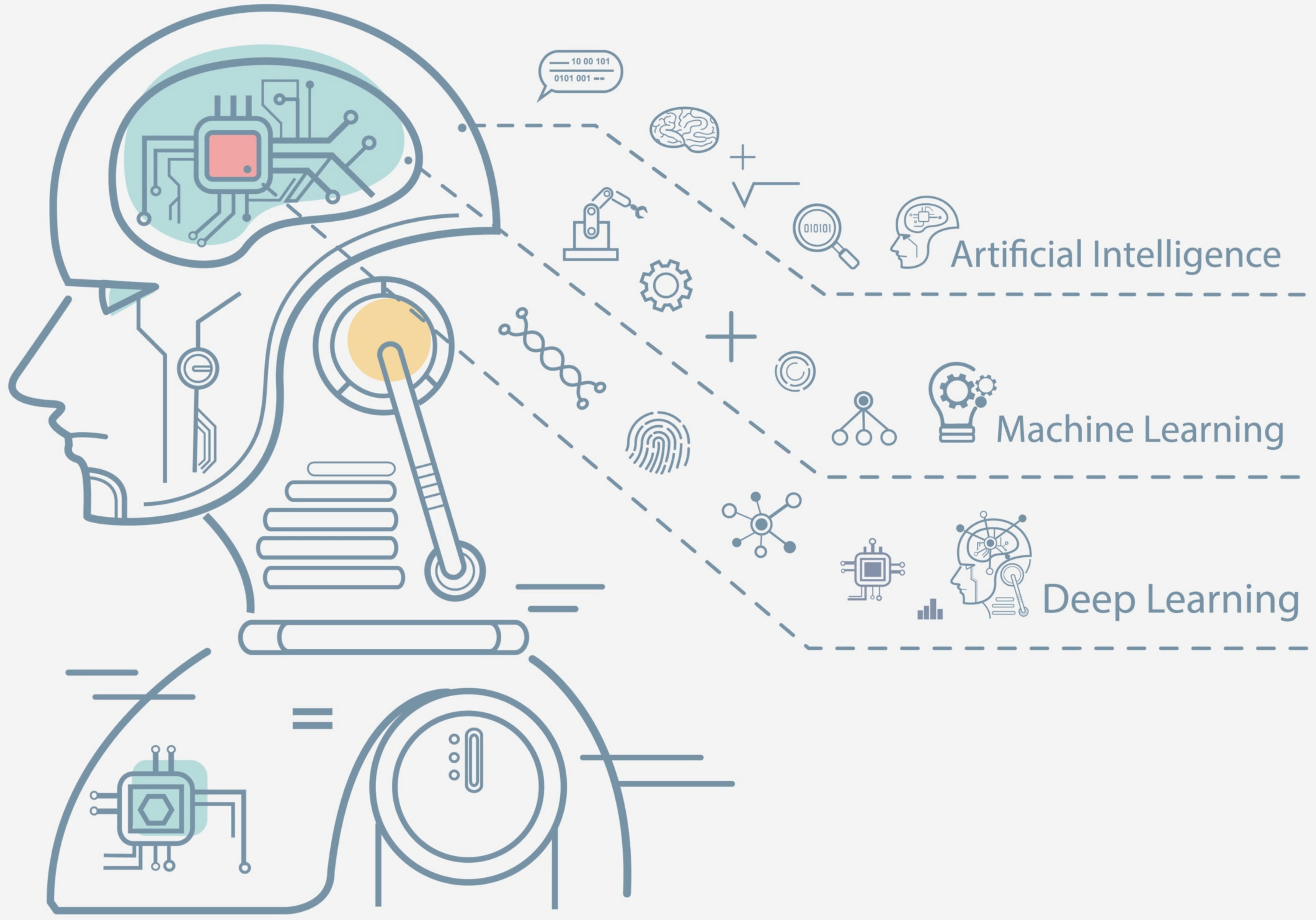
Hall-B AI projects

Track reconstruction and identification with AI

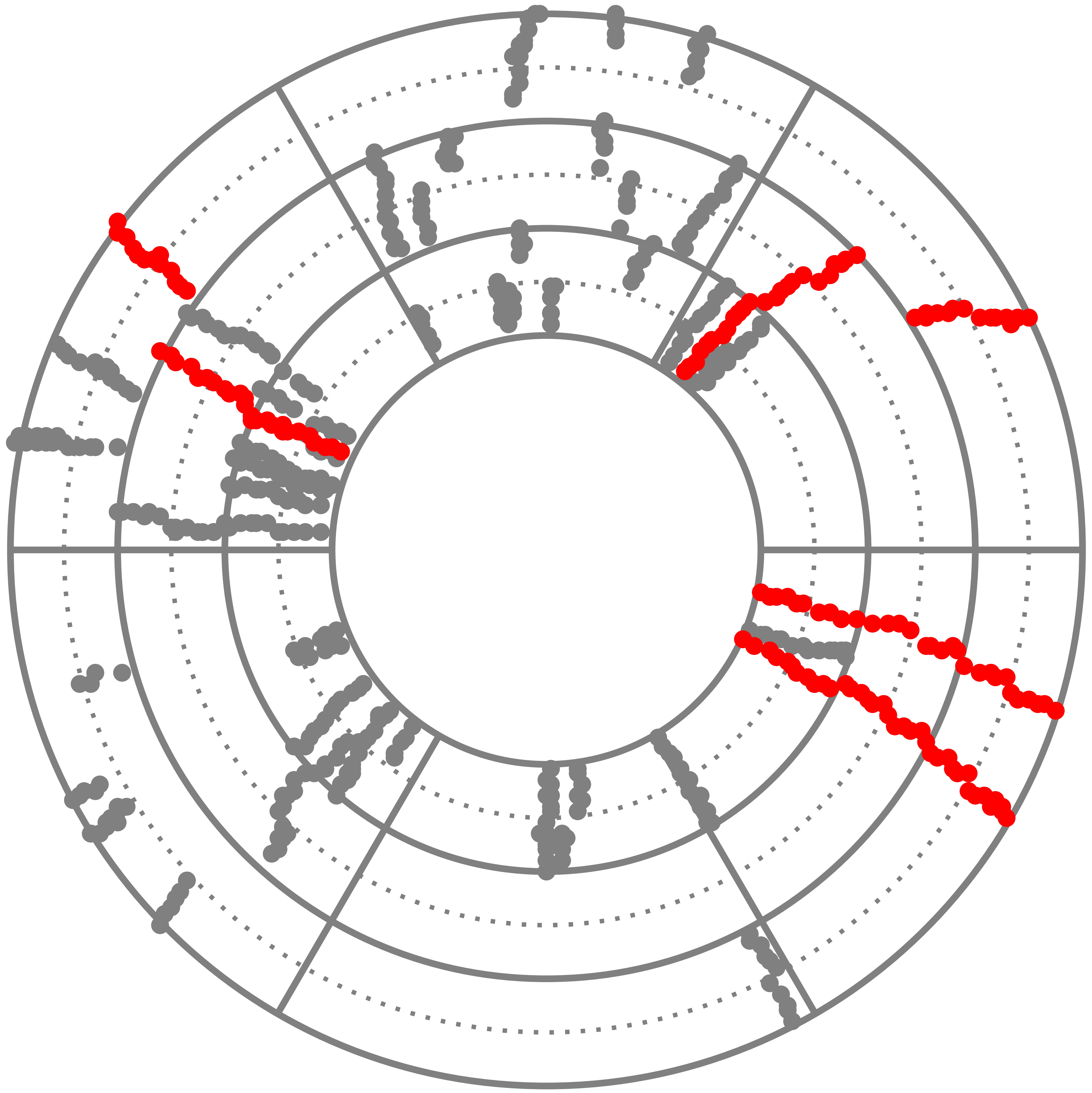
G.Gavalian (JLAB)

Angelos Angelopoulos (CRTC)
Polykarpos Thomadakis (CRTC),
Nikos Chrisochoides (CRTC)
Department of Computer Science,
Old Dominion University, Norfolk, VA, 23529



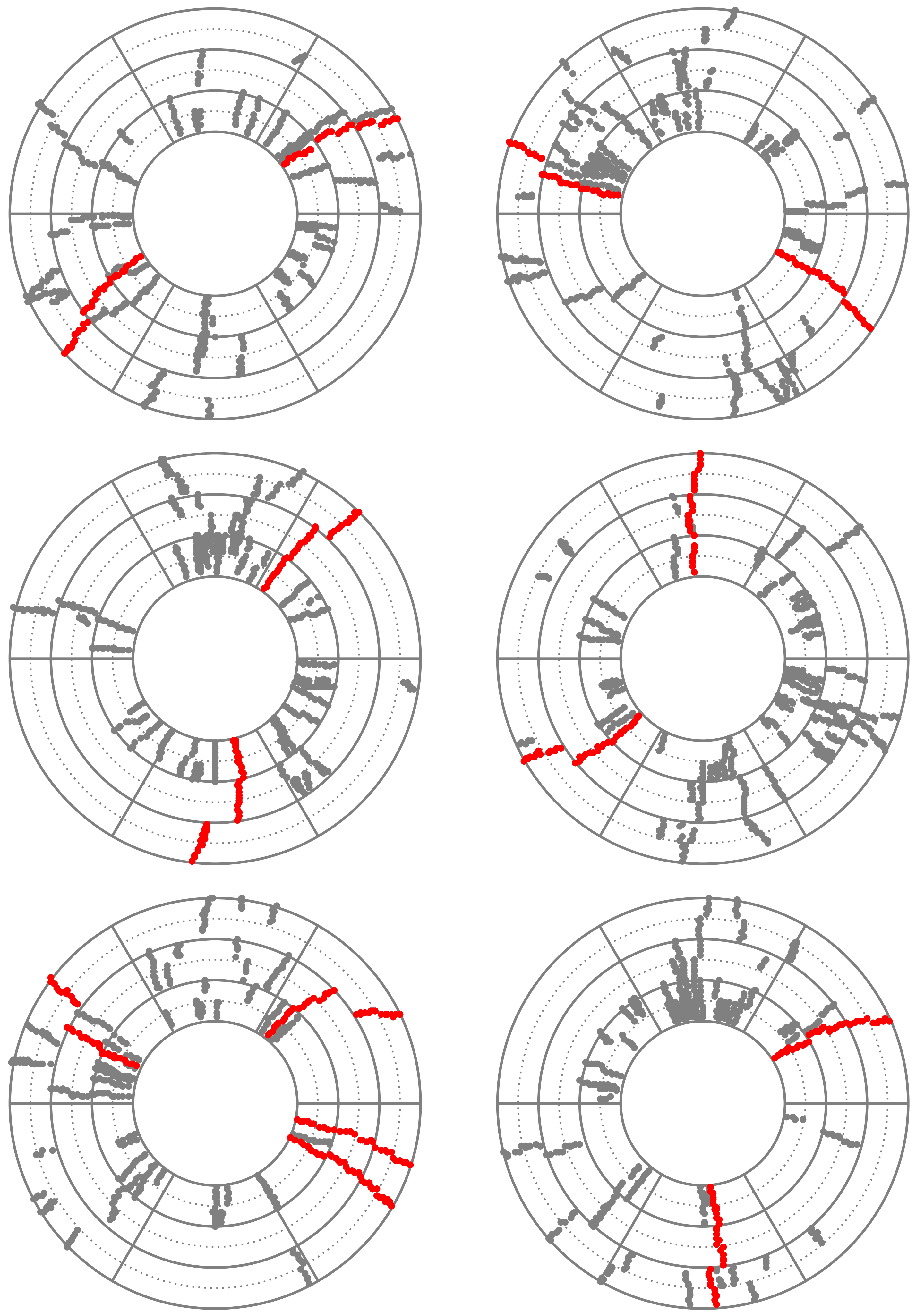
AI Tracking
CLAS12 Tracking with Artificial Intelligence

- Motivation
- Completed AI projects
- Ongoing projects
- Workflow of tracking
- Online AI project(s)
- Summary



AI Tracking

CLAS12 Tracking with Artificial Intelligence



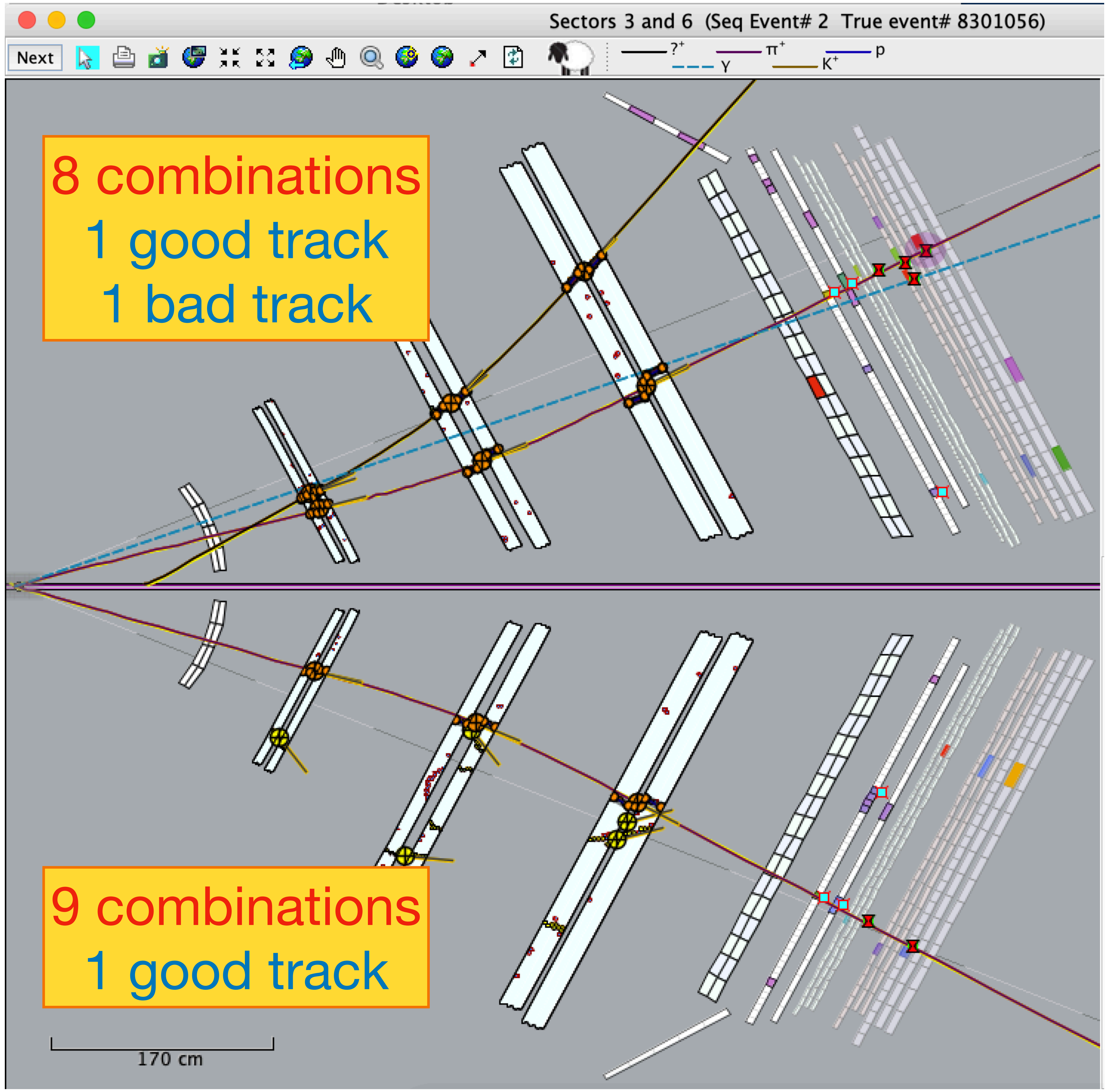
Why Tracking ?

- Tracking is computationally intensive (~94% reconstruction time)
- Many combinations of segments have to be considered for track candidates
- In high luminosity runs efficiency drops due to many noisy hits in region one chamber.
- With holes developing in drift chambers segments can be missing from tracks. AI can help in identifying 5 super-layer tracks.

AI Tracking

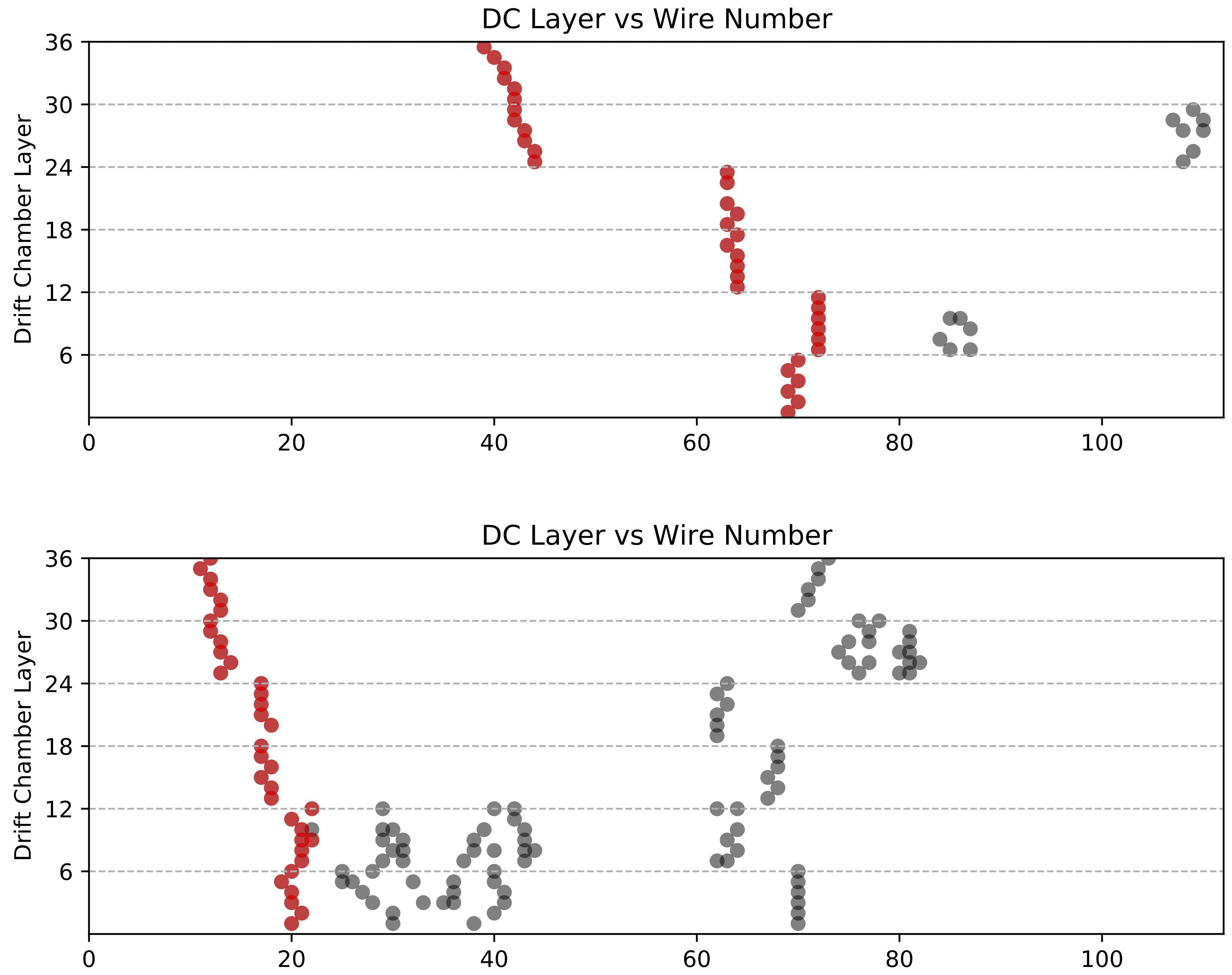
CLAS12 Tracking with Artificial Intelligence

- Tracking is computationally intensive (~94% reconstruction time)
- It relies on fitting tracks with Kalman-Filter
- Reduction of track candidates to fit can lead to significant speed up of the code.
- DC tracking with clusters:
  - Many combinations of clusters to form a track.
  - Many end up not as valid track, though time is spend on fitting them.
  - Even after fitting, some tracks are not traced to the target and have to be discarded.

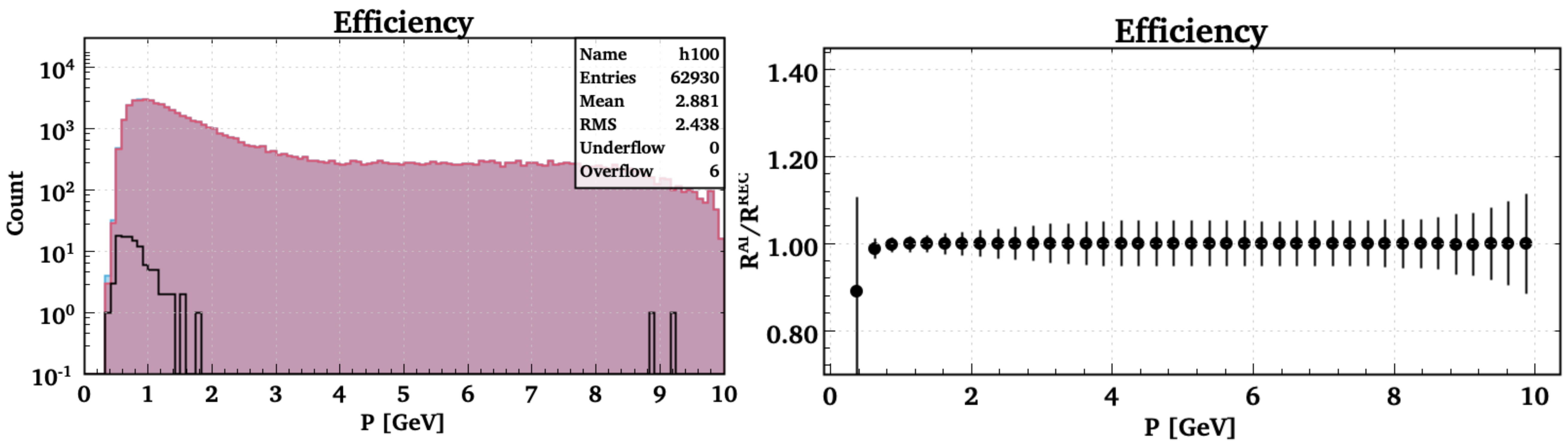


AI Tracking

CLAS12 Tracking with Artificial Intelligence



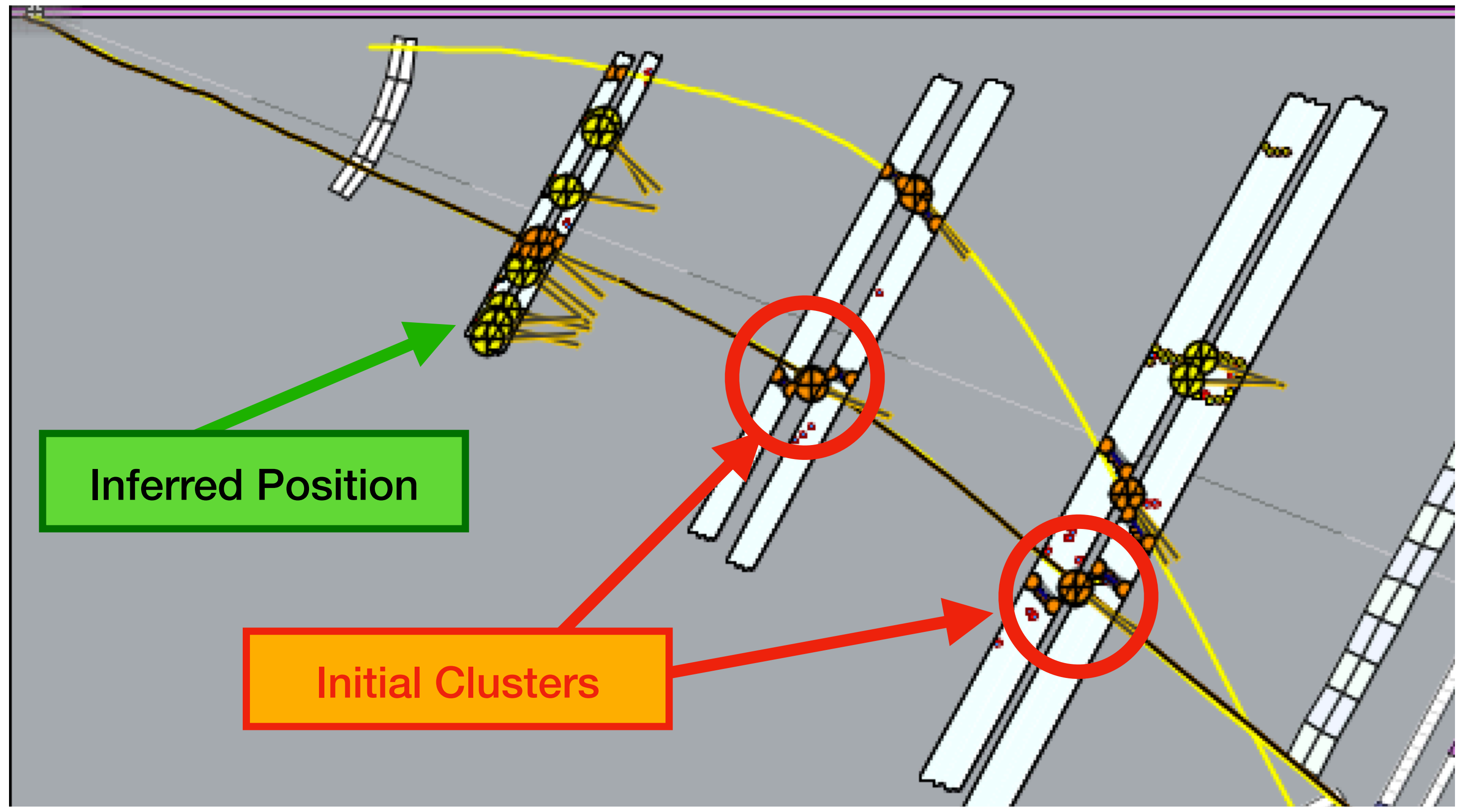
- **Track Candidate classification:**
  - Software is ready for use for identifying track candidates from segment combinations.
  - CLARA service is implemented to provide AI predictions to tracking algorithm.
- **Tracking efficiency with AI**
  - Provided track candidates with AI is nearly 99.7%



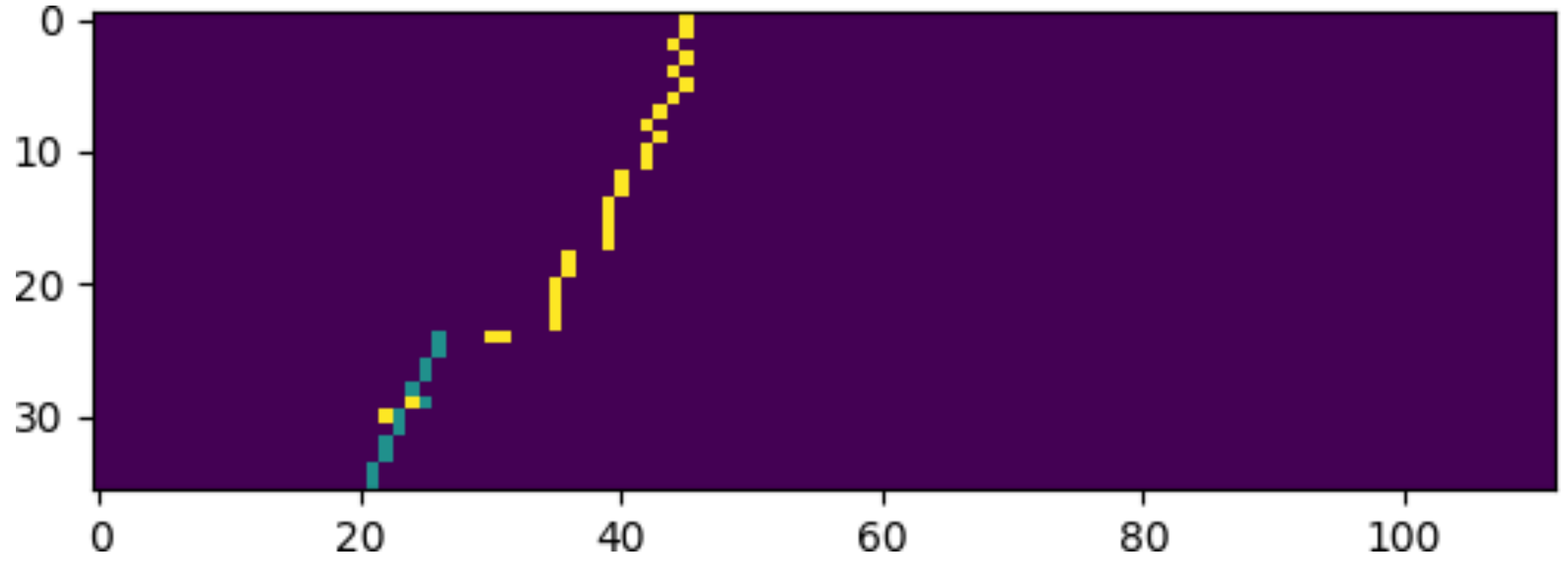
- **Preliminary Tests:**
  - Reconstructed track segments identified are 99.7%
  - Only 113 tracks not identified from 62.9K events
  - Majority of (~82) un-identified tracks are outside of fiducial region.
  - Need fiducial cuts software for selecting the training sample and for running track identification validation.

