ICSC
Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

Simulation Based on Garfield++ and Hyperparameter Optimization for Deep Learning Model Using High Performance Computing

INFN
Istituto Nazionale di Fisica Nucleare

Speaker : Muhammad Numan Anwar

PhD Student at POLIBA INFN, Bari

Bari-Lecce Meeting
Date: 23 Oct, 2024

de' remi facemmo ali
1990
POLITECNICO DI BARI
Politecnico di Bari

Outline

- Simulation based on Garfield ++ for 2024 data
- Training of Long Short Term Memory (LSTM) Model Using HPC Resources
- Long Short Term Memory (LSTM) Model for Peak Finding Algorithm
- Training of Convolutional Neural Network(CNN) Model Using HPC Resources
- Convolutional Neural Network(CNN) Model for Clusterization Algorithm
- Future planing

Simulation Based on Garfield ++ for 2024 data

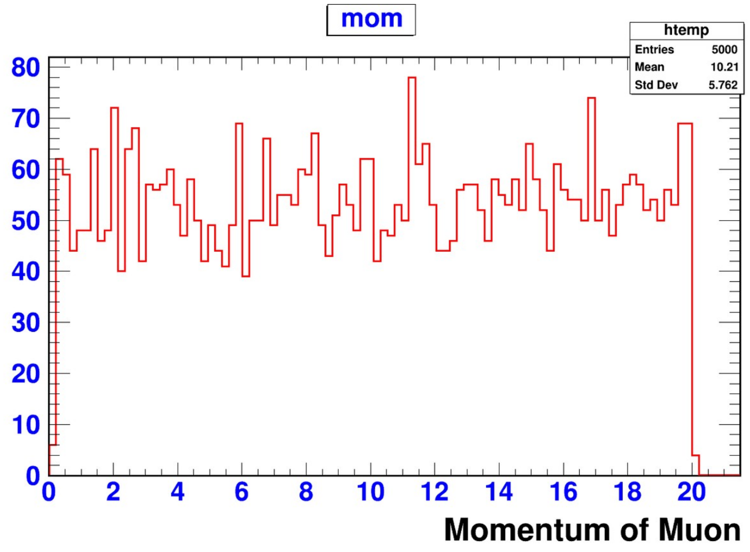
Simulation Based on Garfield ++

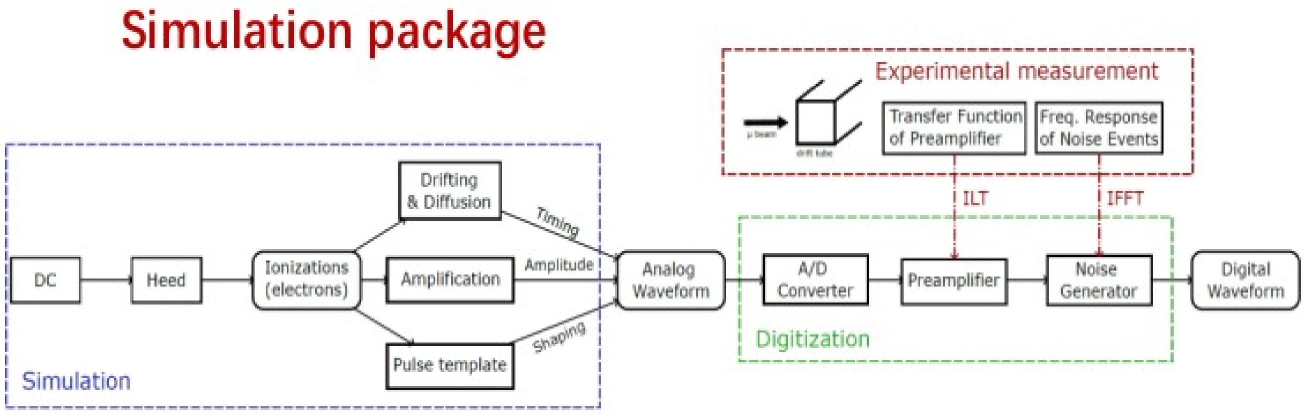
- Muon particles is passed through mixture of gas having 90% He and 10% Isobutane C_4H_{10} and create electron & ion pair drift toward their opposite polarity and generate induce current

- The LSTM and CNN model were trained on the mometa ranges from 0.2 to 20 GeV/C and then we applied this trained model on the sample of 2, 4, 6, 8, and 10 GeV/c momenta as testing to check the performance of the models

- Following the simulation in Garfield++, I proceeded to plot various results for the study of the cluster counting techniques

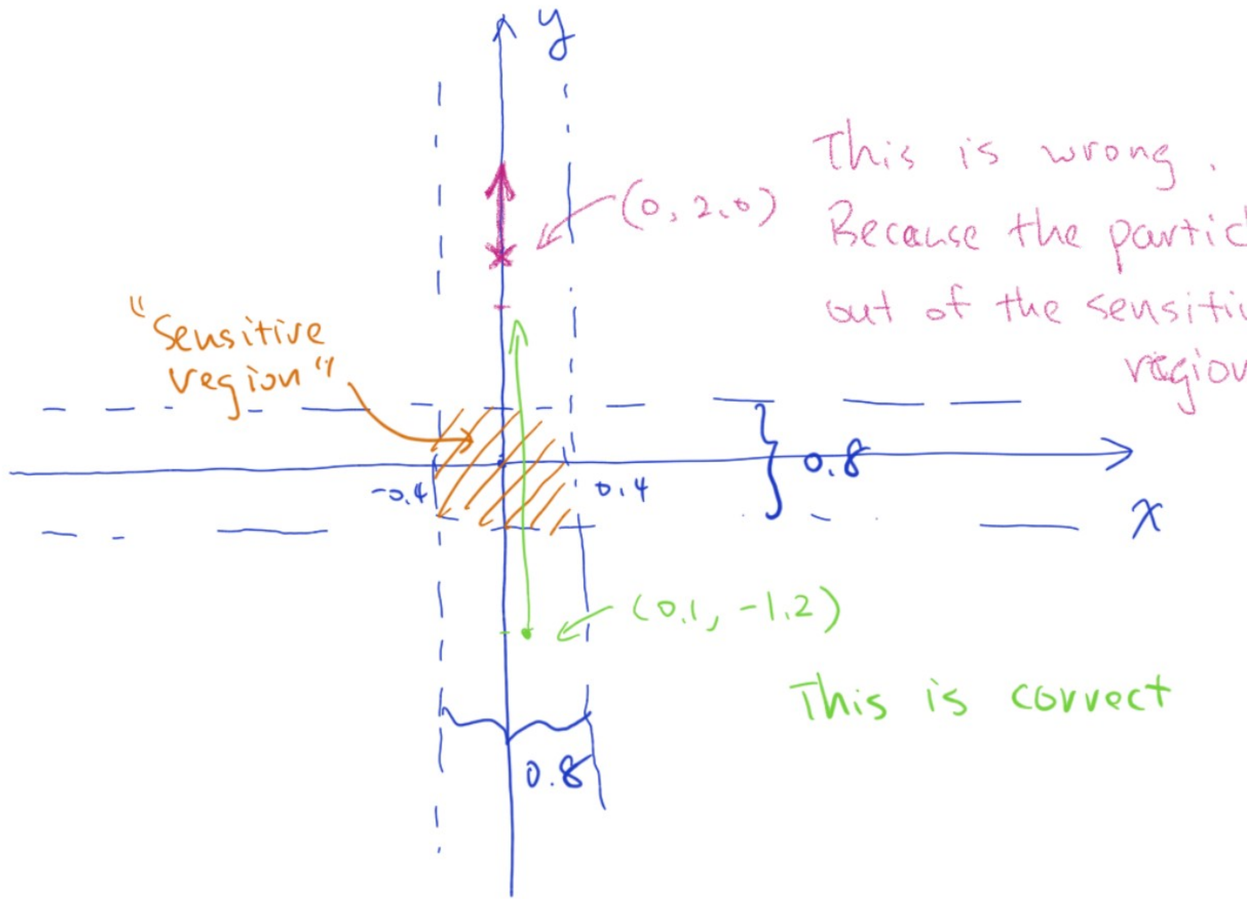
- The simulation package creates analog induced current waveforms from ionizations (HEED). The digitization package incorporates electronics responses taken from experimental measurements and generates realistic digital waveforms





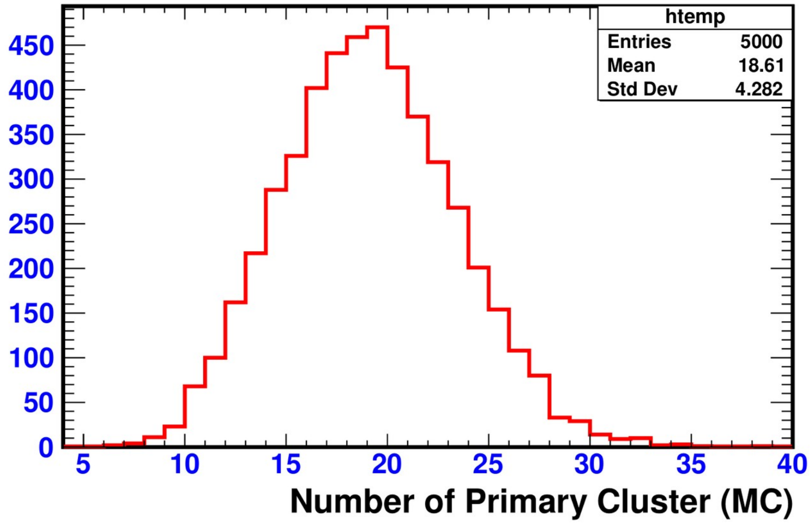
Simulation Parameters Based on Garfield ++

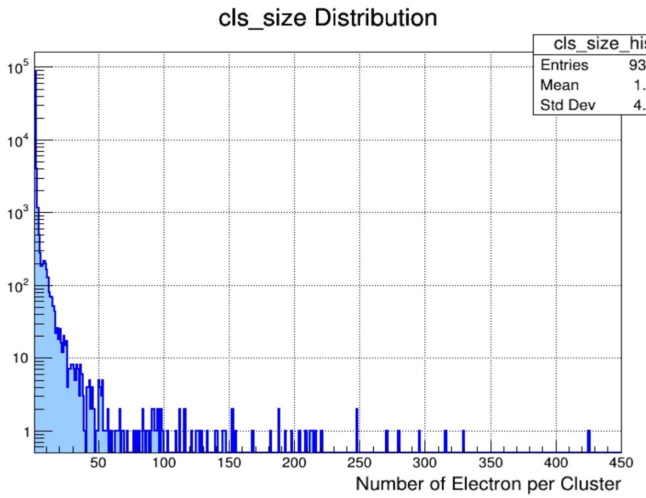
Sampling Rate	2 GHz
Gas Mixture	He (90%) & C_4H_{10} (10%)
Cell Size	0.8 cm
Momentum (GeV/C)	10, 8, 6, 4, 2
Angle between the z axis of drift tube chamber and track of the muon particle	45 degree
Particle	Muon

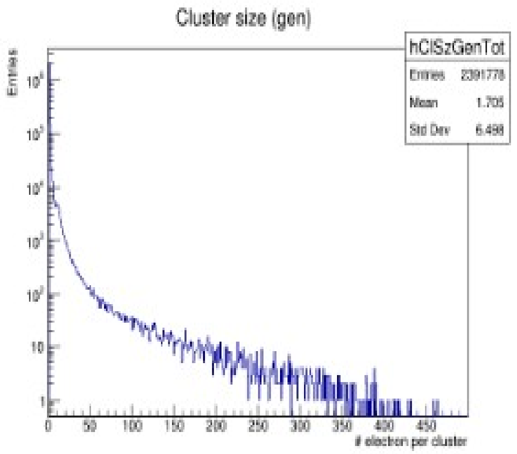


- ***All the simulation parameters like cell size, different momentum, gas mixture etc are shown in the table***

