CLAS12 DC Tracking with Machine Learning

G. Gavalian & V. Ziegler

On Behalf of the CLAS12 AI Project Group

AI Workshop – Jefferson Lab, 03/04/2020

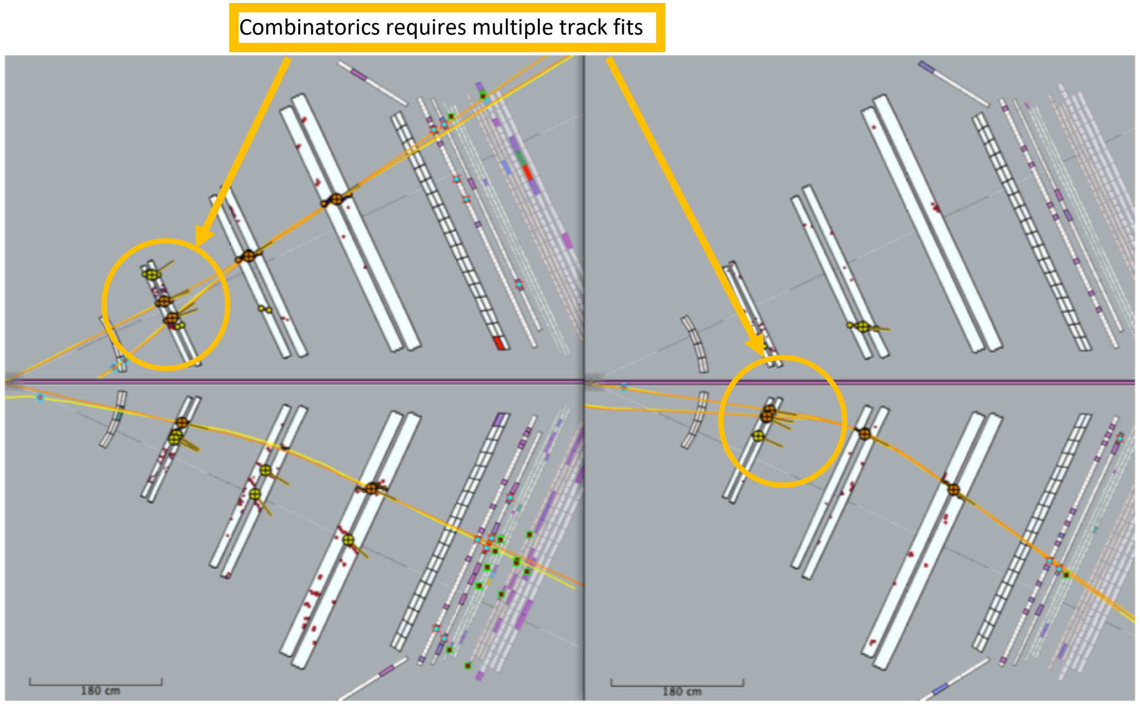
Machine Learning for CLAS12: Motivation

Largest CPU resource driver for event reconstruction is charged particle tracking

- DC Pattern recognition ~ 2 % CPU usage
- DC hit-based tracking ~ 58% CPU usage
- DC time-based tracking ~ 37% CPU usage

Targeted Areas of Improvement for CLAS12 DC Tracking:

- Processing speed
- More efficient noise rejection
- Combinatorics (ghost tracks)



AI Project Team

G. Gavalian (lead) – Neural Network evaluation software integration into CLAS12 software infrastructure.

V. Ziegler – AI-Assisted tracking code implementation. Benchmarking, testing and debugging of tracking code.

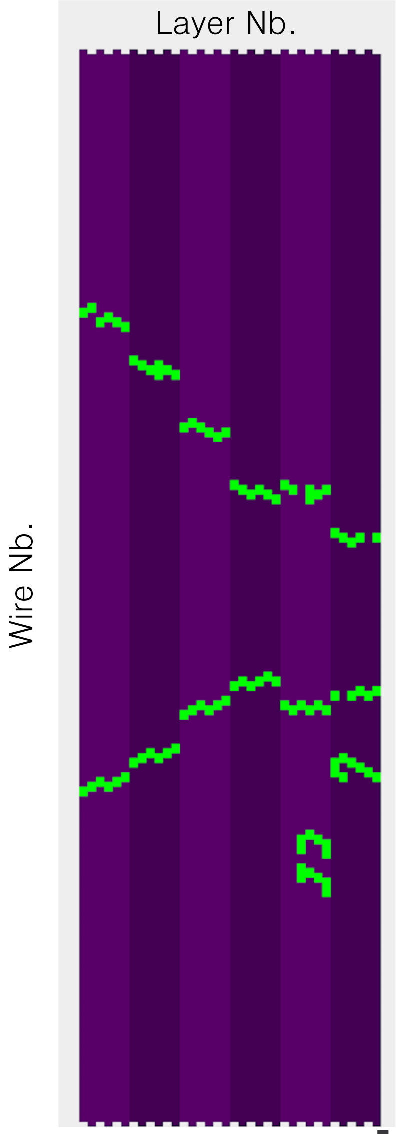
Center For Real-Time Computing (CRTC): N. Chrisochoides, P. Thomadakis, A. Angelopoulos – NN training and interface to CLAS12 framework; computing resources for the project (GPU farm equipped with high-end NVIDIA V100 GPUs).

