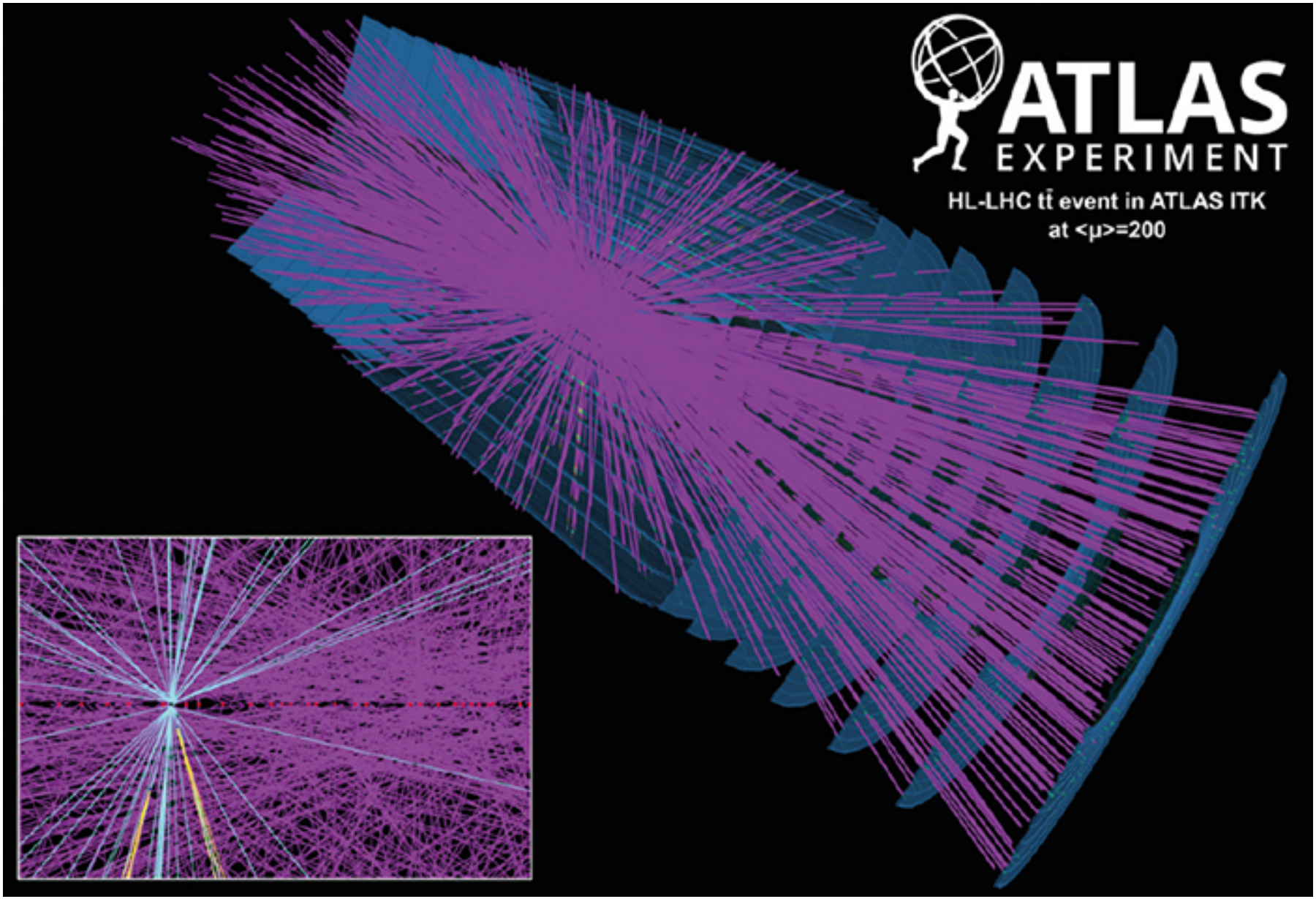
Hashing Classification for charged particle tracking

