Particle Track Building with Recurrent Neural Networks

Steve Farrell[1], Dustin Anderson[2], Josh Bendavid[2], Paolo Calafiura[1], Giuseppe Cerati[3], Lindsey Gray[3], Jim Kowalkowski[3], Mayur Mudigonda[1], Prabhat[1], Panagiotis Spentzouris[3], Maria Spiropulu[2], Aristeidis Tsaris[3], Jean-Roch Vlimant[2], Stephan Zheng[2]

[1]*Lawrence Berkeley National Laboratory*
[2]*California Institute of Technology*
[3]*Fermi National Accelerator Laboratory*

HEP TrkX

https://heptrkx.github.io

Tracking at the LHC

The problem

Reconstruct thousands of particles from tens of thousands of spacepoint "hits".

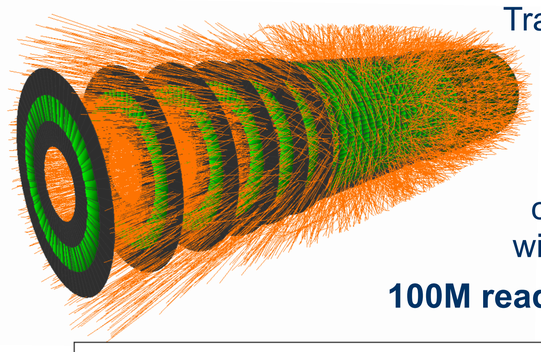
Traditional algorithm approach

- **Seeding**: construct initial segments of tracks (seeds) of 2-3 hits using trajectory constraints
- **Track building**: extrapolate seeds and assign hits using combinatorial Kalman Filter
- **Track fitting**: resolve remaining ambiguities between candidates and fit trajectory parameters

Limitations of traditional algorithms

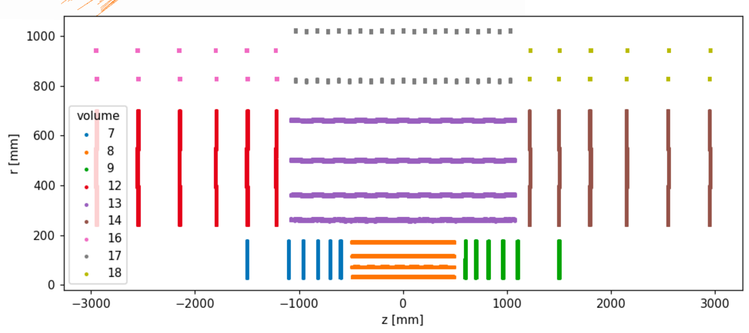
- Quadratic (or worse) scaling with occupancy
- Hand-engineered features and methods
- Inherently serial

LHC Detectors



Tracking detectors composed of highly granular silicon sensors arranged on cylindrical layers with endcap disks

100M readout channels!



Why neural networks?

Possible benefits

- Regular, parallelizable computation
- Non-linear modeling
- Learned representations/features

RNN hit predictor model

Replace Kalman Filter with a Recurrent Neural Network:

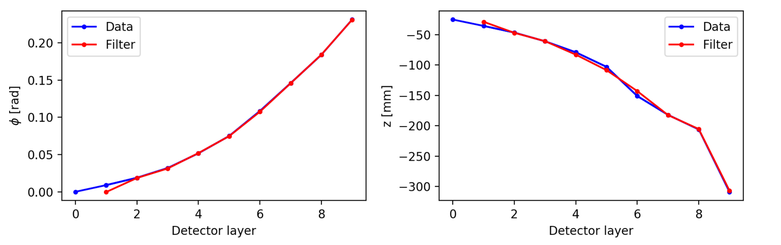
- For a sequence of hit positions, predict the location of the next hit
- Train as a simple MSE regression
- Use to score candidate hits in tree search

Architecture:

