1 and 2 particles event discrimination using a CNN

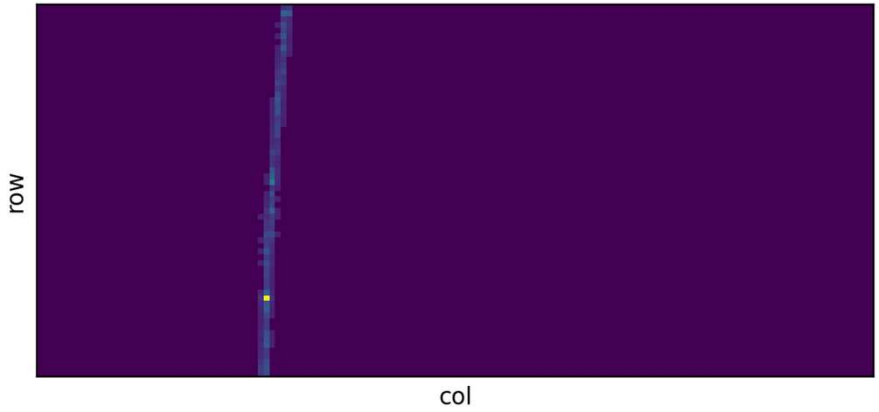
Institut de Física d'Altes Energies

Index

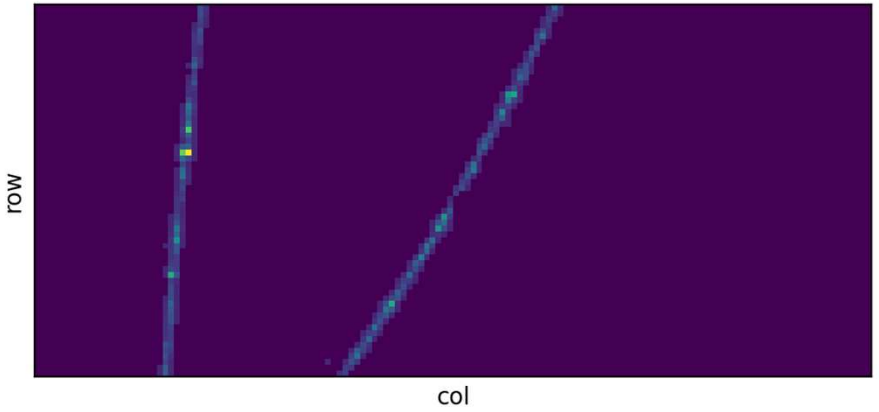
Main Goal

Use a CNN to discriminate between 1 and 2 particles events

1 particle event



2 particle event



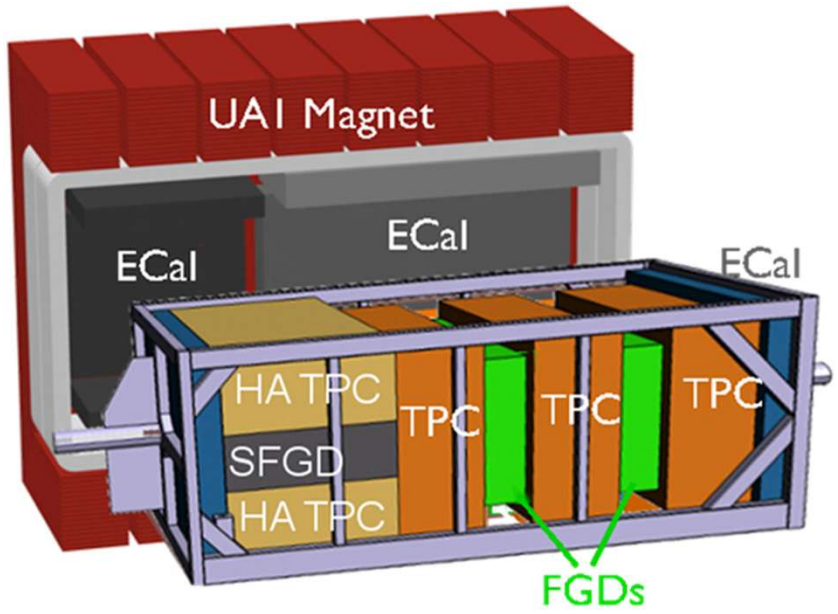
0 Start-Up Experiment

Muons of positive charge, with an energy between 200 MeV and 1 GeV

ND280 Detector

There are 7618 ERAM pads

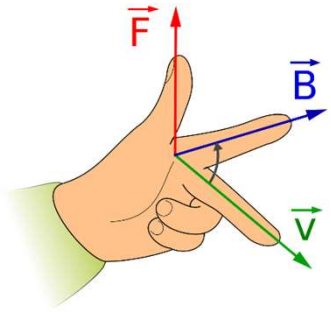
$y, z \rightarrow$ sensor plane

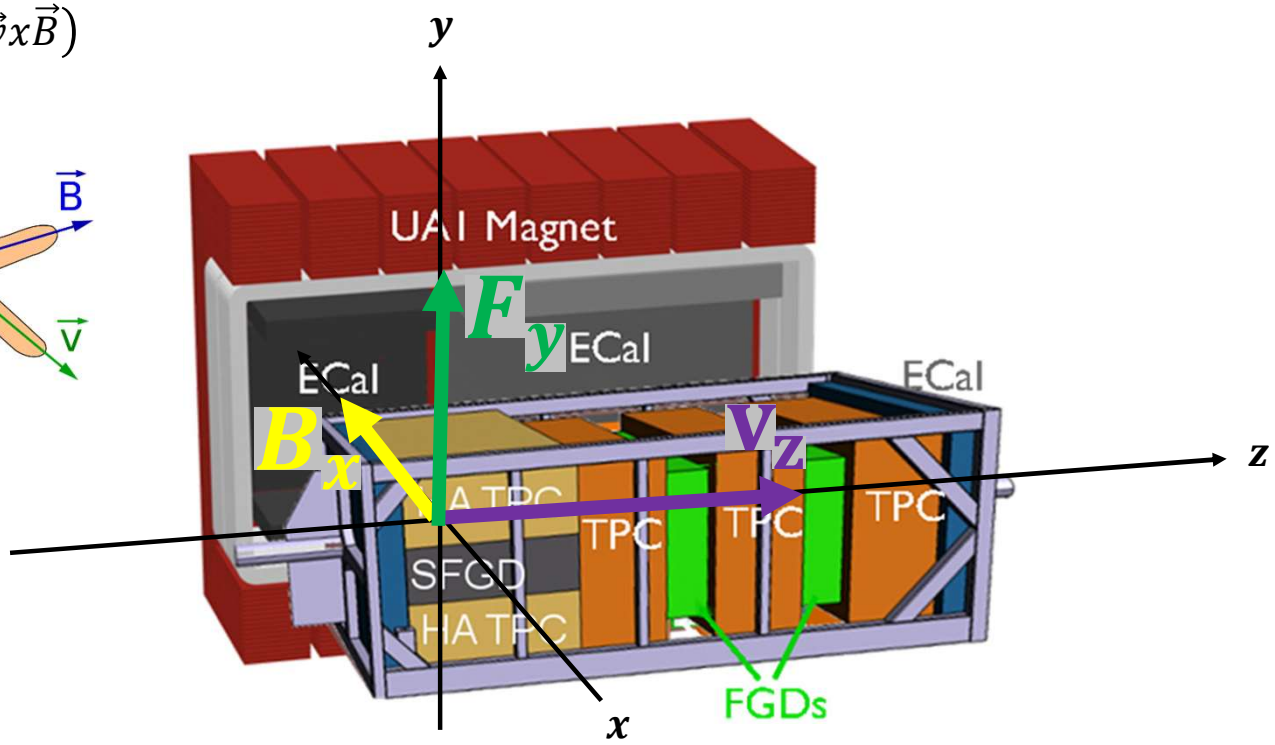


Parameter	Value
Overall x × y × z (m)	2.0 × 0.8 × 1.8
Drift distance (cm)	90
Magnetic Field (T)	0.2
Electric field (V/cm)	275
Gas Ar-CF_4-iC_4H_{10} (%)	95 - 3 - 2
Drift Velocity $cm/\mu s$	7.8
Transverse diffusion ($\mu m/\sqrt{cm}$)	265
Micromegas gain	1000
Micromegas dim. z×y (mm)	340 × 410
Pad z × y (mm)	10 × 11
N pads	36864
el. noise (ENC)	800
S/N	100
Sampling frequency (MHz)	25
N time samples	511

Coordinates definition

$$\vec{F} = q(\vec{v}x\vec{B})$$

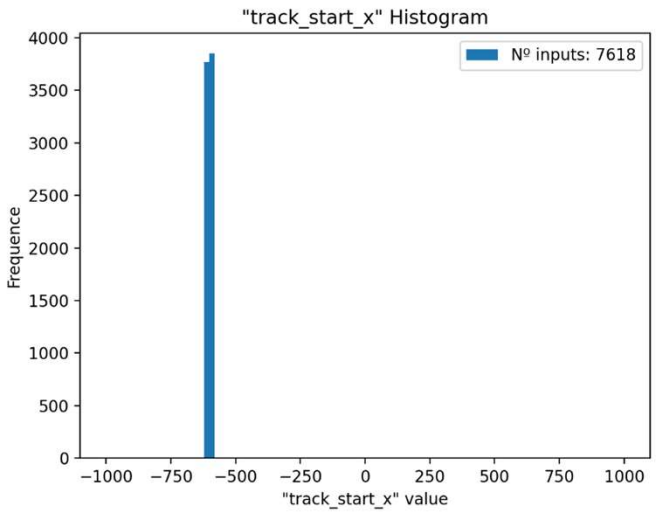


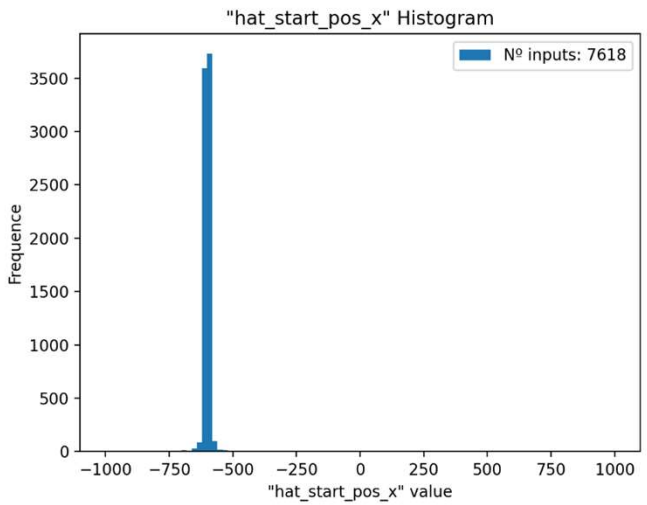


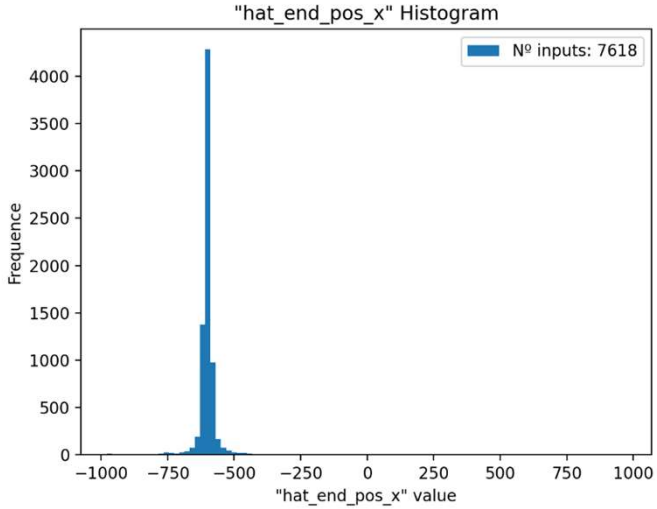
x = 2m
y = 0.8 m
z = 1.8 m

1 Hattracks Histograms (Monte-Carlo True)

x coordinate



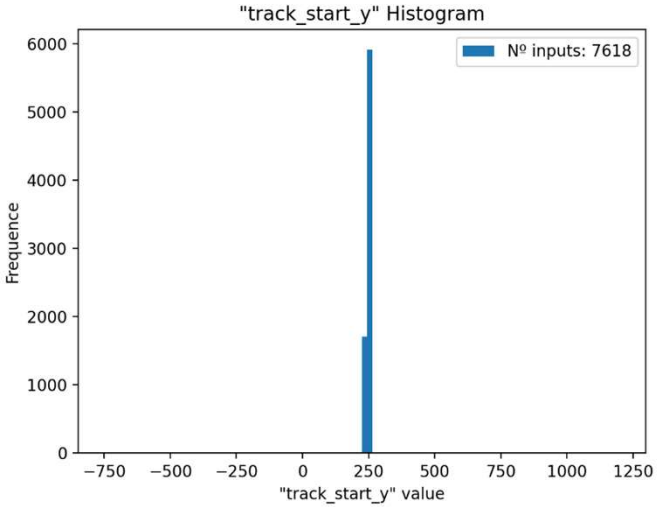


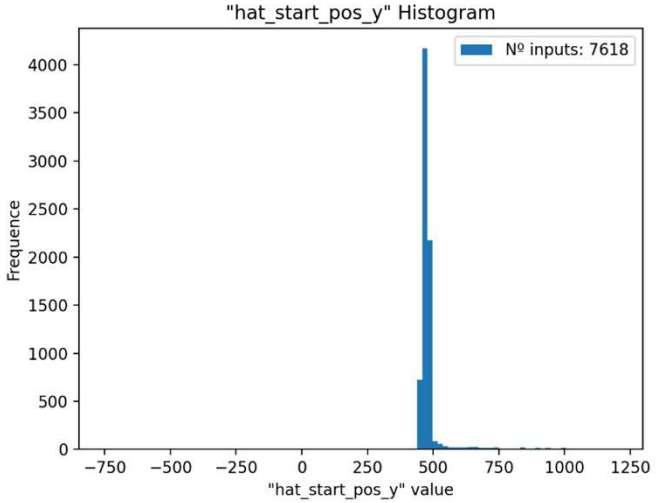


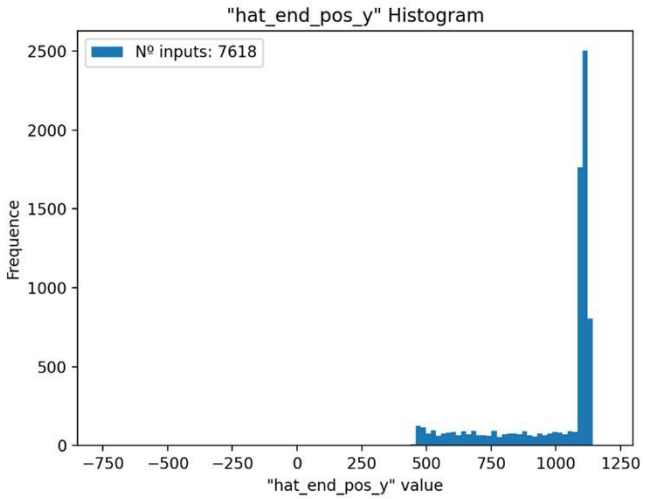
Units in mm

1 Hattracks Histograms (Monte-Carlo True)

y coordinate

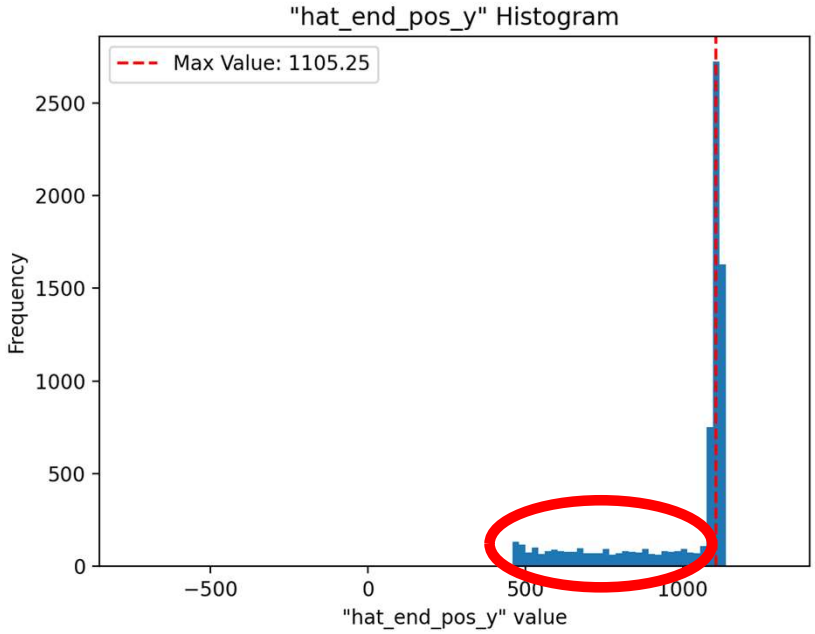


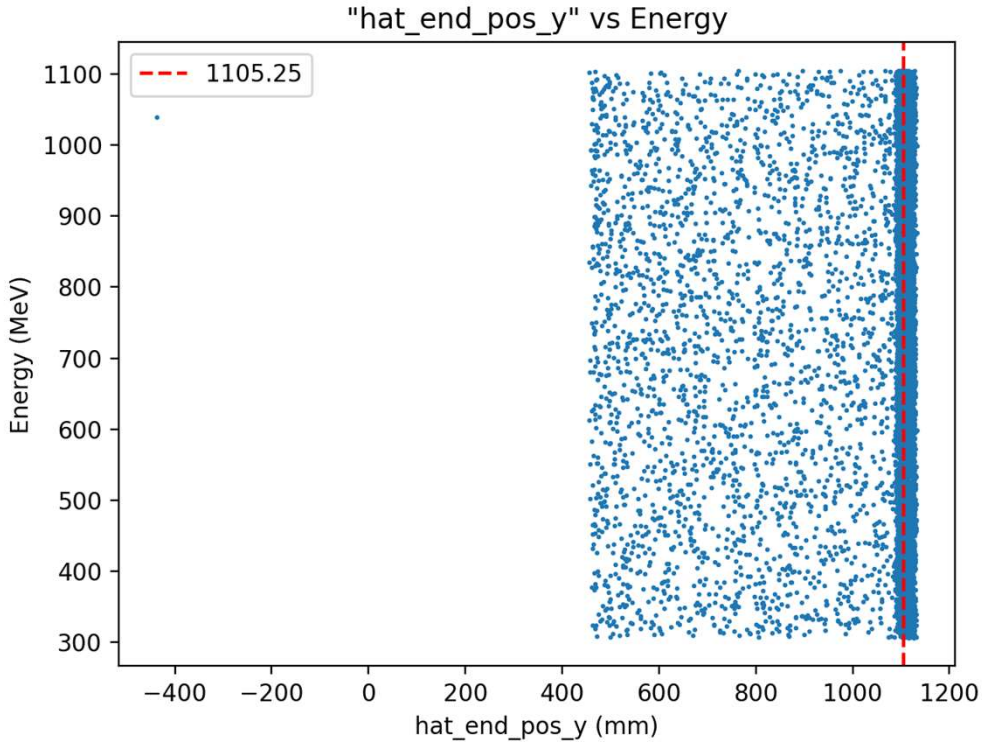




1 Hattracks Histograms (Monte-Carlo True)

Flat base in hat_end_pos_y



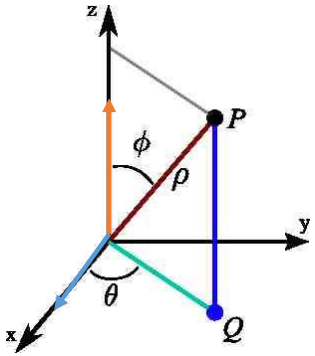


Why does a flat base appear?