Simulation Parameters Based on Garfield ++ for 10 GeV

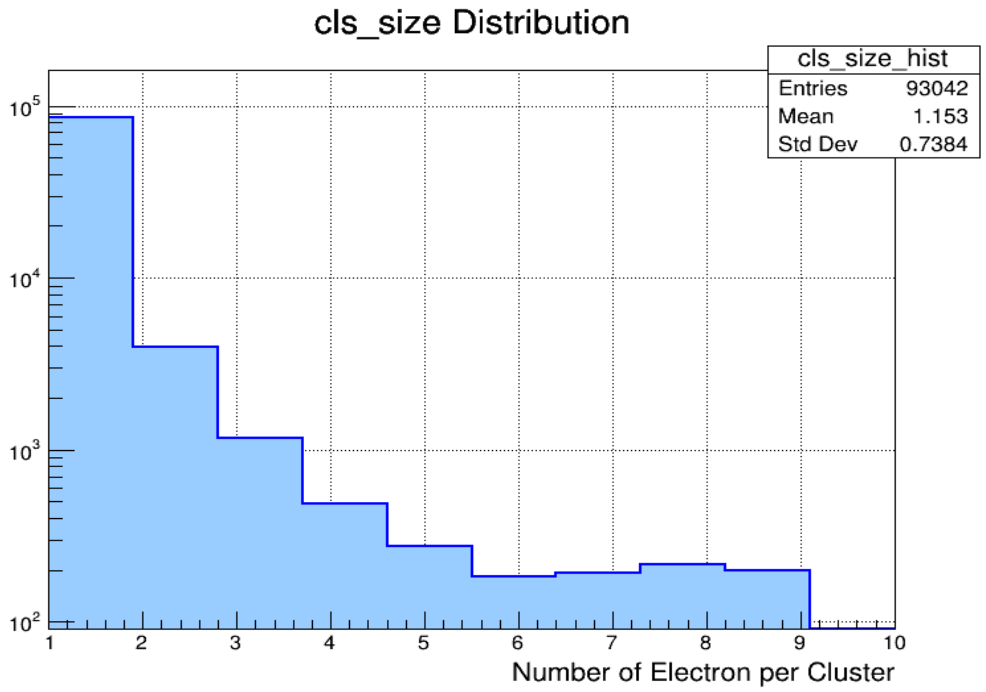


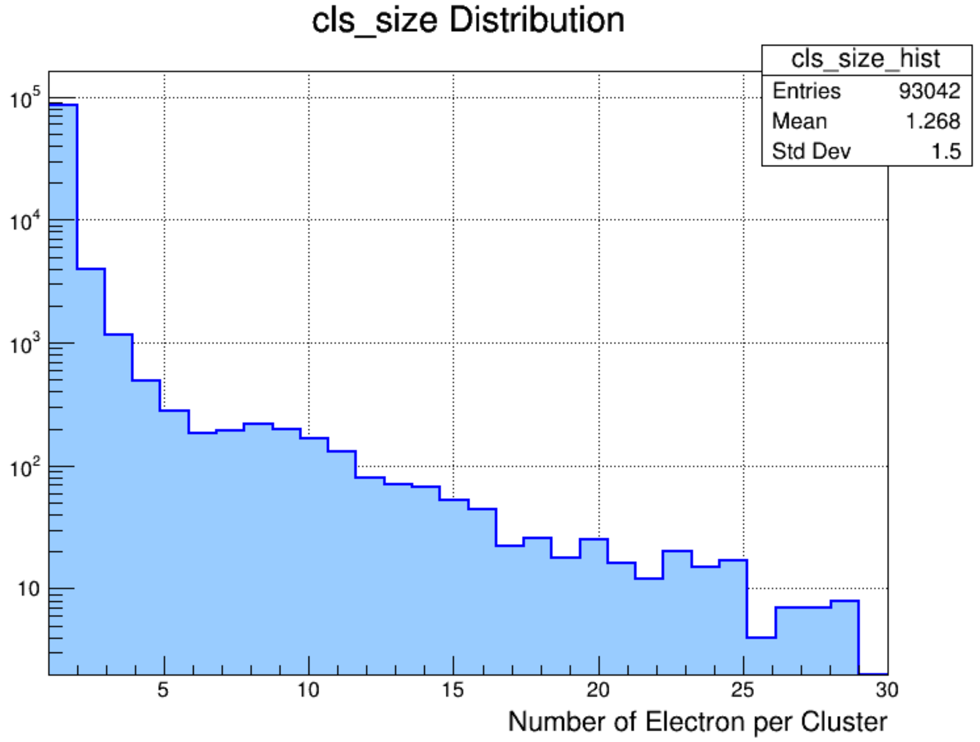




- ***The above distrubution shows the number of primary clusters with mean value 18.61 for 5000 tracks***
- ***The above distrubution shows the number of electrons per clusters with mean value 1.423***
- ***I took this plot as a refrence from the resarch paper of "Simulation of particle identification with the cluster counting technique"***

Simulation Parameters Based on Garfield ++ for 10 GeV

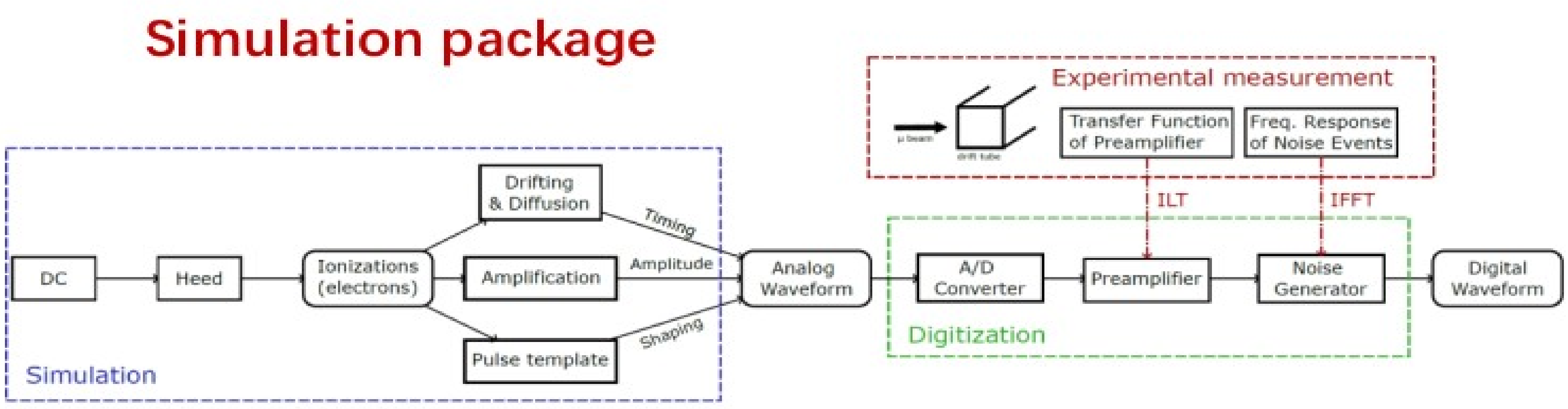


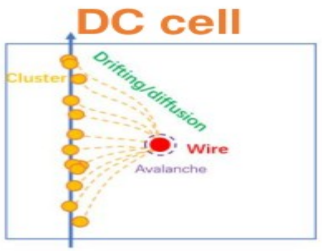


- *The above distrubution shows the number of electrons per clusters with mean value 1.153 in the range from 0-10. of course the mean value would be change because we exclude bins (number of electrons per clusters)*

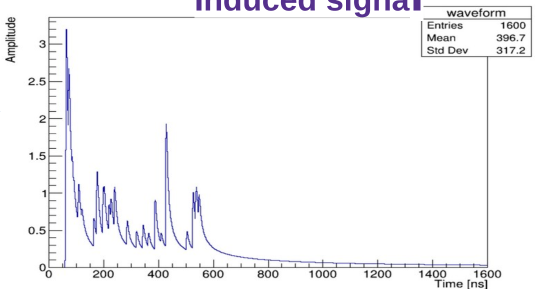
- *The above distrubution shows the number of electrons per clusters with mean value 1.268 in the range from 0- 30. of course the mean value would be change because we exclude bins (number of electrons per clusters)*

Waveform-Based Full Simulation

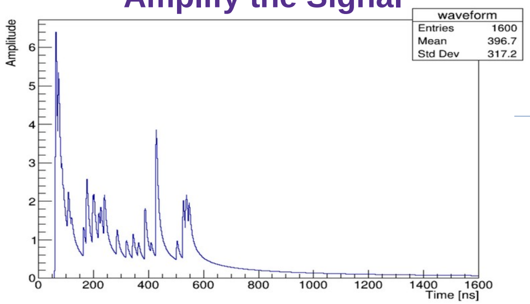




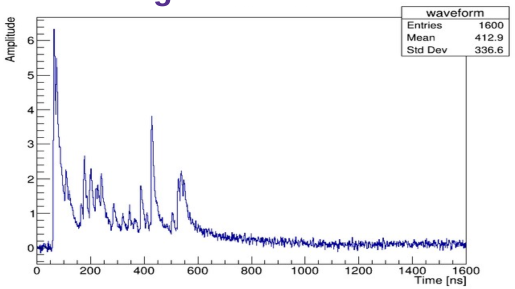
Induced signal



Amplify the Signal



Digital Waveform

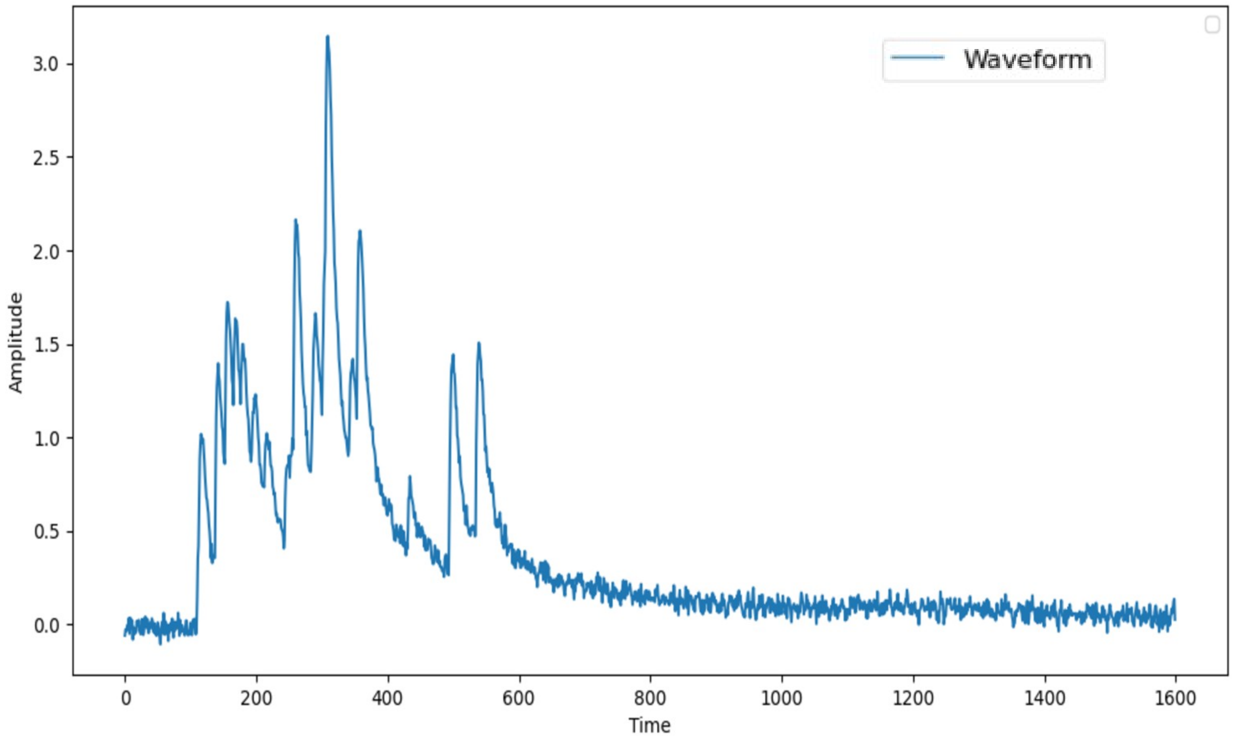


Generation of the Waveform

- A sophisticated simulation framework has been developed to create realistic waveforms for cluster counting, consisting of two main parts: simulation and digitization

- The simulation package generates analog waveforms from ionizations (electrons) caused by relativistic charged particles using Heed, while the digitization package converts these analog waveforms into digital waveforms to mimic real electronic responses

- This conversion involves calculating the pre-amplifier's response using an inverse Laplace transform, combining it with the analog waveform, and adding electronic noise from experimental data using a fast Fourier transform



Training LSTM and CNN Model for Two-Step Reconstruction Algorithm

First Model

Training of Long Short Term Memory (LSTM)

Model Using HPC Resources

Optimization of Hyperparameters for Long Short-Term Memory Models Using HPC Resources

