Updates on Deep Learning for Tracking with STT

Adeel Akram

Uppsala University
adeel.akram@physics.uu.se

$\overline{\mathrm{P}}$ANDA Collaboration Meeting
(GSI Darmstadt, Germany)

June 25, 2019

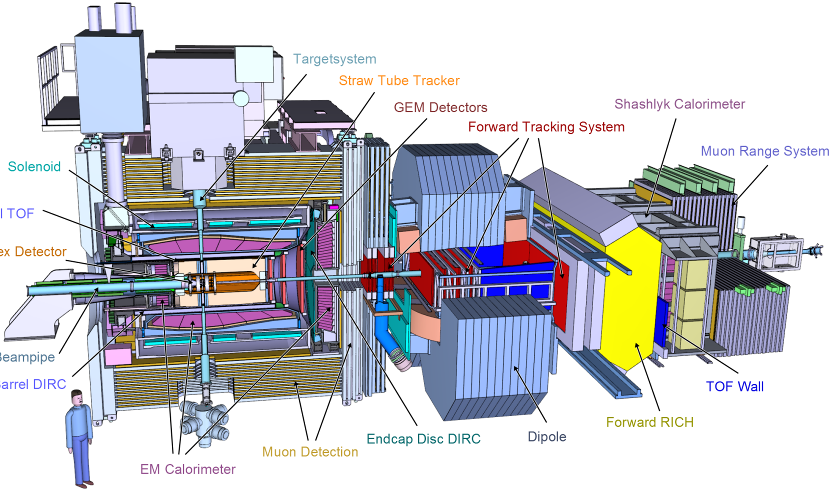
Outline

- PANDA Detector
- Tracking Reconstruction
- Machine/Deep Learning
- Current Updates
- Summary
- Future Plans

Motivation

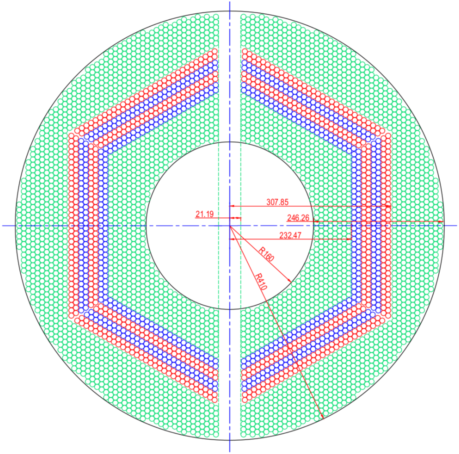
- Testing the method from FTS to STT
- Indentify possible issues
- Dense Networks as Seeder
- Seeder to a track builder algorithm (LSTM)

PANDA Experiment



Straw Tube Tracker (STT)

- $\sim$ 4,224 straws planned
- 15 - 19 axial layers (green)
- 8 stereo layers ($\pm 2.9°$) (red and blue)
- internal radius: 15 cm
- external radius: 42 cm
- length of tubes: 150 cm



Track Reconstruction

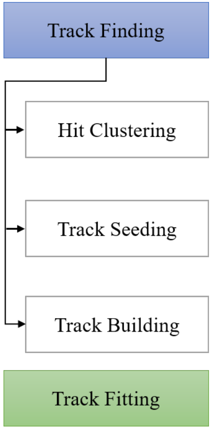
Common approach in High Energy Physics:

- **Track finding**
  - Input: Particle Hits
  - Algorithm: e.g. Cellular Automaton, Hough Transform etc.
  - Output: Track Candidates
- **Track fitting**
  - Input: Track Candidates
  - Algorithm: e.g. Kalman Filter, Helix Fit etc.
  - Output: Track Kinematics

Track Reconstruction (ML)

There are many open questions that how ML should be applied in tracking process (Fig.).

- One way is to apply ML in stages e.g. clustering, seeding and track building.
- The other way is to use ML in end-to-end manner e.g. track finding as a whole.



