CLAS12 Track Reconstruction with Artificial Intelligence

Gagik Gavalian

Jefferson Lab, Newport News, VA, USA

Polykarpos Thomadakis, Angelos Angelopoulos, and Nikos Chrisochoides

CRTC, Department of Computer Science,

Old Dominion University, Norfolk, VA, USA

Raffaella De Vita and Veronique Ziegler

Jefferson Lab, Newport News, VA 23606, USA

Abstract

In this article we describe the implementation of Artificial Intelligence models in track reconstruction software for the CLAS12 detector at Jefferson Lab. The Artificial Intelligence based approach resulted in improved track reconstruction efficiency in high luminosity experimental conditions. The track reconstruction efficiency increased by $10 - 12\%$ for single particle, and statistics in multi-particle physics reactions increased by $15\% - 35\%$ depending on the number of particles in the reaction. The implementation of artificial intelligence in the workflow also resulted in a speedup of the tracking by 35%.

INTRODUCTION

Nuclear Physics experiments have become increasingly complex over the past decades, with more complex detector systems and higher luminosities. In emerging experiments where data collection rates are higher, there is a need for new approaches for data processing that can improve data reconstruction accuracy and speed. New developments in the Artificial Intelligence (AI) field present promising alternatives to conventional algorithms for data processing. Machine Learning (ML) algorithms are being employed in various stages of experimental data processing, such as: the detector data reconstruction, particle identification, detector simulations and physics analysis.

In this paper we present the implementation of machine learning models into the CLAS12 charged-particle track reconstruction software. Detailed analysis of the reconstruction performance are presented, comparing track reconstruction efficiency and speed improvements to conventional algorithms.

CHARGED PARTICLE TRACKING

The CLAS12 [1] forward detector is built around a six-coil toroidal magnet which divides the active detection area into six azimuthal regions, called "sectors". Each sector is equipped with three regions of drift chambers [2] designed to detect charged particles produced by the interaction of an electron beam with a target. Each region consists of two chambers (called super-layers), each of them having 6 layers of wires. Each layer in a super-layer contains 112 signal wires, making a super-layer a 6x112 cell matrix. The schematic view of one region is shown on Figure 1 (right panel).

Particles that originate at the interaction vertex travel through the magnetic field and pass through all three regions of the drift chambers in a given sector are reconstructed by tracking algorithms. First, in each super-layer adjacent wires with a signal are grouped together into clusters (called segments), shown on Figure 1. The positions of these clusters (segments) in each super-layer are used to fit the track trajectory to derive initial parameters, such as momentum and direction. After the initial selection, good track candidates are passed through Kalman filter [3] to further refine measured parameters.
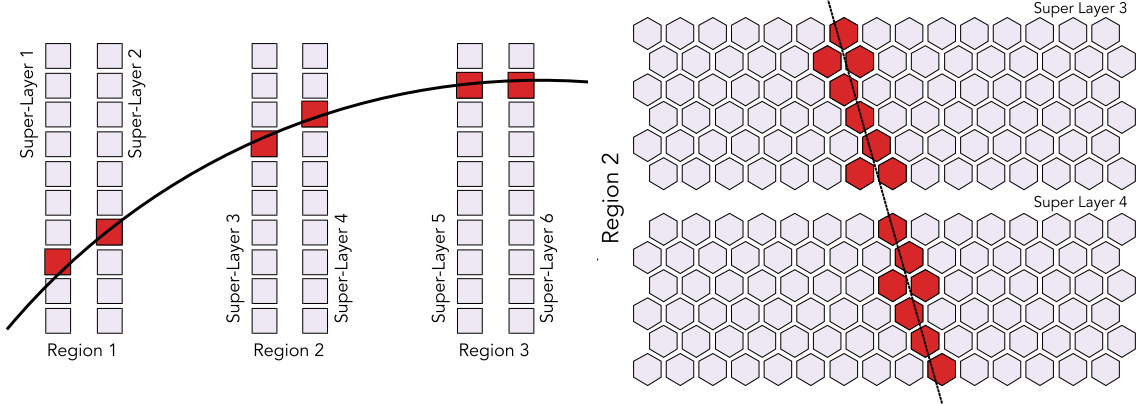
FIG. 1: Schematic view of signals generated in the drift chambers when a particle passes through. The segments in each super-layer are shown along the trajectory of the track (left panel), and view of the activated cells in the two super-layers of one region along the track trajectory (right panel).
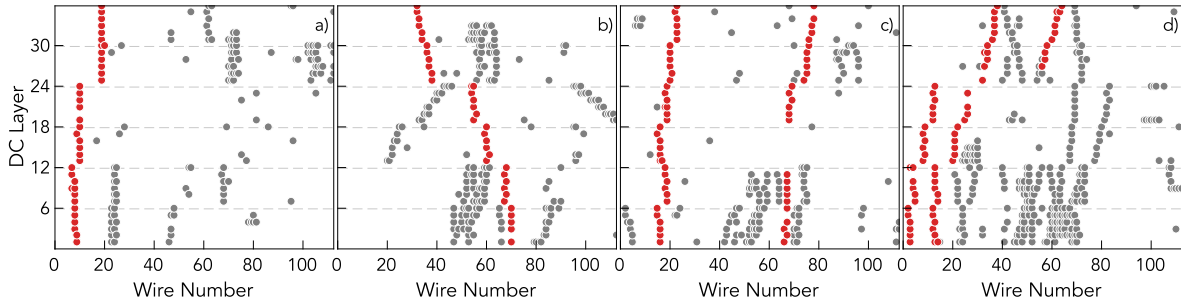