- Testing different NN to determine which one is most suitable for CLAS12
- Writing of software package using Tensorflow/Keras/SciLearn to train drift chamber data, and run inference.
- Development of Python interface for reading HIPO data and writing inference results into Output.

Aims and Approach

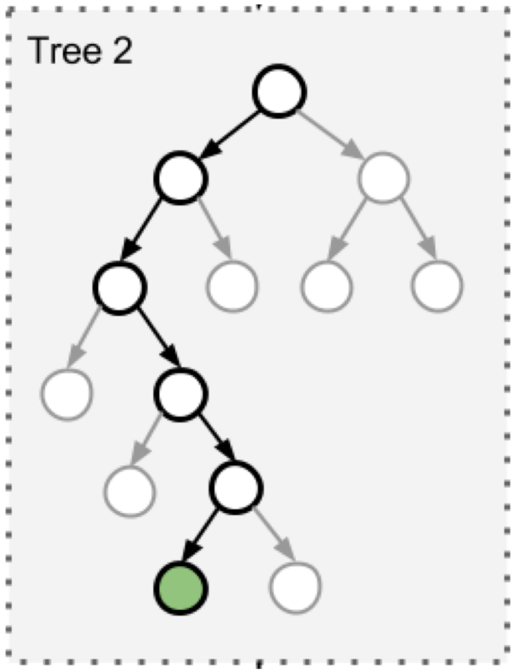
- **AI project phase 1 goals:**
  - Use AI to identify which DC track segments are consistent with being *on-track*.
  - Save these data in a dedicated data structure that is used by the tracking code as input, by-passing traditional pattern recognition phase, and hit-based tracking combinatorial selection algorithms.
  - Input: DC hits corresponding to wires (ordered by sector, layer, component number). Hits are stored in row-wise data structured called `banks'.

- **Approach:**
  - Provide samples consisting of hits belonging to track segments to the neural network: training samples and testing samples.

