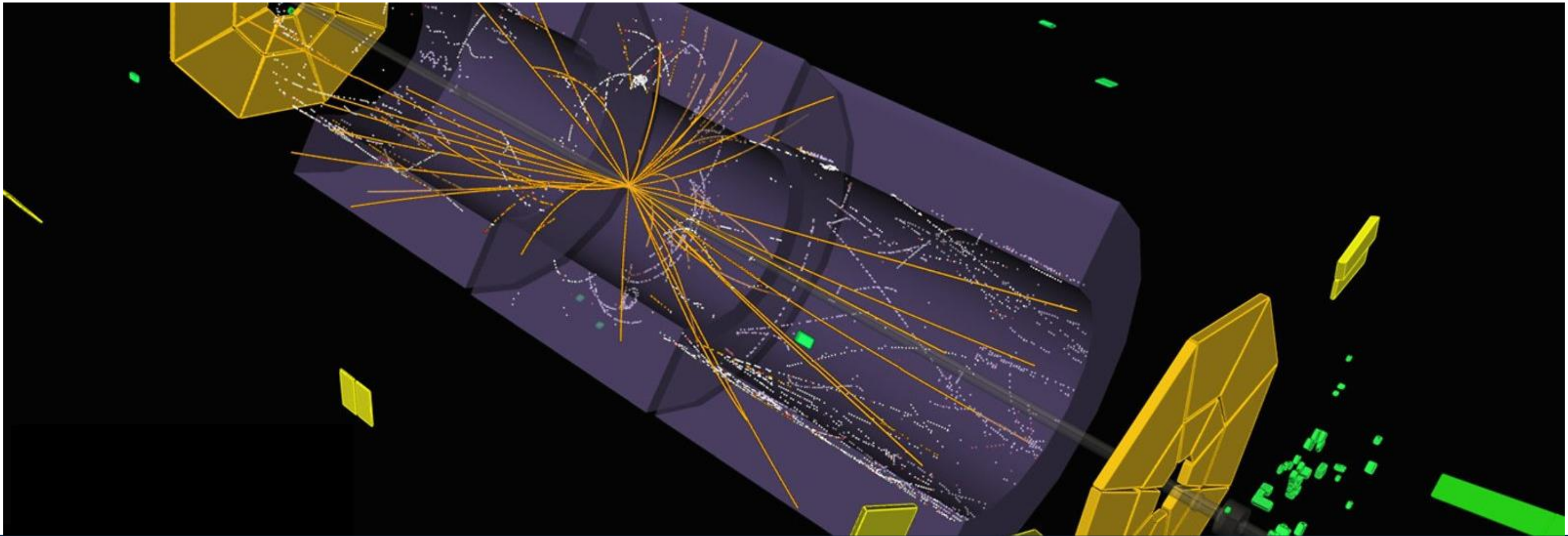


Machine Learning Based Track Reconstruction

Joint GlueX-PANDA-EIC ML virtual workshop

22.09.2020 I **Waleed Esmail**, Tobias Stockmanns and James Ritman
Institut für Kernphysik (IKP), Forschungszentrum Jülich

Outlines:

- Introduction
- TrackML challenge
- PANDA FTS
- ANN and RNN application to FTS
- GNN application to FTS
- Track fitting
- Hands-on tutorials

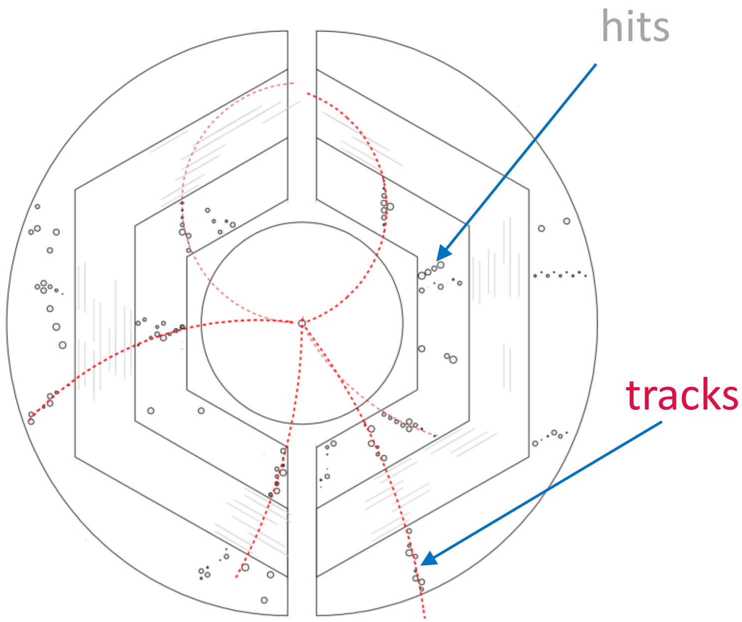
Introduction:

- **Track reconstruction is a pattern recognition task**
- **Two main steps**: **Track Finding** and **Trak Fitting** (usually done in iterative procedure).
- **Track Finding**: assign position measurements (hits) to track candidates (particle paths)
- **Track Fitting**: determine track parameters and covariance matrix for each track
- **Track finding is usually the most time-consuming part in the reconstruction process**

- There are two generic approaches for track finding:
  1. **Local approach**: find track candidates consecutively
  2. **Global approach**: find all track candidates at once

- Good tracking algorithm should be

  high efficiency,
  high purity,
  low fake rate, and
  fast algorithm



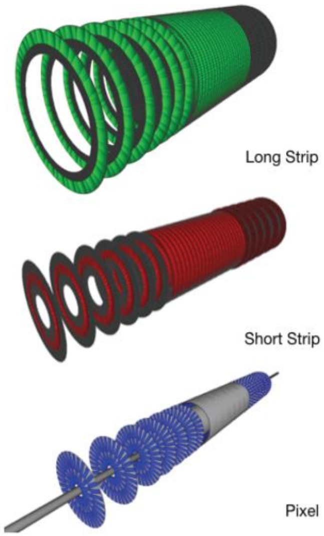
Data Analysis Techniques for High-Energy Physics, R. Frühwirth, (1990)

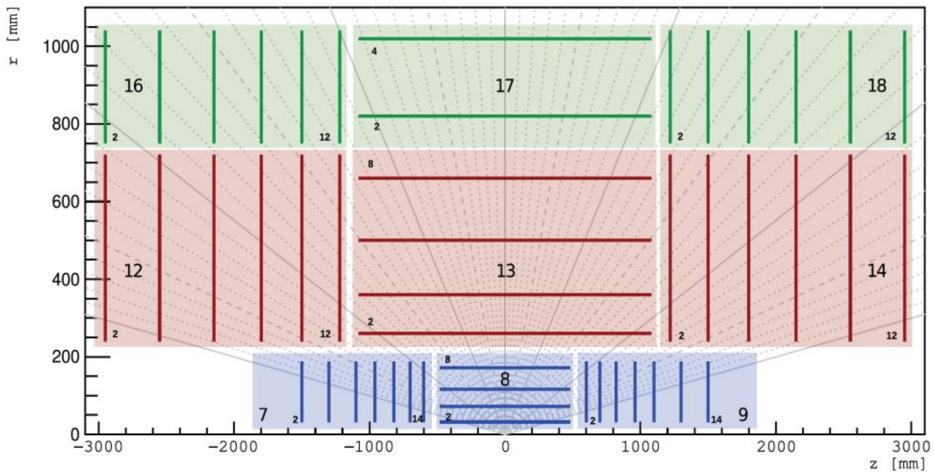
TrackML challenge I:

- A **competition** hosted by Kaggle (Accuracy) and Codalab (Accuracy & Speed)
- A participant is challenged to build an algorithm that quickly reconstructs particle tracks from 3D points (hits)
- Can ML tracking compete traditional approaches (for HL-LHC)?!
- Realistic detector model to simulate measured particle hits (ACTS simulation)
- A hard QCD interaction overlayed with soft QCD interactions (pileup)

~ 10000 tracks/event

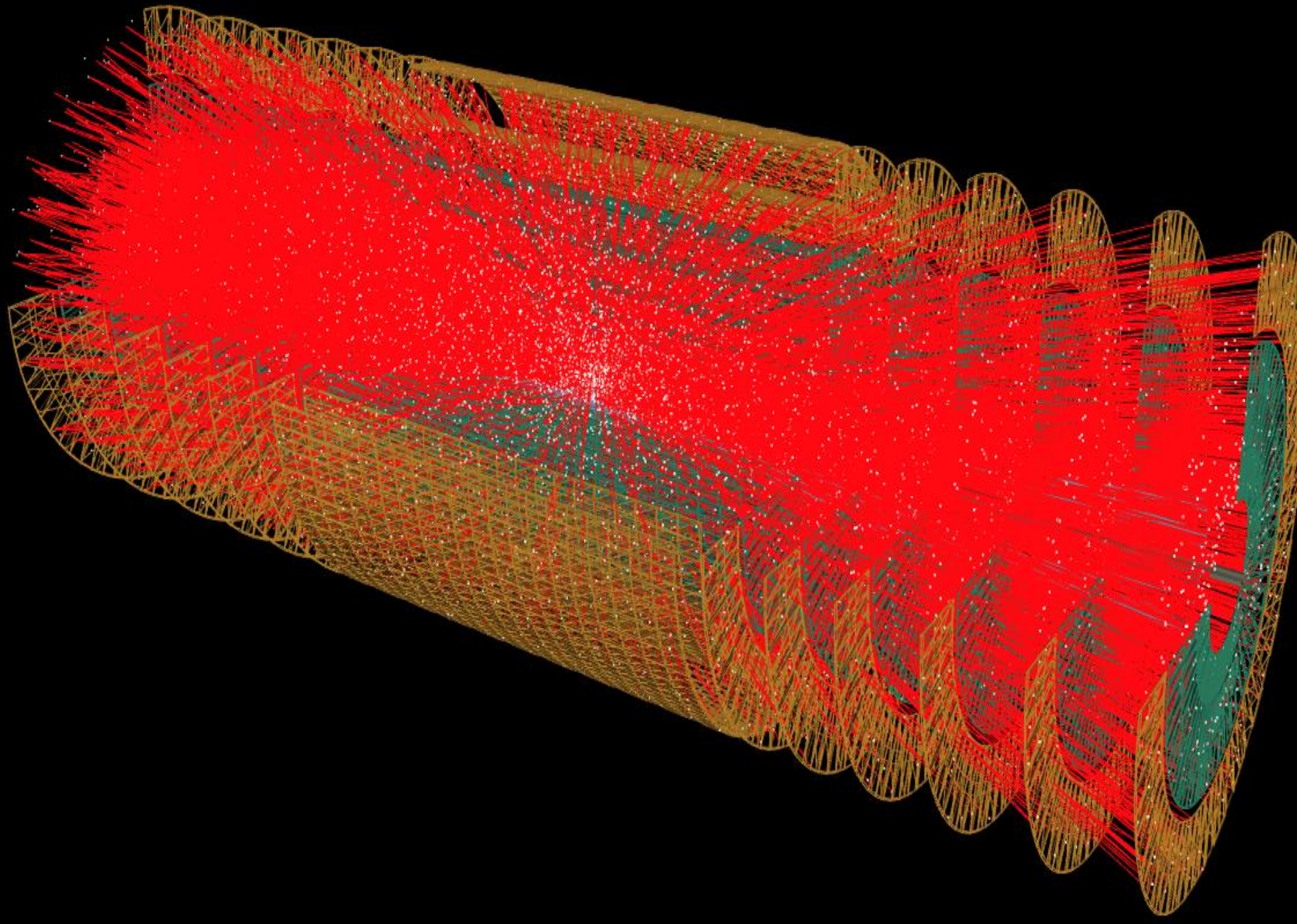
~ 10 hits/track





EPJ Web of Conferences **214**, 06037 (2019)

JÜLICH Forschungszentrum



TrackML challenge II:

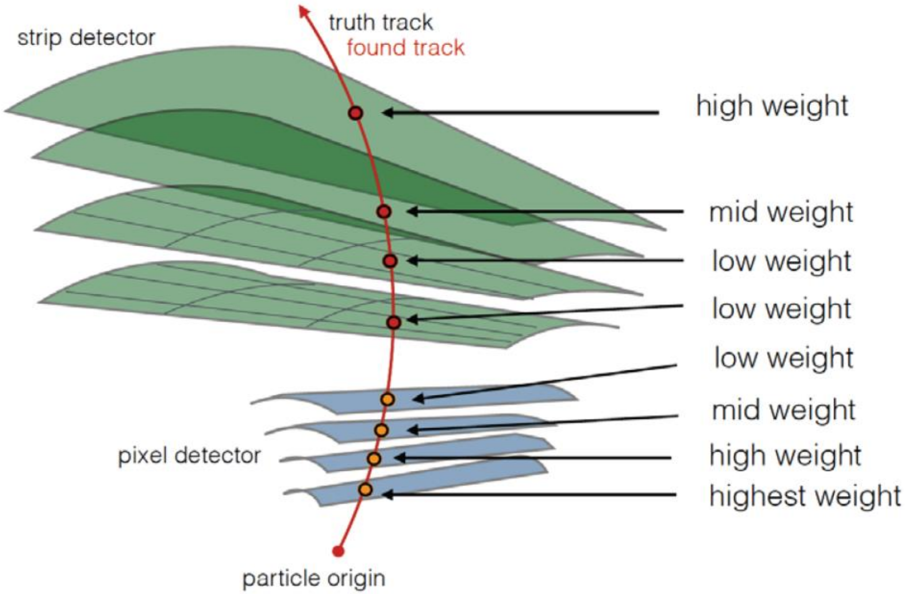
- What is provided
  1. 3D space points in global coordinate system (hits)
  2. Cells: Each hit originates from one or more active detector cells
  3. Ground Truth information (for supervised models)

- What is expected
  1. High score

$$S \sim \sum_{\{\text{events}\}} \sum_{\{\text{tracks}\}} \begin{cases} 0 & \#\text{good hits} < 50\,\%, \#\text{hits} < 3 \\ \sum_{\{\text{good hits}\}} w_i & \text{else} \end{cases}$$

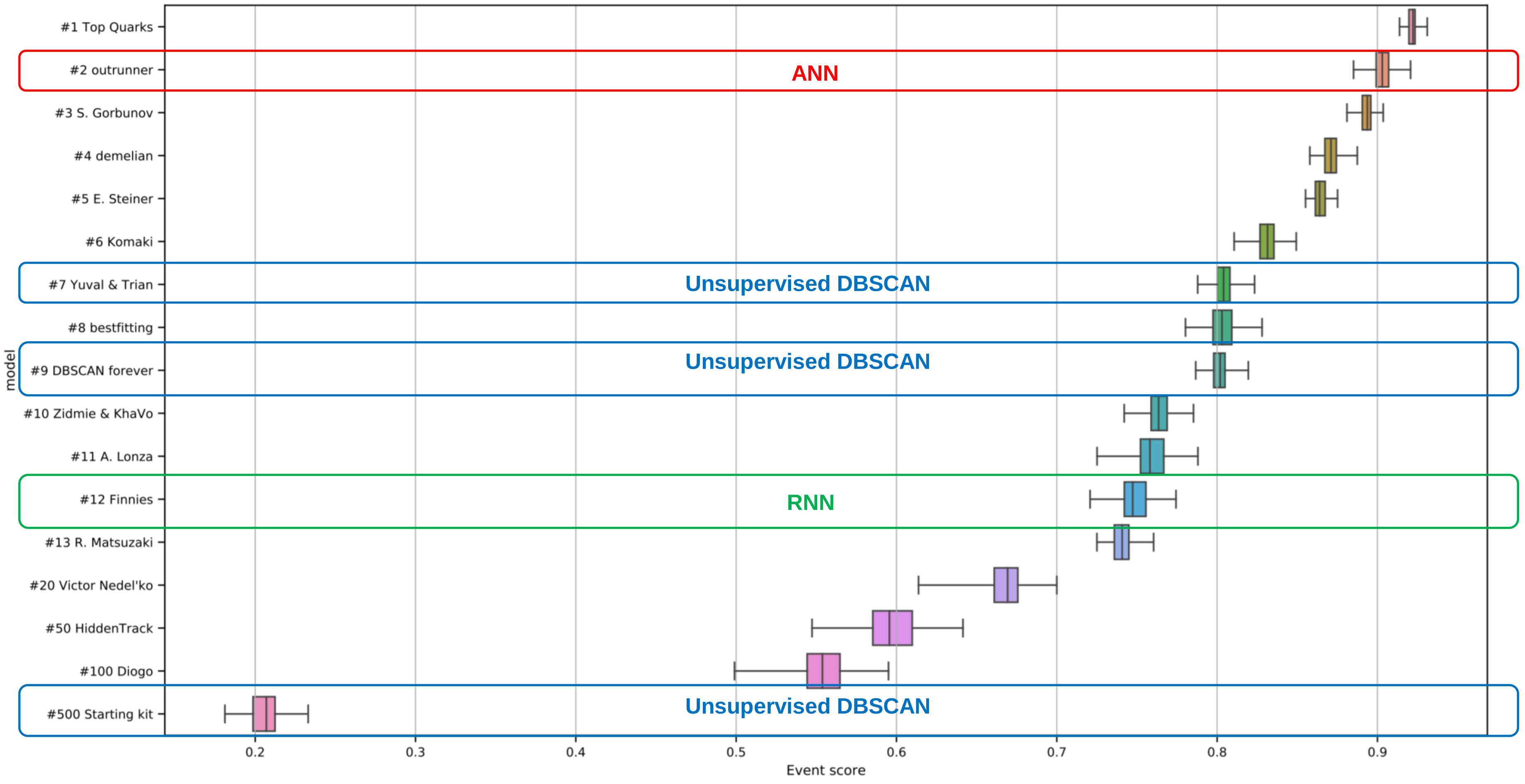
$$S_{perfect} = 1$$

$$w_i = w_i\,(\text{hit order, particle } p_\perp) \ .$$



EPJ Web of Conferences **214**, 06037 (2019)