Track Trajectory Prediction

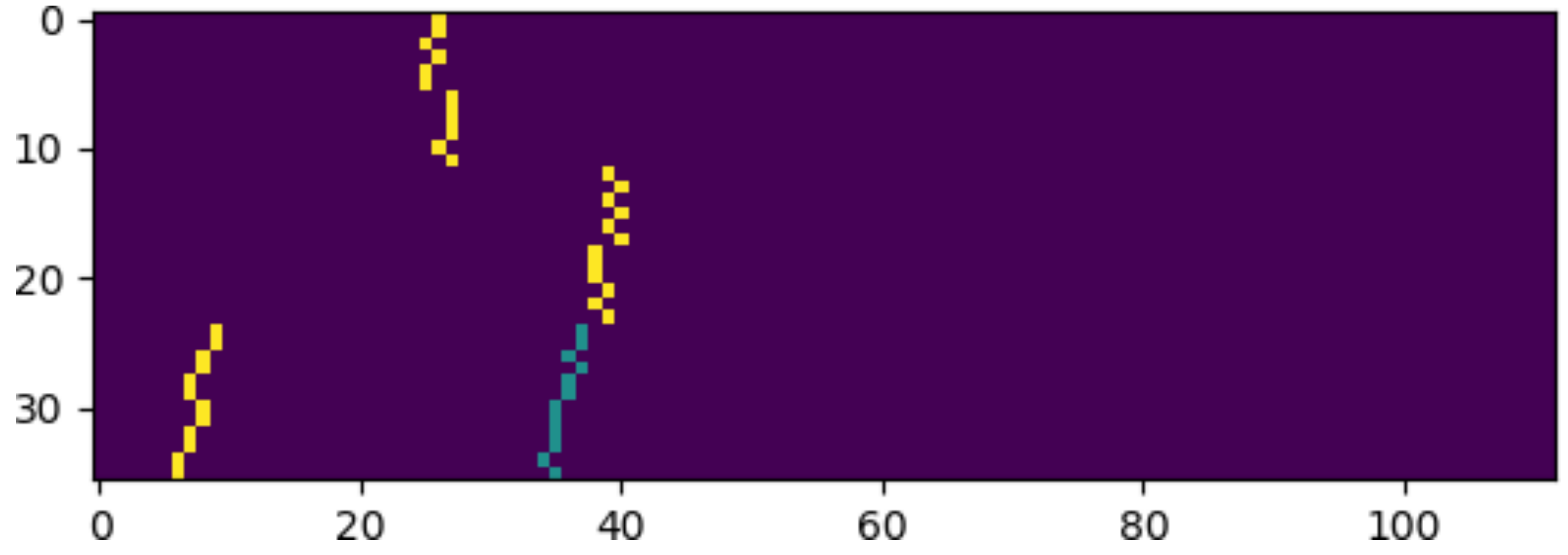
CLAS12 Tracking with Artificial Intelligence



Passing True Track Candidate to the LSTM



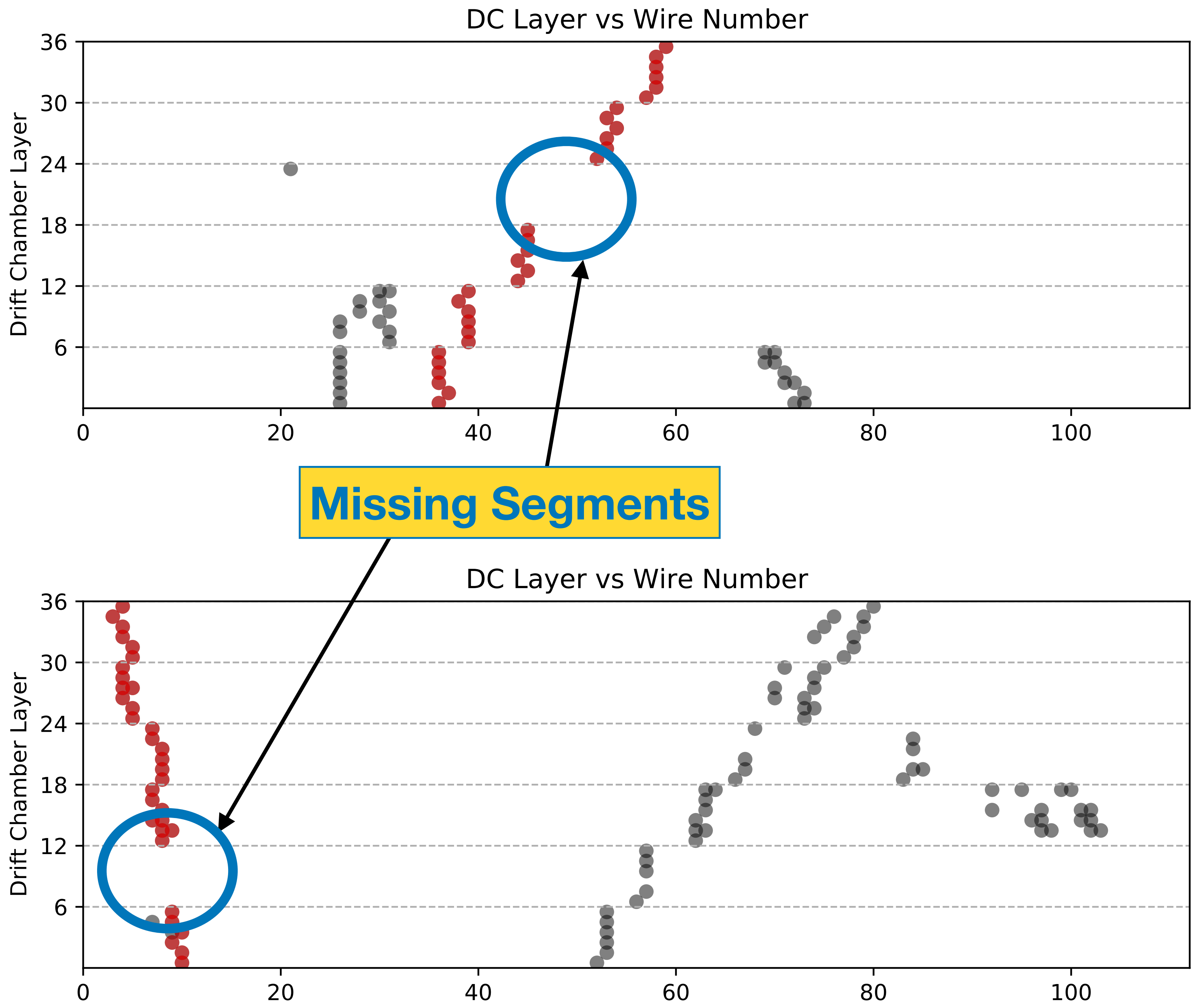
Passing False Track Candidate to the LSTM



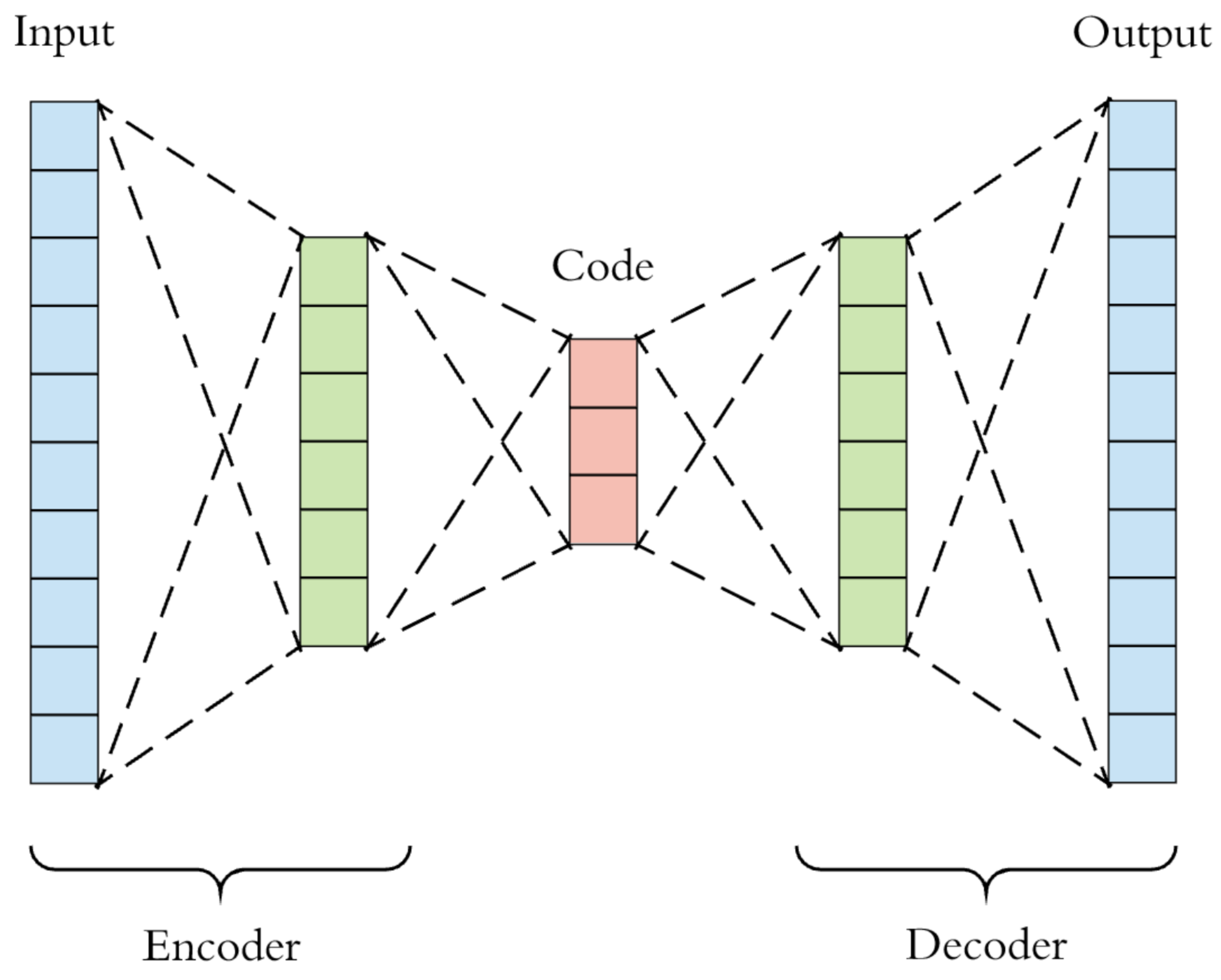
- **Track Trajectory Prediction:**
  - Region 2&3 (furthest from the beam) have less noise and clustering efficiency is high
  - Region 1 (closest to the beam line) has high background and hits can not be clustered efficiently in high luminosity runs, causing for tracking efficiency to drop.
  - If we can predict the position of hits in region 1 based on region 2&3 information we can use this to increase tracking efficiency
  - Combined with the previous project (track candidate classification) will improve reconstruction speed and clustering and tracking efficiency.
- **Status of The Project:**
  - Initial LSTM network was constructed to test on sample data.
  - Test show that the algorithm provides very high efficiency of finding the track trajectory with average mean deviation of 1.18 wires.
- **To Do (need ODU/CRTC student):**
  - Refine data sample used for training, includes eliminating tracks with bad Chi2, and include only tracks that come from target.
  - Develop the full workflow to extract the training sample from reconstructed data and process the training of Neural Network
  - Implement trajectory predictor in the software (Java based, initial test were done in Python) and integrate it with reconstruction software.
  - Modify the DC code to recounted region 1 hits, based on trajectory predictions before passing clusters to AI predictor (from previous project, fully integrated) for track candidate classification.

Model Type	Loss (MAE)	Time to Train	Time to Predict / sample
RNN/GRU	~1.18	374 *sec*	688 μs

AI Tracking
CLAS12 Tracking with Artificial Intelligence



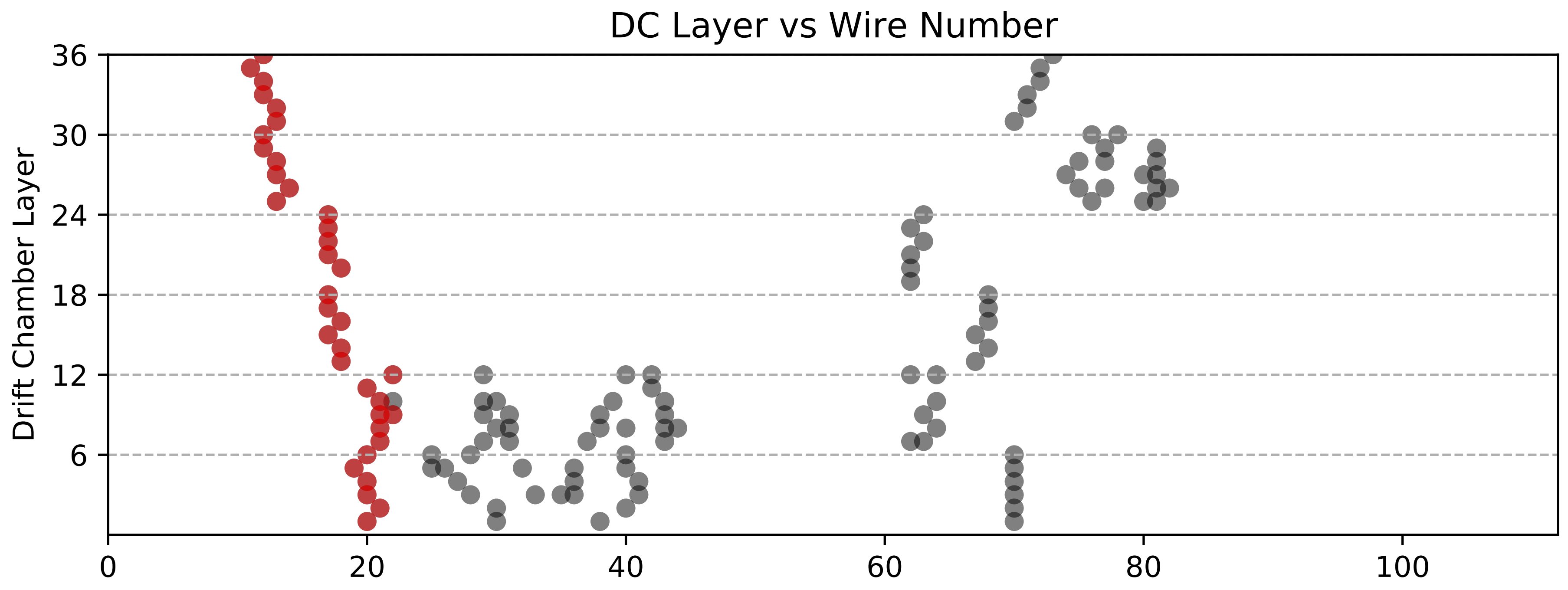
- **Missing Segments:**
  - It is easy to reconstruct the chain of segments with LSTM given first 4 segments and inferring 2 last segments
  - This approach can be used to identify segments in noisy super layer closest to the beam
  - However, using LSTM for identifying missing random segments is not possible.
  - Another approach was taken : use AutoEncoders



- **Autoencoders:**
  - An **autoencoder** is a type of artificial neural network used to learn efficient data codings in an unsupervised manner. The aim of an **autoencoder** is to learn a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore signal “noise”
  - The input and output of encoder is vector of the same size. And it learns the output vector even if there is a corruption in the input.

AI Tracking

CLAS12 Tracking with Artificial Intelligence



$$x_1 = \sum_{i=1..6} \frac{w_i}{6} \qquad x_4 = \sum_{i=19..24} \frac{w_i}{6}$$

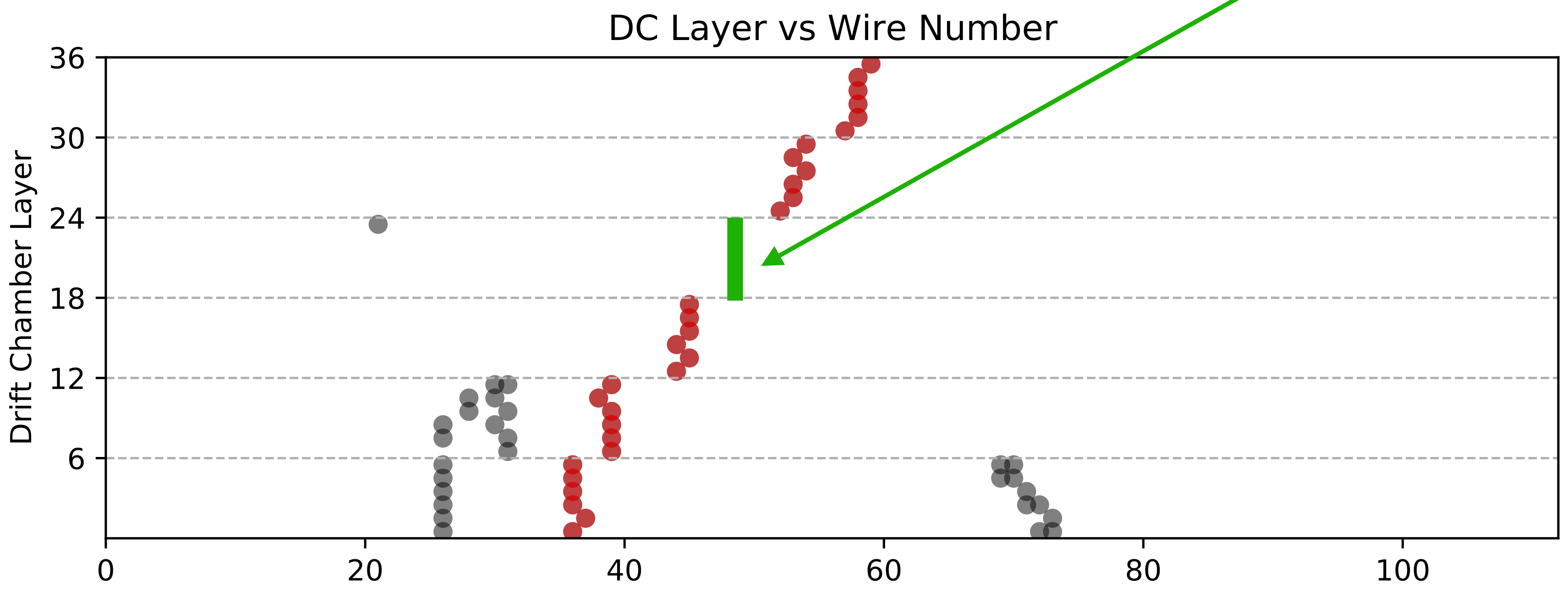
$$x_2 = \sum_{i=7..12} \frac{w_i}{6} \qquad x_5 = \sum_{i=25..32} \frac{w_i}{6}$$

$$x_3 = \sum_{i=13..18} \frac{w_i}{6} \qquad x_6 = \sum_{i=33..36} \frac{w_i}{6}$$

$$X(x_1, x_2, x_3, x_4, x_5, x_6) => Y(x_1, x_2, x_3, x_4, x_5, x_6)$$

corrupted vector should also be reconstructed

$$X(x_1, x_2, x_3, 0, x_5, x_6) => Y(x_1, x_2, x_3, x_4, x_5, x_6)$$



AI Tracking

CLAS12 Tracking with Artificial Intelligence

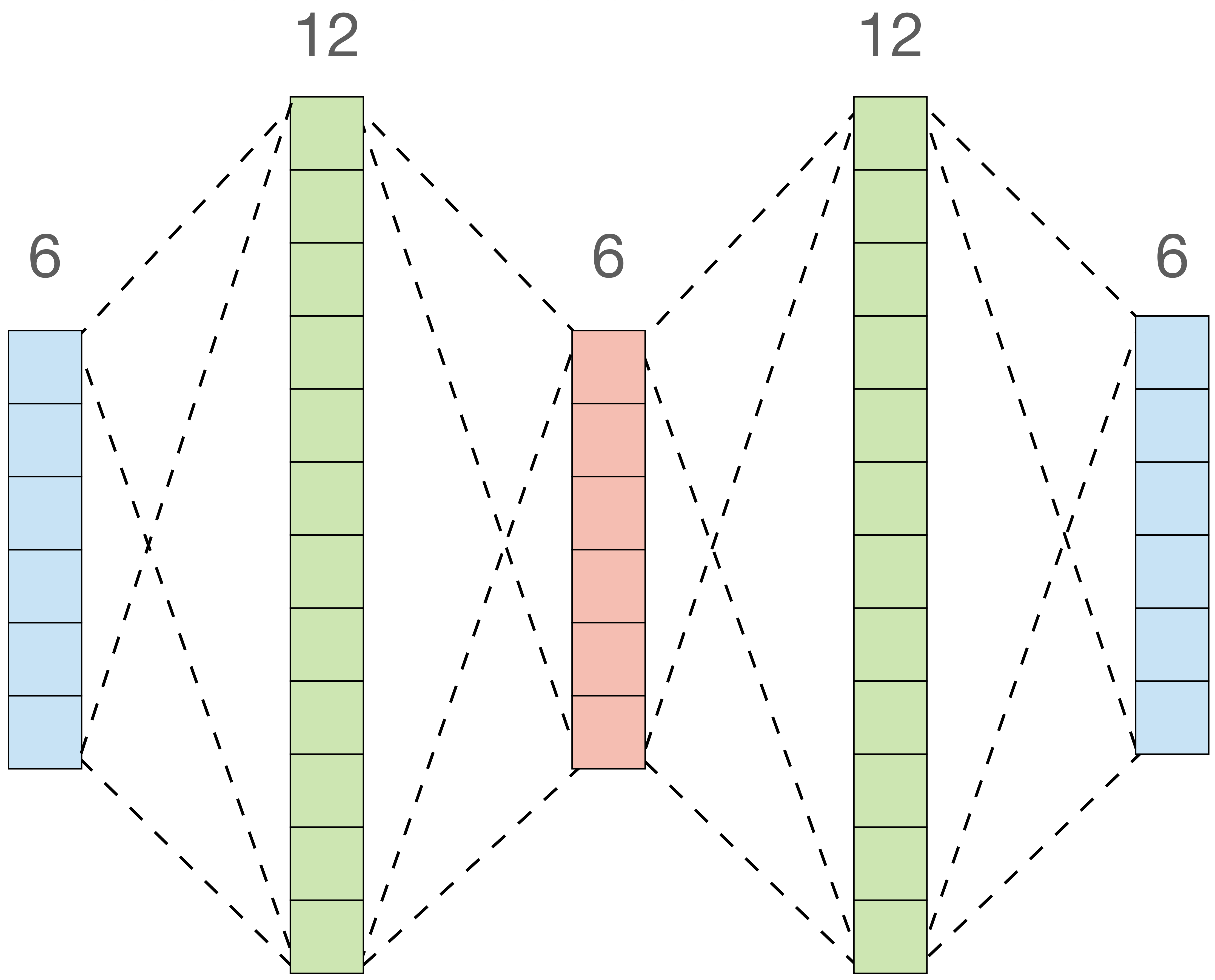
- **Random Corruption:**
  - Introduce a corruption in a random super layer
  - And feed the network with corrupted data as input and the real data in the output

$$(x_1, x_2, x_3, x_4, x_5, x_6) \begin{cases} i = rndm(1..6) & x_i = 0.0 \\ X(x_1, 0.0, x_3, x_4, x_5, x_6) \rightarrow & Y(x_1, x_2, x_3, x_4, x_5, x_6) \end{cases} \quad (6)$$

