Using Artificial Intelligence for Particle Track Identification in CLAS12 Detector - Part One

Gagik Gavalian*[1], Polykarpos Thomadakis†[2], Angelos Angelopoulos †[2], Veronique Ziegler[1], and Nikos Chrisochoides[2]

[1] *Thomas Jefferson National Accelerator Facility, Newport News, VA 23606*
[2] *Department of Computer Science, Old Dominion University, Norfolk, VA 23569*

Abstract

In this article we describe the development of machine learning models to assist the CLAS12 tracking algorithm by identifying the best track candidates from combinatorial track candidates from the hits in drift chambers. Several types of machine learning models were tested, including: Convolutional Neural Networks (CNN), Multi-Layer Perceptron (MLP) and Extremely Randomized Trees (ERT). The final implementation was based on an MLP network and provided an accuracy $> 99\%$. The implementation of AI assisted tracking into the CLAS12 reconstruction workflow and provided a 6 times code speedup.

*Corresponding author
†Co-first author. Contributed equally.

1 Introduction

The CLAS12[1] detector is built around a six-coil toroidal magnet which divides the active detection into six azimuthal regions, called "sectors". The torus coils are approximately planar. Each sector subtends an azimuthal range of 60° from the mid-plane of one coil to the mid-plane of the adjacent coil. The sector mid-plane is an imaginary plane which bisects the sectors azimuth.

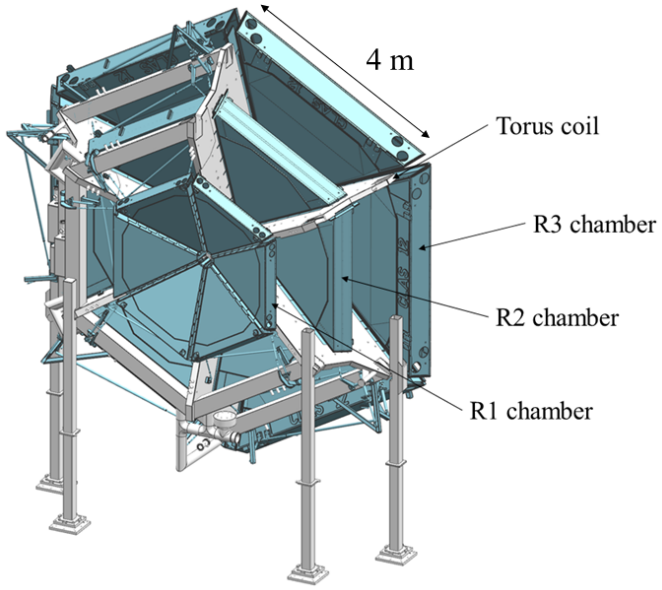

Figure 1: CLAS12 Drift Chambers inside the toroidal field. Three different Regions (R1, R2 and R3) consist of two Super Layers of chambers with 6 layers of wires in the Super Layer.

Charged particles in the CLAS12 detector are tracked using drift chambers[2] inside the toroidal magnetic field. There are six identical independent drift chamber systems in CLAS12 (one for each azimuthal sector). Each sector of drift chambers consists of three chambers (called "regions"), shown in Figure 2 b) separated from each other along the beam direction.

Each region of drift chambers consists of two parts called super-layers with wires in each super-layer running at 6° relative to the main axis of the chambers, shown on Figure 1 b). Particles passing through the drift chambers leave a signal in the wires.

Reconstruction Procedure Reconstruction of tracks from the hits in drift chamber relies on separating hits that belong to a track by removing background hits, then trying to fit the track using Kalman-Filter to measure track parameters (momentum, and angles). An example is shown in Figure 3, where three example events are shown with all clusters reconstructed in one of the sectors of drift chamber, the dark colored clusters belong to a particle track reconstructed by conventional Kalman-Filter. The lighter colored clusters are background noise that were not reconstructed as tracks.