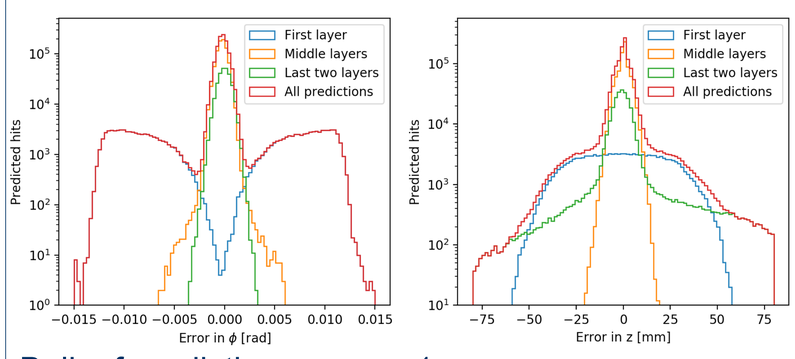
$\vec{r} = (r, \phi, z) \quad \hat{\vec{r}} = (\hat{\phi}, \hat{z})$

$\vec{r}_0, \vec{r}_1, \ldots, \vec{r}_{N-1}$ → LSTM → FC → $\hat{\vec{r}}_1, \hat{\vec{r}}_2, \ldots, \hat{\vec{r}}_N$

Trajectories:



Residual errors:



Bulk of prediction errors < 1mm

RNN Gaussian hit predictor model

This model produces predictions as bi-variate Gaussian probability distributions.

- Now we have predictions with uncertainty!
- Trained with Gaussian log-likelihood loss

Architecture:

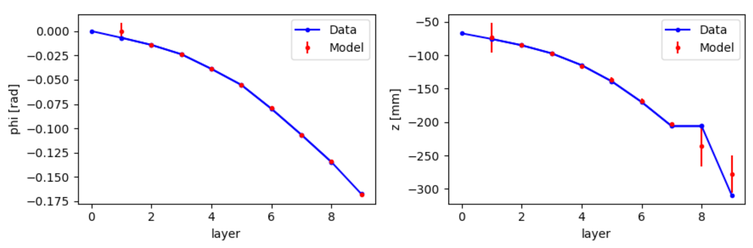
$\vec{r}_0, \vec{r}_1, \ldots, \vec{r}_{N-1}$ → LSTM → FC → $(\hat{\vec{r}}_1, \Sigma_1), (\hat{\vec{r}}_2, \Sigma_2), \ldots, (\hat{\vec{r}}_N, \Sigma_N),$

$\vec{r} = (r, \phi, z) \qquad \hat{\vec{r}} = (\hat{\phi}, \hat{z}) \quad \Sigma = \begin{pmatrix} \sigma_\phi^2 & \sigma_{\phi z}^2 \\ \sigma_{\phi z}^2 & \sigma_z^2 \end{pmatrix}$

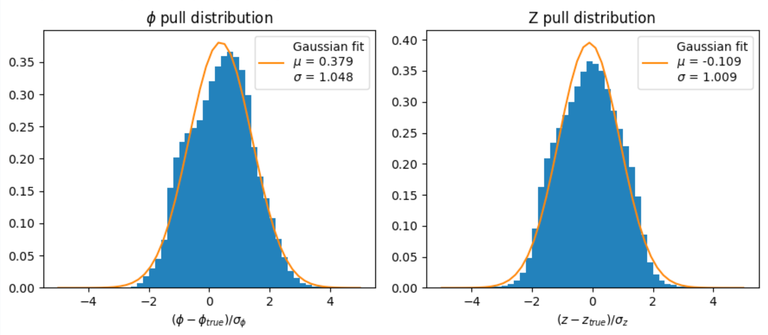
Loss:

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

Trajectories:



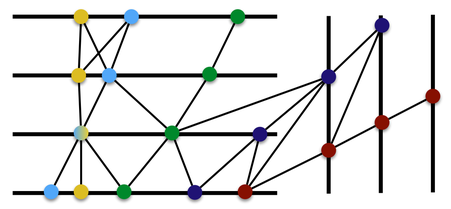
Pull distributions:



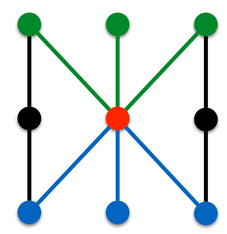
Some non-Gaussian features, but promising results

Recurrent graph neural network

Represent the data as a *graph* of connected hits constructed with geometric constraints (delta-phi, delta-z)