- **Complete Set**
  - Introduce a corruption in every super layer
  - feed the network 6 samples from each event, corrupted ones as input and real data as output

$$(x_1, x_2, x_3, x_4, x_5, x_6) \begin{cases} X(0.0, x_2, x_3, x_4, x_5, x_6) \rightarrow & Y(x_1, x_2, x_3, x_4, x_5, x_6) \\ X(x_1, 0.0, x_3, x_4, x_5, x_6) \rightarrow & Y(x_1, x_2, x_3, x_4, x_5, x_6) \\ X(x_1, x_2, 0.0, x_4, x_5, x_6) \rightarrow & Y(x_1, x_2, x_3, x_4, x_5, x_6) \\ X(x_1, x_2, x_3, 0.0, x_5, x_6) \rightarrow & Y(x_1, x_2, x_3, x_4, x_5, x_6) \\ X(x_1, x_2, x_3, x_4, 0.0, x_6) \rightarrow & Y(x_1, x_2, x_3, x_4, x_5, x_6) \\ X(x_1, x_2, x_3, x_4, x_5, 0.0) \rightarrow & Y(x_1, x_2, x_3, x_4, x_5, x_6) \end{cases} \quad (7)$$

Auto-Encoder Architecture

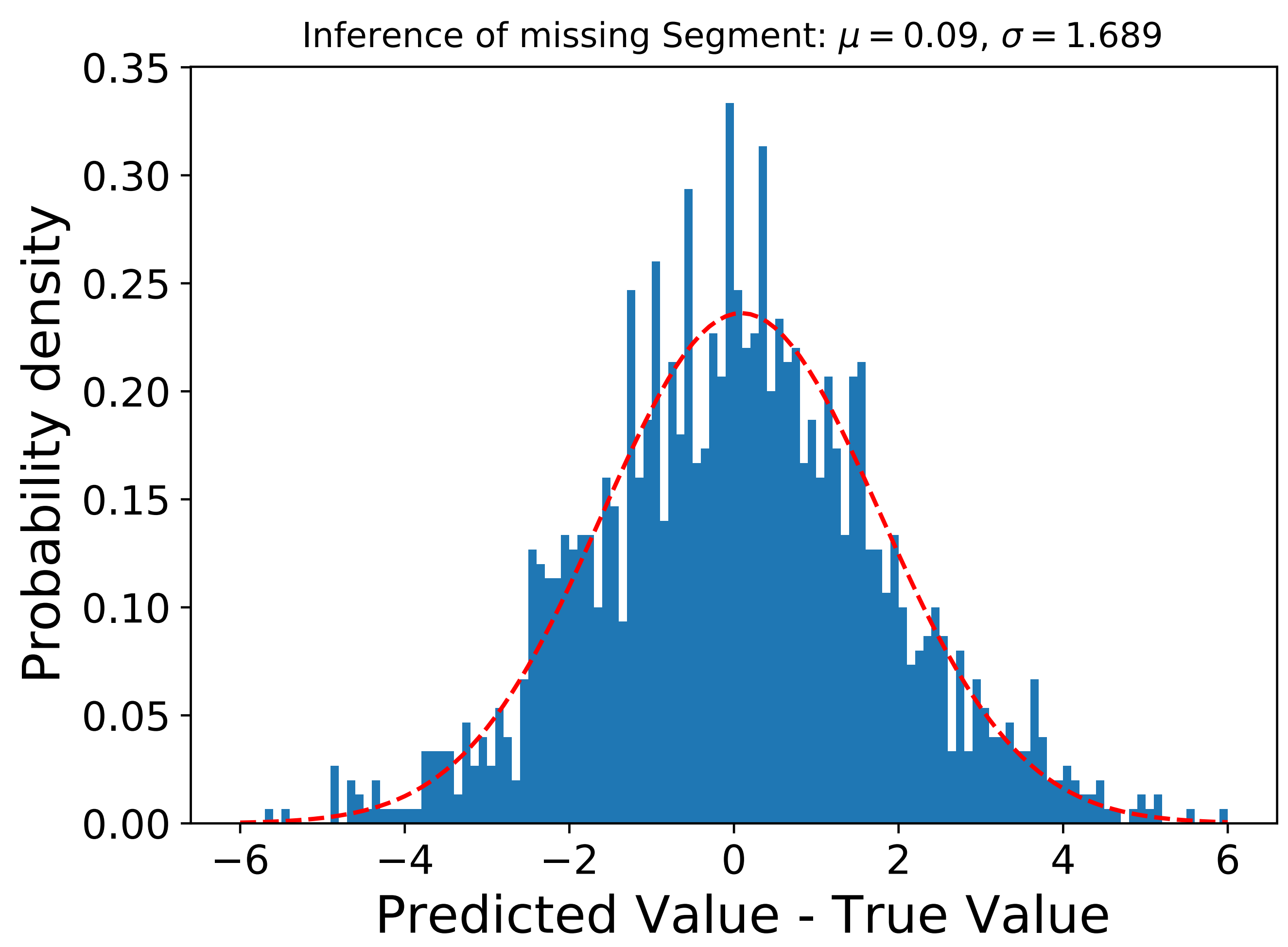


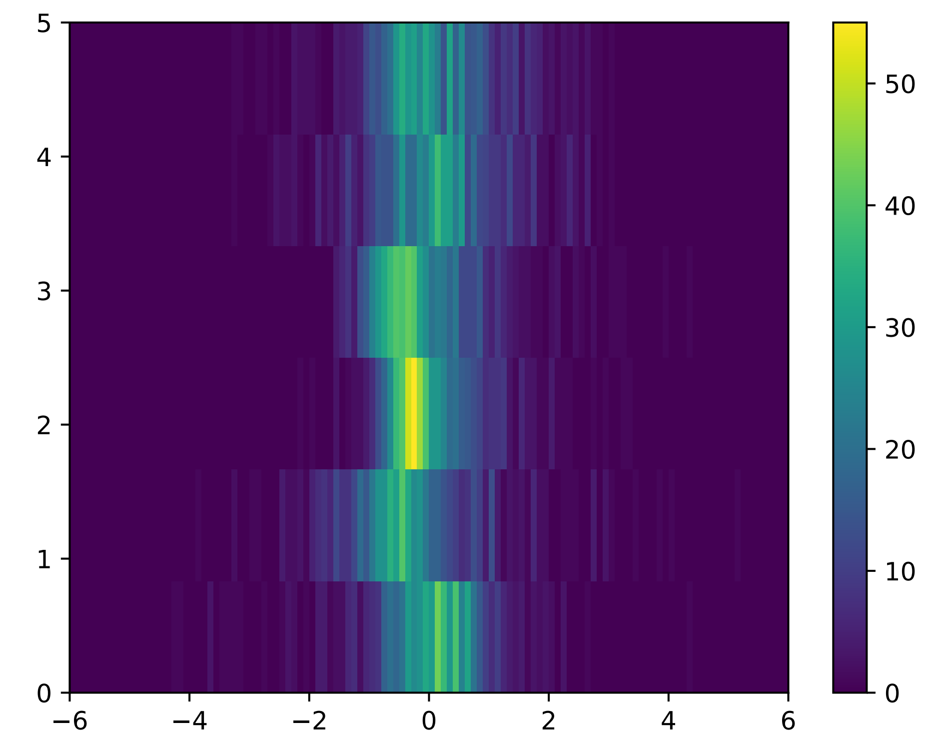
AI Tracking

CLAS12 Tracking with Artificial Intelligence

- **Random Corruption:**
  - Introduce a corruption in a random super layer
  - And feed the network with corrupted data as input and the real data in the output

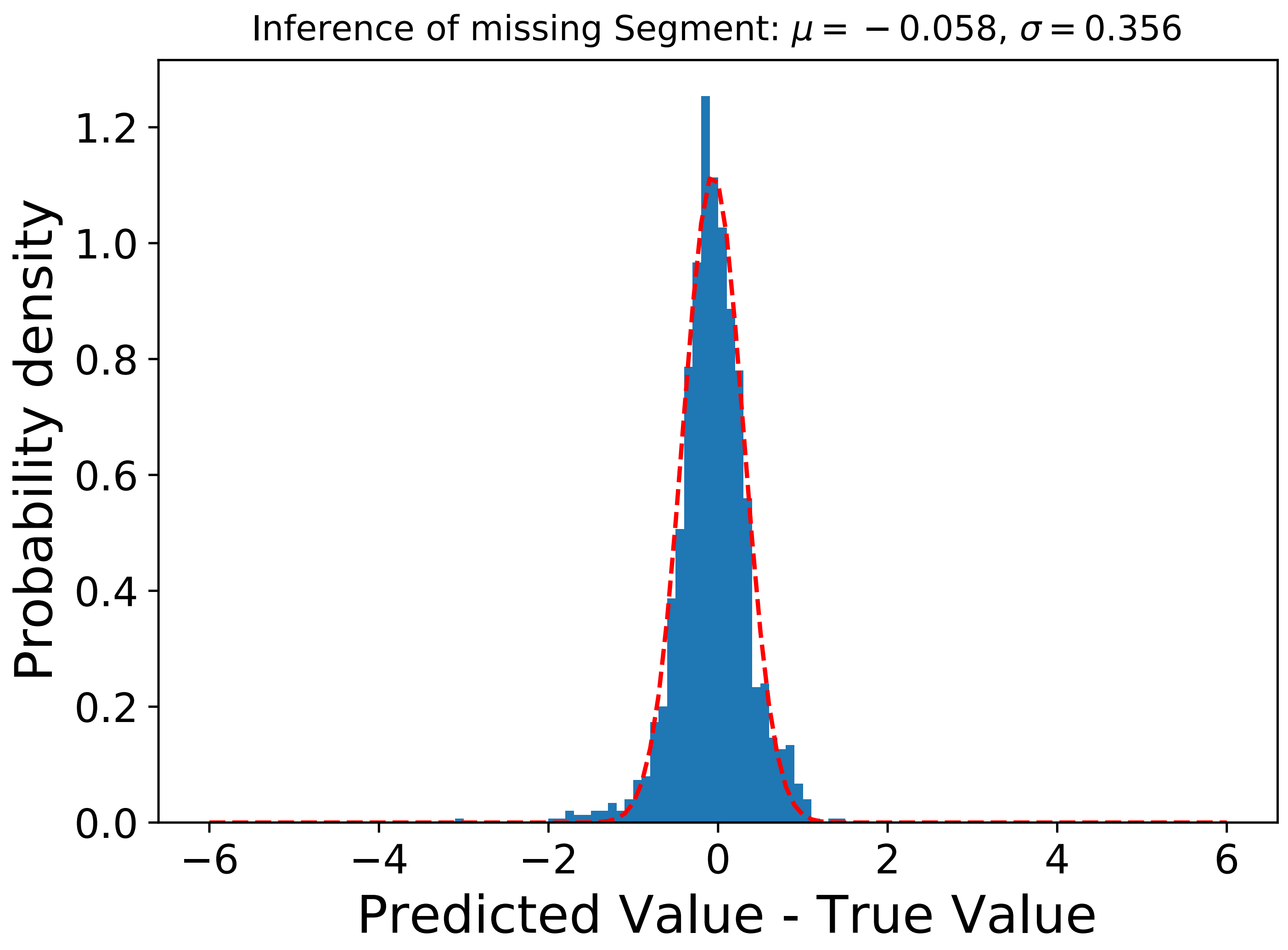
$$(x_1,x_2,x_3,x_4,x_5,x_6)\begin{cases} i = rndm(1..6) & x_i = 0.0 \\ X(x_1,0.0,x_3,x_4,x_5,x_6)\rightarrow & Y(x_1,x_2,x_3,x_4,x_5,x_6) \end{cases} \tag{6}$$

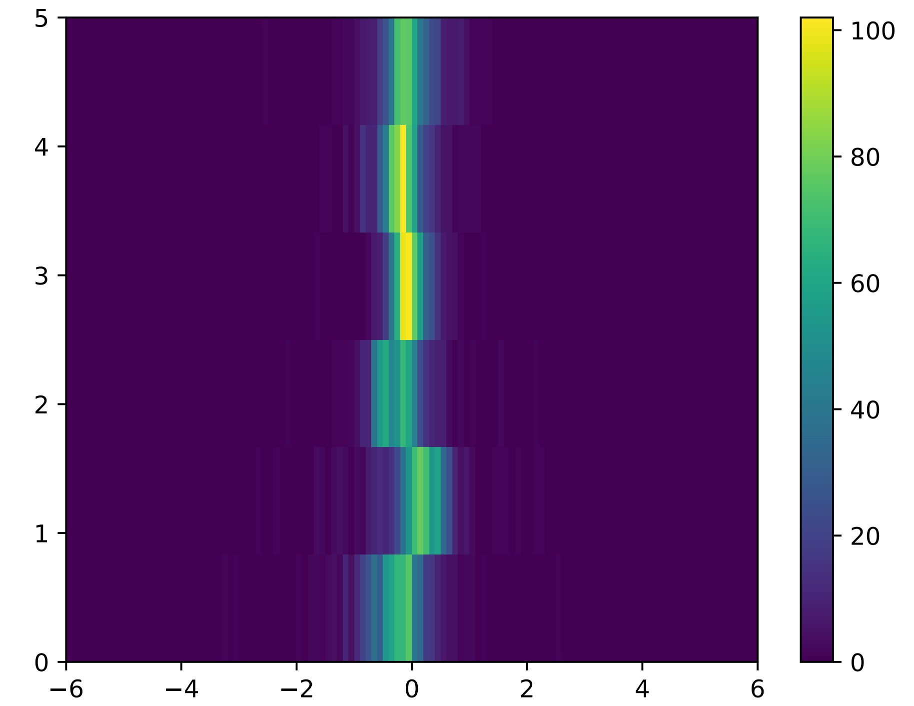




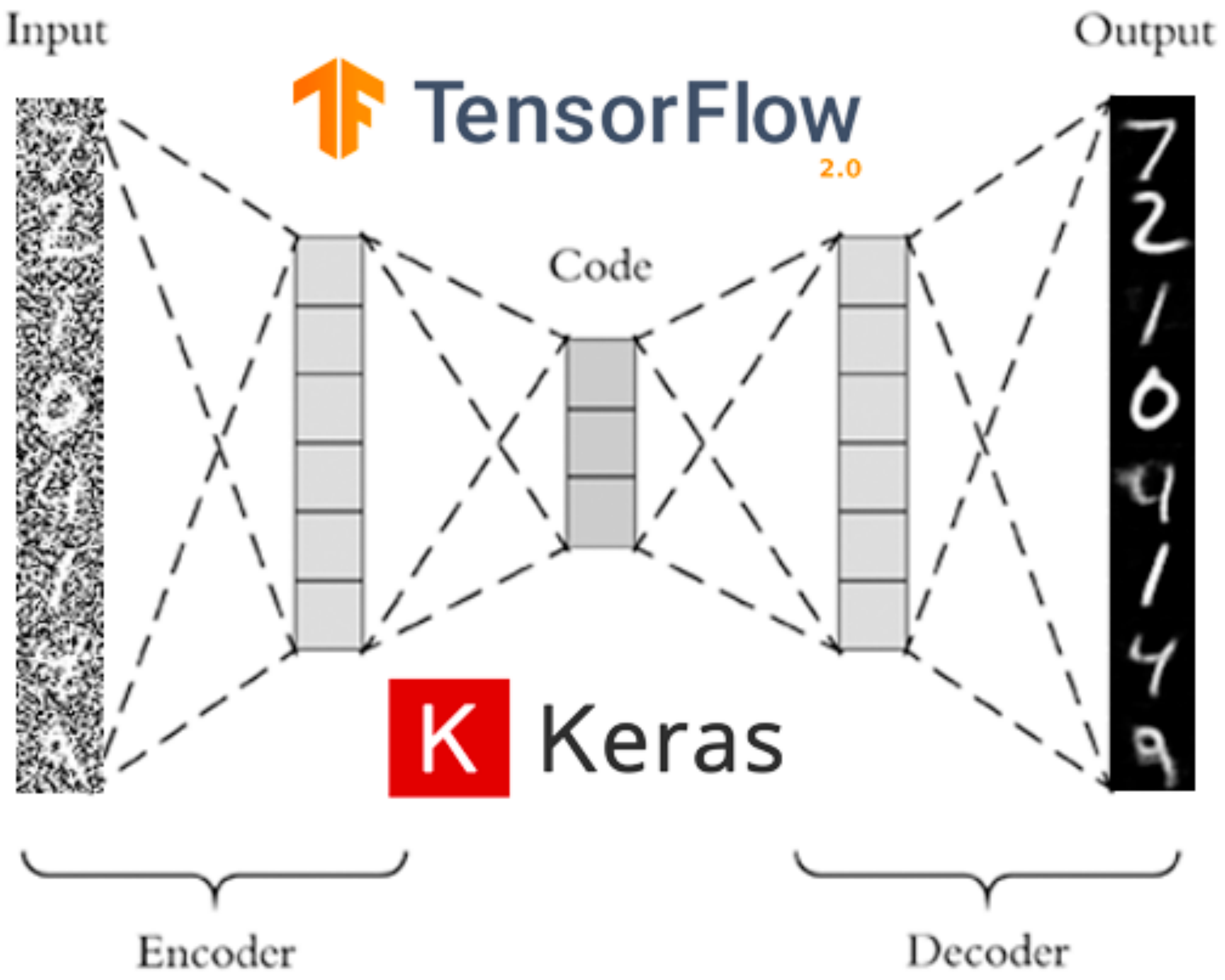
- **Complete Set**
  - Introduce a corruption in every super layer
  - feed the network 6 samples from each event, corrupted ones as input and real data as output

$$(x_1,x_2,x_3,x_4,x_5,x_6)\begin{cases} X(0.0,x_2,x_3,x_4,x_5,x_6)\rightarrow & Y(x_1,x_2,x_3,x_4,x_5,x_6) \\ X(x_1,0.0,x_3,x_4,x_5,x_6)\rightarrow & Y(x_1,x_2,x_3,x_4,x_5,x_6) \\ X(x_1,x_2,0.0,x_4,x_5,x_6)\rightarrow & Y(x_1,x_2,x_3,x_4,x_5,x_6) \\ X(x_1,x_2,x_3,0.0,x_5,x_6)\rightarrow & Y(x_1,x_2,x_3,x_4,x_5,x_6) \\ X(x_1,x_2,x_3,x_4,0.0,x_6)\rightarrow & Y(x_1,x_2,x_3,x_4,x_5,x_6) \\ X(x_1,x_2,x_3,x_4,x_5,0.0)\rightarrow & Y(x_1,x_2,x_3,x_4,x_5,x_6) \end{cases} \tag{7}$$

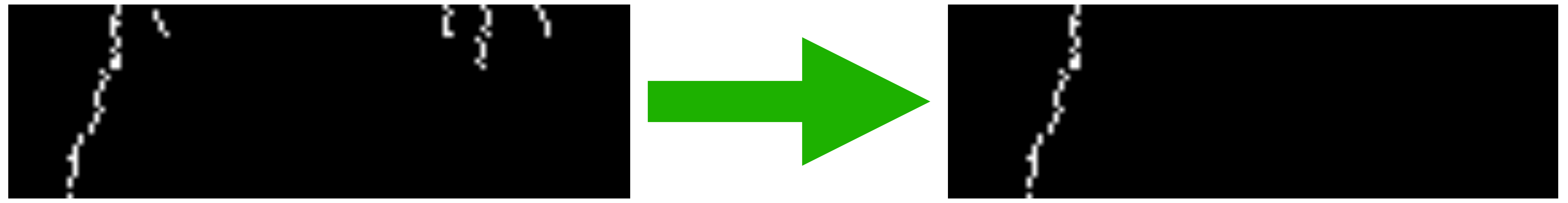




AI Tracking
Denoising AutoEncoders

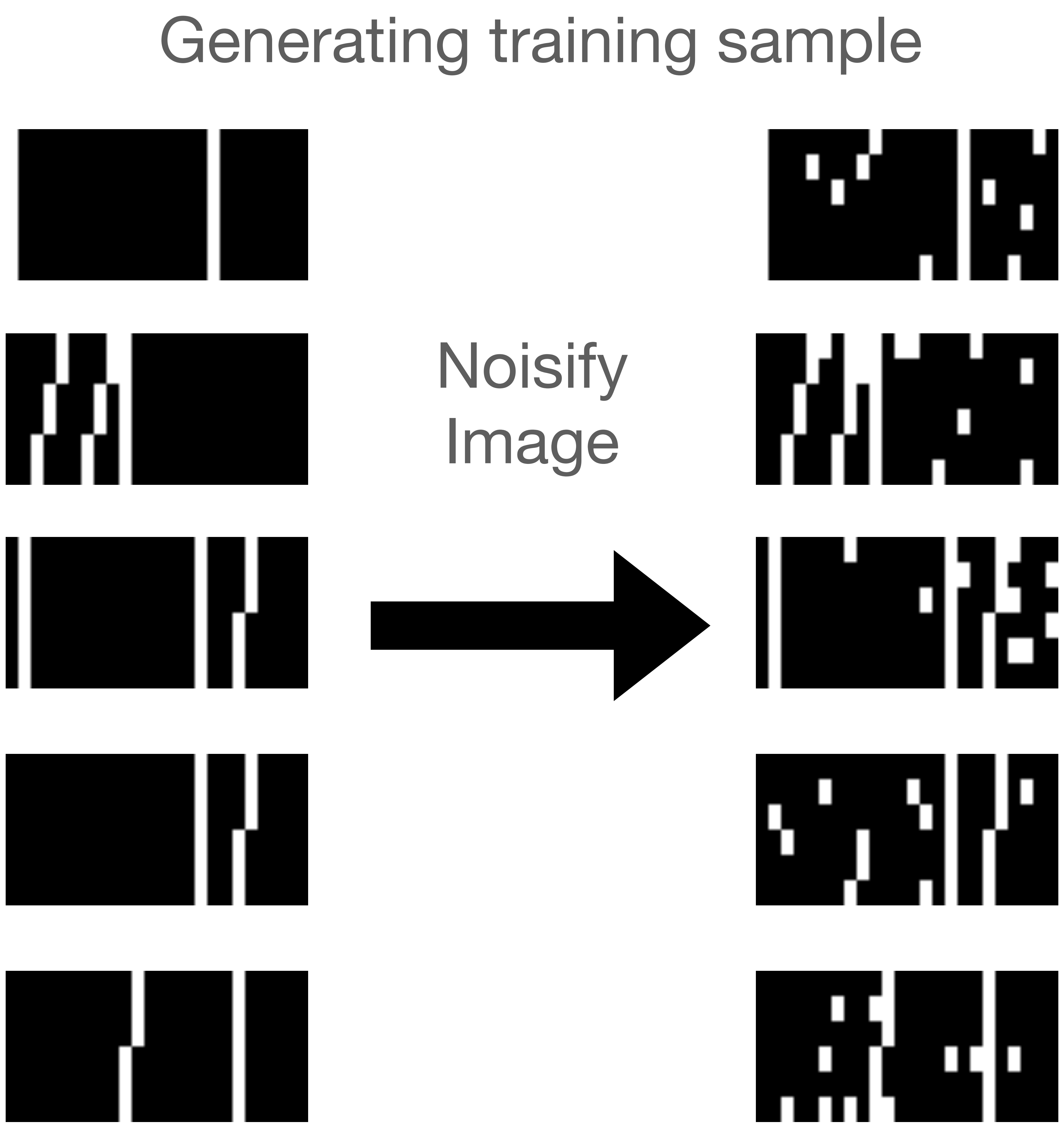


- **Auto-Encoder Uses**
  - Another use of auto encoders is de-noising of the image
  - Unlike corruption correction used in the previous section where information was filled in based on surrounding data de-noising is removing irrelevant information from the image.
  - Most advertised case study is de-noising of MNIST (hand Written digits Data set) data set.
  - This is very similar to Drift chamber data where we have segment related to the track and segment that do not belong to the track.