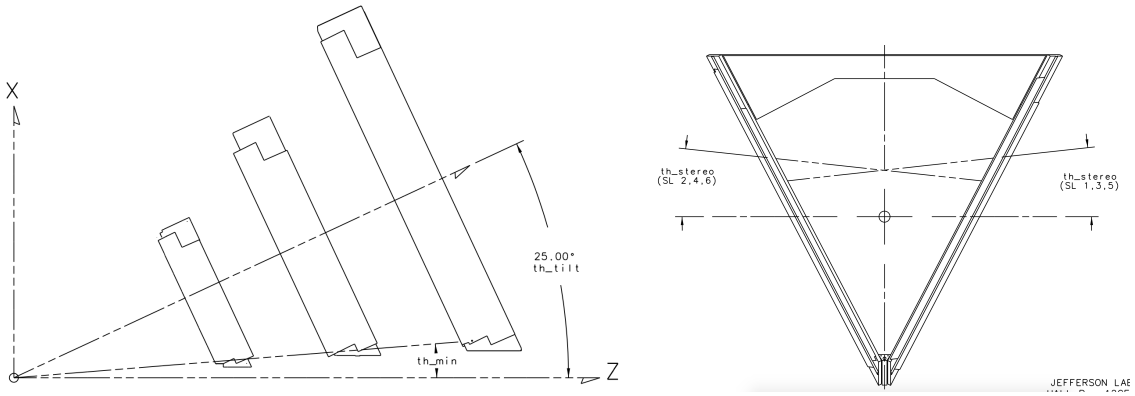

Figure 2: Side and front view of drift chambers. a) the layout for one sector showing three regions of drift chamber. Z-axis is direction of incoming beam. b) diagram of wire directions in each super layer.

Motivation Tracking is the most computationally intensive part of data reconstruction in CLAS12 (about 94% of the reconstruction time). The conventional algorithm first identifies clusters in each super-layer. The track candidates are formed by considering all combinations of 6 clusters (one from each super layer). For each track candidate the algorithm determines initial momentum (by polynomial fit) and then processes it iteratively through Kalman-Filter to get the momentum and quality of the fit. The tracks that do not pass quality criteria are discarded. In a high-luminosity environment there are many plausible combination of clusters that can form a track and all of them have to be considered.

The purpose of this work is to investigate a machine learning approach to identifying tracks from combinatorial track candidates.

2 Related Work

Farrel et al.[3] used space-point data arranged as sequences and connected graphs. For the first approach a RNN is presented with a sequence of hit coordinates and predicts the coordinates of the next hits. The authors experiment with this approach using a traditional regression model as well as a Gaussian distributions model for the produced predictions. Their results on low occupancy tests show very good performance for both models in this approach. In the second approach they present the data as a graph of connected hits and use a Graph Neural Networks (GNN) to perform track reconstruction. Two paths are examined, one uses GNNs to classify graph nodes as part of the same track while the other performs classification on graph edges instead and is able to detect many tracks at once.

In another effort, Farrel et al.[4] investigated the applicability of image-based and point-based models. In the image-based approach they presented each layer of the detector as an image which is fed to a RNN that predicts the correct location of a target track. A variation of this approach uses a CNN that is fed with the entire detector image and classifies the points belonging to the same track. While this approach performs well the authors argue that it would not be possible to scale it up as the dimensionality and sparsity increases. In