- **Currently, I designed a task involving the simultaneous submission of several jobs using local HPC Resources**
- **The purpose of this task is to train Long Short-Term Memory (LSTM) models to classify signals rom background, a process known as a classification task.**
- **To achieve this task, I utilized various hyperparameters, including activation functions, optimizer Epochs, batch size, patience, and dropout rates etc**
- **Additionally, I managed different resources such as memory requests, Job duration, and CPU Usage etc**
- **Then, I selected the best model based on evaluation metrics such as the highest AUC value among all configurations**

```
# Arrays defining different configurations
vminimizer=("sgd" "rmsprop" "adam")
vneuron=("relu relu" "relu sigmoid"  "selu sigmoid")
vpatiences=("30")
vbatches=("150" "250")
vtopologies=("96 128 1" "32 64 1" "16 32 1" "8 16 1")
dropout=("0.0"  "0.1" "0.2")
vepochs=(100 200)
```

Different hyperparameters for LSTM peak finding model are shown in the screen shot

Criteria to Select Best Long Short Term Memory Model by Using HPC Resources

AUC Score	**0.98054**

- **I selected best long short term memory (LSTM) model based on the highest Area under the curve value among all the configurations**

- **The above table shows us the highest value of Area under the curve to choose best LSTM model among all configurations**

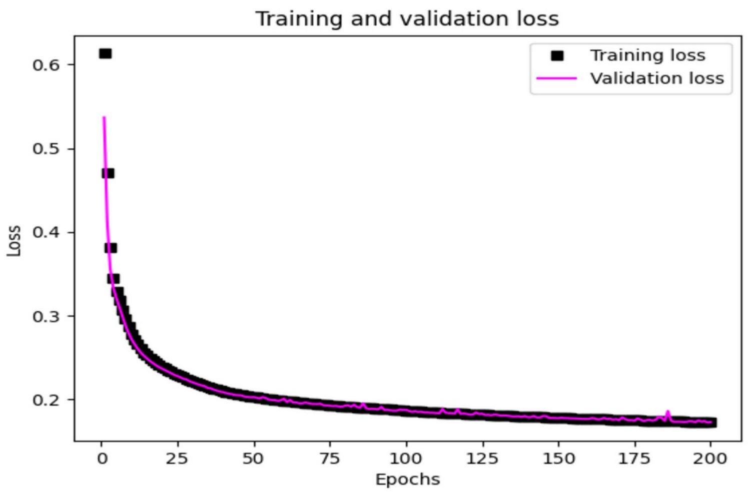
```
number of signal = 113768, number of background = 3886382
(125144, 15, 1)
(102392, 15, 1)
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 15, 32)            4352
_________________________________________________________________
flatten (Flatten)            (None, 480)               0
_________________________________________________________________
dense (Dense)                (None, 32)                15392
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 33
=================================================================
Total params: 19,777
Trainable params: 19,777
Non-trainable params: 0
```

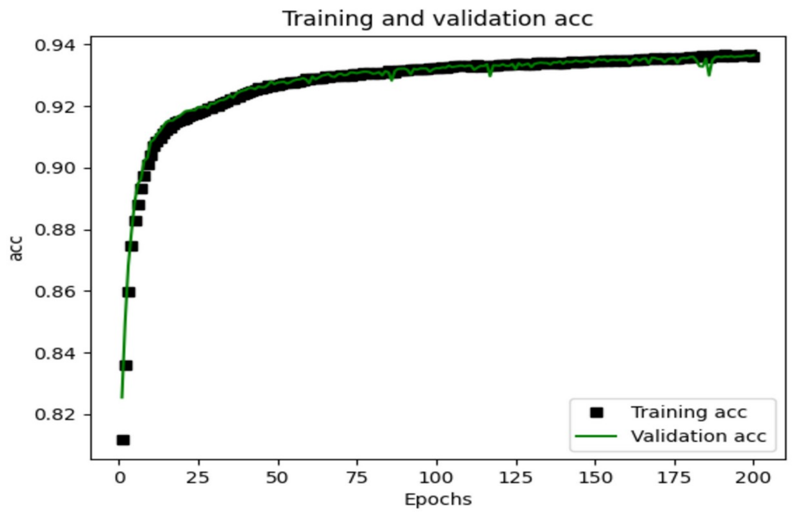
- **The above snapchat shows us the structure of best LSTM Peak Finding model**

Partionable Resources	Usage	Request
CPUS	1.2	4
Memory (MB)	968	5000
Run Remote Usage	1min 35sec	2hr/job

- **The above table shows us different HPC Local Resources of the RECAS like CPUS, Memory Usage and Run remote Usage**

Plots of the Best Peak Finding LSTM Model





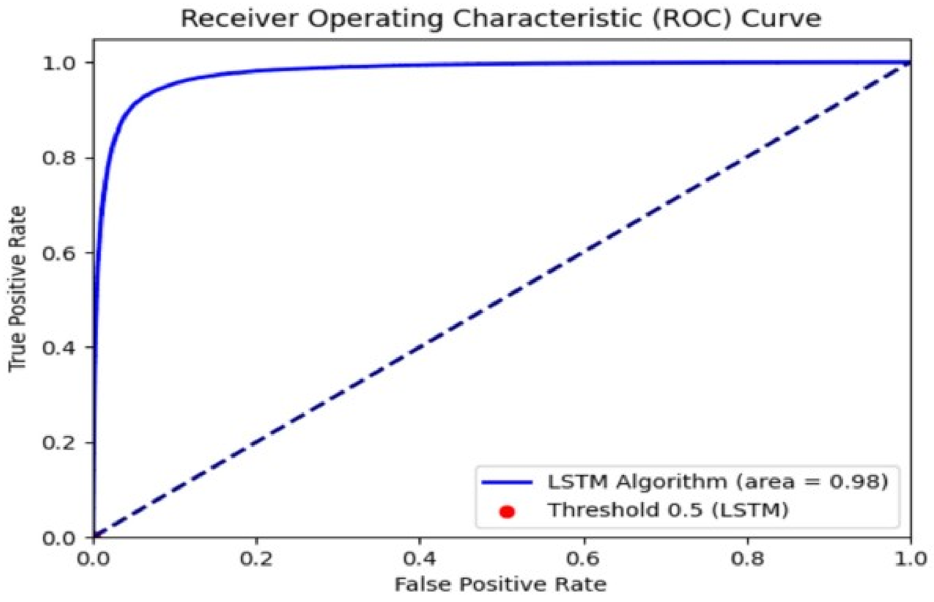
Optimizer	sgd
Topology	[32 32 1]
Bach size	[250]
Number of Epochs	200
Activation function	Relu, sigmoid
Train/ Validation Split	0.7
Patience/Early Stopping call	30

- **The upper left sided plot loss VS epoch show us that the training and validation loss decreases over the epochs and then it become approximately constant which shows a best trained model**

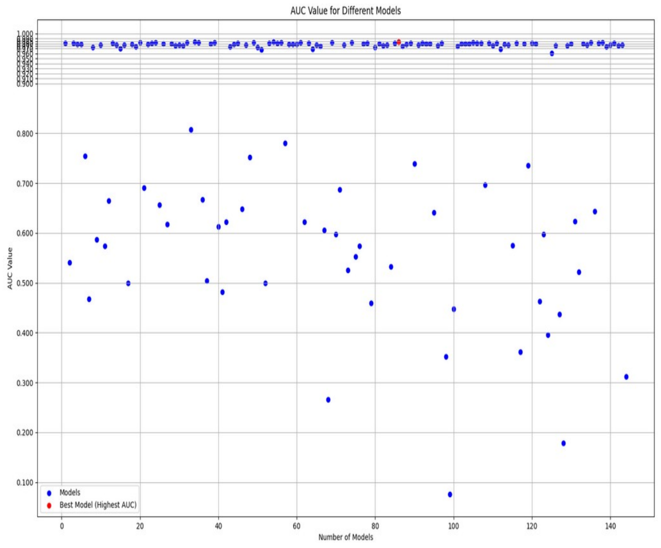
- **The upper right sided plots Accuracy VS Epoch show us that the training and validation accuracy increases over the epochs and then it become approximately constant which shows a best trained model**

- **Different Hyperparameters for the trained LSTM Model are shown in the Table**

Plots of the Peak Finding LSTM Model



		Prediction	
		Sig	Noise
Truth	Sig	TP	FN
	Noise	FP	TN



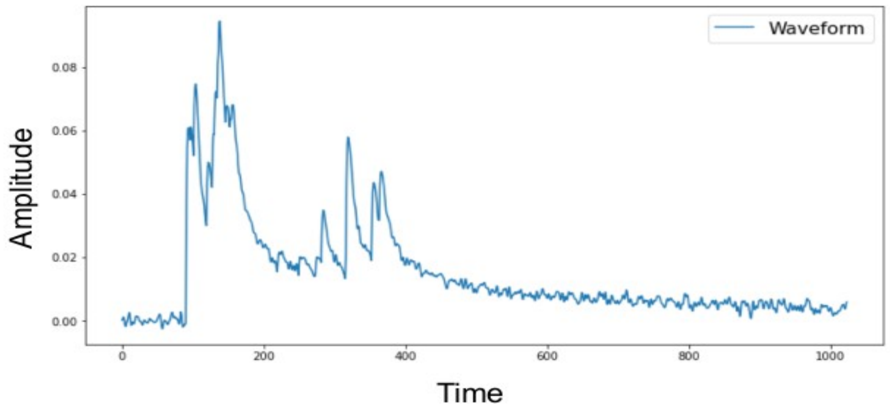
I

- The above plot show ROC curve for the LSTM model with Area under the curve value 0.98 which show a best classification to discrimate sugnal from background

- The above table tell us about the concept of classification (TP, TN) and misclassification (FP, FN)

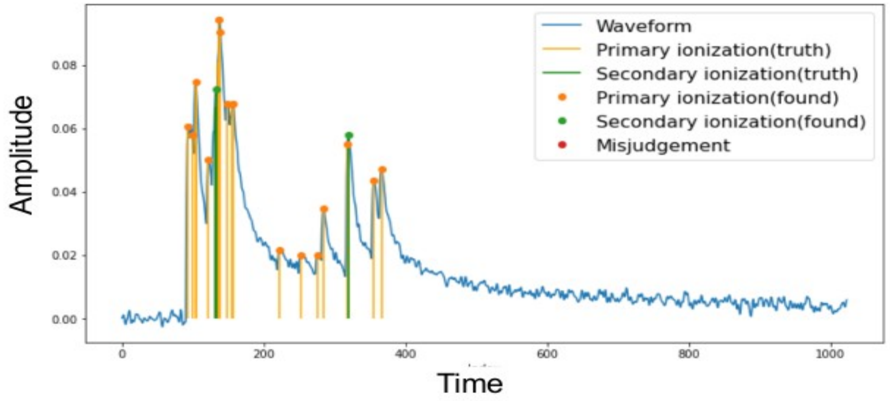
- **The above plot shows us different configuration models with Area under the curve value.The red dot shows us the best model among all**

Two Step Reconstruction Algorithm



Step1. Peak Finding

Discriminate peaks (both primary and secondary) from the noises (classification problem)



Step2. Clusterization:

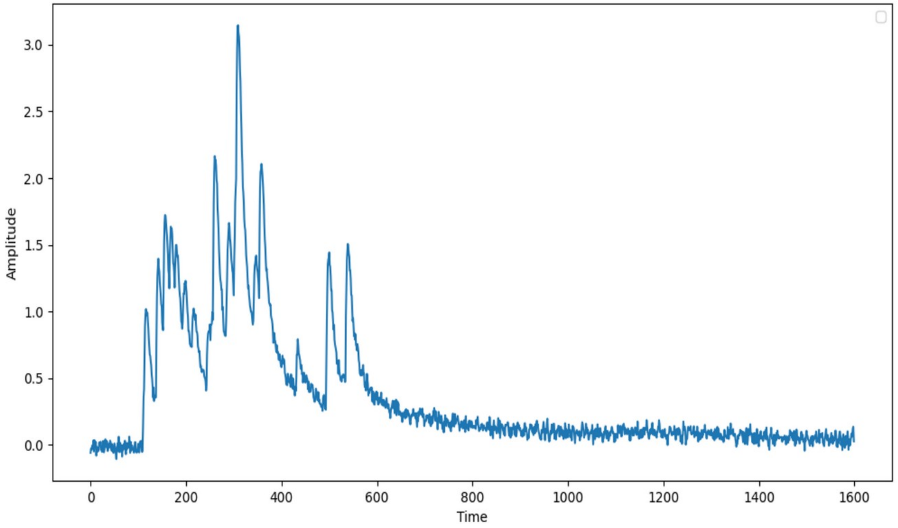
Determine the number of clusters (N_{cls}) from the detected peaks (regression problem)