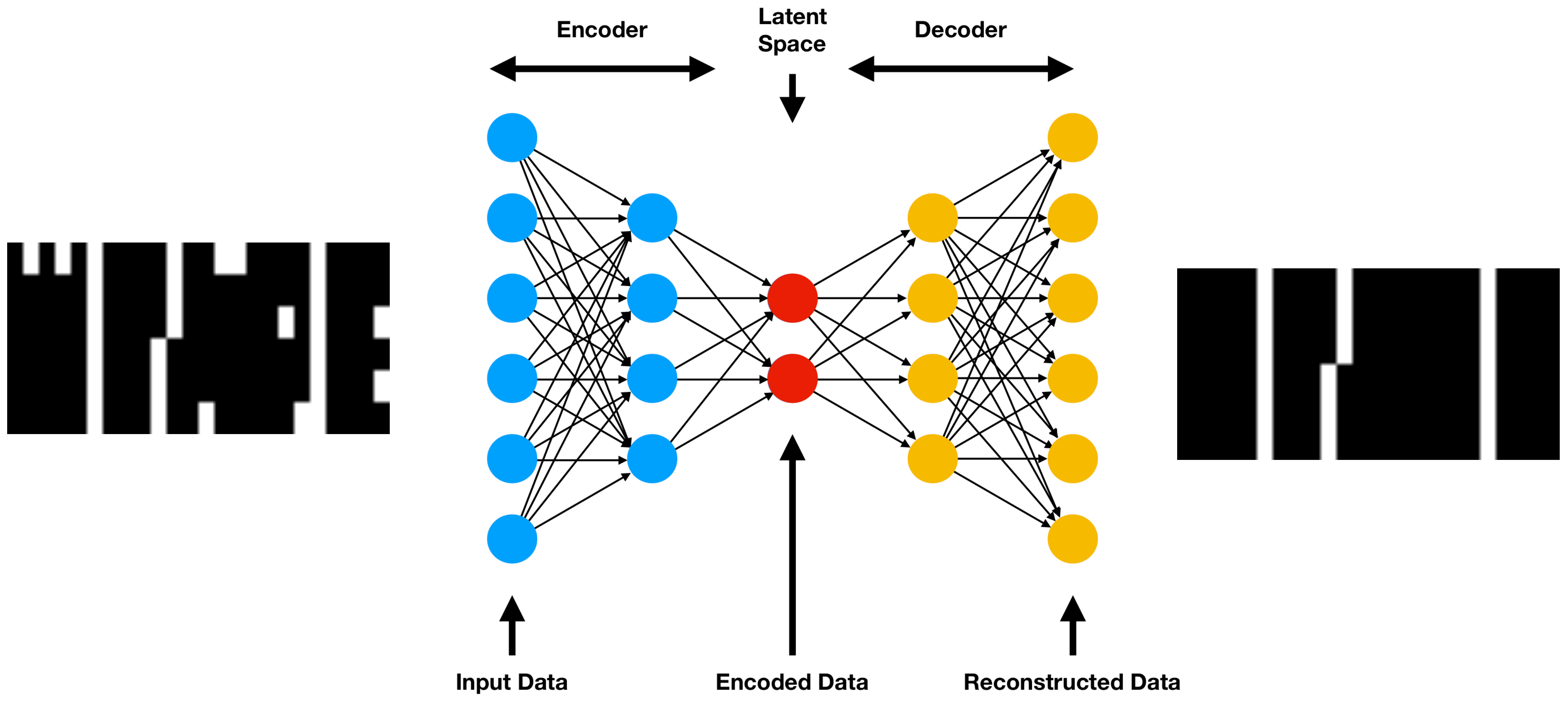


AI Tracking

De-noising

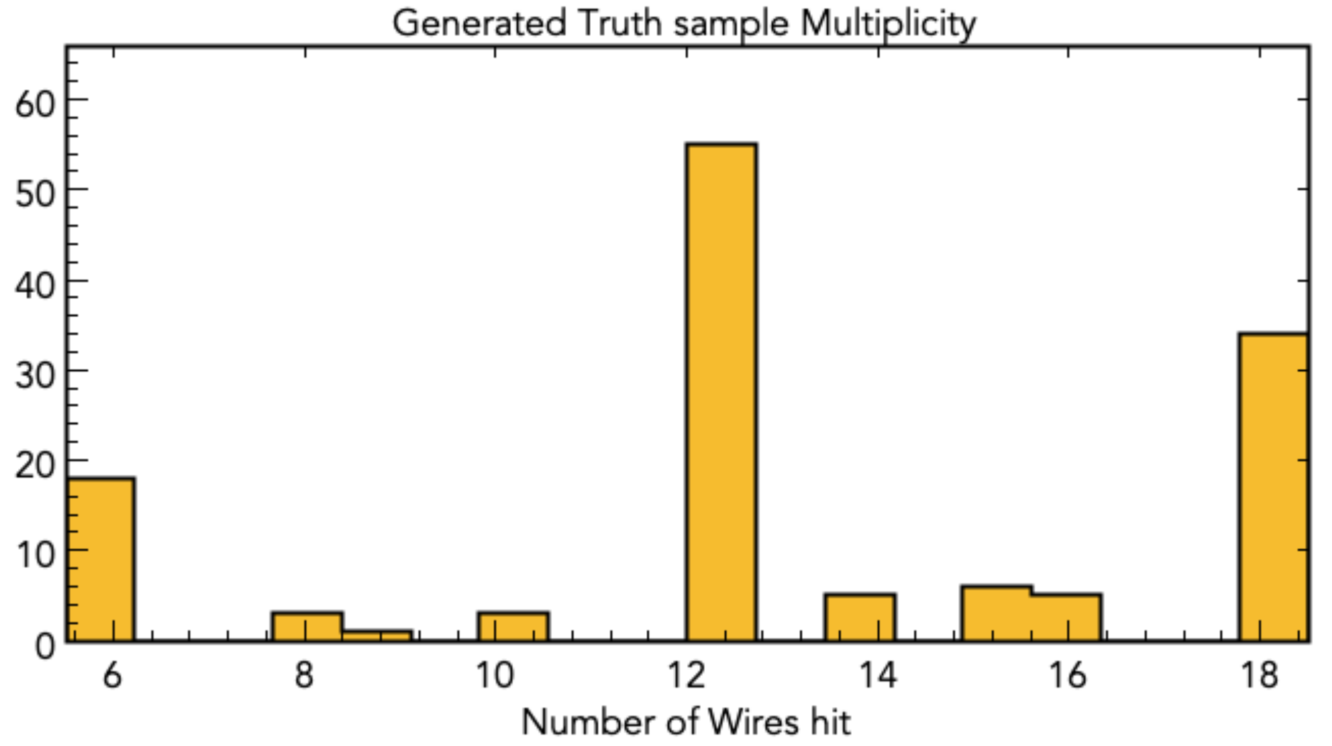


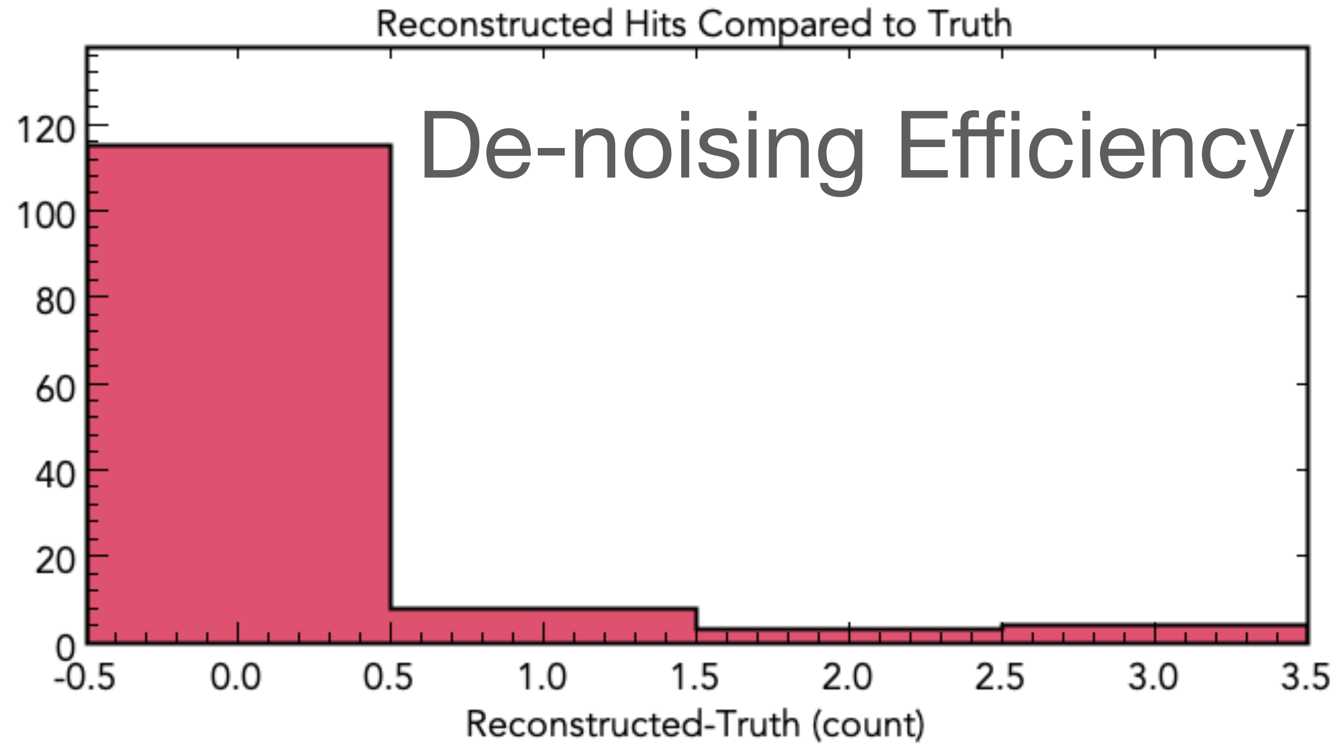
- **De-noising Auto-Encoder test**
  - Simple case was studied with 24 wires and 6 layers
  - Track segments were generated in the chamber
  - random number of segments 1,2 or 3 segments per event
  - Noise was added to the image
  - Notified image was used as input to the neural network encoder and real image as output of decoder
  - This simplistic test was done to assess the capability of the auto-encoder network to de-noise images.

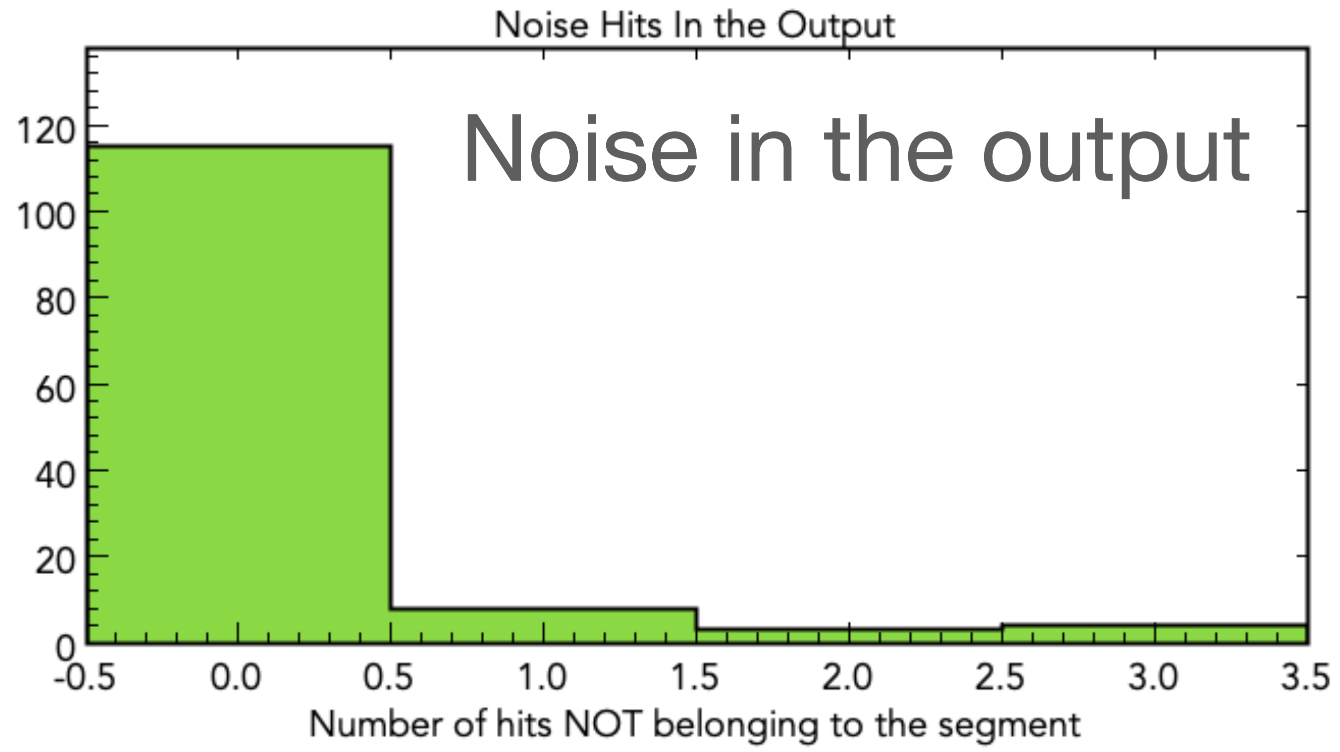


AI Tracking
De-noising

- **De-noising Auto-Encoder test**
  - Randomly generated sample was used to test the trained network performance
  - Similar to training sample number of segments generated were 1,2 or 3.
  - Overall network performance was quantified by measuring how many hits from the truth were reconstructed and how many noisy hits made it through

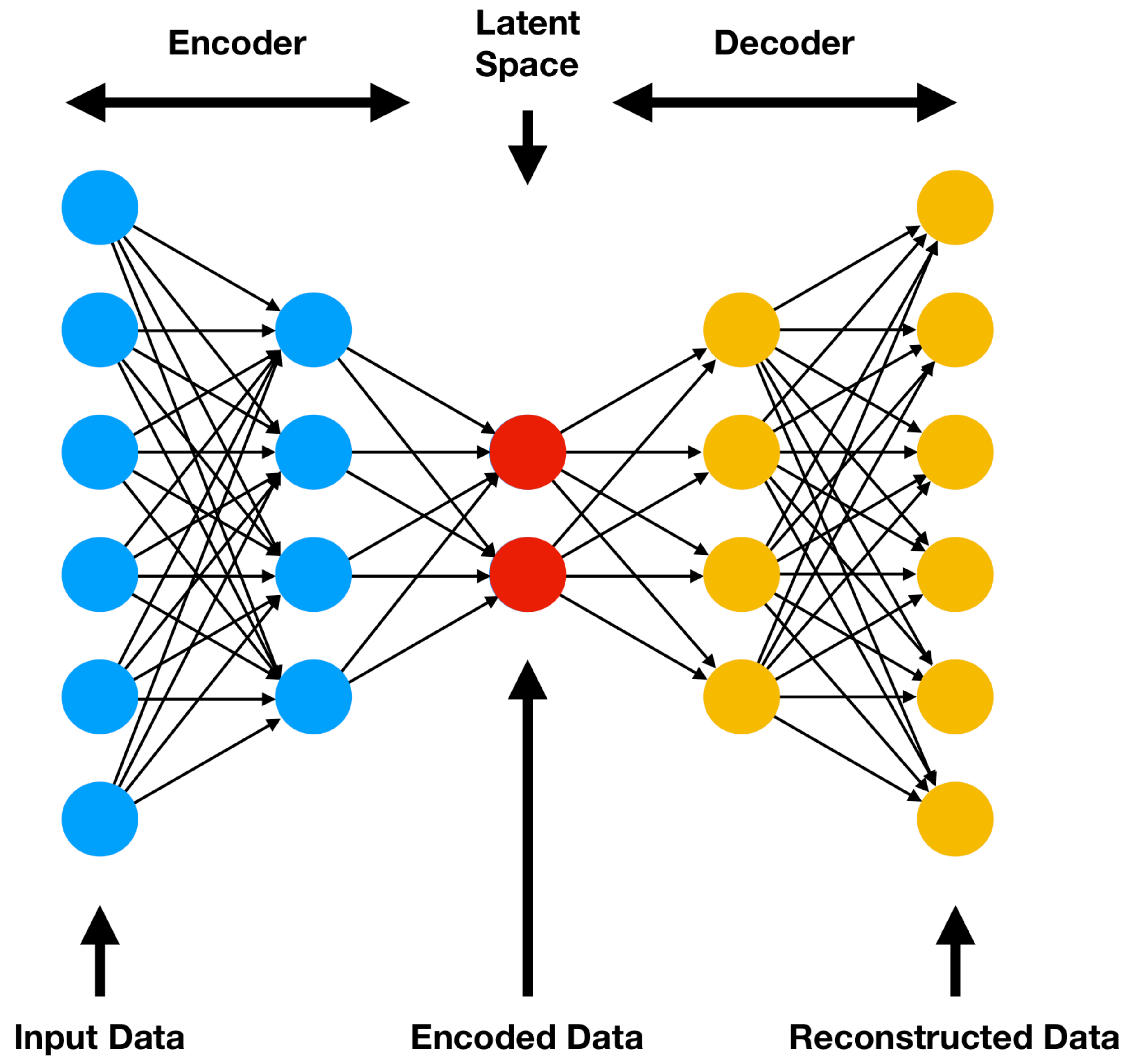




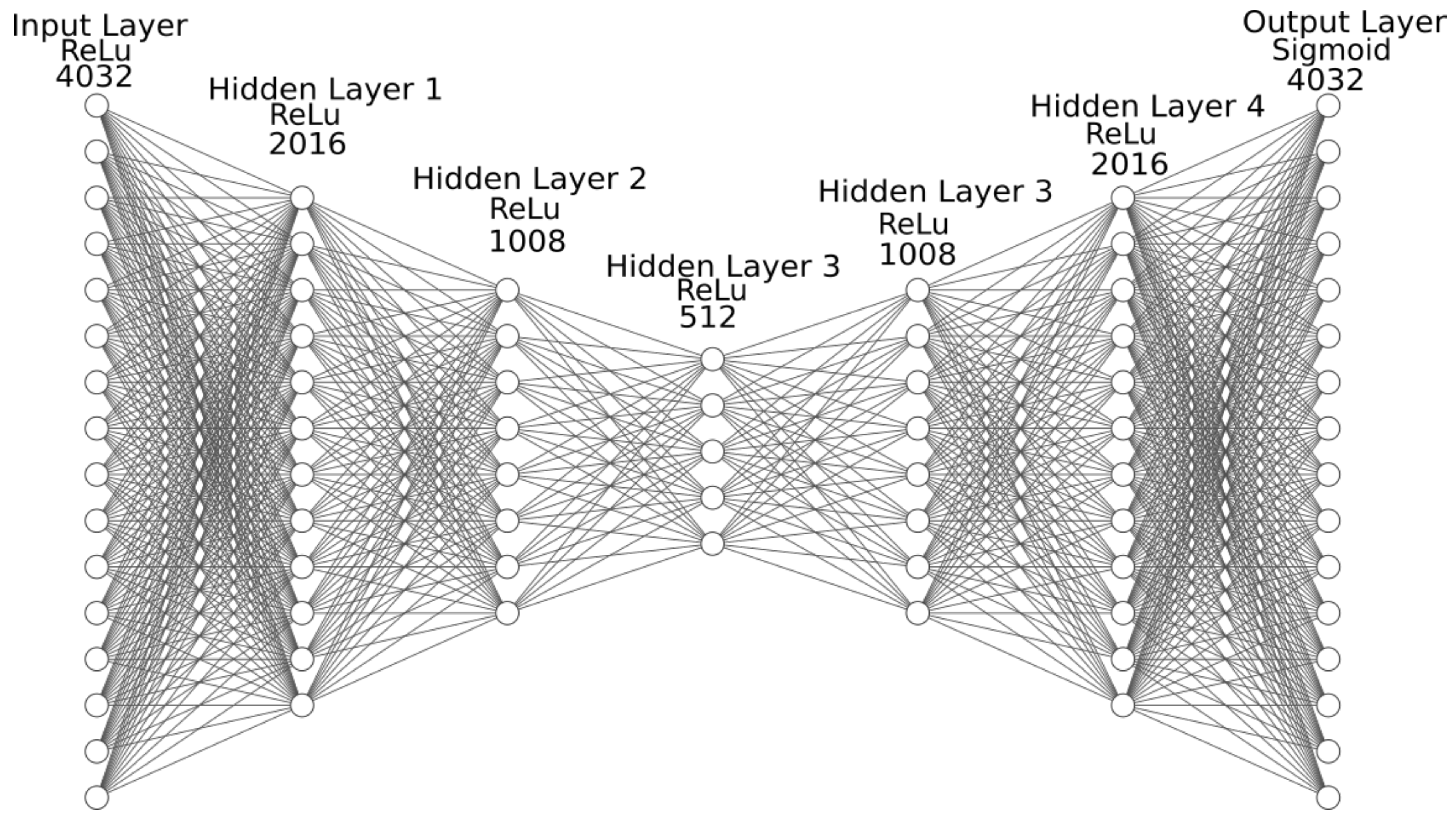


AI Tracking
De-noising

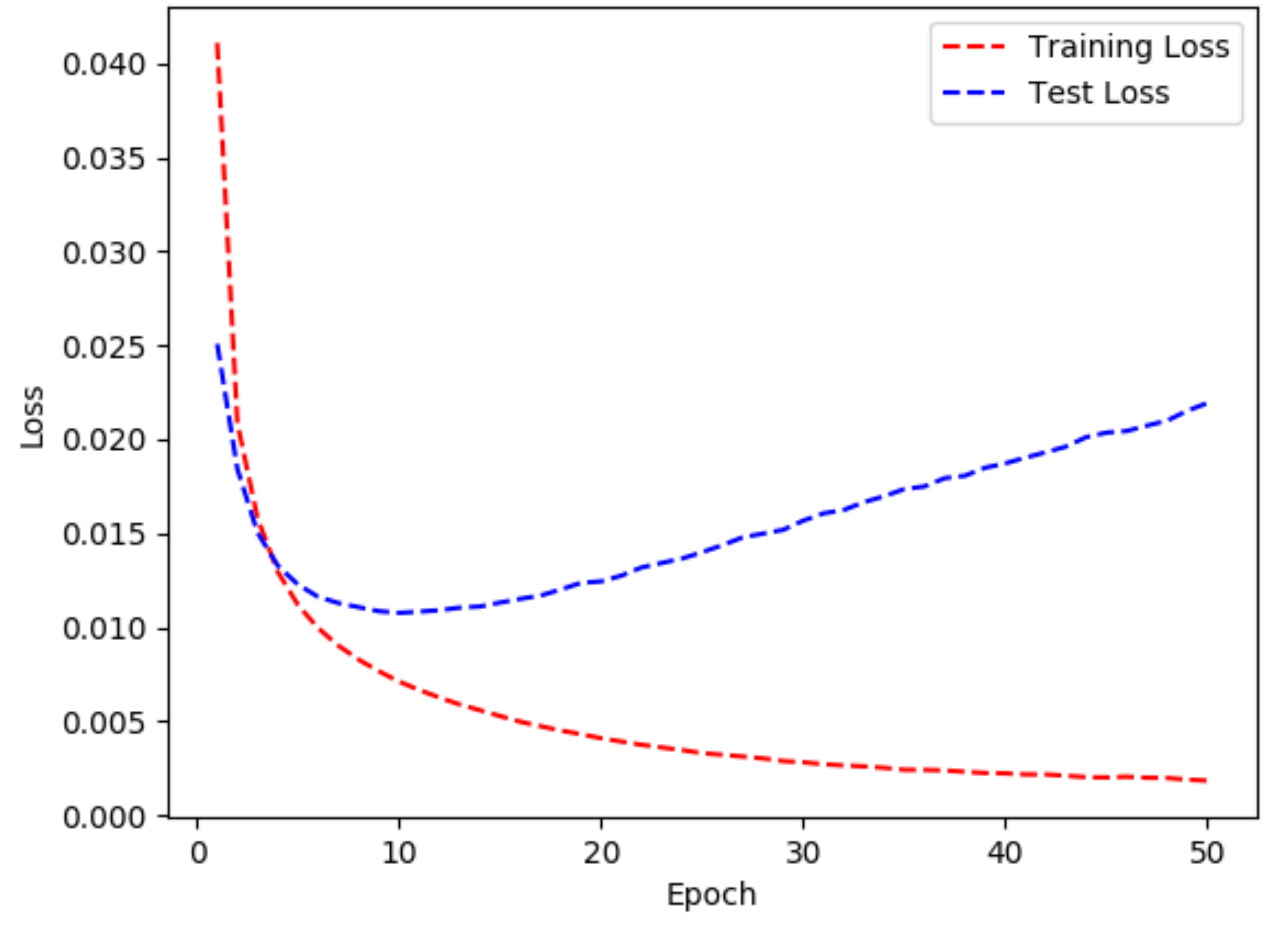
Now we want to try this with real data.
Only single track events were selected to gradually study the network performance