Learning Types (ML)

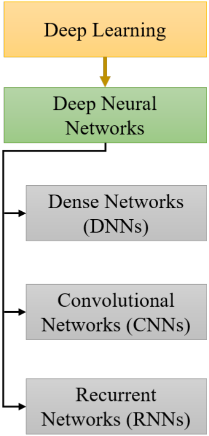
It is the ability of machines to learn complex representations of data through learning. Learning approaches are;

- **Supervised Learning**
  - Classification, Regression
- Unsupervised Learning
  - Clustering, Density Measurements
- Reinforcement Learning
  - Robotics etc

Deep Learning (ML)

It's an approach to introduce hidden layers in a model more generally deep neural networks:

- Dense Netowks
- Recurrent Networks
- Convolutional Networks

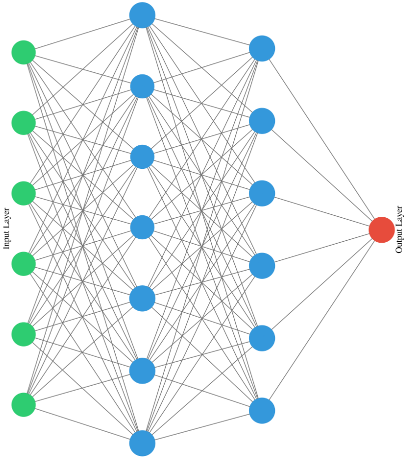


⇒Supervised Deep Learning
⇒Dense Networks

Dense Neural Networks

$$H^{[l]} = a^{[l]}(W^{[l]}.H^{l-1} + b^{[l]})$$
$$\hat{y} = H^{[l=L]}$$

$$l = 1, 2, ..., L$$
$$H = \text{output vector of layer } l$$
$$W = \text{weight matrix}$$
$$b = \text{bias vector}$$
$$a = \text{activation function}$$
$$\hat{y} = \text{final prediction}$$



Erorr Backpropagation

Cross-entropy cost function (J):

$$J(\hat{y}, y) = -1/m \sum_{i=1}^{m} [\hat{y}^{(i)} \log(y^{(i)}) + (1 - \hat{y}^{(i)}) \log(1 - y^{(i)})]$$

Gradient descent (epoch $\equiv$ 1 execution cycle):

$$dW^{[l]} = \partial J / \partial W^{[l]}$$
$$db^{[l]} = \partial J / \partial b^{[l]}$$

Parameter update with learning rate (α):

$$W^{[l]} := W^{[l]} - \alpha .\, dW^{[l]}$$
$$b^{[l]} := b^{[l]} - \alpha .\, db^{[l]}$$

Data Generation

The data is generated with PandaRoot using the Box Generator.

- 1000 events
- Momentum range $3 - 7$ GeV/c
- Polar angle 5^0 - 90^0
- Five muons per event

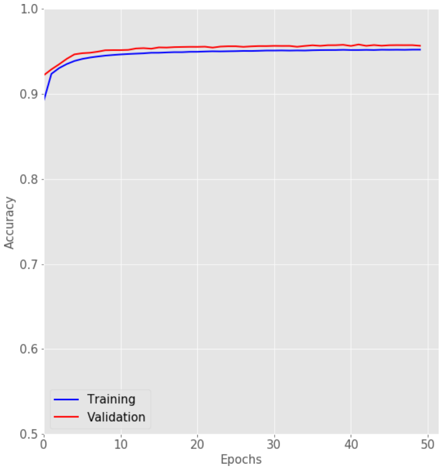
Training the Neural Network

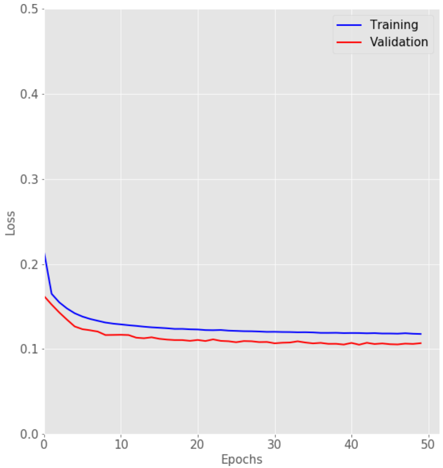
Hit-pairs (hit position) as input features and labels (0 or 1) as an output.

Features (X)		Labels (y)
(x_i, y_i)	(x_j, y_j)	0/1

Roughly, 11 million hit-pairs are trained on Neural Network.

