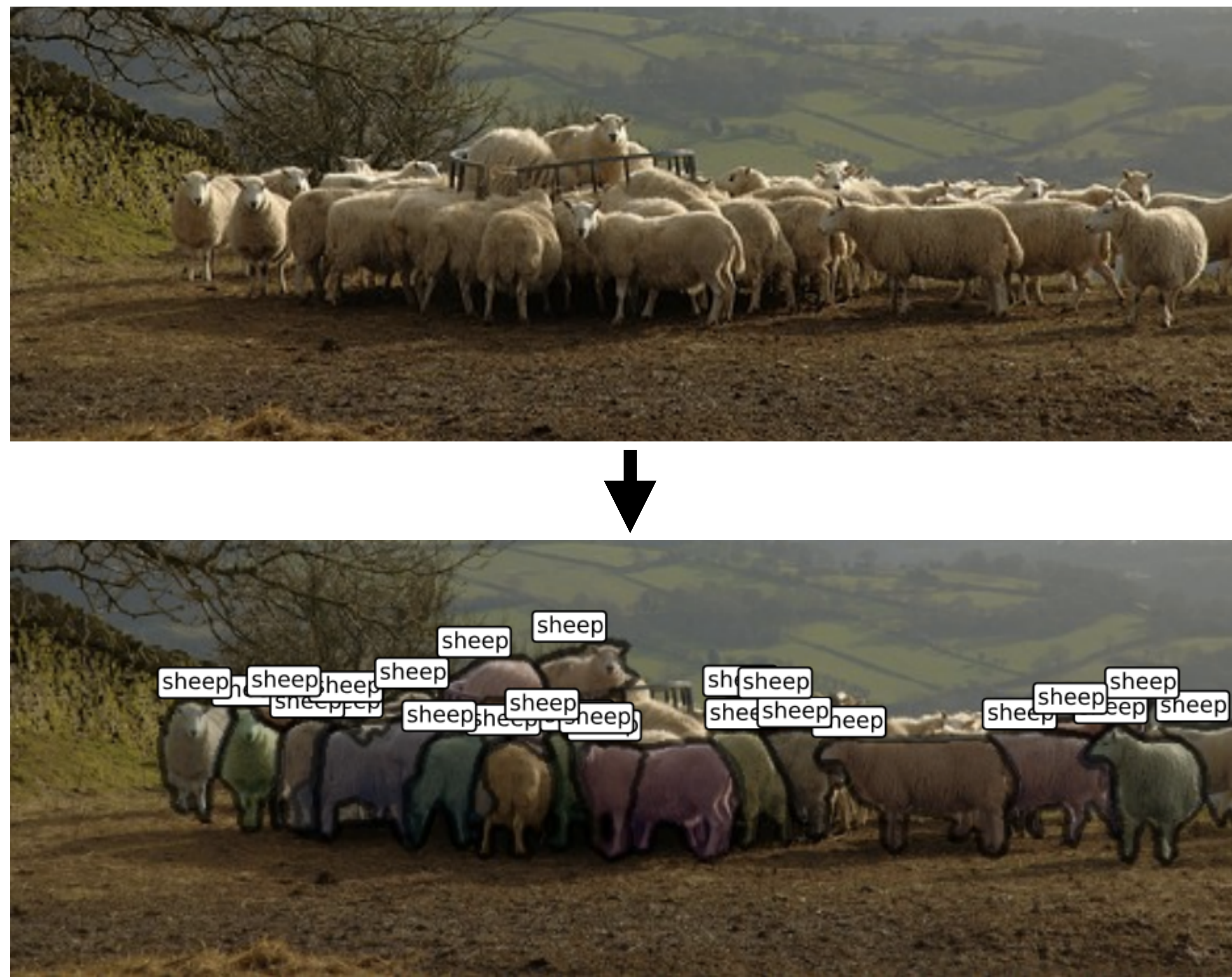
Tracking at the HL-LHC



- Upgraded LHC detectors will have O(**100M**) tracker readout channels
- O(**5000**) charged particles per event → O(**10^5**) 3-D position measurements
- **Scaling of existing track algorithms is not favorable**

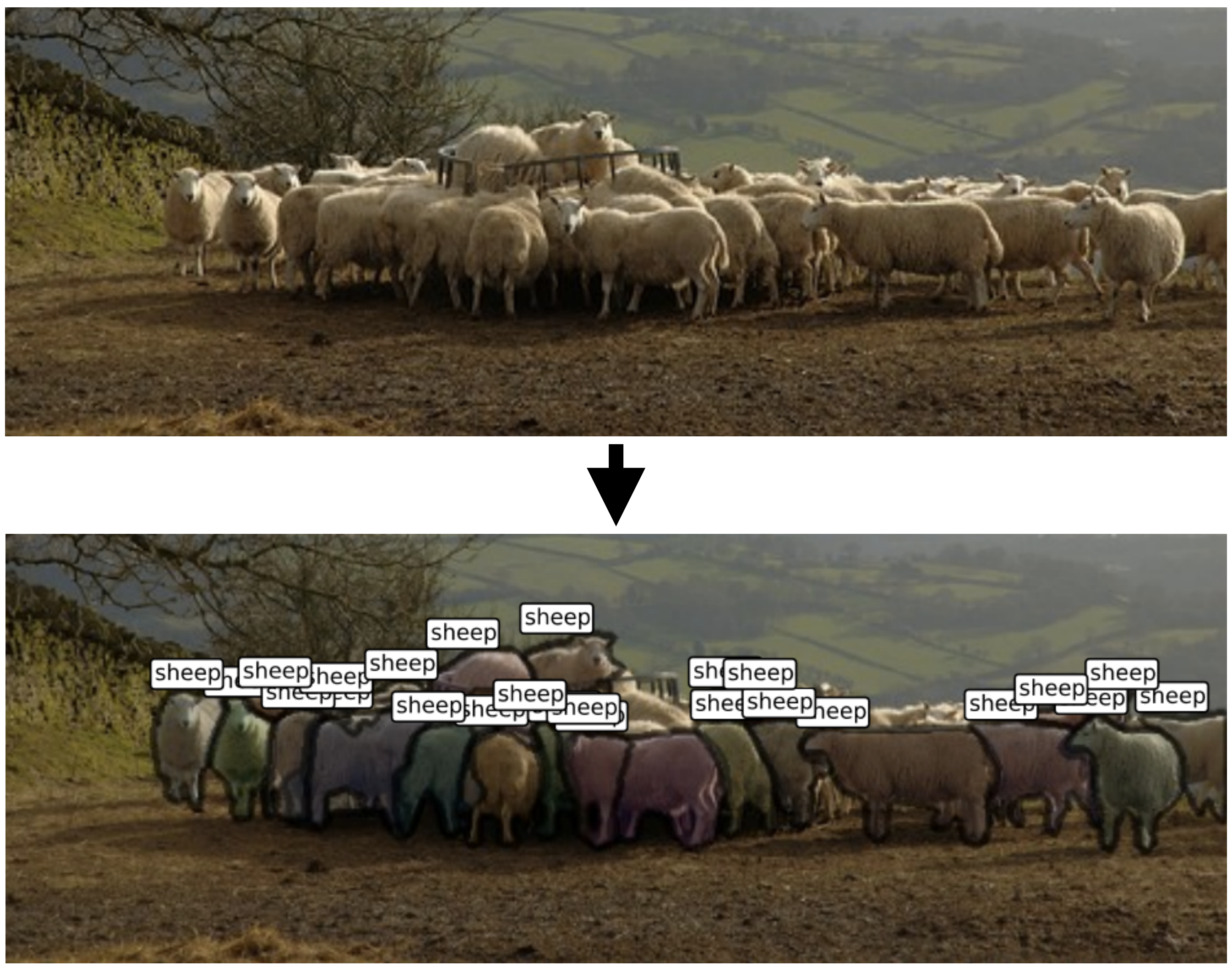
Deep Learning

- Track finding is similar to problems on which **deep neural networks** have seen success:
  - Image captioning
  - Sequence prediction
  - Scene labeling/ partitioning



Zagoruyko et al, https://arxiv.org/pdf/1604.02135.pdf

Deep Learning



Zagoruyko et al, https://arxiv.org/pdf/1604.02135.pdf

Our goal (more or less…):

