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

RAW HITS

Wire Number

Layer Number

NN INPUT HITS (i.e. SEGMENTS)

RECONSTRUCTED TRACK (CONV. TRACKING)

NN TRACK PREDICTION (highest prob.)

NN TRACK PREDICTION (lowest prob.)
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-identified 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.

<table>
<thead>
<tr>
<th>Method/Train set</th>
<th>Split Set/Format</th>
<th>Method/Train set</th>
<th>Split Set/Format</th>
<th>Method/Train set</th>
<th>Split Set/Format</th>
<th>Method/Train set</th>
<th>Split Set/Format</th>
<th>Method/Train set</th>
<th>Split Set/Format</th>
<th>Best Acc</th>
<th>Worst Acc</th>
<th>Time to train (sec)</th>
<th>Time to predict (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN/10-7-19</td>
<td>10/90 10-7-19/4032 0.915 0.511 0.832 0.084 0.9389 0.900 199 0.0012</td>
<td>ExtraTrees/10-7-19</td>
<td>10/90 10-7-19/6 0.923 0.241 0.914 0.077 1.0 0.92 0.2 0.000005</td>
<td>MLP/10-7-19</td>
<td>10/90 10-7-19/6 0.965 0.202 0.921 0.034 0.947 N/A 252 (CPU) 0.000004</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Accuracy Scores for AI Networks Tested

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

**A1**: # samples with correctly identified tracks / # input samples; no mis-identified tracks.

**Ac**: # samples with correctly identified track + mis-identified 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.
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).
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.