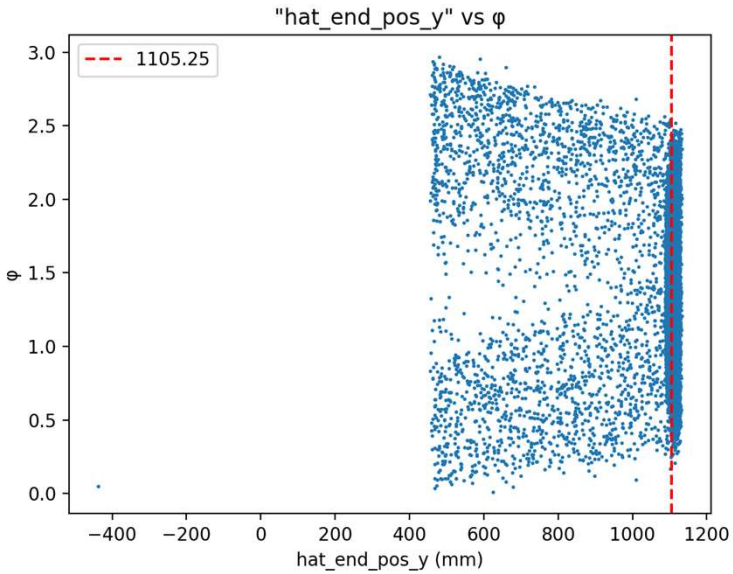
1 Hattracks Histograms (Monte-Carlo True)

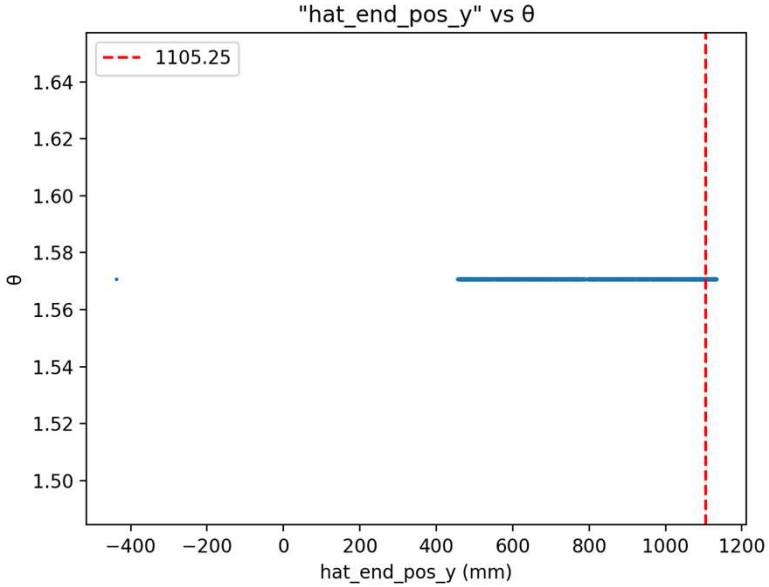
Flat base in hat_end_pos_y

Angle definition:



$$p_x = |p| \sin\phi \cos\theta$$
$$p_z = |p| \cos\ \phi$$

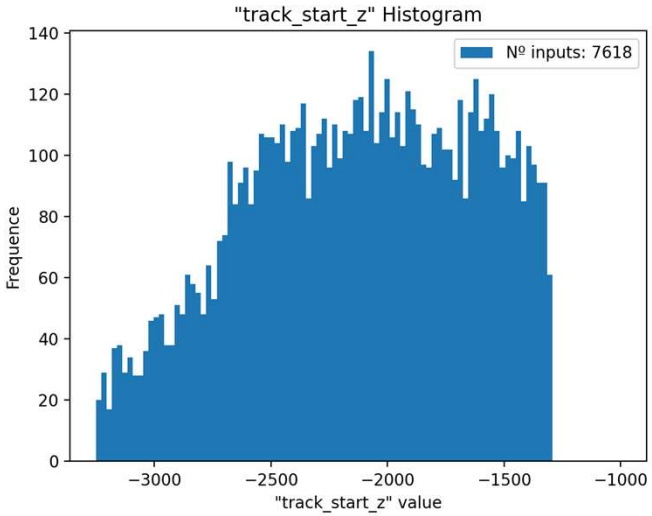


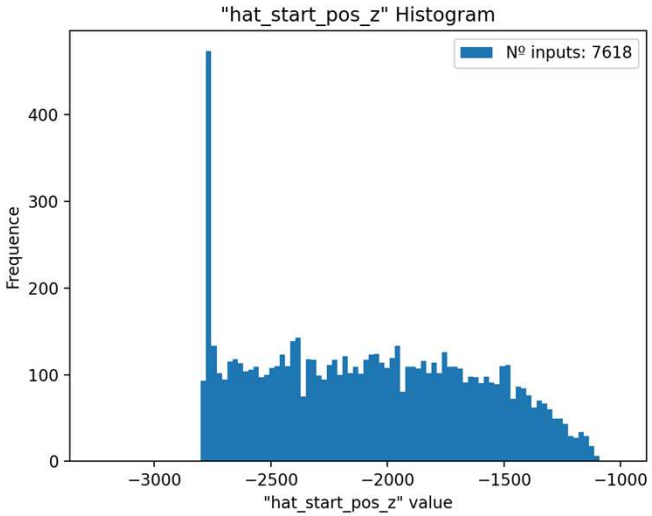


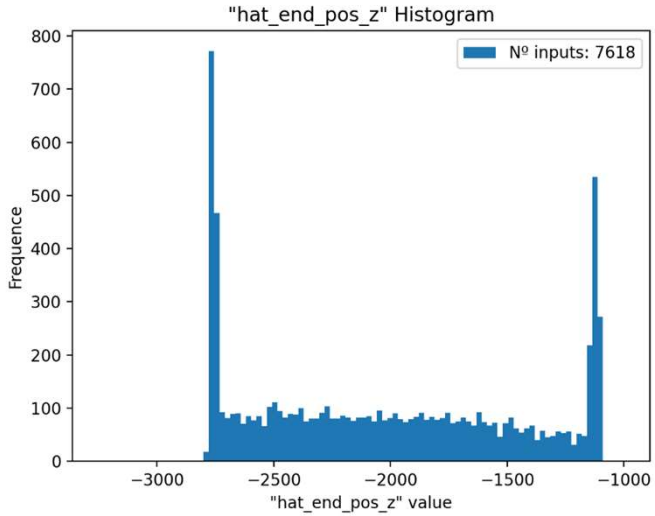
Particles generated at a fixed angle θ

1 Hattracks Histograms (Monte-Carlo True)

z coordinate

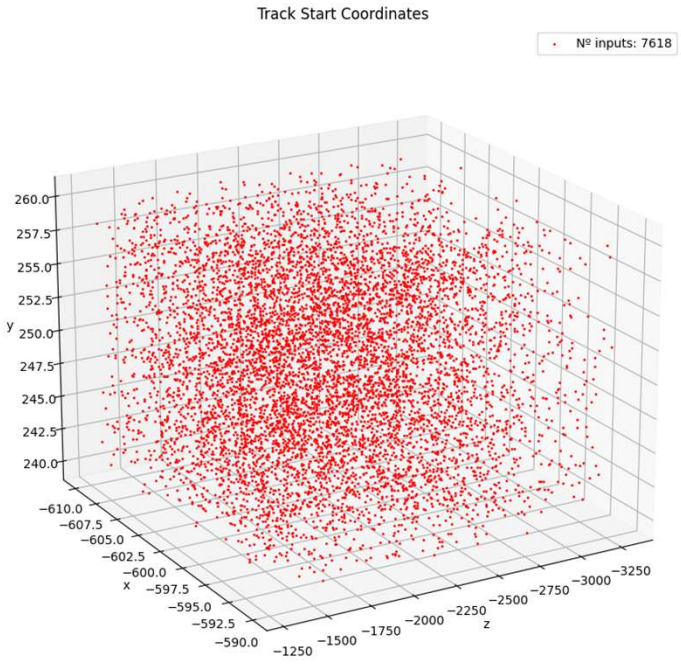


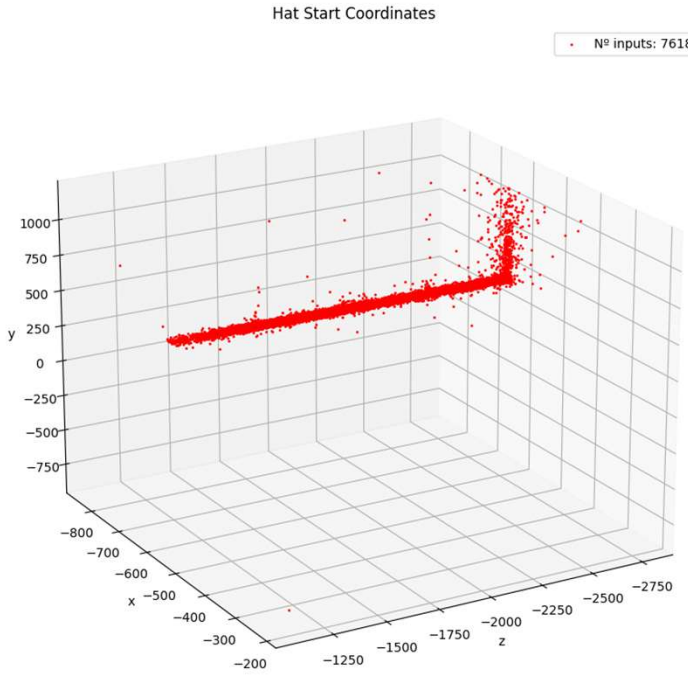


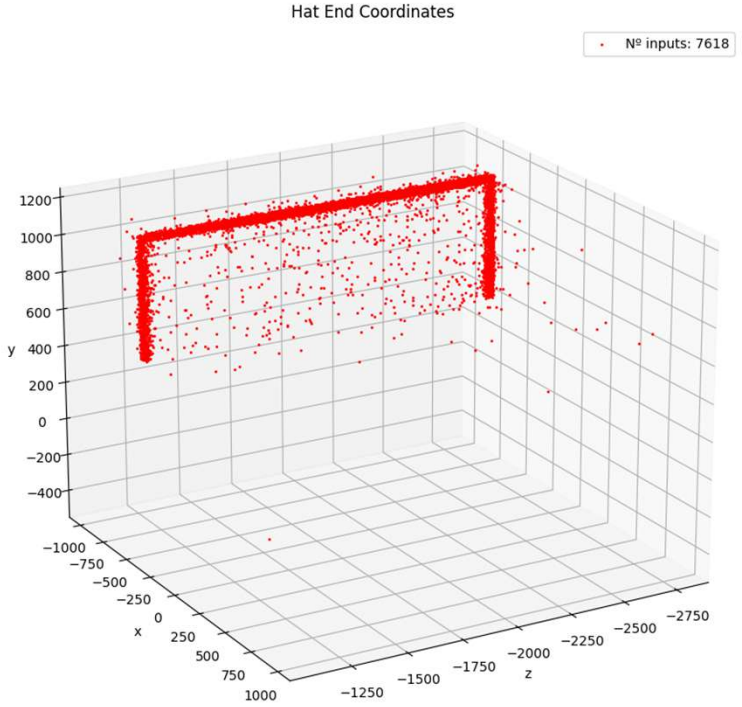


1 Hattracks Histograms (Monte-Carlo True)

3D Coordinates Graphs

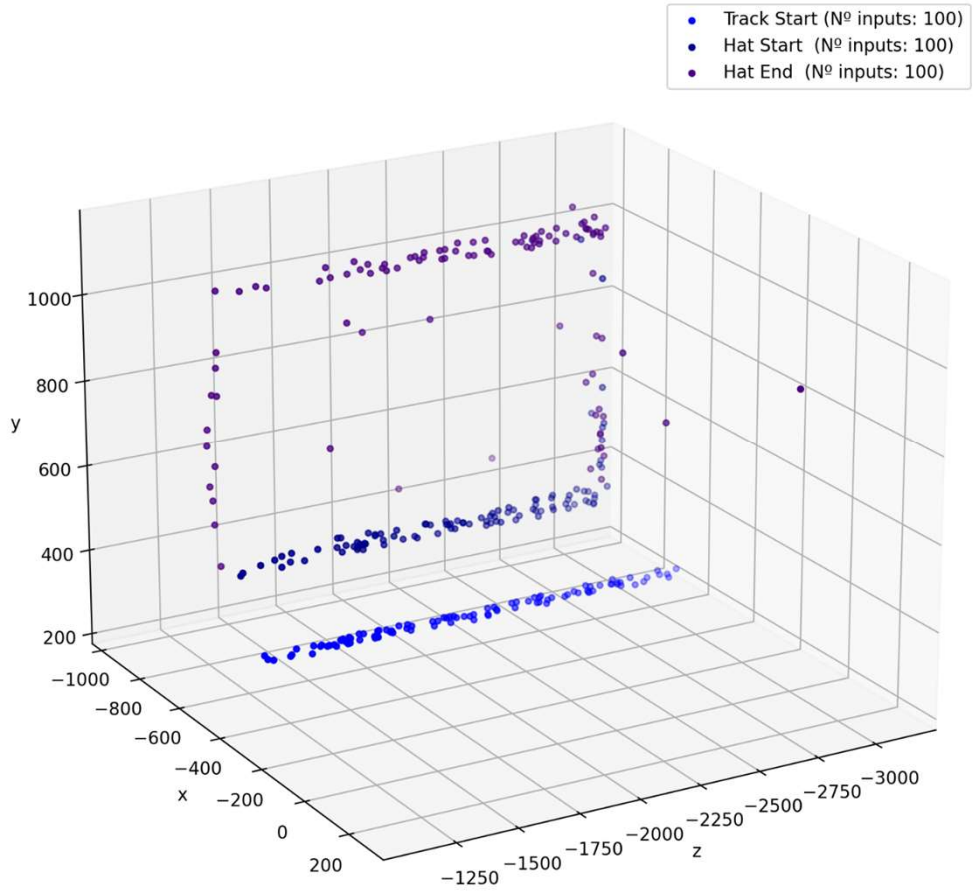






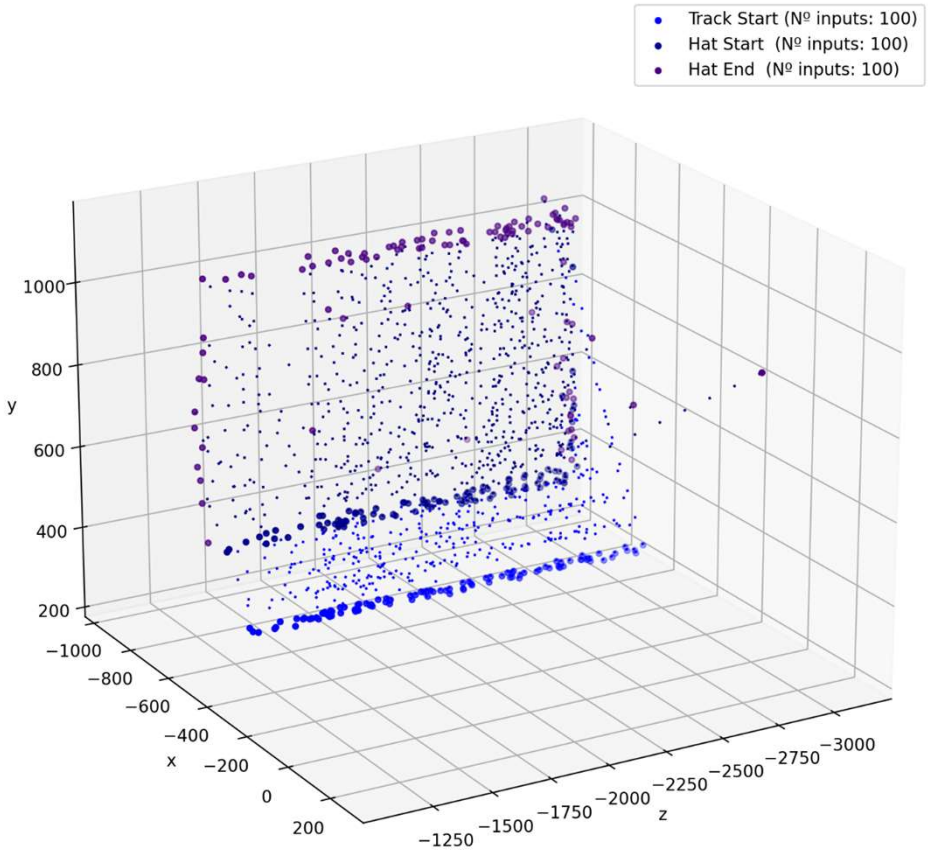
1 Hattracks Histograms (Monte-Carlo True)