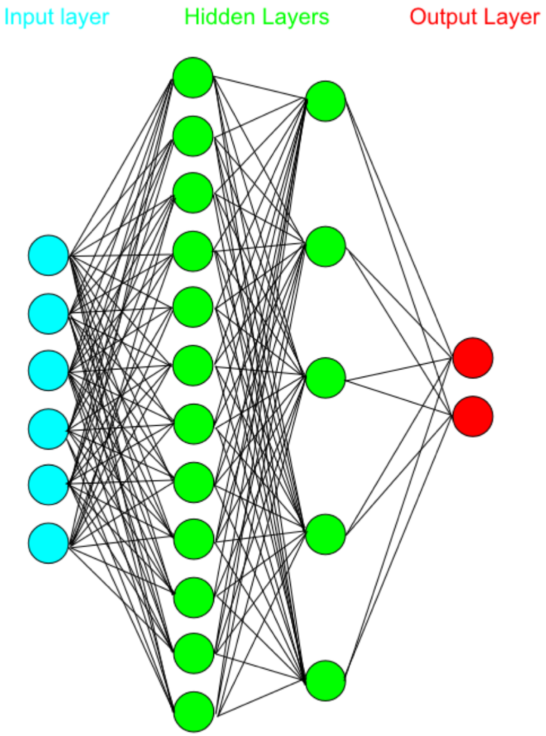


Choosing an AI Network

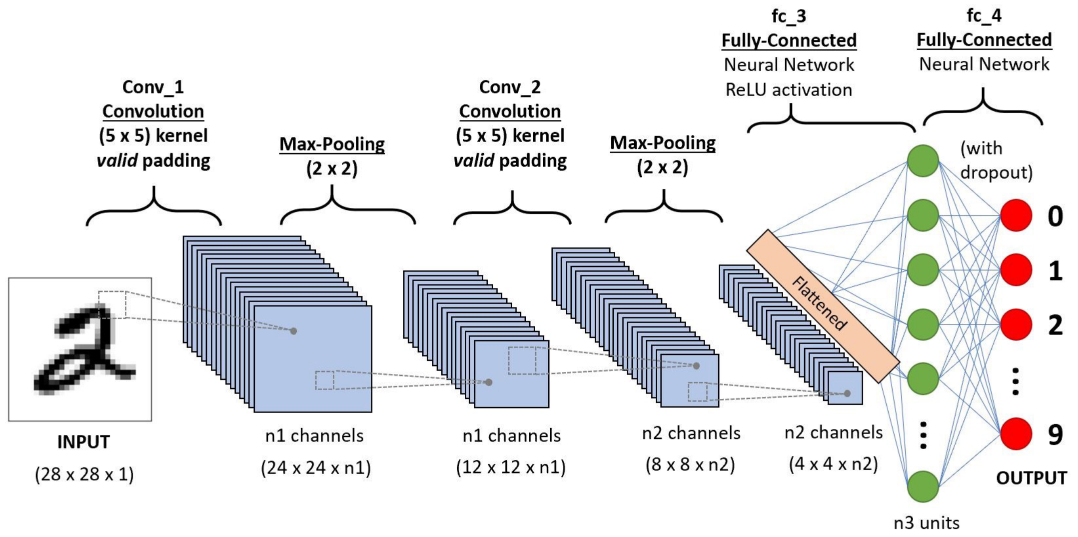
Boosted Decision Trees



Multilayer Perceptron

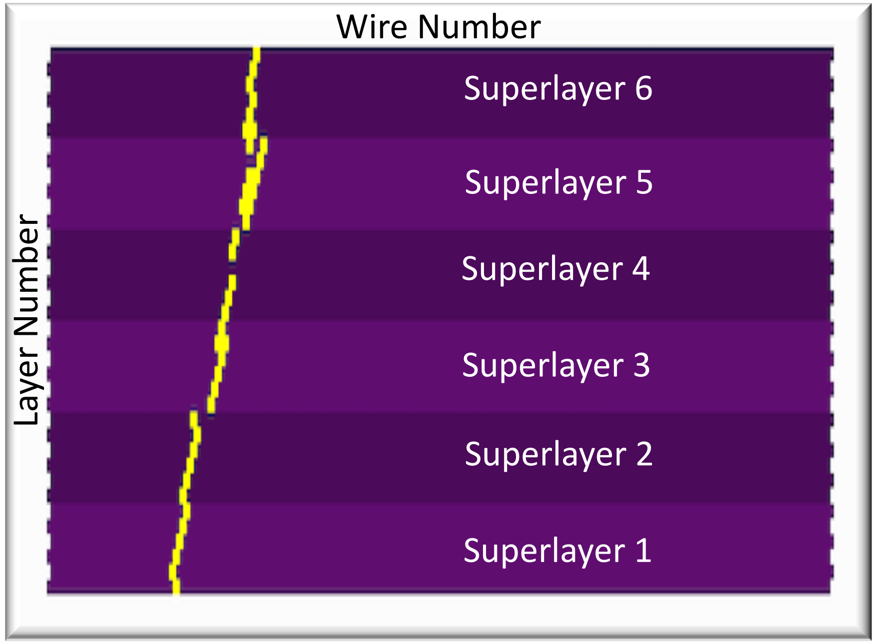


Convolutional Neural Network



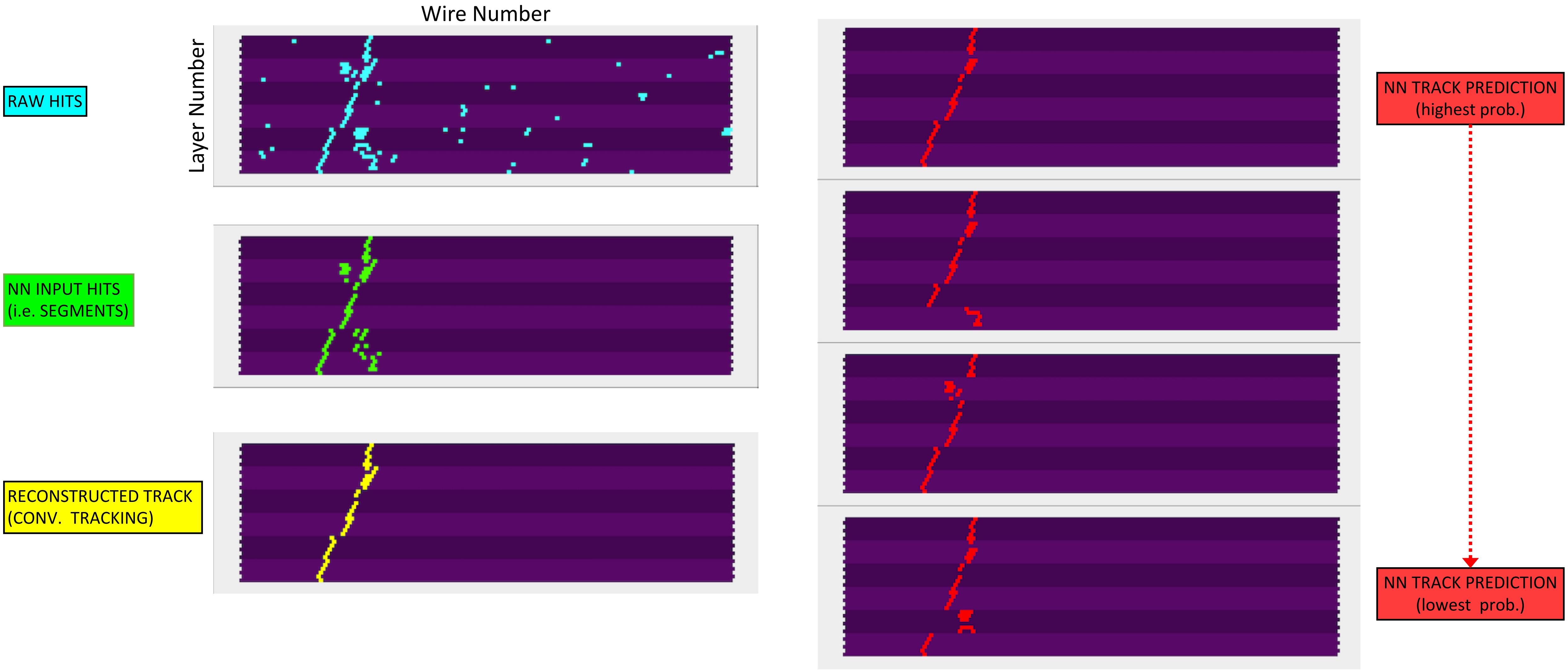
Neural Network Samples Used for Training