- **Taken from the Guang presentation just to know about what are the main steps of our algorithm in cluster counting Techniques**

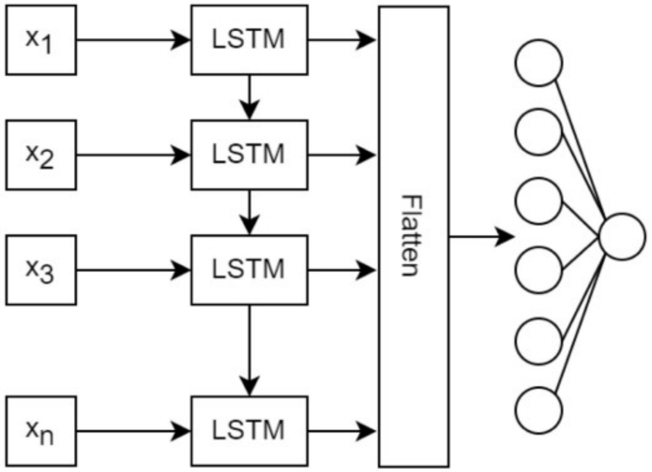
Applying LSTM Model for Peak Finding Algorithm

Two-Step Reconstruction Algorithm

Waveform

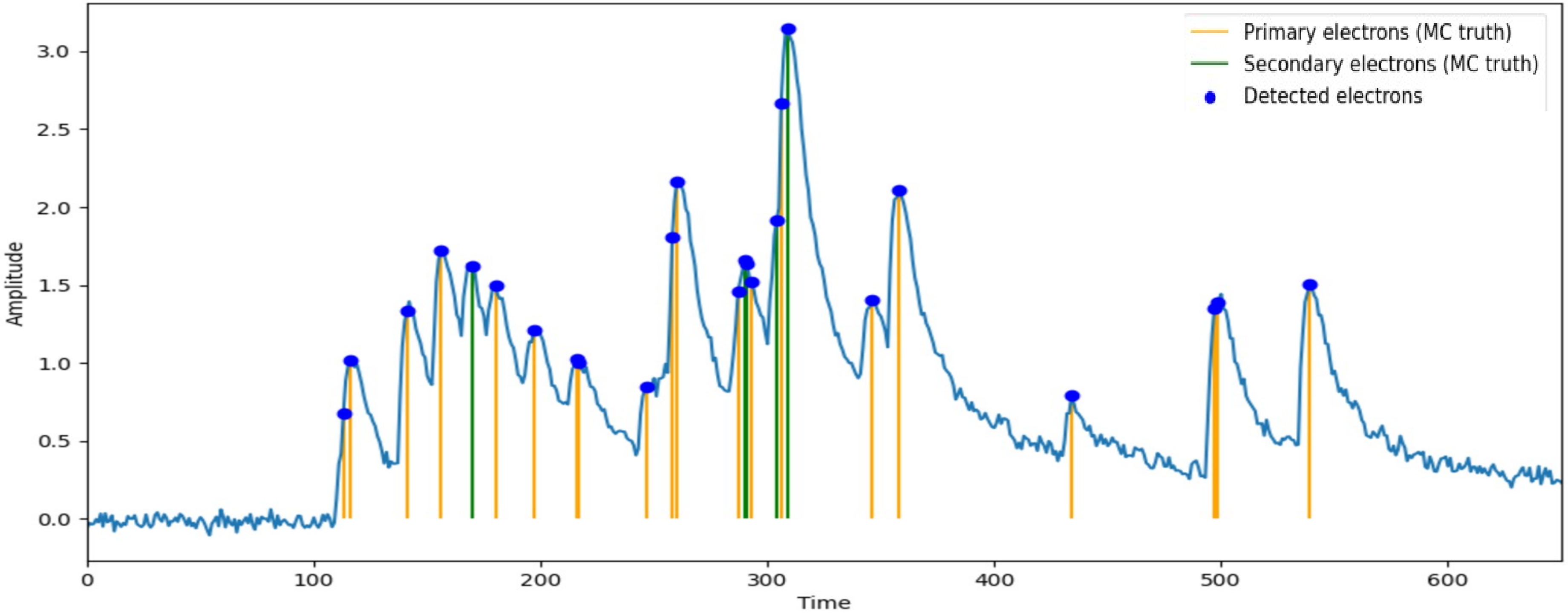


Step1: Peak Finding



- The task of peak finding can be framed as a classification problem in machine learning
- The waveforms are divided into segments, each comprising 15 bins. Each segment can represent either a signal or a nois
- The list of the amplitudes of a segment, subtracted by their mean and normalized by their standard deviation, is served as the input feature for the neural network
- The data of waveform is time sequence data, which suitable for especially Long Short Term Memory Model

Evaluation by Waveform and Total Detected Peaks by LSTM



- We applied a Long Short-Term Memory (LSTM) model to the waveform to classify signals (primary and secondary electrons) from the Noise using a peak-finding algorithm known as classification
- Detected peaks from both primary and secondary electrons are shown by blue dots

Second Model

Training of Convolutional Neural Network(CNN) Model Using HPC Resources

Optimization of Hyperparameters for Convolutional Neural Network(CNN) Model Using HPC Resources

- **Currently, I designed again task involving the simultaneous submission of several jobs using local HPC Resources**
- **The purpose of this task is to train convolutional neural network models to detect number of primary ionization clusters based on the detected peaks, a process known as a regression task**
- **To achieve this task, I utilized various hyperparameters, including activation functions, optimizer Epochs, batch size, patience, and dropout rates etc**
- **Additionally, I managed different resources such as memory requests, Job duration, and CPU Usage etc**
- **Then, I selected the best model based on evaluation metric such as the mean square error (mse)**

```
# Arrays defining different configurations
vminimizer=("rmsprop" "sgd" "Adam")
vneuron=("relu selu" "selu selu" "relu relu")
vpatiences=("30")
vbatches=("150")
vtopologies=("32 16" "16 32" "32 64" "8 16")
dropout=("0.1" "0.0")
vepochs=(50)
```

Different hyperparameters for CNN clusterization model are shown in the screen shot

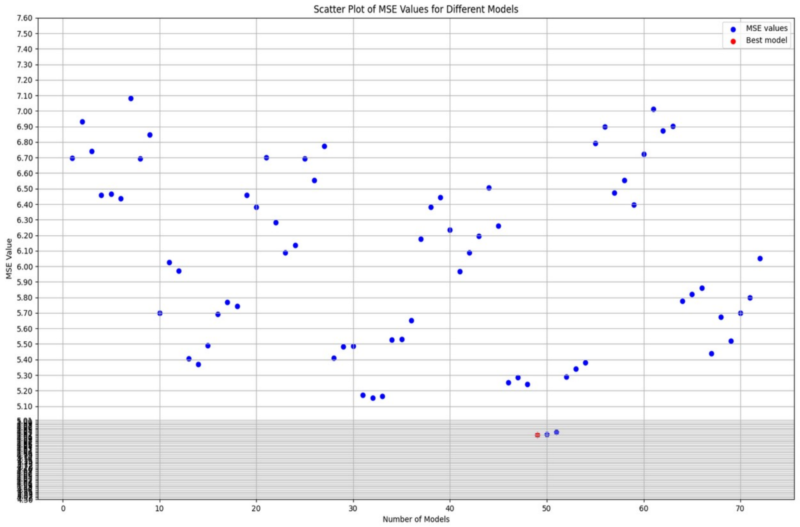
Criteria to Select Best CNN Model Based on the Lowest Mean Absolute Error (MAE)

Mean Square Error (MSE)	4.9148

- **I selected best CNN model based on the lowest mean square error (MSE) value among all the configuration**

- **The above table shows us the value of different evaluation metrics to choose best CNN model among all configurations**

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

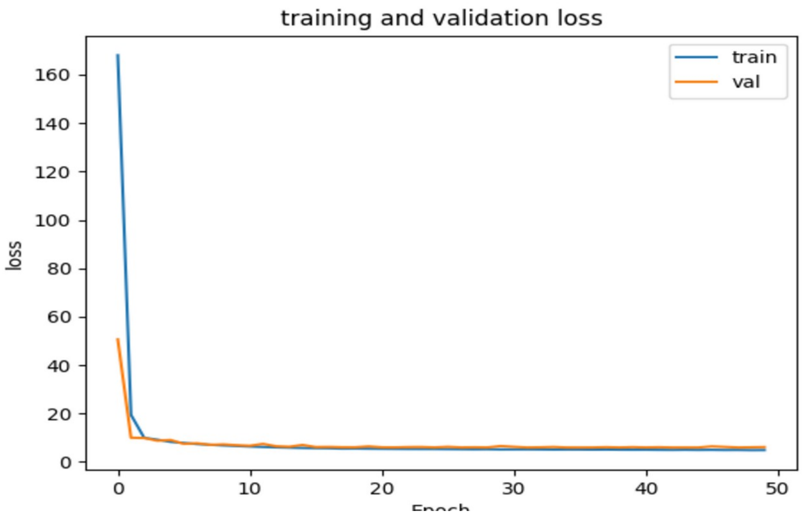


- **The above plot shows us different configuration models with Mean Absolute error value. The red dot shows us the best model among all**

Partionable Resources	Usage	Request
CPUS	2.55	4
Memory (MB)	146	5000
Run Remote Usage	43 sec	2hr/job

- **The above table shows us different HPC Local Resources of the RECAS like CPUS, Memory Usage and Run remote Usage**

Best CNN Regression Model



- The plots show us that the training and validation loss mean square error decreases over the epochs and then it become constant which show us the best result

Optimizer	Rmsprop
Number of Filters	[32 64]
Filter Size	4
Bach size	[150]
Number of Epochs	50
Activation function	selu, selu
Train/Validation Split	0.7
neurons	[32, 1]

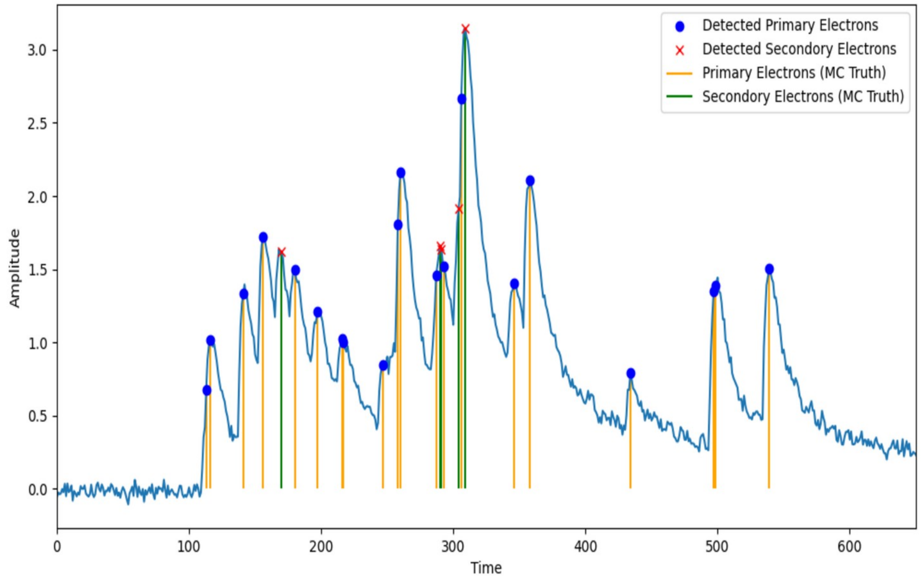
- **Different Hyperparameters for the trained LSTM Model are shown in the Table**