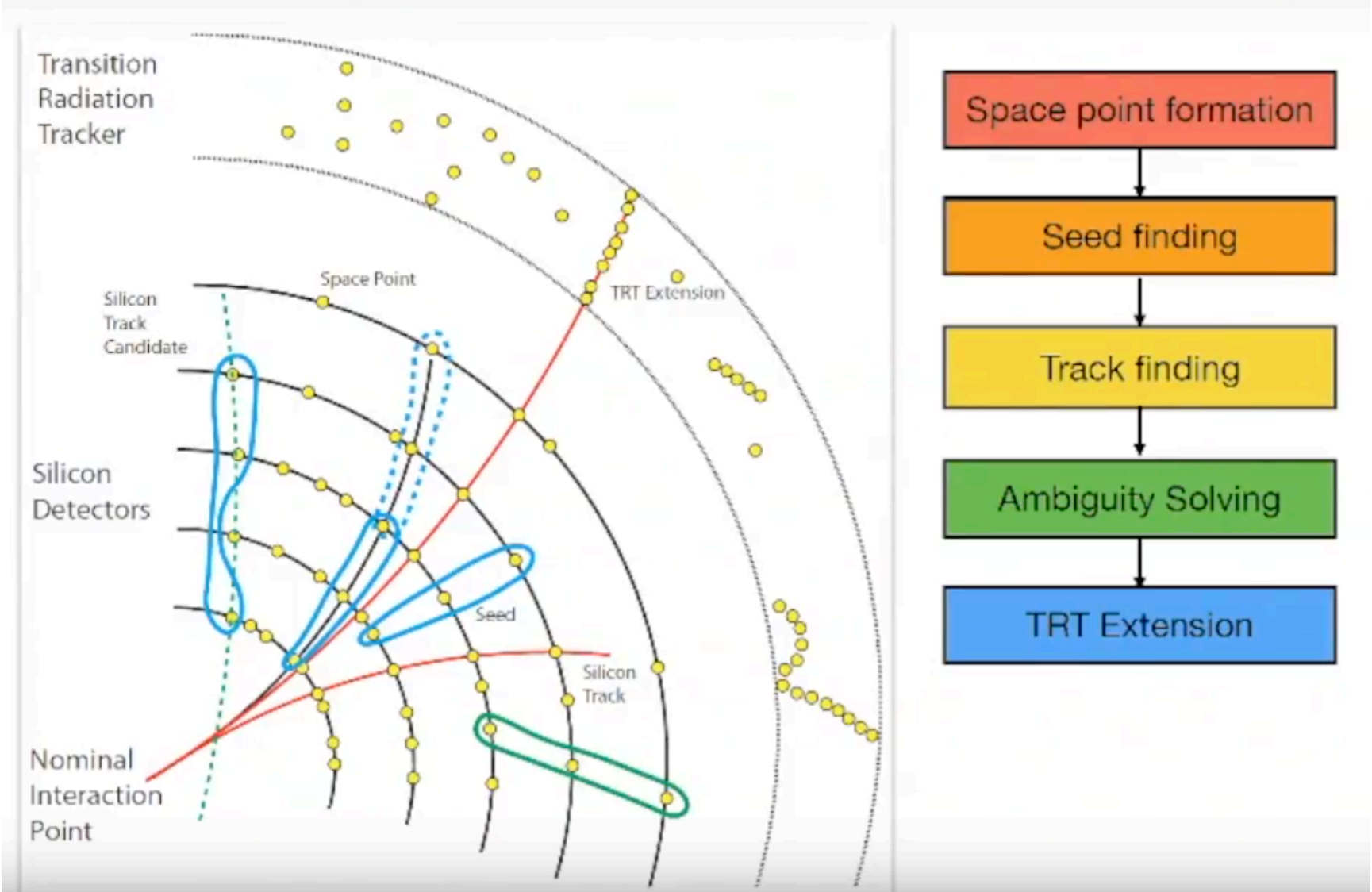
Luiza Adelina Ciucu,

31 March 2020

Because of the large amount of data and event complexity, the traditional data analysis techniques are not enough.