- 3 samples with different input parameters tested for training
  - BDT & MLP inputs *[sample 6]*
    - Average wire number of superlayer 1 segment → mapped to a local point
    - Angle between segment local points in superlayer 1 & 2
    - Average wire number of superlayer 3 segment → mapped to a local point
    - Angle between segment local points in superlayer 3 & 4
    - Average wire number of superlayer 5 segment → mapped to a local point
    - Angle between segment local points in superlayer 5 & 6



  - MLP inputs *[sample 36]*
    - Array of 36 numbers with wire (H.O.T.) number (or average wire number for double hit) for each of the 36 DC layers.
  - CNN inputs *[sample 4032]*
    - Picture with 36 x 112 dimensions passed to the network: if wire (H.O.T.) active → white pixel, else, black pixel.

Illustration of Selected Segments from the MLP after Training



AI Performance Accuracy Categorization

- NN returns a probability (*softmax fcn in last layer of classifier*) for track candidates.

- Based on this probability a label is created (1: true, 0: false) to flag candidates.

- Categorization of NN outcome based on correct estimation of the track candidate:
  - A1: # samples with correctly identified tracks / # input samples; no mis-identified tracks.
    - Only one track identified in given *group* of hits. This track candidate is the correct one.
  - Ac: # samples with correctly identified track + mis-id^ed candidates/ # input samples; i.e. contains *False Positives*.
    - Multiple candidates identified. Contains candidates with highest probability that do not correspond to correct tracks (*False Positives*).
  - Ah: # samples with correctly identified tracks / # input samples; with the valid track assigned the highest probability.
    - Multiple candidates identified. Candidates with highest probability that are correctly identified.
  - Af: # samples with correct track not-identified / # input samples; i.e. *False Negatives*