Neural Network Performance



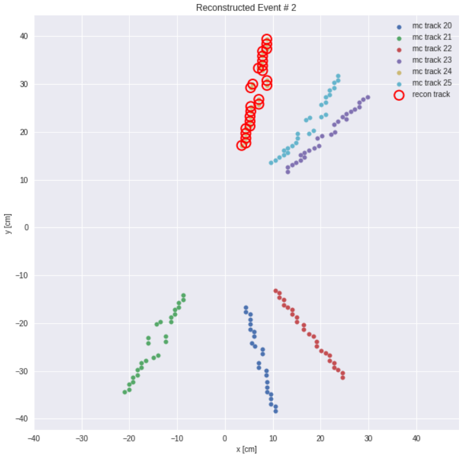


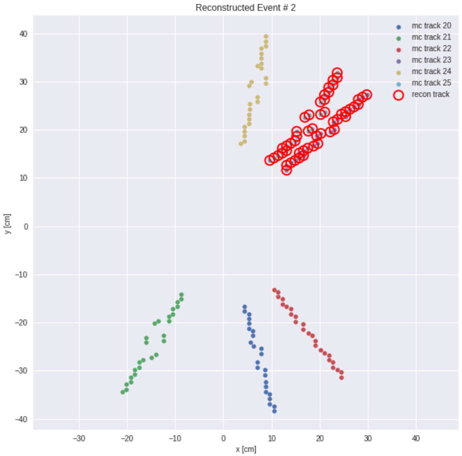
Track Building (Algorithm)

Algorithm for building the track from this approach is as follows:

- Get probability of hit pairs from Neural Network
- If prob $(h_i, h_j) > 0.5$ and prob $(h_j, h_k) > 0.5$ then h_i, h_j, h_k are on the same track

Reconstructed Event (Full)





Issues/Solutions

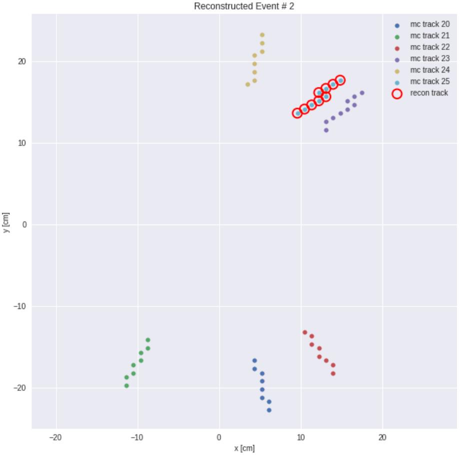
This method has some issues;

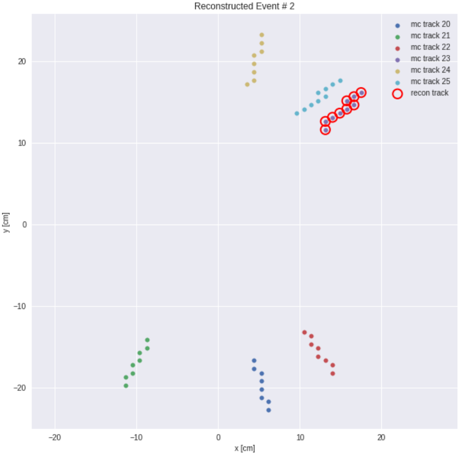
- Difficulty separating close tracks
- Even more difficult if two track cross each other

Possible solutions can be;

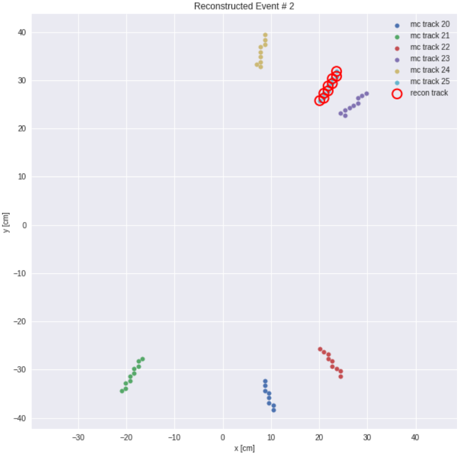
- Azimuthal angle between track segments
- Train on hit triplets rather than pairs
- Hit inclusion from the adjacent layers only