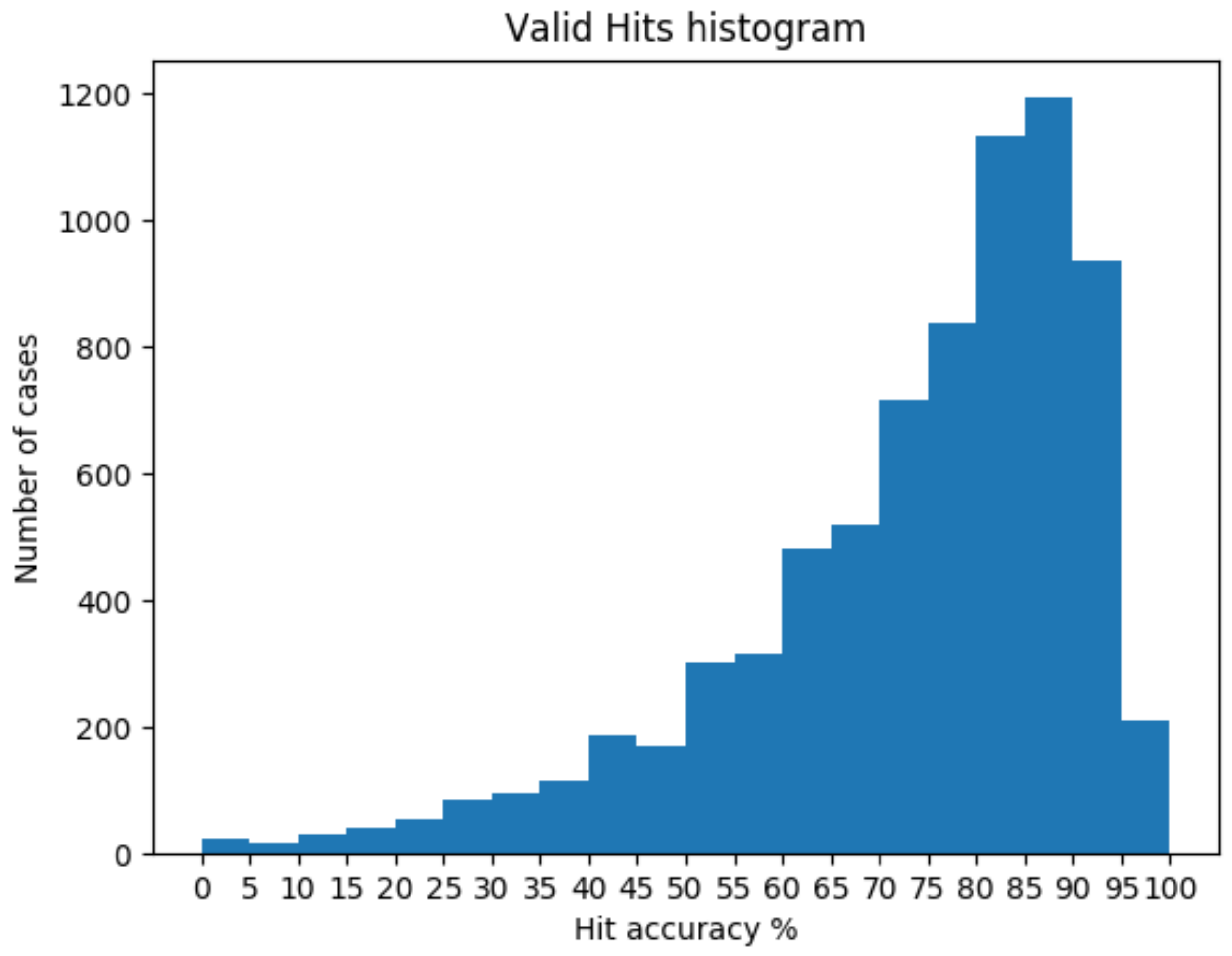


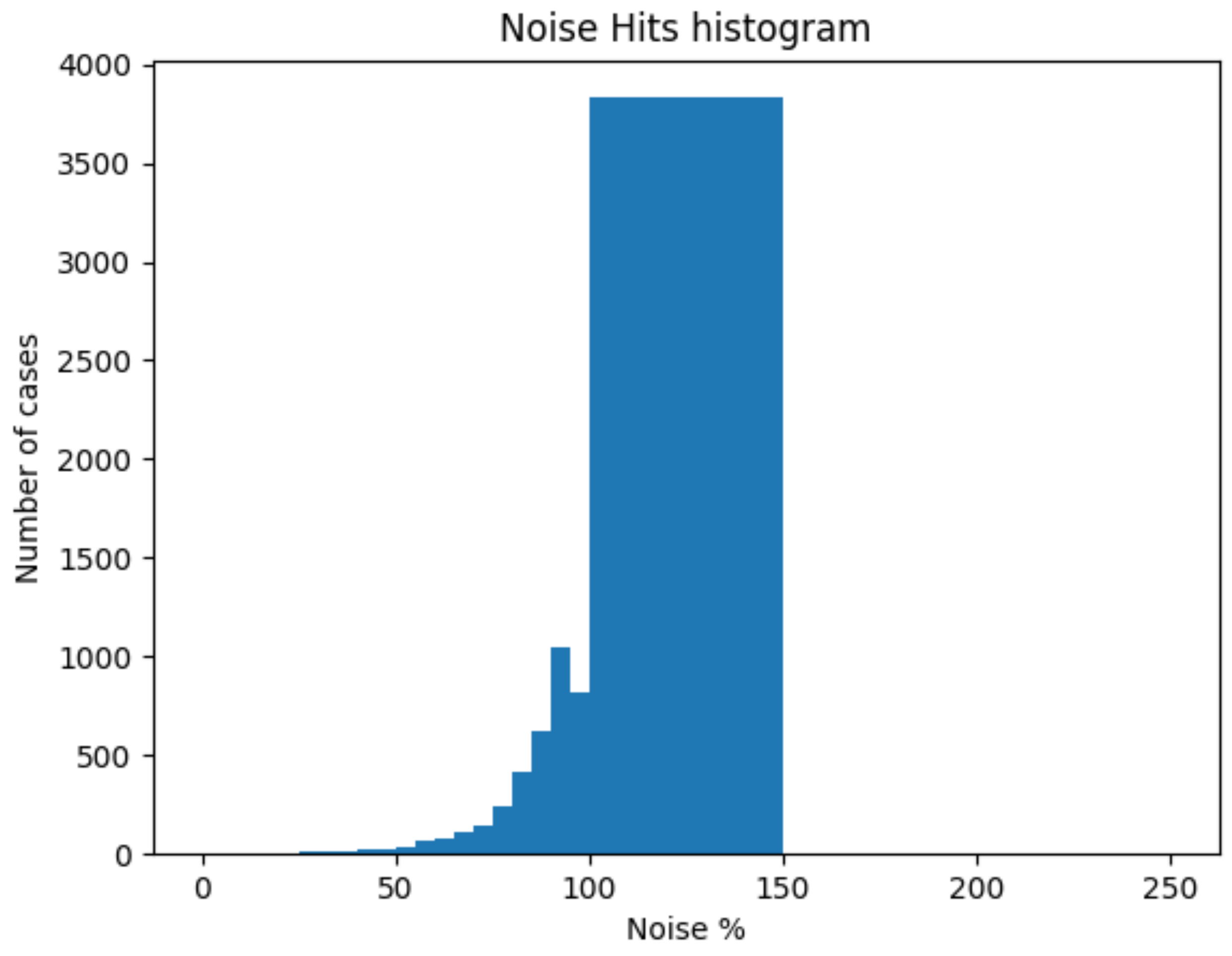
AI Tracking
De-noising



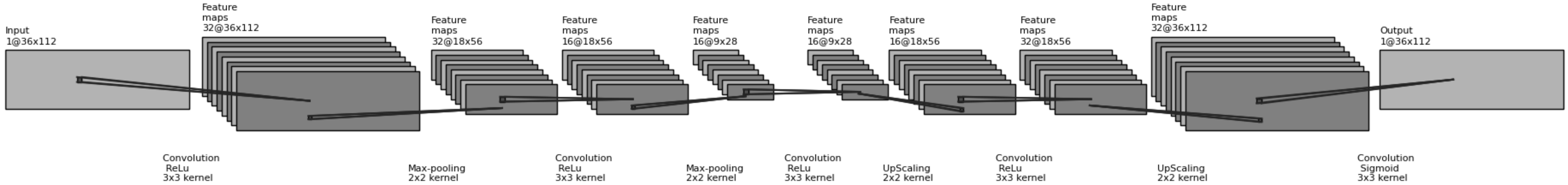
- **Multi-Layer Perceptron (MLP)**
  - MLP network was developed to de-noise DC
  - Training does not converge (it memorizes training sample)
  - Unable to work well on testing sample.



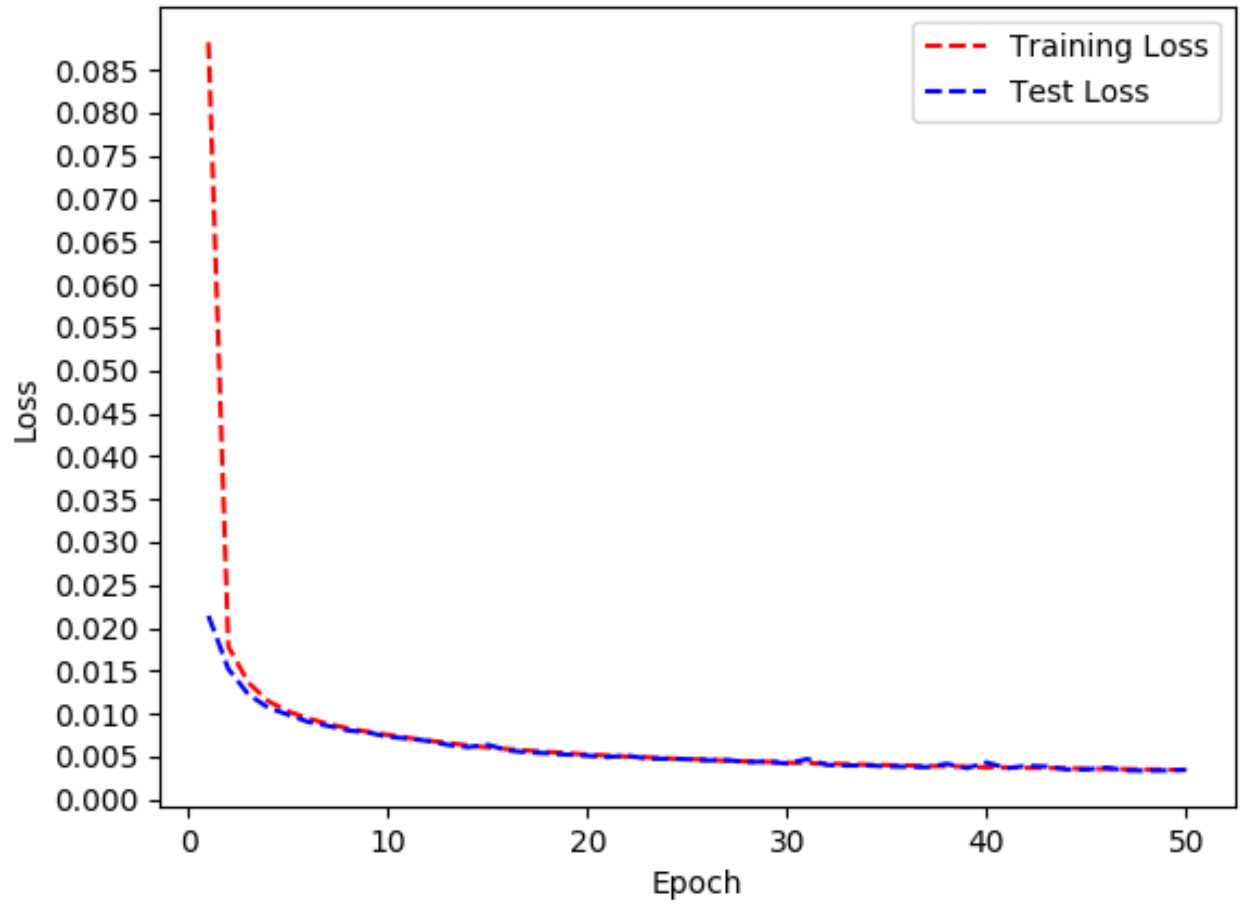


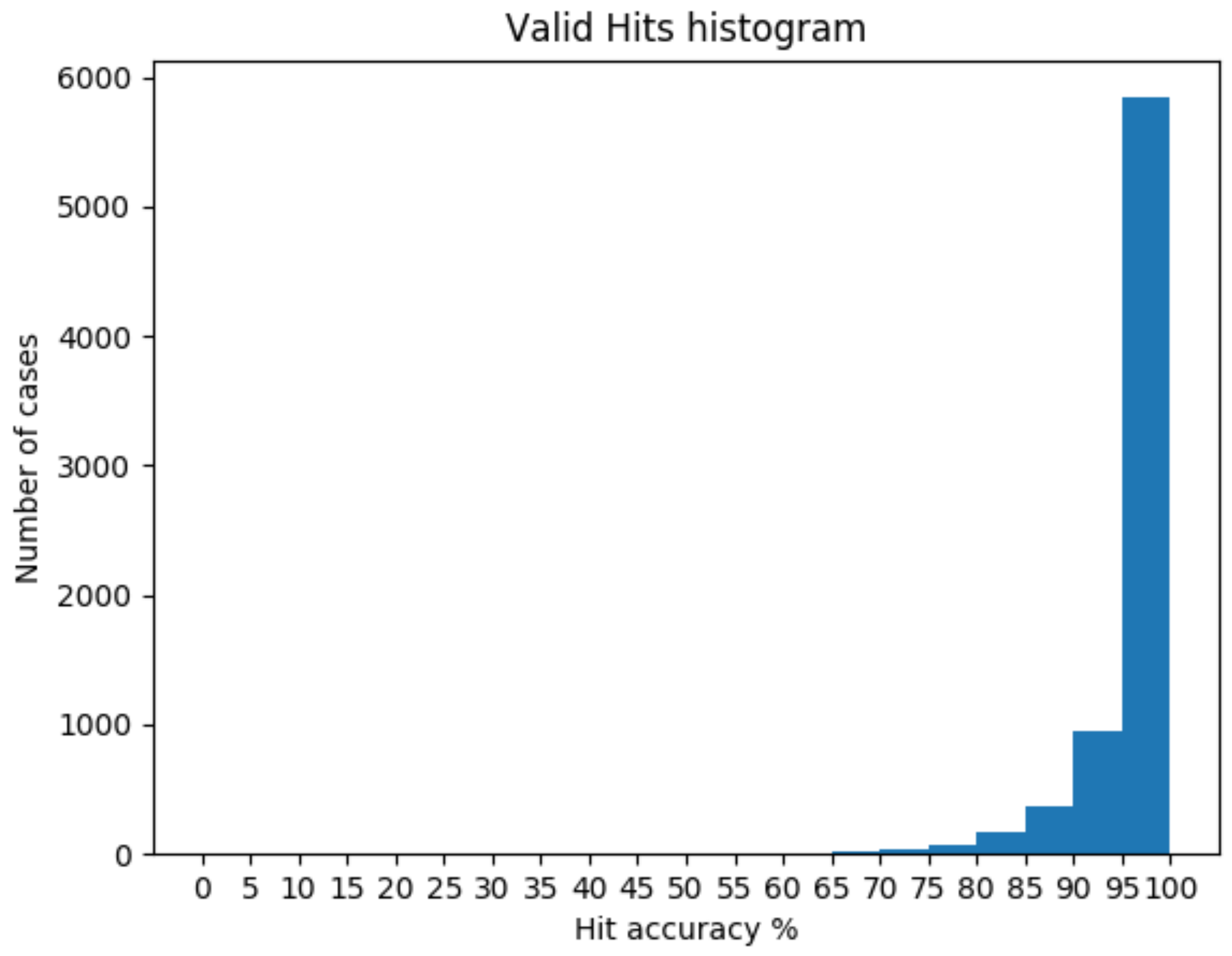


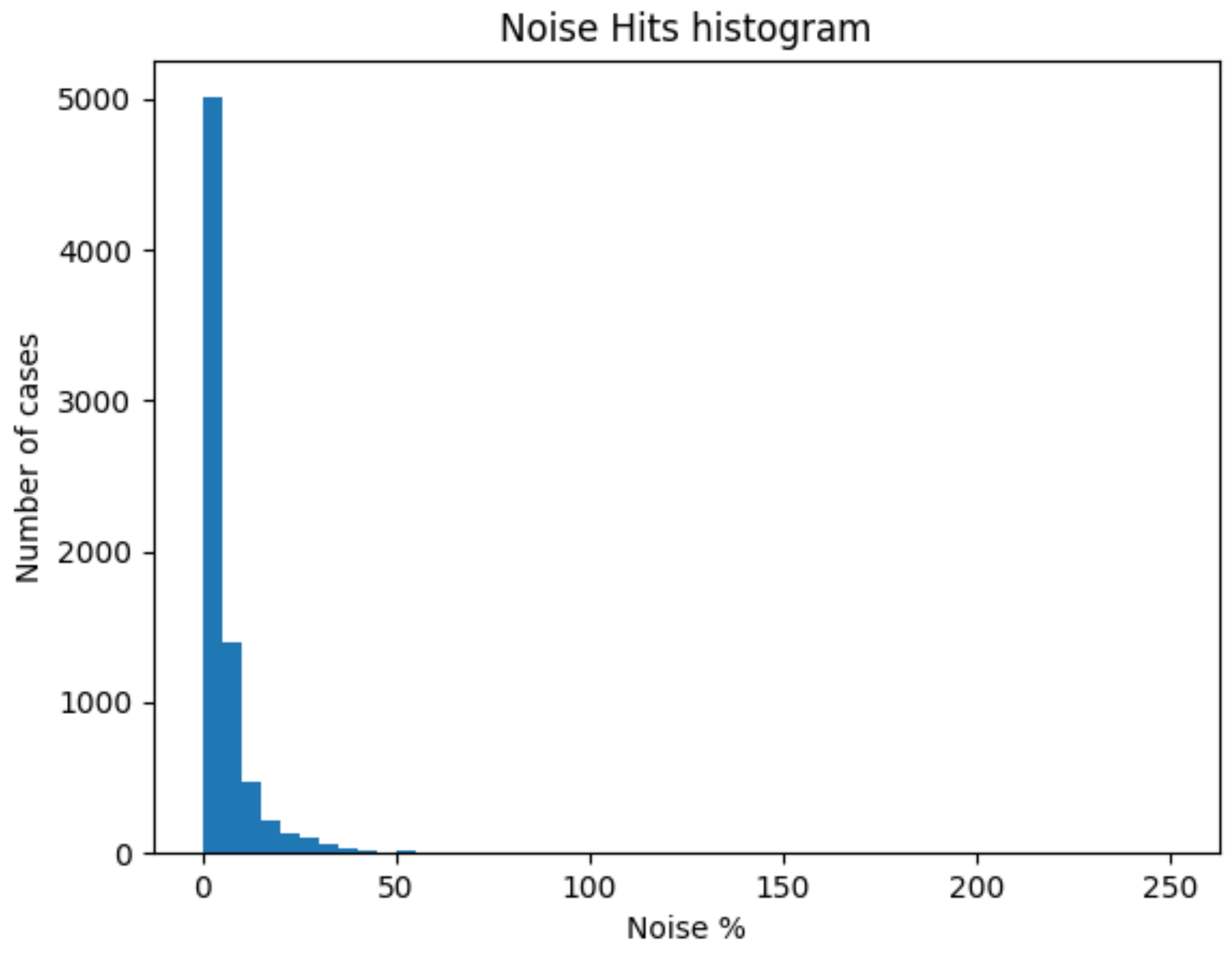
AI Tracking
De-noising



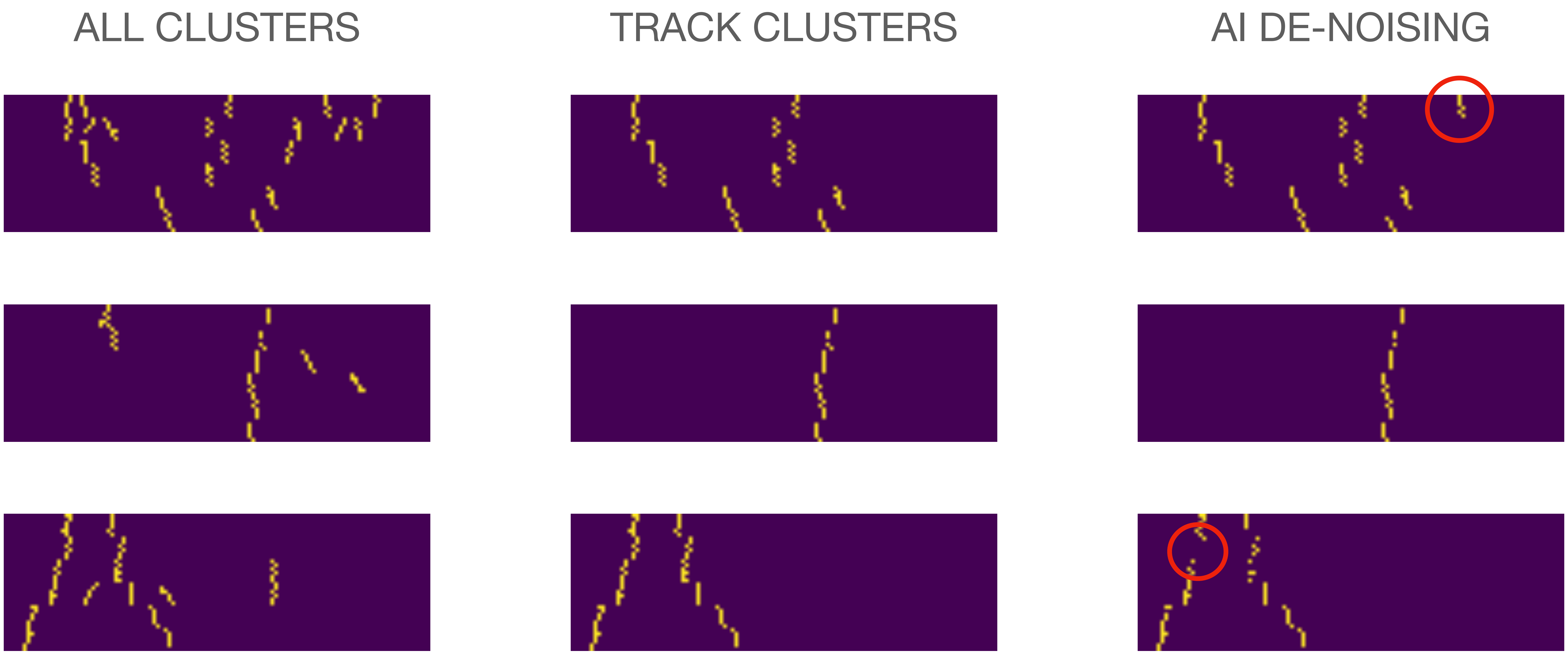
- **Convolutional Neural Network (CNN)**
  - CNN solution works better. Trains well where training loss is similar to testing loss.





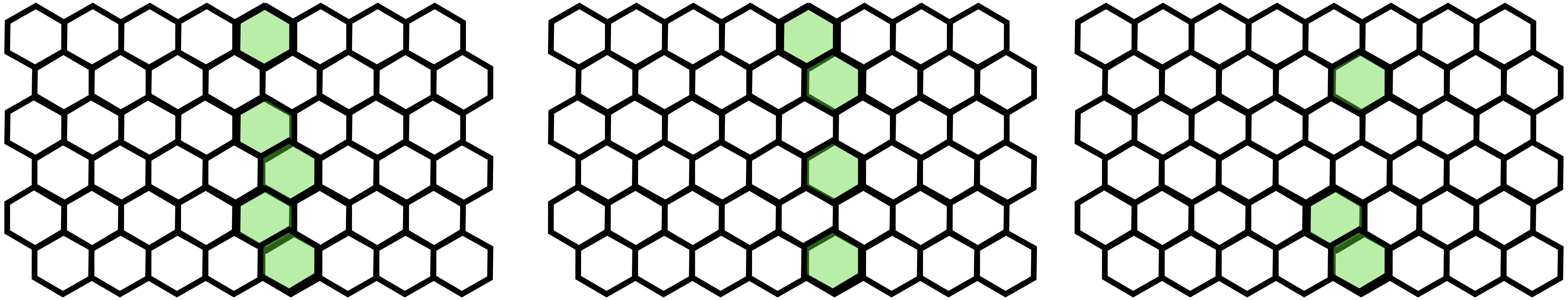


AI Tracking

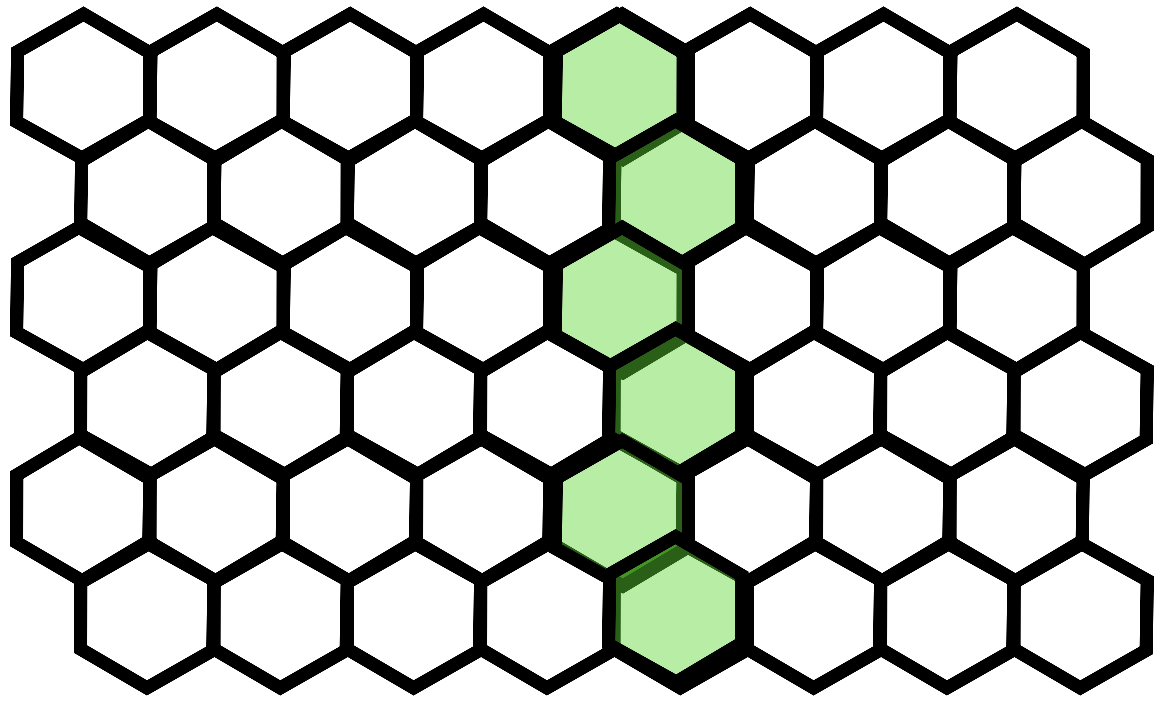


AI Tracking
Efficiency Metrics

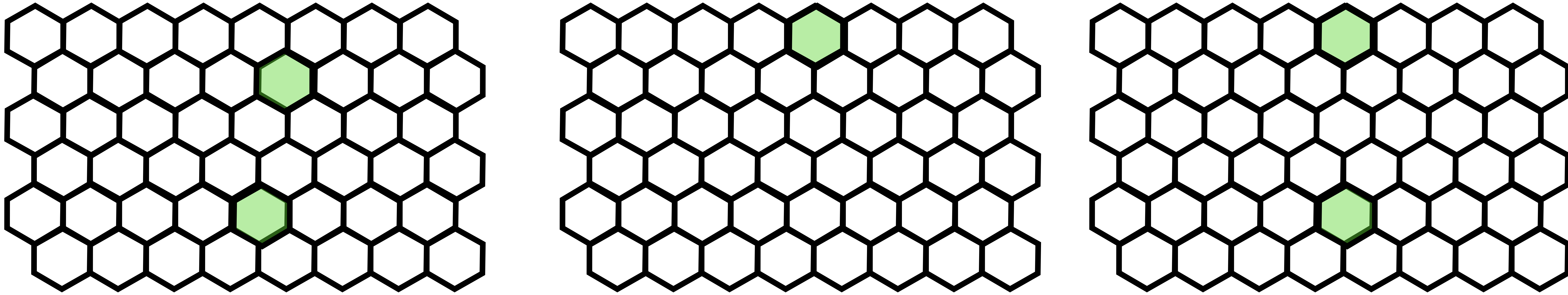
100% efficiency (3 or more coinciding hits)



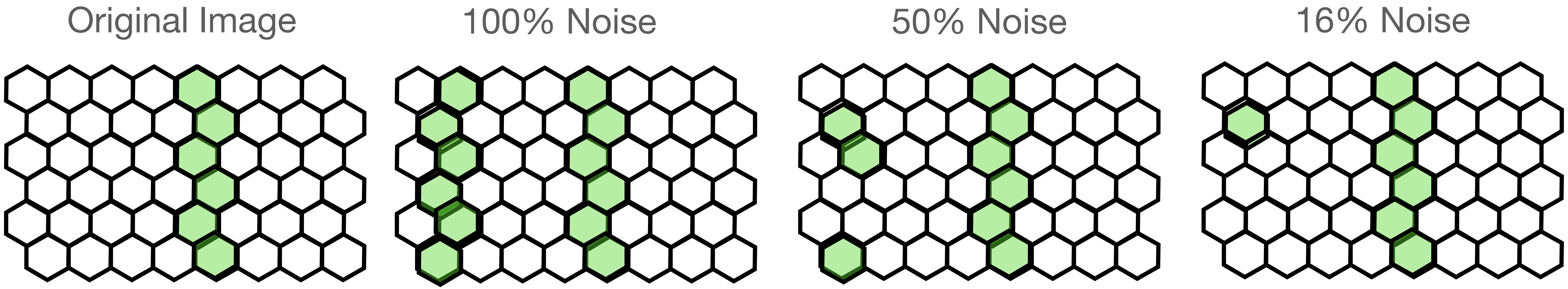
Original Image



0% efficiency (2 or less coinciding hits)