TrackML Solutions



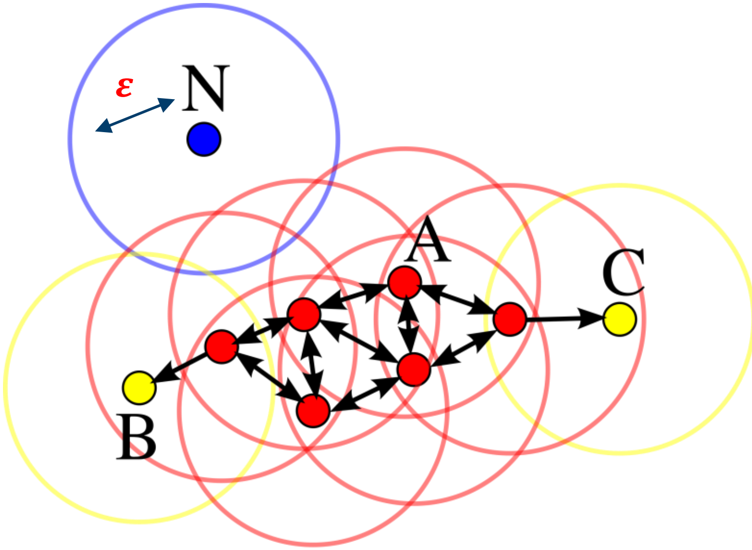
EPJ Web of Conferences **214**, 06037 (2019)

TrackML Solutions: (DBSCAN I)

- Track finding is a clustering process, so why not to use **unsupervised** methods!
- **Clustering**: cluster data points (**hits)** that are more similar to each other
- **DBSCAN** Density Based Spatial Clustering of Applications with Noise
- **Density is parametrized by a hyperparameter ε.**

MinPts = 4

- Label is assigned to each data point
  1. **core point** (**>=** min # of points **MinPts** within ε)
  2. **boarder point** (**<** min # of points **MinPts** within ε)
  3. all other points are **noise points**.



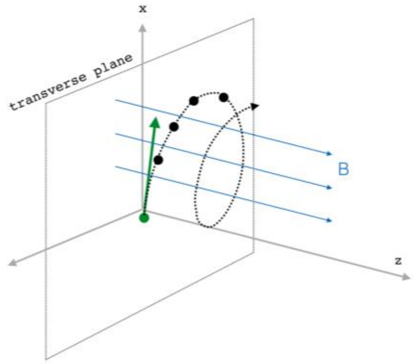
Wikipedia/DBSCAN

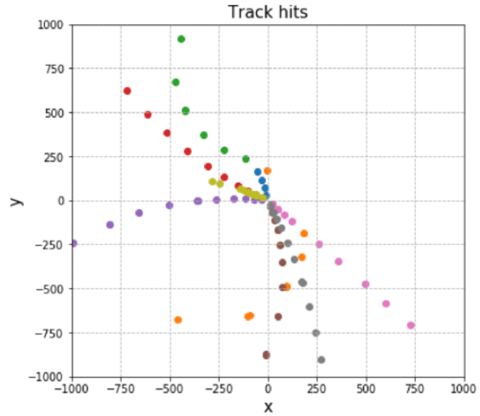
- A point q is **directly reachable** from p if point q is within distance ε from core point p, or if there is a path of points.

https://towardsdatascience.com/dbscan-algorithm-complete-guide-and-application-with-python-scikit-learn-d690cbae4c5d

TrackML Solutions: (DBSCAN II)

- One of the baseline solution with accuracy ~ 0.25
- Idea is to do **hit transformation**





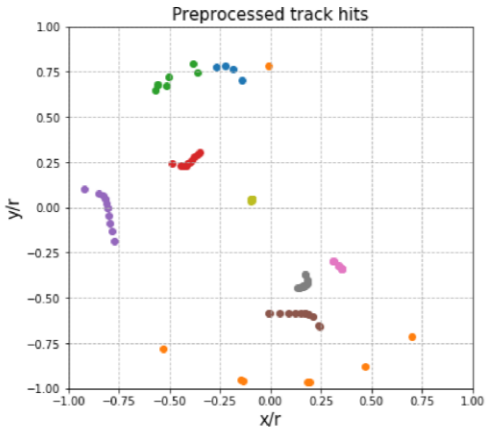
$$r_1 = \sqrt{x^2 + y^2 + z^2}$$

$$x_2 = x/r_1$$

$$y_2 = y/r_1$$

$$r_2 = \sqrt{x^2 + y^2}$$

$$z_2 = z/r_2$$



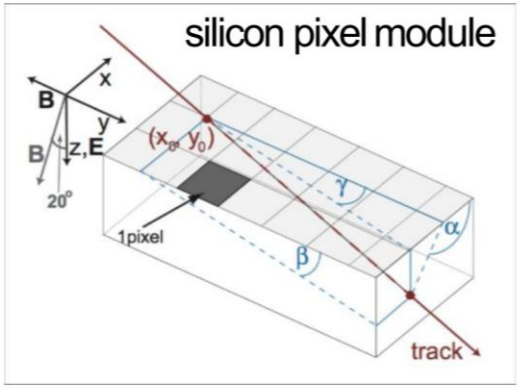
- Many other solutions based on DBSCAN are heavily dependent on **preprocessing (feature engineering)**
- Core idea is to **unroll the helix**

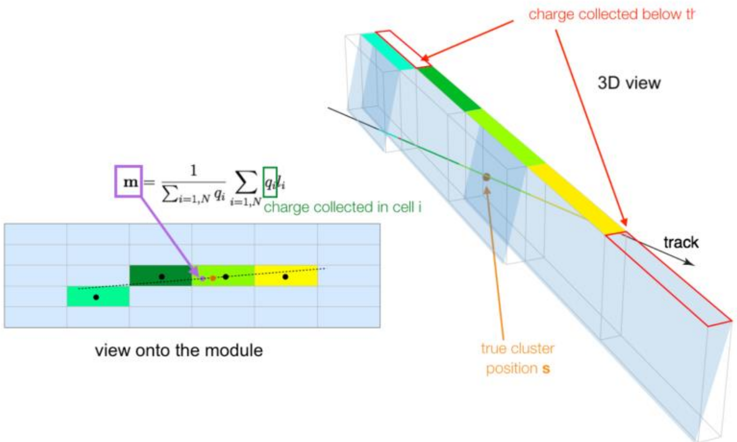
EPJ Web of Conferences **214**, 06037 (2019)

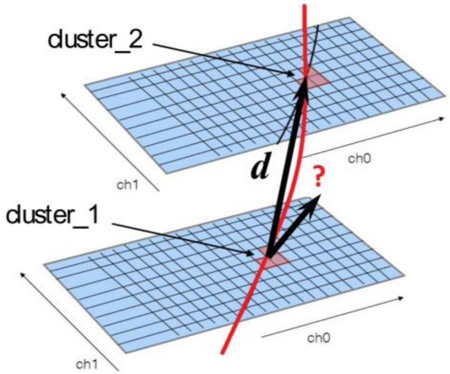
TrackML Solutions: (DL solution I)

➢ The solution that ranked **second in the challenge** is using an artificial neural network.

- **Input [two hits] -> DNN -> output [pair quality]**
- **Input features (x, y, z, direction from cells, …)**
- **Output (pair probability) -> (Adjacency Matrix)**







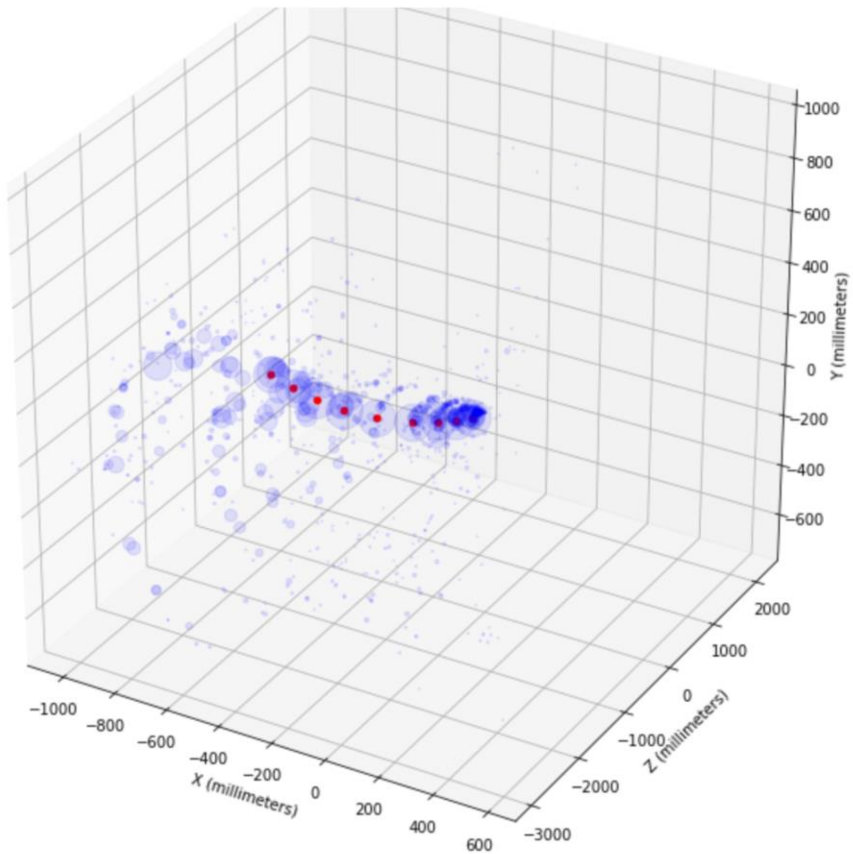
EPJ Web of Conferences **214**, 06037 (2019). Kaggle/TrackML

JÜLICH
Forschungszentrum

TrackML Solutions: (DL solution II)

- Build tracks by maximizeing the sum of probabilities

	h1	h2	h3	h4	h5
h1	-	0.8	0.2	0.9	0.4
h2	0.8	-	0.5	0.7	0.7
h3	0.2	0.5	-	0.3	0.4
h4	0.9	0.7	0.3	-	0.4
h5	0.4	0.7	0.4	0.4	-



- If threshold = 0.65

	h1	h2	h3	**h4**	h5
h1	-	0.8	-	0.9	-
h2	0.8	-	-	0.7	0.7
h3	-	-	-	-	-
h4	0.9	0.7	-	-	-
h5	-	0.7	-	-	-

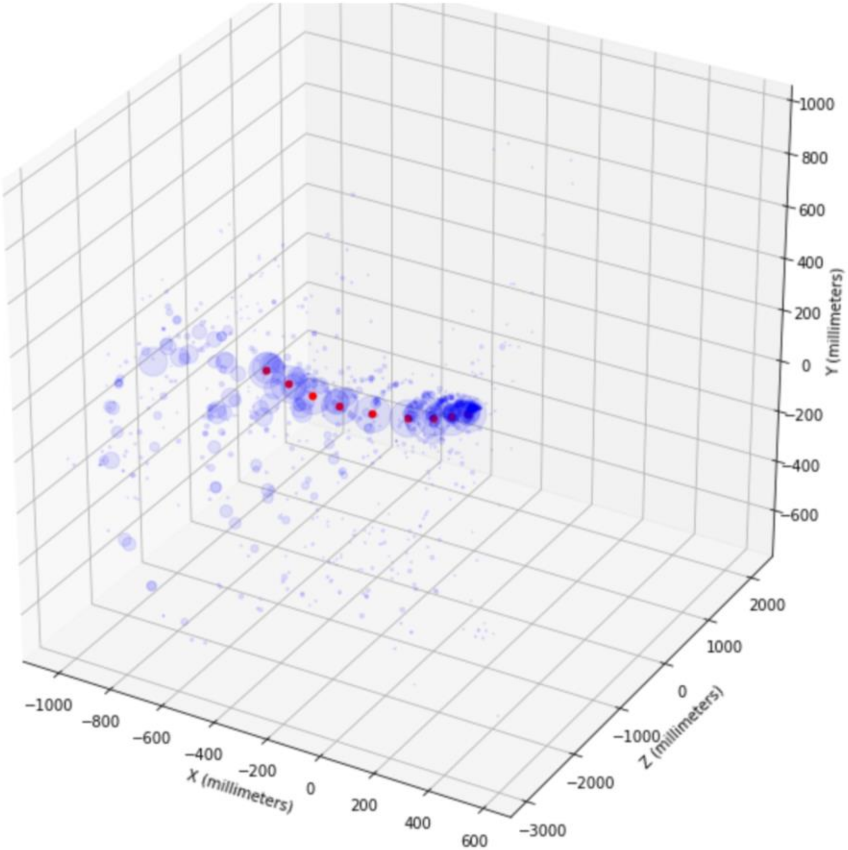
- p(h1,h4) = 0.9 > 0.65

EPJ Web of Conferences **214**, 06037 (2019). Kaggle/TrackML

TrackML Solutions: (DL solution II)

- Build tracks by maximizeing the sum of probabilities

	h1	h2	h3	h4	h5
h1	-	0.8	0.2	0.9	0.4
h2	0.8	-	0.5	0.7	0.7
h3	0.2	0.5	-	0.3	0.4
h4	0.9	0.7	0.3	-	0.4
h5	0.4	0.7	0.4	0.4	-



- If threshold = 0.65

	h1	h2	h3	**h4**	h5
h1	-	0.8	-	0.9	-
h2	0.8	-	-	0.7	0.7
h3	-	-	-	-	-
h4	0.9	0.7	-	-	-
h5	-	0.7	-	-	-

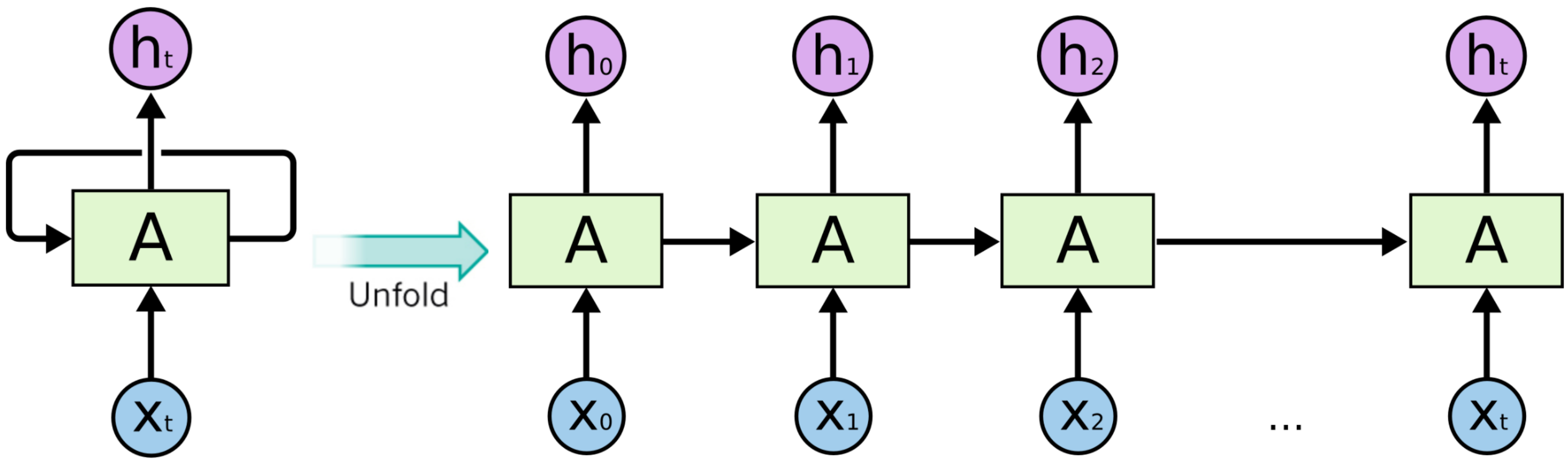
- p(h1,h2,h4) = 1.5 > 0.65 -> h1,h2,h4 same track