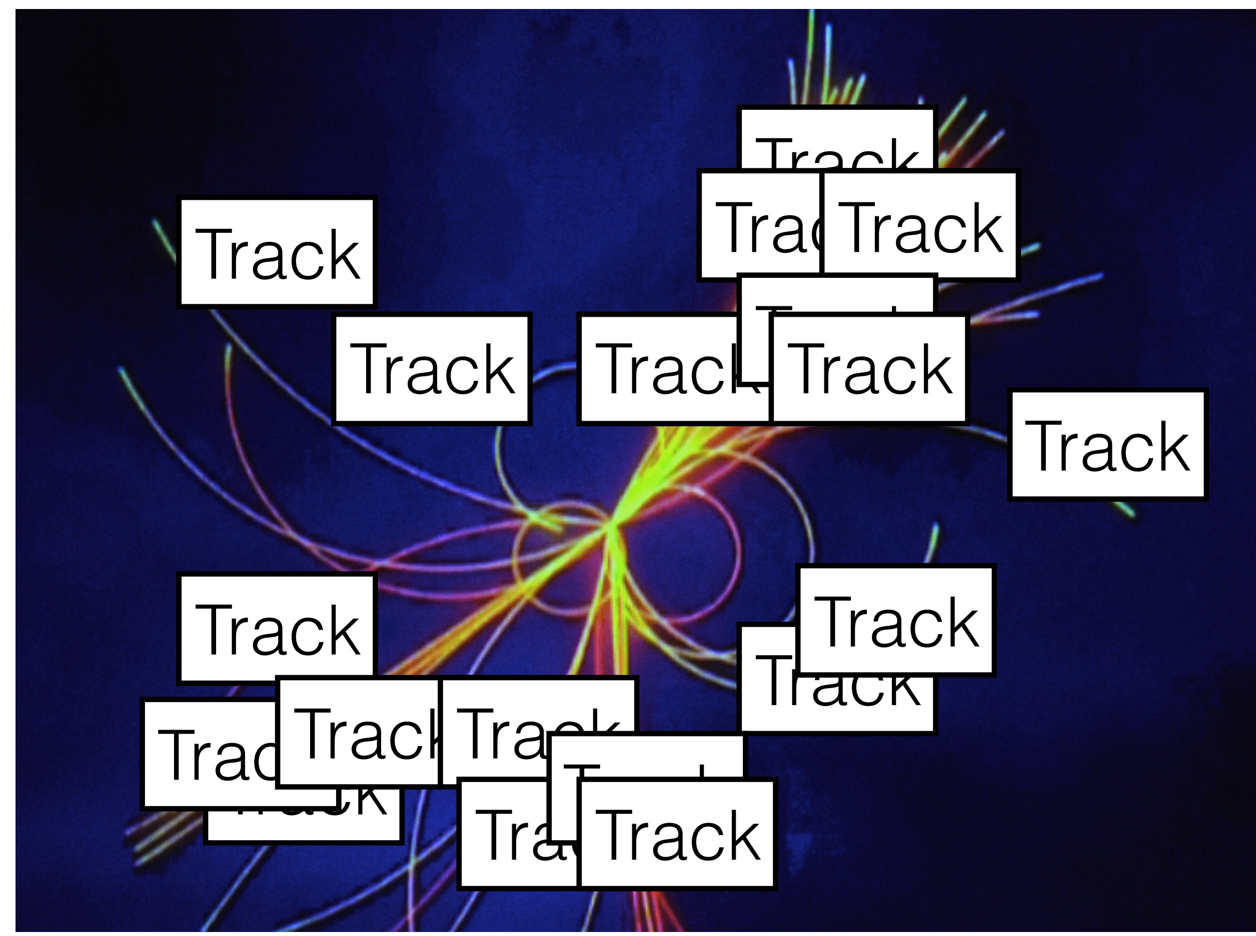


Photo by Pier Marco Tacca/Getty Images

HEP.TrkX Project

- **HEP.TrkX**: a one-year pilot project within the DOE HEP Center for Computational Excellence
- **Goals**:
  - ➡ Explore and develop new tracking algorithms based on modern ML techniques
  - ➡ Demonstrate a scalable algorithm with the potential to reconstruct tracks in HL-LHC conditions

HEP.TrkX Project

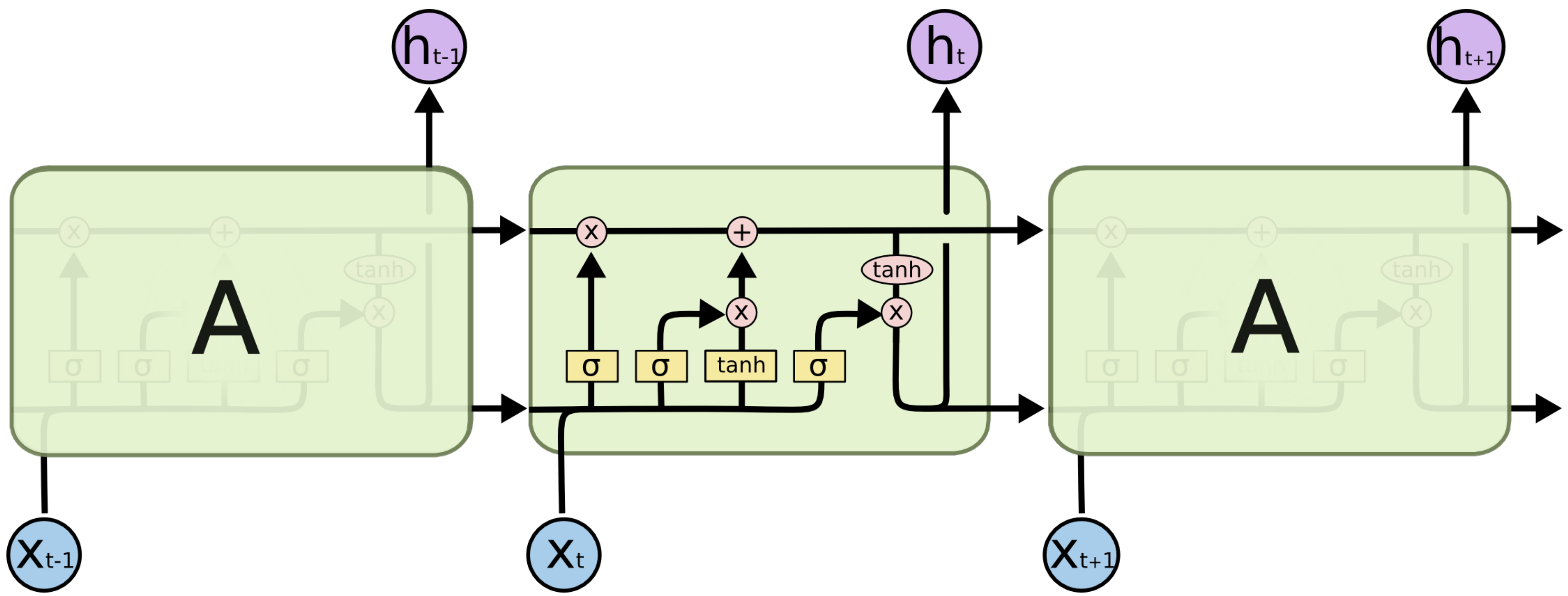
- Our collaboration:
  - ➡ **Caltech** : Dustin Anderson, Josh Bendavid, Maria Spiropulu, Jean-Roch Vlimant, Stephan Zheng
  - ➡ **Fermilab** : Giuseppe Cerati, Lindsey Gray, Jim Kowalkowski, Panagiotis Spentzouris, Aristeidis Tsaris
  - ➡ **LBL** : Paolo Calafiura, Steve Farrell, Mayur Mudigonda, Prabhat

Exploring the Space of Ideas

- **Track Extension** — replace Kalman Filter with a smarter or faster iterative algorithm
- **Seed Finding** — improve on N^3 scaling of current algorithms
- **End-To-End Methods** — cluster hits directly into tracks or produce values for track parameters