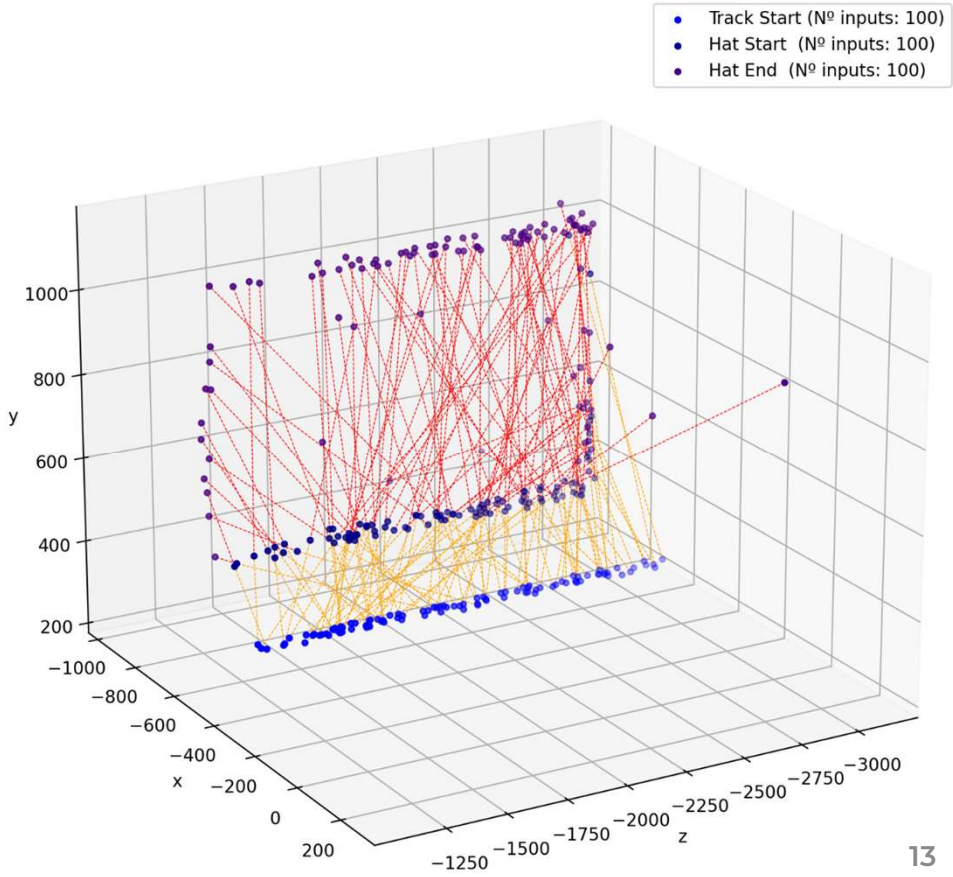
3D Coordinates Graph



1 Hattracks Histograms (Monte-Carlo True)

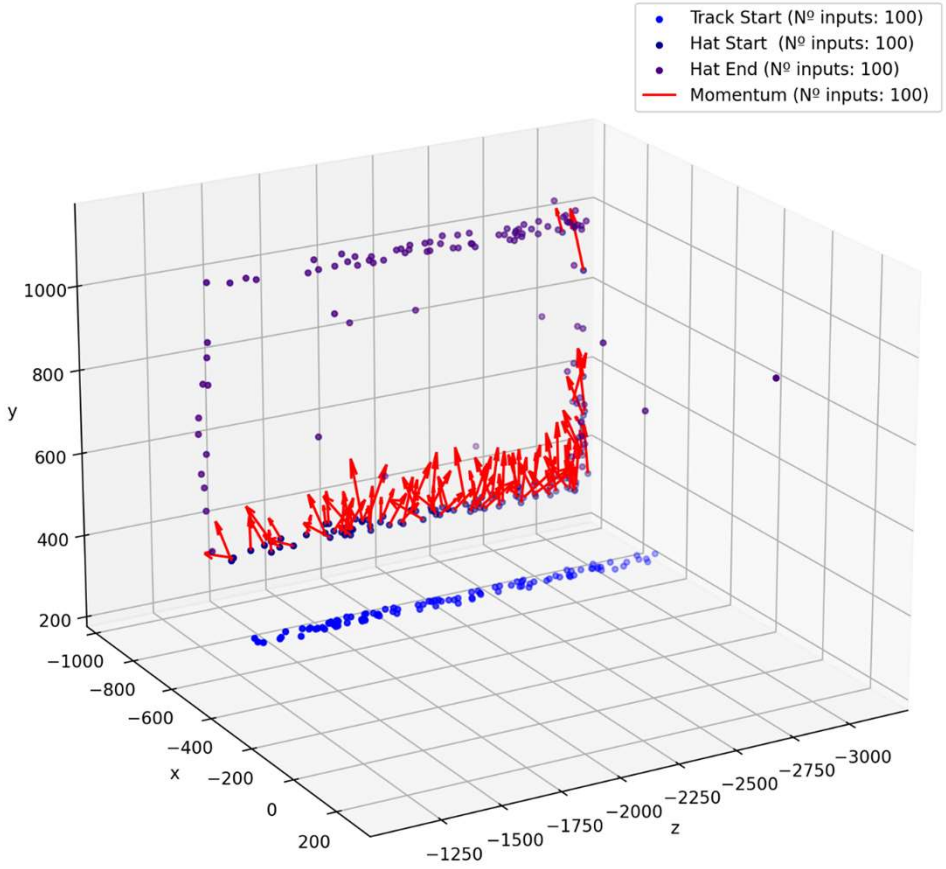
3D Coordinates Graph

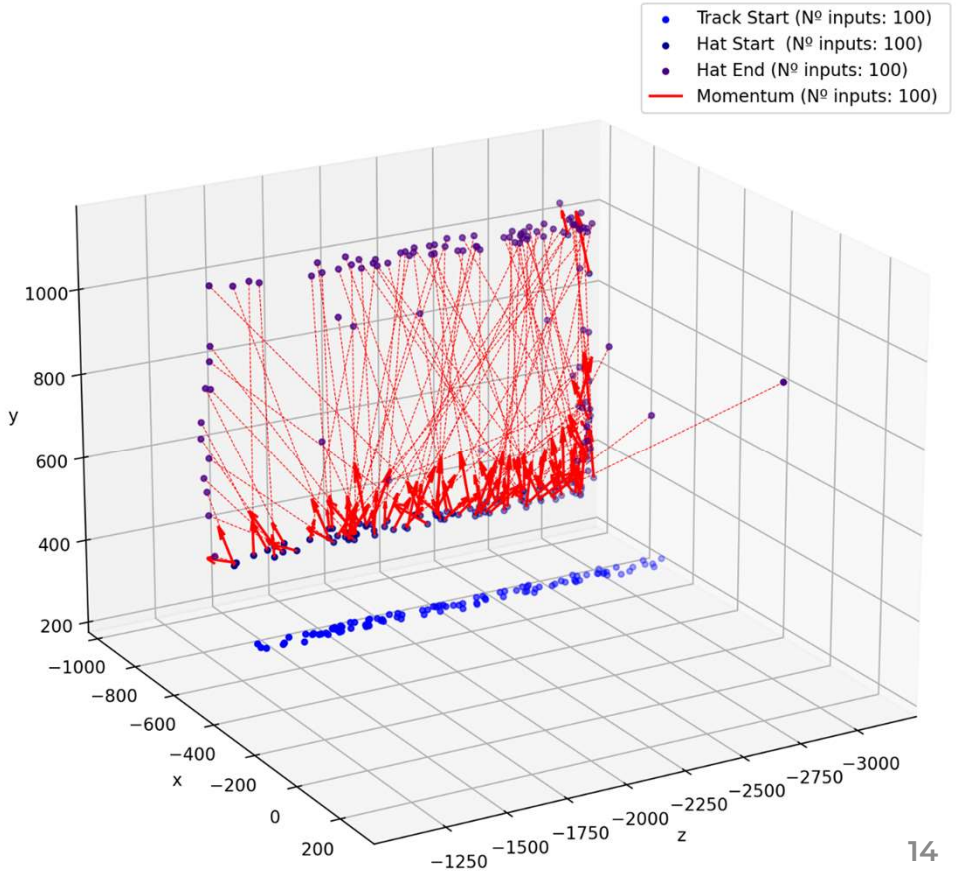




1 Hattracks Histograms (Monte-Carlo True)

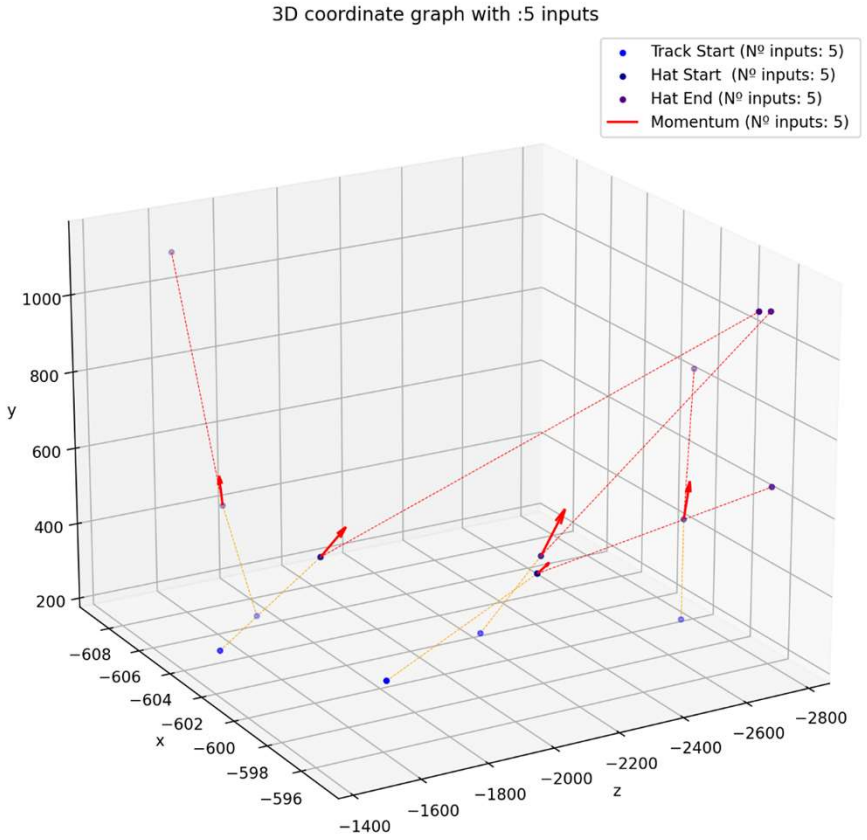
3D Coordinates Graphs with Momentum

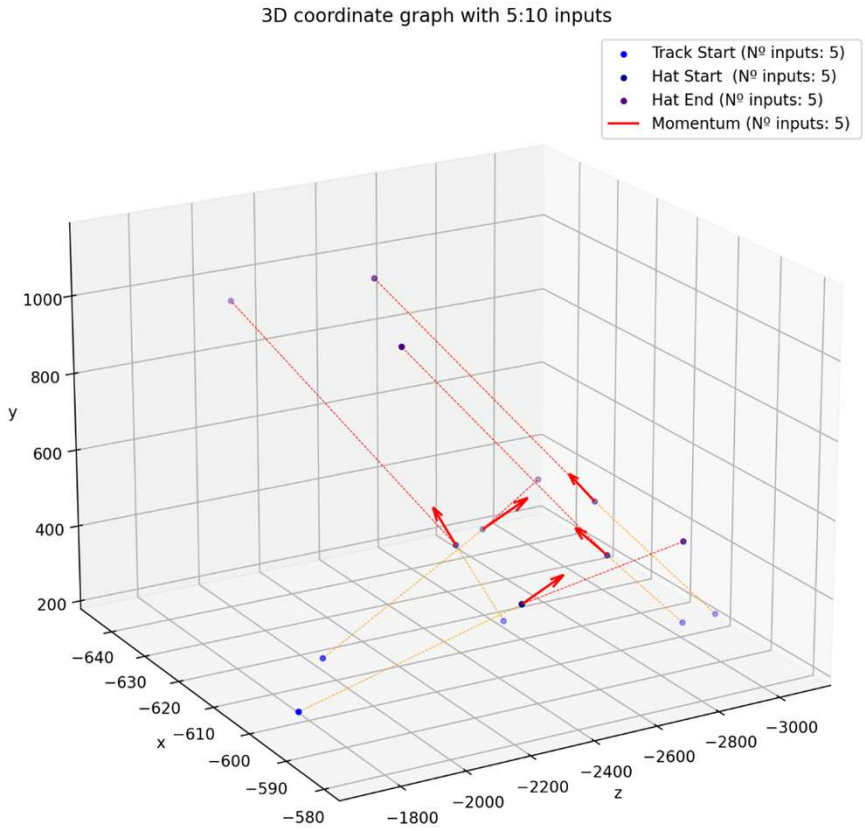




1 Hattracks Histograms (Monte-Carlo True)

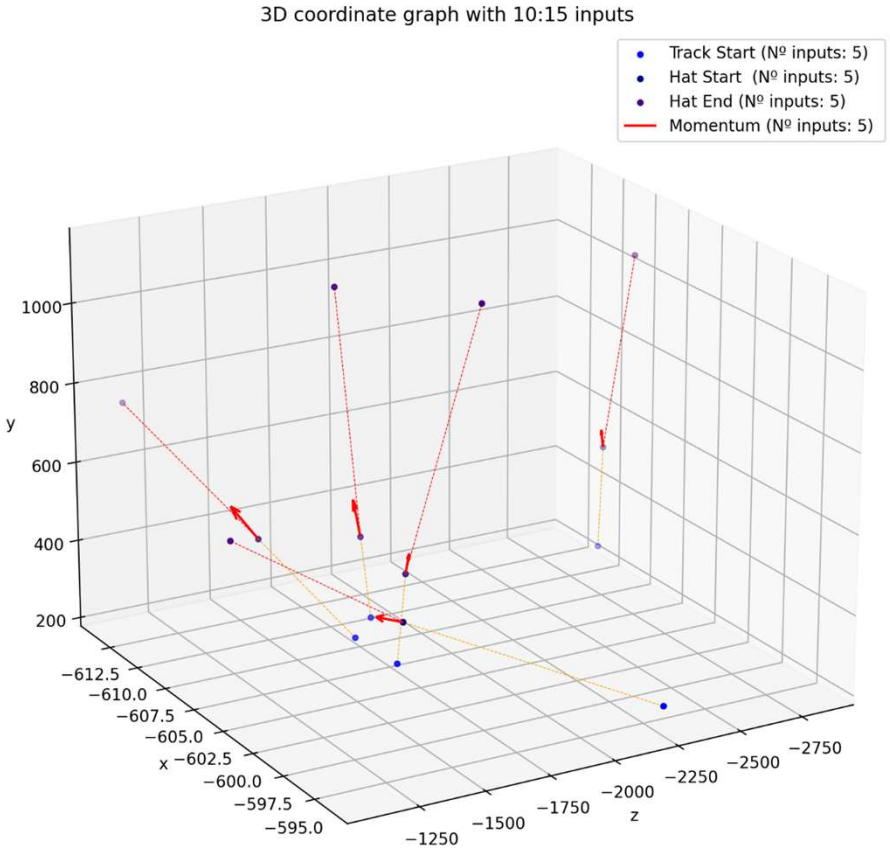
3D Coordinates Graphs with Momentum

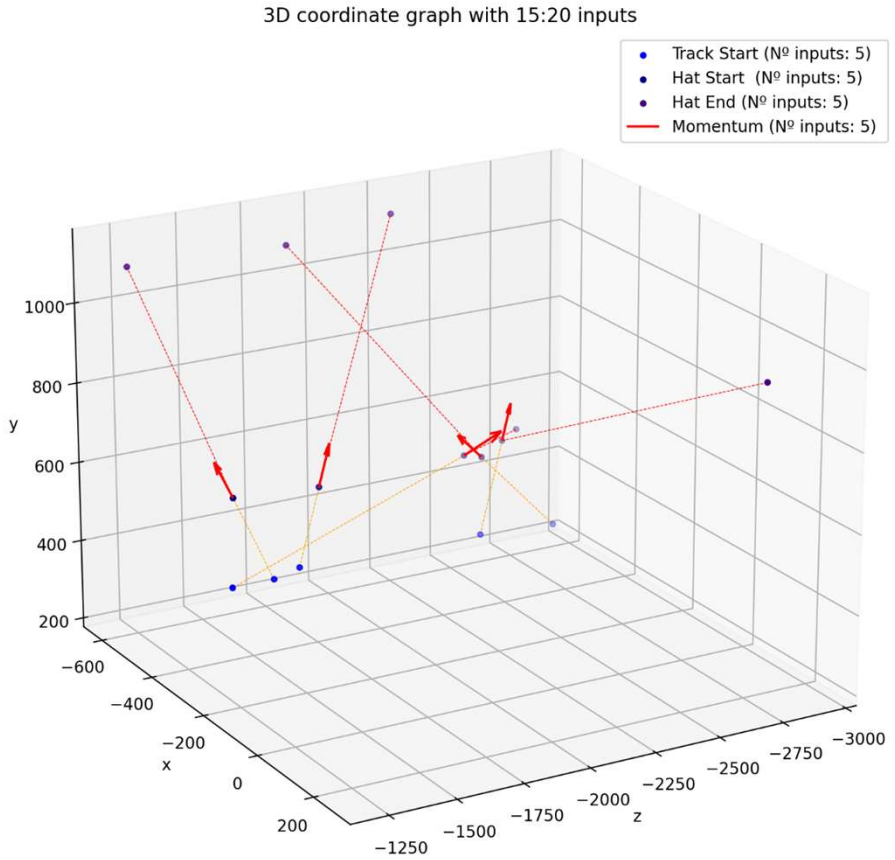




1 Hattracks Histograms (Monte-Carlo True)

3D Coordinates Graphs with Momentum

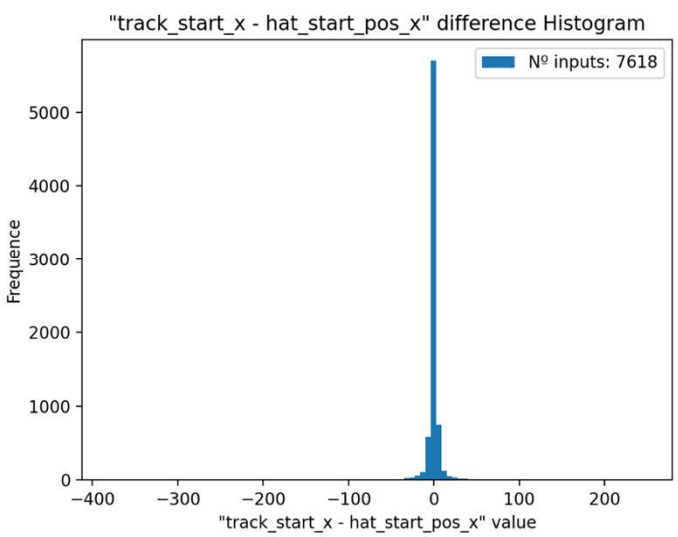




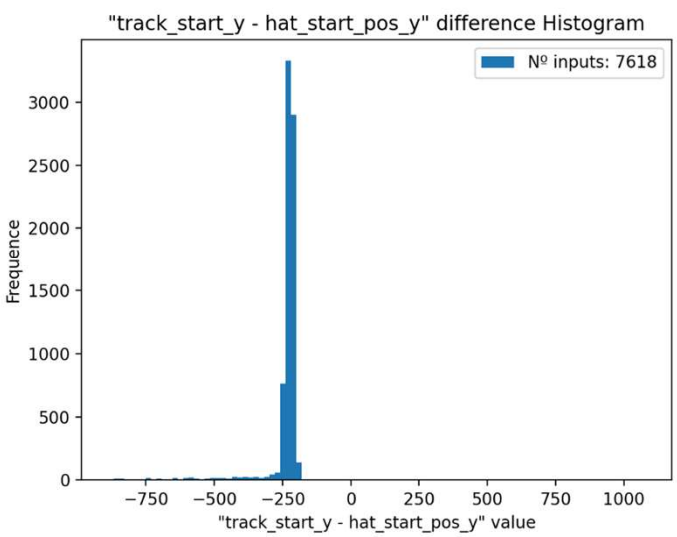
1 Hattracks Histograms (Monte-Carlo True)

Track Start - Hat Start Position Histogram

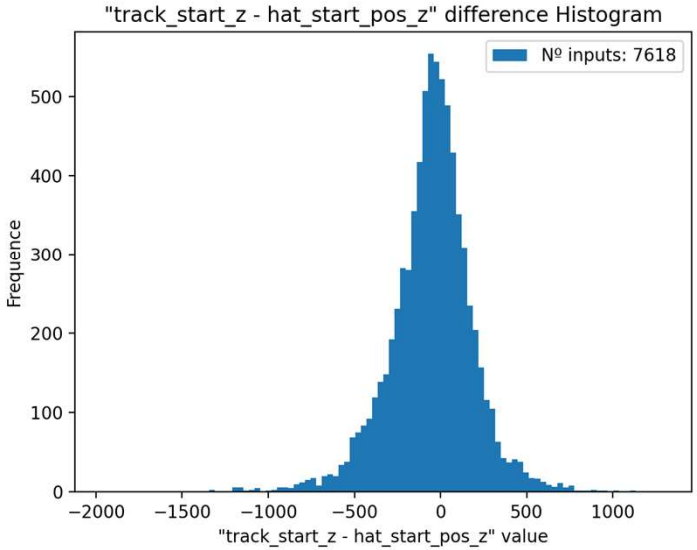
x



y



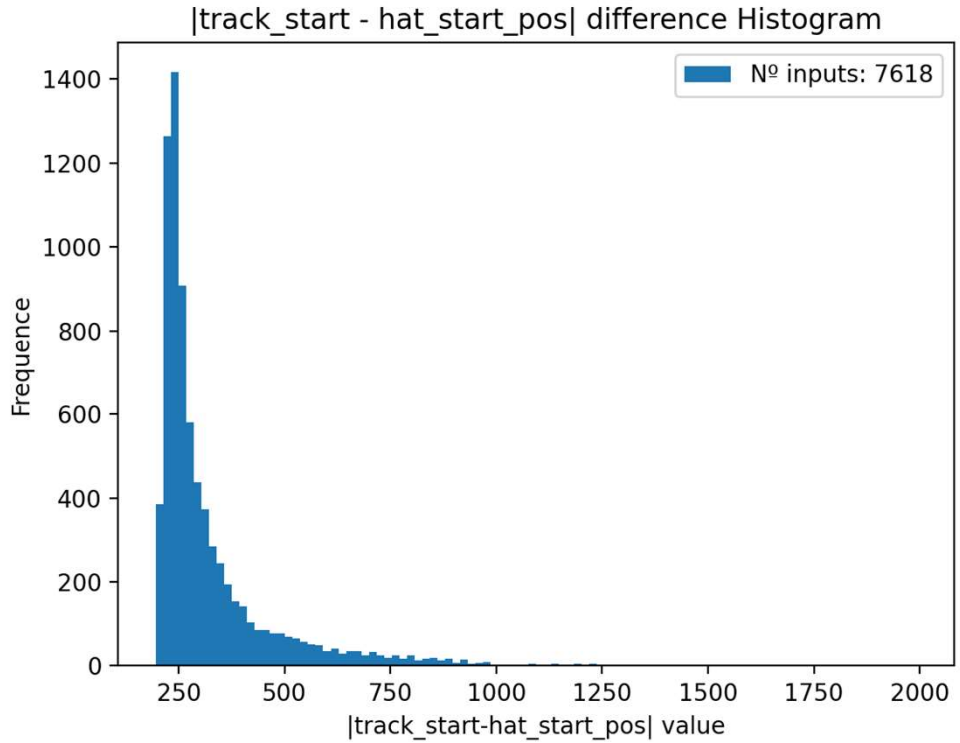
z



1 Hattracks Histograms (Monte-Carlo True)

Track Start - Hat Start Position Histogram

$$|track - hat\_start| = \sqrt{track - hat\_start_x^2 + track - hat\_start_y^2 + track - hat\_start_z^2}$$