FIG. 2: Example of signals in drift chambers for four events. Each plot represents one sector with a 36x112 matrix of wires. Background hits (in gray) are shown along with the hits of reconstructed identified tracks (in red). Dashed lines represent boundaries between super-layers.

For each beam-target interaction or "event", drift chambers produce many segments, some belonging to a track and some are background, or partial trajectories of low momentum tracks. In Figure 2 drift chamber signals in one sector are shown for four different events, in each sector data are hits represented as a 36x112 matrix (36 layers and 112 wires per layer), showing all hits including those that were determined to be part of a track.

Due to inefficiencies in the drift chambers, it is possible to have one missing segment along the trajectory of the particle, and the track has to be reconstructed using only 5 segments. An example of a 5-segment track is shown on Figure 2 c), where super-layer 3 does not have

3

any segment detected. For these types of tracks, candidates have to be identified from a large number of combinatorics consisting of all combinations of clusters that form a 5-segment candidates.

Tracking is computationally intensive and makes up $80\% - 90\%$ of the total CLAS12 event processing time (depending on background conditions and track multiplicity). The procedure of finding tracks from a list of track candidates is where we found AI can provide real benefits. Such benefits include: improved accuracy in identifying good tracks, and improved data processing speed by significantly reducing the number of candidates that have to go through initial fitting and then through Kalman filter. AI can also help in identifying 5 super-layer track candidates.

MACHINE LEARNING

The Machine Learning is used to train a neural network to recognize good tracks from a large list of candidates, constructed using one cluster from each super-layer. Two neural networks were developed: a classifier that can identify a good track from 6-segment track candidates and an auto-encoder that can take a list of 5-segment tracks and turn them into 6-segment track candidates by adding a pseudo-segment. The composed 6 super-layer track candidates can be processed with the classifier network to identify and isolate "good" track candidates.

Track Classifier

To determine what type of machine learning model works best with the CLAS12 drift chamber data, we investigated different types of models [4], including Convolutional Neural Network (CNN) , Extremely Randomized Trees (ERT) [5], and Multi-Layer Perceptron (MLP) [6]. The study showed that Multi-Layer Perceptron was best suited for the CLAS12 reconstruction needs (based on inference speed and accuracy). The implemented architecture is shown in Figure 3, where an input layer with 6 nodes is used (each node representing the average wire position of the segment in super-layer) and 3 output nodes for the classes "positive track", "negative track" and "false track".

The network is trained using a sample of tracks reconstructed by the conventional algo-
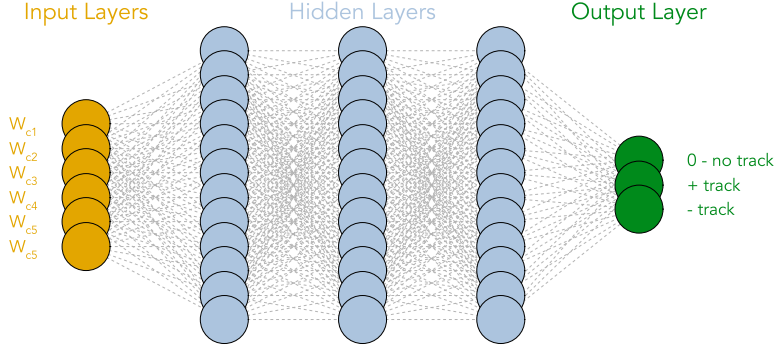
FIG. 3: Architecture of Multi-Layer Perceptron used for track classification. Network has 6 input nodes, corresponding to average wire positions of the segment in each super-layer, three hidden layers with 12 nodes in each and 3 output nodes.

rithm, selected with χ^2 cuts to retain the highest quality ones, which are fed to the network with their respective labels (i.e. positive or negative tracks). For false tracks, a combination of segments (6-segments forming a track candidate) that was not identified as a track by the conventional algorithm is chosen.

Corruption Auto-Encoder

A second neural network was developed to fix the corruption in possible track candidates due to inefficiencies of drift chambers. This network was used to identify track candidates which have one of the segments missing. We used an auto-encoder architecture to implement the corruption-recovery neural network [7]. The structure of the network can be seen in Figure 4, with 6 input nodes and 6 output nodes.

To train the corruption auto-encoder network the same sample used for the classifier training was used. The output for the network was set to the good track parameters (where all 6 segments have non-zero values) and the input was modified by setting one of the nodes (randomly) to zero. The network learns to fix the node containing zero, by assigning it a value based on the other 5 segment values.