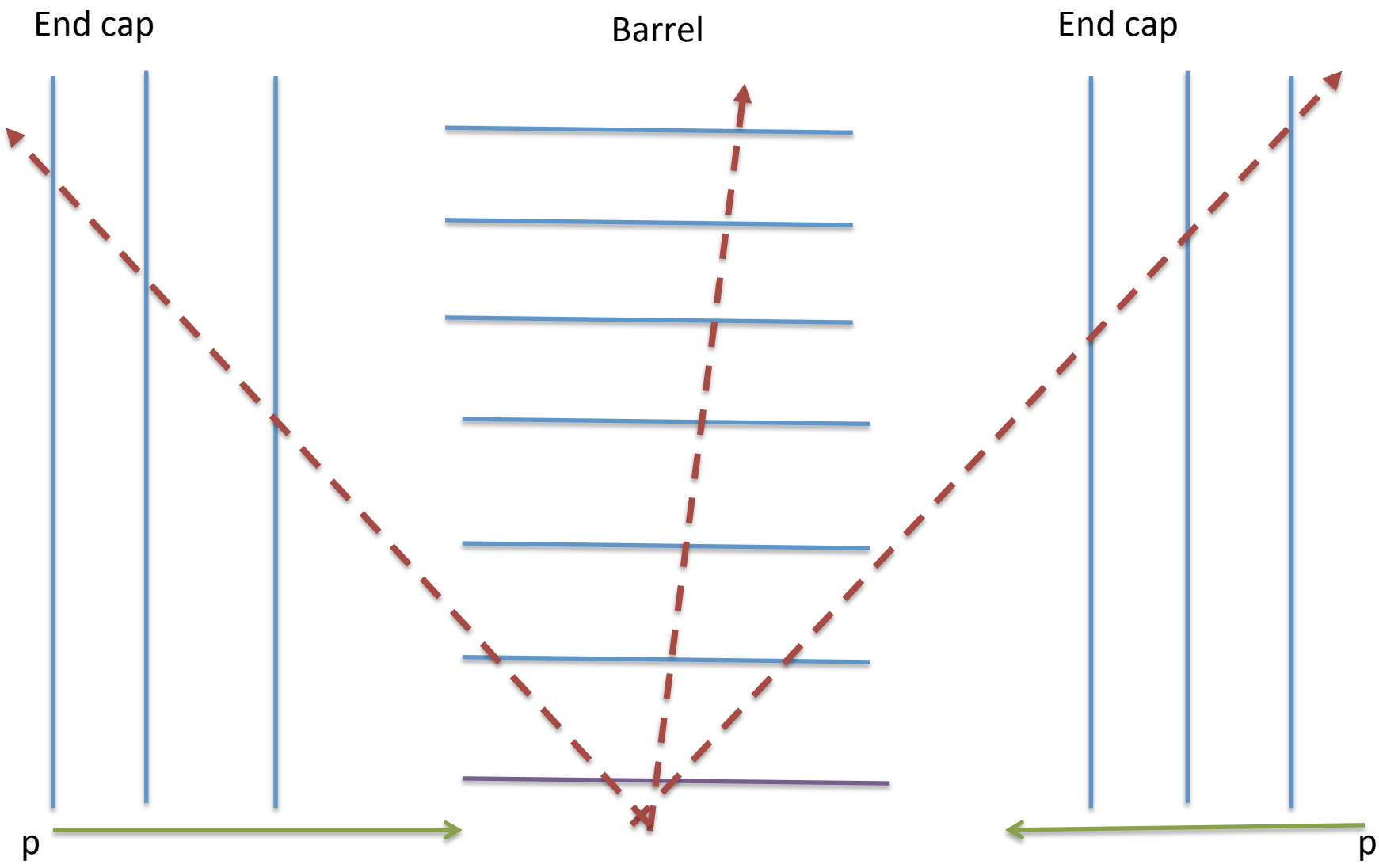


LHC detectors reconstruct and measure the kinematic properties of particles produced in proton-proton collisions. Charged particles ionize matter, producing hits in the various layers of the inner detector. Our task is to reconstruct tracks from hits.