EPJ Web of Conferences **214**, 06037 (2019). Kaggle/TrackML

TrackML Solutions: (RNN solution I)

Recurrent Neural Networks

- The solution that ranked **12th in the challenge** is using recurrent neural networks
- **Artificial Neural Networks ANN** is also known as **feed forward network**, because each input shown to them is processed independently
- **Recurrent Neural Network RNN** processes sequences by **iterating** through the **sequence** elements and maintaining a state containing information relative to what it has seen so far



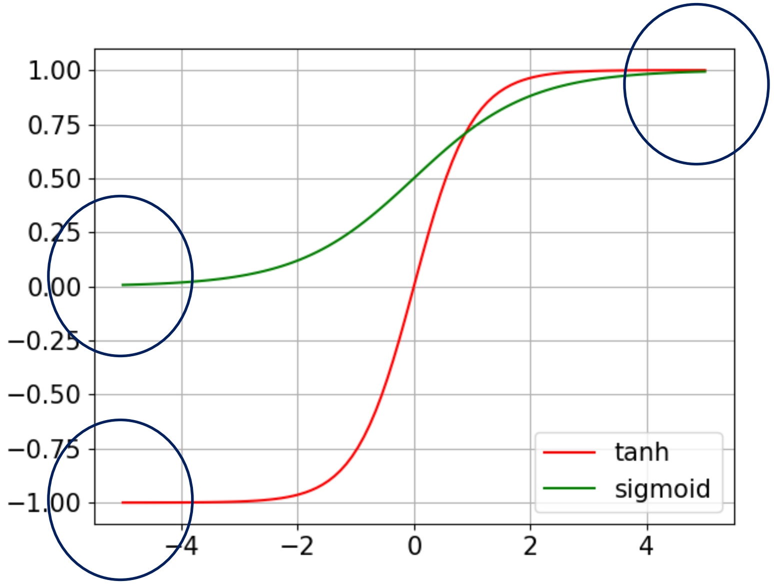
$$\mathbf{y}_{(t)} = \phi\left(\mathbf{W}_x{}^{\top}\mathbf{x}_{(t)} + \mathbf{W}_y{}^{\top}\mathbf{y}_{(t-1)} + \mathbf{b}\right)$$

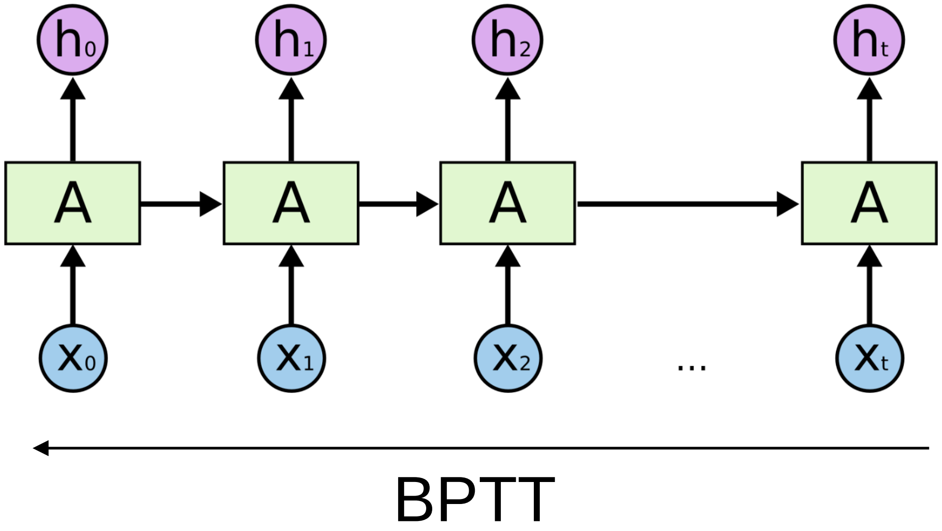
colah.github.io/Understanding LSTM Networks

TrackML Solutions: (RNN solution II)

Recurrent Neural Networks

- RNNs are trained using the **backpropagation through time (BPTT)**
- Processing RNN for **long sequences** leads to **vanishing/exploding gradient** problem.
- ***lower layers do not learn anything.***

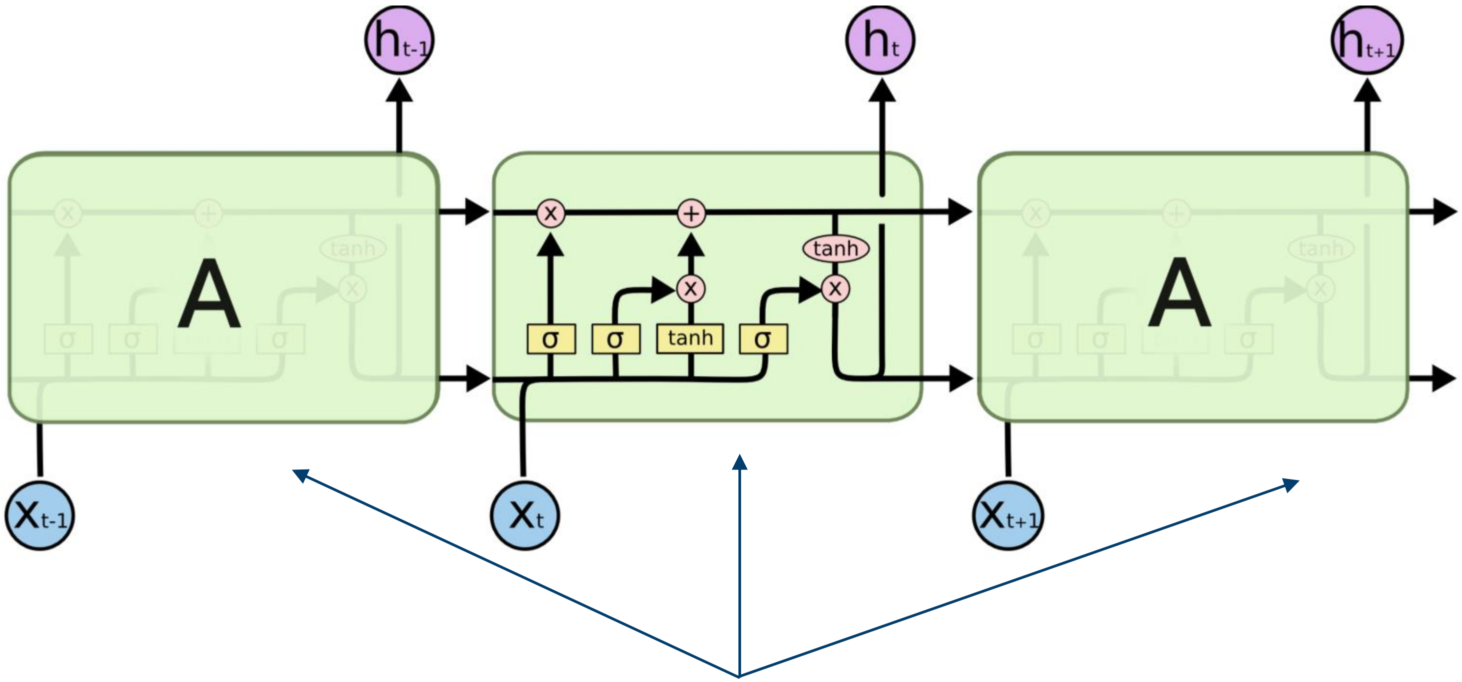




TrackML Solutions: (RNN solution III)

Long Short-Term Memory LSTM

- LSTM is a variant of RNN that overcomes the vanishing/exploding gradient.
- LSTM has **memory cells** and can process very long sequences.
- **Gates** regulate the information flow

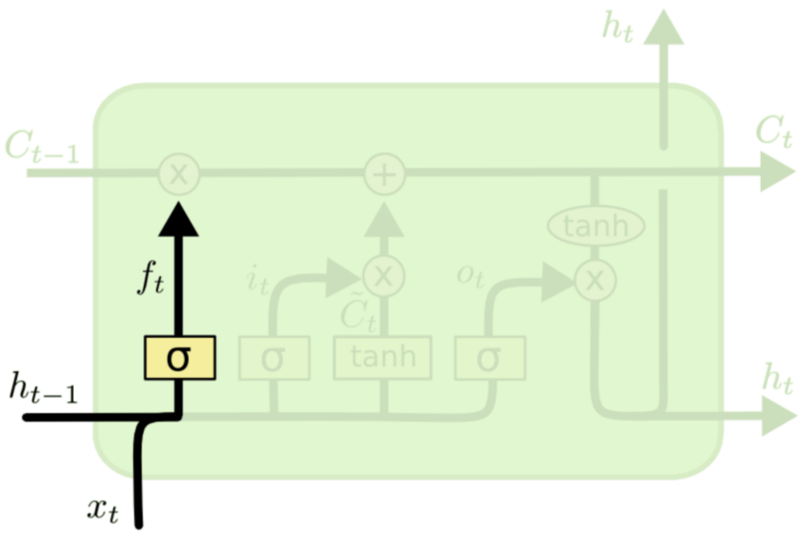


Memory cell unfolded

colah.github.io/Understanding LSTM Networks

TrackML Solutions: (RNN solution IV)
Long Short-Term Memory LSTM

➢ **Forget gate**



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

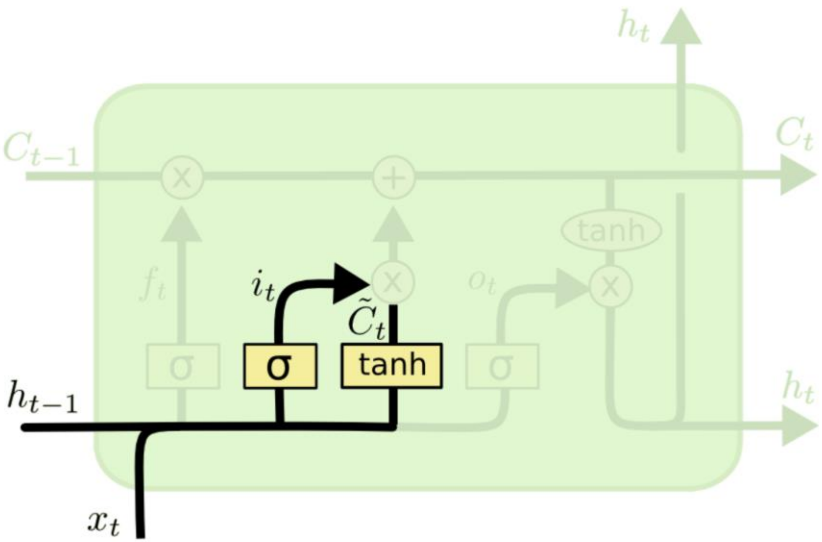
colah.github.io/Understanding LSTM Networks

JÜLICH Forschungszentrum

TrackML Solutions: (RNN solution IV)

Long Short-Term Memory LSTM

➢ **Input gate**



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

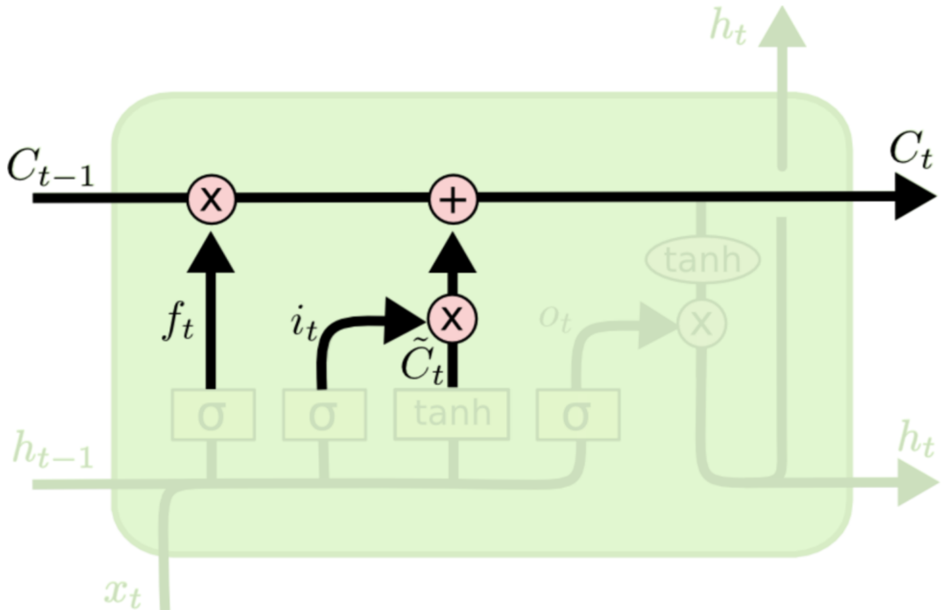
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

colah.github.io/Understanding LSTM Networks

TrackML Solutions: (RNN solution IV)

Long Short-Term Memory LSTM

➢ **Input gate (cell state)**



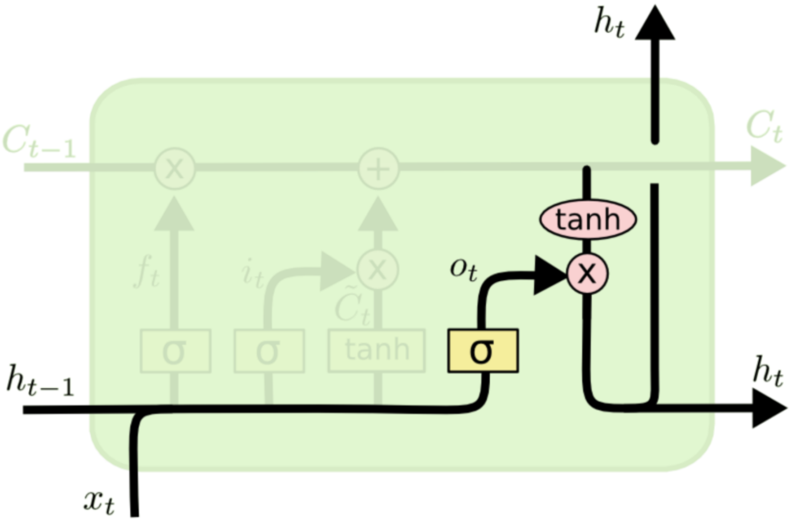
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

colah.github.io/Understanding LSTM Networks

TrackML Solutions: (RNN solution IV)

Long Short-Term Memory LSTM

- **Output gate**



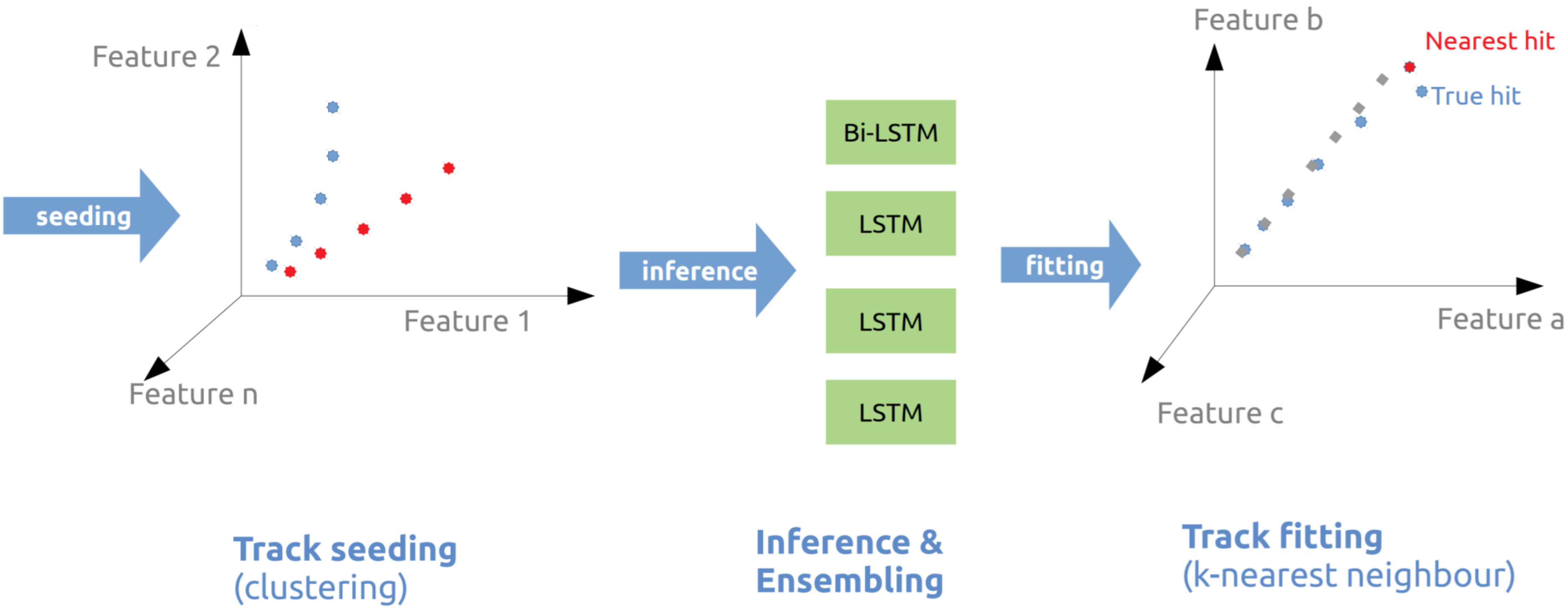
$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