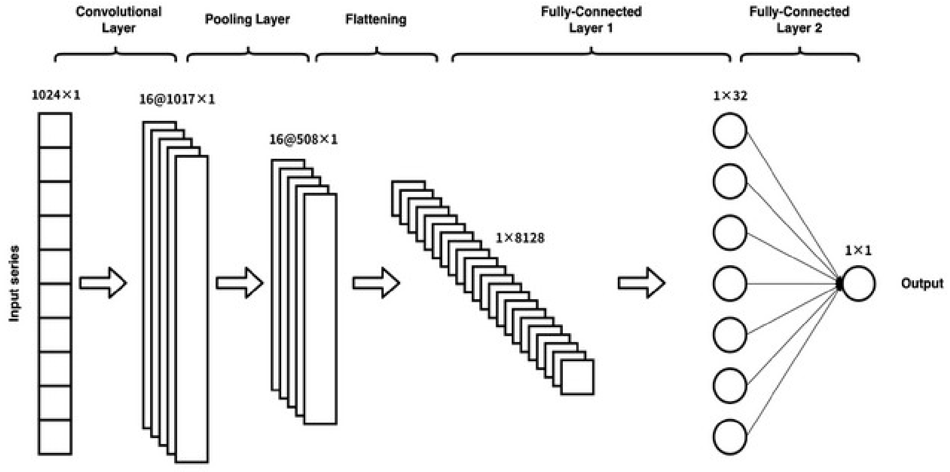
```
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 1021, 32)          160
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 510, 32)           0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 507, 64)           8256
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 253, 64)           0
_________________________________________________________________
flatten (Flatten)            (None, 16192)             0
_________________________________________________________________
dense (Dense)                (None, 32)                518176
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 33
=================================================================
Total params: 526,625
Trainable params: 526,625
Non-trainable params: 0
```

Structure of Best Trained CNN Model in the snapchat

Applying CNN Model for Clusterization Algorithm

Step2: Clusterization





- A regression problem to predict Number of primary clusters based on the detected peaks by using Convolutional Neural Network (CNN) model

- The peaks found by peak finding Algorithm would be training sample of this algorithm

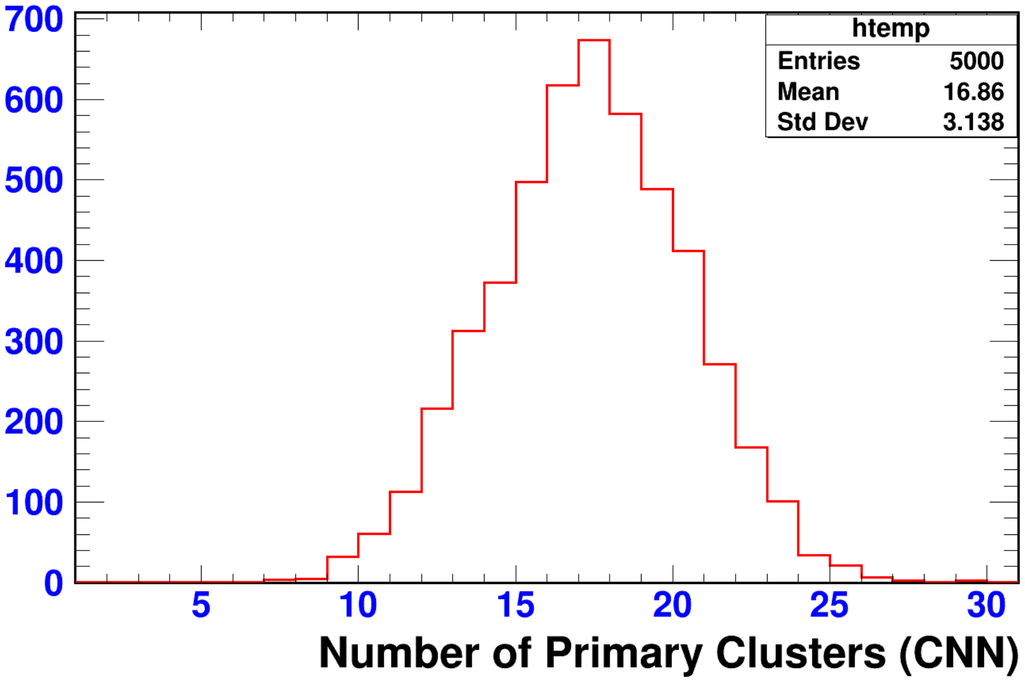
- Labels: Number of clusters from MC truth

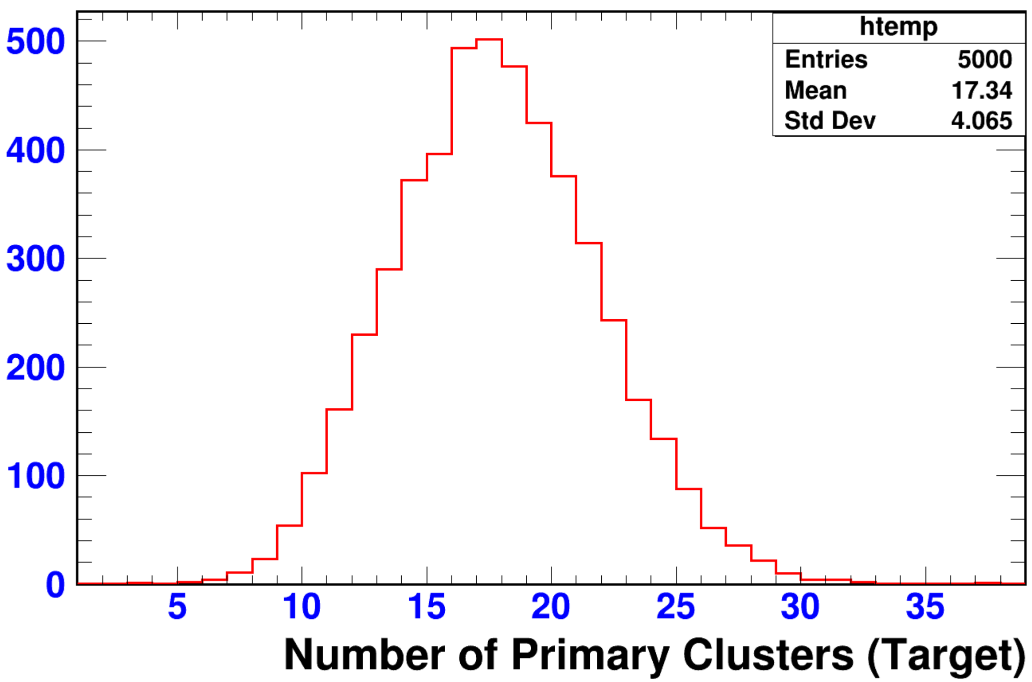
- Features: Time list of the detected times in the previous step encoding in an (1024, 1) array.

- A regression problem

Applying Best CNN Model for Clusterization Algorithm

Final results of the reconstruction for 10 GeV

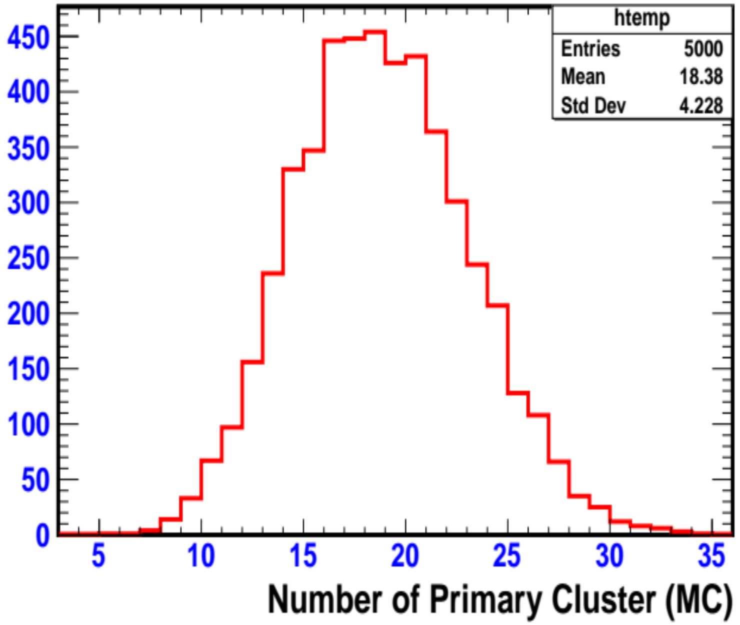


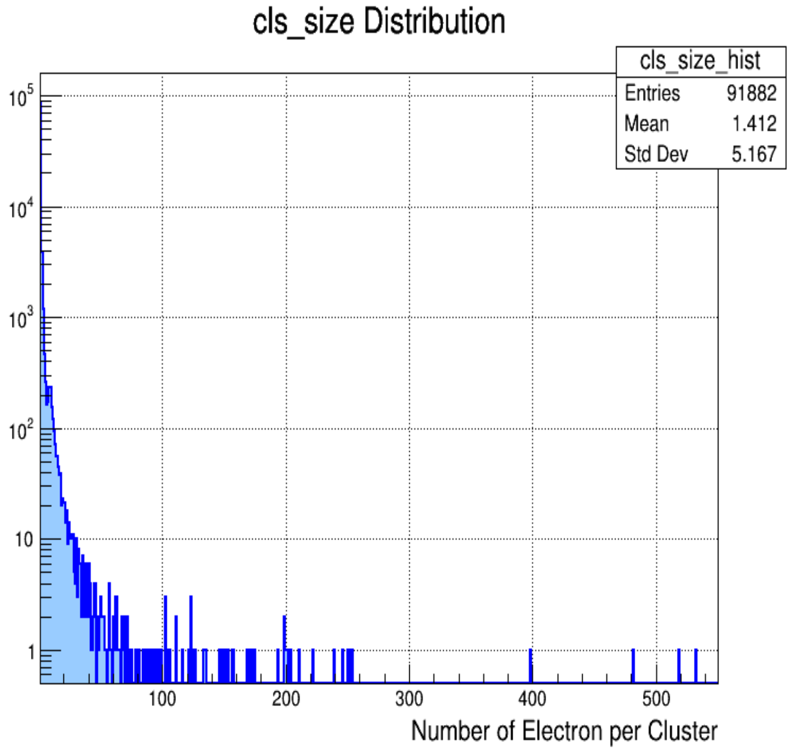


- Number of Primary clusters with mean value (16.86) detected by CNN Model based on the detected primary peaks with mean value (17.34)

Repeating the Above Process for 8 GeV, 6 GeV, 4 GeV and 2 Gev momenta

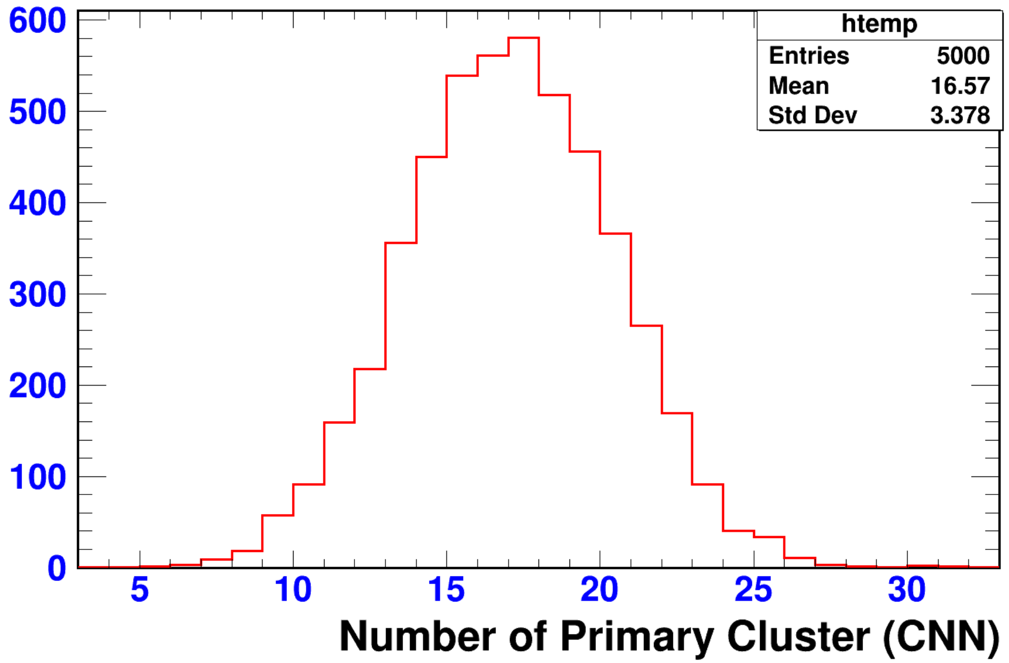
Simulation Parameters Based on Garfield ++ for 8 GeV