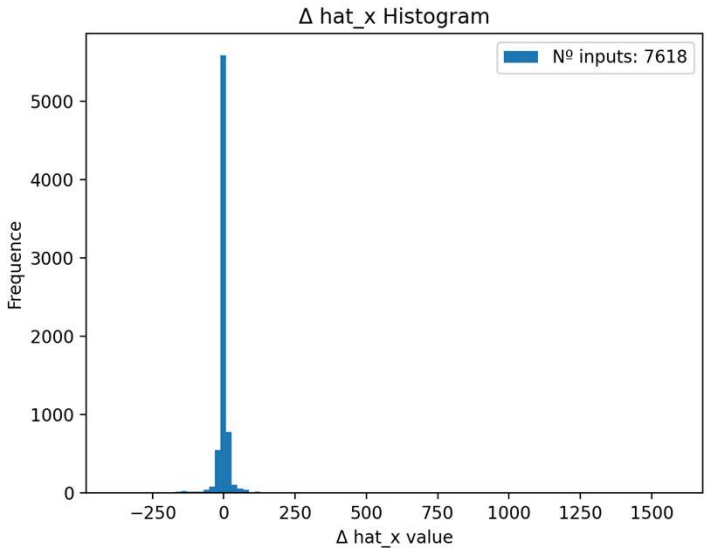


1 Hattracks Histograms (Monte-Carlo True)

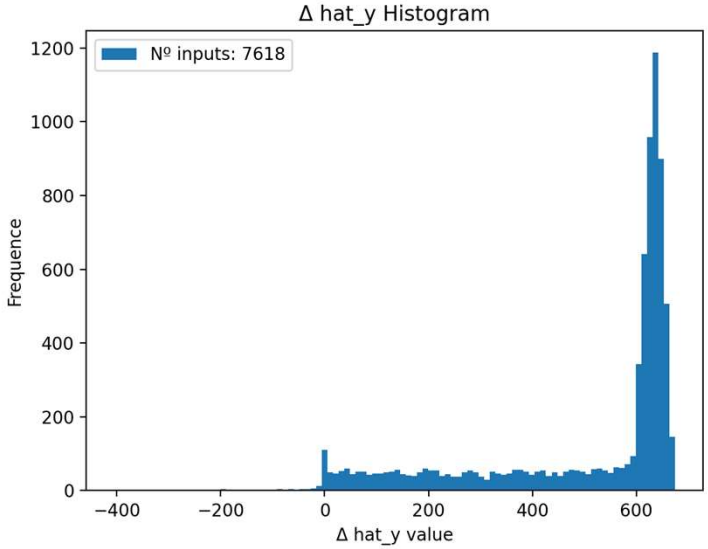
Δ Hat Histogram

$$\Delta\, hat = hat\_end - hat\_start$$

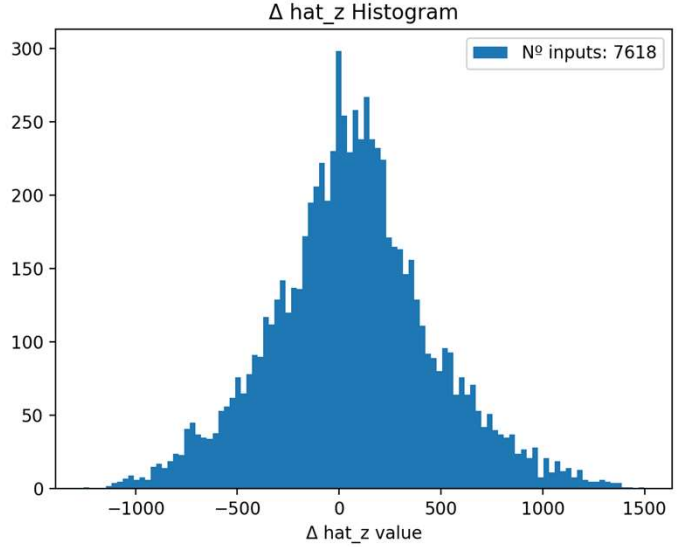
x



y



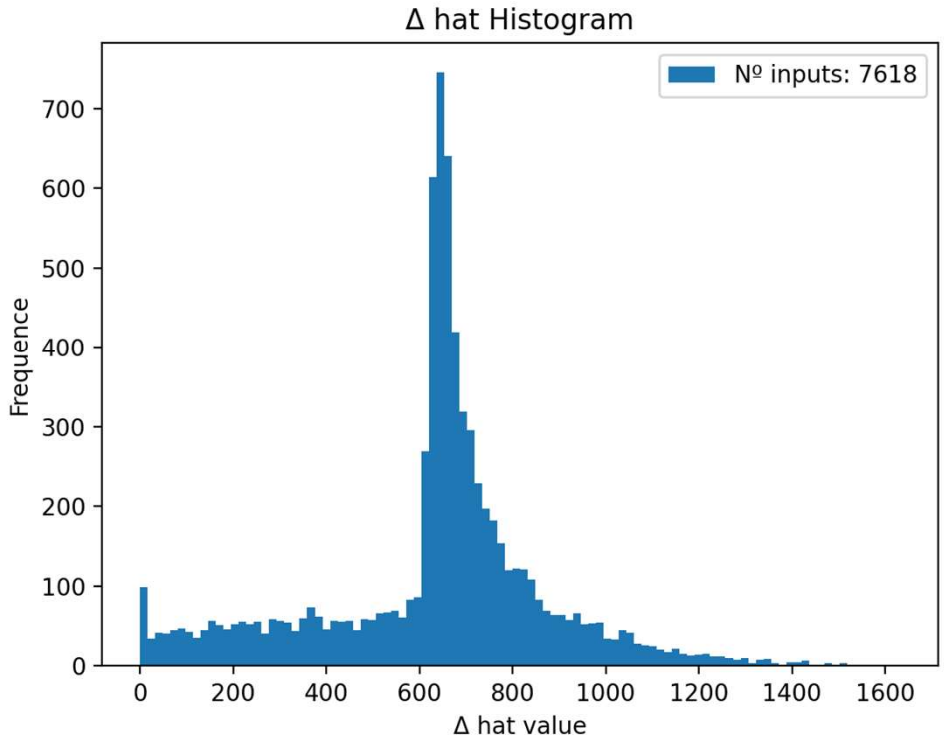
z



1 Hattracks Histograms (Monte-Carlo True)

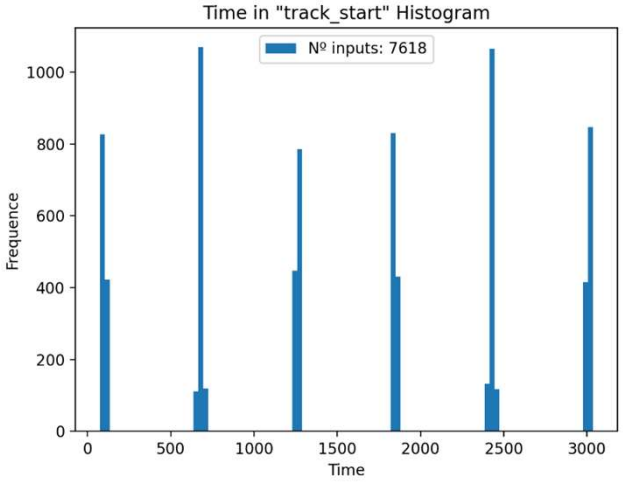
Δ Hat Histogram

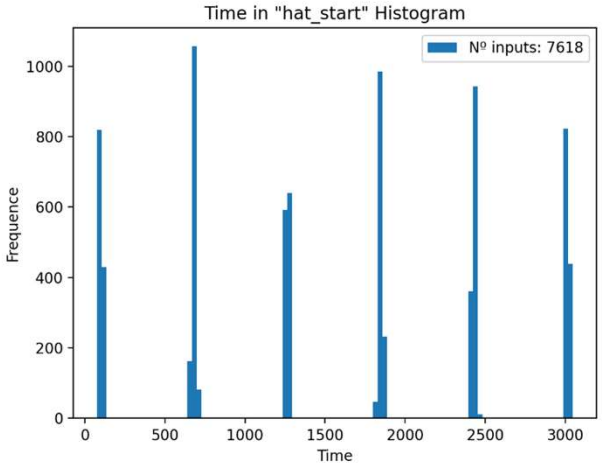
$$|\Delta\, hat| = \sqrt{\Delta\, hat_x^2 + \Delta\, hat_y^2 + \Delta\, hat_z^2}$$

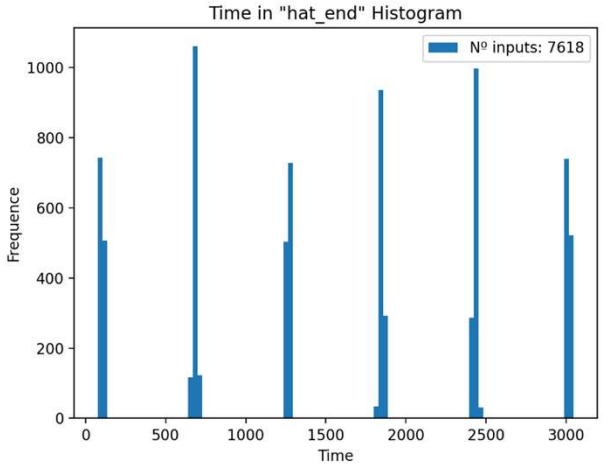


1 Hattracks Histograms (Monte-Carlo True)

Time Histograms



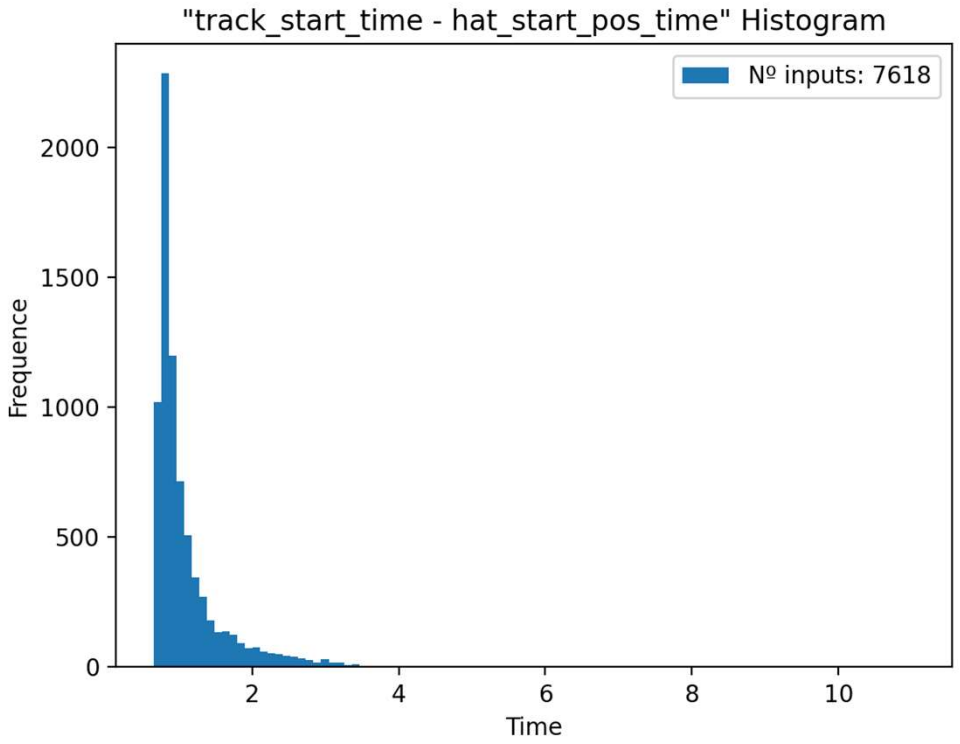




Time in ns

1 Hattracks Histograms (Monte-Carlo True)

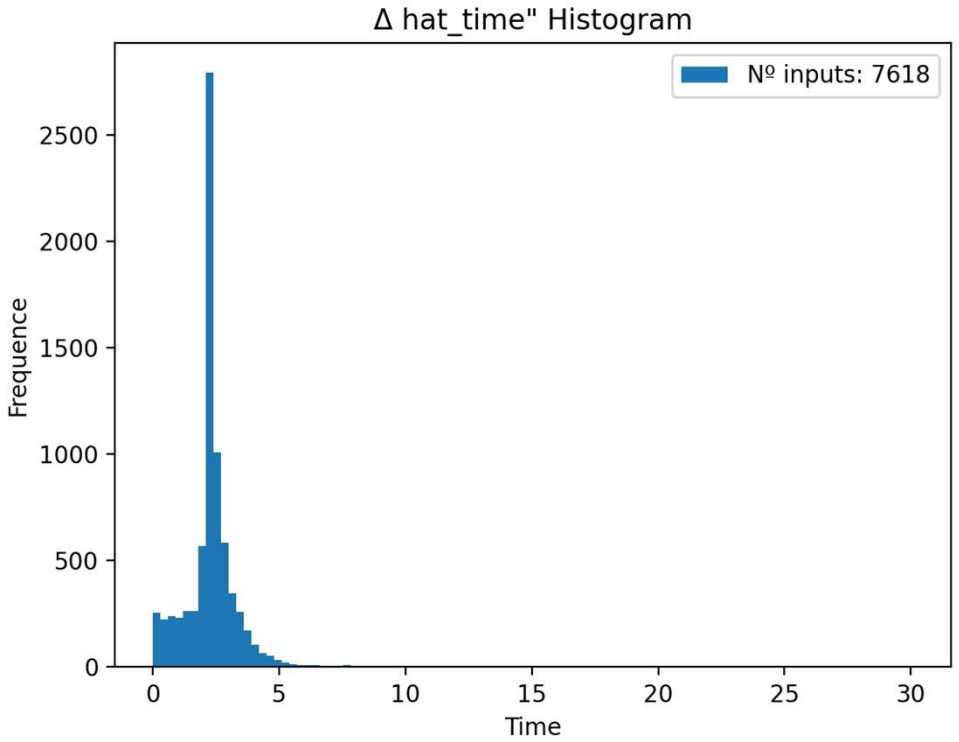
Track Start - Hat Start Time Histogram



1 Hattracks Histograms (Monte-Carlo True)

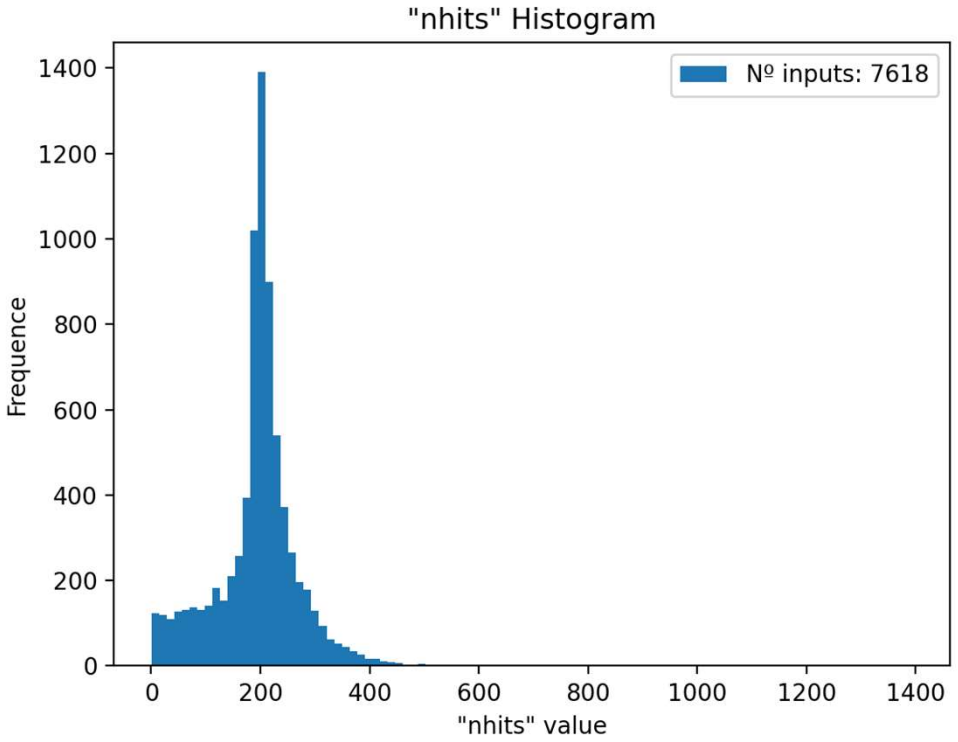
Δ Hat Time Histograms

$$\Delta\ hat\ time = hat\_time\_end - hat\_time\_start$$

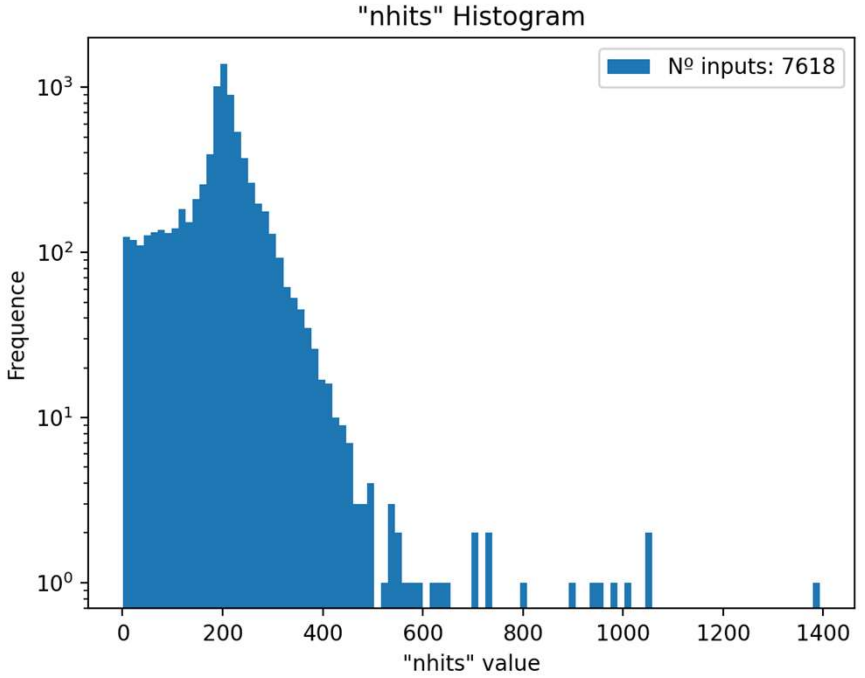


1 Hattracks Histograms (Monte-Carlo True)

Nhits

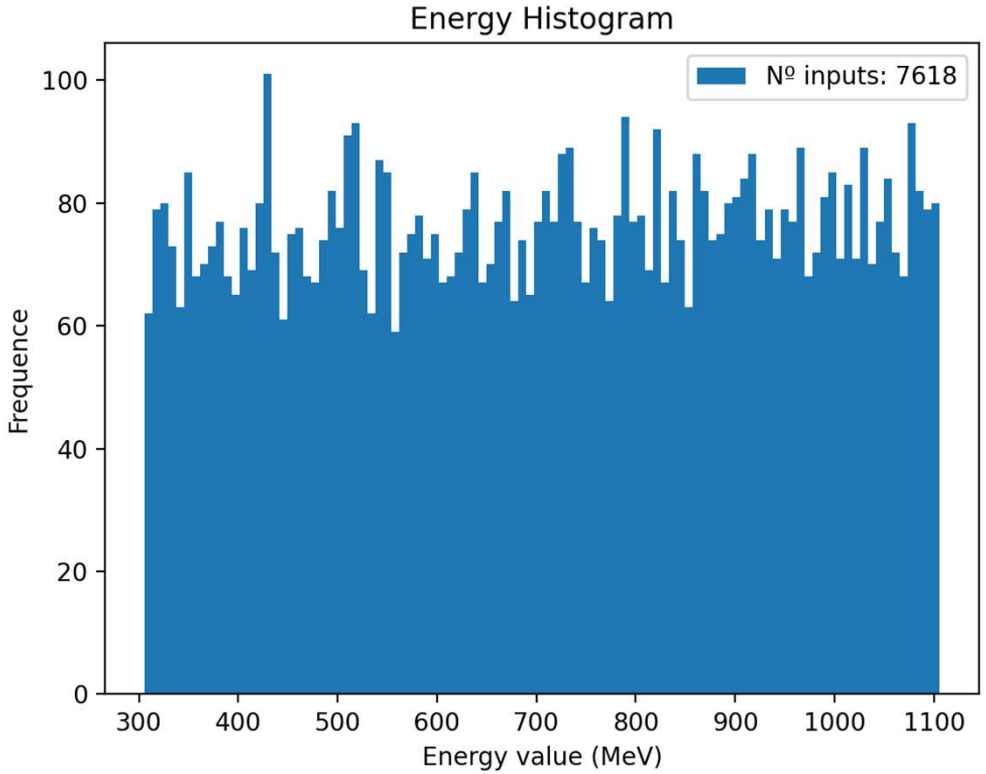


Logarithmic representation



1 Hattracks Histograms (Monte-Carlo True)

Energy Histogram



$$E^2 = (m\,c^2)^2 + (p\,c)^2$$

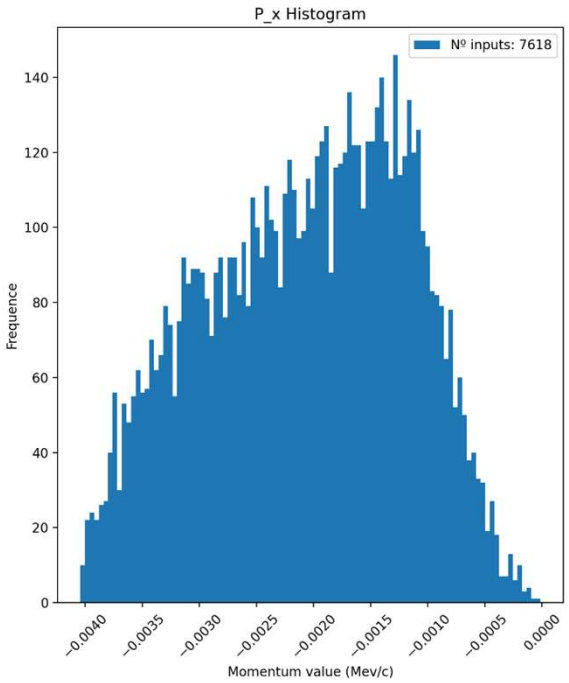
- $(m\,c^2)$ → rest energy
- $T = E - m\,c^2$ → kinetic energy

$$Muon\ rest\ mass = 105.7\ MeV/c$$

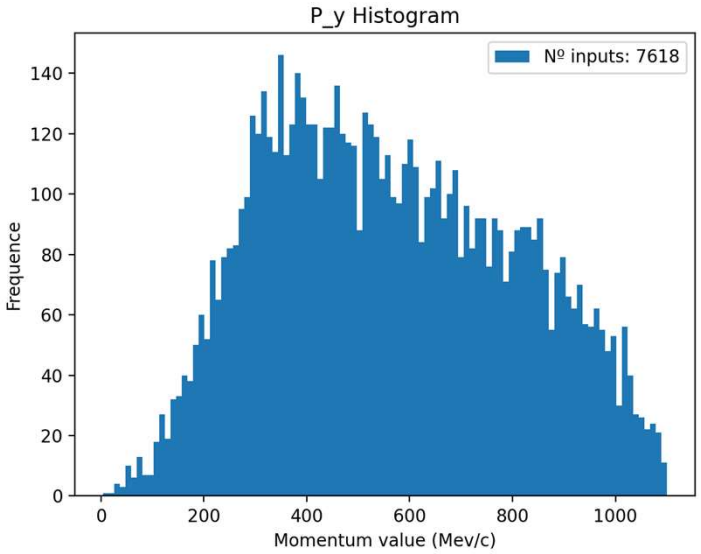
1 Hattracks Histograms (Monte-Carlo True)