Track Extension Algorithms

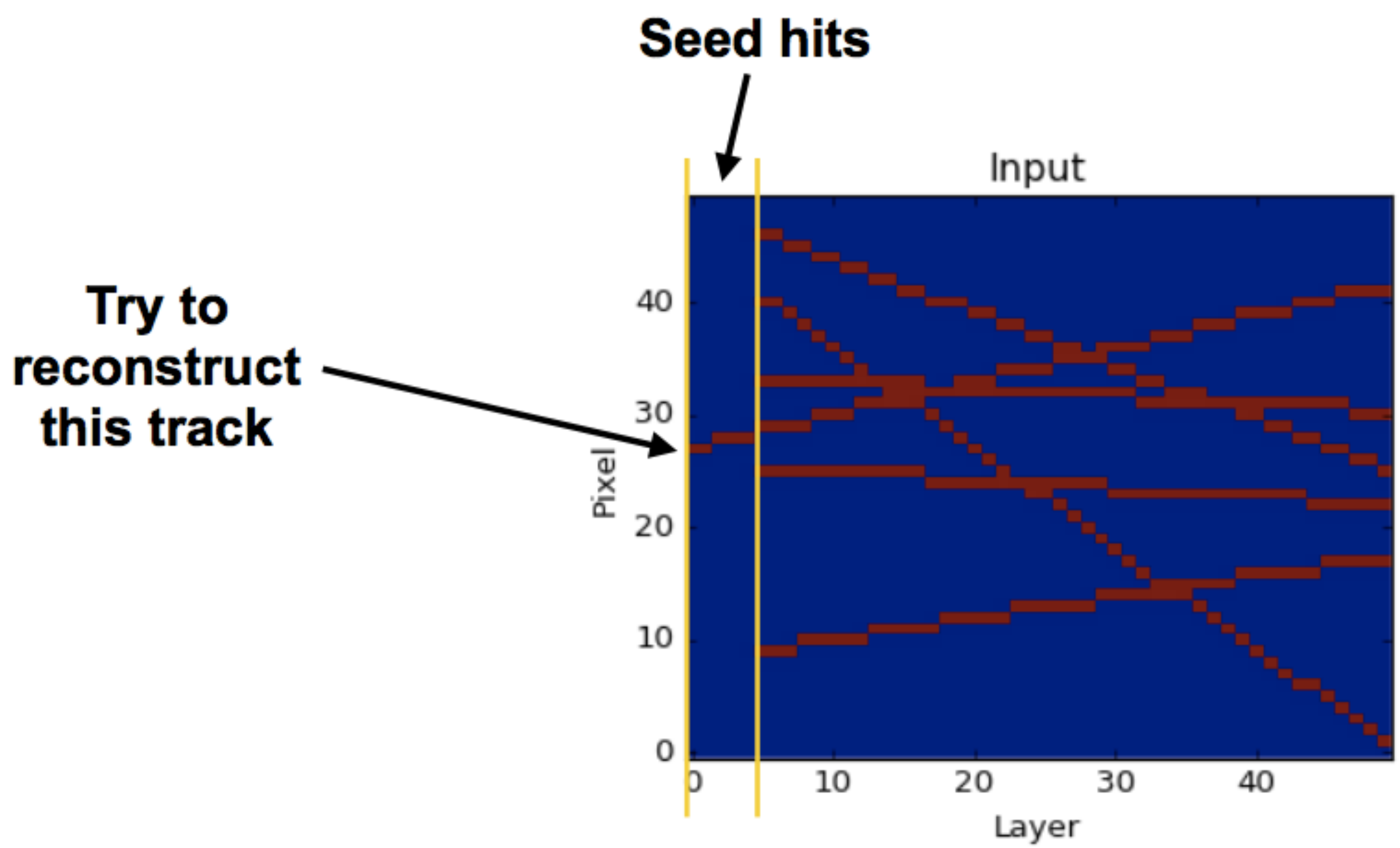
- Long-Short-Term Memory (**LSTM**) recurrent neural networks:
  - ➡ Produce a sequence of outputs, like Kalman Filter does
  - ➡ Have a state update equation learned from training data
- An LSTM-based track extension algorithm could alleviate the combinatorial scaling problem present in current KF algorithms



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

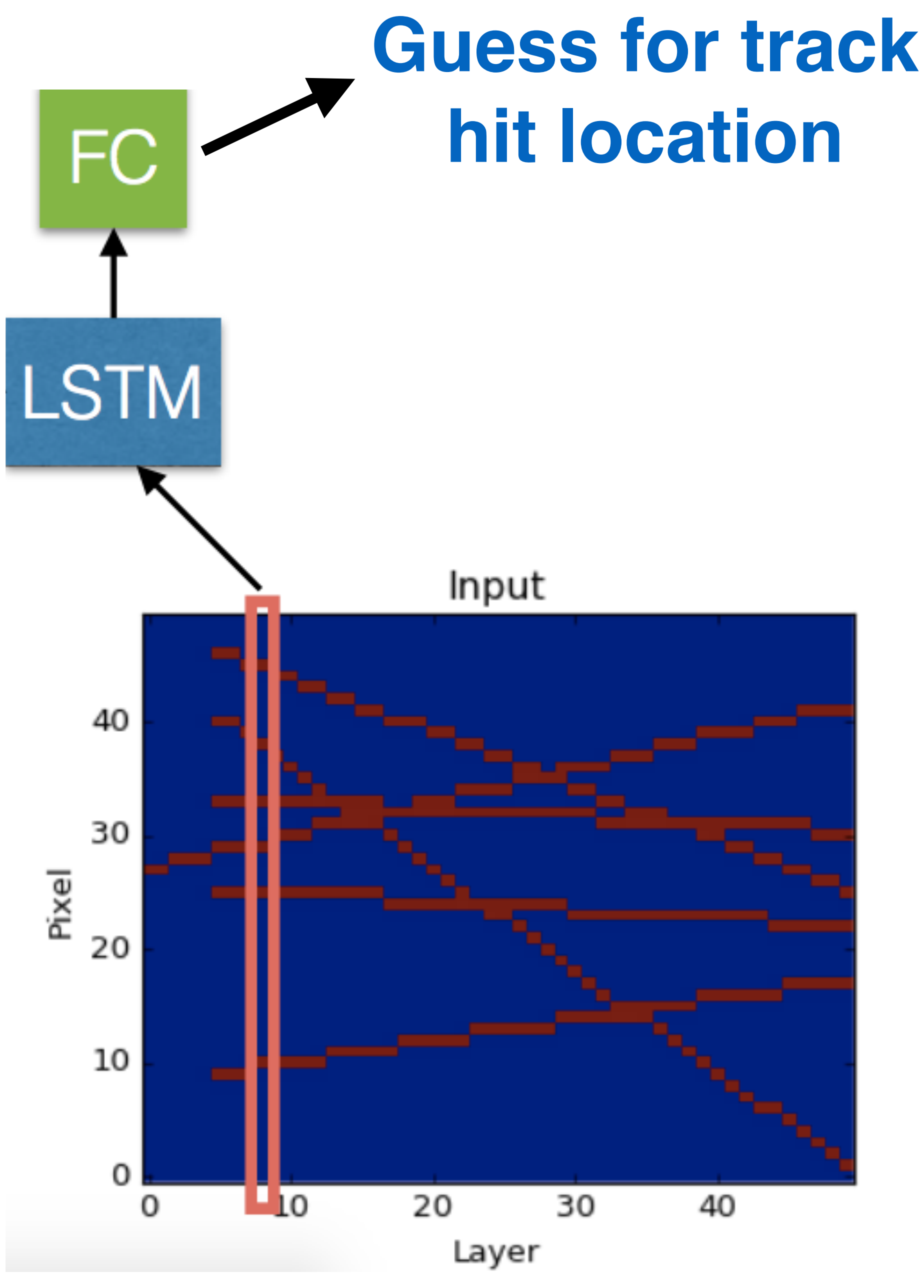
Track Extension with LSTM

- Proof-of-concept on toy detector data
  - Square, 2-D detector
  - Straight-line tracks
  - No missing hits



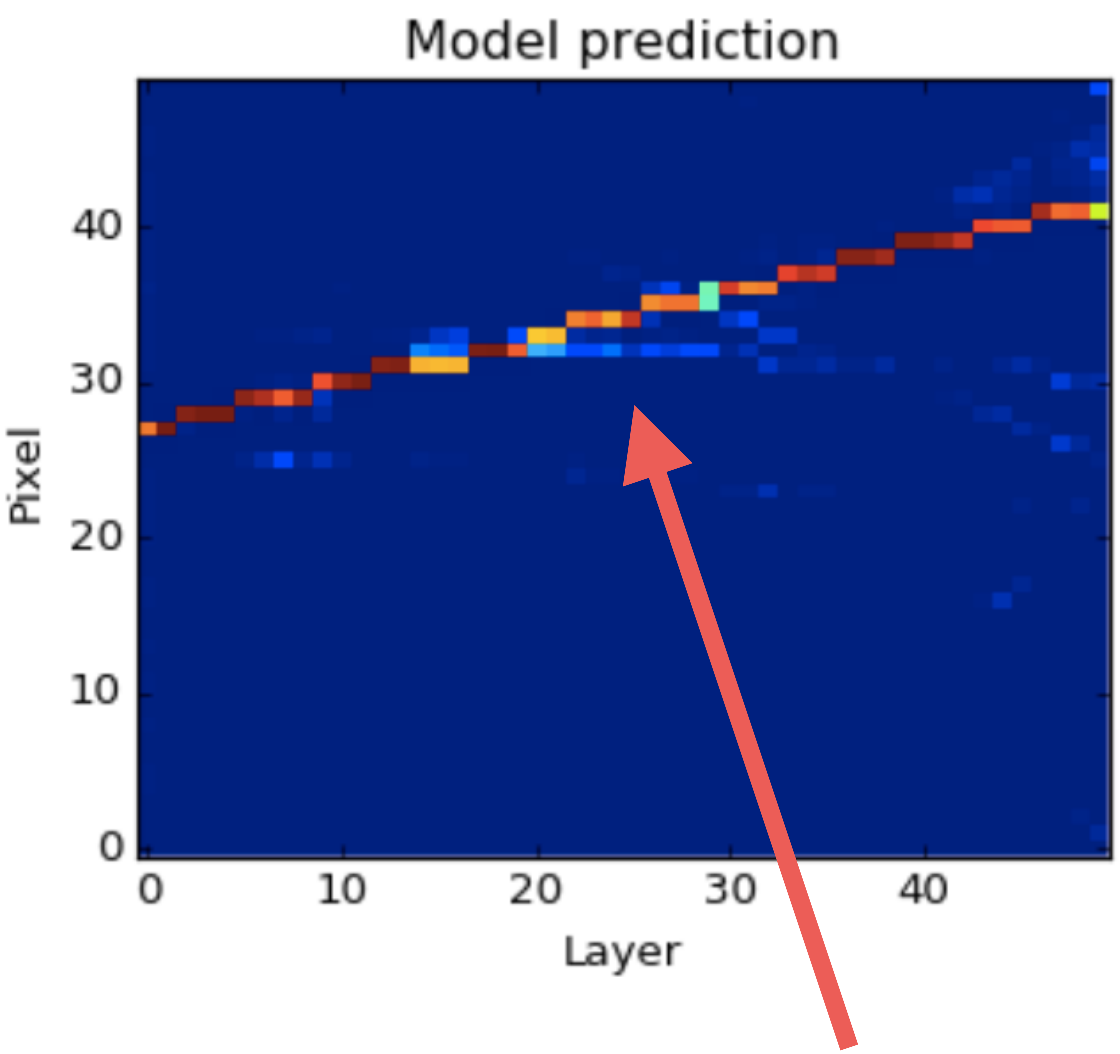
Track Extension with LSTM

- Starting from a seed, the model builds the track **iteratively**
- At each step, it considers a slice of the detector
- It outputs a **probabilistic estimate** of the track hit location in the current slice