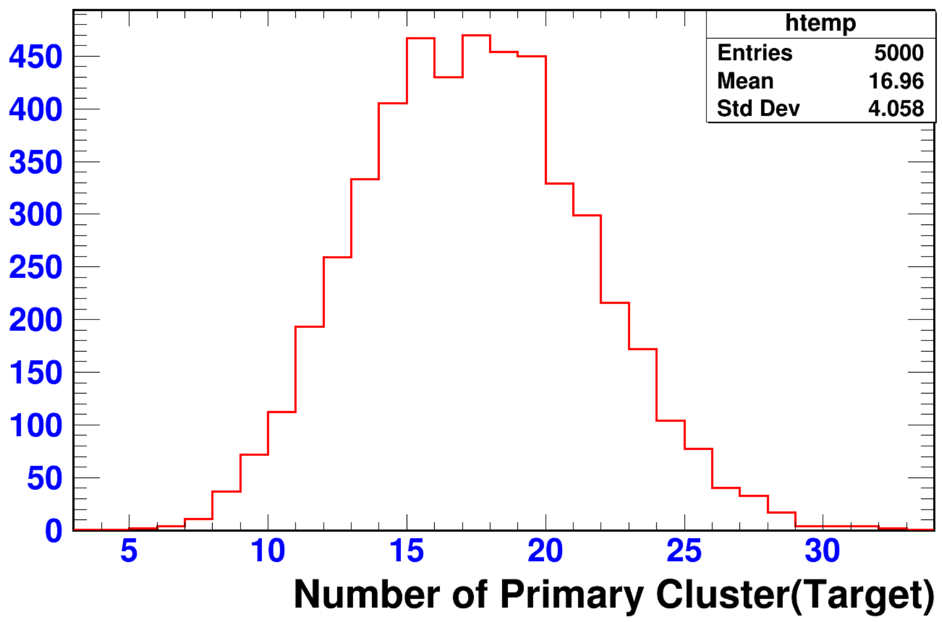




- ***The above distrubution shows the number of primary clusters with mean value 18.38 for 5000 tracks***
- ***The above distrubution shows the number of electrons per clusters with mean value 1.412***

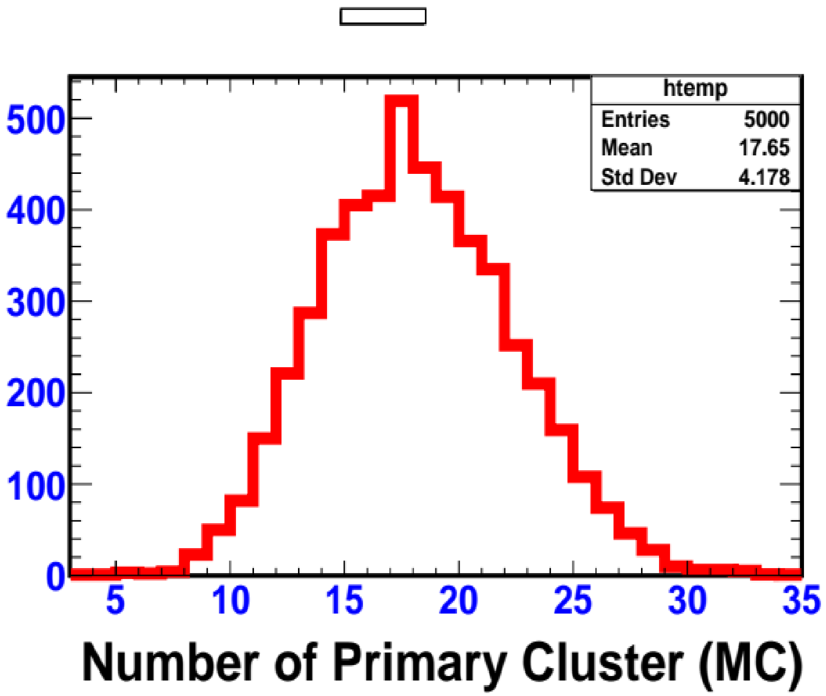
Final results of the reconstruction for 8 GeV

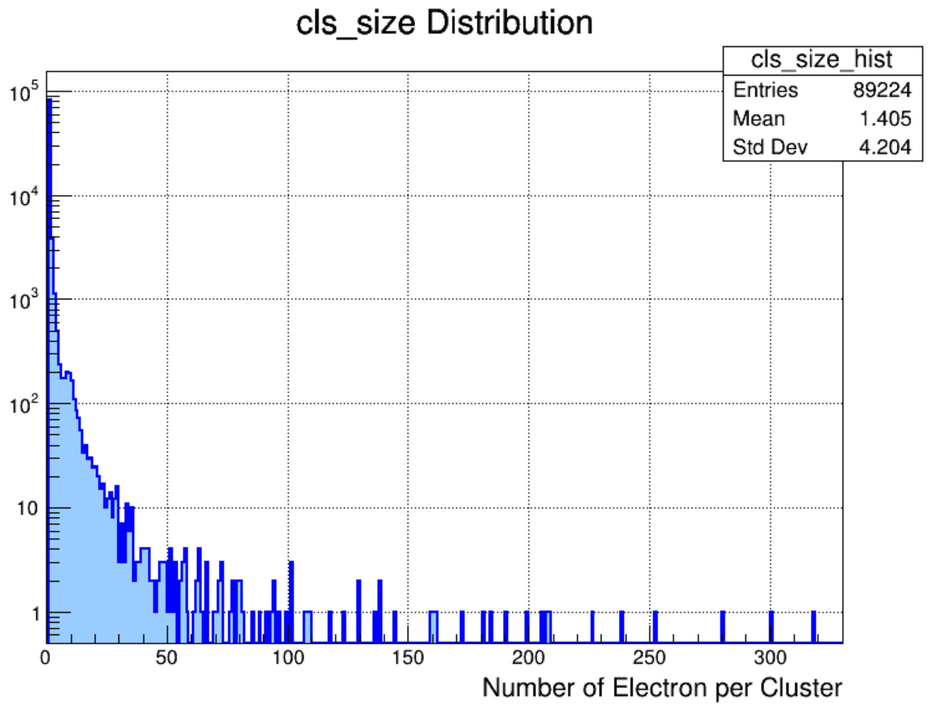




- **Number of Primary clusters with mean value (16.57) detected by CNN Model based on the detected primary peaks with mean value (16.96)**

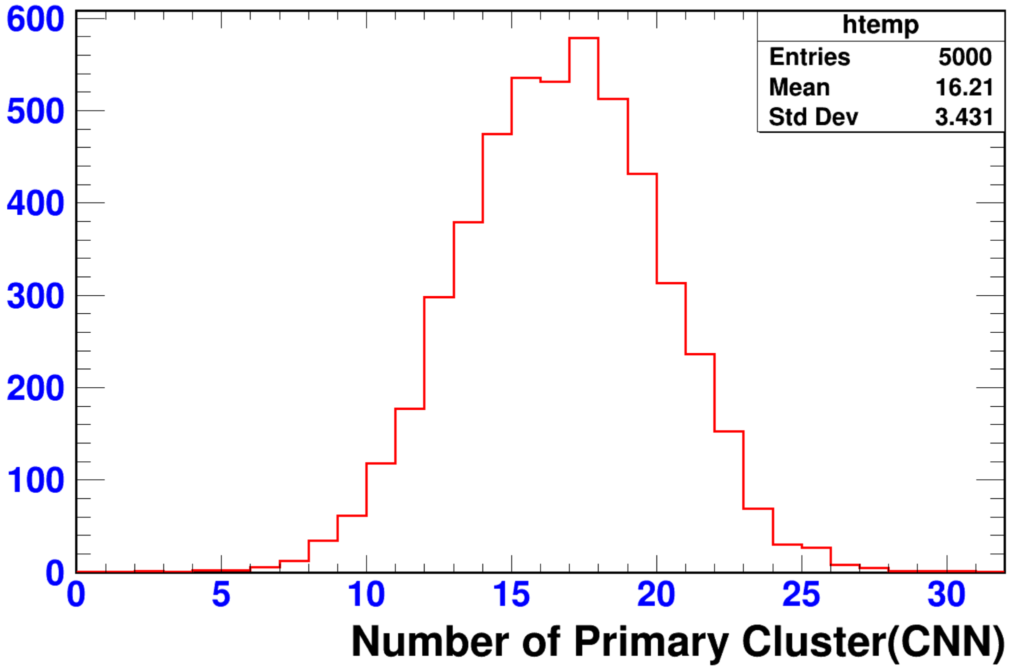
Simulation Parameters Based on Garfield ++ for 6 GeV

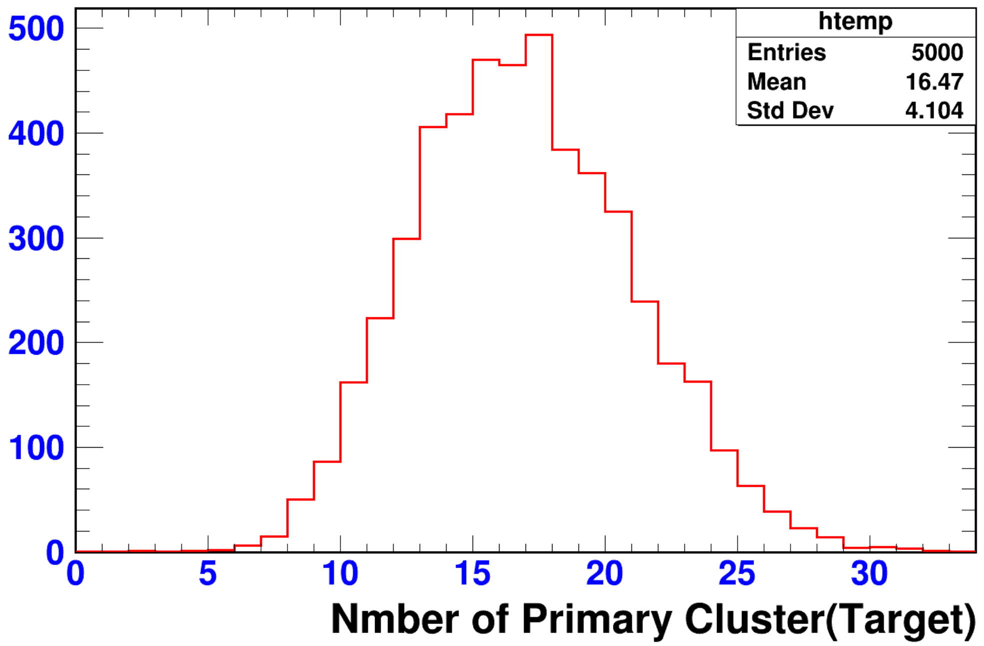




- ***The above distrubution shows the number of primary clusters with mean value 17.65 for 5000 tracks***
- ***The above distrubution shows the number of electrons per clusters with mean value 1.405***

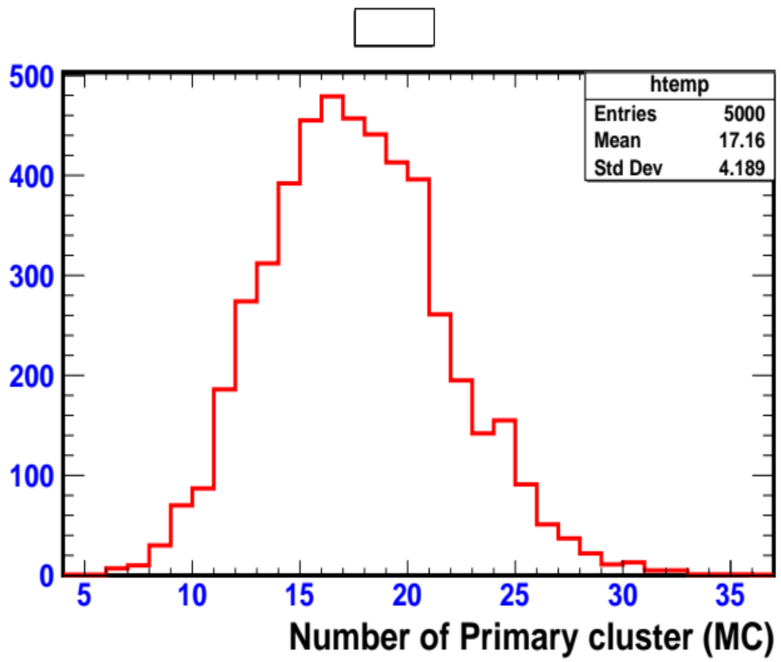
Final results of the reconstruction for 6 GeV