Momentum Histogram

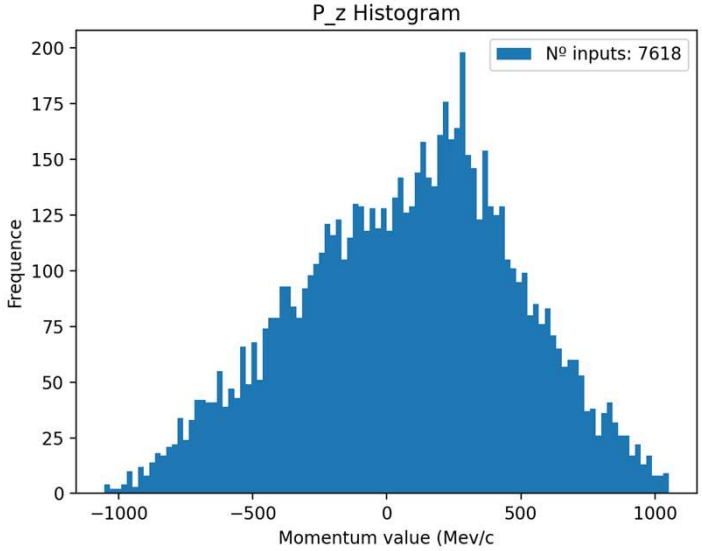
x



y

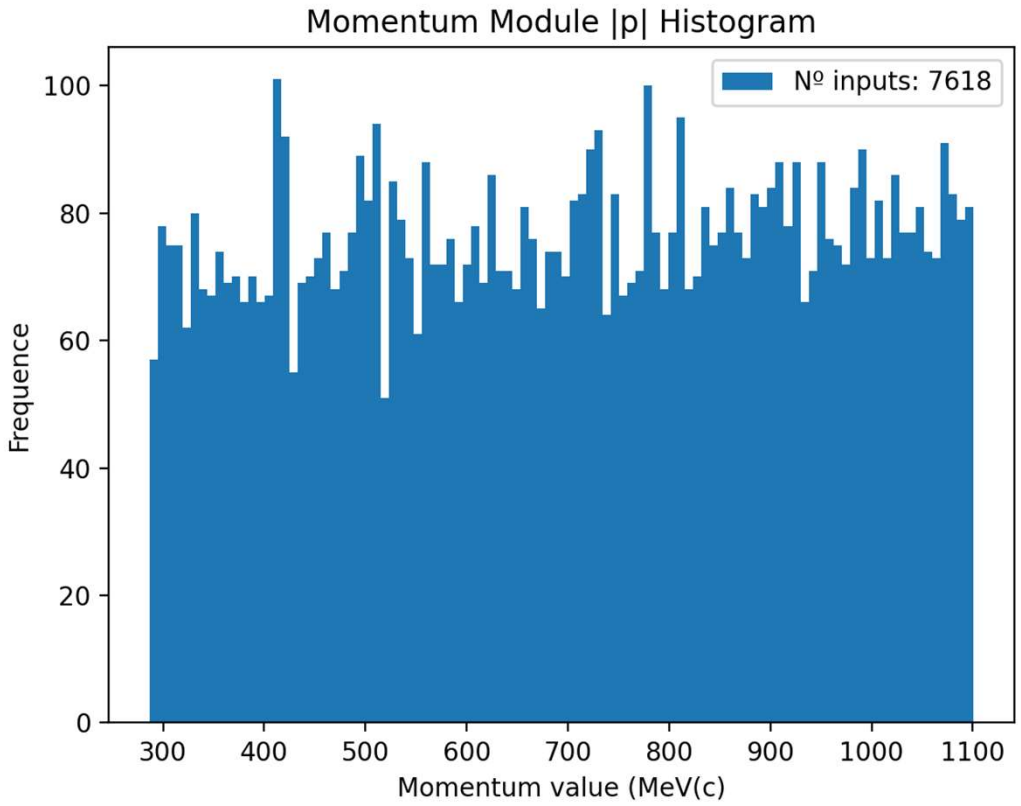


z



1 Hattracks Histograms (Monte-Carlo True)

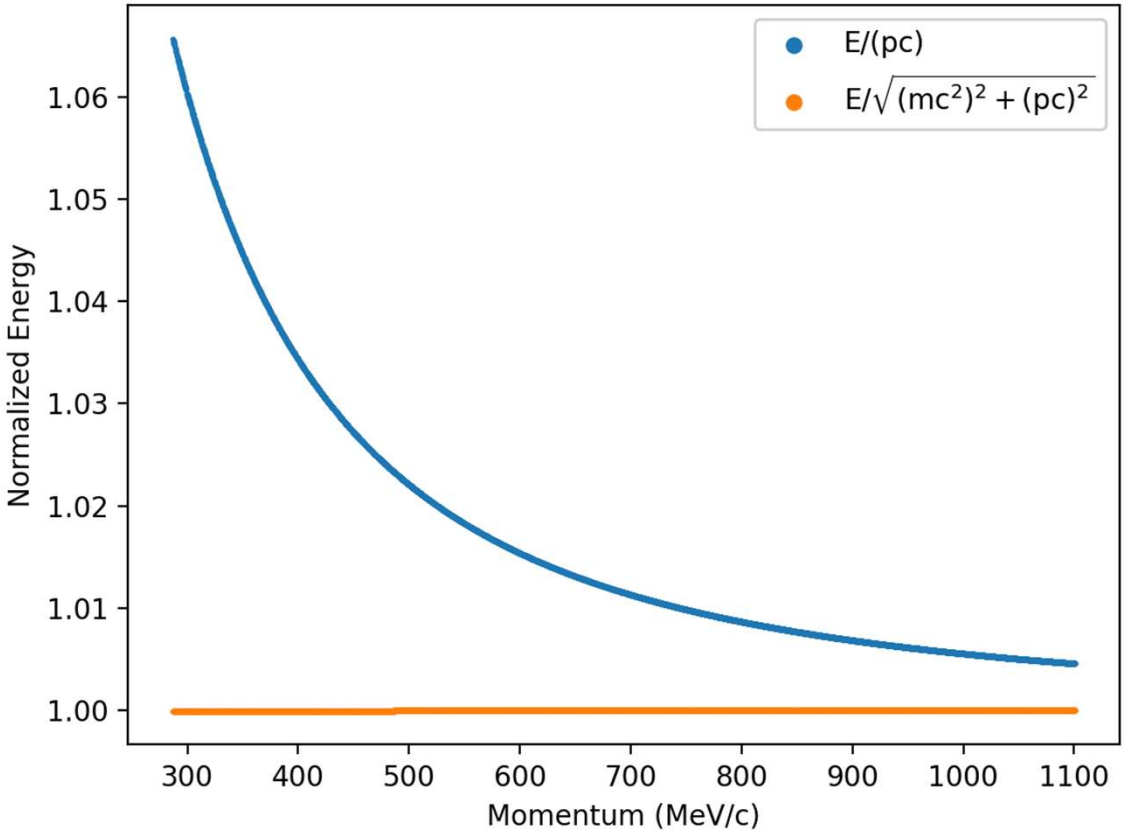
Momentum Histogram



$$p = \sqrt{p_x^2 + p_y^2 + p_z^2}$$

1 Hattracks Histograms (Monte-Carlo True)

Energy Momentum Comparision



2 Hatdigits

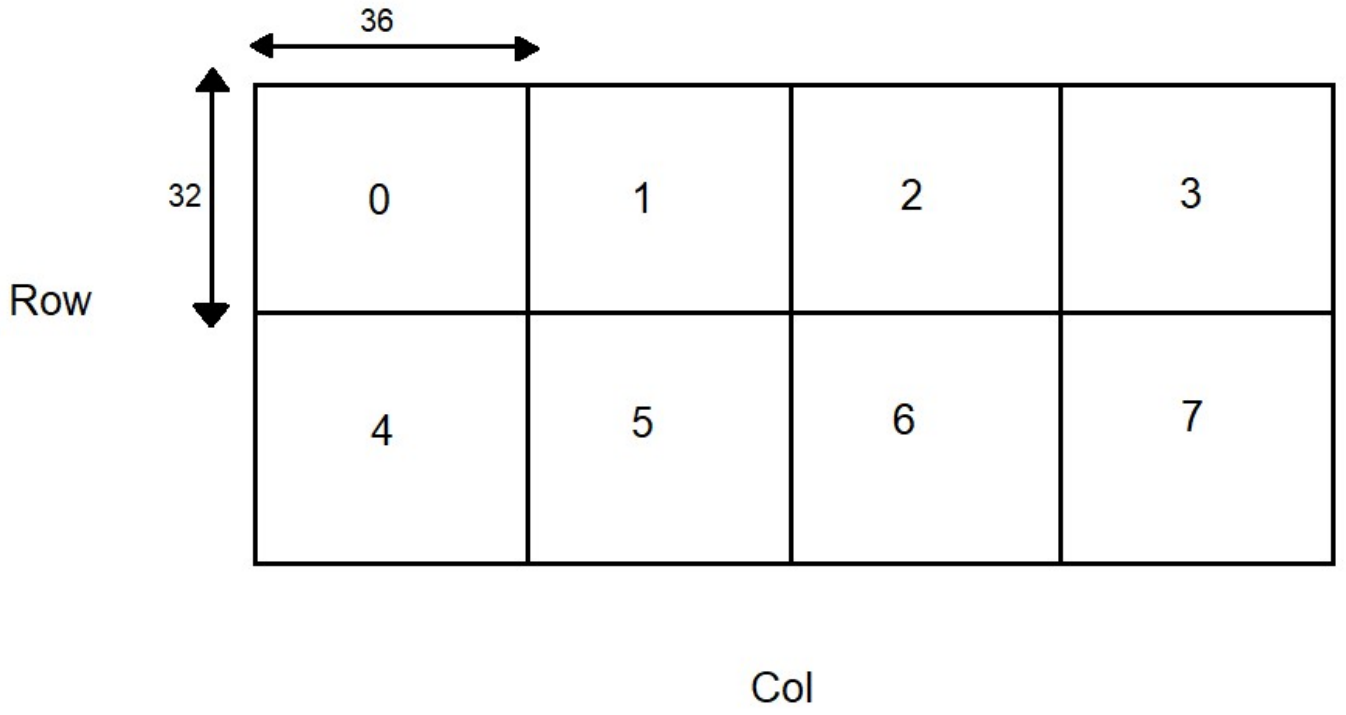
```
df_unique_entries = df.groupby('entry').first()
```

This line of code groups rows in the DataFrame by the 'entry' column and retains the first element of each one, because we are not interested in the different adc values, we are only interested in qmax

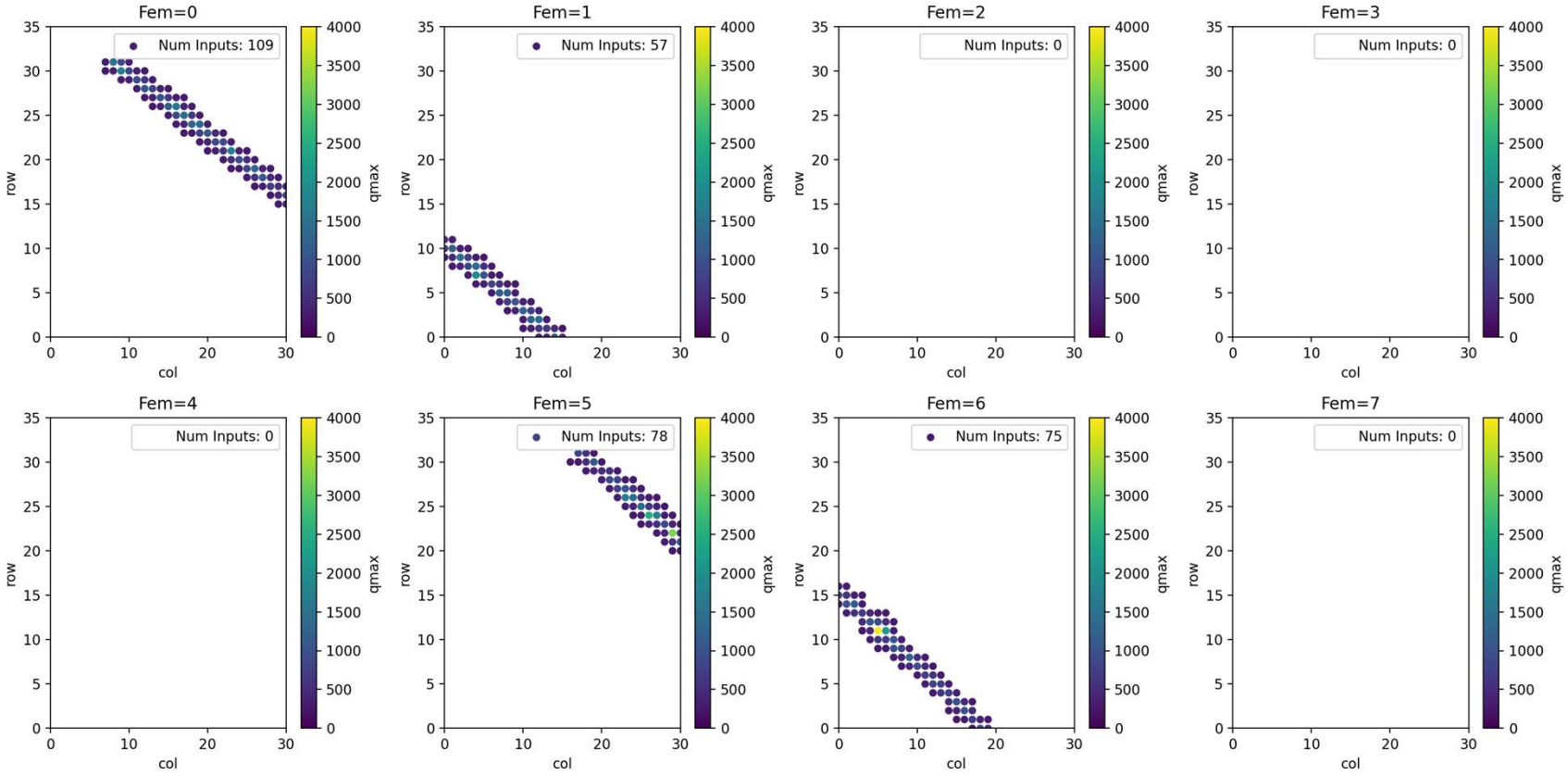
entry	event	hat	plate	fem	fec	asic	channel	time	nsamples	adc	row	col	y	z	qmax	tmax	fwhm
0	0	0	0	2	1	3	9	144	52	246	4	24	849.315002	-1660.115967	292	154	24
1	0	0	0	2	1	3	10	91	68	246	4	25	849.315002	-1648.835938	650	113	22
2	0	0	0	2	1	3	11	86	49	257	4	26	849.315002	-1637.555908	1899	105	14
3	0	0	0	2	1	3	20	134	66	281	5	25	859.505005	-1648.835938	304	158	30
4	0	0	0	2	1	3	21	101	119	253	5	26	859.505005	-1637.555908	311	141	61
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Several hits (entries) describe an event