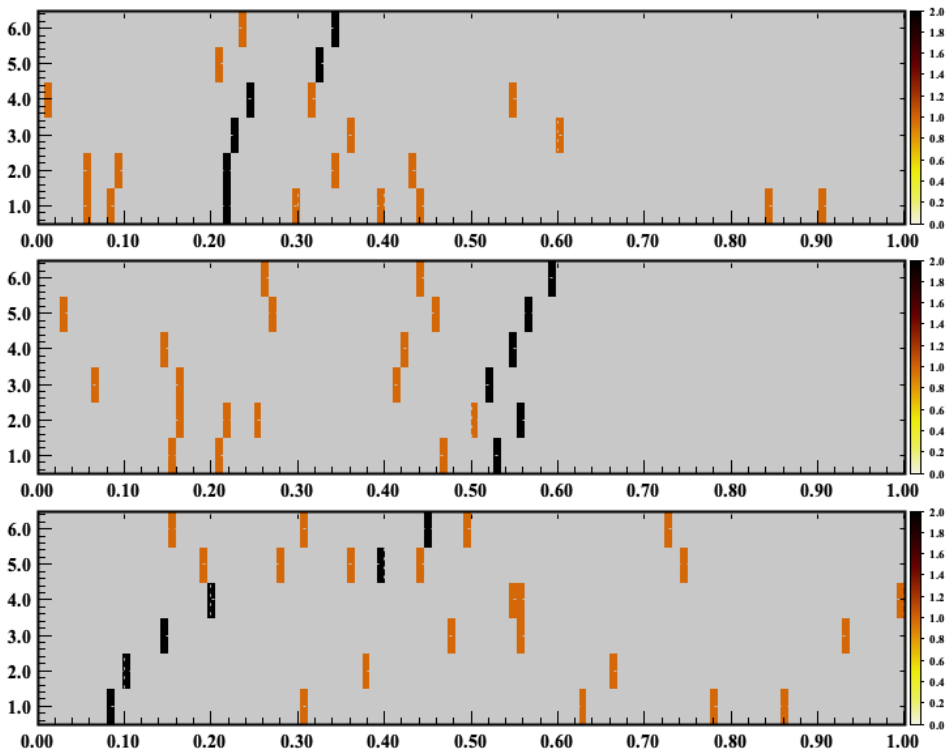

Figure 3: Drift chamber clusters shown for three events. The lines represent a cluster in each super layer, the dark colored clusters belong to a track reconstructed by conventional tracking algorithm.

the point-based approach, the spacepoints of the detector are sorted before being fed to a RNN. The RNN can then classify the spacepoints to the respective target tracks. The results presented on this work are collected by running on toy data and not real experiments. Tsaris et al.[5] employed the models developed here to more realistic datasets and show that they can achieve promising results.

In [6] Baranov et al. propose a two-step technique consisting of a hit preprocessing step, followed by the employment of a deep neural network that classifies detected hits as parts of a particle track. In the preprocessing step, the authors use directed search algorithm to filter out the possible candidates and narrow the search space. The deep neural network that follows consists of a one-dimensional convolutional layer followed by a RNN of two GRU layers. The filtered hits are fed to the neural network which is then able to classify them as valid points and assign them to their respective track. With this approach they achieve performance of up to 97.5 % for a dataset created by using a Monte-Carlo generator. An improved version of this model, in terms of performance in accuracy and speed, is presented in [7]. The authors here replace the two-step approach with a single neural network by enhancing the neural network presented in the original work with a regression component that estimates the location of a hit in the next detector layer.

3 Data and Methods

Data Selection First we have to create a data set that machine learning algorithms can be trained with. We use data that already has been processed by the conventional track reconstruction algorithm and extract good quality tracks and write out their hit patterns. An example of one event is shown on Figure 3, where all hit clusters are shown along with clusters that make a good track. The training data set consists of 6 features, which are average wire number of the cluster in each super-layer (the wire number is normalized to 112, which is the number of wires in each super-layer). The cluster combinations that form a real track (which comes from reconstruction) are labeled as positive sample, another combination of 6 clusters (one from each super-layer), that did not get identified as a valid track is labeled as negative sample. For the training data, we balanced negative and positive samples, and with each positive track from an event one negative example was provided. Because many negative samples are present in the event alongside the positive sample, the choice of negative example for the network is arbitrary, but we did investigate several options, which is discussed in later chapters.

Track Candidate Classification In order to solve this problem we developed three separate supervised machine learning models: an extremely randomized trees model (ERT), a multi-layer perceptron (MLP), and a convolutional neural network (CNN). The ERT and MLP models were developed using the scikit-learn library [8] and run on the CPU while being able to utilize multiple processing cores. The CNN model was developed using Keras [9] and runs on the GPU. Training and evaluation of these models utilized tens of thousands of labeled particle trajectory samples from CLAS12 detector experiments. Samples included multiple particle tracks, of which one was valid. All these models were developed to have high performance, as they need to be capable of real-time inference and high-throughput.