The test results of the trained network are shown in Figure 5, where the difference between the true value of the segment position and the one reconstructed by the network is plotted,
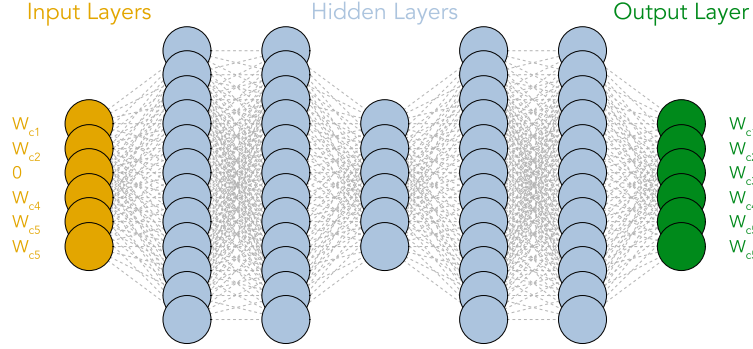
FIG. 4: Corruption-recovery auto-encoder architecture with 6 input nodes representing track segments mean wire values with one of the values is set to 0, and 6 output nodes with the correct value for the node that has 0 in the input.
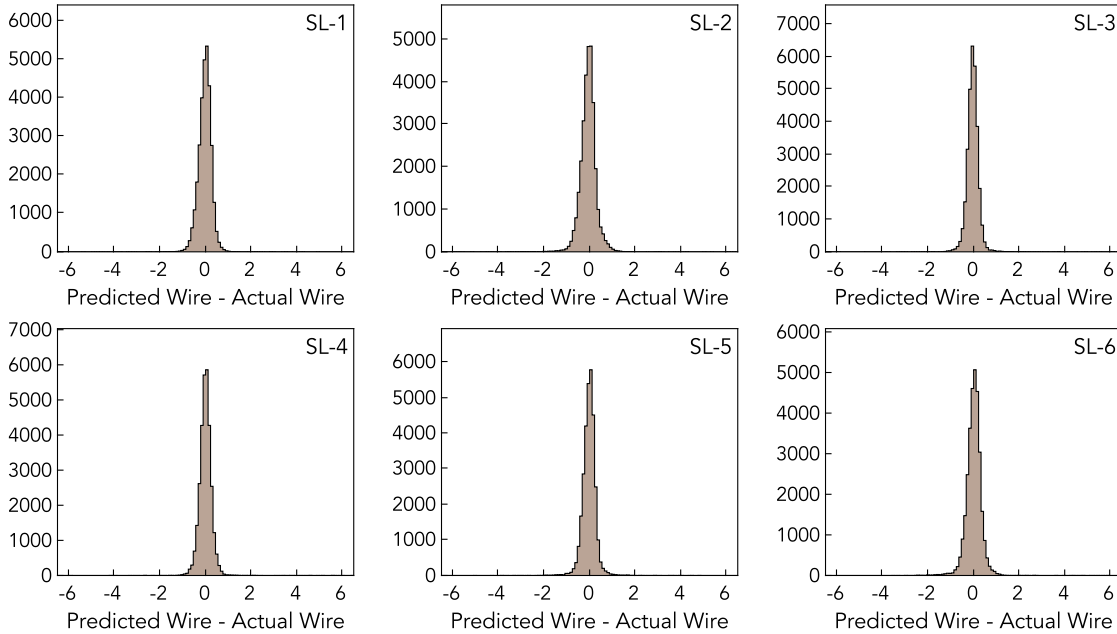

FIG. 5: Performance of corruption recovery auto-encoder foe each of six super-layers. The difference between predicted and actual wire position is shown for all super-layers. Average accuracy of wire position prediction is 0.36 wire.

showing a reconstruction accuracy of 0.36 wires. In Figure 4 this difference is shown for each super-layer which was corrupted in the input. As can be seen, the performance of the

6

network is uniform across all super-layers.

IMPLEMENTATION OF THE NEURAL NETWORK IN CLAS12 SOFTWARE

The models described in the previous sections were implemented in the CLAS12 tracking software.

Track Identification Workflow

Track identification consists of two phases, programmed to be done in two passes. In the first pass over the data, signals from each sector of drift chambers are analyzed to create a track candidate list, each consisting of 6 segments. The resulting track candidates are evaluated by the classifier neural network and are assigned a probability for being either positive or negative track. The list of track candidates is sorted by probability and passed to another algorithm that is responsible for removing tracks that have lower probability of being a "good" track and have clusters that are shared with a higher-probability candidate.

In this procedure the algorithm iterates over track candidates sorted according to the probability of being a good track. Iteration starts at position number 2 and runs to the end of the list. Candidates that share a cluster with candidate at position number 1 are removed from the list. Track candidate at position 1 is moved from the track candidate list into the identified track list. This procedure is repeated until there are no track candidates left in the candidate list.