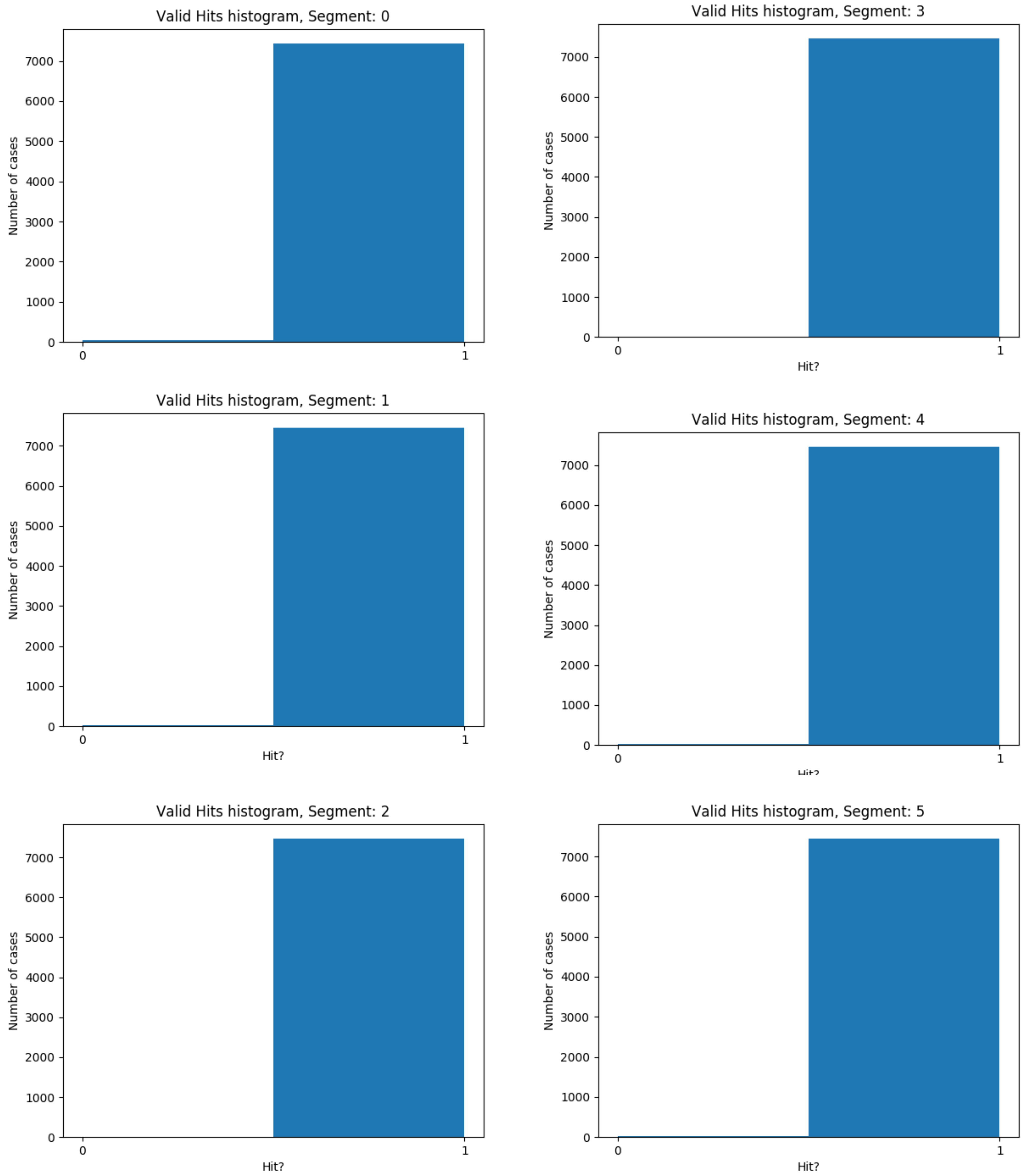


AI Tracking
Noise Metrics

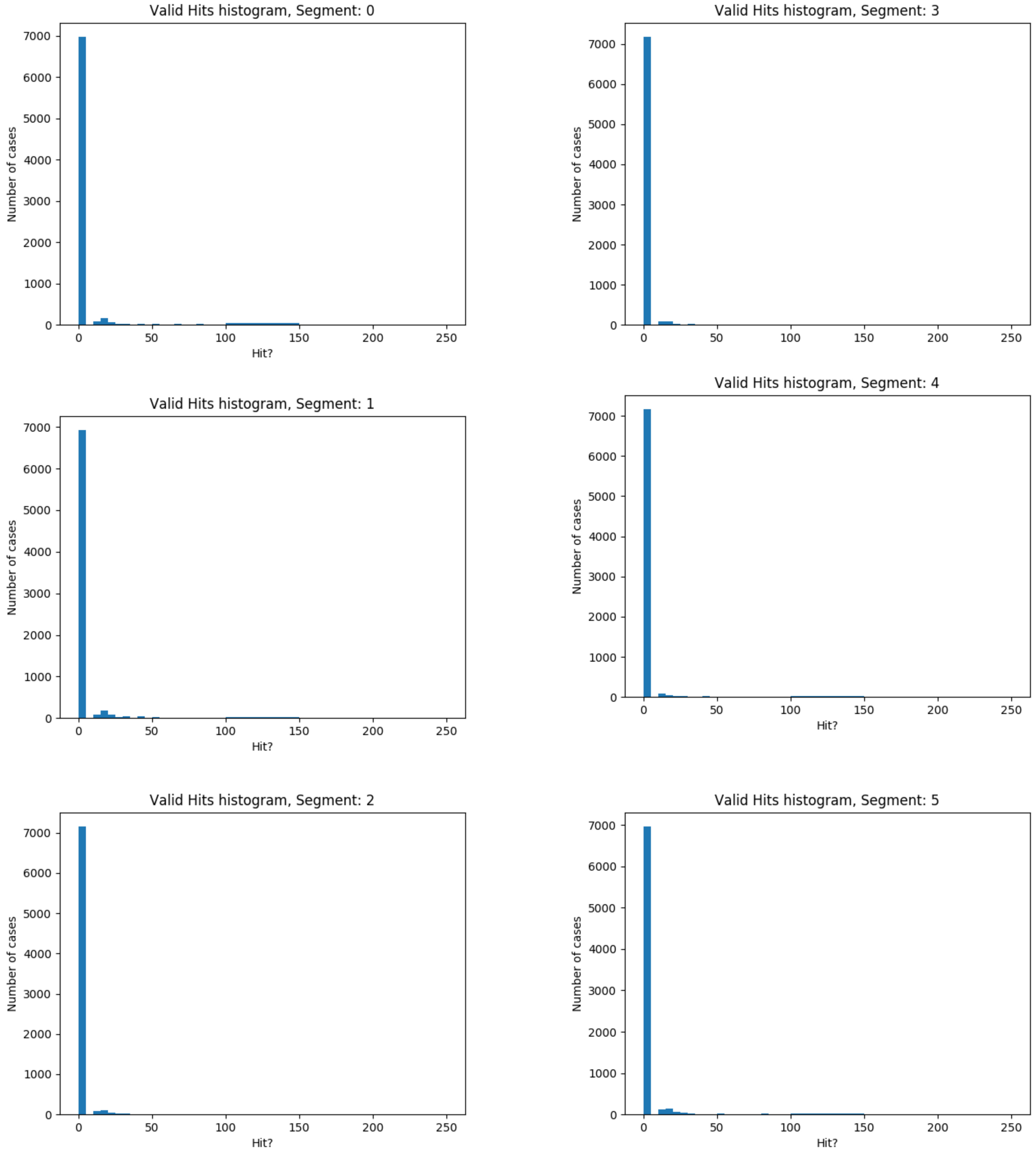


AI Tracking

Efficiency per Super-Layer

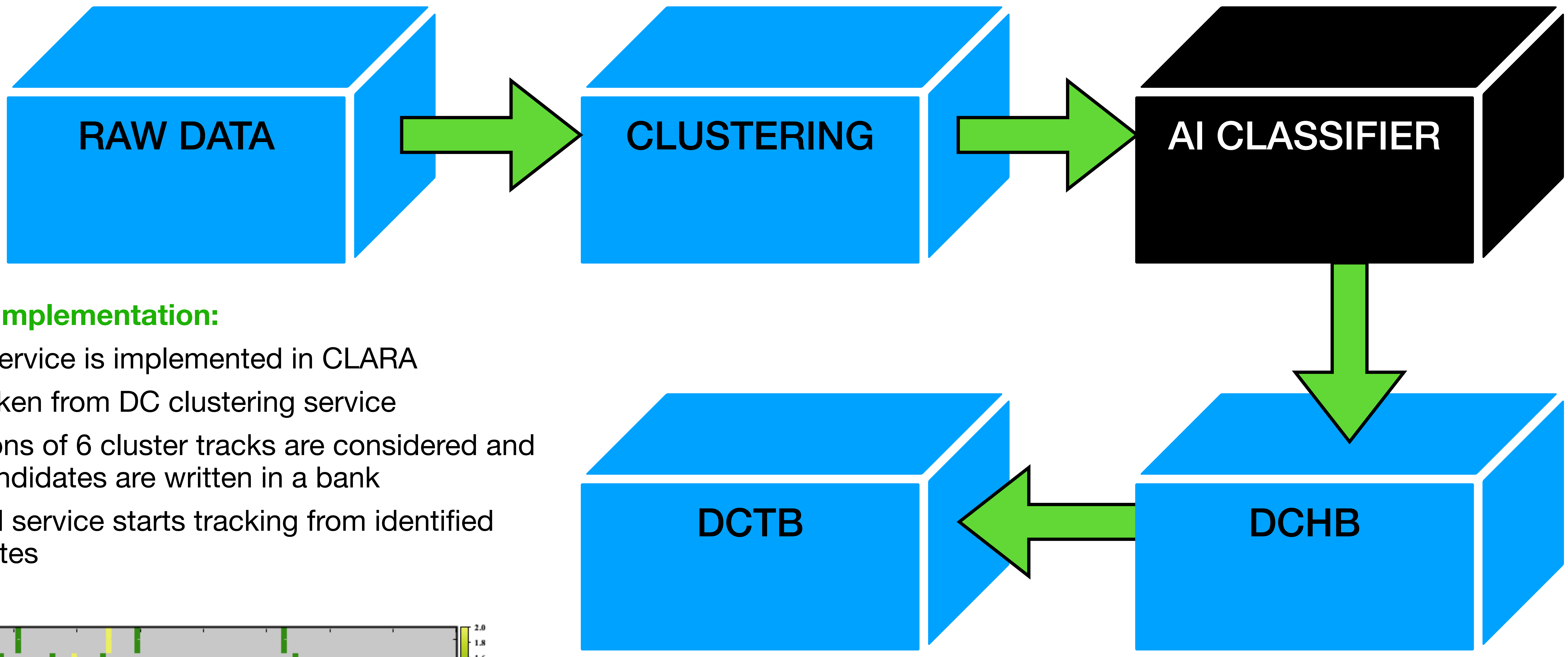


Noise per Super-Layer

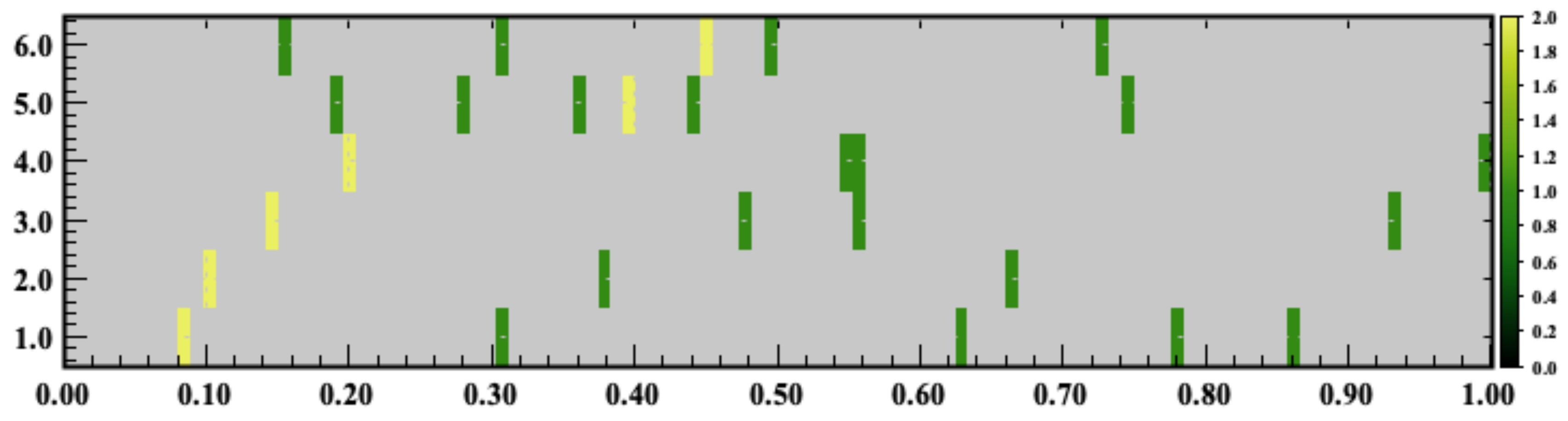


AI Tracking Workflow
Service Composition (current service chain)

Choice of the color is not a coincidence

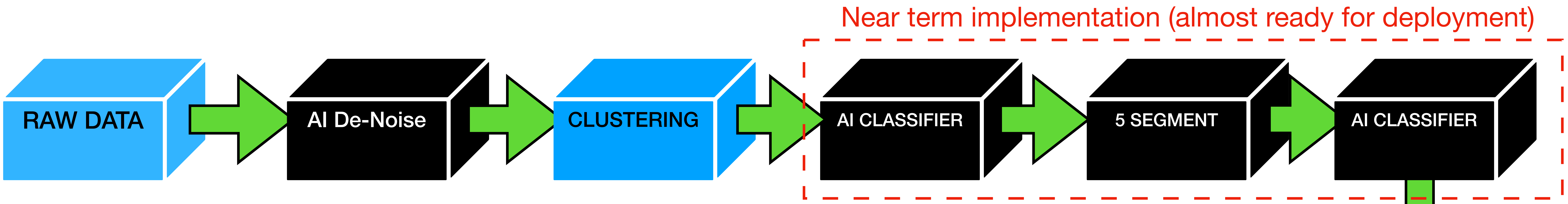


- **Current Service implementation:**
  - AI Classifier service is implemented in CLARA
  - Cluster are taken from DC clustering service
  - all combinations of 6 cluster tracks are considered and valid track candidates are written in a bank
  - DCHB special service starts tracking from identified track candidates

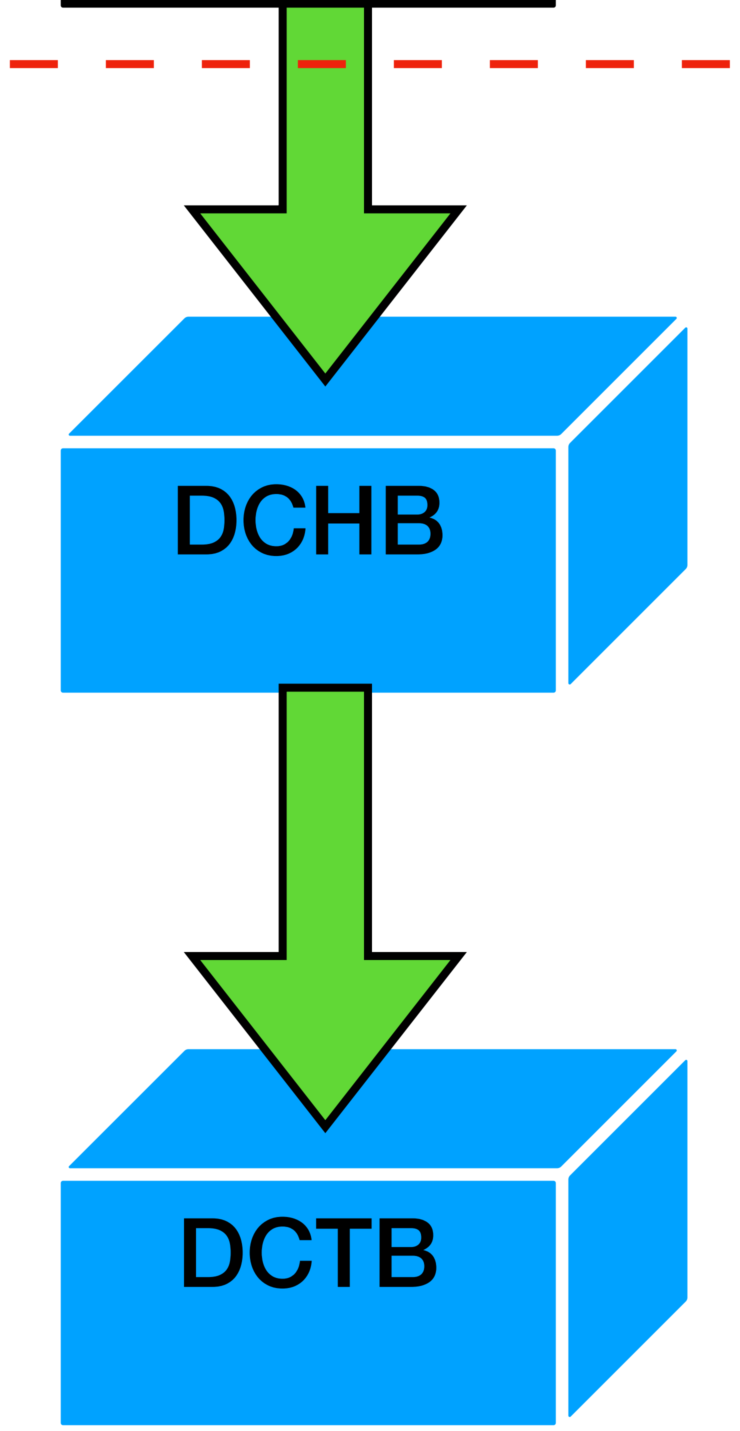


AI Tracking Workflow

Service Composition (future full service chain)



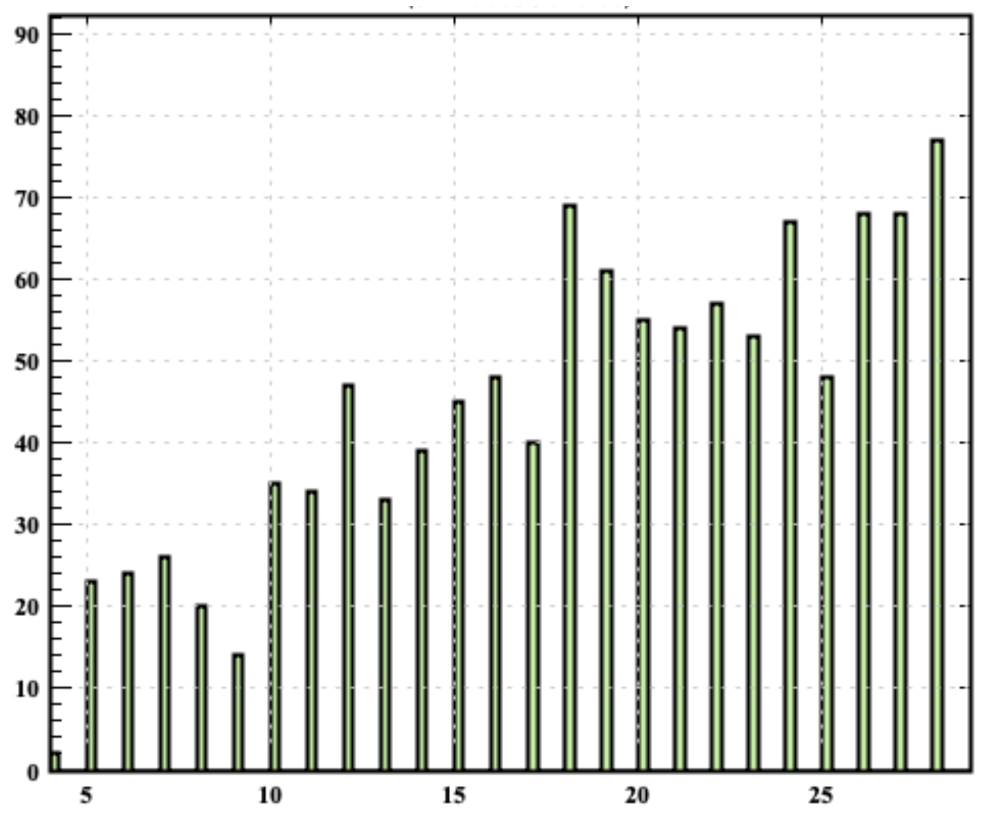
- **AI De-Noise**
  - Remove hits that do not belong to tracks
  - Possibly clean up super layer 1&2 from background hits
- **AI CLASSIFIER**
  - Identify cluster combinations that form a valid track.
  - Works only 6 cluster combinations
- **5 Segment**
  - From remaining clusters form all combinations of 5 cluster track candidates
  - Fix the corruption by identifying possible 6th cluster position.
  - Run fixed (uncorrupted) track candidates through AI classifier to select possible track candidates.



Track Parameter Estimation (in collaboration with GlueX)

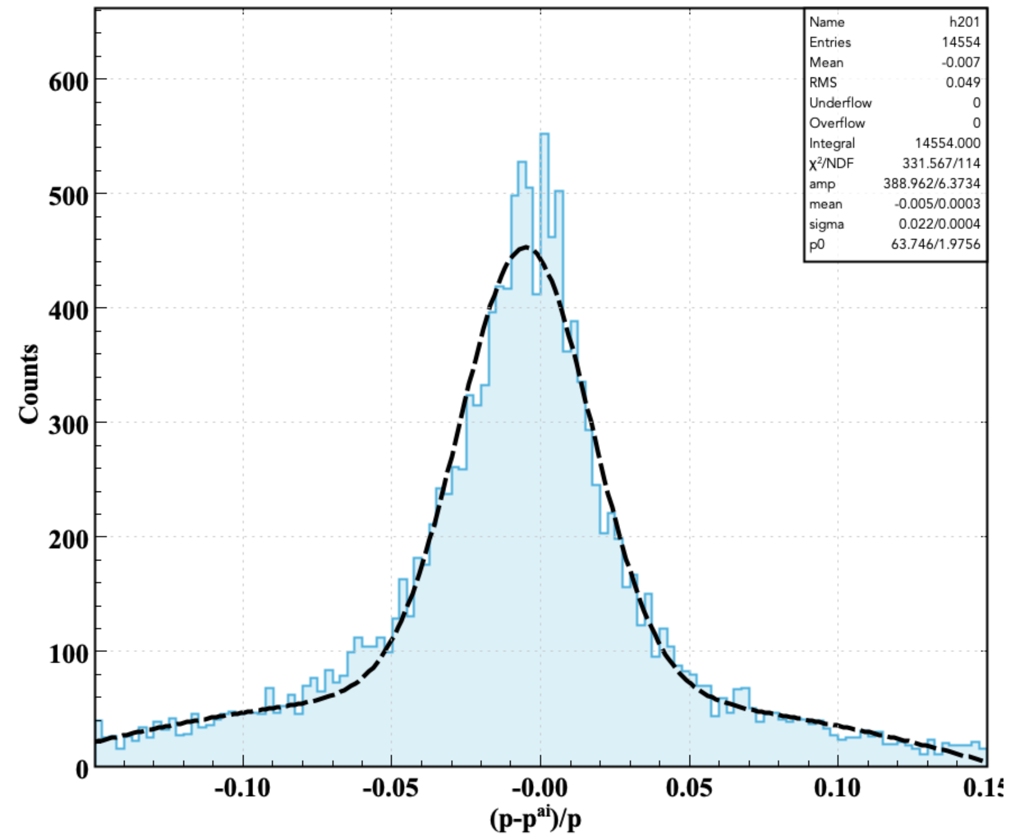
CLAS12 Tracking with Artificial Intelligence

Hit Based Tracking



Number of Iteration of Kalman-Filter

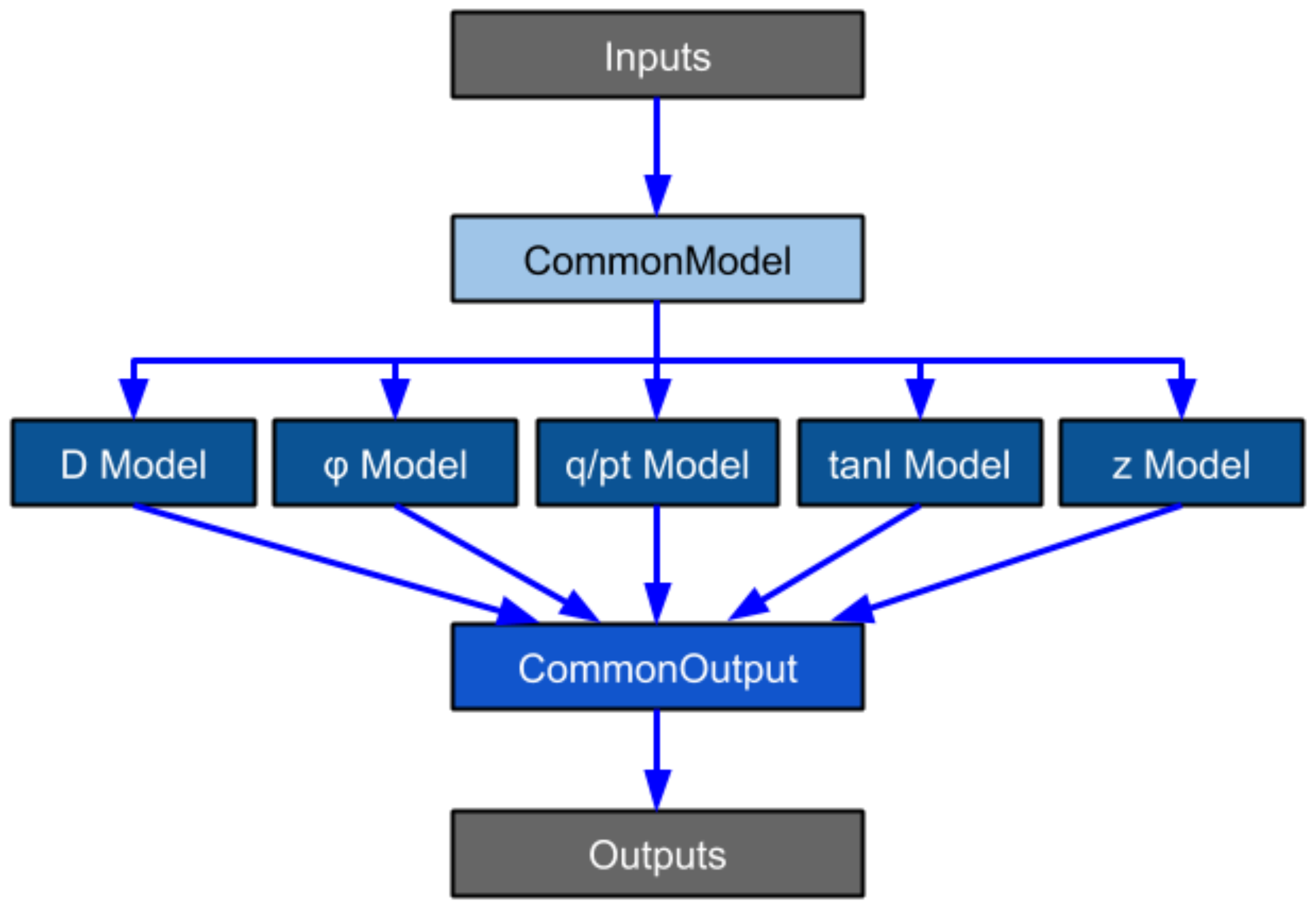
Momentum Predicted by Neural Network



Resolution 2.2%

- **Hit Based Tracking:**
  - The initial momentum of the track in hit based tracking is calculated using polynomial fit to the clusters.
  - Then the track candidate is run through Kalman-Filter several times for the track parameters (i.e. momentum) to converge. (Average number of iterations is ~18)
  - The hit based tracking fit results in ~5% resolution of the track.
- **Track Parameter Estimation:**
  - Initial implementation of track parameter estimation was done in MLP.
  - Resulting resolution achieved was ~2%, better than hit based tracking final result.
  - If integrated in the workflow will provide more accurate initial state vector for hit based tracking, hence decreasing number of iterations needed by Kalman-Filter to converge.
  - Estimated speed gains for reconstruction is about 2-3 times.
- **To Do (need ODU/CRTC student):**
  - Implement the state vector estimator with different network configurations (try CNN,ERT)
  - Implement the training and inference software in Java (using DL4J) for integration with reconstruction software.
  - Integrate the parameter estimator with track candidate classifier to provide full information about the track candidate including momentum and angles.
  - Investigate if we can predict the real sate vector parameters based on the hit pattern.