colah.github.io/Understanding LSTM Networks

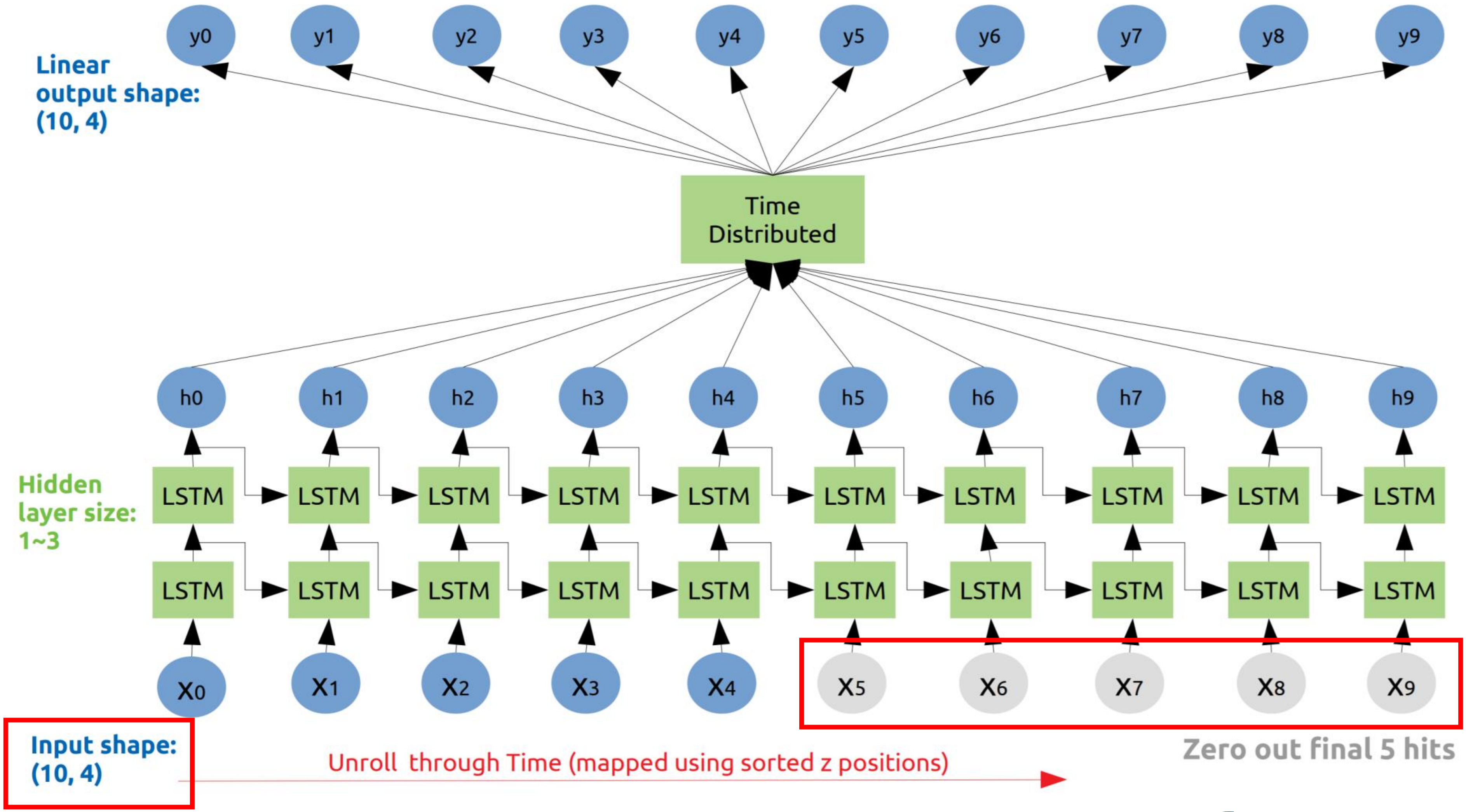
TrackML Solutions: (RNN solution V)

- Solution Pipeline:
  1. Seed finding (DBSCAN)
  2. LSTM for Track Following
  3. k-D tree search for hit association



EPJ Web of Conferences **214**, 06037 (2019). Kaggle/TrackML

TrackML Solutions: (RNN solution VI)



EPJ Web of Conferences **214**, 06037 (2019). Kaggle/TrackML

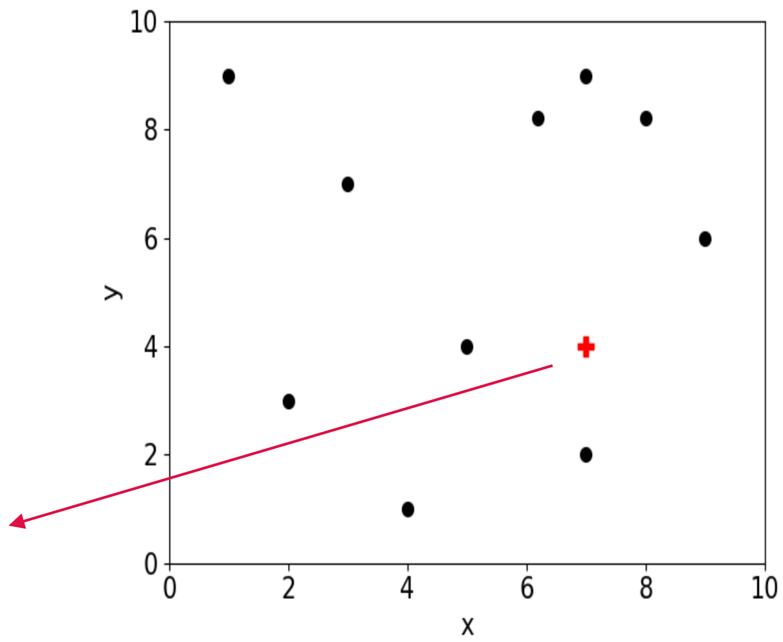
TrackML Solutions: (RNN solution VI)

- Multiple architectures LSTMs are trained
- Eensembled with averaging to provide the final prediction
- Build a binary tree to search for the **nearest neighbor (4-D Tree)**

A simple k-D Tree

(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)

Query point



EPJ Web of Conferences **214**, 06037 (2019), kNN.15 K-d tree algorithm

TrackML Solutions: (RNN solution VI)

- Multiple architectures LSTMs are trained
- Eensembled with averaging to provide the final prediction
- Build a binary tree to search for the **nearest neighbor (4-D Tree)**

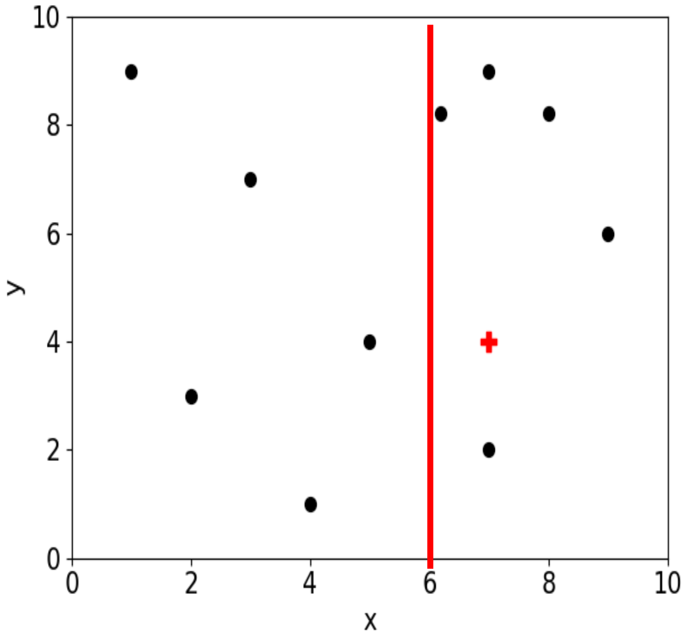
A simple k-D Tree

(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)

$\boxed{x \geq 6}$

(1,9), (2,3), (4,1), (3,7), (5,4),

(6,8), (7,2), (8,8), (7,9) (9,6)



EPJ Web of Conferences **214**, 06037 (2019), kNN.15 K-d tree algorithm

TrackML Solutions: (RNN solution VI)

- Multiple architectures LSTMs are trained
- Eensembled with averaging to provide the final prediction
- Build a binary tree to search for the **nearest neighbor (4-D Tree)**

A simple k-D Tree

(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)

$x \geq 6$

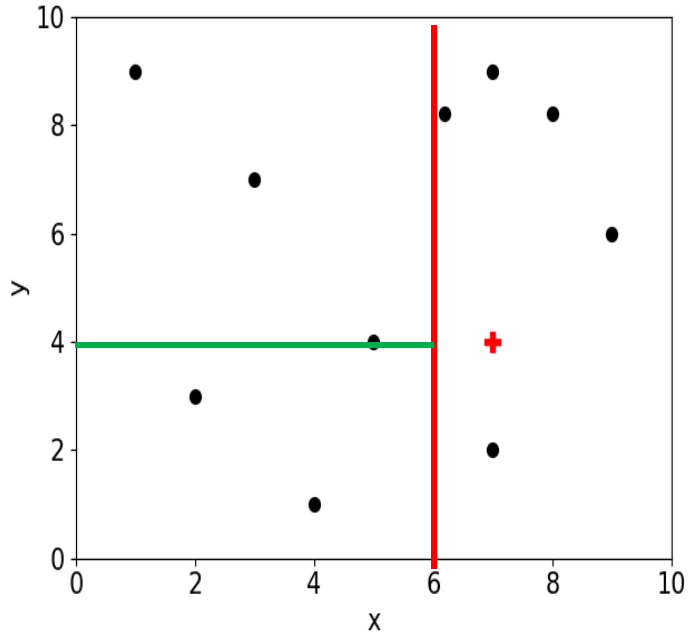
(1,9), (2,3), (4,1), (3,7), (5,4),

(6,8), (7,2), (8,8), (7,9) (9,6)

$y \geq 4$

(2,3), (4,1)

(1,9), (3,7), (5,4)



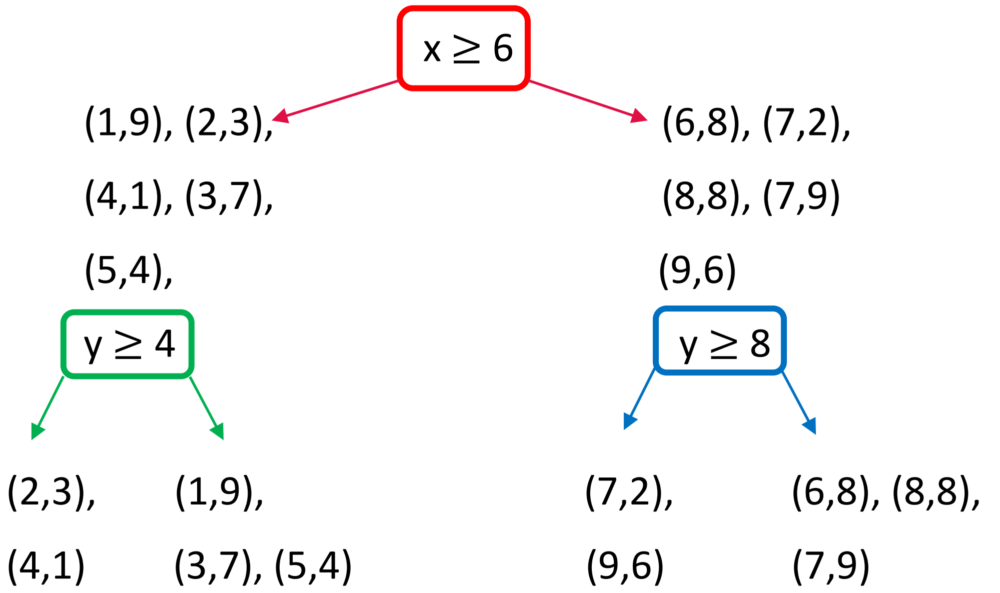
EPJ Web of Conferences **214**, 06037 (2019), kNN.15 K-d tree algorithm

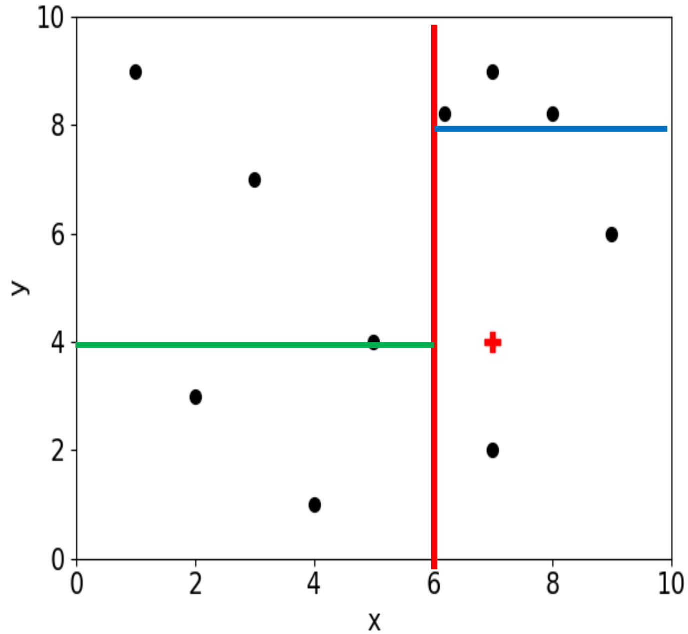
TrackML Solutions: (RNN solution VI)

- Multiple architectures LSTMs are trained
- Eensembled with averaging to provide the final prediction
- Build a binary tree to search for the **nearest neighbor (4-D Tree)**

A simple k-D Tree

(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)





EPJ Web of Conferences **214**, 06037 (2019), kNN.15 K-d tree algorithm

TrackML Solutions: (RNN solution VI)

- Multiple architectures LSTMs are trained
- Eensembled with averaging to provide the final prediction
- Build a binary tree to search for the **nearest neighbor (4-D Tree)**

A simple k-D Tree

(1,9), (2,3), (4,1), (3,7), (5,4), (6,8), (7,2), (8,8), (7,9), (9,6)

$x \geq 6$

(1,9), (2,3), (4,1), (3,7), (5,4),

(6,8), (7,2), (8,8), (7,9) (9,6)

$y \geq 4$

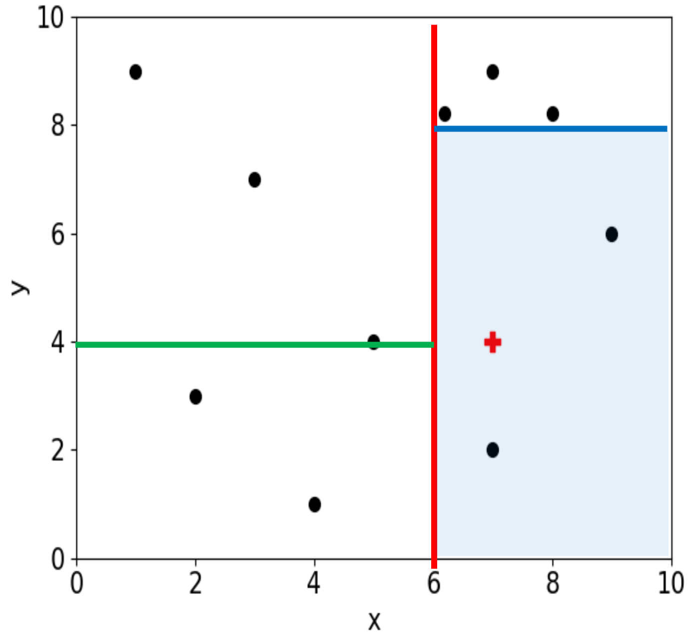
(2,3), (4,1)

(1,9), (3,7), (5,4)

$y \geq 8$

(7,2), (9,6)

(6,8), (8,8), (7,9)