The second stage of track identification starts by constructing a list of track candidates with combinations of 5 clusters out of 6 from all existing clusters (one per super-layer). The candidates that share a cluster with tracks identified at the first stage of classification are removed from the list. For each track candidate with a missing cluster in one of the super-layers, a pseudo-cluster is generated using the Corruption Auto-Encoder Network and the missing super-layer cluster is assigned the inferred value, hence turning all track candidates to 6 cluster track candidates. The cured (or fixed) track candidate list is finally passed to the track classifier module described above, which evaluates the list isolating candidates with the highest probability of being a good track.
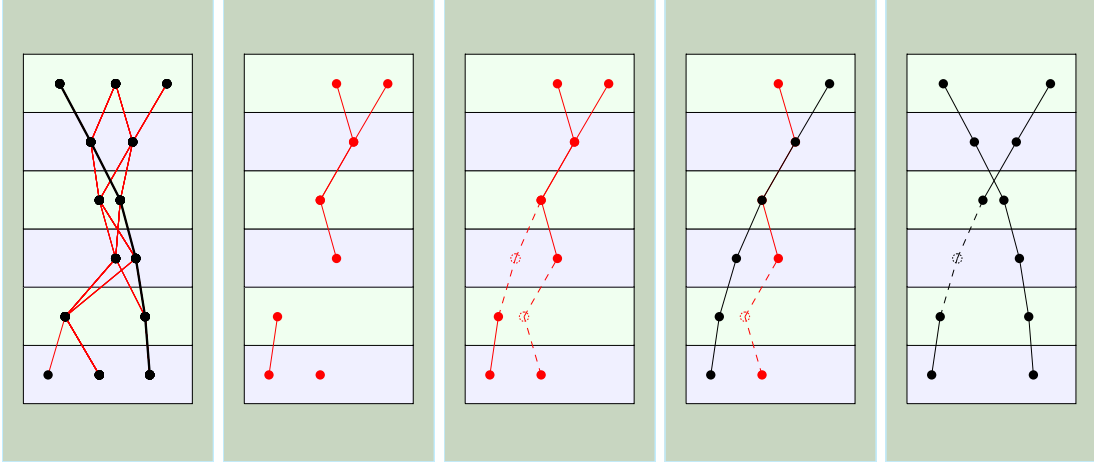
FIG. 6: Stages of Neural Network track identification procedure. 1) identifying 6 super-layer tracks. 2) removing all hits belonging to a identified track and constructing 5 super-layer track candidates. 3) generating pseudo-clusters for 5 super-layer track candidates using corruption fixing auto-encoder. 4) identify good track candidates from the list of 6 super-layer (one of the super-layers is pseudo-cluster) track candidates. 5) isolate both identifies (6 super-layer and 5 super-layer) tracks for further fitting with Kalman-Filter.

Implementation in reconstruction software

The CLAS12 reconstruction software framework is a Service Oriented Architecture platform implemented in Java (CLARA [8]). The reconstruction software consists of several microservices, each responsible for processing data from one detector [9]. The reconstruction procedure for some of the detector components can also be broken down into smaller logical microservices to add some flexibility in changing the implementation of the small parts, and provide alternative reconstruction procedures for some of the components. Reconstruction of tracks in drift chambers is a complex task and consists of several parts.

The first stage of the process, called clustering service, is to isolate clusters from the hits in drift chambers. Once the clusters are isolated, track candidates are formed from all combinations of 6 segments. Once 6 segment tracks have been identified, the remaining segments are then used to form 5 segments combinations. These two steps are known as track-finding or seeding. Both 6 and 5 segments candidates are fitted defining the hit positions from the wire coordinates (hit based tracking), resulting in a first list of reconstructed tracks.

In a second stage of the tracking code (time-based tracking), the tracks identified at the previous stage are refitted with a Kalman Filter algorithm, which uses drift time information to refine the hit positions. This produces the final track list.
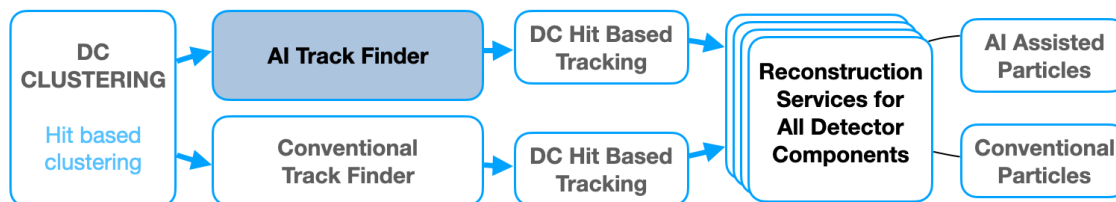

FIG. 7: Diagram of the tracking workflow with Artificial Intelligence included. The Workflow is split into two parallel branches, one where track-finding is done by the conventional algorithm, and one where only AI-isolated tracks are fed to the following stages.

In order to implement our neural network into the reconstruction workflow, we designed two parallel branches in the reconstruction code where we run two algorithms to identify good tracks from the candidate lists, one based on the conventional algorithm and one based on the neural network. The two algorithms store their track suggestions in separate data structures, and pass them to the next stage where track parameters are reconstructed by conventional tracking algorithm, first by hit-based fitting and then using the Kalman filter. This approach lets us have two parallel outputs from the tracking code that enables a detailed comparison of the performance of each method.