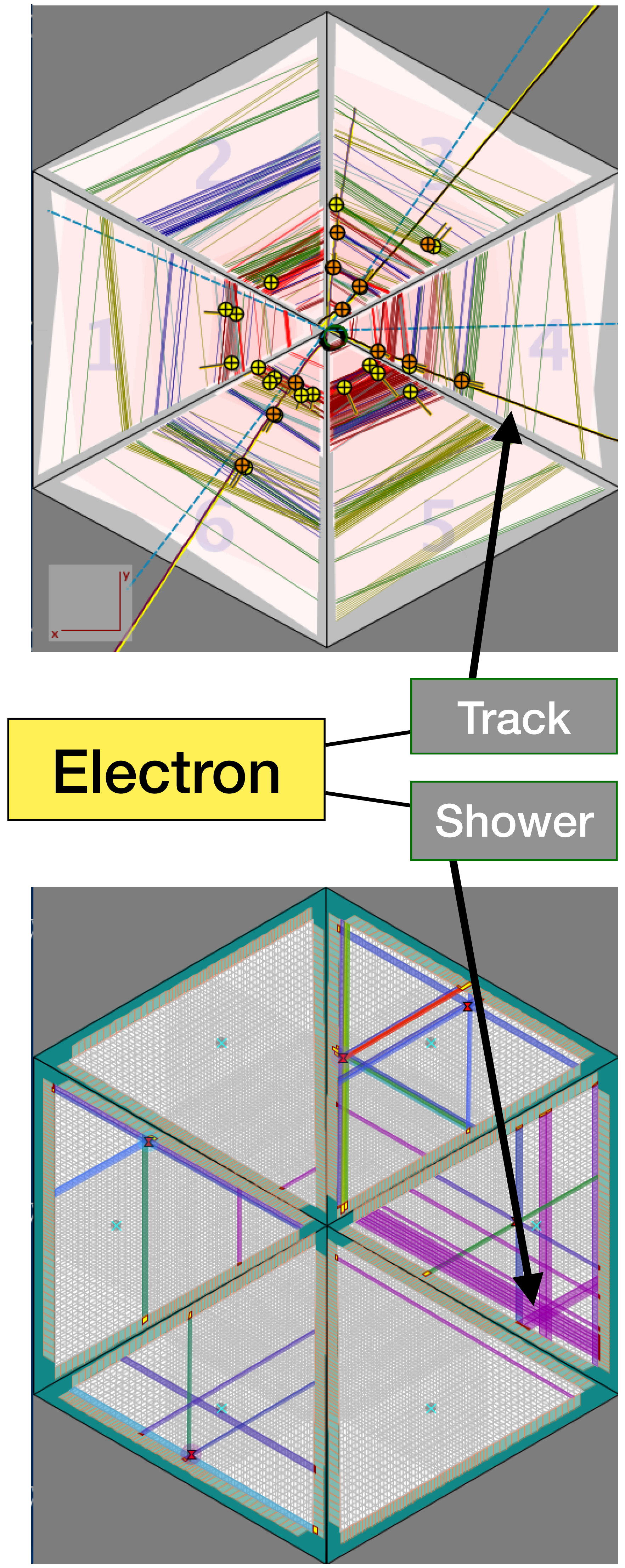
- **GlueX + CLAS12:**
  - We started collaboration with GlueX to join efforts in track state vector estimator, then (COVID-19).
  - Now that initial tests on CLAS12 show good potential in estimating track parameters based on just hit pattern, we should resume this collaboration and try to design a network that can work for both halls.



- Developing regression model for predicting 5-parameter state vector
- Wire hit times and event start times are inputs
- Customized loss function fully incorporates covariance matrix during training
  - ***train on distance between two points in 5-D space which properly includes uncertainty of the labels***

Level-3 Trigger
CLAS12 Tracking with Artificial Intelligence

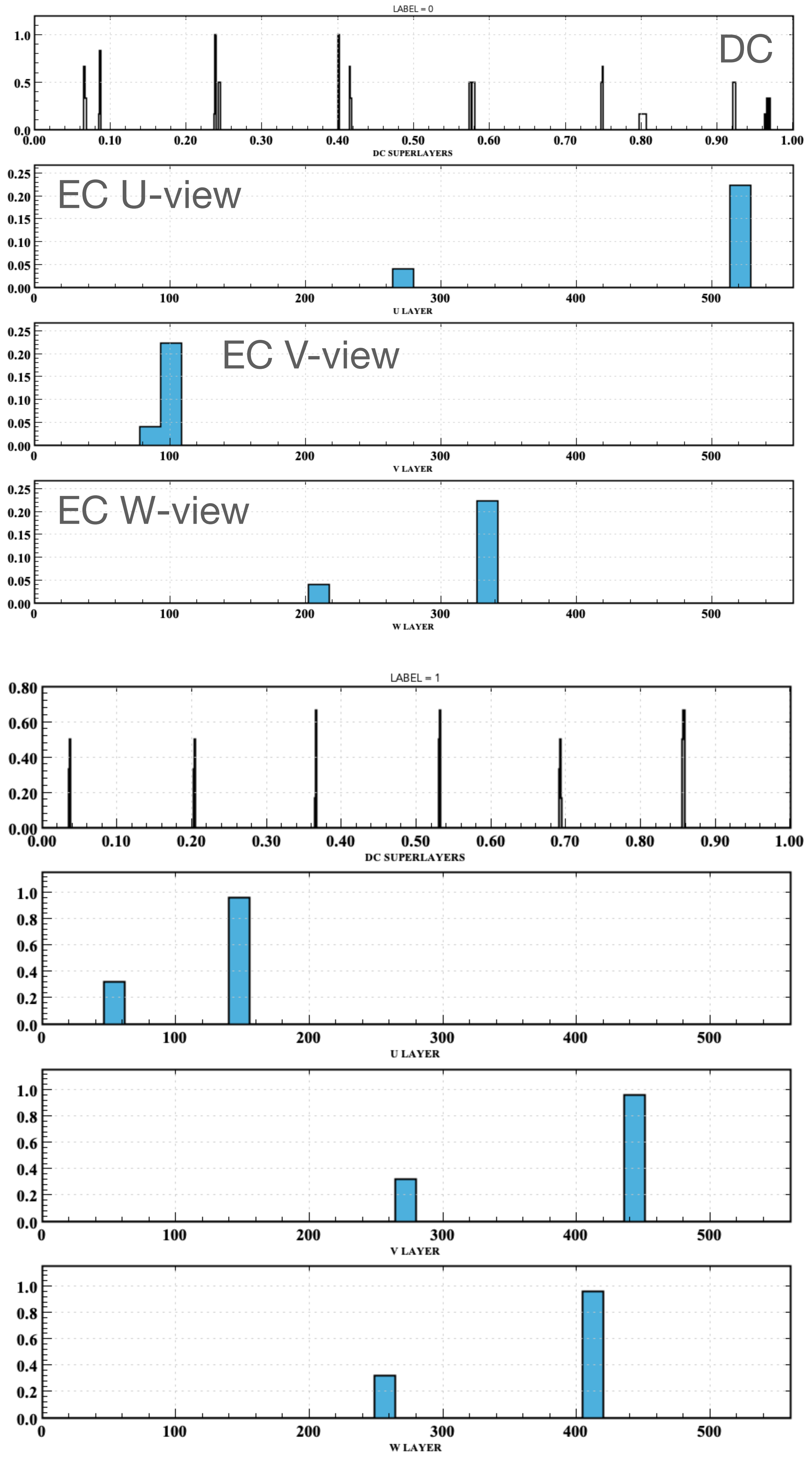
Drift Chambers (Front View)



Calorimeter (Front View)

- **CLAS12 Electron Trigger:**
  - In CLAS12 electrons are identified by matching a track with Calorimeter hit.
  - Energy deposited in the calorimeter should be >25% of the tracks momentum.
- **Level-3 Trigger:**
  - Currently trigger identifies tracks in given sector a calorimeter hit which is larger than some threshold
  - If the criteria is met in any of the sector the event is written out, no matching is done between calorimeter and tracking
- **Neural Network Solution:**
  - Raw information from DC and EC are combined into 780 input nodes.
  - Reconstructed data is used to identify sectors that contain electron and sectors with a track and a calorimeter hit that are not electrons
  - Positive and negative samples are used to train the network and then run evaluation on existing data. (shown on the RIGHT)
  - Results:
  - Network provided electron identification accuracy of ~97.2%
  - The inference speed is 85Kz (running on single CPU), when running on raw data (no preprocessing is needed)
- **To Do (need ODU/CRTC student):**
  - Improve network architecture to increase accuracy, investigate how classification threshold can affect accuracy vs purity. (should not loose any good events, while keep noise events count low)
  - Implement the network in the online workflow for constant training and inference.

RAW hits from TDC and ADC (normalized)



Summary
AI Projects

- Track classification network is implemented as a service
  - ☑ First validation yields to 99.7% accuracy in track candidate identification.
  - ☑ There is significant speed up Hit Based Tracking x4, Time Based Tracking x2
  - ☑ Full validation is in progress (service work)
- 5 cluster track identification software is developed
  - ☑ Development and testing of the algorithm is done (published on ArXiv)
  - ☐ Will be implemented as a service soon.
  - ☐ Needs to be validated after implemented as full part of the tracking code
- Research is ongoing on de-noising network
  - ☐ Will be implemented as service after all details are ironed
  - ☐ needs testing in high luminosity setting to see if improves efficiency
  - ☐ Service will be deployed before clustering to clean up the DC
- Level-3 Trigger:
  - ☐ Initial Development shows good potential for effective electron classification
  - ☐ Work is ongoing to improve the network and to integrate is online

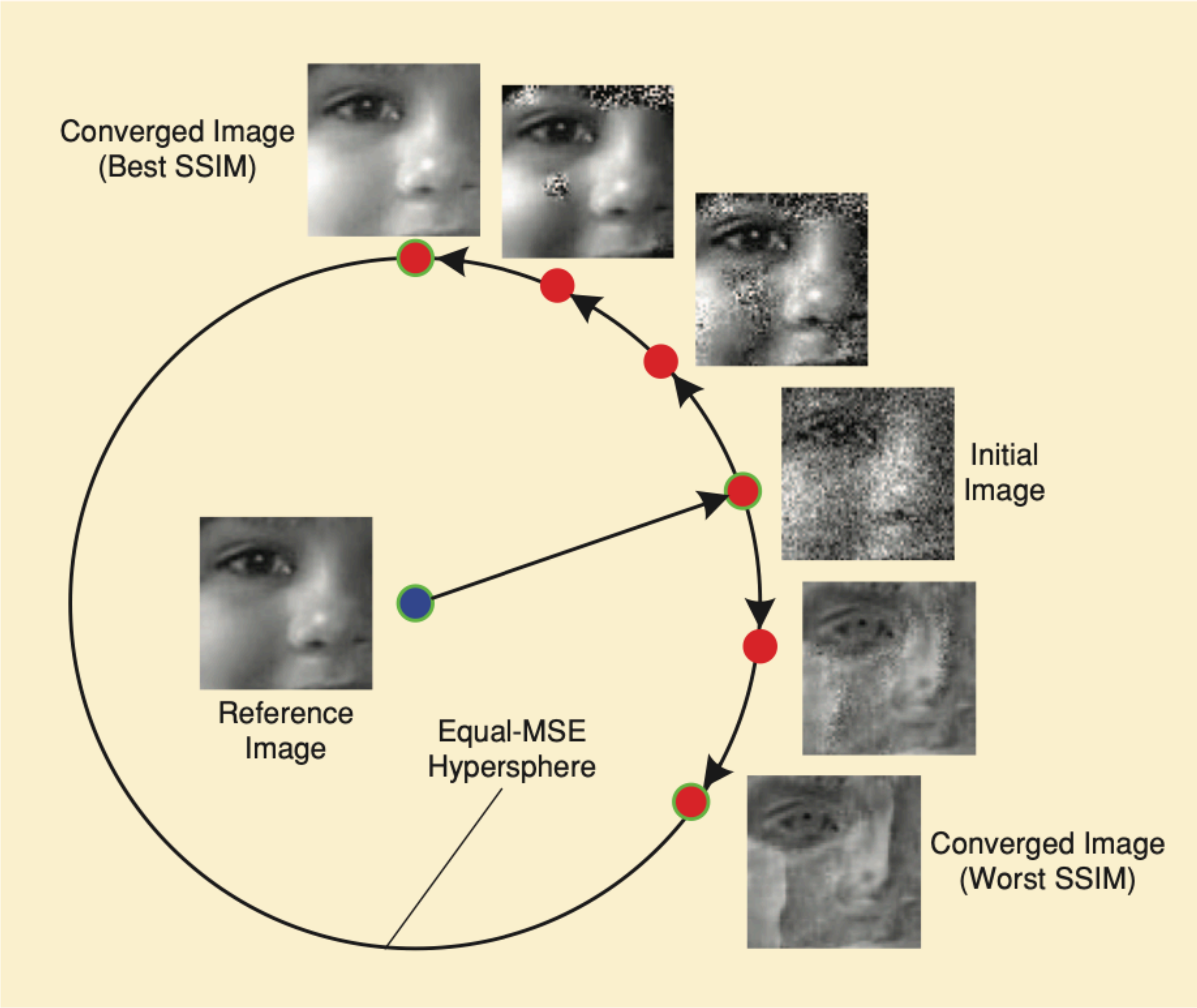
The End

"Now you'll have more time to binge things."

BACKUP SLIDES

Track Candidate Classification

CLAS12 Tracking with Artificial Intelligence

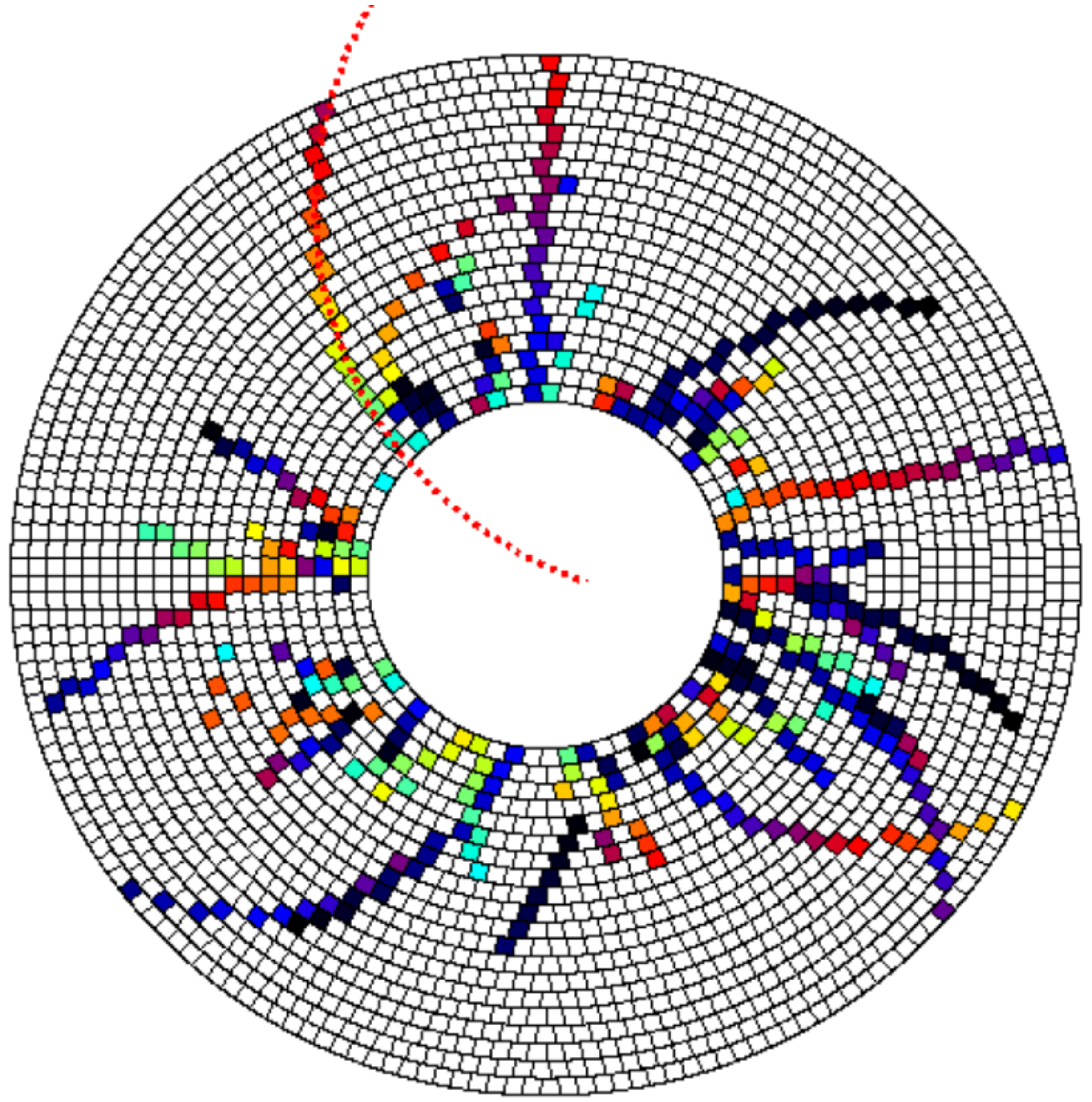


- **Image reconstruction:**
  - Most algorithms for classification and regression are using Mean Square Error (MSE) for training network and deriving the gradient of weight change.
  - It has been shown that for image comparisons this metrics might not be the best to identify similarity between images.
  - Drift chamber data is also presented as an image including noise hits and might need rethinking what metrics to use to drive training of network.
- **Status of The Project:**
  - We are investigating literature to see what is used in de-noising and image scaling algorithms as reconstruction closeness metrics.
  - Will implement several loss functions and evaluate network performance for all types of loss functions.
- **Different Metrics:**
  - Structural Similarity Index (SSI) was developed for comparing images.
  - Mainly used for image enlargements they claim this is best way to compare two images.
  - This seems applicable for Drift chamber data, since we are looking for specific structures in the output image.
  - This is new, we just started working on this, will report results in the next progress report.

AI

JLAB Tracking with Artificial Intelligence

Tag = Low momentum proton
Detect in mTPC (multiple – TPC)
in solenoidal field.

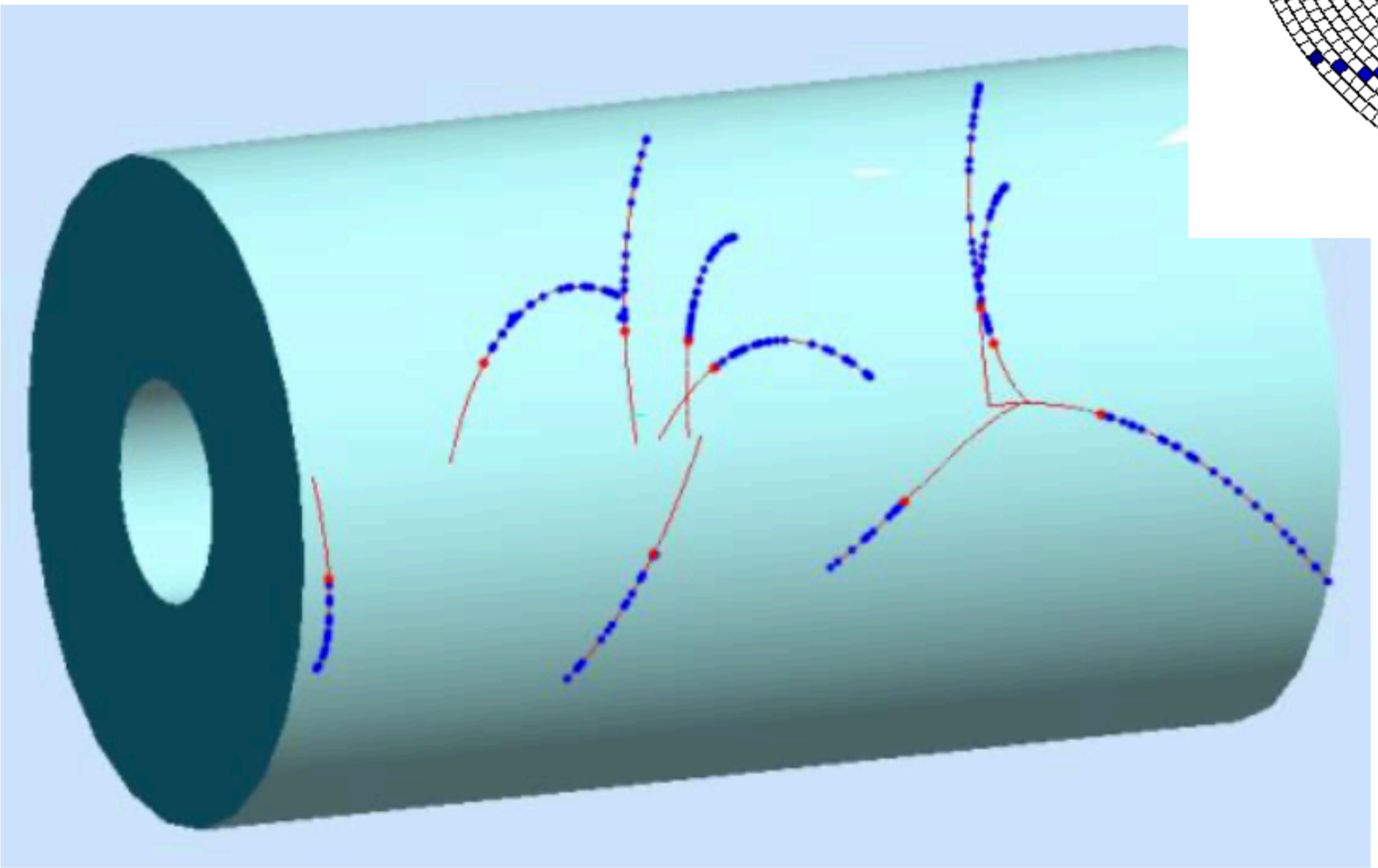


100's of accidental proton's per electron trigger

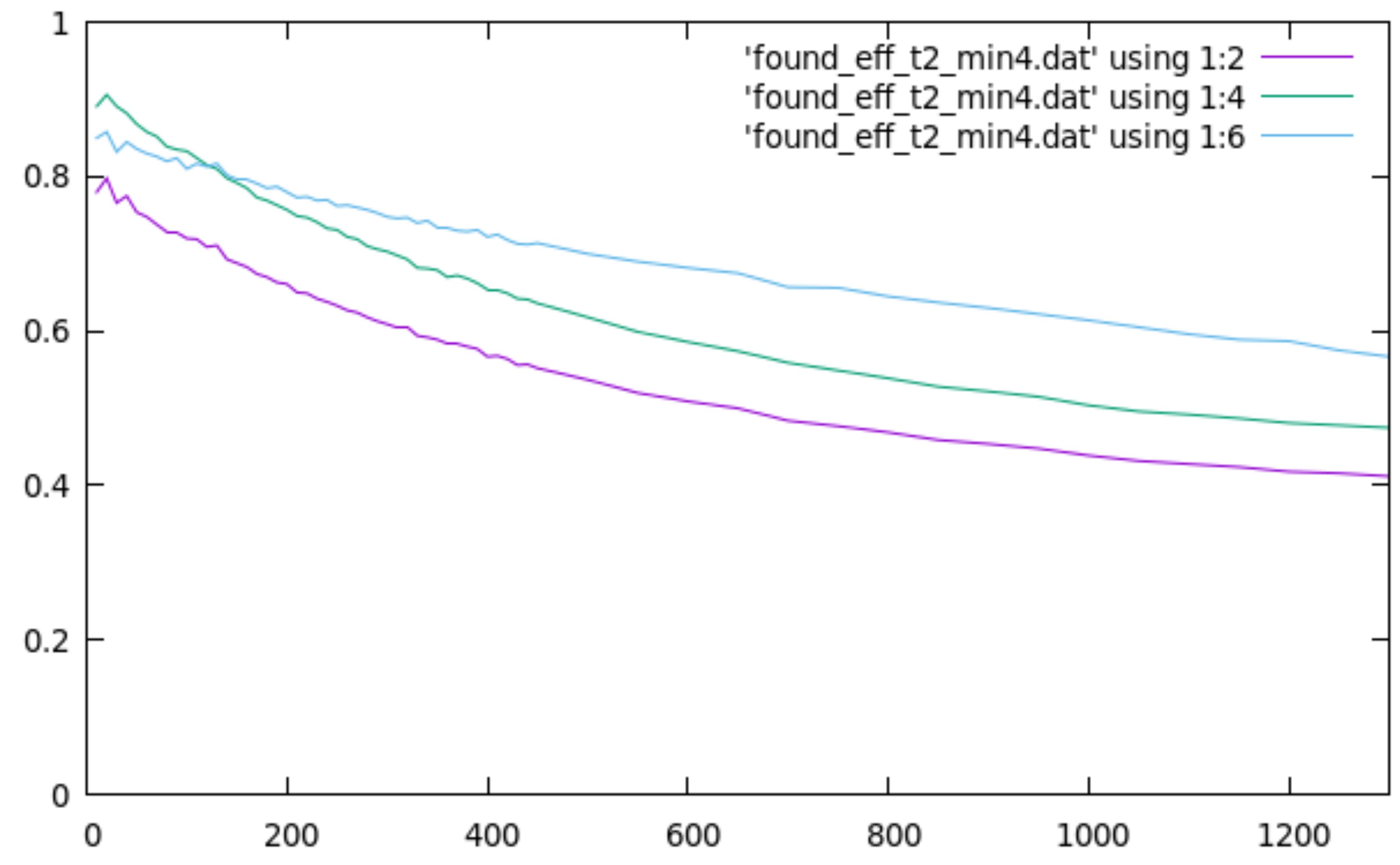
Hit information - (x, y, tdc)
tdc = $z/v_{drift} + t_{unknown\_offset}$ + smear

Simple tracking algorithm does well (~ 50% tracks found) with 1000 tracks/event in toy (not G4) simulation.