Accuracy Scores for AI Networks Tested

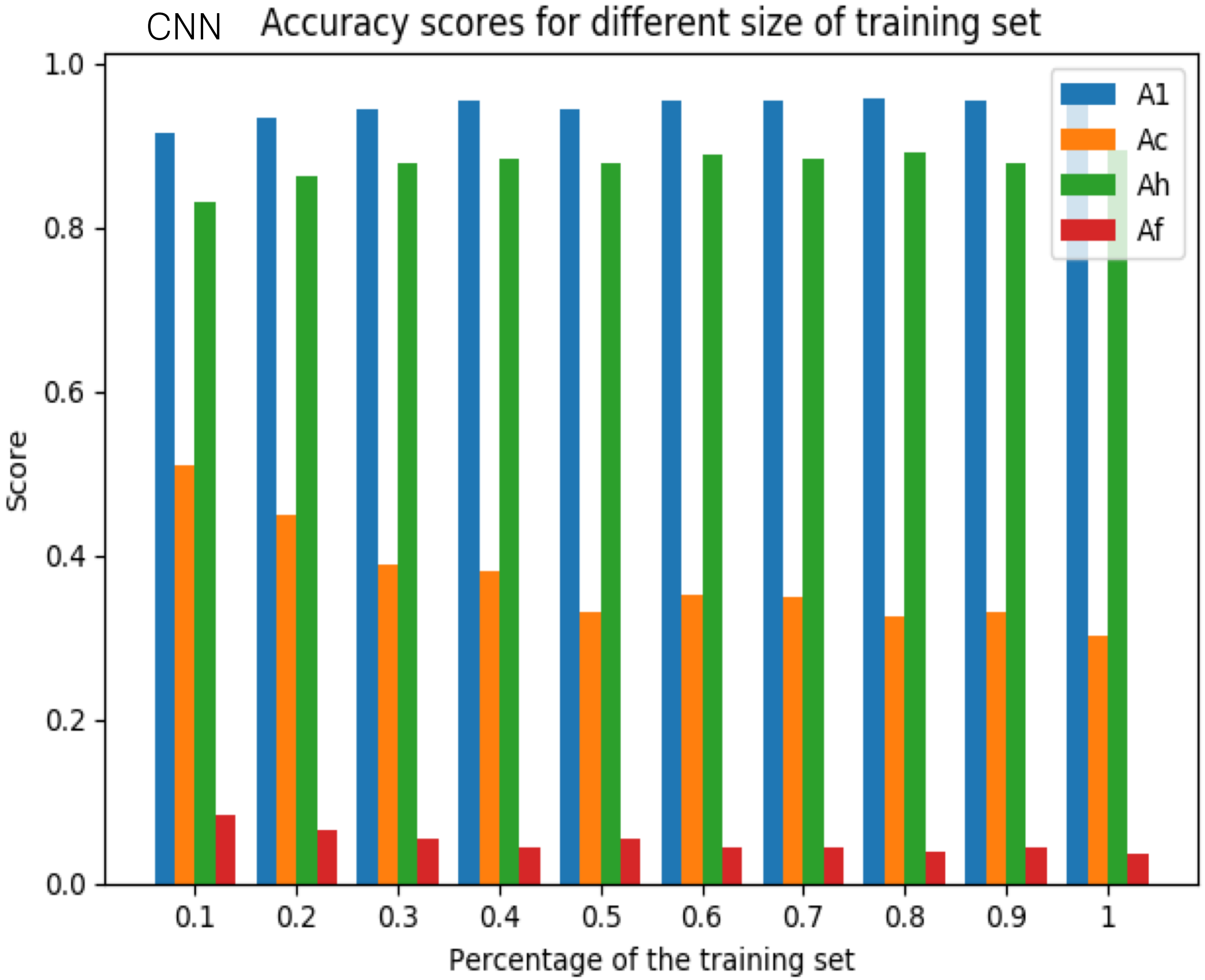
- Preliminary results obtained with training samples split into multiple sets
  - Split data using DC data corresponding to a 50 nA sample (Run 5038)
    - Using sectors, 1, 3, 4, 5, 6 for training;
    - Using sector 2 for testing.
- Best track finding accuracy with CNN and MLP.
- More tests being done.

Method/Train set	Split	Test Set/Format	A1	Ac	Ah	Af	Training Acc	Validation Acc	Time to train (sec)	Time to predict (sec)
CNN/10-7-19	10/90	10-7-19/4032	0.915	0.511	0.832	0.084	0.9389	0.900	199	0.0012
	20/80	10-7-19/4032	0.935	0.450	0.863	0.064	0.9310	0.9136	226.1	0.0012
	30/70	10-7-19/4032	0.945	0.389	0.878	0.054	0.934	0.924	257.1	0.0012
	40/60	10-7-19/4032	0.955	0.382	0.885	0.044	0.934	0.926	288.1	0.001
	50/50	10-7-19/4032	0.944	0.331	0.880	0.055	0.934	0.931	319	0.001
	60/40	10-7-19/4032	0.954	0.353	0.888	0.045	0.934	0.93	347.4	0.0013
	70/30	10-7-19/4032	0.954	0.350	0.885	0.045	0.9343	0.9325	380.8	0.0014
	80/20	10-7-19/4032	0.959	0.327	0.892	0.040	0.9344	0.9315	412.4	0.0014
	90/10	10-7-19/4032	0.956	0.332	0.880	0.043	0.9347	0.9332	444.3	0.0014
	100/0	**10-7-19/4032**	**0.964**	**0.301**	**0.894**	**0.035**	**0.934**	**N/A**	**457**	**0.0012**
ExtraTrees/10-7-19	10/90	10-7-19/6	0.923	0.241	0.914	0.077	1.0	0.92	0.2	0.000005
	20/80	10-7-19/6	0.921	0.217	0.914	0.078	1.0	0.924	0.3	0.000005
	30/70	10-7-19/6	0.923	0.217	0.912	0.076	1.0	0.925	0.4	0.000005
	40/60	10-7-19/6	0.927	0.205	0.914	0.072	1.0	0.927	0.6	0.000005
	50/50	10-7-19/6	0.930	0.204	0.916	0.069	1.0	0.928	0.7	0.000005
	60/40	10-7-19/6	0.929	0.206	0.918	0.070	1.0	0.915	0.8	0.000005
	70/30	10-7-19/6	0.932	0.203	0.92	0.067	1.0	0.93	1.0	0.000005
	80/20	10-7-19/6	0.933	0.199	0.918	0.066	1.0	0.917	1.2	0.000005
	90/10	10-7-19/6	0.933	0.197	0.919	0.066	0.9999	0.931	1.3	0.000005
	100/0	**10-7-19/6**	**0.933**	**0.199**	**0.919**	**0.066**	**0.9999**	**N/A**	**1.7**	**0.000005**
	100/0	10-7-19/36	0.910	0.339	0.883	0.089	0.9999	N/A	8.4	0.000005
MLP/10-7-19	**100/0**	**10-7-19/6**	**0.965**	**0.202**	**0.921**	**0.034**	**0.947**	**N/A**	**252 (CPU)**	**0.000004**
	100/0	10-7-19/36	0.920	0.312	0.849	0.079	0.921	N/A	546 (CPU)	0.00001