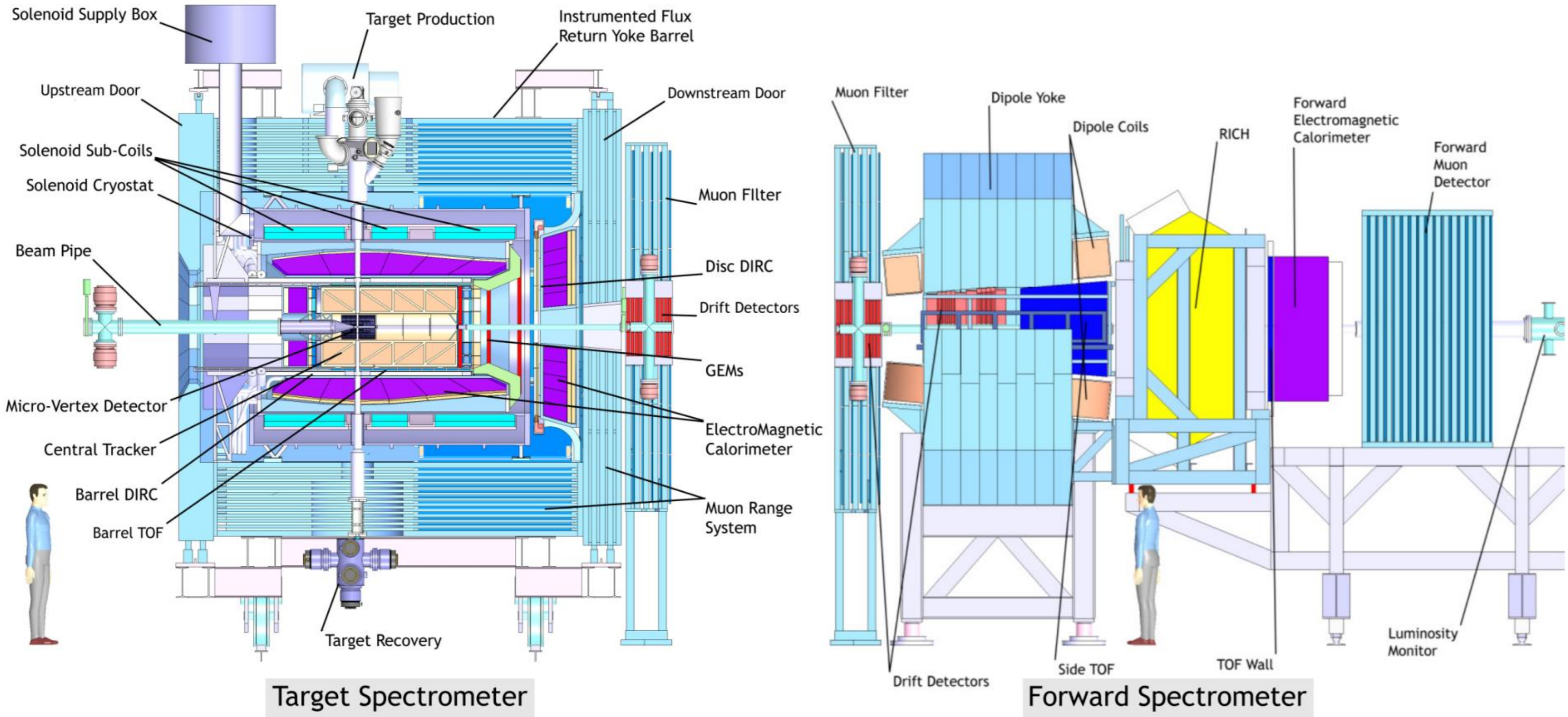


EPJ Web of Conferences **214**, 06037 (2019), kNN.15 K-d tree algorithm

ML Based Track Finding at PANDA FTS

PANDA Detector:

antiProton ANnihillation at DArmstadt



PANDA FTS:

Forward Tracking Stations:

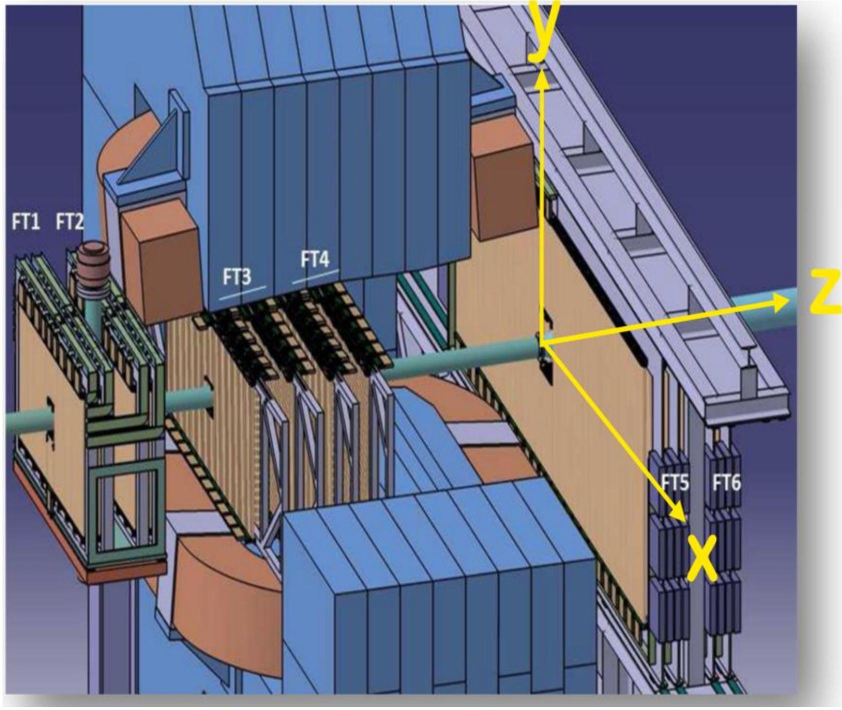
➢ Straw tubes, same as in the barrel, vertically arranged in double layers

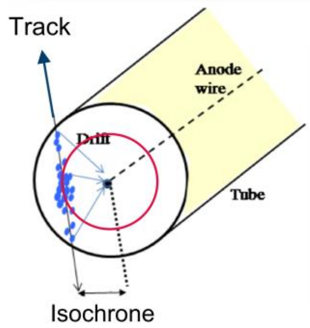
➢ 3 stations with 2 chambers each
- FTS1&2 : No magnetic field
- FTS3&4 : Inside the field (2Tm)
- FTS5&6 : No magnetic field

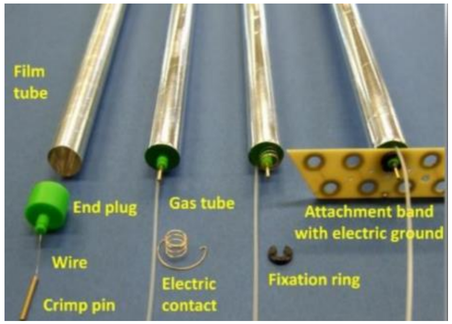
➢ 8 double layers per chamber.
➢ Orientations 0°/+5°/-5°/0° per chamber

➢ Tracks are defined by distance of closest approach to the wire (Isochrones)

➢ **Inputs: Wire position (hits), Isochrones, …**





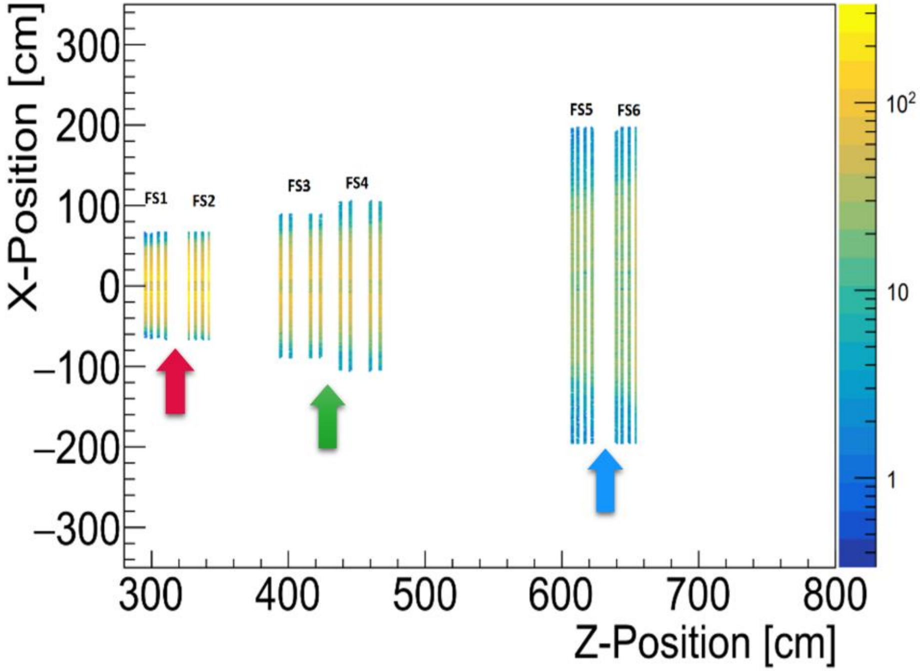


PANDA FTS Algorithm I:

Local approach:

I. **Create track segments (tracklets) using Artificial Neural Network**

FTS1 & FTS2
FTS3 & FTS4
FTS5 & FTS6



II. **Connect the segments using LSTM**

Make all possible combinations of tracklets

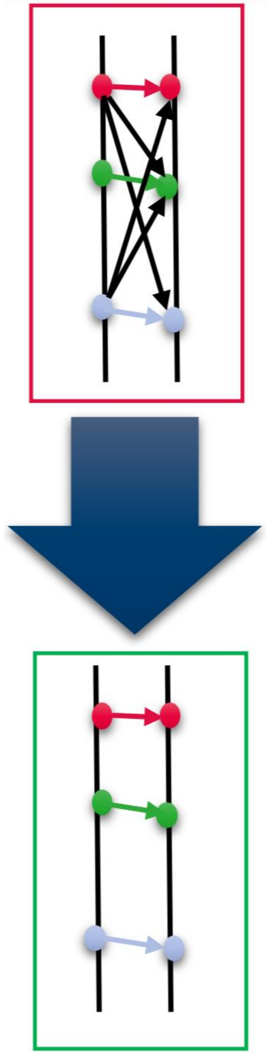
PANDA FTS Algorithm I: Step I

- All possible combinations of hit pairs **ONLY adjacent layers**
- **ONLY vertical layers**
- Network predict the quality of the pair
- **Input Hit Pair(x,z,r) -> DNN ->** Probability
- Training data -> 5 tracks/event (particle gun)

- Clustering using the probability output (threshold)

1. if Prob(h_1,h_2) > threshold and,
2. Prob(h_2,h_3) > threshold

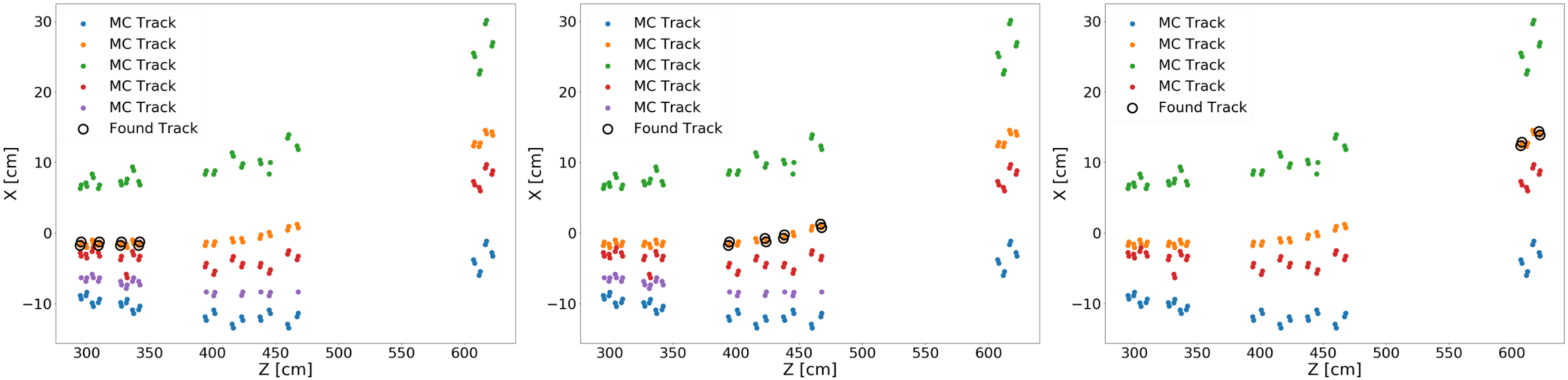
   (h_1, h_2, h_3) on the **same track**



Accuracy ~ 96%

Machine Learning For Track Finding, (CTD/WIT 2019)

PANDA FTS Algorithm I: Step I

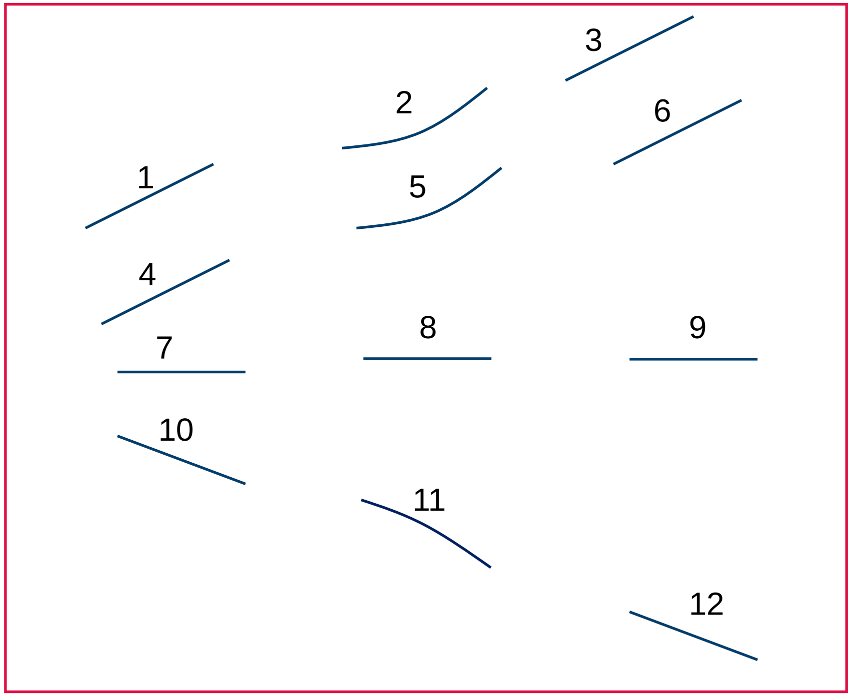


PANDA FTS Algorithm I: Step II

- All possible combinations of tracklets

  [1,2,3], [1,2,6], [1,5,3], …

- LSTM is used as a classification model ~ 98%
- Network predict the quality of the track candidate



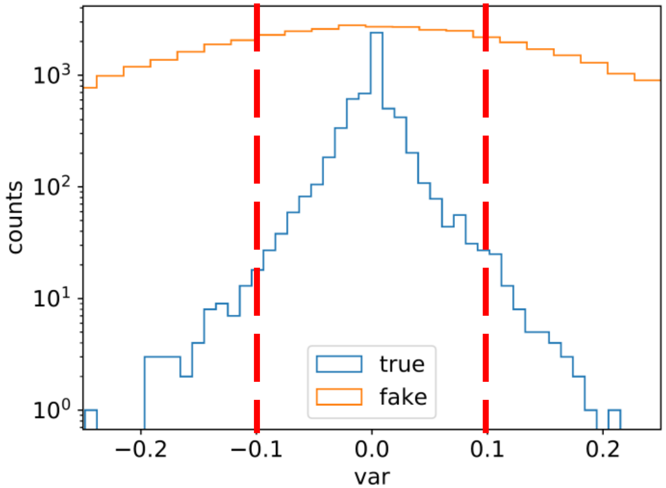
Machine Learning For Track Finding, (CTD/WIT 2019)

PANDA FTS Algorithm I: Step II

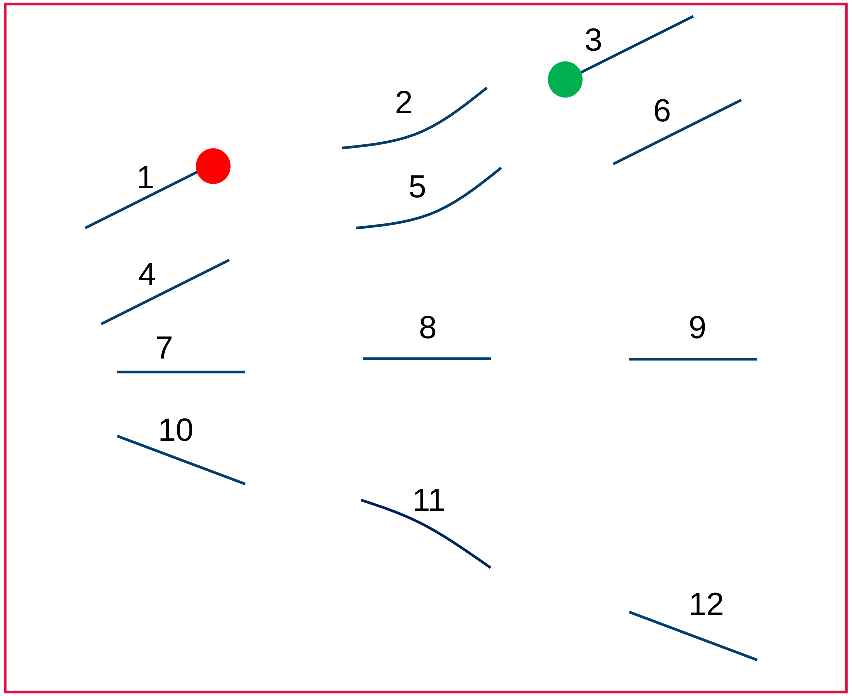
- All possible combinations of tracklets