In order to determine our models' accuracy, we devised and utilized several metrics in addition to using the standard accuracy metric in scikit-learn and Keras. These new accuracy metrics are:

1. **A1**: The ratio of samples where the valid particle track was correctly detected.
2. **Ac**: The percentage of A1 for which there were invalid tracks misidentified as valid ones (false positives).
3. **Ah**: The percentage of A1 for which the valid particle track had the highest probability of being valid out of all tracks in a sample.
4. **Af**: The ratio of samples where the valid track was not detected (false negatives). This metric as very critical for us to minimize, as we don't want to miss valid particle tracks.

3.1 Extremely Randomized Trees

Architecture The extremely randomized trees model was built using the extremely randomized trees classifier [10] from the *scikit-learn* [8] library. Being an extremely randomized

trees model, it is a random forest where node splits are selected from a set of randomly generated possible splits. From these random possible splits, the one that results in the highest score is selected. For the parameters of the model we used three hundred estimators (decision trees), with no limit in the number of features to be considered when splitting. In addition, we used the information gain (entropy) split quality criterion. The rest of the parameters were kept as their default values. The input for the model is a 6-feature dataset.

Results The ERT model was tested on 29,810 samples, with a total of 291,367 rows of data. The ERT model achieved very good results, with 100% training accuracy and 99.13% overall testing accuracy. The model had 100% accuracy in identifying the valid particle track in a sample (A1 metric). In approximately 6.14% of the samples where the valid track was identified, there were also some false positives (Ac metric). In 100% of these samples, the valid track had the highest probability of being valid out of all other tracks (Ah metric). Finally, there were no samples where the valid track was not detected (Af metric). These results are compiled in Table 1. Table 2 shows the confusion matrix generated by the results from evaluating the model with the testing dataset.

Metric	**Result**
Training Accuracy	100.00%
Testing Accuracy	99.13%
A1	100.00%
Ac	6.14%
Ah	100.00%
Af	0.00%
Time to Train	19 sec
Time to Predict/sample	306 μs

Table 1: Shows the results for some metrics used to evaluate the model. The ERT model executed on a multi-core CPU.

	Predicted: Invalid	**Predicted: Valid**
Actual: Invalid	259,044	2,513
Actual: Valid	0	29,810

Table 2: Shows the confusion matrix generated by the results from evaluating the model with the testing dataset.

3.2 Multi-Layer Perceptron

Architecture The multi-layer perceptron was built using the multi-layer perceptron classifier [11] from *scikit-learn*. The neural network has three hidden layers each with sixty-four neurons. The optimizer chosen was *Adam* [12]. We used a batch size of thirty-two, and an adaptive learning rate, meaning that the learning rate decreases if there is no training loss

decrease of two consecutive epochs. The rest of the parameters were kept as their default values.

Results The MLP model was tested on 29,810 samples, with a total of 291,367 rows of data. The MLP model achieved very good results, with 99.65% training accuracy and 98.68% overall testing accuracy. The model had 99.96% accuracy in identifying the valid particle track in a sample (A1 metric). In approximately 10.77% of the samples where the valid track was identified, there were also some false positives (Ac metric). In 98.88% of these samples, the valid track had the highest probability of being valid out of all other tracks (Ah metric). Finally, the valid track was not detected in 0.04% of samples (Af metric). These results are compiled in Table 3. Table 4 shows the confusion matrix generated by the results from evaluating the model with the testing dataset.

Metric	**Result**
Training Accuracy	99.65%
Testing Accuracy	98.68%
A1	99.96%
Ac	10.77%
Ah	98.88%
Af	0.04%
Time to Train	254 sec
Time to Predict/sample	120 μs

Table 3: Shows the results for some metrics used to evaluate the model. The MLP model executed on a multi-core CPU.

	Predicted: Invalid	**Predicted: Valid**
Actual: Invalid	257,736	3,821
Actual: Valid	11	29,799

Table 4: Shows the confusion matrix generated by the results from evaluating the model with the testing dataset.

3.3 Convolutional Neural Network

Architecture The convolutional neural network was built using Keras [9]. It consists of three convolutional layers (of 32, 64 and 128 filters), with a 3x3 kernel size, each followed by a 2x2 max-pooling layer with same padding and dropout of 0.25, 0.25 and 0.4. Two dense layers, of 128 and two neurons respectively, follow the convolutional layers with a droupout of 0.3 between them. The activation function for all layers except the last one is LeakyReLU, while softmax is used for the last layer. Adam [12] was used as the optimizer and categorical cross-entropy as the loss function. We trained the model for twenty epochs using a batch size of 16.