Software Packages

As mentioned above, CLAS12 reconstruction is implemented in Java. The final implementation of the the track classifier was therefore in Java for easy integration into the reconstruction workflow. The initial tests and prototyping were done using the Keras/TensorFlow [10] (python) package. The final implementation was based on the DeepNetts [11] community edition library (in native Java) used for both the track candidate classifier and corruption-recovery auto-encoder. DeepNetts is a light weight library with minimal dependencies, which makes it ideal for providing portable code that can be used on variety of platforms with-

9

out the need of installing large number of platform-dependent packages (like in the case of keras/tensorflow). The inference procedure was implemented using Efficient Java Matrix Library (EJML) [12], which is optimized for speed and is thread safe, matching the requirements of the CLAS12 reconstruction software.

The analysis and data visualization for this article was done using GROOT data visualization package [13] developed for CLAS12 software infrastructure (in Java) and is included in the Java data analysis library for high energy physics Jas4pp [14].

ANALYSIS OF TRACK RECONSTRUCTION WITH AI

After implementing the track identification service in the CLAS12 reconstruction software, the outputs from the conventional tracking algorithm and AI-assisted tracking algorithm were analyzed event by event to assess the improvements of tracking.

Particle Reconstruction efficiency

The Neural Network for track classification was trained on experimental data after it was processed with conventional tracking. Tracks that had "good" fit quality and were tracked back to the target location were used as training samples for both the MLP classifier and Auto-Encoder corruption-recovery network. Performance was assessed on data recorded at CLAS12 production settings, specifically with 45 nA electron beam impinging on a 5 cm-long hydrogen target for an instantaneous luminosity close to $0.6 \times 10^{35} cm^{-2} s^{-1}$.

The results are shown in Figure 8, where the dependence of number of "good" reconstructed negatively charged tracks are shown as a function of particle momentum (top row), polar angle in laboratory frame (middle row) and interaction vertex (bottom row). The reconstructed distributions from conventional tracking are plotted with filled histograms and the tracks reconstructed using assistance from AI are plotted with solid lines. As can be seen from the figure, there is a large gain in the number of reconstructed tracks with the 5 super-layer configuration compared to full 6 super-layer tracks. Typically for nominal 45 nA experimental data increase in track efficiency averages about $3\% - 6\%$ for 6-sperlayer tracks and to $70 - 120\%$ for 5-superlayer tracks. In conventional data reconstruction, tracks that are identified with 5 super-layers usually comprise about 10% of all reconstructed tracks,
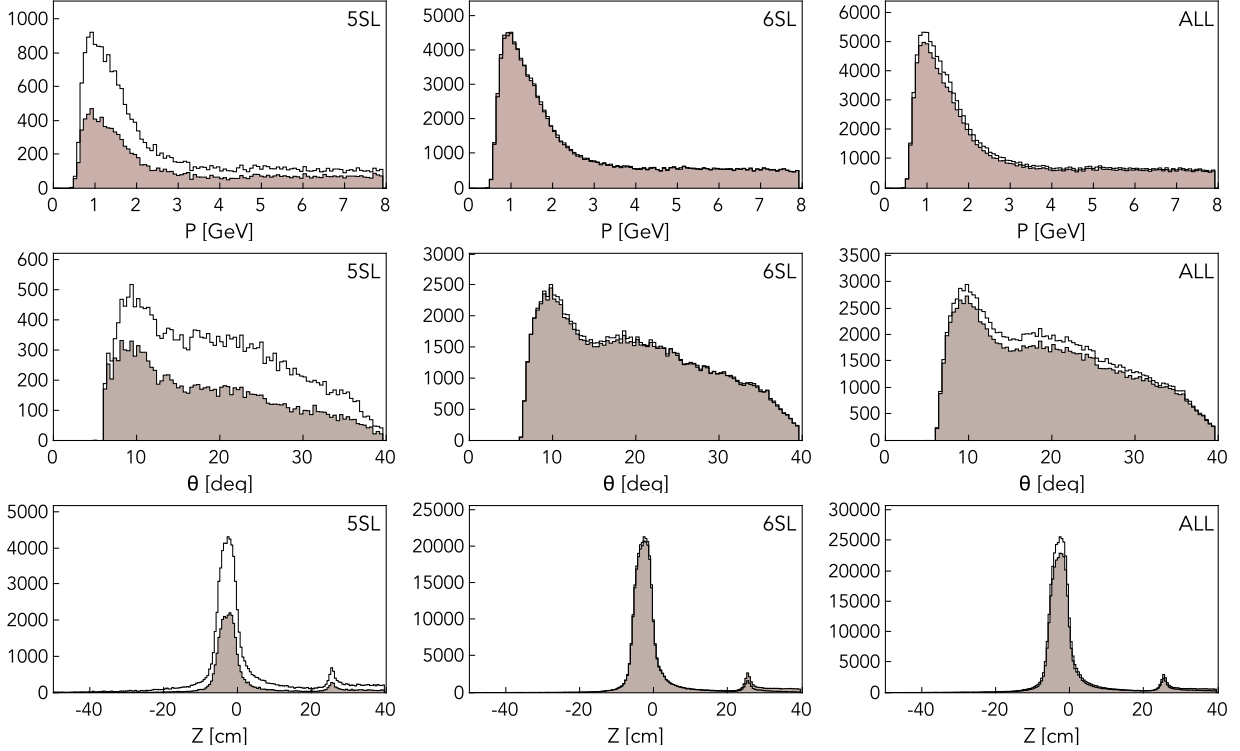
FIG. 8: Comparison of number of tracks reconstructed with the conventional algorithm (filled histograms) vs AI-assisted tracking code (open black outline histograms) as a function of momentum, polar angle and particle interaction vertex. The comparison is shown for 5 super-layer, 6 super-layer tracks (left two columns), and the total number (right column).