Accuracy Scores for AI Networks Tested

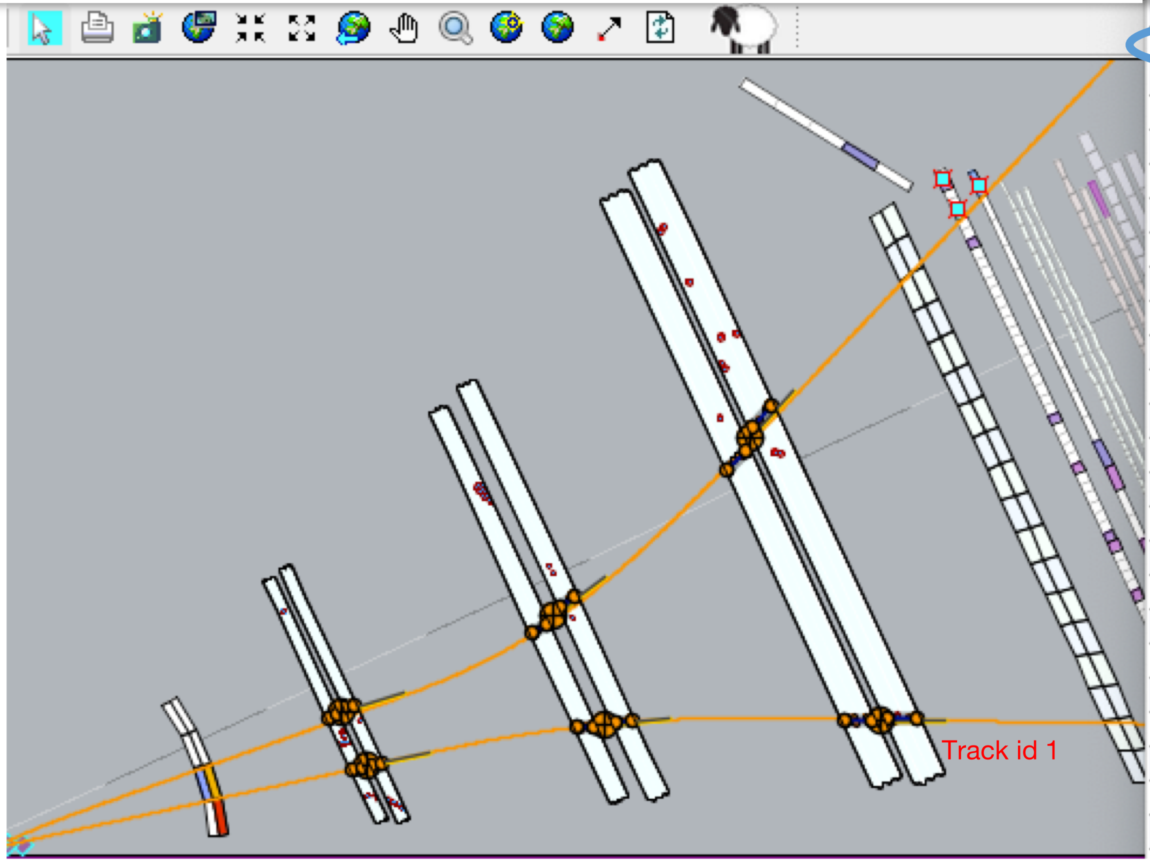
NN	A1	Ac	Ah	Af	Training Accuracy	Training Time	Prediction Time
CNN	0.964	0.301	0.894	0.035	93.4%	457 sec	0.0012 sec
BDT	0.933	0.199	0.919	0.066	99.9%	1.7 s	0.000005 sec
MLP	0.965	0.202	0.921	0.034	94.7%	252 (CPU)	0.000004 sec

A1: # samples with correctly identified tracks / # input samples; no mis-identified tracks.
Ac: # samples with correctly identified track + mis-id[ed] candidates/ # input samples; i.e. contains *False Positives*.
Ah: # samples with correctly identified tracks / # input samples; with the valid track assigned the highest probability.
Af: # samples with correct track not-identified / # input samples; i.e. *False Negatives*



Implementation in Current Tracking

- Creation of new bank read to get track seeds at hit-based level
- Dedicated DC service to use this bank to reconstruct track seeds → passed to hit-based fitting.