Results The CNN model was tested on 4818 samples, with a total of 17311 rows of data. The CNN model achieved good results, with 96.11% accuracy in identifying the valid particle track in a sample (A1 metric). In approximately 28.11% of the samples where the valid track was identified, there were also some false positives (Ac metric). In 90.16% of these samples, the valid track had the highest probability of being valid out of all other tracks (Ah metric). Finally, the valid track was not detected in 3.89% of samples (Af metric). These results are compiled in Table 5. Table 6 shows the confusion matrix generated by the results from evaluating the model with the testing dataset.

Metric	**Result**
Training Accuracy	94.26%
A1	96.11%
Ac	28.11%
Ah	90.16%
Af	3.89%
Time to Train	457 sec
Time to Predict/sample	1.2 ms

Table 5: Shows the results for some metrics used to evaluate the model. The CNN model executed on one Tesla V100-SXM2-16GB.

	Predicted: Invalid	**Predicted: Valid**
Actual: Invalid	10707	1786
Actual: Valid	187	4631

Table 6: Shows the confusion matrix generated by the results from evaluating the model with the testing dataset.

4 Accuracy Improvement

Overall, the three models performed well, with the MLP model having an edge over the ERT and the CNN. The two best models were the MLP and the CNN, but while the CNN was negligibly better than the MLP in detecting valid particle trajectories, the MLP had significantly less false positives than the CNN. The results are summarized in Table 7.

Optimization with Training Data Selection When selecting training data we discovered that negative example selection can affect the network accuracy. In each reconstructed event with a valid track there are many track candidates that we have to choose as a false sample. In training sample we balance the number of "true" and "false" samples, so we have to choose only one "false" sample from each event. There are several options to consider. We could take a random track candidate that does not match the good track, or we can

Model Type	A1	Ac	Ah	Af	Training Accuracy	Training Time	Inference Time per sample
ERT	100.00%	6.14%	100.00%	0.00%	100.00%	19 sec	306 μs
MLP	99.96%	10.77%	98.88%	0.04%	99.65%	254 sec	120 μs
CNN	96.11%	28.11%	90.16%	3.89%	94.26%	457 sec	1.2 ms

Table 7: Summarizes the results for our three models for particle trajectory classification. The MLP and ERT models executed on a multi-core CPU, while the CNN executed on one Tesla V100-SXM2-16GB.

choose closest track candidate to the real track. In order to see if the accuracy depends on training sample choice (primarily what we choose as a "false" sample) we decided to make small systematic study.