and a significant increase in identification of such tracks leads to overall tracking efficiency increase of $10\% - 15\%$.

Track Configuration	Conventional	AI Assisted	Gain	Relative
6 Super-Layer	242,145	256,175	14030	1.0579
5 Super-Layer	24,155	52,839	28684	2.1875
All	267,339	309,058	51719	1.1561

TABLE I: Summary of reconstructed tracks and gain with assistance from Artificial Intelligence algorithm.

The comparison of 5 and 6-segment track statistics and their relative gain is summarized in Table I. As can be seen from the table, the gain in 6-segment tracks is about 5.7% but

with significant gain in 5 super-layer tracks the overall gain in reconstructed tracks elevates to $> 15\%$. These results are intuitive since the number of track candidates composed of 5 super-layers, assuming a uniform number of clusters in the six super-layers, is significantly higher than 6 super-layer track candidates, and in our tests AI performs better in choosing the right combination with increasing combinatorics.

Luminosity Dependence

As discussed in the previous section, AI is more efficient than the conventional track-finding algorithm in identify good candidates from a large pool. One would expect that, if the number of combinations decreases, the efficiency of the conventional track selection should approach the efficiency of AI-assisted track identification. Similarly, when the number of combinations increases the gain of AI over the conventional algorithm should increase. Based on this we expect AI to perform better in higher luminosity, higher background settings. To evaluate the AI-assisted tracking efficiency dependence on background we analyzed several different runs that were taken in different conditions (i.e. beam current) ranging from 5 nA to 70 nA. To determine the tracking efficiency we first calculated the number of electrons (N_e) detected in the analyzed data sample (typically one run, or 2 $hours$ of data taking) and then the number of positive and negative hadrons that were detected with the electron inclusively (N_{h^+e} and N_{h^-e} respectively).

Then the efficiency for the data set was calculated as:

$$L_t^+ = \frac{N_{h^+e}}{N_e}, L_t^- = \frac{N_{h^-e}}{N_e} \tag{1}$$

where L_t^+ is the efficiency of positively and L_t^- is the efficiency of negatively charged particles respectively. In order to estimate the charged particle reconstruction efficiency as a function of the beam current, the multiplicity, $L_t^{+/-}$, is fitted with a linear function:

$$L_t^{+/-} = a + b \times I \tag{2}$$

Here a and c are the fit parameters and I is the beam current. Then it was assumed that the reconstruction efficiency, $E = 1$ at $I = 0$ nA:

$$E^{+/-} = 1 + c \times I \tag{3}$$

with $c = \frac{b}{a}$. The slope parameter c represents the variation of the reconstruction inefficiency per nA [15].
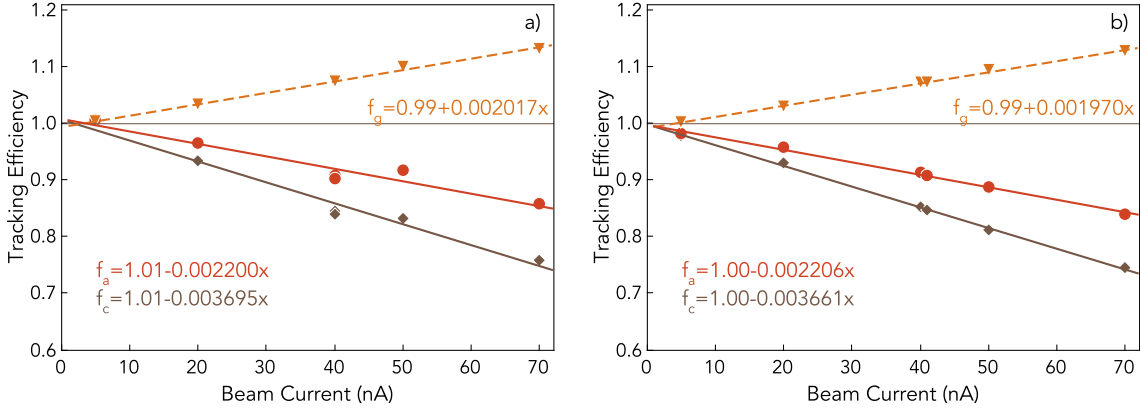

FIG. 9: Tracking reconstruction efficiency of the conventional algorithm (diamonds) is compared to AI-assisted tracking efficiency (circles) for positively and negatively charged particles as a function of beam current (luminosity).