Fem distribution and Image building

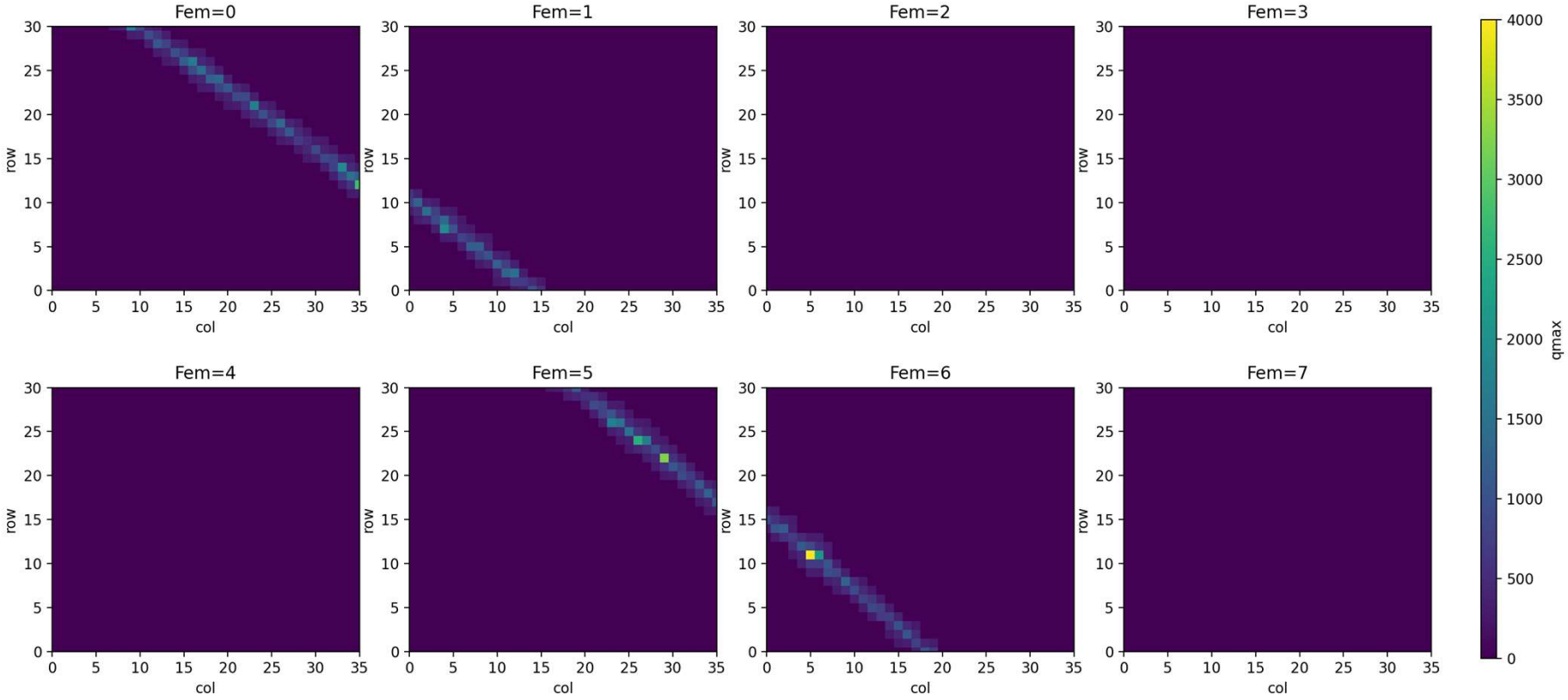


Fem distribution and Image building

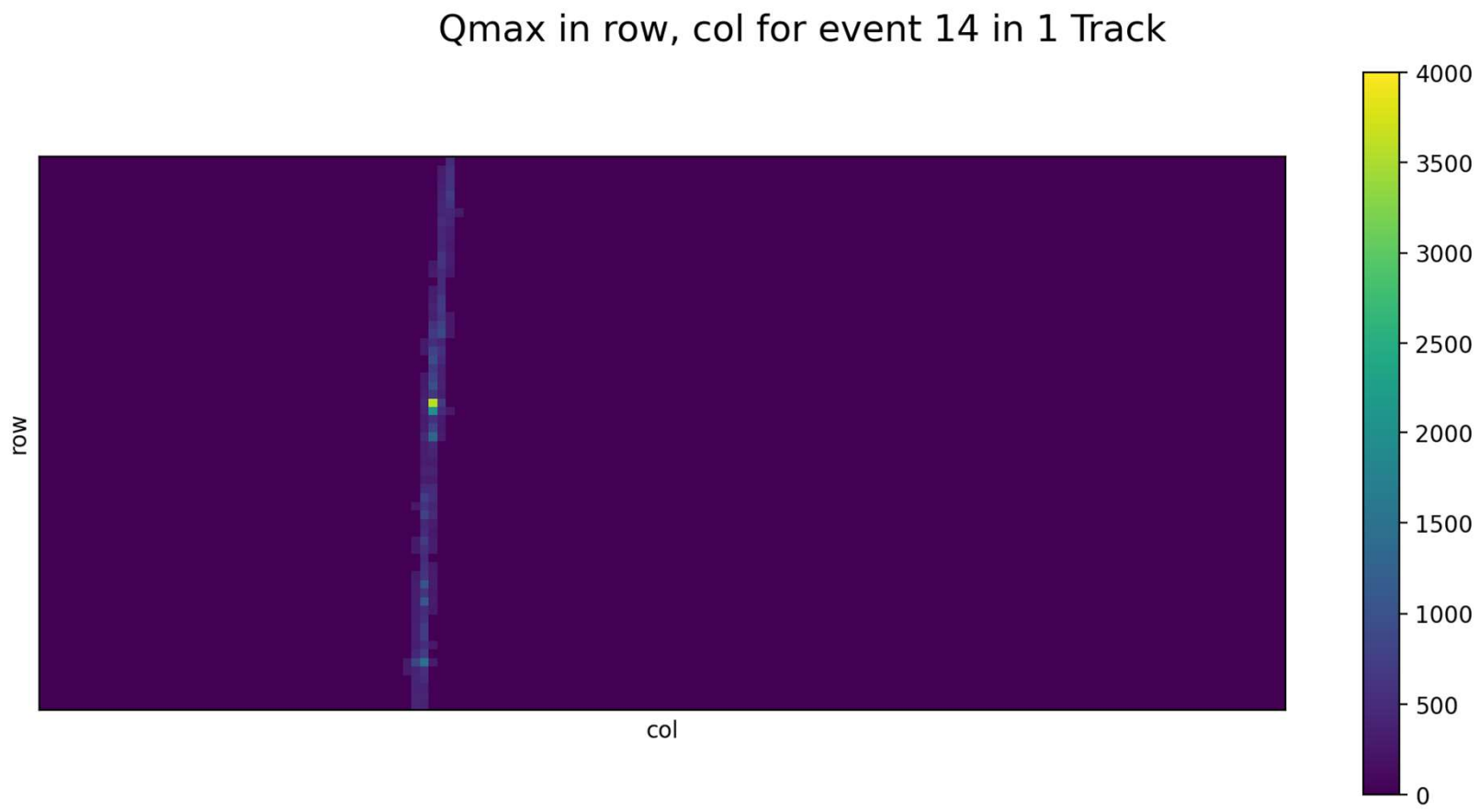


2 Hatdigits

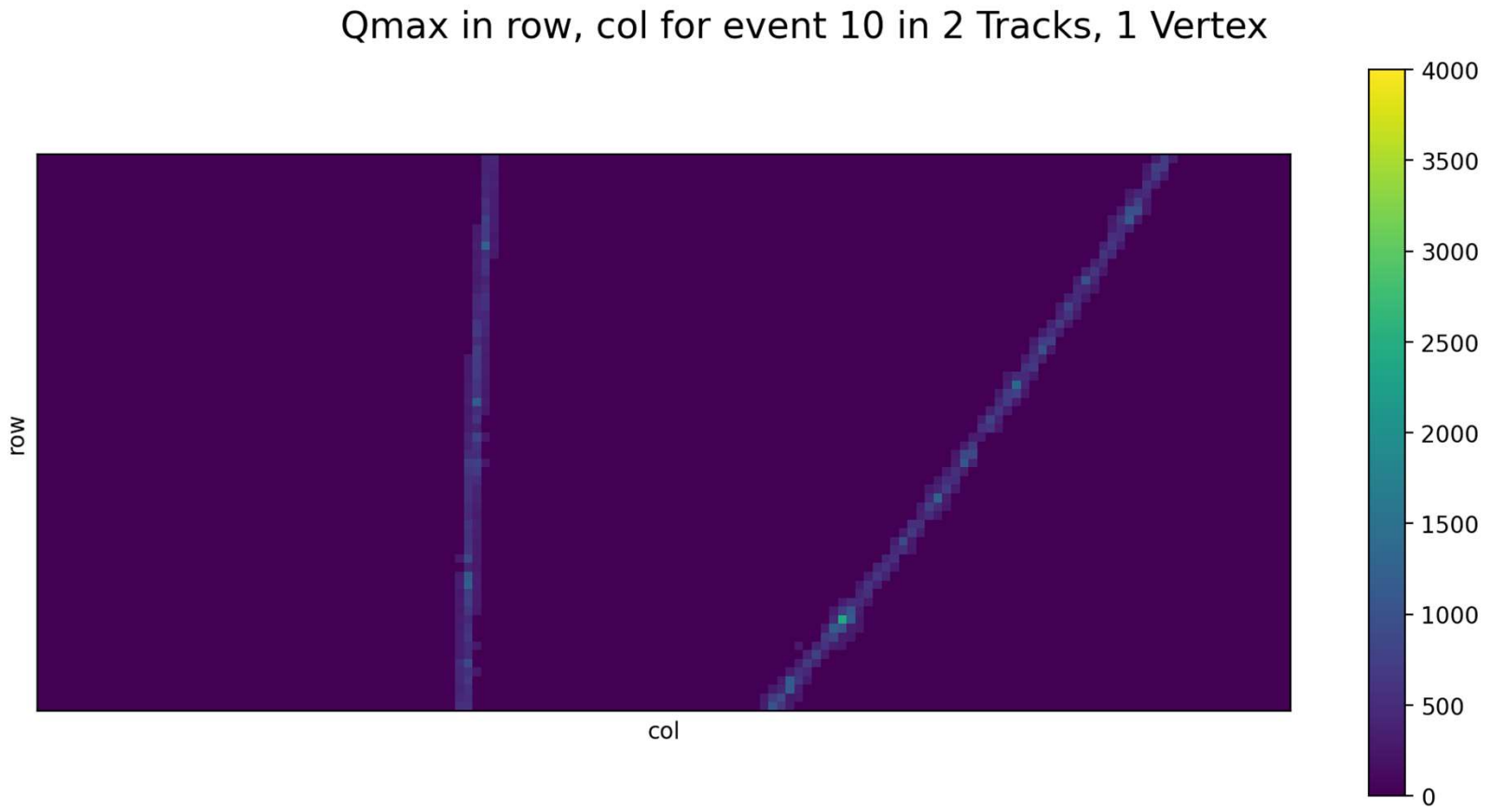
Fem distribution and Image building



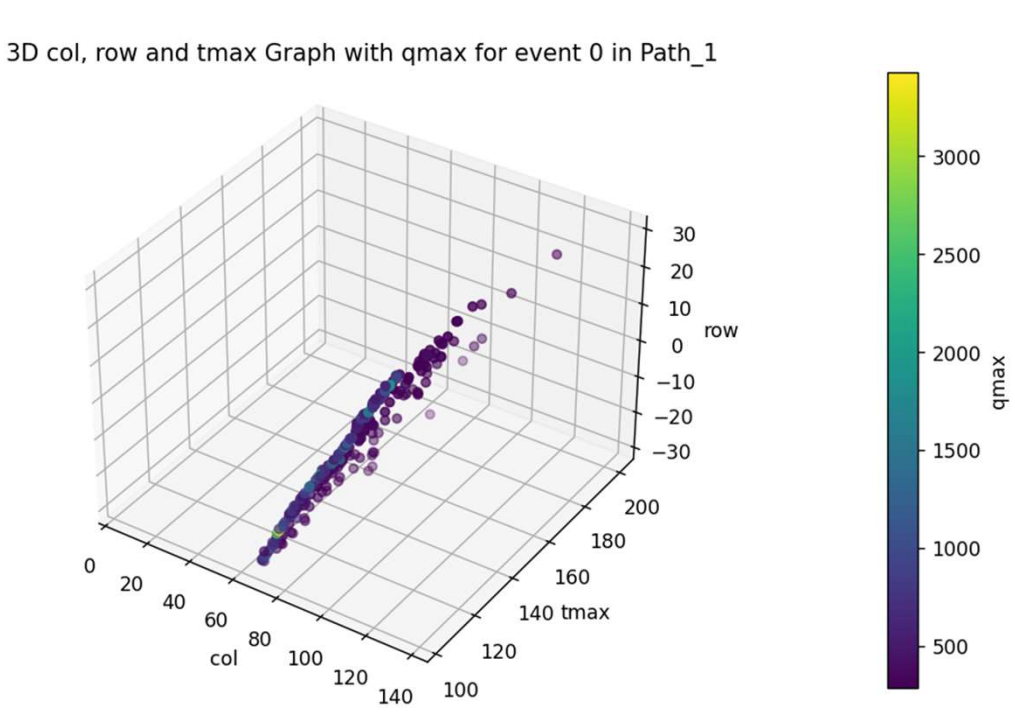
Row, Col Graph with qmax for 1 particle event

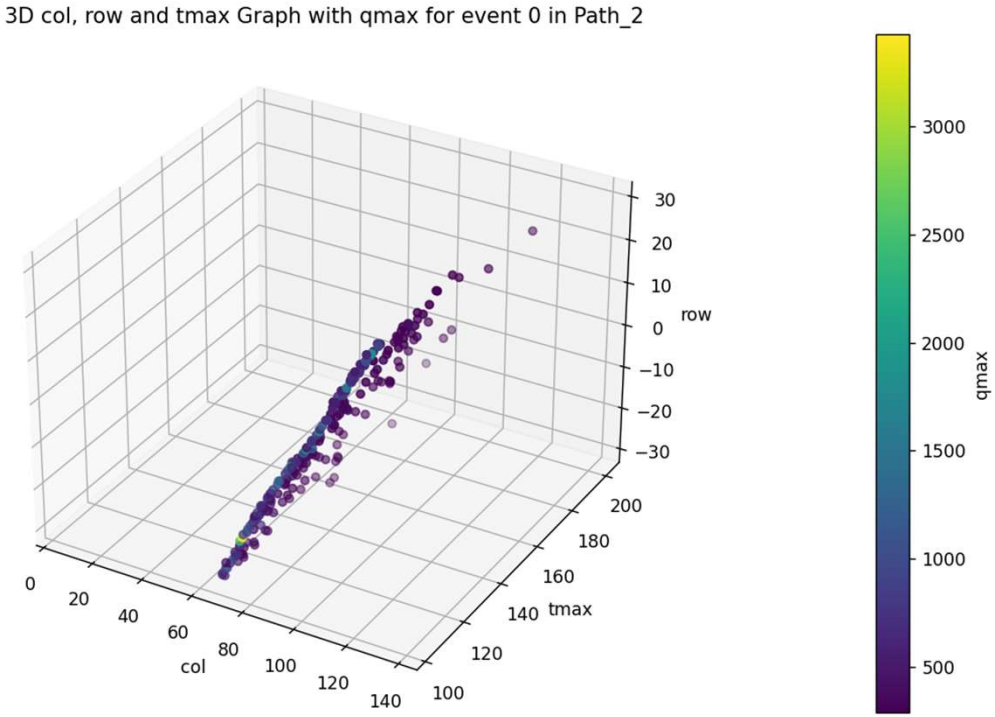


Row, Col Graph with qmax for 2 particle event



3D Row, Col, tmax Graph with qmax





3 Preparing the Input for CNN

Data Structure

Simplify the input: instead of storing all the 0s in an array, store only the pads with the signal in a [row, col, qmax, tmax] format.

We want to difference between:

- path_1 corresponds to events with a single particle involved
- path_2 corresponds to events with two particles involved

3 Preparing the Input for CNN

Step 1

Define path_1 and path_2 dataframes and assign the label column ('npart'):

	event	nhits	[row, col, qmax, tmax]	npart
0	0	223	[[4, 24, 292, 154], [4, 25, 650, 113], [4, 26,...	1
1	1	207	[[4, 8, 293, 64], [0, 5, 671, 48], [0, 6, 1093...	1
2	2	217	[[2, 16, 300, 81], [2, 17, 450, 56], [1, 16, 3...	1
3	3	207	[[0, 26, 328, 59], [9, 34, 287, 100], [9, 35, ...	1
4	4	215	[[4, 8, 477, 64], [5, 8, 297, 80], [0, 4, 692,...	1
...	...	...	...	...
9994	9994	223	[[4, 34, 287, 136], [4, 35, 526, 90], [3, 33, ...	1
9995	9995	210	[[6, 17, 298, 125], [5, 16, 291, 123], [5, 17,...	1
9996	9996	197	[[8, 25, 373, 97], [8, 26, 714, 91], [9, 25, 2...	1
9997	9997	223	[[4, 26, 300, 113], [0, 23, 628, 90], [0, 24, ...	1
9998	9998	222	[[8, 8, 336, 79], [4, 4, 289, 97], [4, 5, 504,...	1

9999 rows × 4 columns

	event	nhits	[row, col, qmax, tmax]	npart
0	0	222	[[2, 35, 316, 88], [1, 34, 326, 94], [1, 35, 7...	2
1	1	147	[[4, 26, 292, 159], [0, 23, 872, 119], [0, 24,...	2
2	2	217	[[4, 5, 313, 92], [4, 6, 420, 75], [4, 7, 428,...	2
3	3	224	[[4, 23, 304, 126], [4, 24, 630, 93], [4, 25, ...	2
4	4	226	[[8, 5, 318, 85], [8, 6, 665, 49], [8, 7, 1022...	2
...	...	...	...	...
9994	9994	203	[[0, 24, 352, 62], [0, 25, 589, 60], [0, 26, 2...	2
9995	9995	120	[[6, 35, 289, 129], [5, 34, 378, 125], [5, 35,...	2
9996	9996	223	[[4, 26, 352, 68], [0, 22, 298, 80], [0, 23, 7...	2
9997	9997	211	[[0, 24, 422, 102], [0, 25, 608, 105], [0, 26,...	2
9998	9998	223	[[4, 8, 290, 141], [0, 5, 564, 105], [0, 6, 77...	2

9999 rows × 4 columns

3 Preparing the Input for CNN

Step 2

Merge both dataframes into one:

	event	nhits	[row, col, qmax, tmax]	npart
0	0	223	[[4, 24, 292, 154], [4, 25, 650, 113], [4, 26,...	1
1	1	207	[[4, 8, 293, 64], [0, 5, 671, 48], [0, 6, 1093...	1
2	2	217	[[2, 16, 300, 81], [2, 17, 450, 56], [1, 16, 3...	1
3	3	207	[[0, 26, 328, 59], [9, 34, 287, 100], [9, 35, ...	1
4	4	215	[[4, 8, 477, 64], [5, 8, 297, 80], [0, 4, 692,...	1
...	...	...	...	...
19993	9994	203	[[0, 24, 352, 62], [0, 25, 589, 60], [0, 26, 2...	2
19994	9995	120	[[6, 35, 289, 129], [5, 34, 378, 125], [5, 35,...	2
19995	9996	223	[[4, 26, 352, 68], [0, 22, 298, 80], [0, 23, 7...	2
19996	9997	211	[[0, 24, 422, 102], [0, 25, 608, 105], [0, 26,...	2
19997	9998	223	[[4, 8, 290, 141], [0, 5, 564, 105], [0, 6, 77...	2

19998 rows × 4 columns

3 Preparing the Input for CNN

Step 3

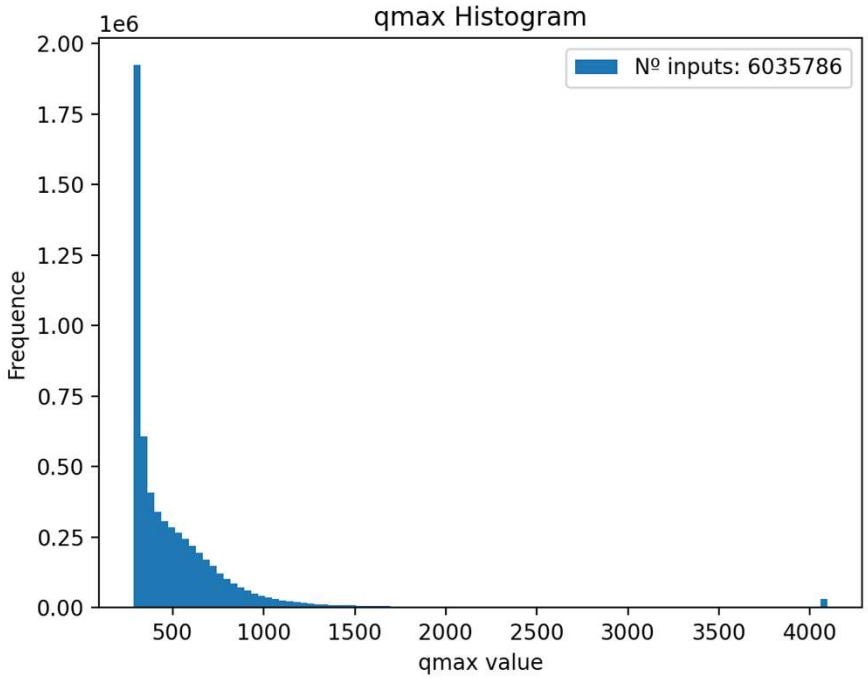
Randomize the inputs entries:

	event	nhits	[row, col, qmax, tmax]	npart
0	3823	223	[[6, 35, 327, 62], [5, 34, 365, 70], [5, 35, 1...	1
1	6068	219	[[5, 35, 584, 120], [4, 33, 313, 129], [4, 34,...	1
2	1862	232	[[0, 26, 606, 89], [1, 26, 287, 119], [9, 34, ...	1
3	7765	242	[[0, 24, 293, 166], [0, 25, 1539, 120], [0, 26...	2
4	8831	224	[[3, 35, 387, 111], [2, 34, 429, 108], [2, 35,...	1
...	...	...	...	...
19993	9633	215	[[6, 34, 294, 91], [6, 35, 502, 75], [5, 33, 2...	1
19994	4481	220	[[0, 25, 378, 120], [0, 26, 543, 119], [1, 26,...	2
19995	5707	226	[[4, 7, 503, 47], [4, 8, 606, 47], [5, 7, 290,...	1
19996	7614	212	[[0, 26, 868, 90], [1, 26, 341, 95], [10, 35, ...	2
19997	6534	230	[[4, 5, 335, 124], [4, 6, 811, 120], [4, 7, 46...	2

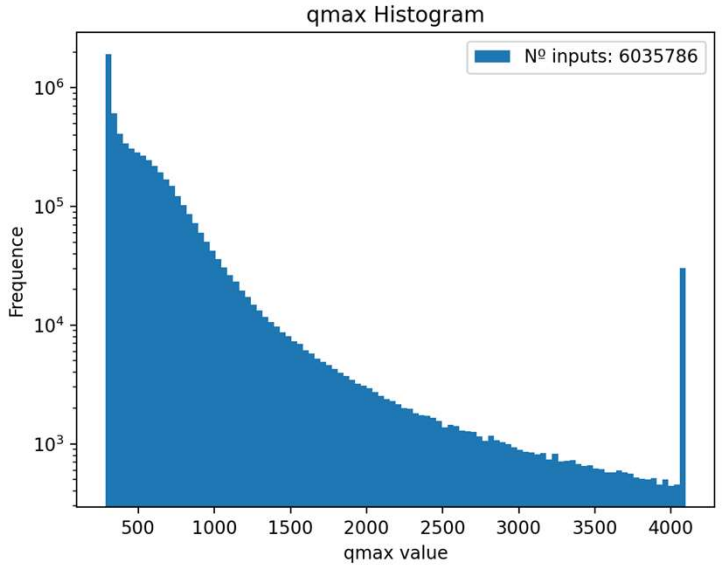
19998 rows × 4 columns

4 Simple ANN to discriminate 1 and 2 particles event

Qmax Histogram

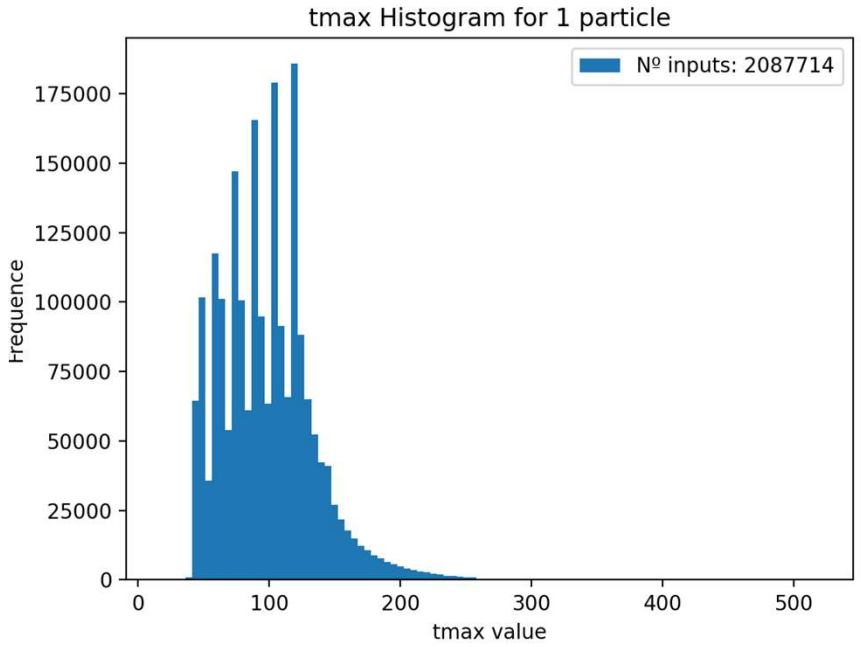


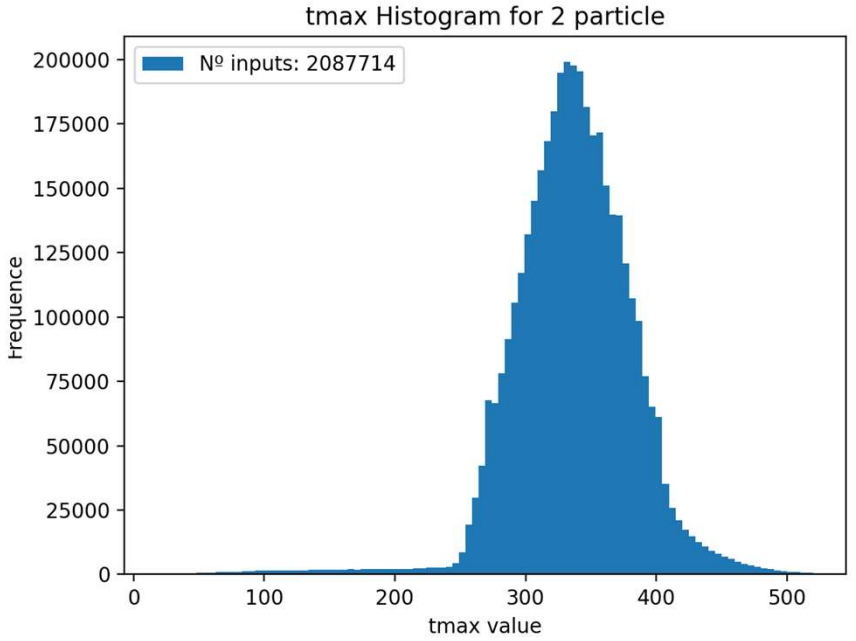
Logarithmic representation



4 Simple ANN to discriminate 1 and 2 particles event

Tmax Histogram

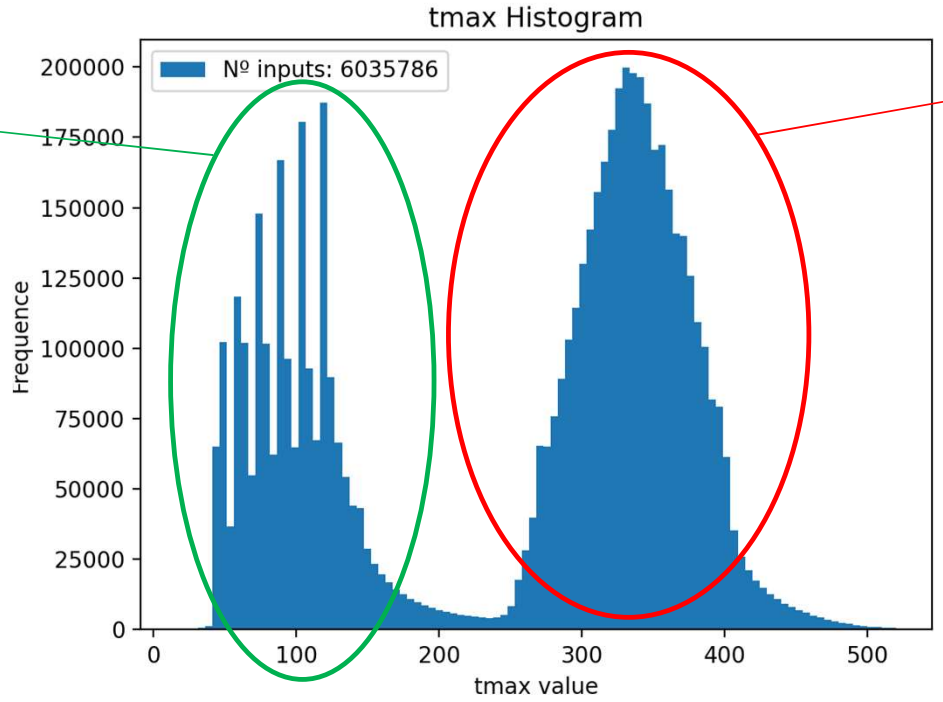




4 Simple ANN to discriminate 1 and 2 particles event

Tmax Histogram

1 particle

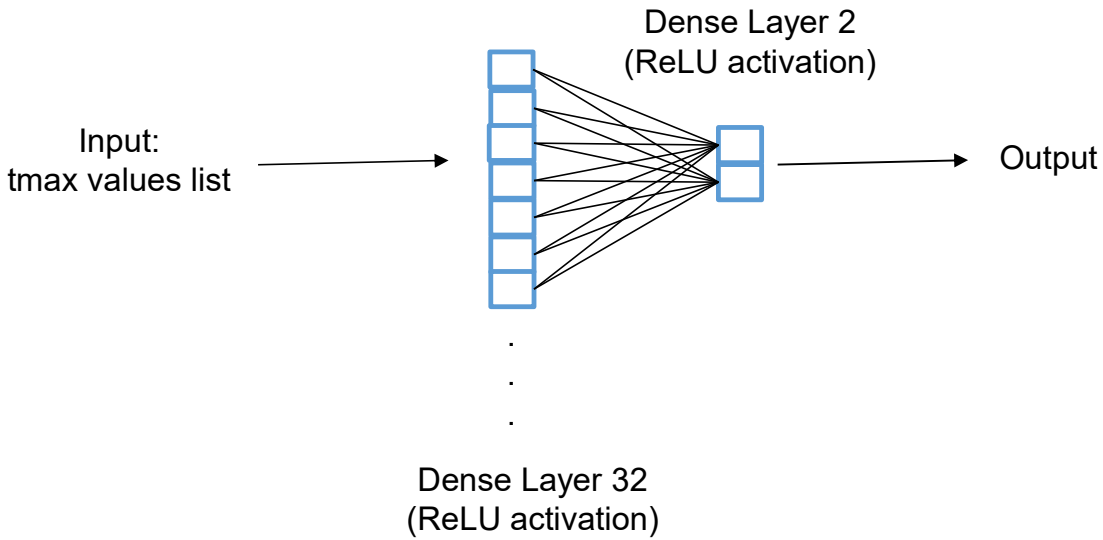


2 particles

There seems to be a clear differentiation between tmax values and whether they correspond to 1 or 2 particle events, so a simple ANN with tmax values as input can be used to discriminate between 1 and 2 particle events.