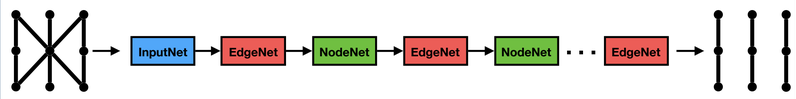
Use ***Graph Neural Networks*** to learn on this representation



Two network components operate on the graph *locally*:

- ***Edge network*** uses the **node features** to compute **edge weights**
- ***Node network*** aggregates **forward** and **backward** node features with the edge weights and computes **new node features**

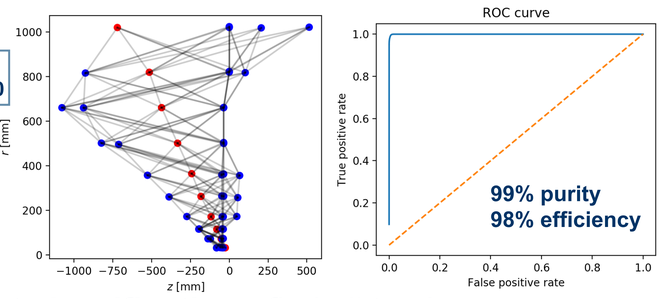
Chain these together as a recurrent neural network



Hit classification - find the hits that belong to one seeded track via binary classification of the graph nodes

pT > 1 GeV
pileup μ = 10

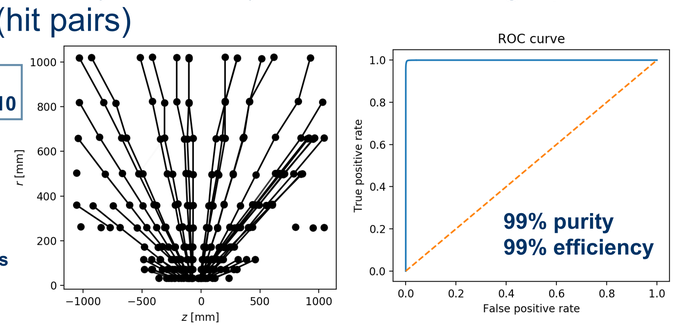
- **7 graph iterations**
- **26k model parameters**



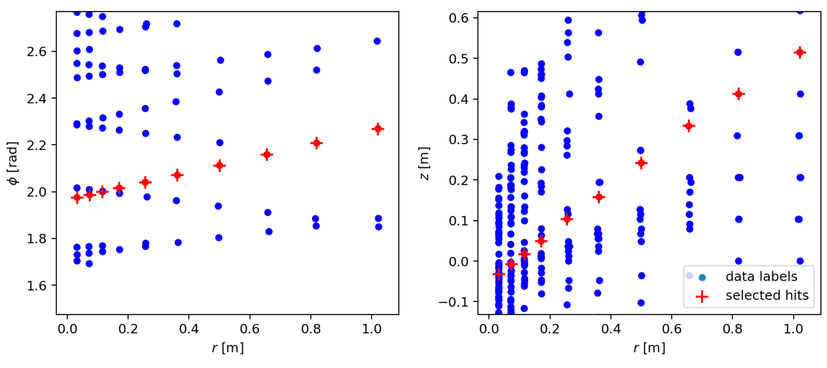
Segment classification - find all tracks simultaneously via binary classification of graph edges (hit pairs)

pT > 1 GeV
pileup μ = 10

- **4 graph iterations**
- **7k model parameters**



Building tracks with hit predictor models



Use hit predictor models to score candidate hits like a Kalman Filter and build tracks

- Tested on μ=10 pt>1GeV data
- Start with 3-hit seed
- Choose best hit at each successive layer
- No combinatorial branching

Model	Hit selection accuracy
Simple	99.93%
Gaussian	99.98%

→ *Easy scenario but important first sanity check!*

Summary

Deep learning applications seem promising for HEP particle track reconstruction

- RNNs can function like state estimation filters for particle track dynamics
- Models can learn to produce predictions *with uncertainties*
- Graph representations allow for powerful GNN models for finding tracks in events
  - Most promising approach thus far

CHEP 2018