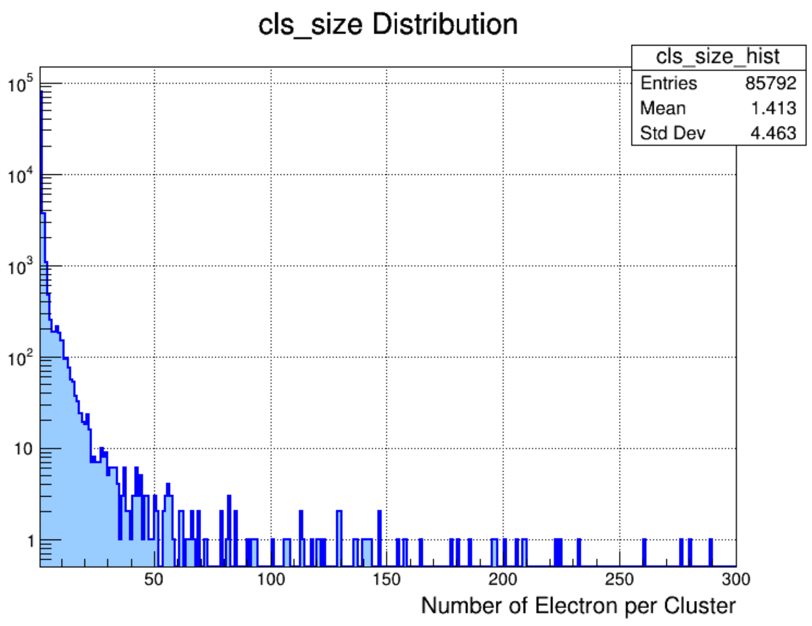




- Number of Primary clusters with mean value (16.21) detected by CNN Model based on the detected primary peaks with mean value (16.47)

Simulation Parameters Based on Garfield ++ for 4 GeV

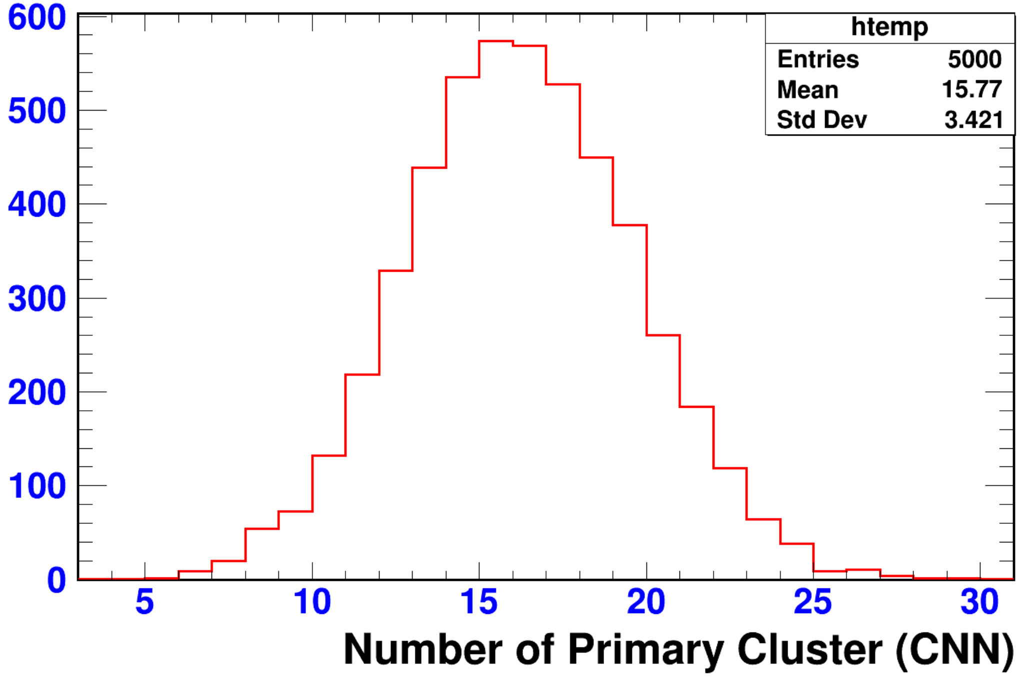


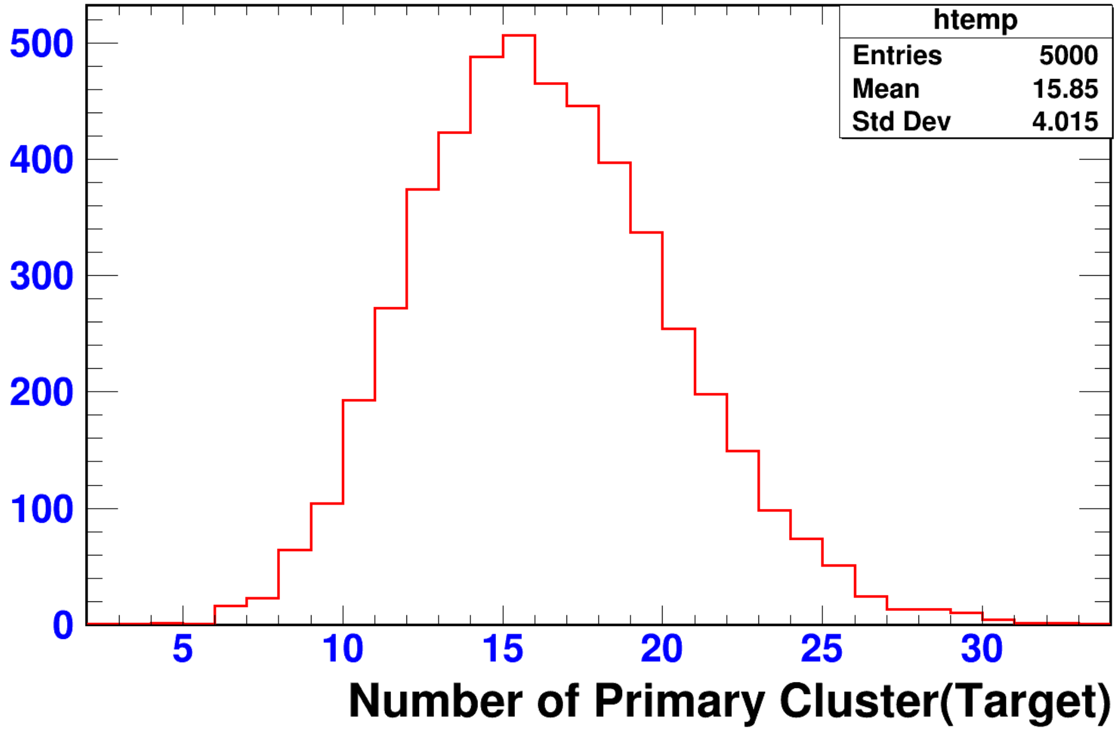


- *The above distrubution shows the number of primary clusters with mean value 17.16 for 5000 tracks*

- *The above distrubution shows the number of electrons per clusters with mean value 1.413*

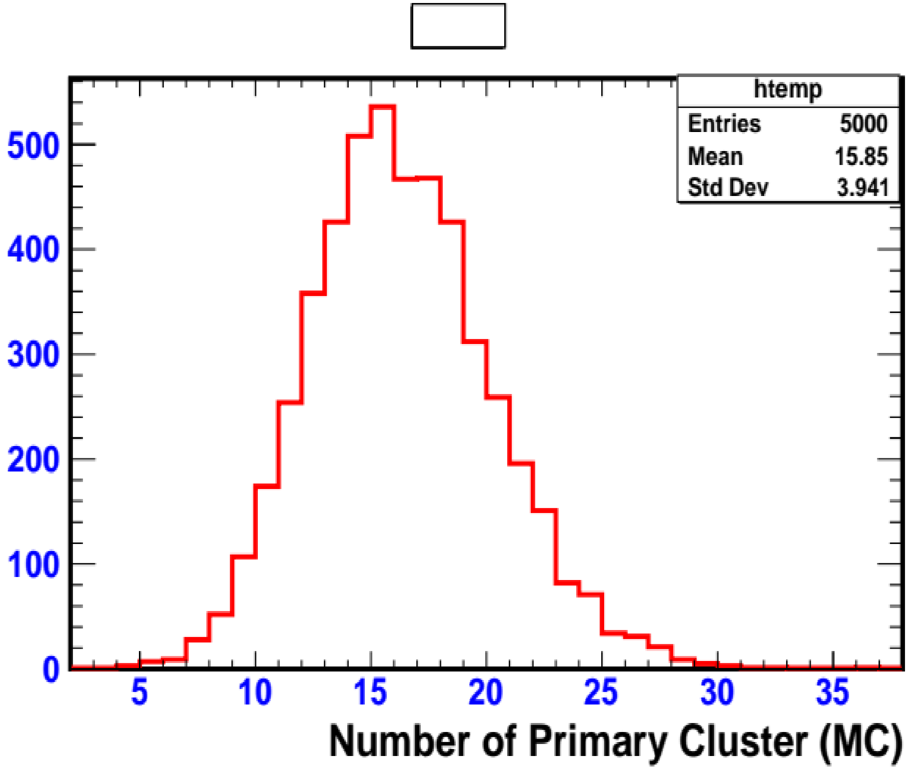
Final results of the reconstruction for 4 GeV

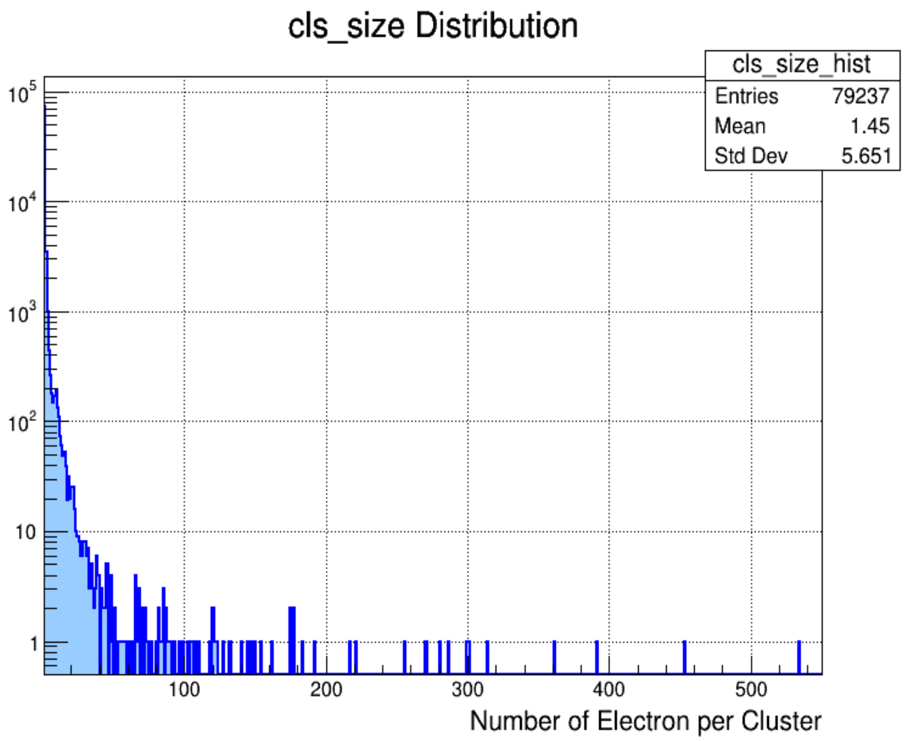




- **Number of Primary clusters with mean value (15.77) detected by CNN Model based on the detected primary peaks with mean value (15.85)**