4 Simple ANN to discriminate 1 and 2 particles event

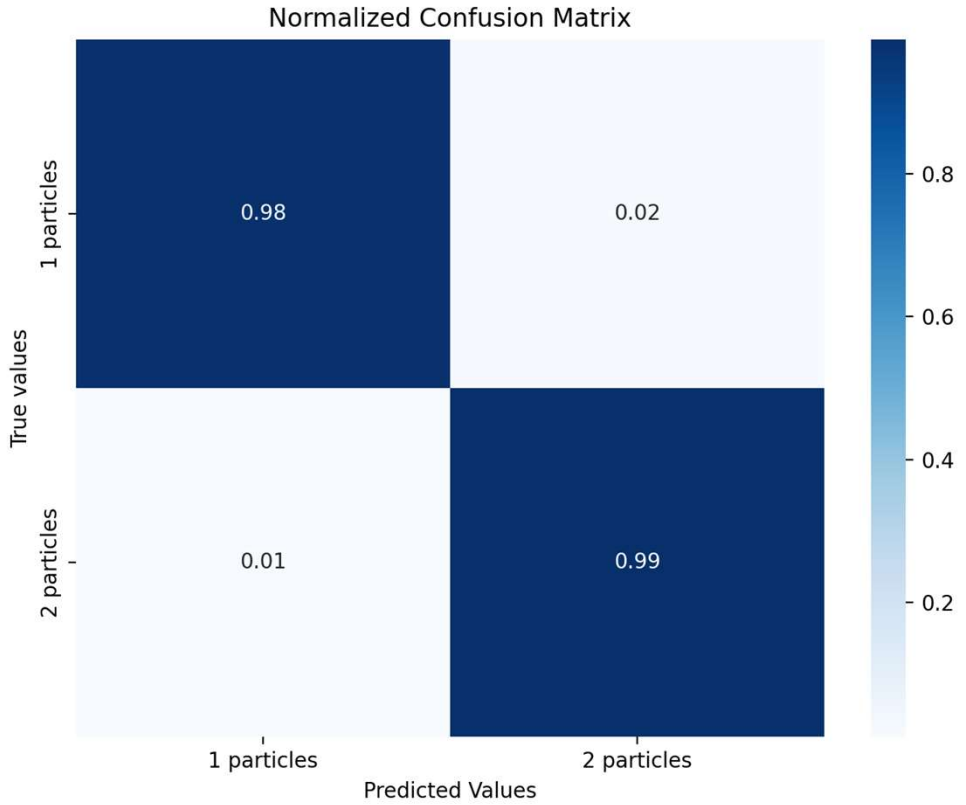
ANN architecture




```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

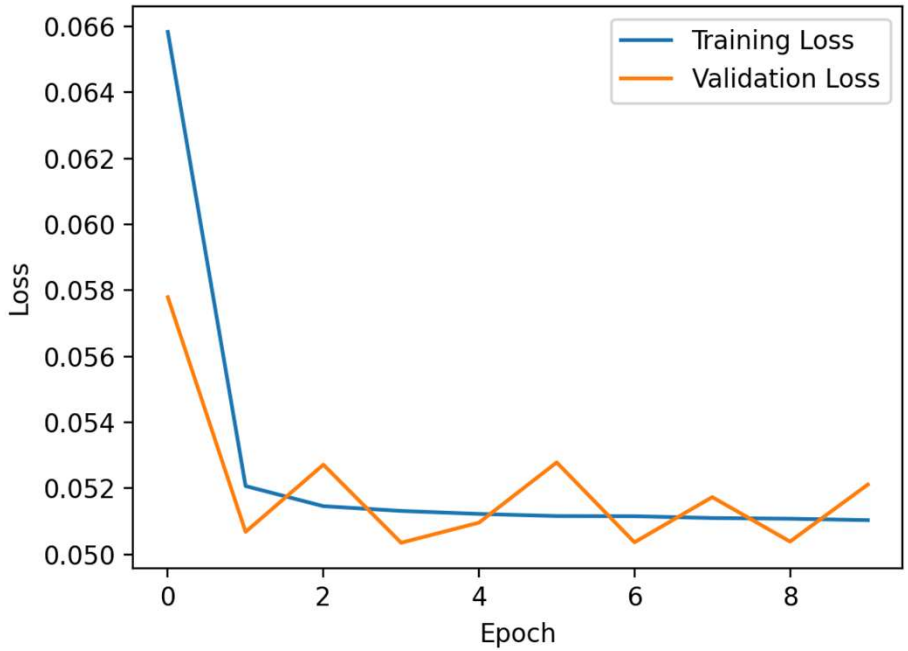
4 Simple ANN to discriminate 1 and 2 particles event

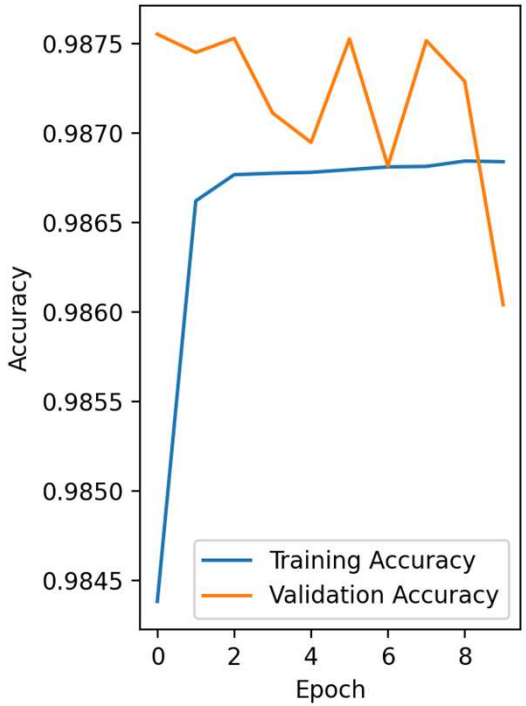
Result



4 Simple ANN to discriminate 1 and 2 particles event

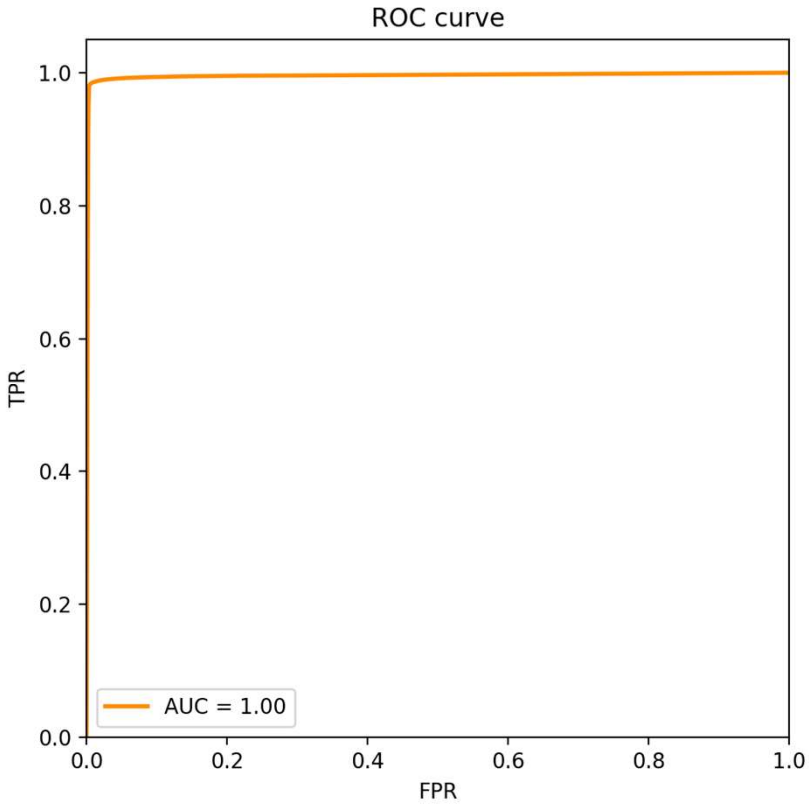
Result





4 Simple ANN to discriminate 1 and 2 particles event

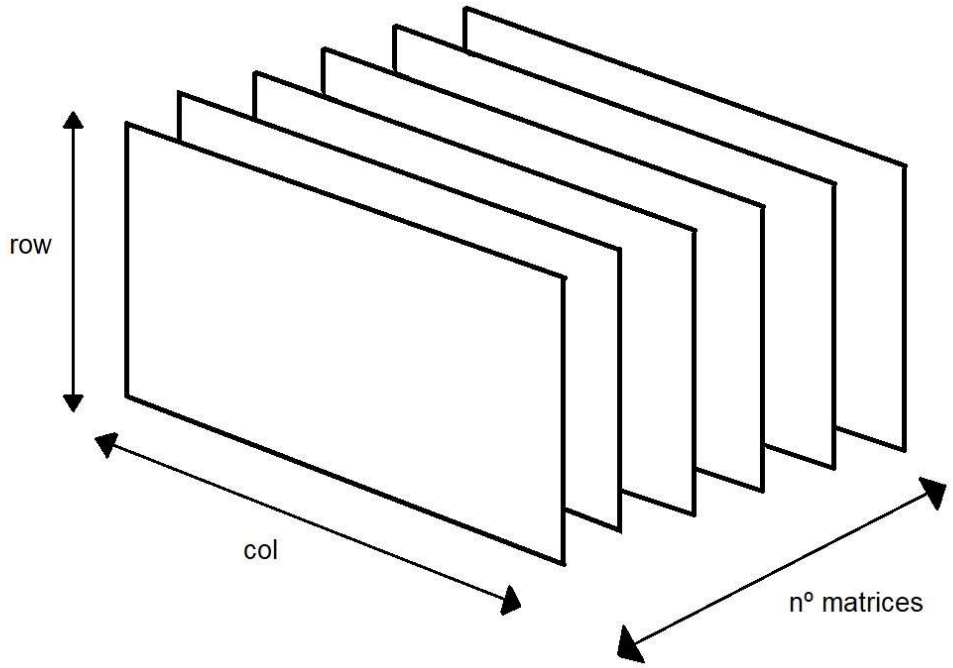
Result



5 1 and 2 particles event discrimation using a CNN

Data Structure

- Input:



A tensor is built with the input images

Dimensions: $(n^{\underline{o}}\ matrices \times row \times col)$

↓

A dimension referring to the color channel of the images (1 for gray, 3 for RGB) has to be added to the tensors:

$(n^{\underline{o}}\ matrices \times row \times col \times n^{\underline{o}}\ color\ channels)$

(1 color channel has been used)

5 1 and 2 particles event discrimation using a CNN

Data Structure

- Output:

There are two classes (1 and 2 particles), to use the CNN we have to convert 1s to 0s, and 2s to 1s:

$$output = output\ - 1$$

To correctly implement the output in the CNN, we have to convert the dependent variable into integers to a binary class matrix of dimension:

$$(n^{\underline{o}}\ outputs, 2)$$

$n^{\underline{o}}\ outputs = n^{\underline{o}}\ labels$

2 classes

Data distribution

Data is divided into:

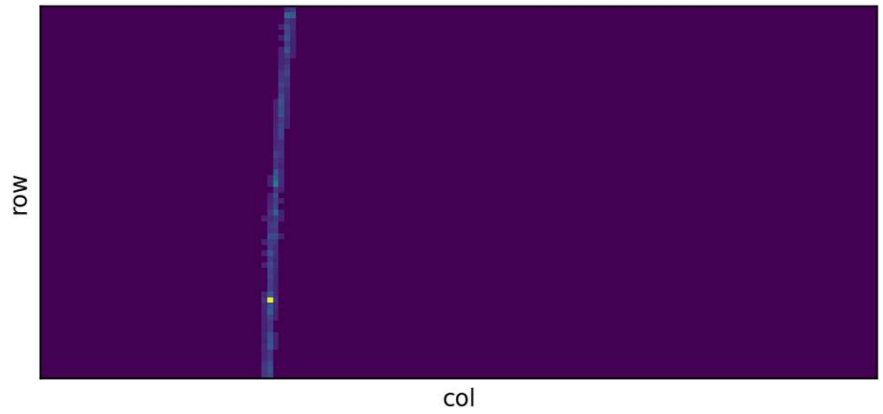
- Training Data (~70%): x_train (input) and y_train (output)
- Test Data (~30%): x_test (input) and y_test (output)

5 1 and 2 particles event discrimation using a CNN

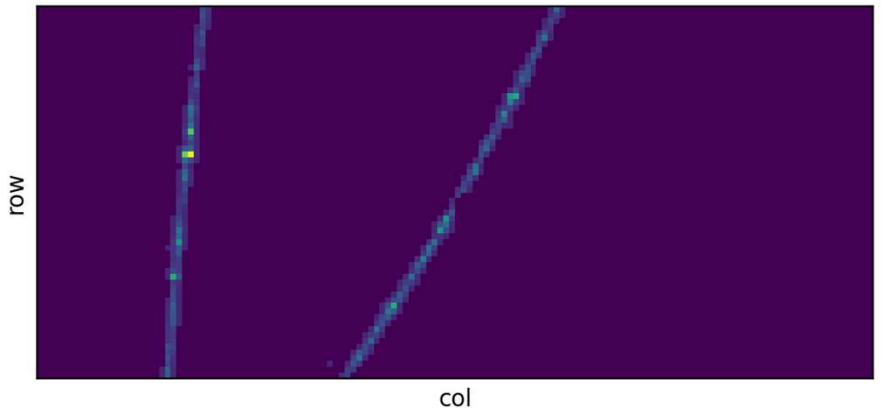
Data Structure

These are the types of images we want CNN to differentiate:

1 particle event

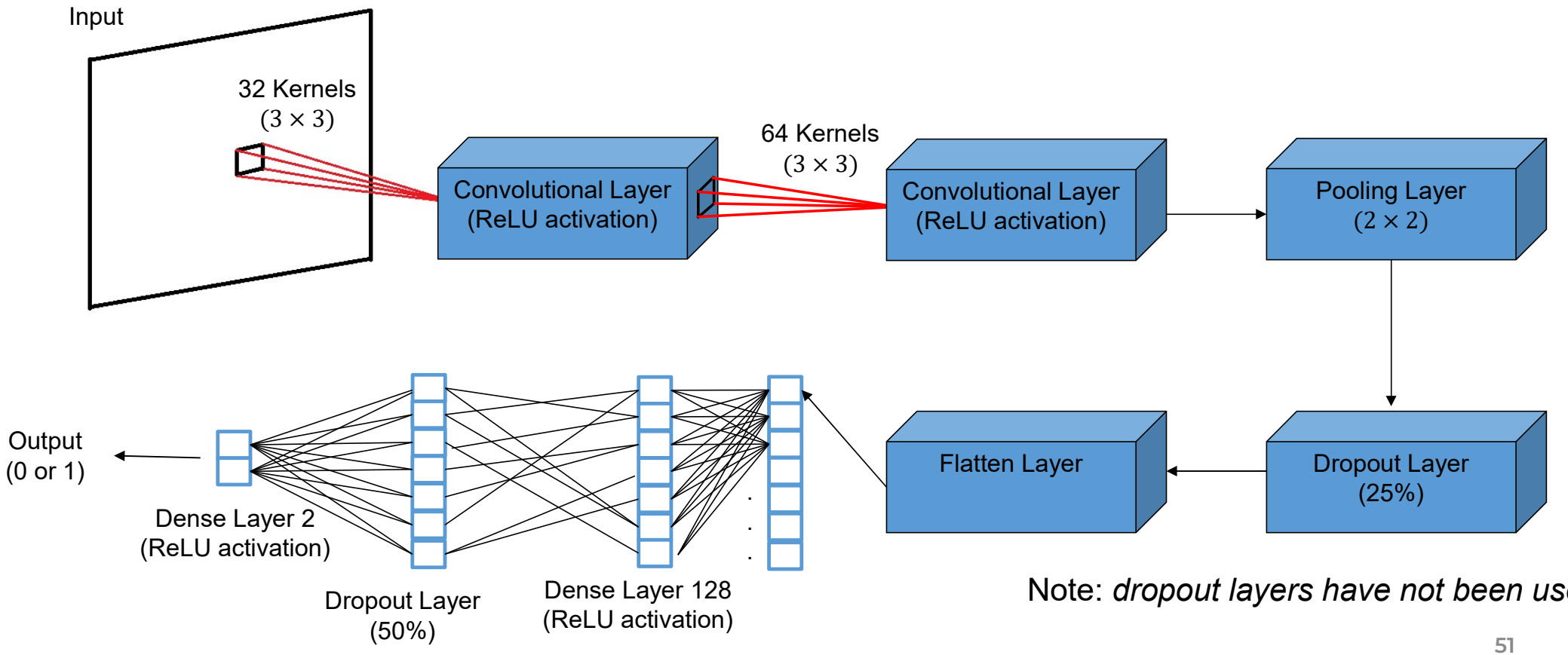


2 particle event



5 1 and 2 particles event discrimation using a CNN

CNN Structure



Note: *dropout layers have not been used*

5 1 and 2 particles event discrimation using a CNN

Loss and optimizer function