TDC bank

DC::tdc

	sector	layer	component	order	TDC
78	1	5	24	0	213
79	1	2	25	0	243
80	1	6	25	0	186
81	1	1	25	0	153
82	1	4	37	0	74
83	1	5	36	0	40
84	1	3	36	0	34
85	1	2	37	0	33
86	1	2	38	0	12
87	1	3	37	0	97
88	1	4	40	0	45
89	1	4	39	0	24
90	1	5	42	0	73
91	1	5	41	0	60
92	1	4	42	0	52
93	1	6	42	0	22
94	1	5	48	0	223
95	1	3	49	0	242
96	1	4	49	0	209
97	1	6	49	0	189
98	1	1	49	0	175
99	1	2	49	0	155
10	1	3	98	0	417
10	1	3	97	0	413
10	1	4	97	0	398
10	1	4	98	0	380
10	1	12	5	0	485

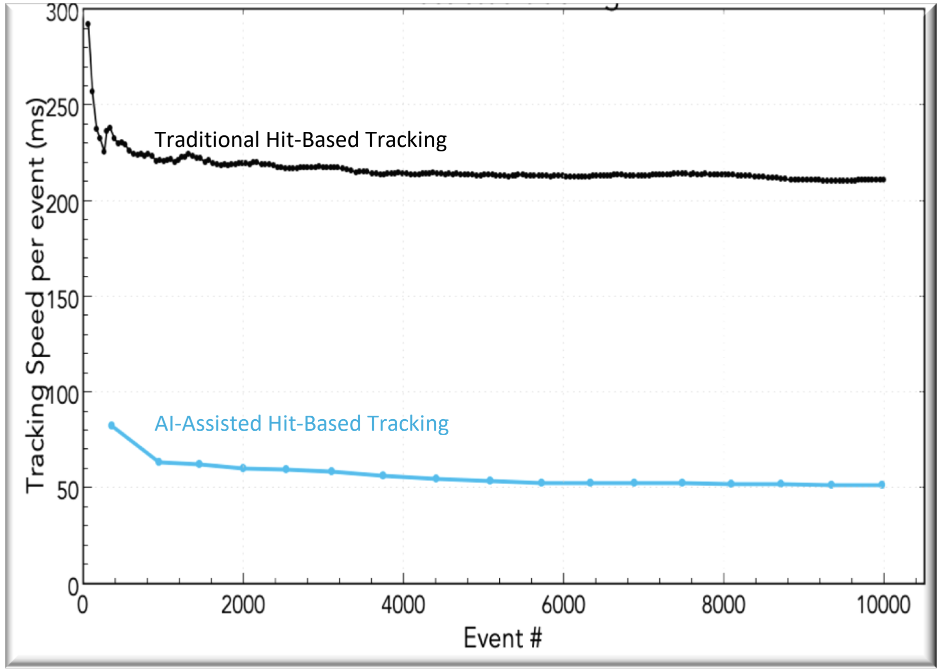
NN bank

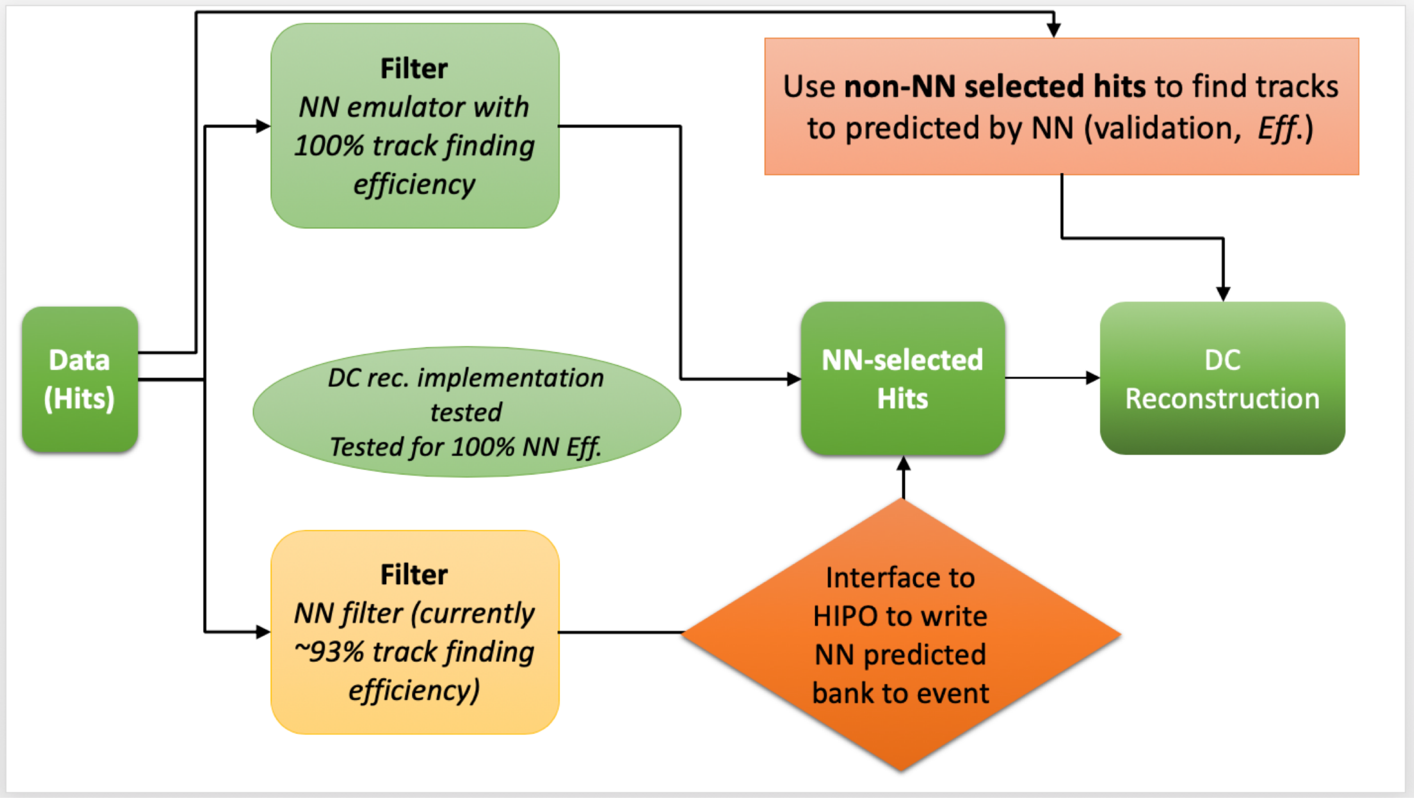
Track id | Index in TDC bank

	id	index
1	1	81
2	1	79
3	1	78
4	1	80
5	1	116
6	1	115
7	1	119
8	1	118
9	1	114
10	1	117
11	1	132
12	1	127
13	1	131
14	1	130
15	1	128
16	1	129
17	1	155
18	1	156
19	1	152
20	1	154
21	1	153
22	1	157
23	1	168
24	1	171
25	1	170
26	1	169
27	1	167

Expected Performance Improvement

- Hits-On-Track saved in NN Bank.
- Developed the API to use new bank for seeding in DC package.
- Dedicated service to run AI reconstruction if the NN hits bank exists in the HIPO file.
- Python interface to HIPO being developed to put the results of the NN into a HIPO file.