[1,2,3], [1,2,6], [1,5,3], …

- Network predict the quality of the track candidate
- **Preprocess -> Balance the training set**

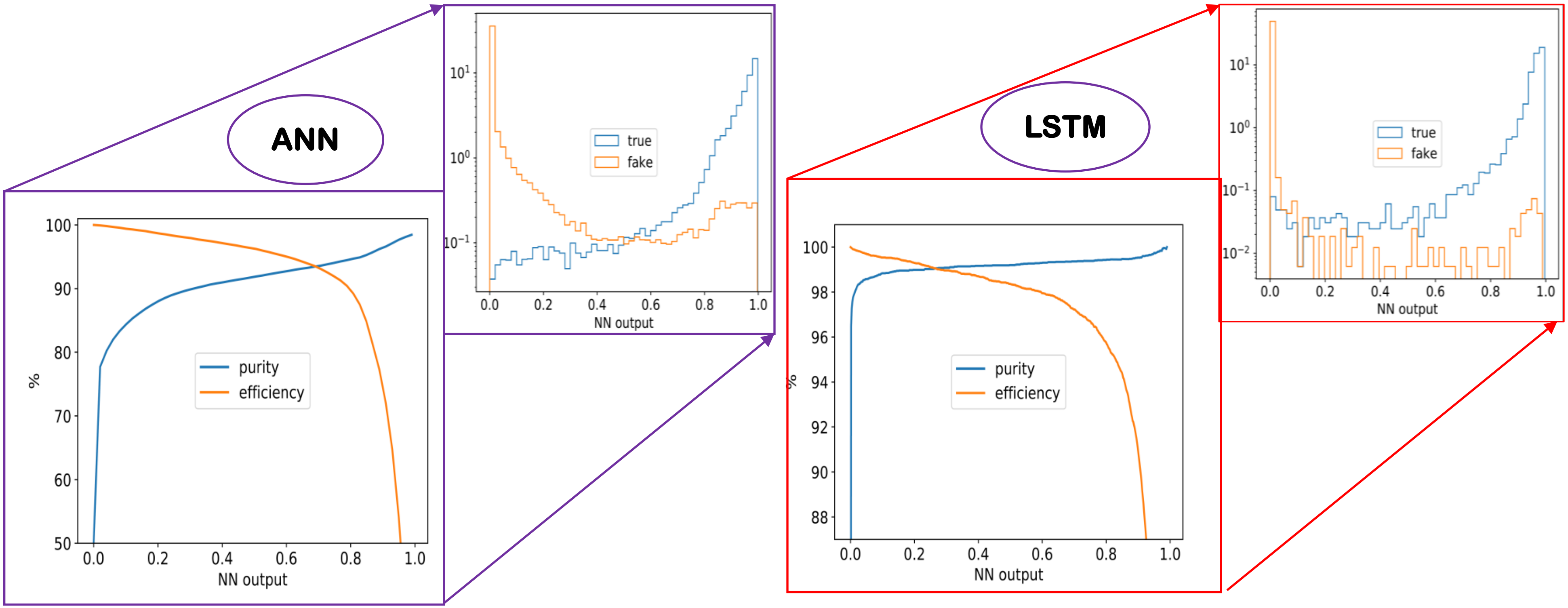


- $a = z/x$
- var = $a_2 - a_1$
- **R(fake/true) ~ 8 --> 4**



Machine Learning For Track Finding, (CTD/WIT 2019)

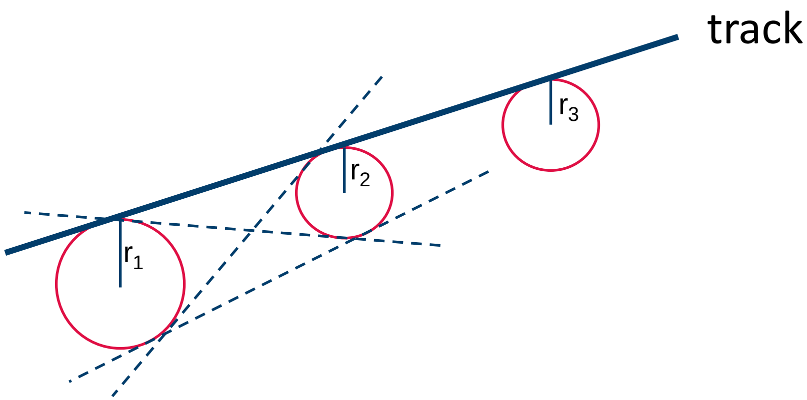
PANDA FTS Algorithm I: Optimizing Probability Cuts:

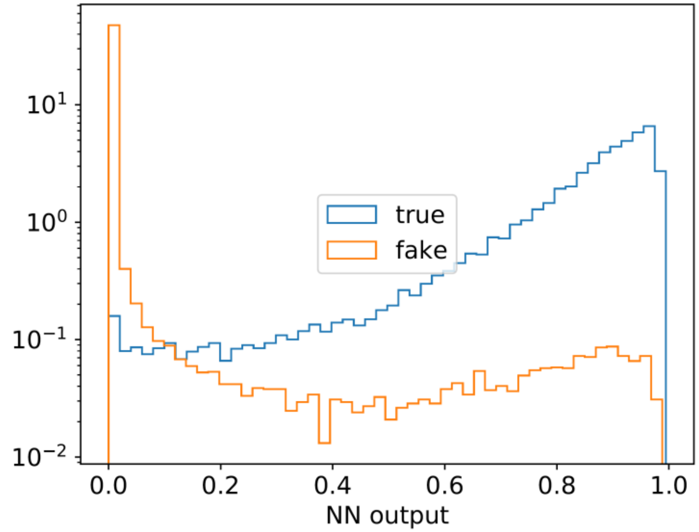


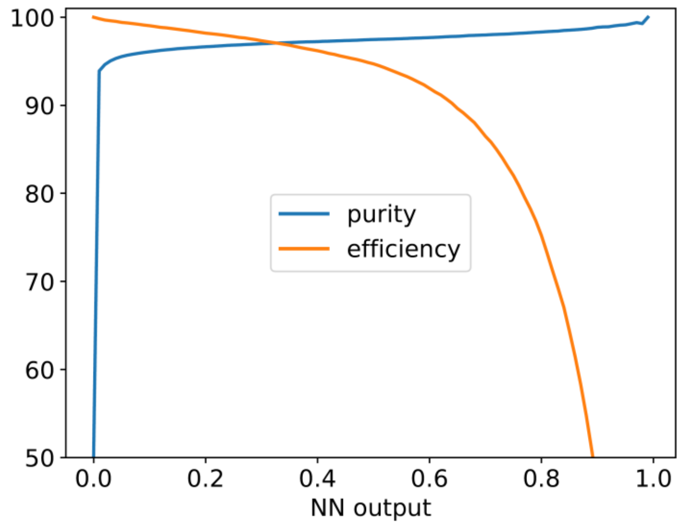
- ➢Purity = true that pass the cut / all that pass the cut
- ➢efficiency = true that pass the cut / all true
- ➢Overall **tracking efficiency** ranging from ~ 80 ~ 100 %

PANDA FTS Algorithm I: Resolving Ambiguity:

- ➢All possible combinations of triplets **ONLY adjacent layers**
- ➢**ONLY vertical layers**
- ➢Network predict the quality of the triplet
- ➢**Input Hit Triplet(x,z,r) -> DNN ->** Probability





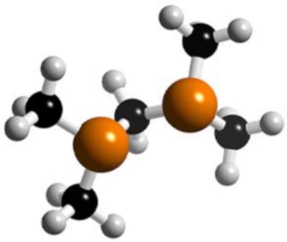


- ➢Overall **tracking efficiency** comparable to hit-pairs.

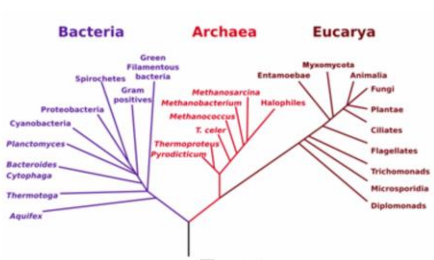
Tracking Using Graph Neural Networks

Geometric Deep Learning GDL

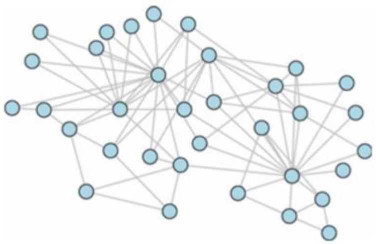
- Images, text, audio, and many others are called **euclidean data**
- **Non-euclidean data** can represent more complex items and concepts with more accuracy than 1D or 2D representation
- **GDL** is about building neural networks that can learn from **non-euclidean data**
- **Non-euclidean data** can be resented as a **Graph**



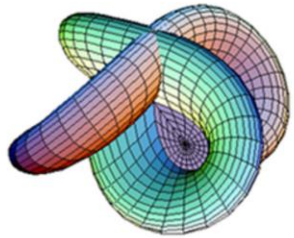
Molecules



Trees



Networks



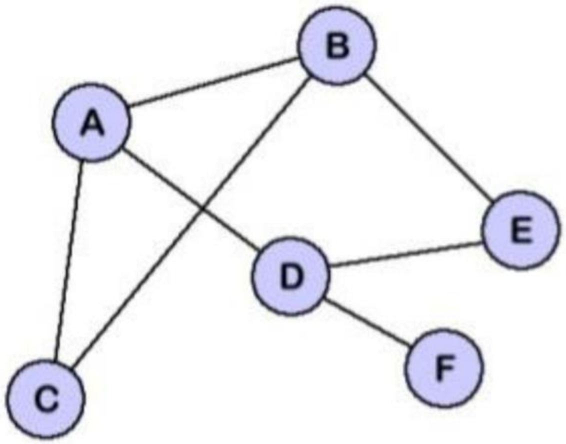
Manifolds

What is Geometric Deep Learning?, Flawnson Tong, medium.com, 2019

Graph Concept

- A **graph** is a data structure comprising of **nodes** (**vertices**) and **edges** connecting nodes

- **Graph = G(X,E)** can be resented by a matrix (e.g. **Adjacency Matrix**)



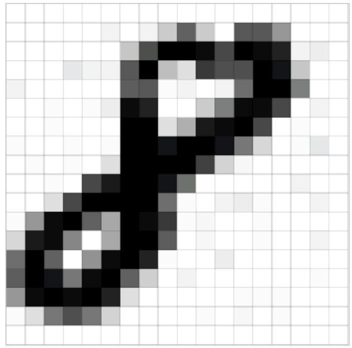
$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

- Graph can be directed or undirected

- The neural network itself can be viewed as a graph, where **nodes are neurons** and **edges are weights**

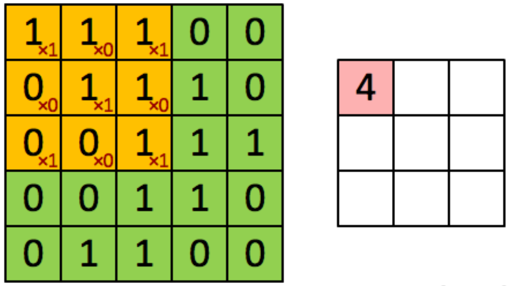
Convolution Operation

- An image can be represented as a matrix of pixel values
- The purpose of Convolution is to **extract features** from the input image



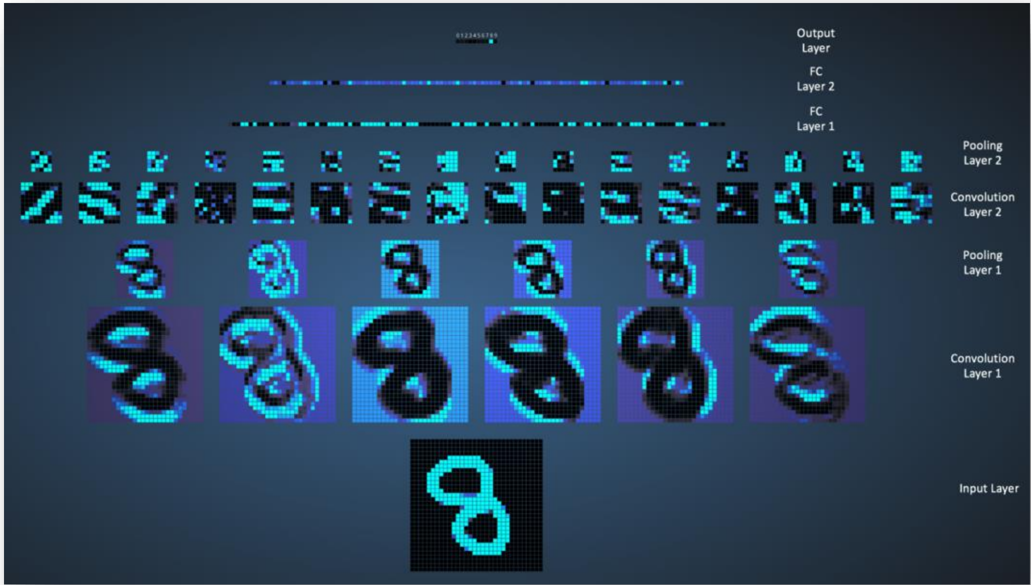
$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Matrix Multiplication



Image

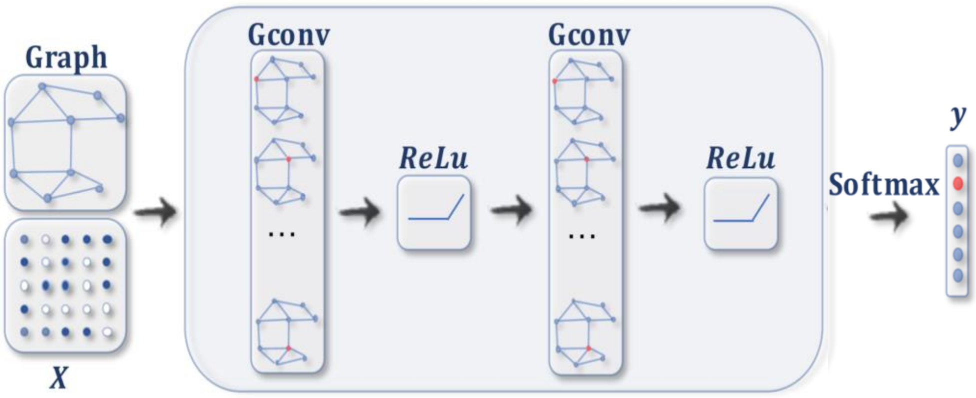
Convolved Feature



1. An Intuitive Explanation of Convolutional Neural Networks 2016

Graph Neural Networks GNN

- Motivated by CNN and graph embeddings
- **RecGNNs, ConvGNNs, GAEs, …**

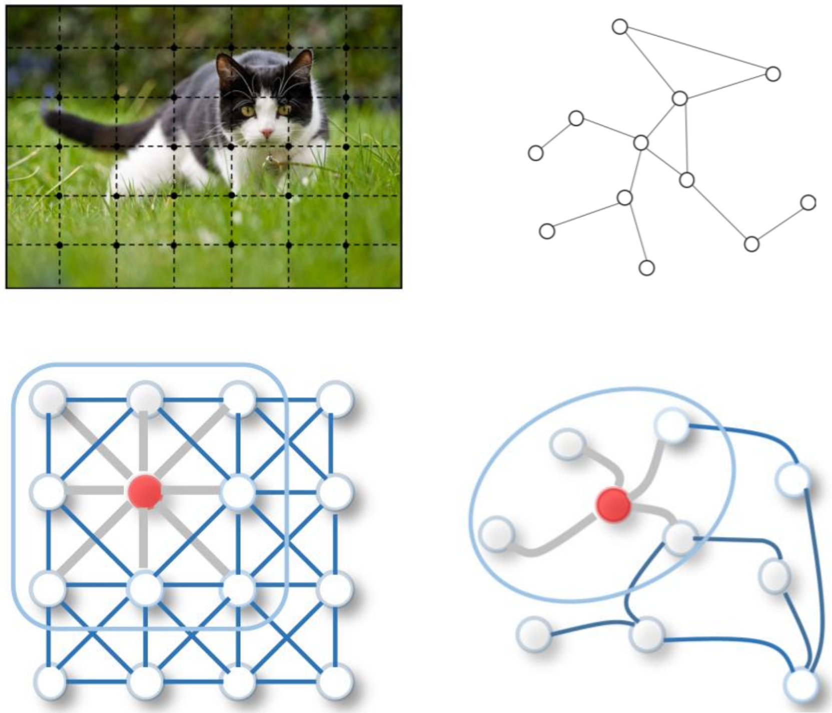


The target of GNN is to learn a state embedding (neighborhood relations)

$$H^{t+1} = F(X, H^{t})$$

Euclidean

Non-Euclidean



Tasks: Node-level, **Edge-level**, Graph-level.