Simulation Parameters Based on Garfield ++ for 2 GeV

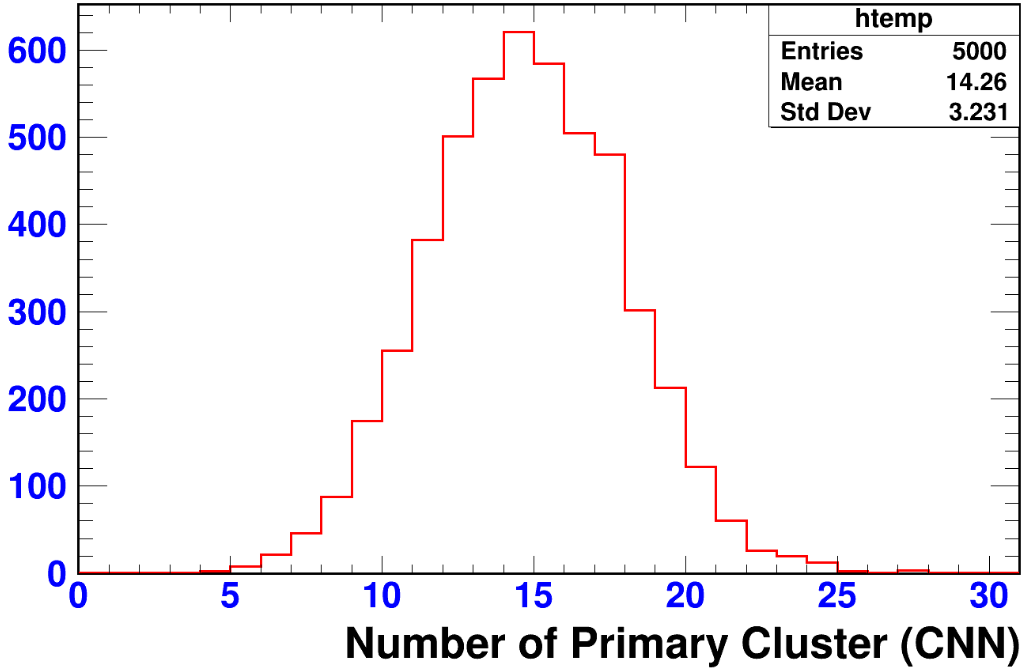


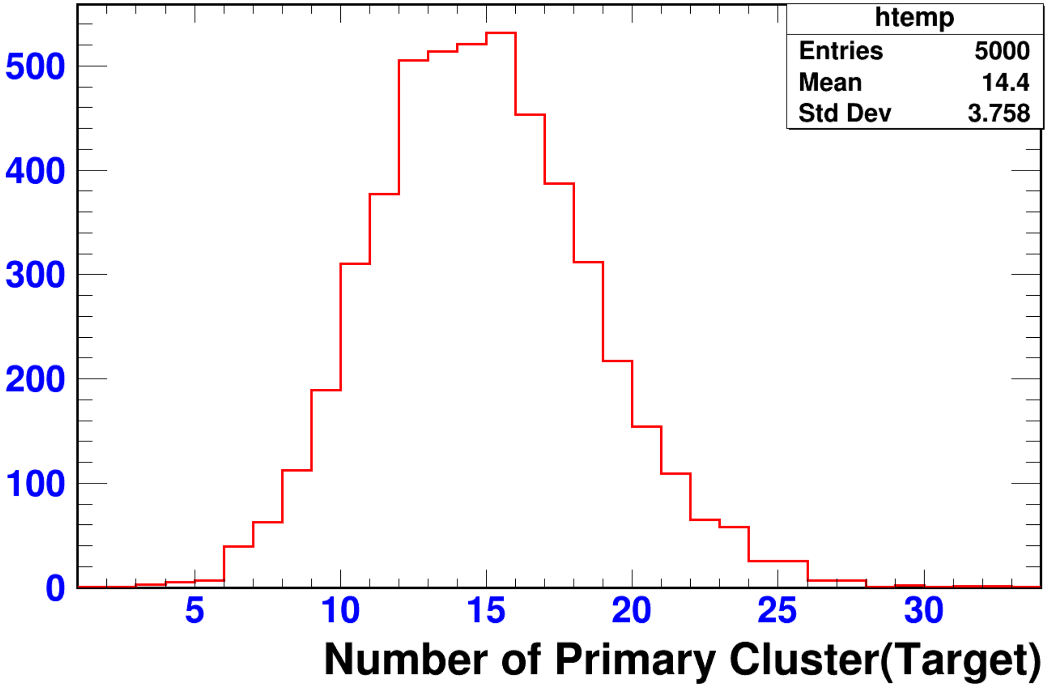


- ***The above distrubution shows the number of primary clusters with mean value 15.85 for 5000 tracks***

- ***The above distrubution shows the number of electrons per clusters with mean value 1.45***

Final results of the reconstruction for 2 GeV





- **Number of Primary clusters with mean value (14.26) detected by CNN Model based on the detected primary peaks with mean value (14.4)**

Conclusion of Final Result of Reconstruction based for different Momenta

Momentum of Muon	Primary Cluster(MC)	Standard Deviation (MC)	Cluster Size (Full Range)	Primary Cluster(LSTM	Standard Deviation (LSTM)	Primary Cluster (CNN)	Standard Deviation (CNN)
2 Gev/c	15.85	3.9	1.55	14.4	3.75	14.26	3.2
4 GeV/c	17.16	4.189	1.54	15.85	4.015	15.77	3.42
6 GeV/c	17.65	4.178	1.605	16.47	4.104	16.21	3.43
8 GeV/c	18.38	4.228	1.54	16.96	4.05	16.57	3.37
10 Gev/c	18.61	4.282	1.54	17.34	4.065	16.86	3.13

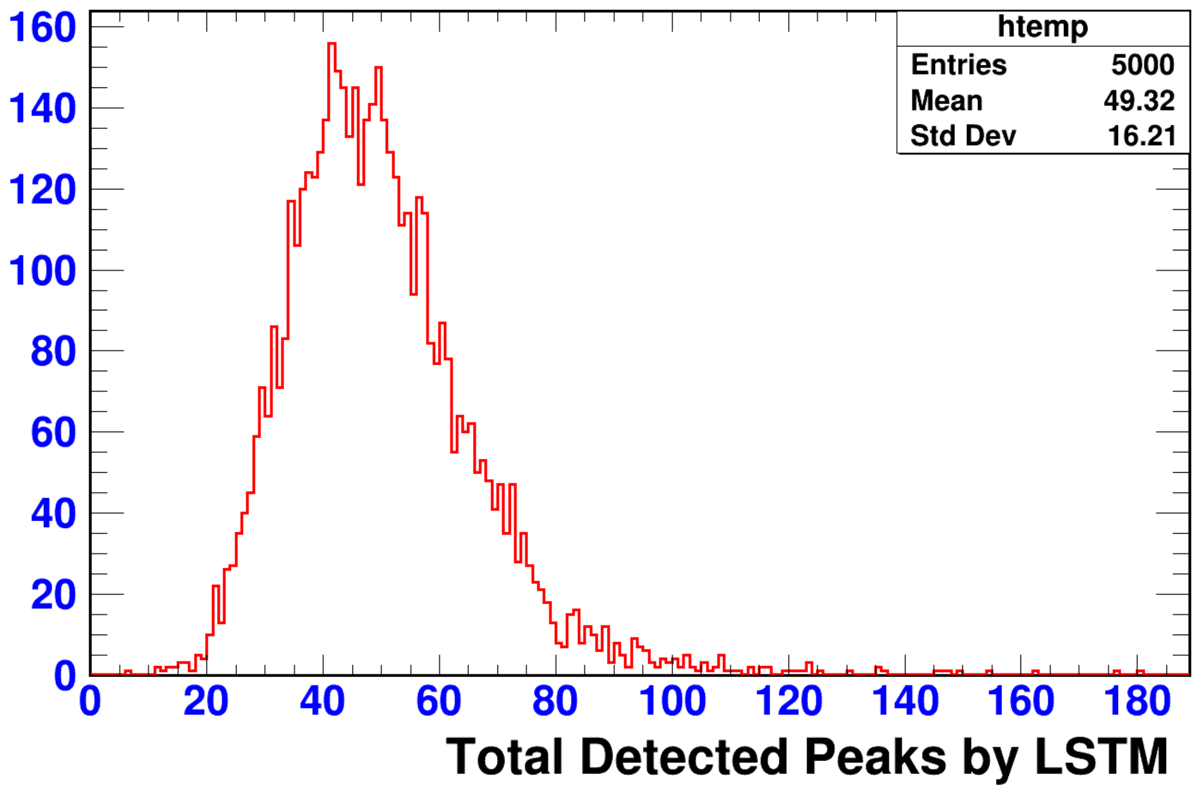
The above table show us different number of primary clusters (MC), cluster size, number of primay cluster (Target) and primary cluster detected by CNN with standard Deviations for different momenta are shown in the table

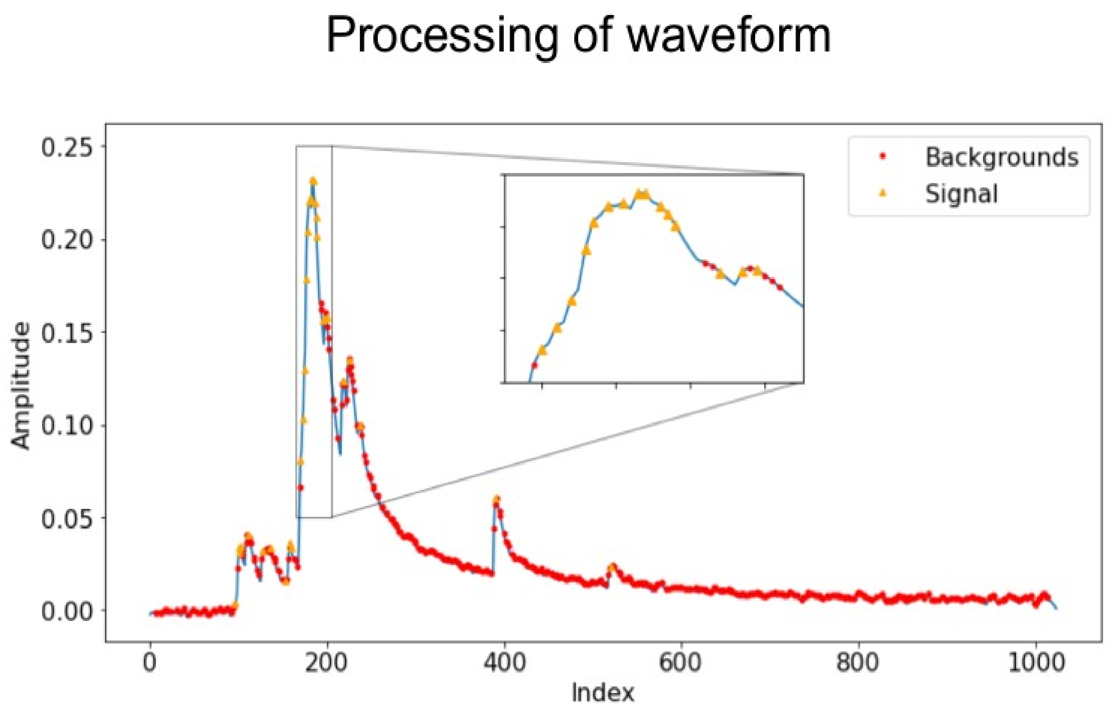

Thank you

Background Slides

Summary of data sets used for training and testing ML-based cluster counting algorithms

Purpose	Algorithm	Number of Event	Momentum
Training LSTM Model	PeakFinding	5000	0 – 20 GeV/C
Testing LSTM Model	Peak Finding	5000	0 – 20 GeV/C
Training CNN Model	Clusterization	5000	0 – 20 GeV/C
Testing CNN Model	Clusterization	5000	2, 4, 6, 8, 10 GeV/C


htemp
Entries 5000
Mean 49.32
Std Dev 16.21
0
20
40
60
80
100
120
140
160
180
Total Detected Peaks by LSTM


Processing of waveform
Amplitude
Index
Backgrounds
Signal

1. Peak Detection Based on Probability Cut:

The detection process involves looping over all events and applying a probability cut to decide whether a peak is considered a valid detection:

Looping Over Events: The script iterates over all entries (events) in the probability file.

Applying the Cut: For each event, it checks if the predicted probability (prob_ml) exceeds the cut threshold (cut), which is set to 0.95/0.65.

Storing Detected Peaks: If the probability exceeds the threshold, the corresponding peak time is stored in the detected_time dictionary, keyed by event number (evtno)

2. Matching Detected Peaks with Truth Data

After detecting peaks, the script matches these peaks with the Monte Carlo (MC) truth data to classify them as primary or secondary:

Truth Data: The truth data (truth_time, truth_tag) contains the actual times of primary and secondary peaks, labeled by truth_tag (1 for primary, 2 for secondary).

Matching Function: The match function compares detected peak times with the truth peak times. For each detected peak, it finds the closest truth peak and assigns the corresponding tag (primary or secondary) based on the truth data.

ID Assignment: The id_list array stores the classification of each detected peak as primary (1) or secondary (2)

3. Counting Primary Peaks: After classifying the detected peaks, the script counts how many of them are primary peaks:

Counting: The script iterates over the id_list and increments ncount_pri for every primary peak (tag 1)