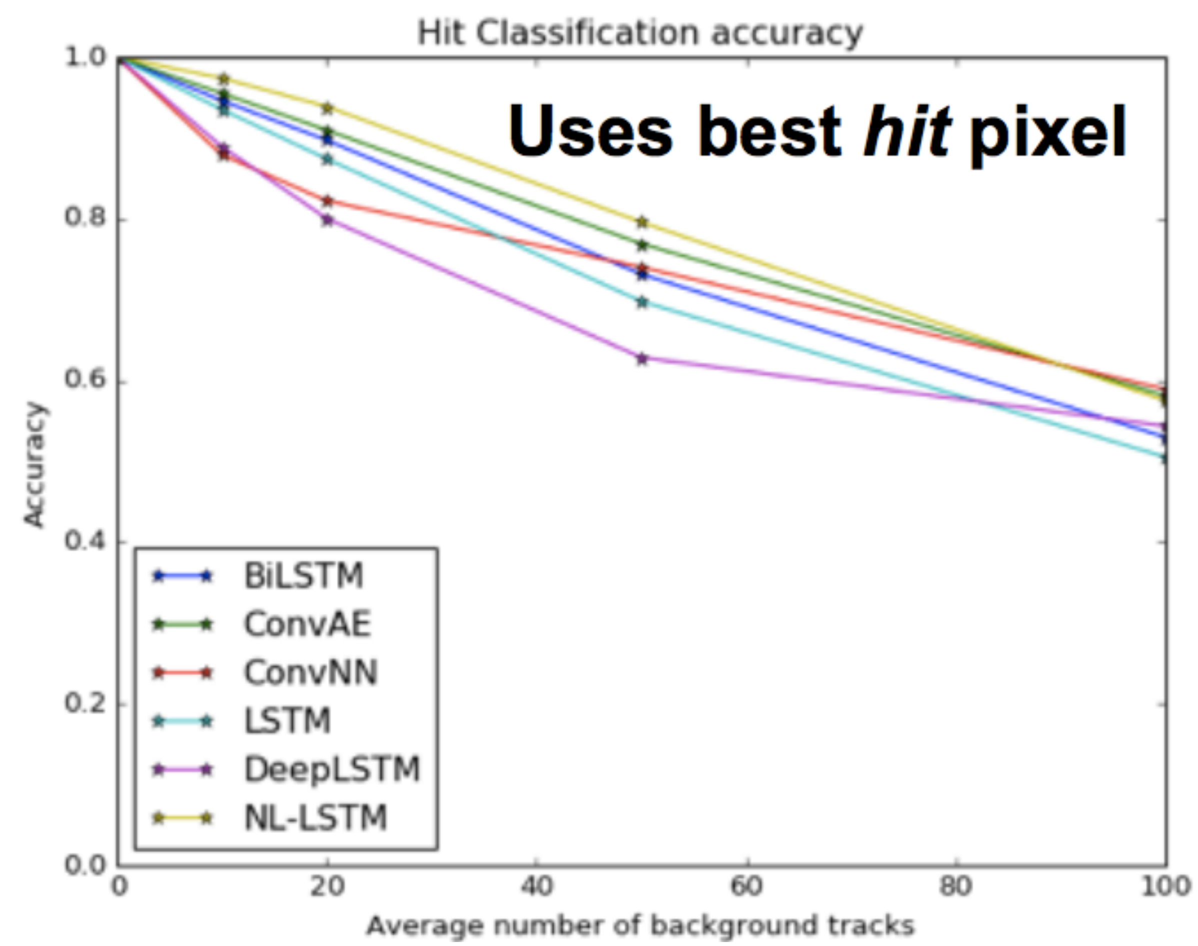
Track Extension with LSTM

- Repeat for each detector slice to obtain full prediction
- The LSTM memory state propagates relevant information from layer to layer



Track Extension with LSTM

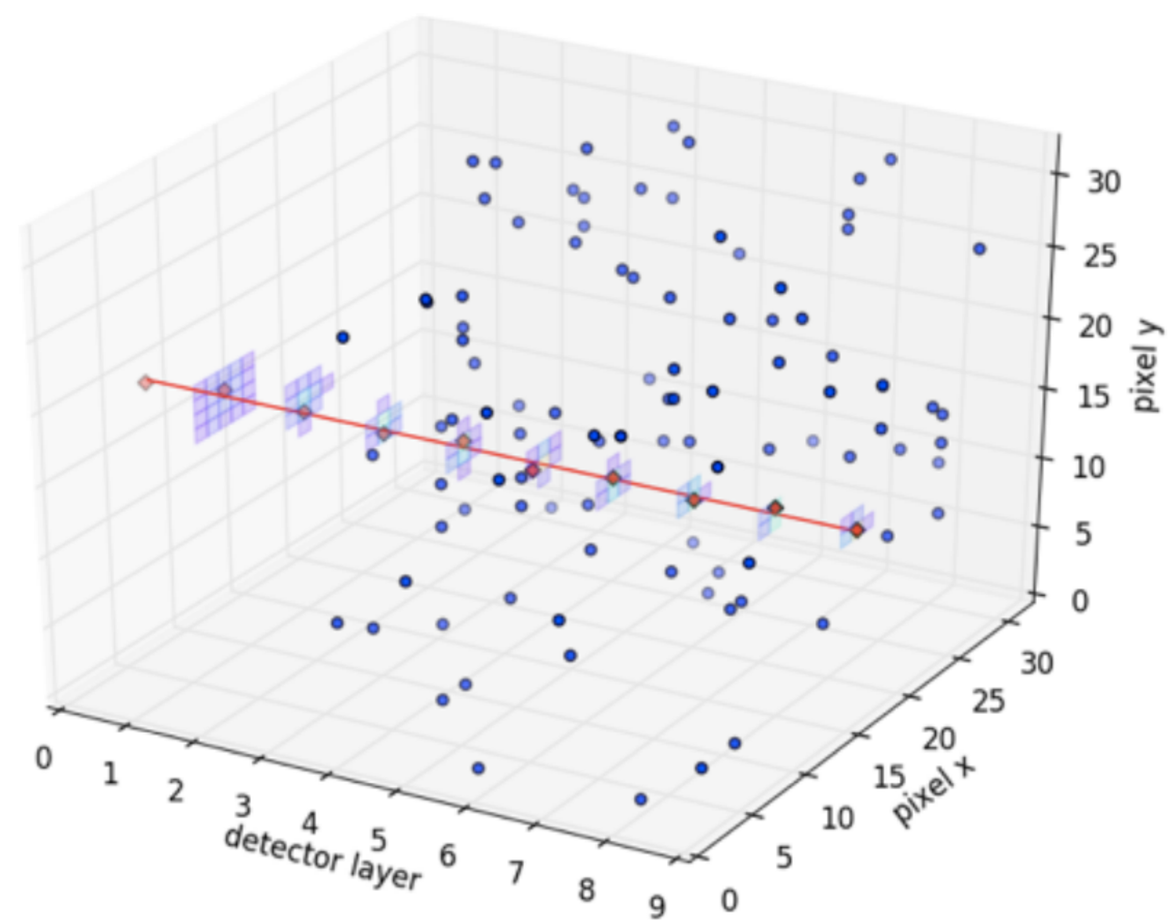
- Variations on this model:
  - ➡ **Deep architecture** with more layers
  - ➡ **Bi-directional LSTM** running forward and backward simultaneously
  - ➡ **Convolutional autoencoder** instead of LSTM for layer-wise prediction