- Even with unoptimized NN efficiency, the gain in reconstruction speed will lead substantial time gains in (re-) calibration of data.

Current Status & Summary

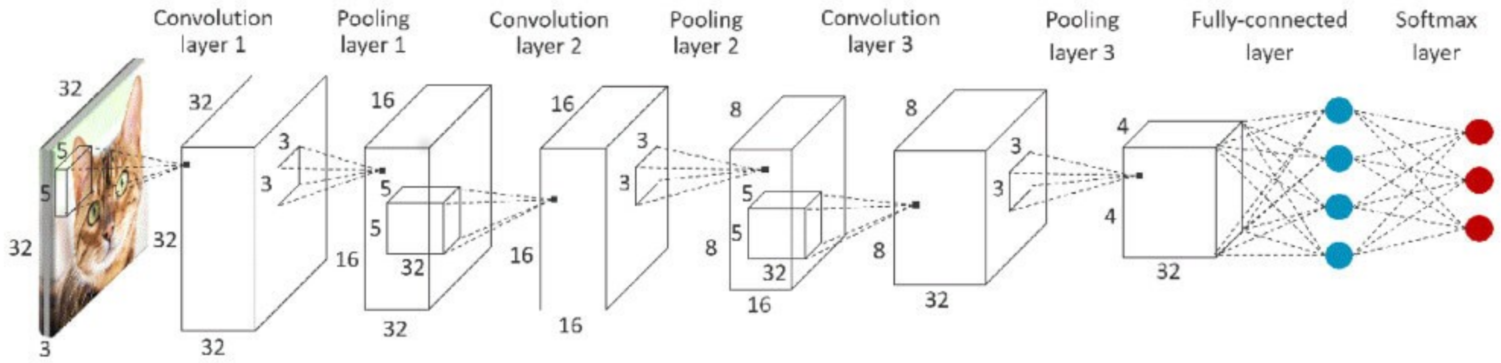
- DC Tracking modified to work with Neural-Network-predicted Hits-On-Track. (Done)
- Framework in python to train and test track candidates (done)
- Validation of network performance with MC (in progress)
- Implementation of Neural Network software into workflow (in progress):
  - Interface to HIPO with python to read track candidates.
  - Interface with TensorFlow to get track candidate predictions and write them into the HIPO file.

- TO-DO
  - Training on HIPO 4 data (varying conditions).
  - Validation of accuracy using data and MC samples (i.e. sample with background-merging).
  - Possibly include the prediction algorithm into decoding.
  - Dedicated clustering service (ongoing).

BACK-UP SLIDES

Network Configuration

- Basic VGG16 model was used to train on track reconstruction images.
- Initial sample of event 20K for positive and negative samples.
- Inference time ~3ms (GPU NVIDIA Tesla K40m)
- Reducing network size will reduce inference time.
- For comparison decoding time per event is ~15ms.



Neural Network model definition