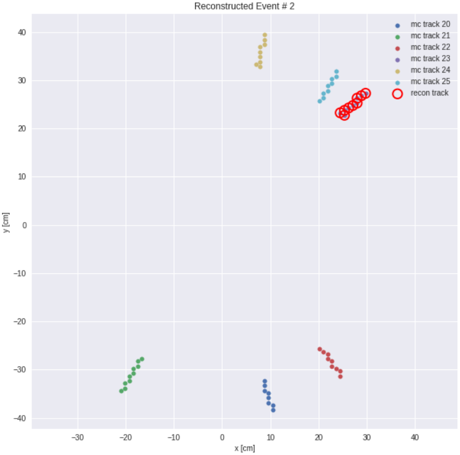
Reconstructed Event (Before Skewed)





Reconstructed Event (After Skewed)





Track Segment Joining

Two track segements can be joined together into a single track using a recurrent neural network (not done yet)

But, same algorithm can be used for different tracking task.

Track Extrapolation (Main)

This algorithm can used as stand-alone algorithm to build track segments or as track seeding.

Track Seeding:

- Build seed using hit pairs/triplets
- Dense Network

Track Building:

- Extrapolate SEEDS to form tracks (similar to Kalman Filter)
- LSTM (same layer prediction)/LSTM (next layer prediction)

Summary

- TrackML style approach
- Algorithm is adapted from FTS (Waleed Esmail) to STT
- Track segments in different parts of detector can be joined together

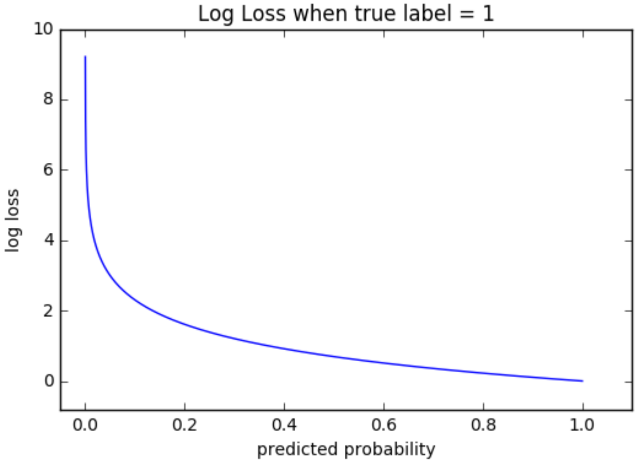
Future Plans

- Introducing azimuthal angle between track segments
- OR rather than hit-pairs, train on hit-triplets
- A Minimalist version to build track SEEDS
- SEEDER (Dense) to Track Builder (LSTM) algorithm

Questions?

Backup Slides

Cross-entropy Loss Function



Track Extrapolation

Recurrent networks are used to track extrapolation from track seeds:

Track Seeding:

- Build seed using hit pairs/triplets
- Dense Networks

Track Building:

- Extrapolate seeds to form tracks (similar to CKF)
- LSTM + FC (Softmax)

Hit Predictor Model

Recurrent networks are used as forward hit predictor model (sequence predition)

- hit prediction (tracks as sequence of hits), temporal component
- Next layer (forward) hit prediction
- LSTM + FC (Sigmoid)

Prediction

Whole execution of a Neural Network is reduced to minimization of **Cost Function (J)**.

$$J(\hat{y}, y) \rightarrow 0 \quad \text{given} \quad \alpha, W, b$$

After all epochs, our model is ready to predict using a threshold (τ).

$$y^{(i)}_{prediction} = \begin{cases} 1, & \hat{y}^{(i)} > \tau \\ 0, & \text{otherwise} \end{cases}$$

RNNs (Application)

RNNs are sequential models to handle data with a temporal component. For example, audio data, DNA sequencing, speech recognition, machine translation etc. Particle tracks can be seen as

Sequence data, a RNN (LSTM) can be used just like Combinatorial Kalman Filter (CKF). However, we need **track seeds** to **build tracks**.

RNNs (Issues)

Vanishing gradient due to backpropagation and in gradient-based learning models.

- More layers, gradient of loss function approaches zero
- Hinder the learning ability of network
- Earlier layers learn less than the later ones

RNNs is sequential, so prediction have higher contribution from the last layer. We can think of current state of node as a memory unit.