Performance comparison for different architectures

Track Extension with LSTM

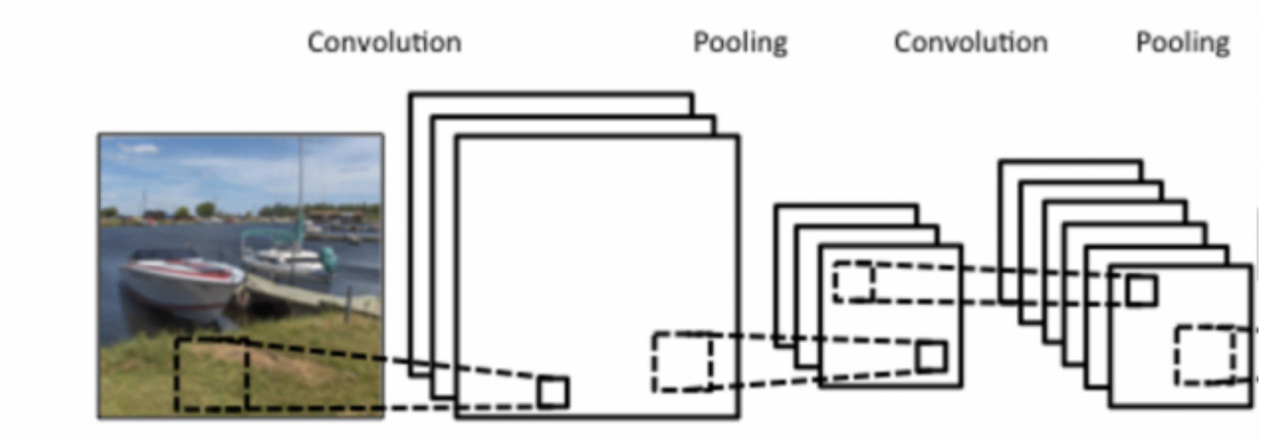
- Variations on this model:
  - ➡ **Deep architecture** with more layers
  - ➡ **Bi-directional LSTM** running forward and backward simultaneously
  - ➡ **Convolutional autoencoder** instead of LSTM for layer-wise prediction



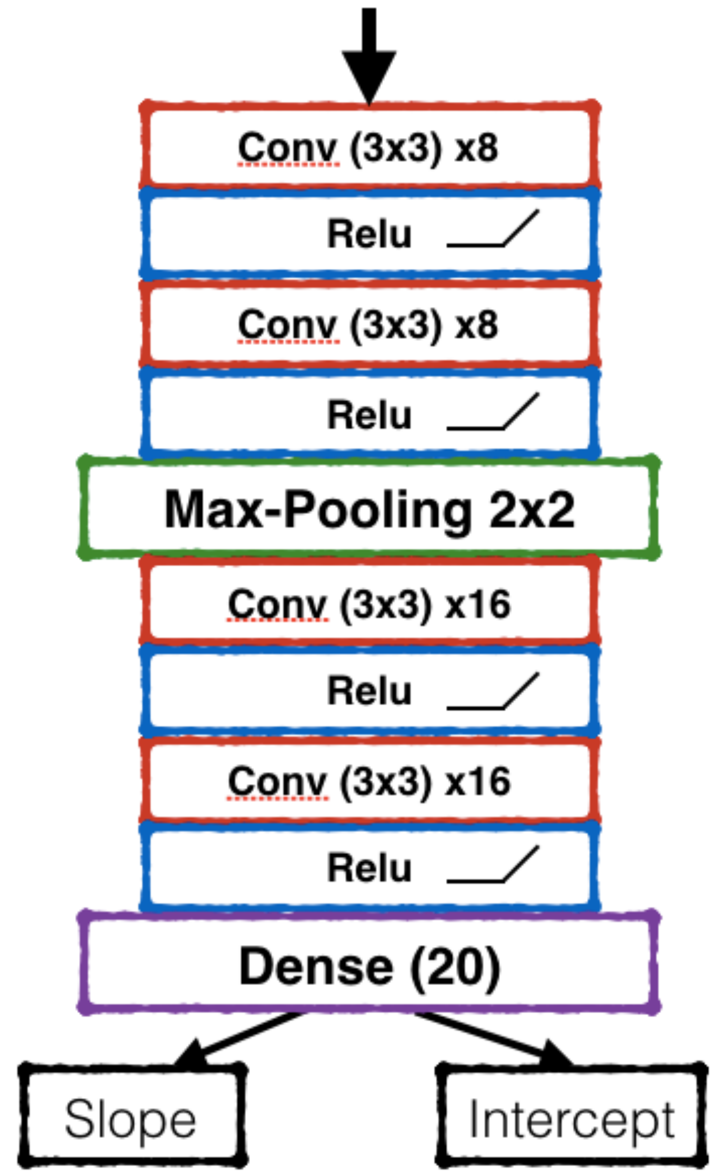
Extend also to 3-D toy data

Predicting Track Parameters

- **Different approach**: treat track finding as an image recognition problem
- Use **convolutional neural networks** — powerful tools for extracting image features



http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

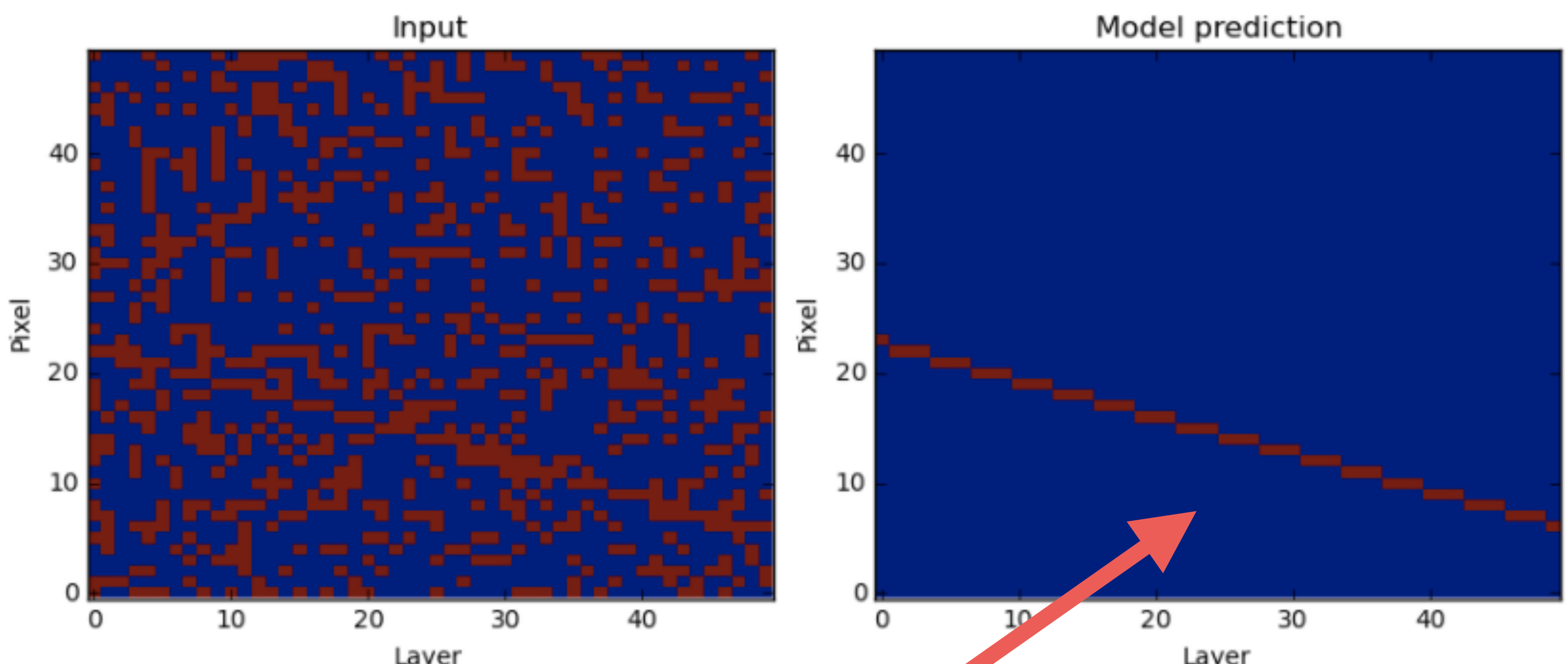


Network Architecture

Predicting Track Parameters

- Given image of a track, the model **directly predicts its parameters** (slope & intercept, in this case)