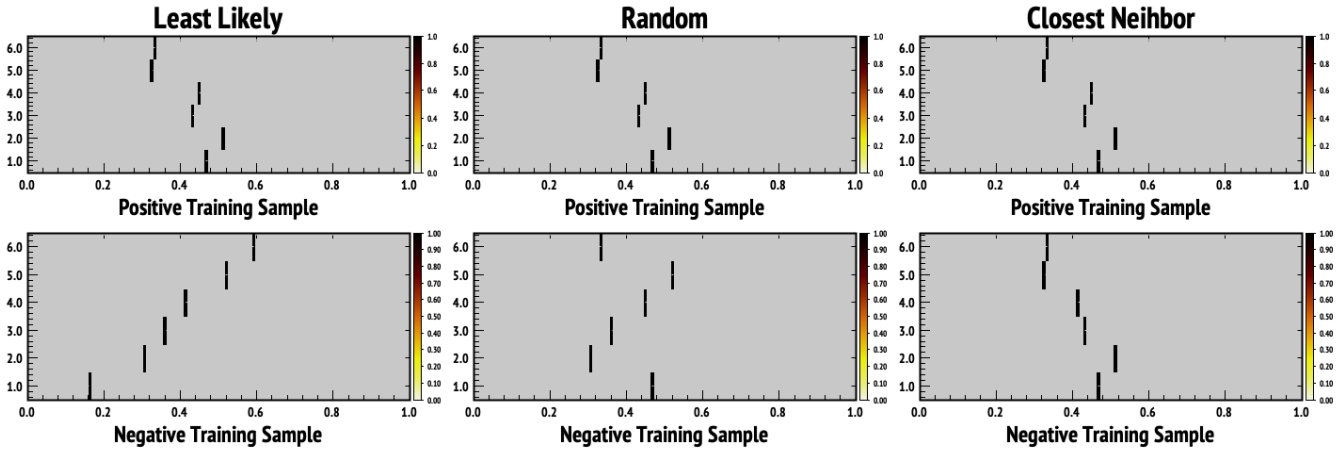

Figure 4: Training sample for one event is shown for three different ways to chose negative sample. On the top row hits belonging to real reconstructed track are shown and on the bottom row different negative sample selections. On the left column negative sample is chosen from hits that as far away from the real track as possible from all combinations called "least likely", on the middle column random combination of clusters are chosen from all possible combinations as a negative sample, on the right column a combination is chosen that is closest to real reconstructed track.

We created three training samples (shown on Firgure 4) and test the network trained on different input data. The samples and choices for "false" track candidate shown on Figure 4 are the following:

- track candidate that looks least like the real track in that event (figure 4 left column)
- track candidate randomly chosen from all combinations (figure 4 middle column)
- track candidate that is closest to the real track, most of the time they differ only by one cluster (figure 4 right column)

Three networks were trained using three data sample and then network evaluation was done on sub sample of "closest neighbor" data.

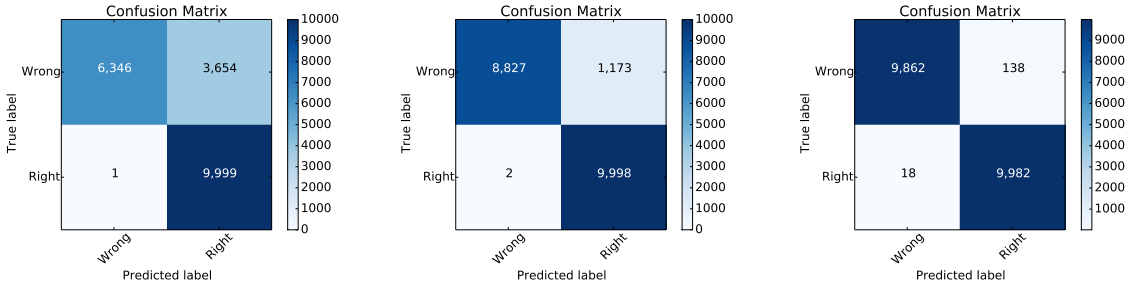

Figure 5: Confusion matrix for different networks. left) trained with data set with least likely track candidates, middle) trained with sample with randomly chosen "false" candidates, right) trained with sample closest neighbor "false" candidates.

As can be seen from confusion matrices the network trained on random and "least likely" sample is able to accurately predict the "true" track, but fails in classifying the "false" track. The interpretation is that because it has not been shown a "false" candidate very close to a "true" candidate, it is not able to separate very well the "true" track from close enough to "true" track candidates. The tests show that network accuracy depends on training sample composition. The best results were achieved by using "closest neighbor" training sample. This method was used for all network evaluations and for the final implementation of training data extraction from experimental data.

5 Case Study - HALL-B

MLP was chosen to be used for CLAS12 data reconstruction due to its accuracy and fast inference speed. The network was implemented and trained on real data samples and was used to provide predictions of track candidates for reconstruction algorithms. The reconstruction algorithm was split into two different services. One of them only used track candidates suggested by the neural network to run them through the Kalman-Filter and fit the track. The second one was the conventional algorithm where all reasonable candidates were considered and fitted. The tests were run using both algorithms in parallel to compare the number of tracks reconstructed by both services.

The results are shown in Figure 6, where the ratio of tracks reconstructed by neural network driven service to conventional algorithm is shown. In some instances, the neural network-driven service reconstructs more tracks than the conventional algorithm.

The AI based service also runs 6 times faster than the conventional algorithm. The big gain in speed comes from reduced number of combinatorial track candidates that have to be fitted by Kalman-Filter.