```
model.compile(loss='mean_squared_error',
      optimizer='adam',
      metrics=['accuracy'])
```

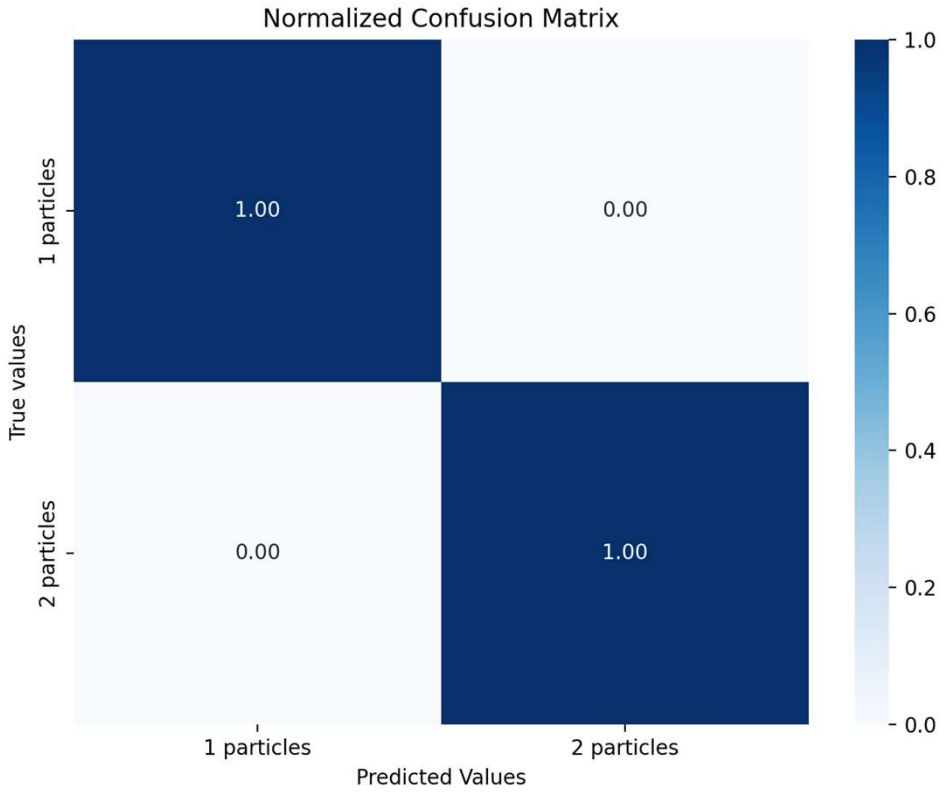
- Loss function: Mean Squared Error (MSE)

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \hat{y}\right)^2}_{\text{The square of the difference between actual and predicted}}$$

- Optimizer: adam, a stochastic gradient descent method based on first and second order moments

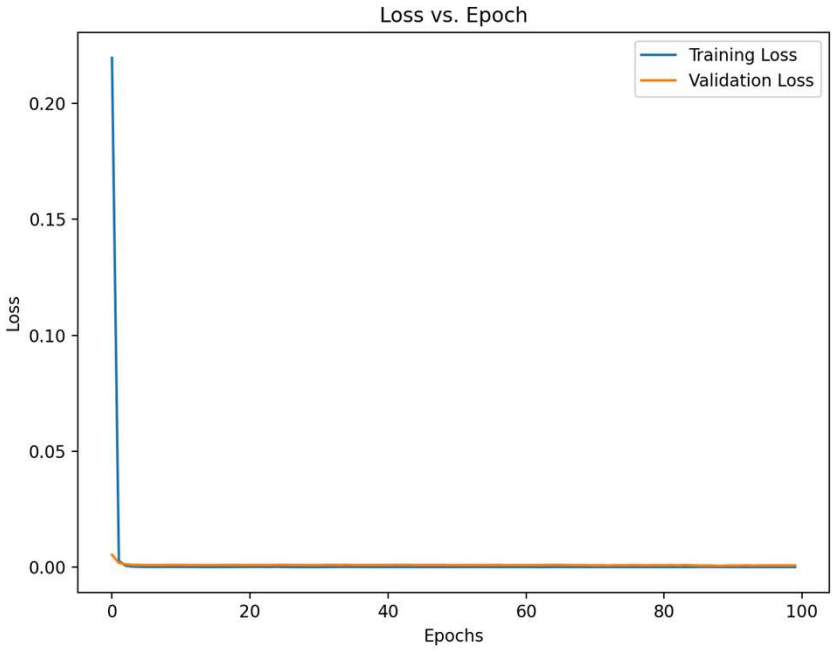
5 1 and 2 particles event discrimation using a CNN

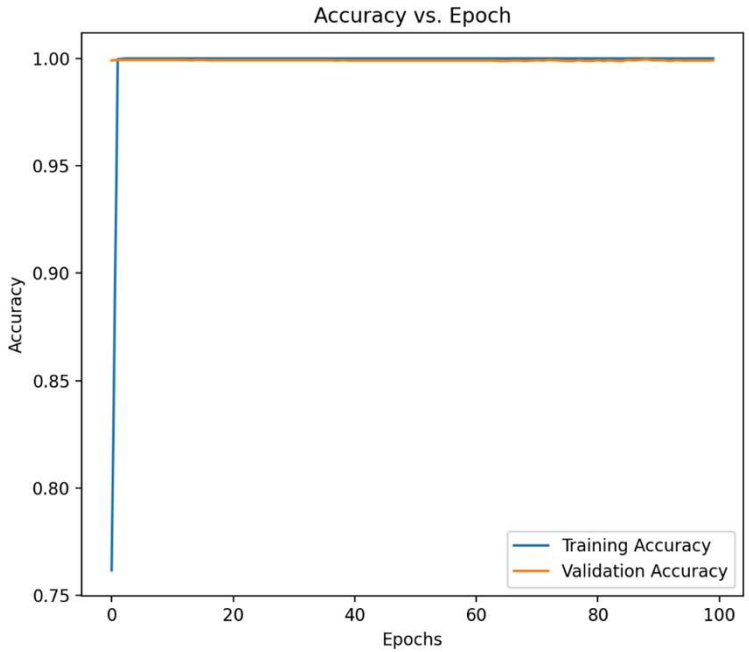
Results



5 1 and 2 particles event discrimation using a CNN

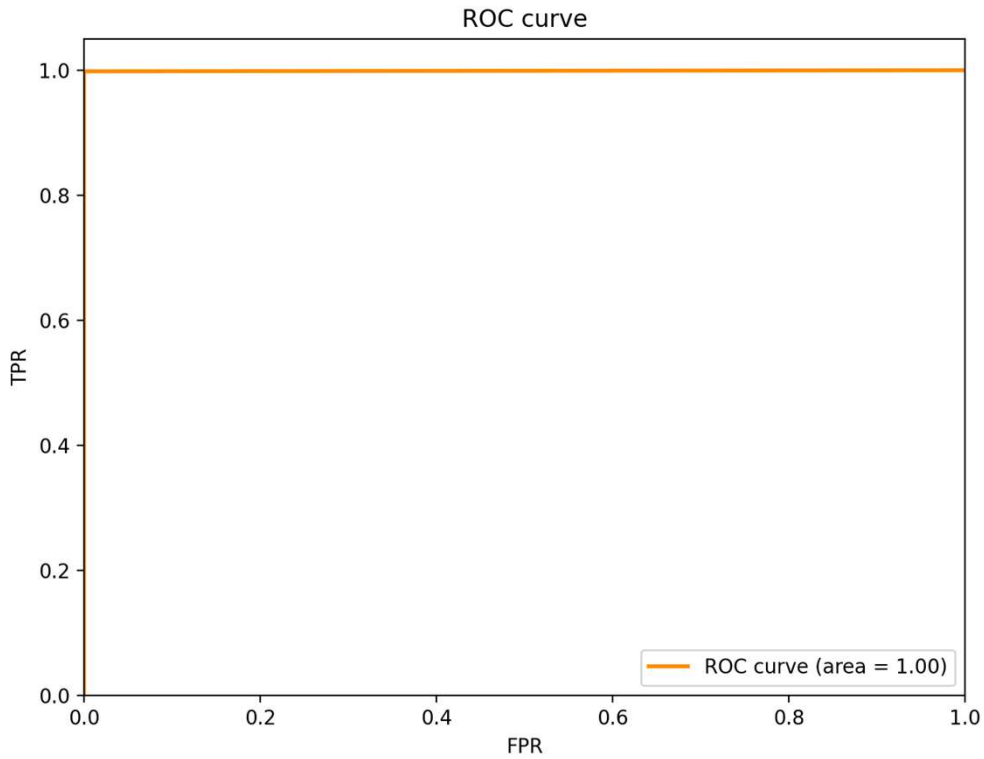
Results





5 1 and 2 particles event discrimation using a CNN

Results



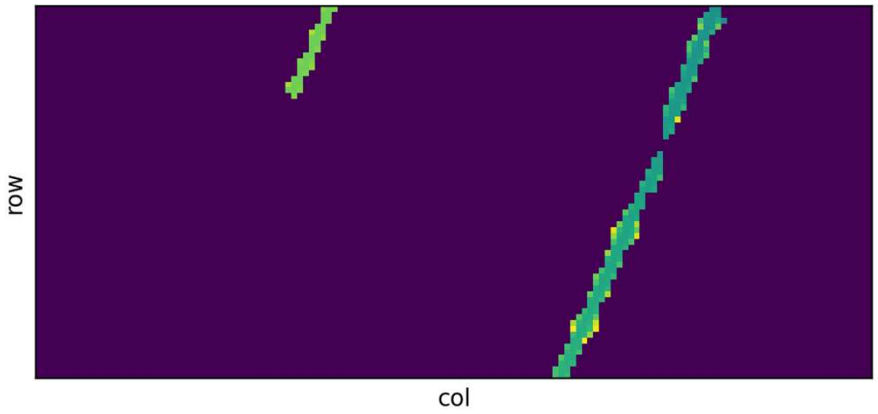
$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}$$

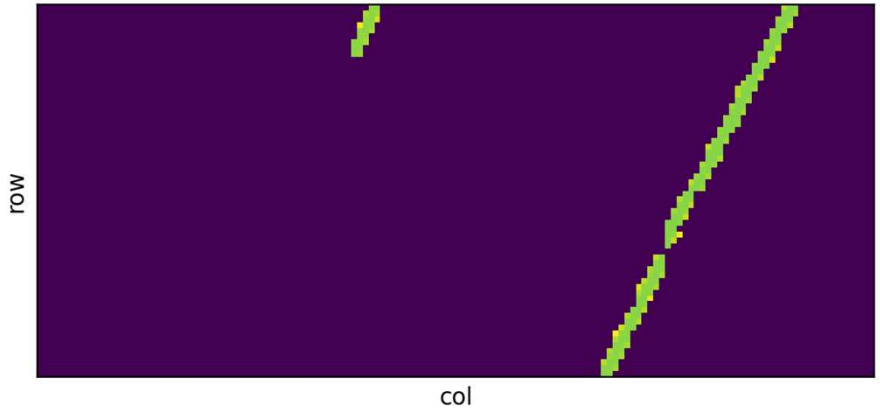
Particle discrimination using a CNN

The idea is to reuse the CNN to discriminate between protons (p+) and muons (μ+)

p+



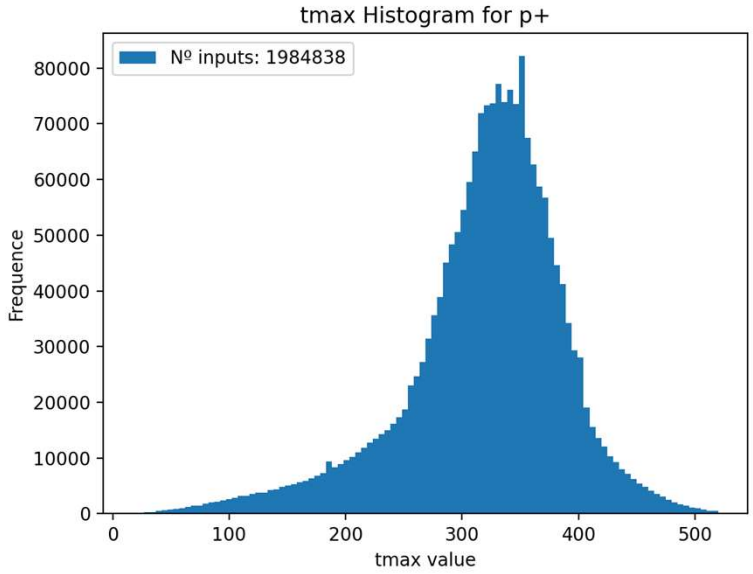
μ+

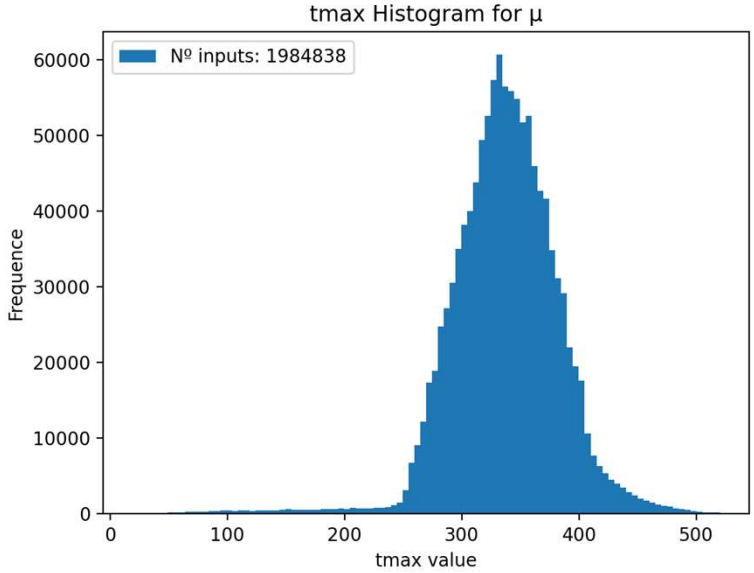


- Input: proton and muon images ⟶ tensor with tmax information
- Output: type of particle ⟶ 0 or 1

6 Next Steps: particle discrimination using a CNN

Tmax Histograms

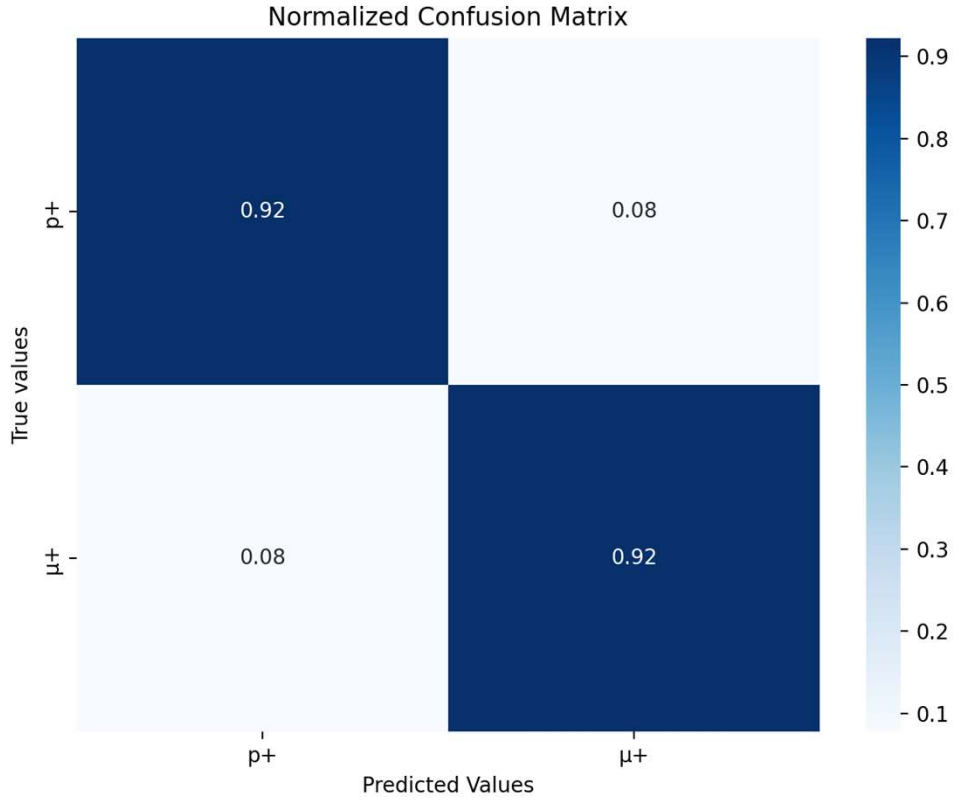




In this case, there is no clear differentiation between tmax values for protons and muons, so using a CNN seems like a good idea.

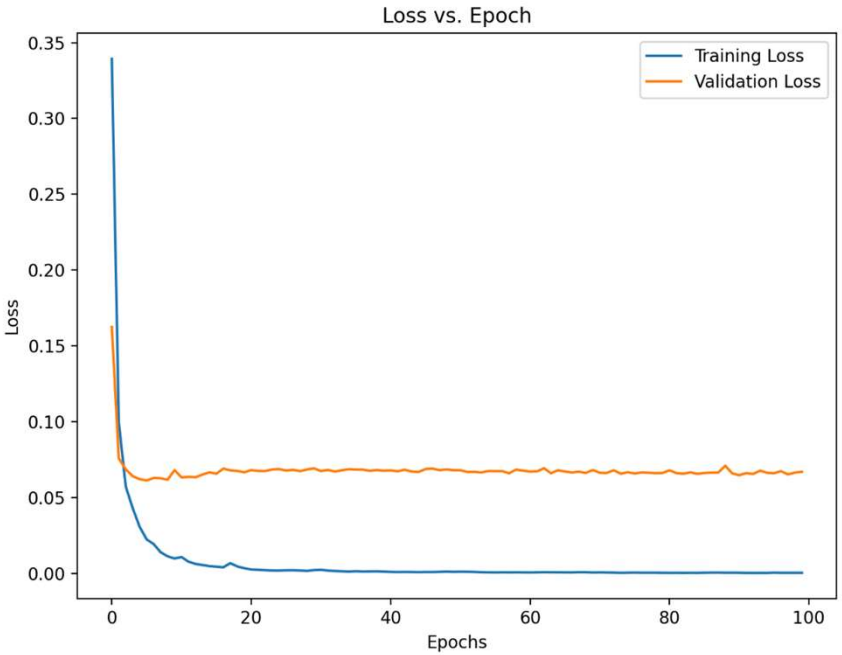
6 Next Steps: particle discrimination using a CNN

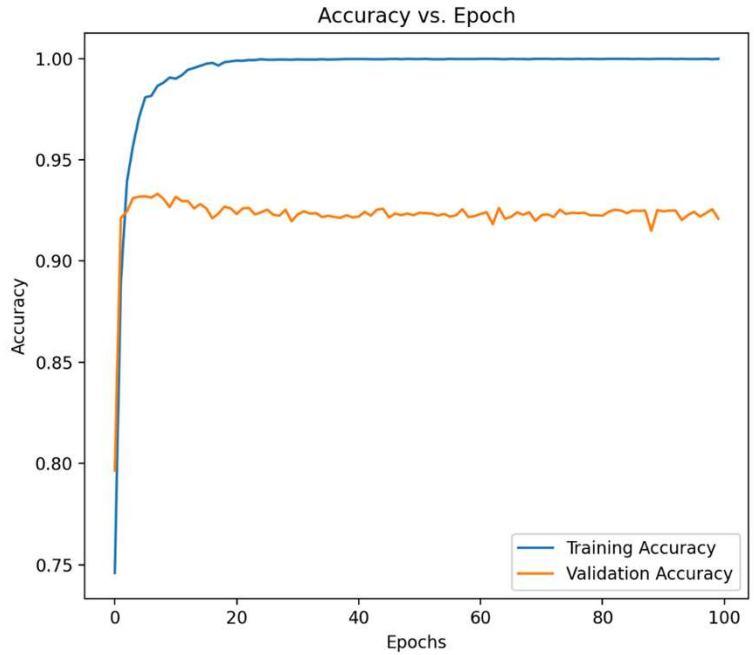
Results



6 Next Steps: particle discrimination using a CNN

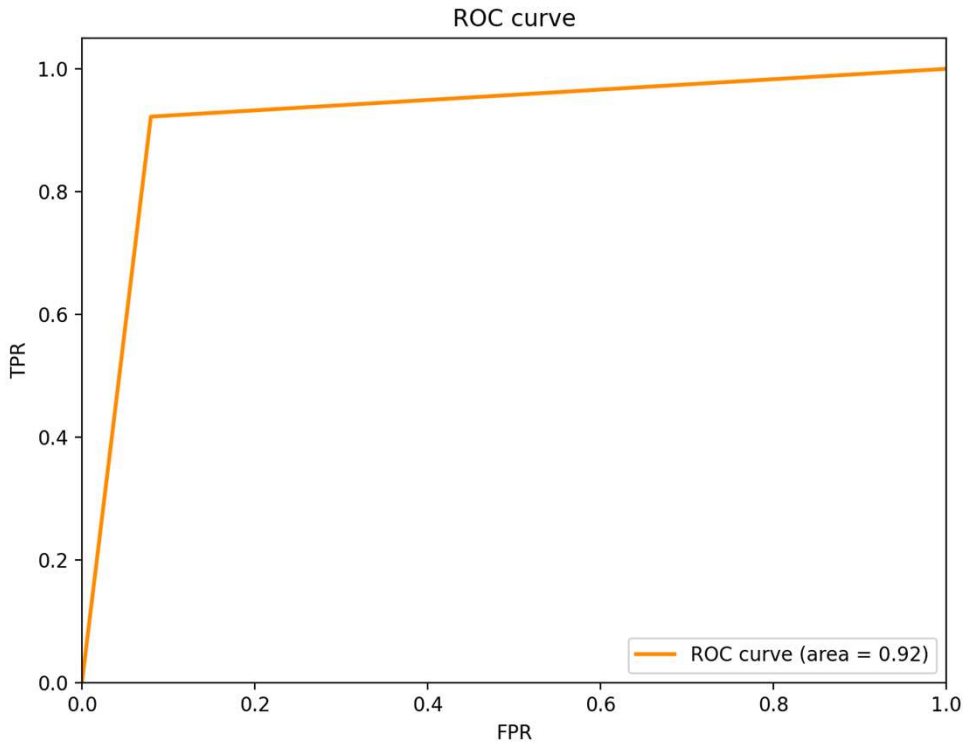
Results





6 Next Steps: particle discrimination using a CNN

Results



6 Next Steps: particle discrimination using a CNN

Next Steps

- Use qmax instead of tmax in the CNNs
  - In the case of CNN for p+ and μ- discrimination it has no physical sense to use tmax, so it should be used qmax
- Use both, qmax and tmax (2 channels) for better performance of the CNNs
- Discriminate between other different particles

References

[1] A Beginner's Guide to Keras: Digit Recognition in 30 Minutes
https://www.sitepoint.com/keras-digit-recognition-tutorial/

[2] Understanding the log loss function
https://medium.com/analytics-vidhya/understanding-the-loss-function-of-logistic-regression-ac1eec2838ce

[3] Adam Keras
https://keras.io/api/optimizers/adam/