- Ex: single track with large noise background

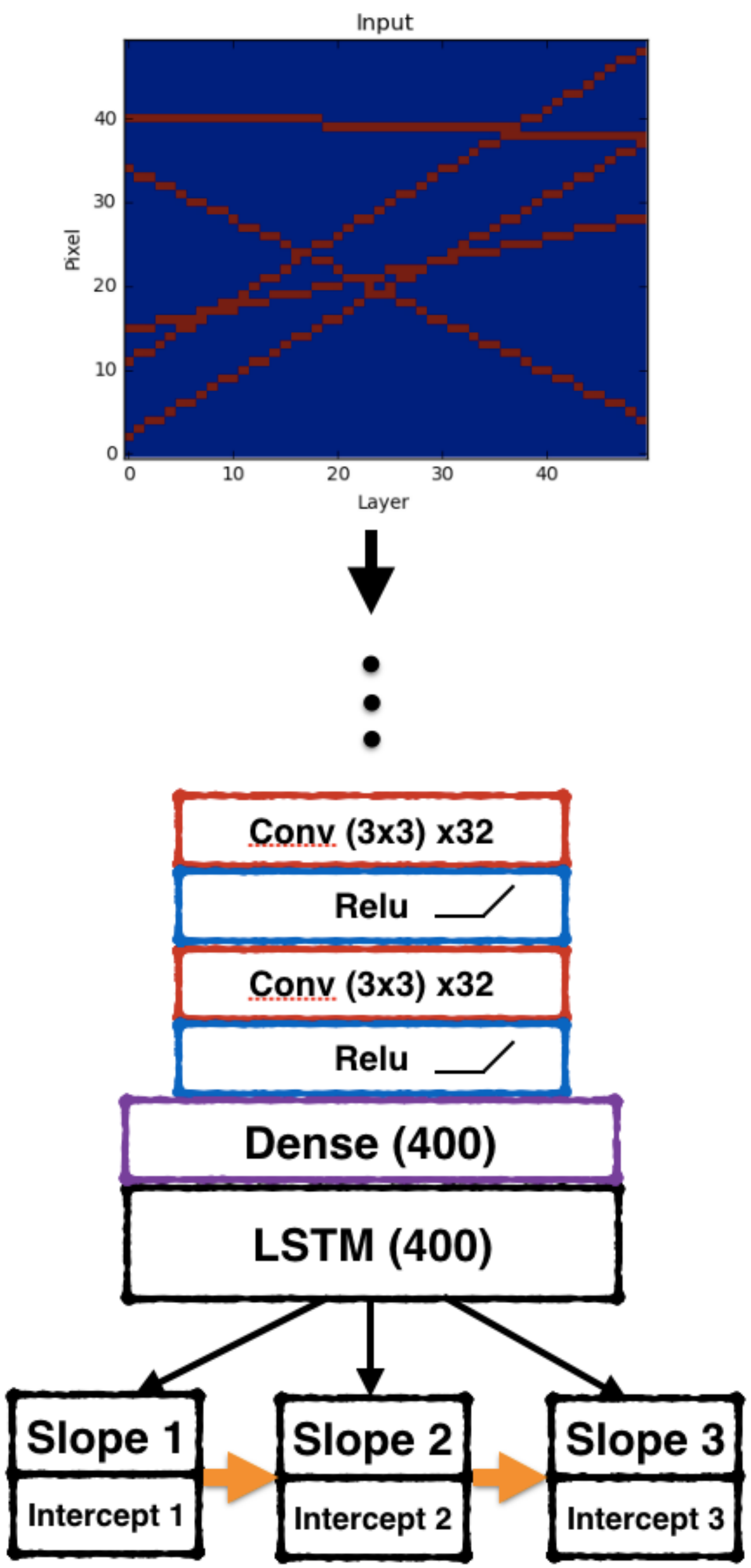


Visualization of predicted track

Caltech

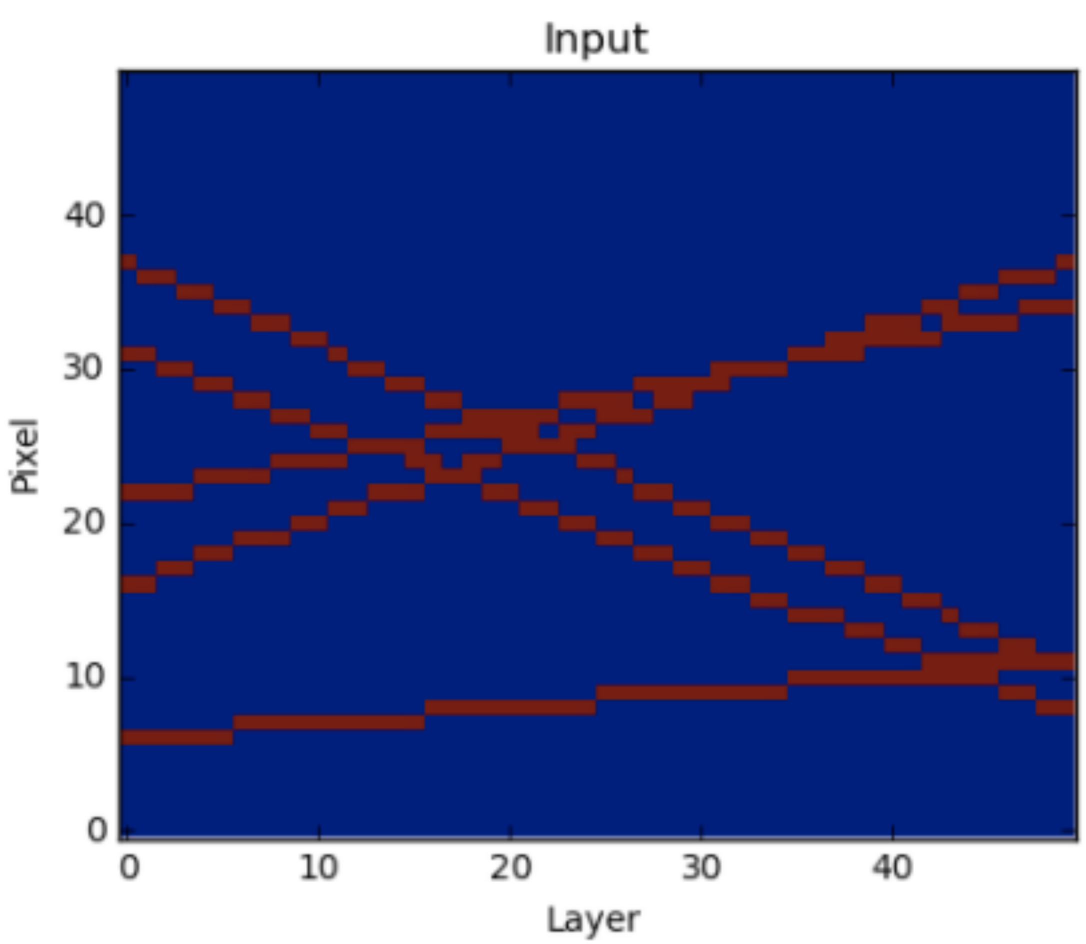
Many Tracks

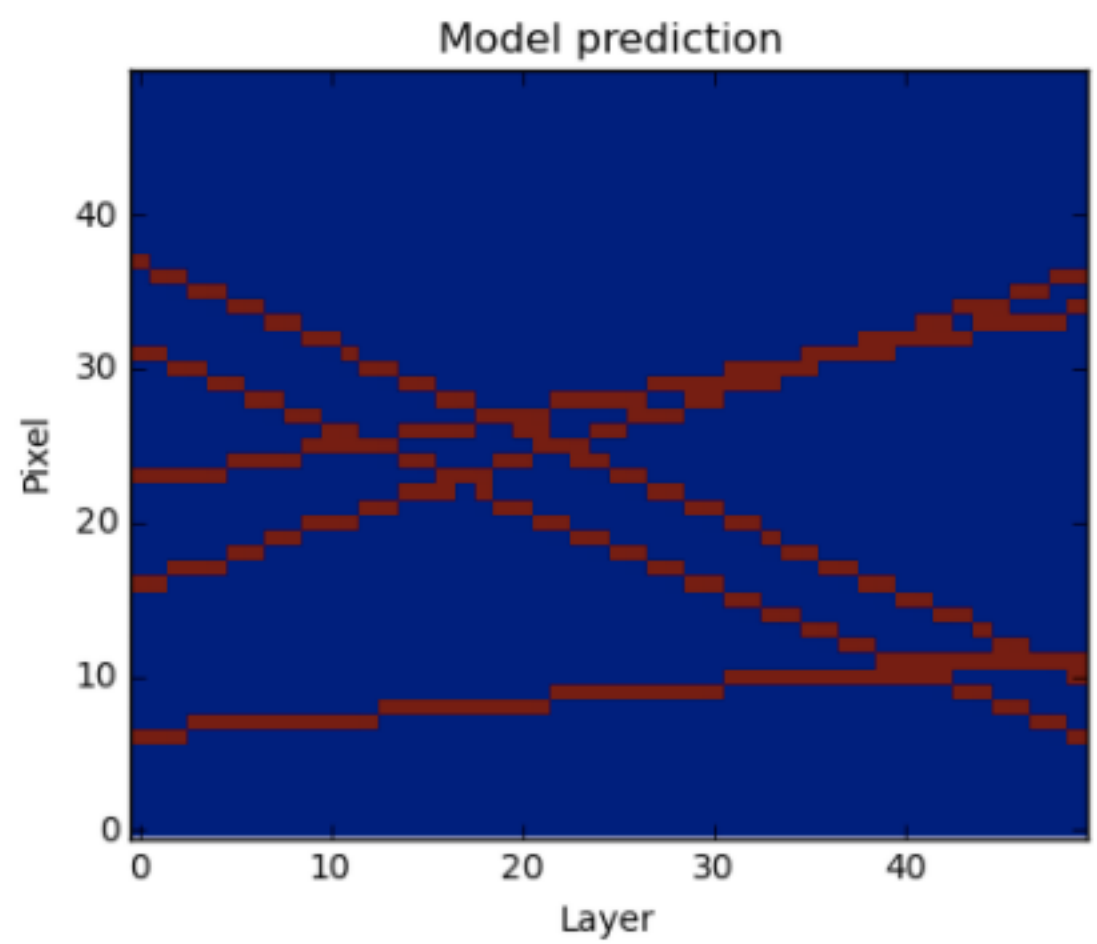
- Deal with multiple tracks per event using an LSTM network
- Different from earlier LSTM application. At each LSTM step:
    - ➡ **It outputs parameters for a complete track**
    - ➡ **The memory cell updates to focus on a new track in the image**