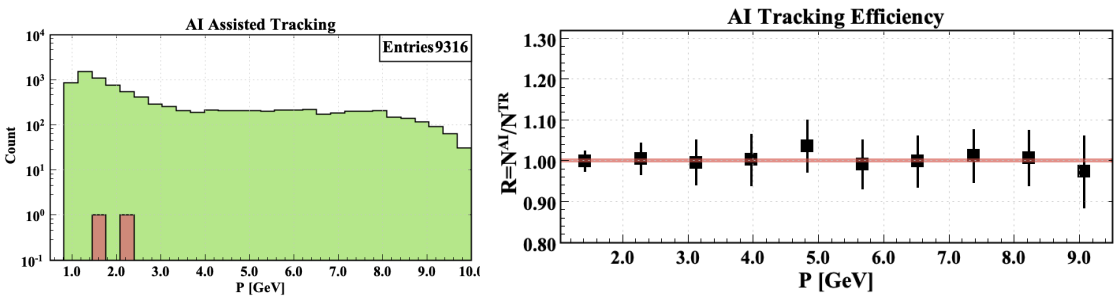

Figure 6: Efficiency of track reconstruction using suggestions from MLP network as a function of particle momentum.

6 Conclusion and Future Work

We have demonstrated the performance of three machine learning models for the track reconstruction process. We have shown that all three models ExtraTrees, MLPs and CNNs perform well for this task with the MLPs showing the best results in terms of accuracy and inference speed on real data. We also run a small systematic study that showed that by training the models on datasets with false samples very similar to true ones, the inference accuracy increases substantially. Finally, we evaluated the performance of the MLP model in comparison to the traditional Kalman-Filter method, which showed an equally good track reconstruction efficiency in approximately 84% less time. In the future, we plan to extend the AI model by introducing a preprocessing step to further filter the candidate tracks. A Recurrent Neural Network (RNN) will be introduced to give an estimate of the path the true track will follow. This estimation will be used to narrow down the amount of candidates the MLP needs to consider.

7 Acknowledgments

This work is in part funded by Jefferson Laboratory, NSF grant no. CCF-1439079, and Richard T. Cheng Endowment. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the funding agencies.

References

[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] S. Farrell, P. Calafiura, M. Mudigonda, Prabhat, D. Anderson, J.-R. Vlimant, S. Zheng, J. Bendavid, M. Spiropulu, G. Cerati, L. Gray, J. Kowalkowski, P. Spentzouris, and A. Tsaris, "Novel deep learning methods for track reconstruction," in *4th International Workshop Connecting The Dots 2018*, 10 2018.

[4] S. Farrell, D. Anderson, J. Bendavid, M. Spiropoulou, J.-R. Vlimant, S. Zheng, G. Cerati, L. Gray, K. Kapoor, J. Kowalkowski, P. Spentzouris, *et al.*, "Particle track reconstruction with deep learning," *31st Annual Conference on Neural Information Processing Systems (NIPS)*, 2017.

[5] A. Tsaris, D. Anderson, J. Bendavid, P. Calafiura, G. Cerati, J. Esseiva, S. Farrell, L. Gray, K. Kapoor, J. Kowalkowski, *et al.*, "Hep. trkx project: Deep learning for particle tracking," in *Journal of Physics Conference Series*, vol. 1085, Institute of Physics, 2018.

[6] D. Baranov, S. Mitsyn, G. Ososkov, P. Goncharov, and A. Tsytrinov, "Novel approach to the particle track reconstruction based on deep learning methods," in *Selected Papers of the 26th International Symposium on Nuclear Electronics and Computing (NEC 2017), Budva, Montenegro*, vol. 2023, pp. 37–45, 2017.

[7] D. Baranov, S. Mitsyn, P. Goncharov, and G. Ososkov, "The particle track reconstruction based on deep neural networks," in *EPJ Web of Conferences*, vol. 214, p. 06018, EDP Sciences, 2019.

[8] "scikit-learn library." `https://scikit-learn.org/stable/`. Accessed: 2020-07-14.

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

[10] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, p. 342, Apr. 2006.

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

[12] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.