The comparison of tracking efficiency as a function of beam current (luminosity) can be seen in Figure 9 where $E^{+/-}$ are shown for positively and negatively charged particles separately. AI-assisted tracking performs significantly better for any given luminosity (beam current) and the decrease of efficiency is much slower as a function of luminosity, 0.22% per nA versus 0.40% per nA for conventional tracking. This is expected and consistent with the assumption that with increased combinatorial background (increased number of track candidates to consider), AI performs better in choosing the best track candidate. We established that AI-assisted tracking leads to more tracks reconstructed for any given beam current setting. The next thing to check is what is the impact of increased track reconstruction efficiency on physics analysis.

Physics Impact

To measure the implications of track reconstruction efficiency improvements on physics analysis, we considered two event topologies with two and three particles in the final state, respectively. The data for analysis were taken with $10.6\ GeV$ electron beam incident on $5\ cm$ liquid hydrogen target, with a beam current of $45\ nA$. We selected events where an

13

electron was detected in the forward detector, and then isolated events where there was an additional positively charged pion (π^+) along with an electron and no other charged particle. The second topology required two pions along with the electron, one positively charged and one negatively charged. The two chosen topologies are denoted by $H(e, e'\pi^+ X)$ and $H(e, e'\pi^+\pi^- X)$. In both cases there is a visible peak of a missing nucleon in the missing mass distribution of detected final state, that we can use to measure the impact of efficiency on physics outcome.
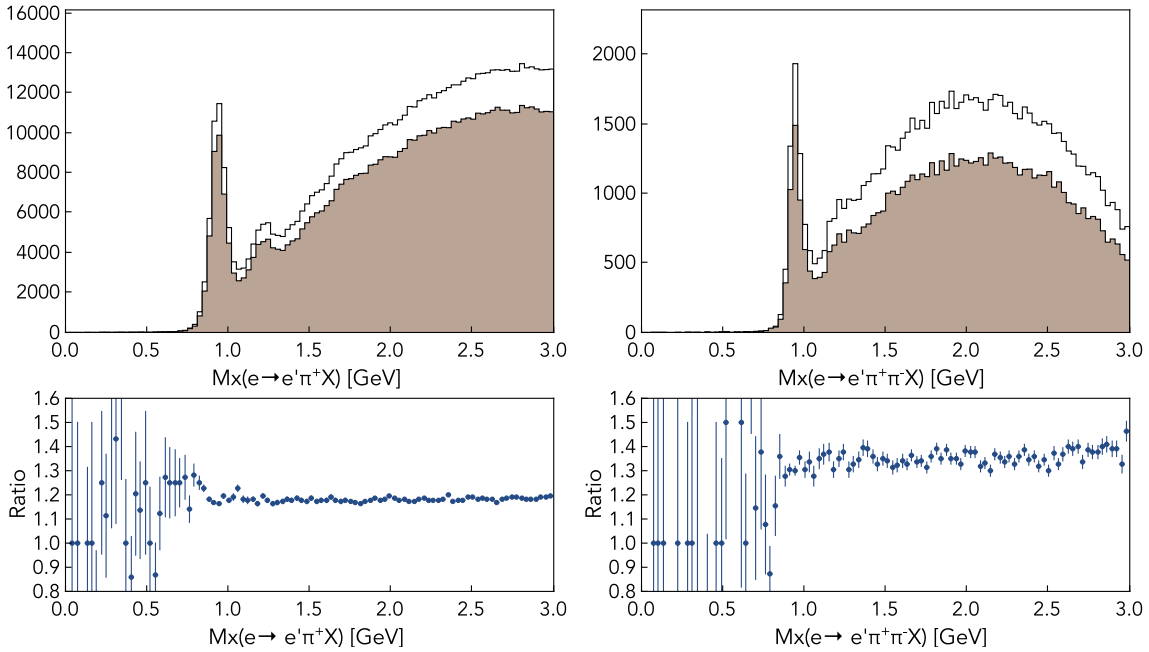

FIG. 10: Reconstructed missing mass distribution for $H(e, e'\pi^+ X)$ and $H(e, e'\pi^+\pi^- X)$ reactions (top row) using the conventional track reconstruction algorithm (filled histogram) and AI-assisted track reconstruction (black line histogram). The ratios of the two histograms are shown on the bottom row.