Could ML help with realistic simulation?



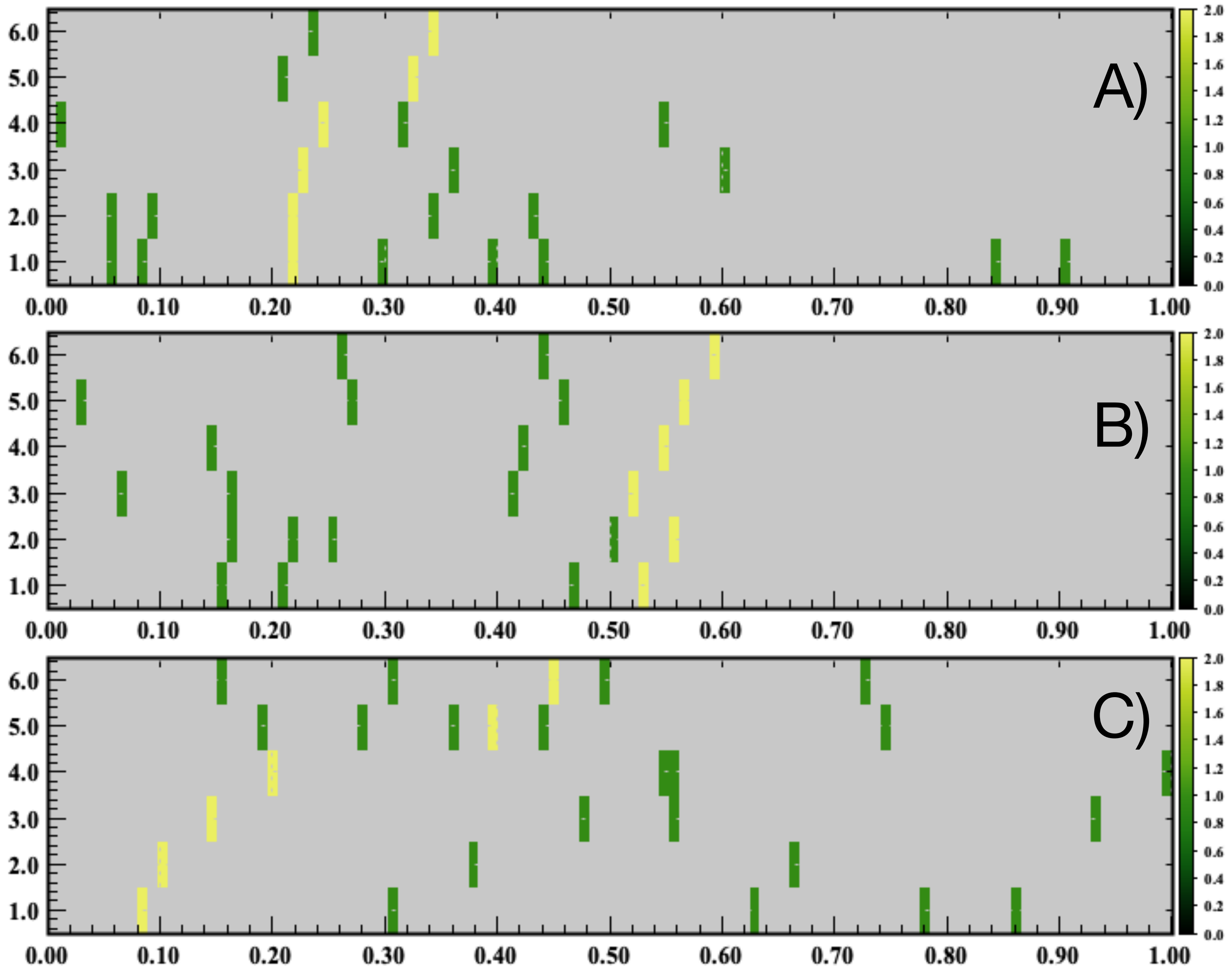
mTPC



Track Candidate Classification

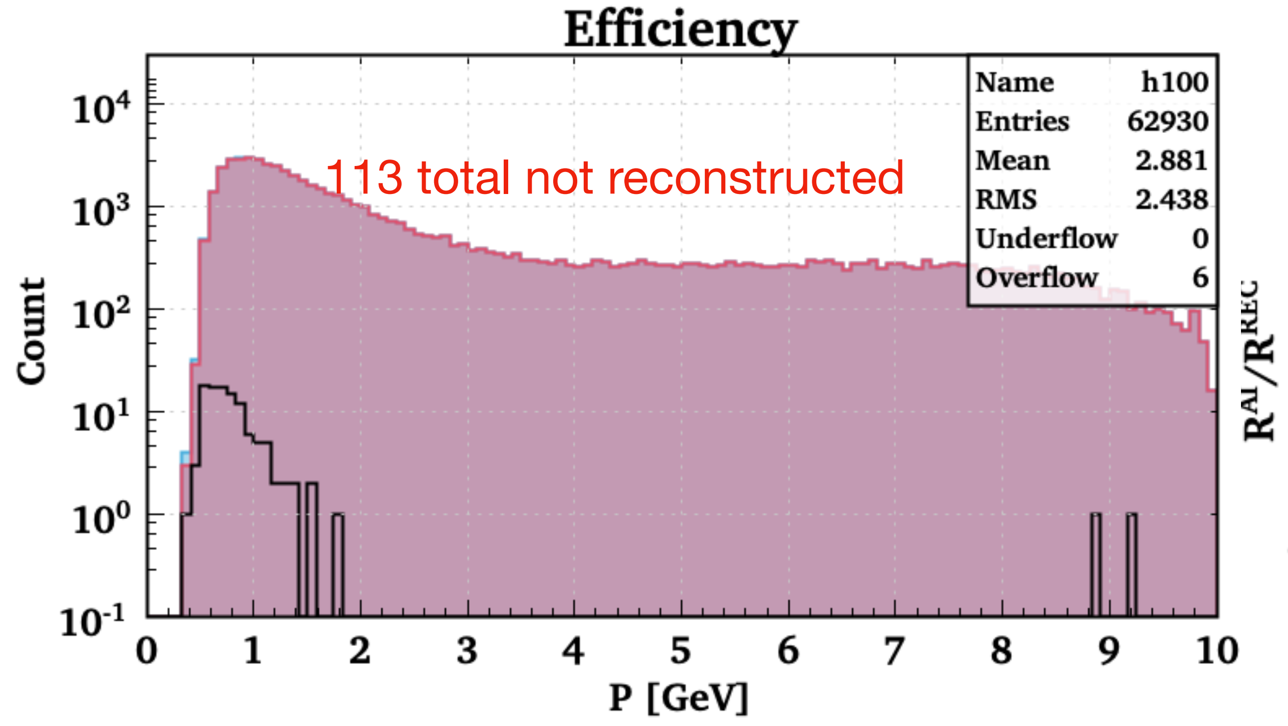
CLAS12 Tracking with Artificial Intelligence

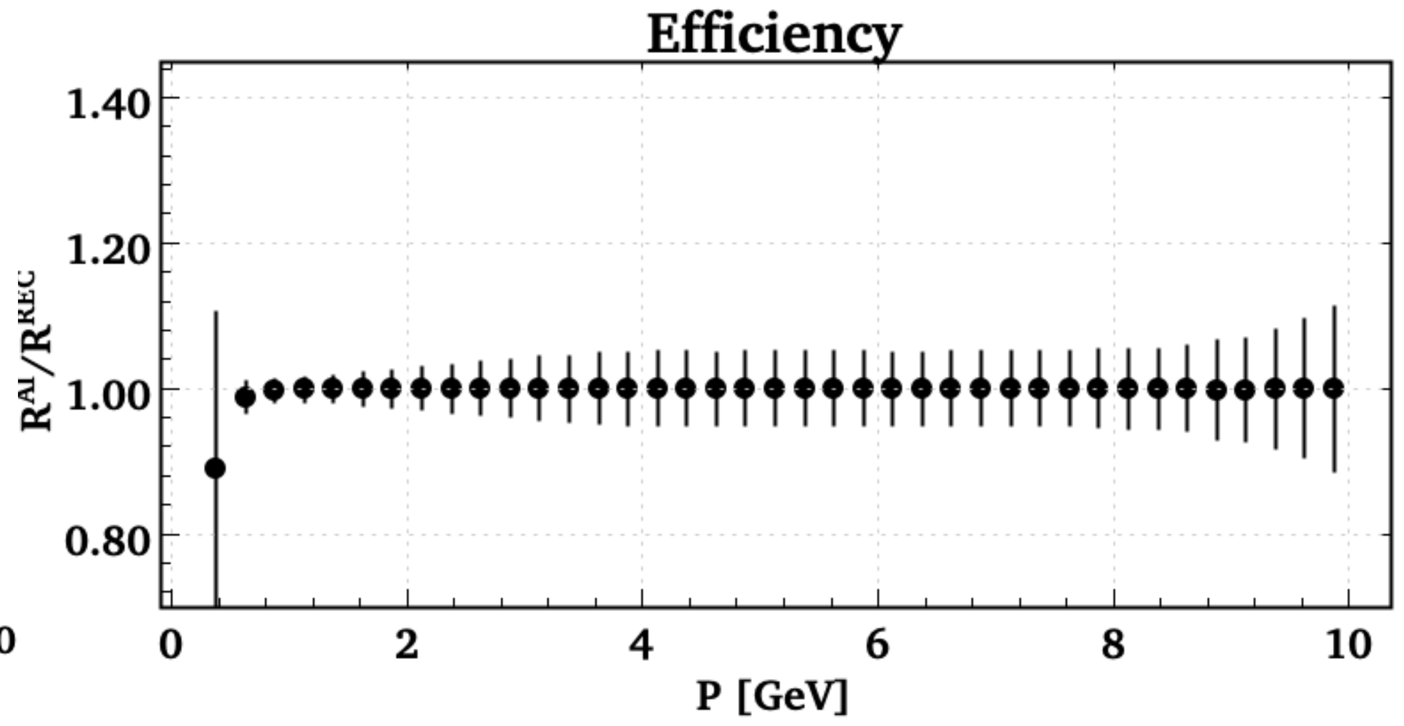
AI Track reconstruction from cluster combinations
Number of combinations: A) 2304, B) 2880, C) 7200
AI successfully picked the right combination of clusters.



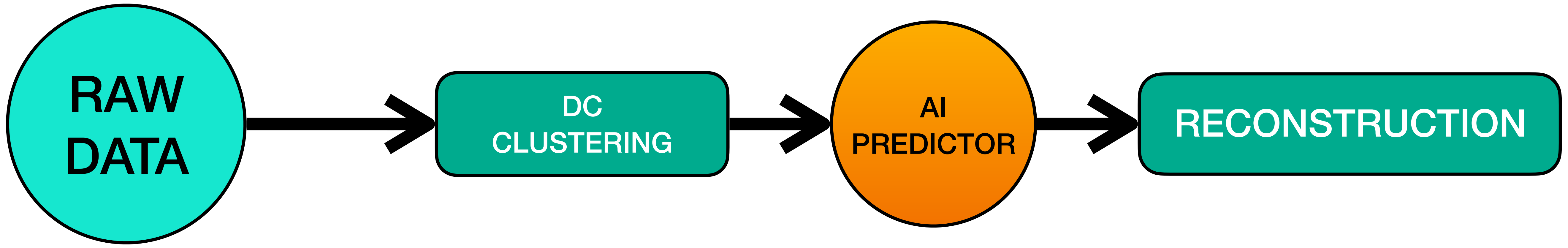
- Neural Network trained on cluster combinations.
- Several Network Architectures are considered
  - Convolutional Neural Networks (CNN)
  - Multi-Layer Perceptron (MLP)
  - Extremely Randomized Trees (ERT)
- Accuracy determined by Confusion Matrix
- Multi-Layer Perceptron performed the best.

Network Architecture	Accuracy	Inefficiency
MLP	**99.7%**	**0.3%**
CNN	95.6%	4.4%
ERT	98.5%	1.5%





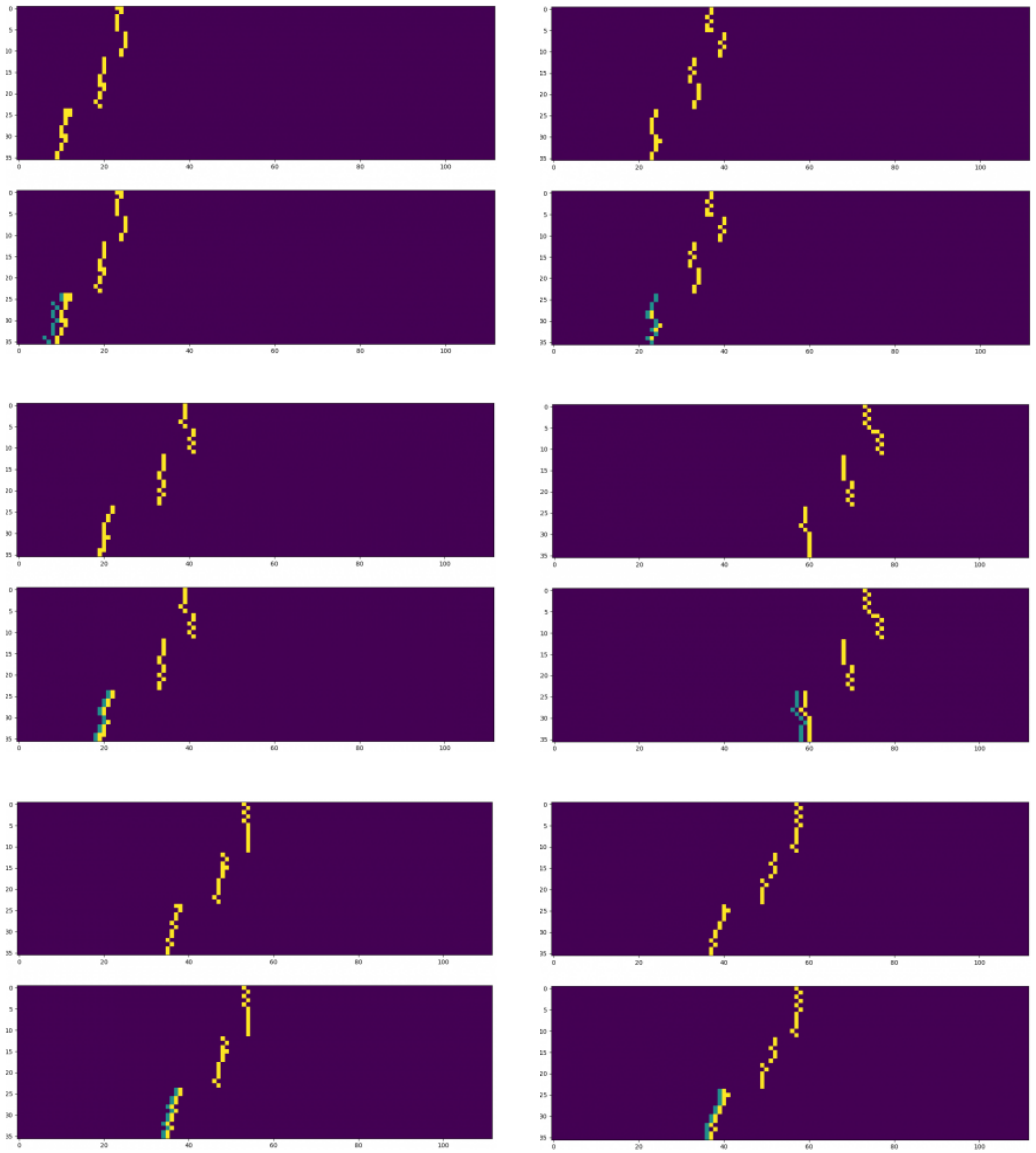
- **AI Track Classification Efficiency:**
  - Only 113 tracks were not identified by AI (out of 62,930) - 0.18%
  - 70 % of not reconstructed tracks are outside of fiducial region (edges of Drift Chamber)
- **Status of The Project:**
  - Utilities are completed for:
    - ☑ Extracting training sample from from reconstructed data sample
    - ☑ Training the Neural Network
    - ☑ Running inference on RAW data with clusters in Drift Chamber
  - The software will be used in next calibration data processing, where efficiency is not important
  - The current test show 5-6 times tracking speed improvement.
  - Further efficiency studies are needed to confidently include this in the production cooking



Published on arXiv: Using Artificial Intelligence for Particle Track Identification in CLAS12 Detector
Status: pending approval, will be submitted to pier-reviewed Journal

Track Trajectory Prediction

CLAS12 Tracking with Artificial Intelligence



- **Track Trajectory Prediction:**
  - Region 2&3 (furthest from the beam) have less noise and clustering efficiency is high
  - Region 1 (closest to the beam line) has high background and hits can not be clustered efficiently in high luminosity runs, causing for tracking efficiency to drop.
  - If we can predict the position of hits in region 1 based on region 2&3 information we can use this to increase tracking efficiency
  - Combined with the previous project (track candidate classification) will improve reconstruction speed and clustering and tracking efficiency.
- **Status of The Project:**
  - Initial LSTM network was constructed to test on sample data (in Python).
  - Test show that the algorithm provides very high efficiency of finding the track trajectory with average mean deviation of 1.18 wires.
- **To Do (need ODU/CRTC student):**
  - Refine data sample used for training, includes eliminating tracks with bad Chi2, and include only tracks that come from target.
  - Develop the full workflow to extract the training sample from reconstructed data and process the training of Neural Network
  - Implement trajectory predictor in the software (Java based, initial test were done in Python) and integrate it with reconstruction software.
  - Modify the DC code to recounted region 1 hits, based on trajectory predictions before passing clusters to AI predictor (from previous project, fully integrated) for track candidate classification.

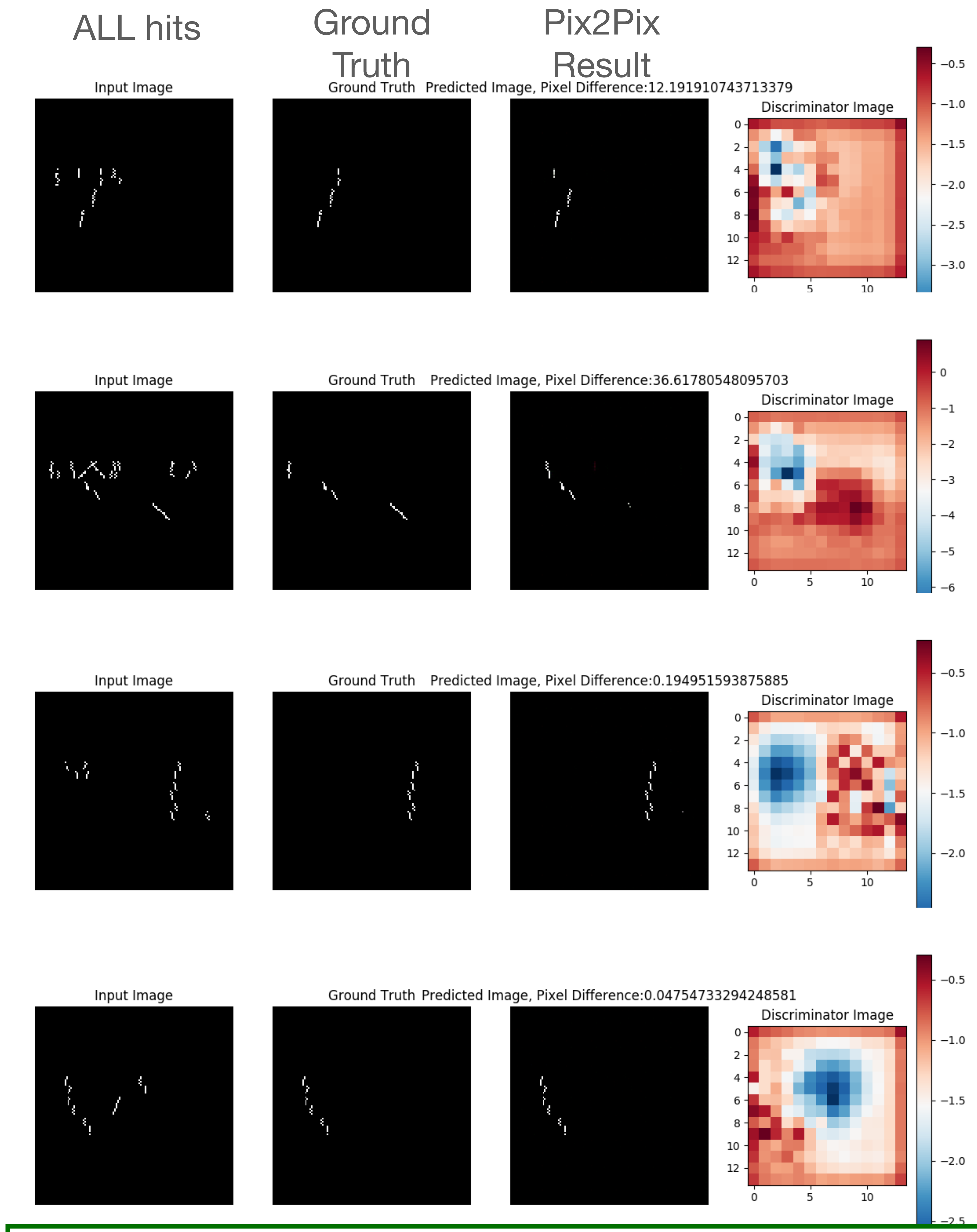
Model Type	Loss (MAE)	Time to Train	Time to Predict / sample
RNN/GRU	~1.18	374 *sec*	688 μs

In Preparation for arXiv: Track Trajectory prediction for CLAS12 using RNN
Status: in progress, will be submitted to pier-review Journal

DriftChamber Hits Cleanup with GAN (pix2pix)

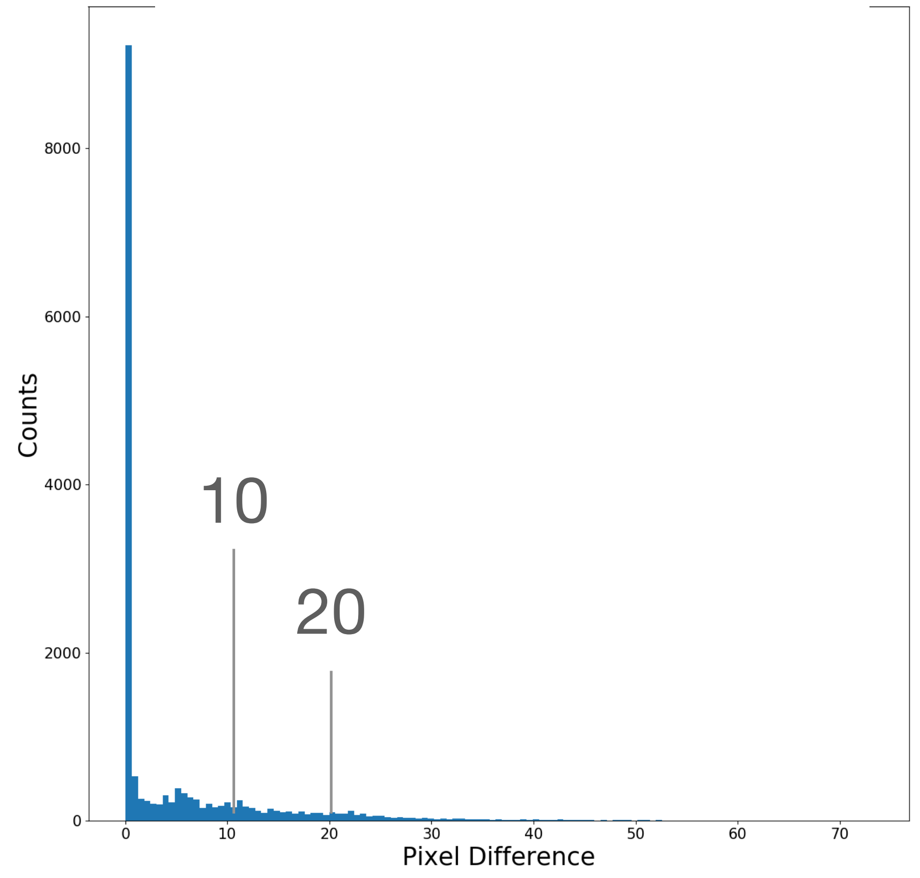
(In collaboration with Davidson College)

CLAS12 Tracking with Artificial Intelligence



- **Drift Chamber hit cleanup**
  - Another approach for increasing efficiency of tracking in high luminosity is to clean noise hits.
  - This will help clustering algorithm to form clean clusters in regions with high occupancy.
- **Pix2Pix:**
  - Pix2Pix is special type of GAN that starts on given image and produces the altered image.
  - The discriminator then measures the difference between the produced image and the ground truth and provides feedback to the GAN
- **Status of The Project:**
  - Data in given sector was selected where reconstruction algorithm reconstructed 1 track (and only 1).
  - AI was given the image with all the hits in the event
  - The discriminator was provided with the hits belonging to the track as ground truth
  - GAN was trained to be able to produce a picture that matches ground truth starting from image of all hits.
- **To Do:**
  - In the initial state of this project we used only data where only 1 track is present.
  - Evolve network to clean the noise hits with more than 1 tracks present in the data (start with 2, then more)
  - Study efficiency in more details, assess accuracy and purity for the network
  - Integrate the pix2pix into data processing workflow
  - Possibly can be used in the on-line data processing to reduce footprint of data.

Pix2pix performance



Pixel difference between GAN's picture and Ground Truth

In Preparation for arXiv: Noise reduction in CLAS12 Drift Chambers using Pix2Pix
Status: in progress