Predicting Track Parameters

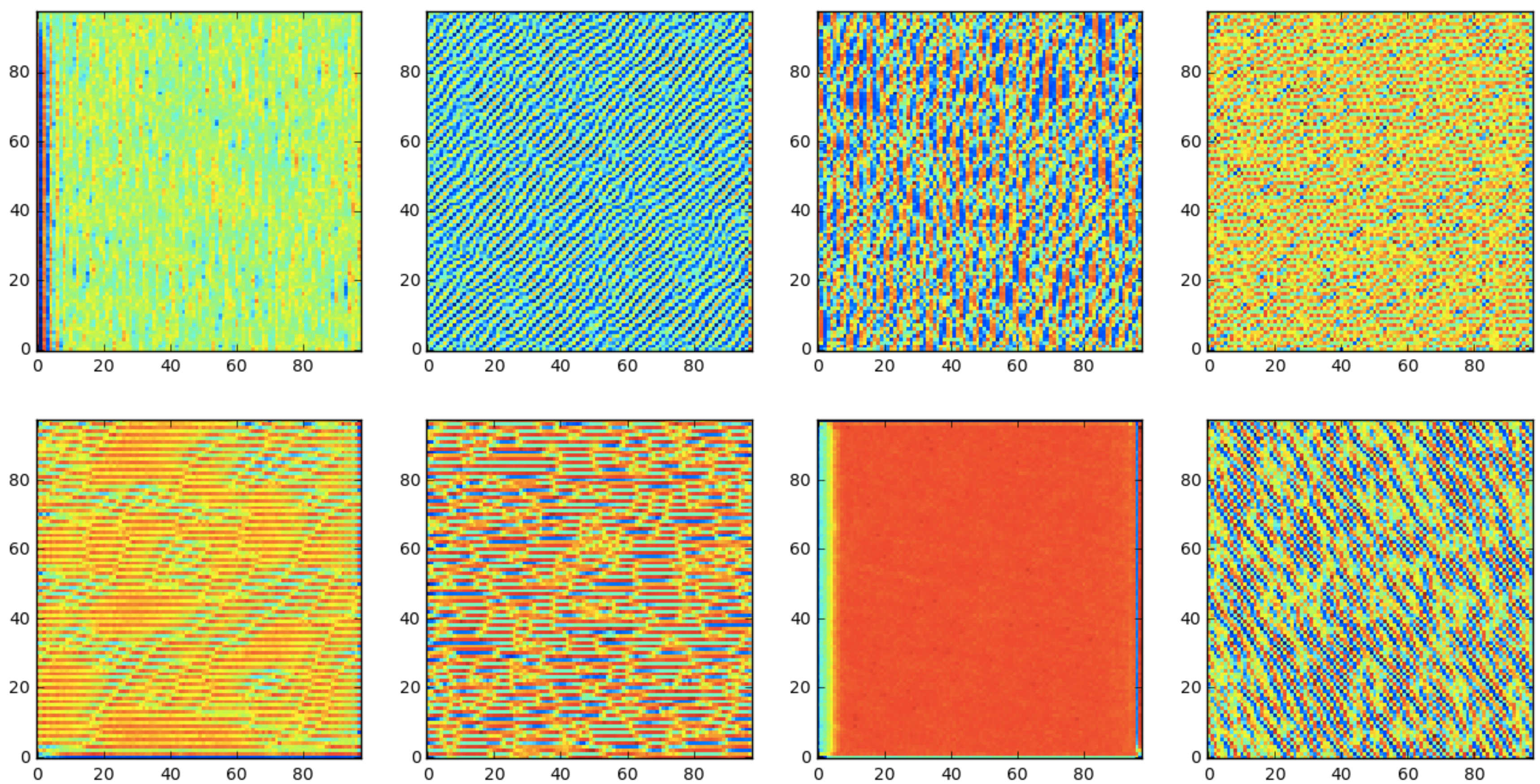
- The model processes the image and identifies all tracks in one pass!





Visualizing Filters

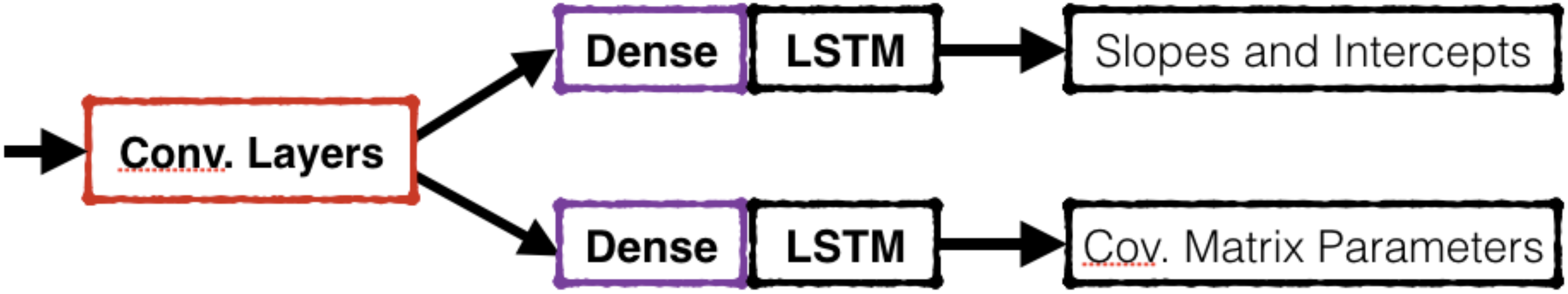
- Visualize the learned filters by finding an image that maximizes each filter’s activation level
- Gives insight into the patterns that the model “sees”



Inspired by: https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html

Expressing Uncertainty

- To be useful, the model must assign uncertainties to its predictions
- **Strategy**: train the model to produce a parameter covariance matrix for each track it finds



Expressing Uncertainty

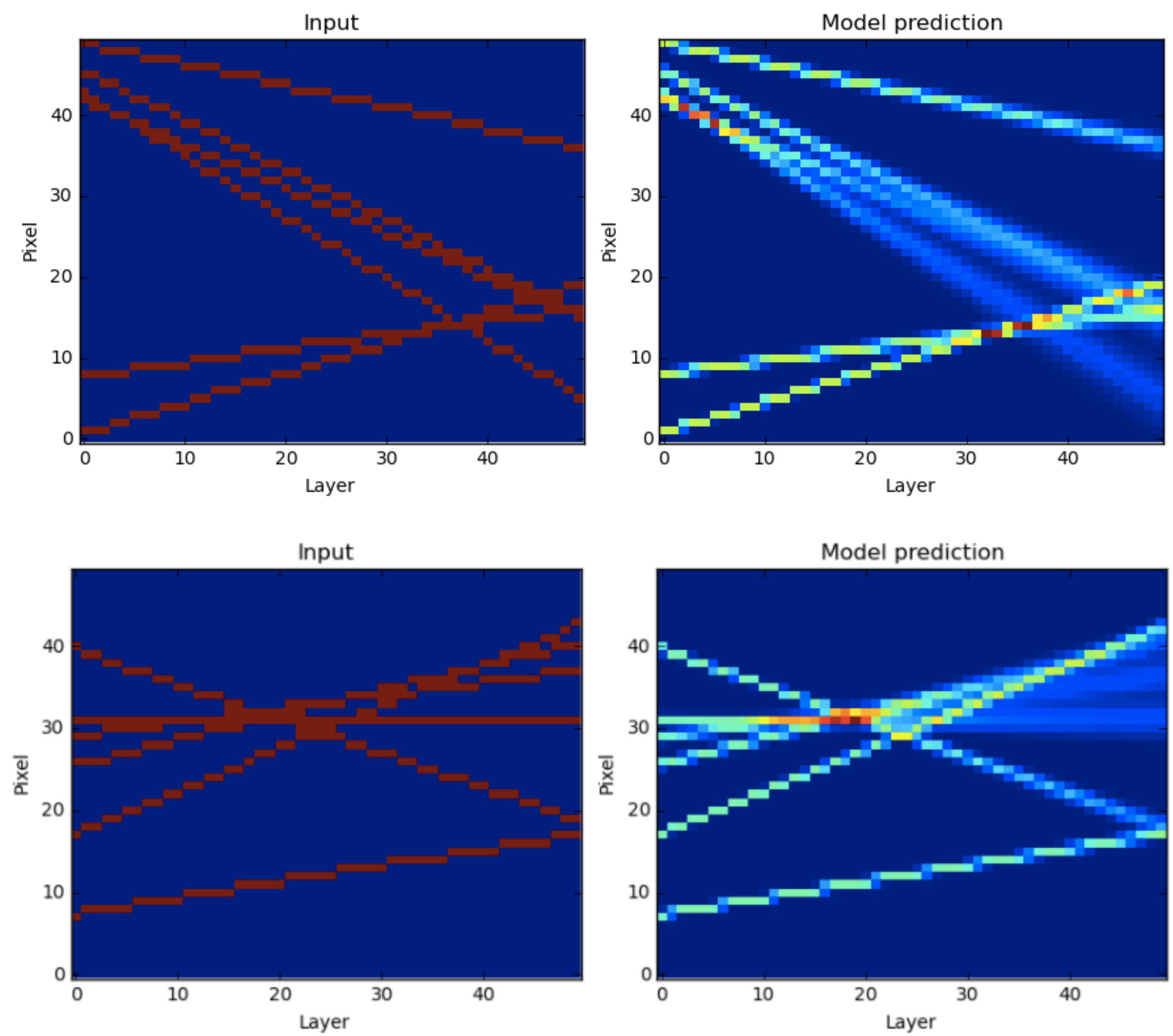
- During training, minimize negative gaussian log likelihood:

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

- The model learns to produce track covariance matrices that accurately reflect its performance

Expressing Uncertainty

- Sample from each track's covariance matrix to visualize the uncertainty on the model predictions

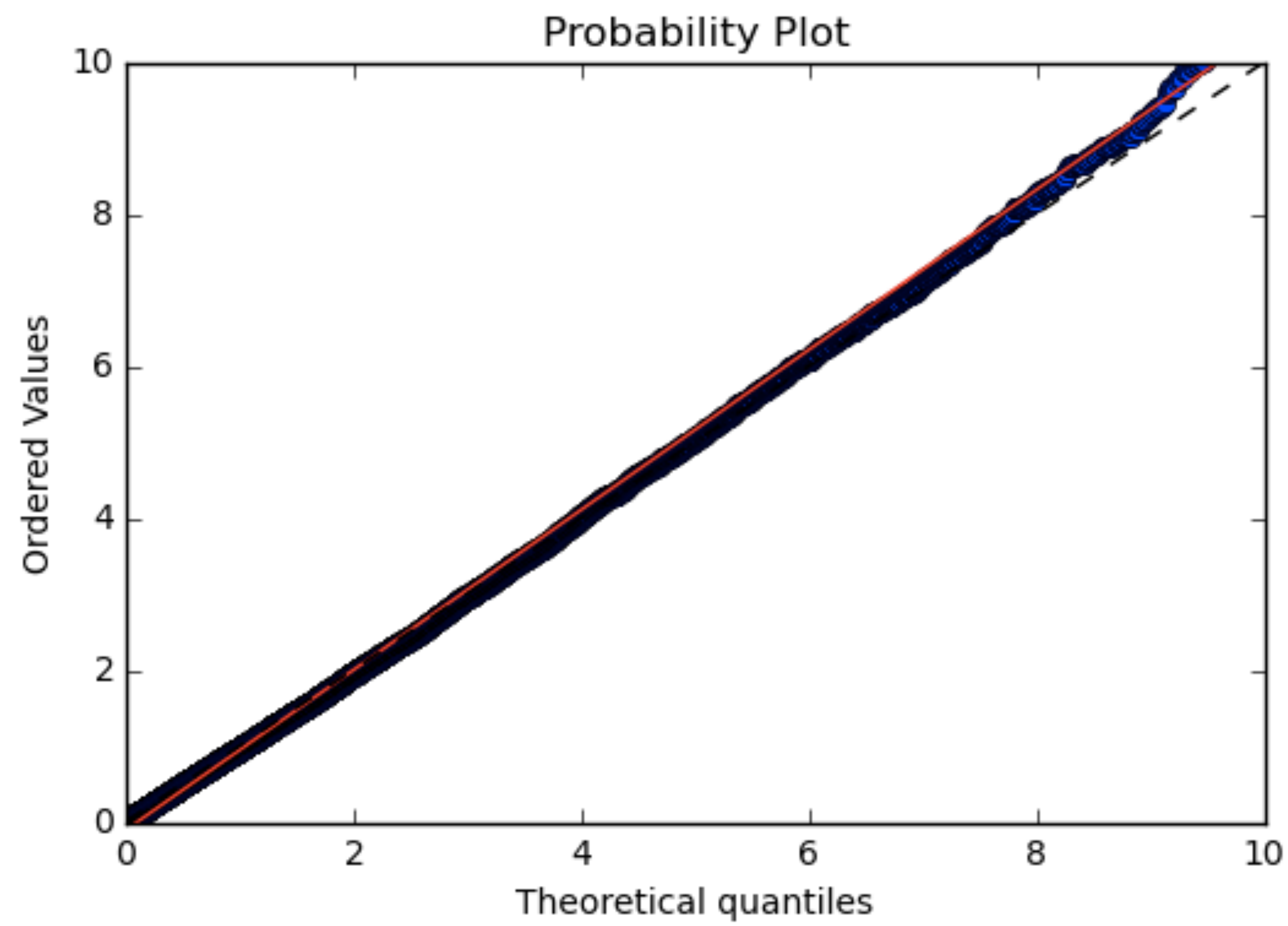


Expressing Uncertainty

- Evaluate the uncertainties via the distribution of **Mahalanobis distances**:

$$D_M(\vec{x}) = \sqrt{(\vec{x}-\vec{\mu})^T S^{-1} (\vec{x}-\vec{\mu})}$$

- They should be chi-square distributed
- After a small by-hand calibration, **the errors have the expected distribution**



Y-axis: quantiles of observed distribution
X-axis: quantiles of chi-square distribution

Exploring Further

- Next plans for the group:
  - Choose **1-2 model architectures** to optimize and scale up
  - Move from toy data to realistic detector simulation — **ACTS data**
  - Compare mature ML algorithms with baseline performance provided by Kalman filter

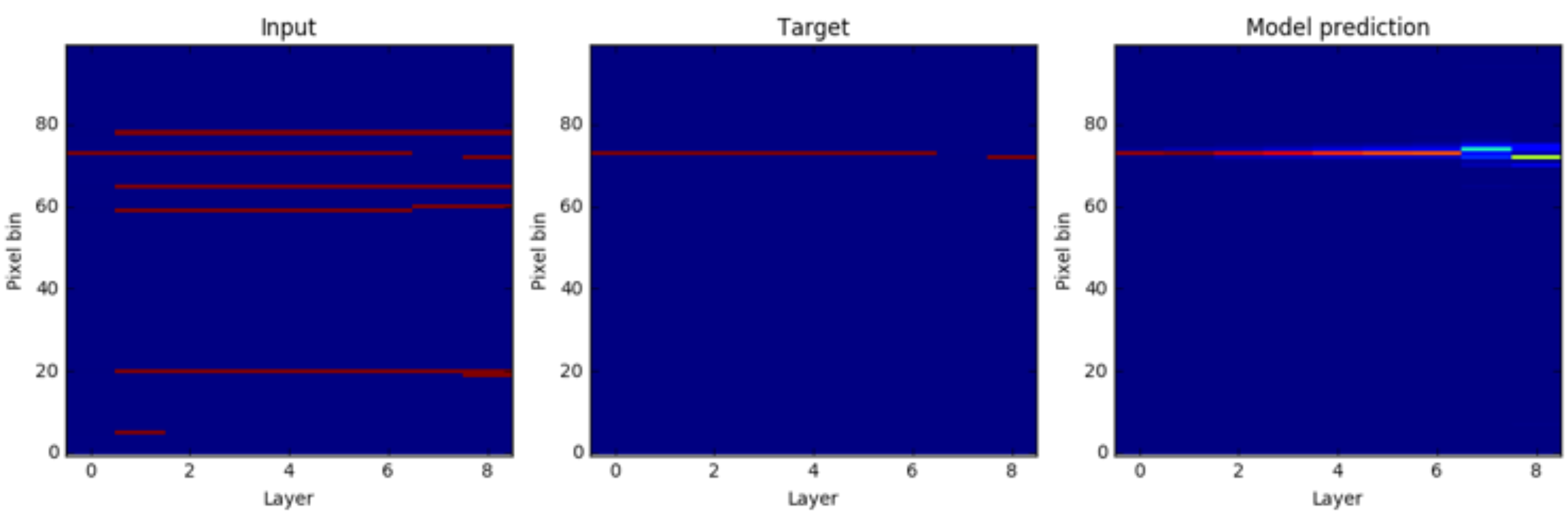
Conclusion

- Developing a new, scalable particle tracking algorithm is critical for detector performance in the HL-LHC era
- The HEP.TrkX project is exploring ML-inspired tracking algorithms, towards this end
- Recurrent and convolutional NN models show promise on simplified detector data
- **Stay tuned for further developments!**

S. Farrell

TrackMLRamp Hackathon at CTD 2017

- 2D tracking challenge with curved tracks, scattering effects, detector inefficiencies, and stopped tracks
  - Goal: cluster the hits in each event into tracks
- Adapted LSTM model as follows
  - Unroll the circular detector and bin hits coarsely in phi to produce square “images”
  - Use first layer hits as “seeds”
  - Use LSTM model to score hits per track
  - Assign hits to their highest scored track
- **Won in the ML category of the challenge with 92.1% reco efficiency**
  - Later adjustments using a high granularity window centered on the track seed boosted performance to 94.9%



Model is robust against holes and other detector effects