1. Graph Neural Networks: A Review of Methods and Applications Jie Zhou 2019
2. A Comprehensive Survey on Graph Neural Networks Zonghan Wu 2019

GNN applied to FTS:

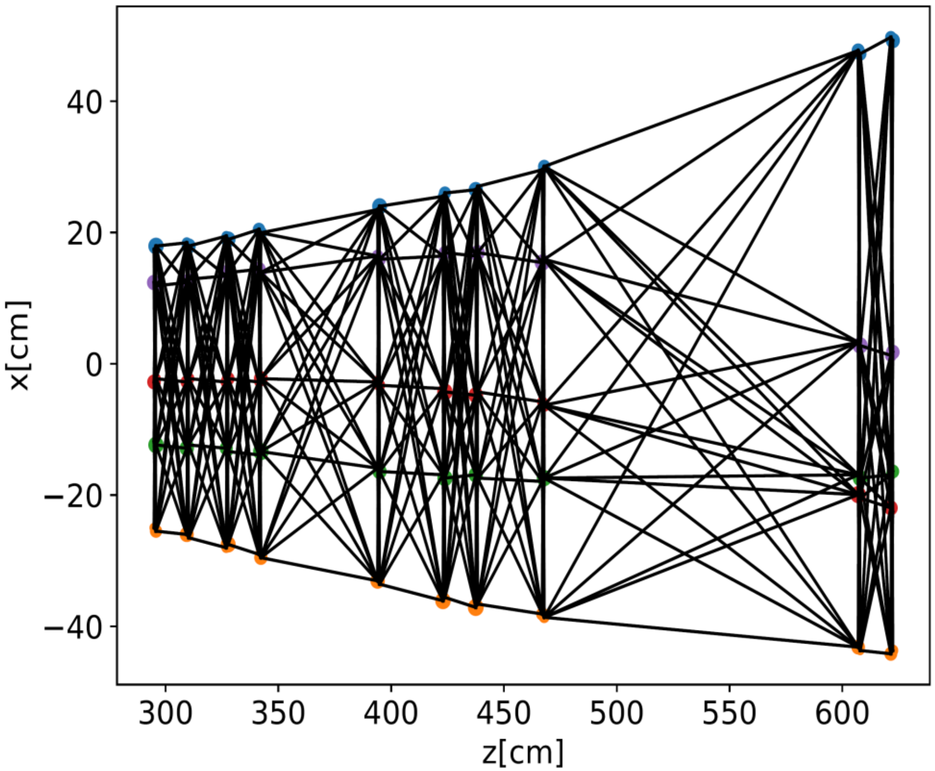
- Global approach
- GNN is used as a binary classifier (**hit-pairs classification or edge classification**)
- Input is a graph (FTS hits of one event).
- Two main components: **edge network** and **node network**
- Edge network uses the node features to compute edge weights
- Node network aggregates node features with the edge weights and updates node features
- With each graph iteration, the model propagates information through the graph, strengthens important connections, and weakens useless ones.

node features = [x, z, isochrone]
graph iterations = 3

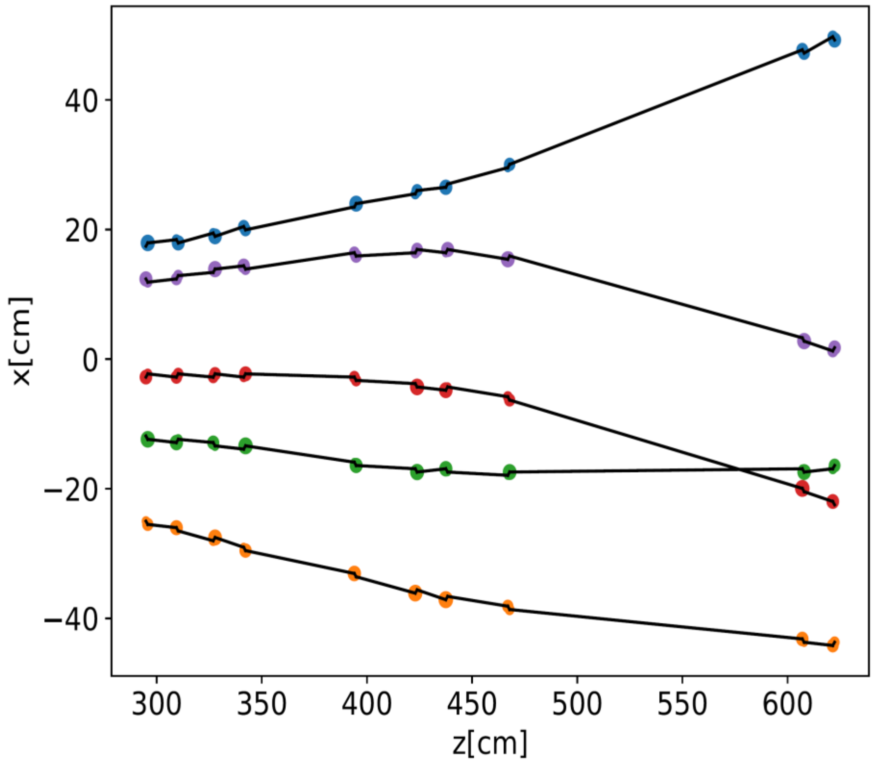
1. Novel deep learning methods for track reconstruction Steve Farrell, CTD/WIT 2018

GNN applied to FTS:

Input Graph

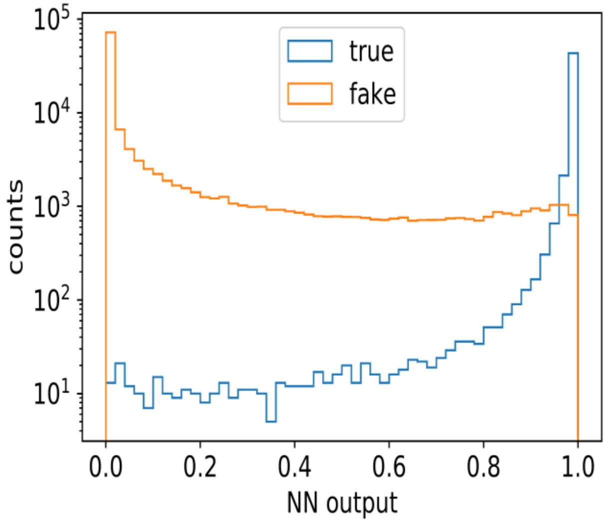


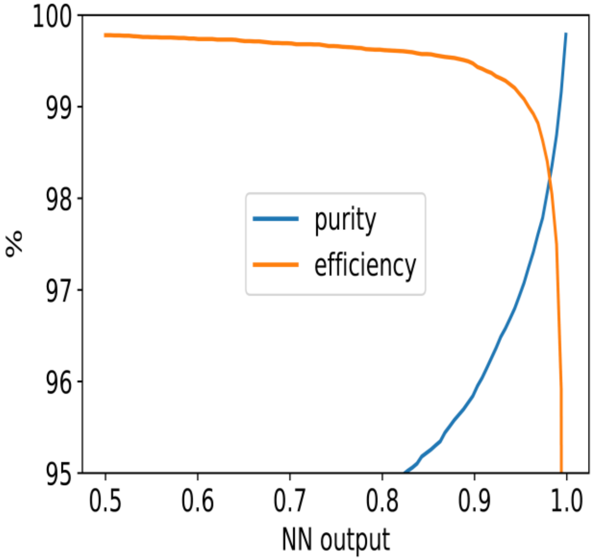
Ideal Graph



Graph Convolution Networks for FTS, Waleed Esmail 2020

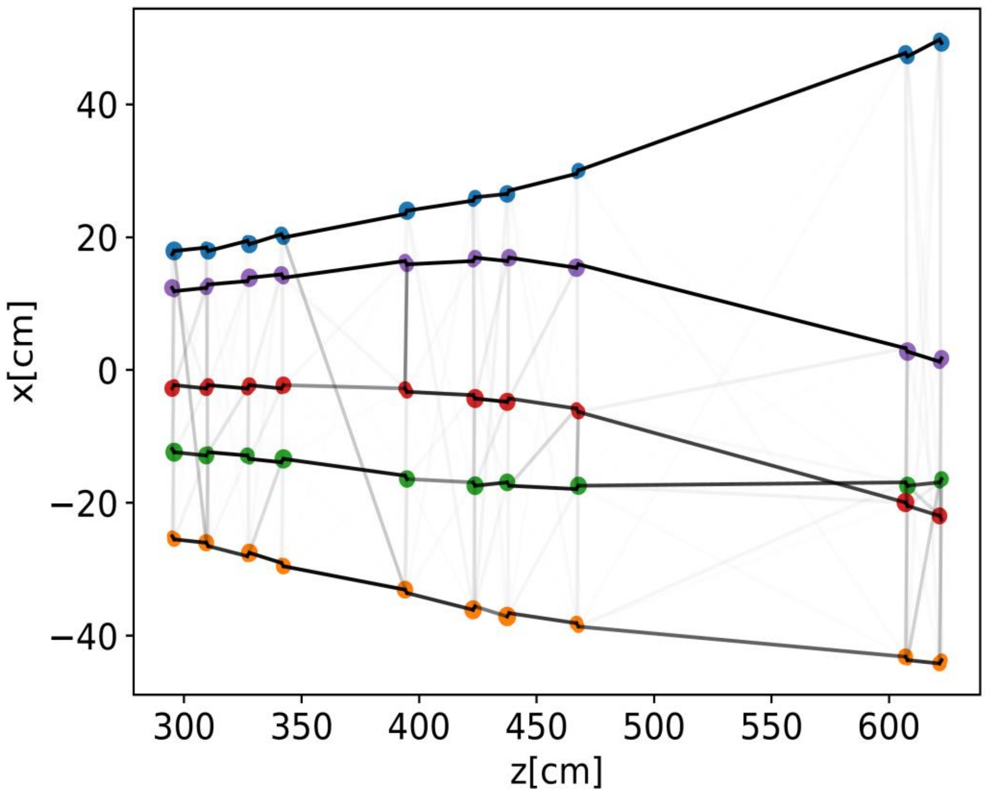
GNN applied to FTS:





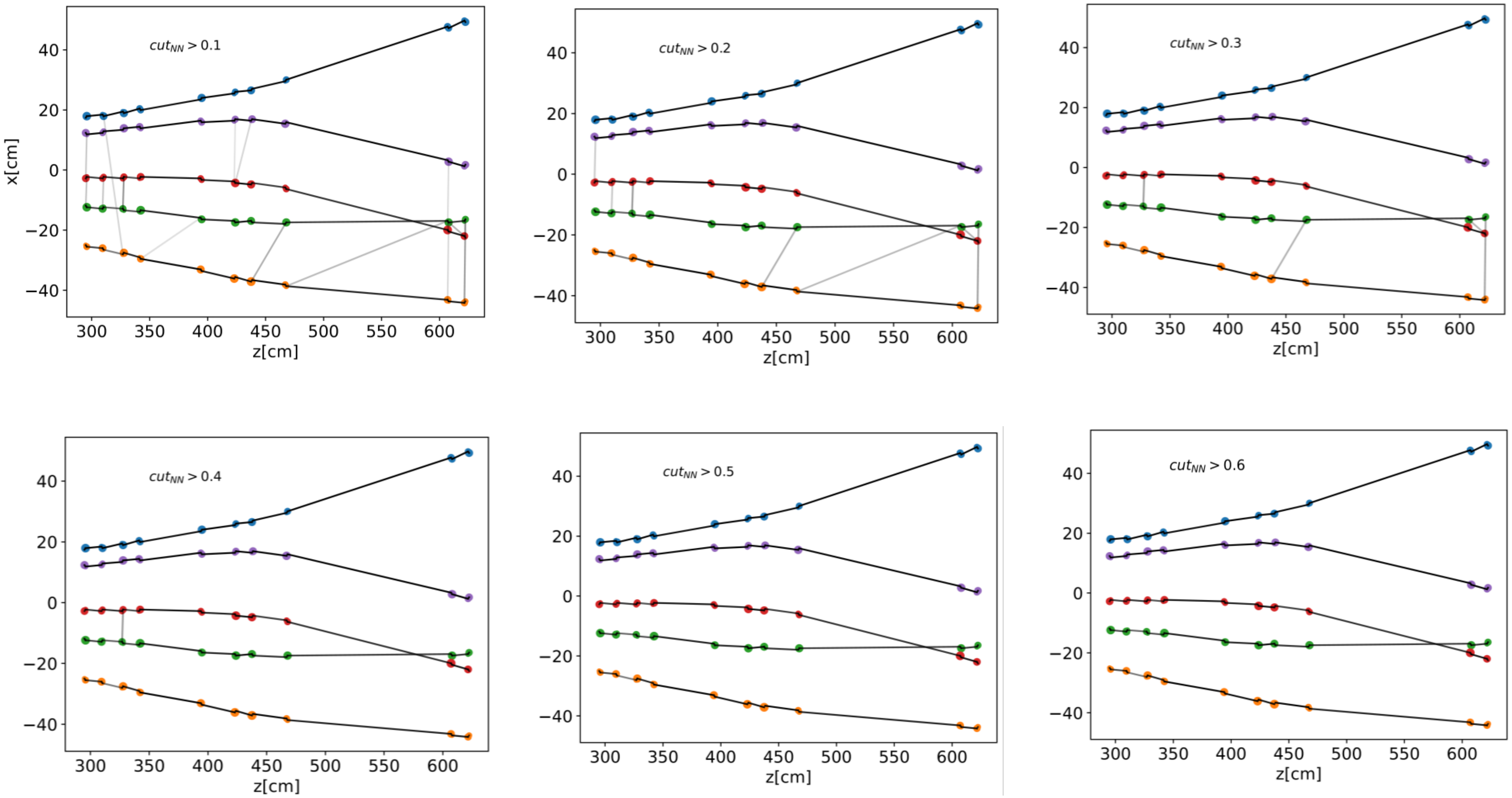
- **Accuracy ~ 99%**

An output graph



Graph Convolution Networks for FTS, Waleed Esmail 2020

GNN applied to FTS:



Graph Convolution Networks for FTS, Waleed Esmail 2020

GNN applied to FTS:

- Finding tracks is finding graph connected components (subgraphs)
- A traversal algorithm, starting at vertex v_i then visit all vertices.

```
Mark all vertices as not visited
For every vertex v:
    if v is not visited call DFS()



DFS()
    Mark v as visited
    store v in a list
        For every edge (adjacent vertices v and u):
            if u is not visited, then recursively call DFS()
```

Tracking QA:

1. Track efficiency

- How many MC tracks have been found by track finderfinder

2. Purity

- Belong all hits of one found track belong to one MC track.