The distributions of missing mass for both final state topologies are shown in Figure 10, where the plots on the top row are missing mass of $H(e, e'\pi^- X)$ and $H(e, e'\pi^+\pi^- X)$, where the filled histogram is calculated from particles reconstructed by the conventional tracking algorithm, and the histogram with black outline are the same distributions calculated from particles that were reconstructed using suggestion from Artificial Intelligence. As can be seen from the figure, there is significant increase in the number of events in the region of the nucleon peak for AI assisted tracking. The ratios of the two histograms (AI-assisted divided

14

by conventional) can be seen on the bottom row of Figure 10. As can be seen from the figure the increase in statistics is uniform over the whole range of the missing mass indicating no systematic abnormalities for AI-assisted tracking. The ratio also indicates that there is an increase for the number of events of about 15% for $H(e, e'\pi^+ X)$ final state and $30 - 35\%$ for the $H(e, e'\pi^+\pi^- X)$ final state. Further studies show that improvements in statistics are larger for higher luminosity (higher incident beam current), which is consistent with our studies of increased efficiency of single particle reconstruction.

SUMMARY

In this paper we present results of the analysis of experimental data from the CLAS12 detector obtained with assistance of Artificial Intelligence to identify tracks from the hits in drift chambers. This work is based on two neural networks developed to classify track candidates from given cluster combinations [4] and to identify missing cluster positions in tracks that do not have complete 6 cluster configuration [7]. After implementing these networks into the CLAS12 reconstruction workflow, AI was able to identify "good" track candidates and pass them to the tracking code to be analyzed in parallel to conventional algorithms that chooses "good" track candidates iteratively considering all possible combinations. Our studies showed that AI-assisted tracking performs better than conventional track identification algorithm, and leads to track reconstruction efficiency increase of $10 - 12\%$ for beam current of 45 nA. The AI also performs better with increasing background (i.e. with increased incident beam current) and improves the efficiency loss from 0.37% per nA to 0.22% per nA. This increased track reconstruction (identification) efficiency directly impacts the outcome of physics analysis, where it leads to an increase of statistics of $15\% - 35\%$, depending on how many particles are detected in the final state. This has significant implication on the choice of experimental running conditions since with increased efficiency the required statistical significance can be reached in a shorter time by running at higher beam current (luminosity). Already collected experimental data can be re-processed with the AI-assisted tracking code which can increase the statistics available for physics analysis up to 35%. Therefor, both future experiments and already completed ones will benefit from this novel development.

Another important outcome of this development was a reduction in data processing times.

Since track candidates were identified by AI, there were fewer marginal quality tracks picked to be analyzed and then later dropped due to non-convergence of Kalman filter, leading to tracking code speedup of 35%.

In summary we proved that AI assistance in tracking is a good approach, and leads to improvements in tracking code speed and efficiency. Using AI leads to a very small and simple codebase, comprised of composing track candidates and feeding them to the neural networks. We also found that performance keeps improving with constant training on new data. We intend to continue this development in extending the approach to other tracking detectors of the CLAS12, and possibly try to adapt our approach for other experimental detector setups at Jefferson Lab.

ACKNOWLEDGMENTS

[1] V. Burkert *et al.*, "The CLAS12 Spectrometer at Jefferson Laboratory," *Nucl. Instrum. Meth. A*, vol. 959, p. 163419, 2020.

[2] M. Mestayer *et al.*, "The CLAS12 drift chamber system," *Nucl. Instrum. Meth. A*, vol. 959, p. 163518, 2020.

[3] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, pp. 35–45, 03 1960.

[4] G. Gavalian, P. Thomadakis, A. Angelopoulos, V. Ziegler, and N. Chrisochoides, "Using Artificial Intelligence for Particle Track Identification in CLAS12 Detector," 8 2020.

[5]

[6] F. Murtagh, "Multilayer perceptrons for classification and regression," *Neurocomputing*, vol. 2, no. 5, pp. 183 – 197, 1991.

[7] G. Gavalian, "Auto-encoders for Track Reconstruction in Drift Chambers for CLAS12," 9 2020.

[8] V. Gyurjyan, D. Abbott, J. Carbonneau, G. Gilfoyle, D. Heddle, G. Heyes, S. Paul, C. Timmer, D. Weygand, and E. Wolin, "CLARA: A contemporary approach to physics data processing," *J. Phys. Conf. Ser.*, vol. 331, p. 032013, 2011.

[9] V. Ziegler *et al.*, "The CLAS12 software framework and event reconstruction," *Nucl. Instrum. Meth. A*, vol. 959, p. 163472, 2020.

[10] "Keras library." `https://keras.io/`. Accessed: 2020-07-14.

[11] Sevarac.Z, "Deep Netts - Deep Learning in Java." `http://www.deepnetts.com`.

[12] "Efficient Java Matrix Library (EJML)." `http://ejml.org`.

[13] "Groot data visualization library." `https://github.com/gavalian/groot`. Accessed: 2022-01-26.

[14] S. V. Chekanov, G. Gavalian, and N. A. Graf, "Jas4pp — A data-analysis framework for physics and detector studies," *Comput. Phys. Commun.*, vol. 262, p. 107857, 2021.

[15] S. Stepanyan *et al.*, "CLAS12 FD charge particle reconstruction efficiency and the beam background merging," *CLAS12-NOTE, 2020-005*, 2020.

[16] DOE, "Spotlight Falls on Science Policy," *Science, Vol 229, Issue 4708*, 1985.

[17] "Google." `http://www.google.com`.