3. Ghosts

- How many hits not belonging to an MC track have been found

4. Partially found

- Not all hits belonging to one track have been found but all hits belong to one MC track

5. Spurious found

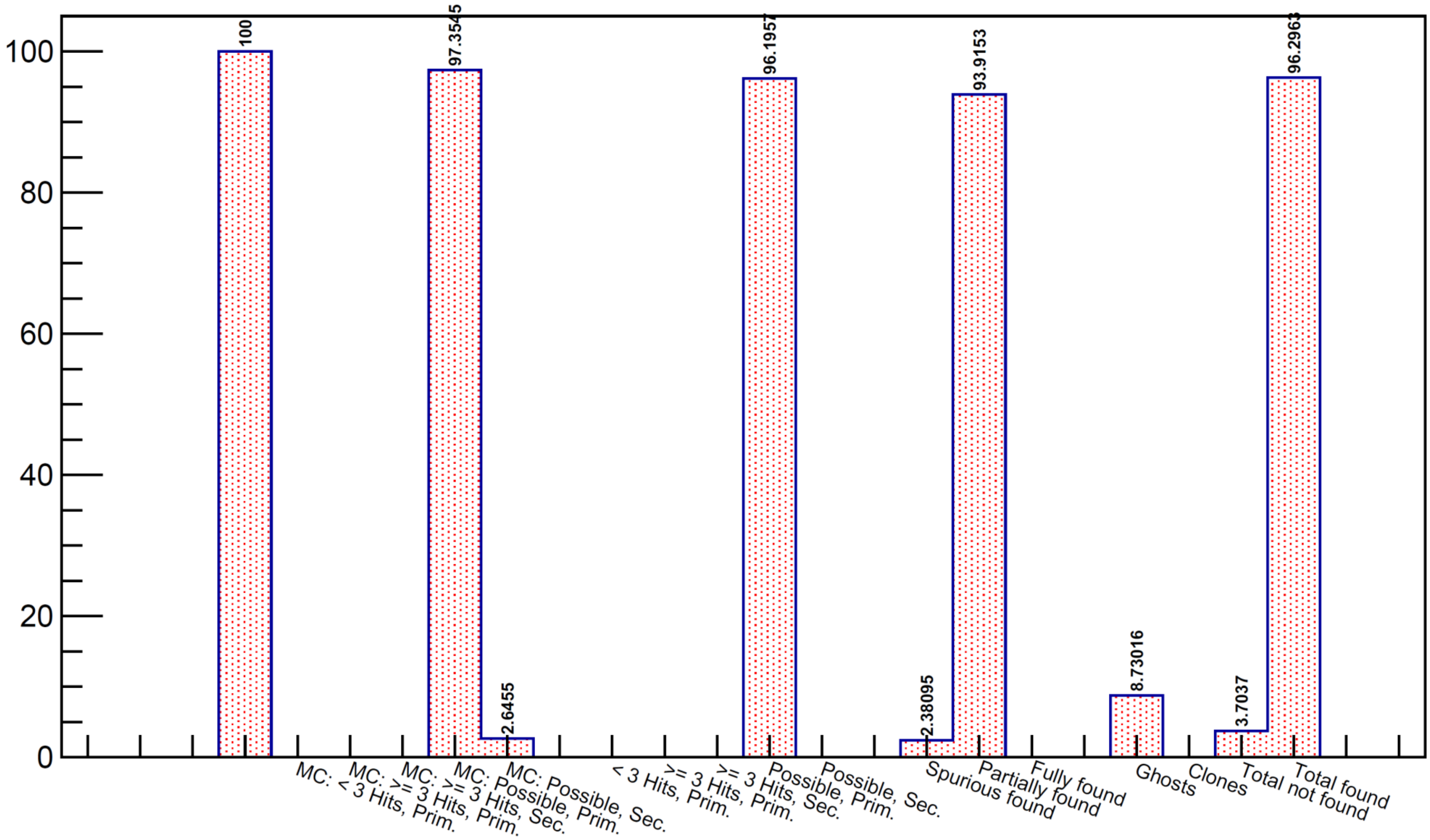
- > 70% found hits belong to one MC track

6. Fully found

- 100 % of MC hits have been found and no other hits are part of the track

Graph Convolution Networks for FTS, Waleed Esmail 2020

Tracking QA:

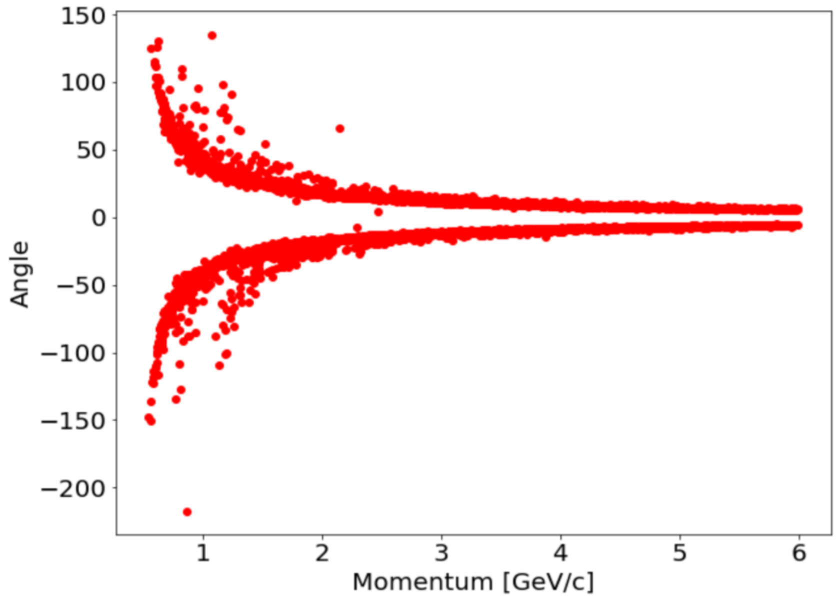


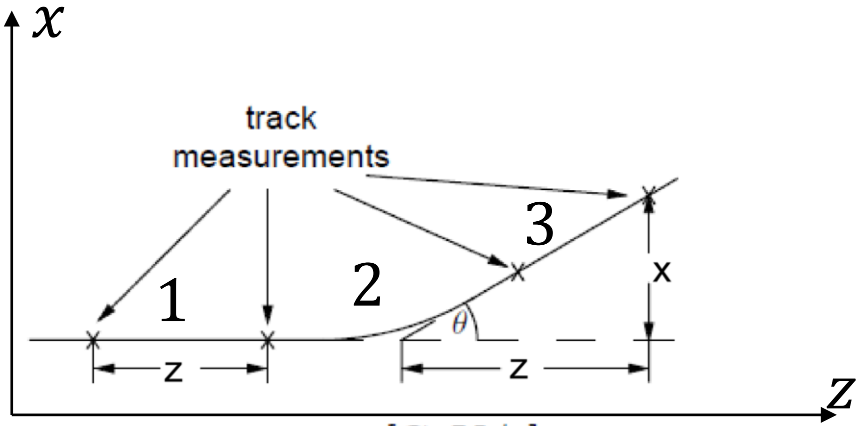
Graph Convolution Networks for FTS, Waleed Esmail 2020

Track Fitting:

- Track Reconstruction = Track Finding + **Track Fitting**
- Standard approach in many experiments is the **Kalman Filter**
- Kalman Filter needs starting values (seed)
- Track Fitting delivers parameters needed for physics analysis (e.g. Momentum)

- Momentum is estimated from track curvature





$$\rho\ [\mathrm{m}] = \frac{p\,[\mathrm{GeV}/c]}{0.3\,B\ [\mathrm{T}]}$$

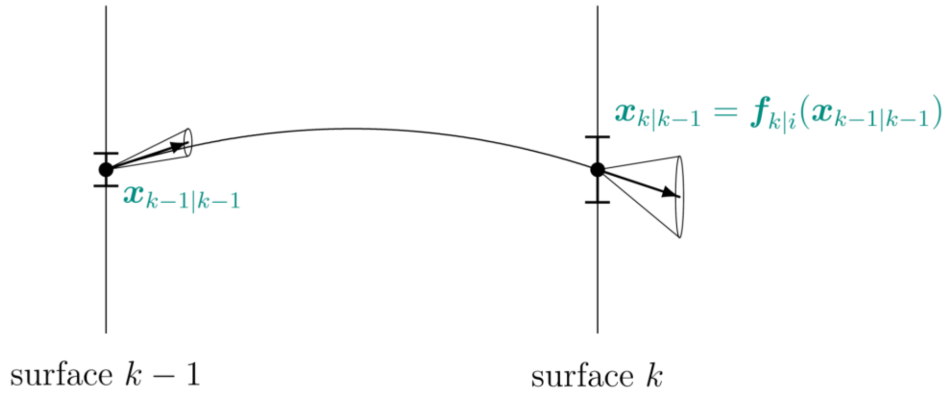
$$\theta = \frac{L}{\rho} = \frac{L}{p} eB$$

Track Fitting:

- **Track Model** is a parametrization of the track (**state vector**)

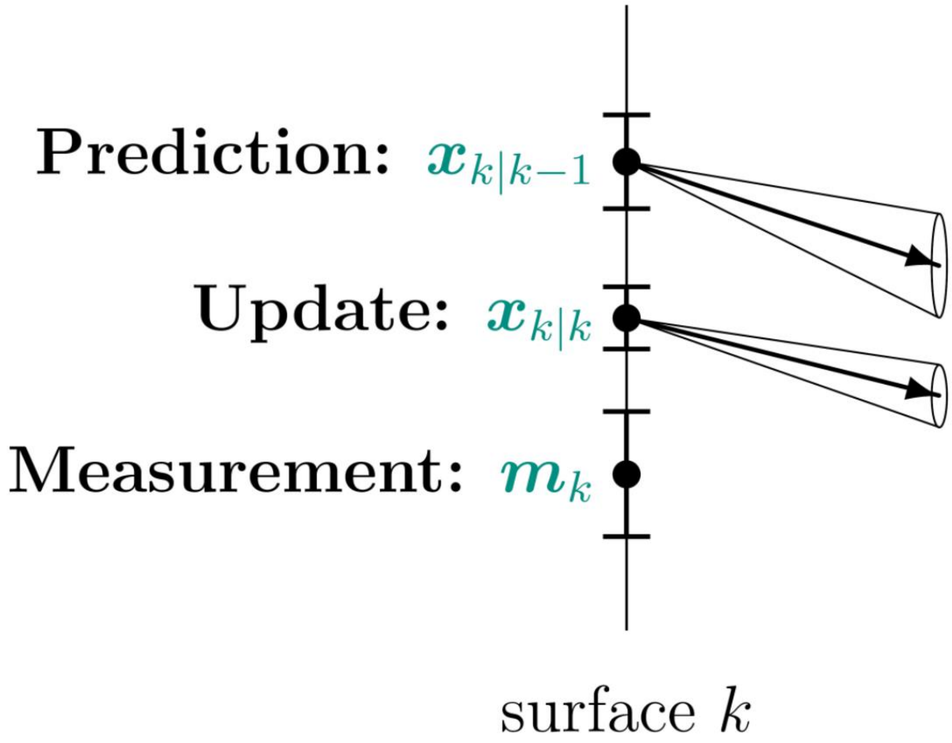
$$\begin{pmatrix} x \\ y \\ t_x \\ t_y \\ q/p \end{pmatrix} \quad \text{with } t_x = \frac{\partial x}{\partial z} \text{ and } t_y = \frac{\partial y}{\partial z}$$

- Kalman filter has two steps that are repeated **prediction** and **update**



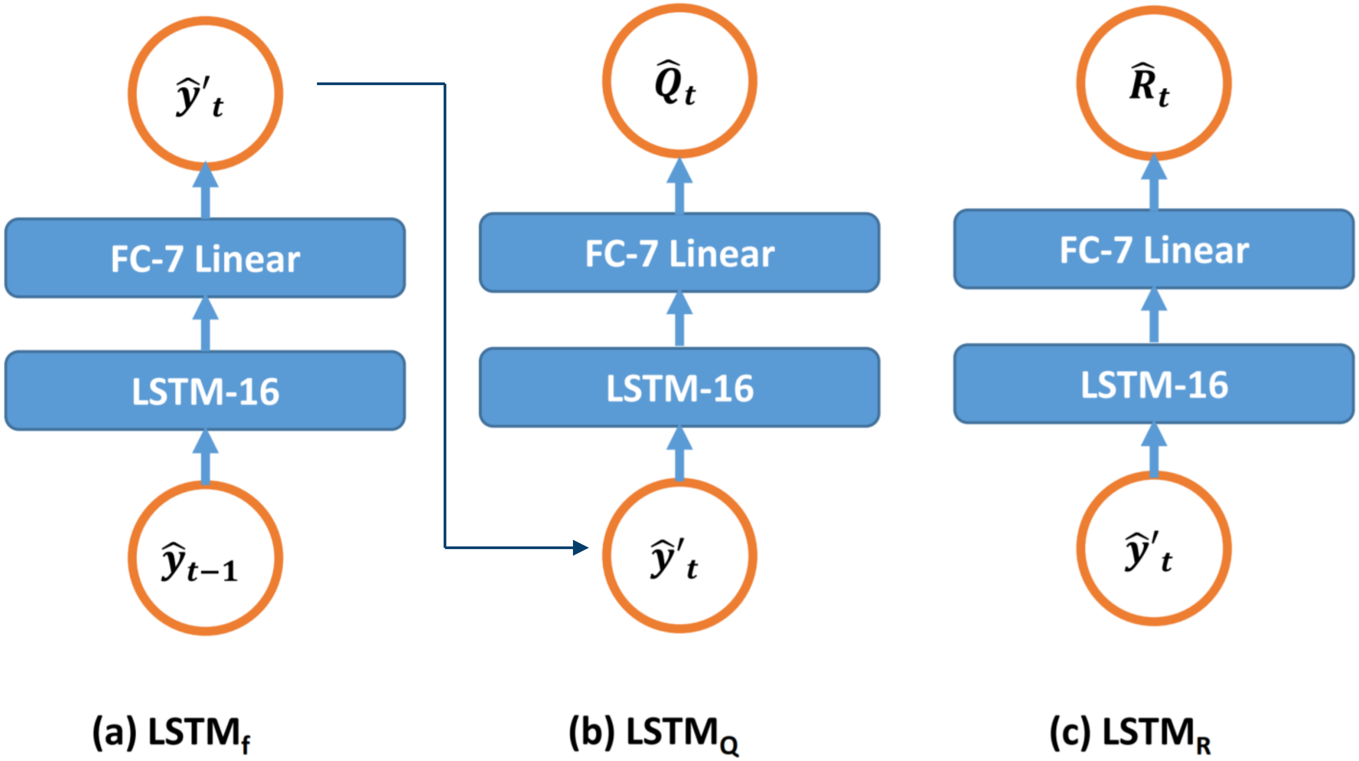
Prediction step of the Kalman filter

Track Fitting:



Deep Learning Track Fitting:

- **Inputs:** seed state vector **--> Model -->** best estimate of state vector
- **LSTM model Kalman update**

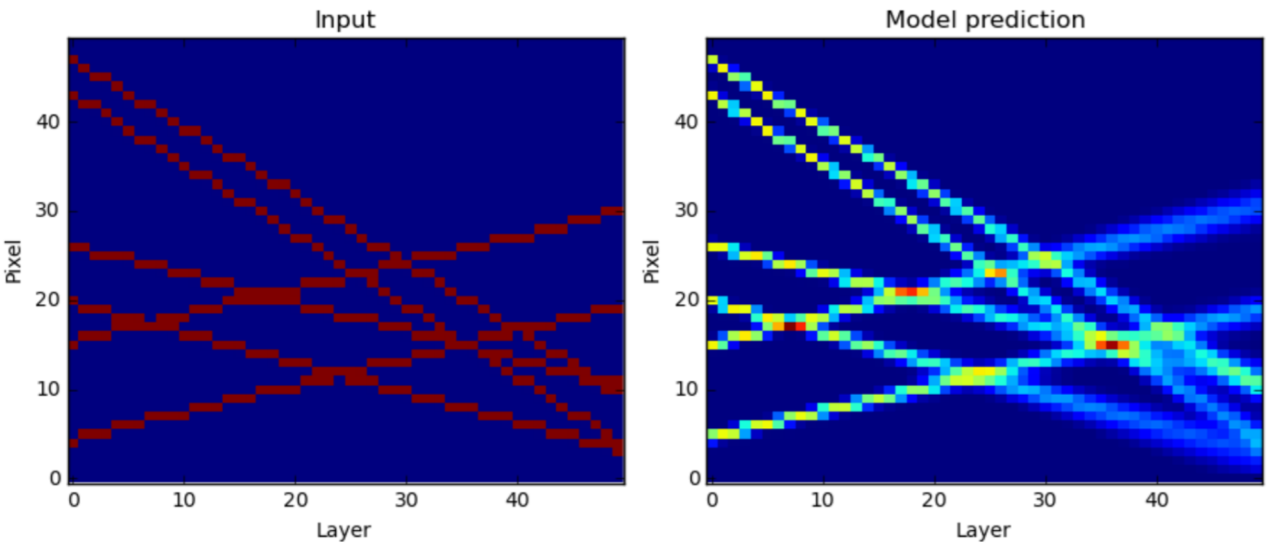


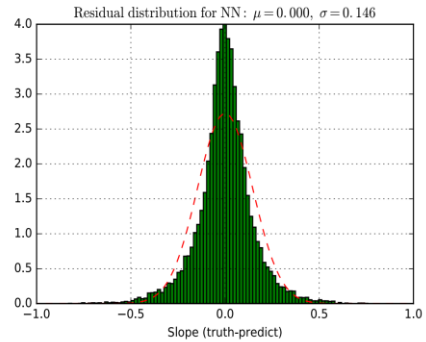
Long Short-Term Memory Kalman Filters: Recurrent Neural Estimators for Pose Regularization, H. Coskun 2017

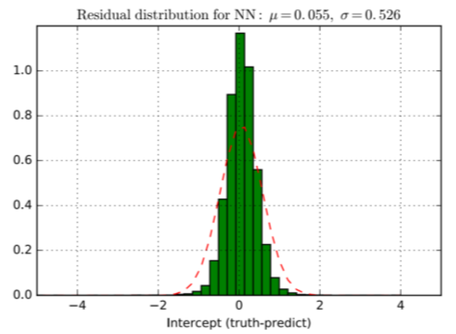
Deep Learning Track Fitting:

- The **HEP.TrkX** project is exploring the applicability of advanced machine learning algorithms to HL-LHC track reconstruction
- **CNN + LSTM** model
- Toy dataset with **custom loss function** (see David's talk Thursday)

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







The HEP.TrkX Project: deep neural networks for HL-LHC online and offline tracking, Steven Farrell 2017

JÜLICH Forschungszentrum

Hands-on Tutorials:

- ➢ https://github.com/wesmail/ML-GlueX-EIC-PANDA

```
git clone https://github.com/wesmail/ML-GlueX-EIC-PANDA.git
```

- ➢ Tutorials in perfect order

1. `data_exploration.ipynb`
2. `DBSCAN.ipynb`
3. `hit_pairs.ipynb`
4. `RNN.ipynb`
5. `GNN.ipynb`

Thank You