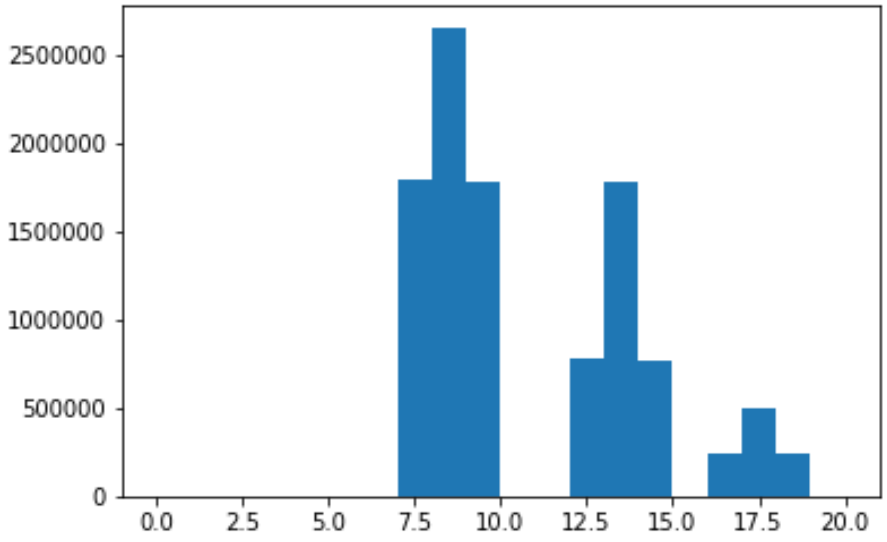


Data Exploration
Defining the problem

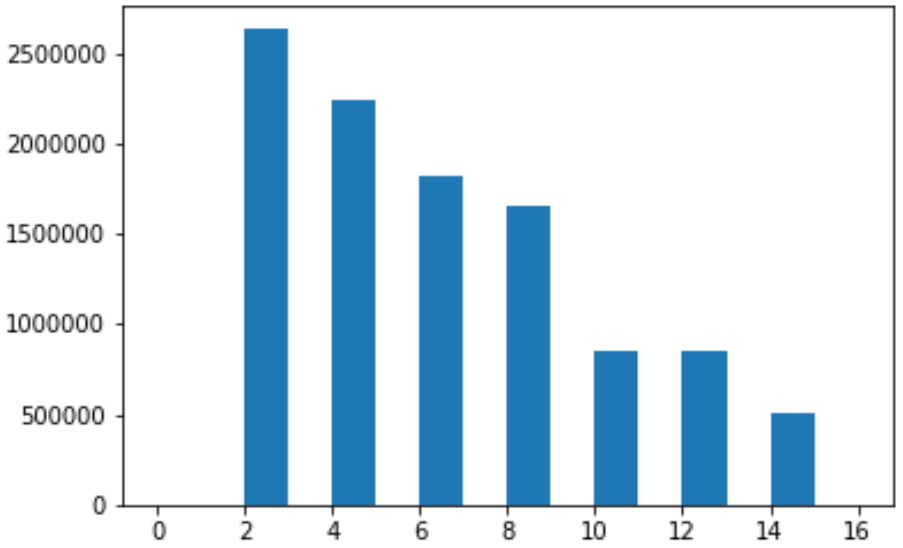
Schematic view of the general inner detector with layers.
Charged particles leave tracks
after starting in the central layer closest to the beam.



Histo counting number of hits for a given volume_id and layer_id.



volume_id



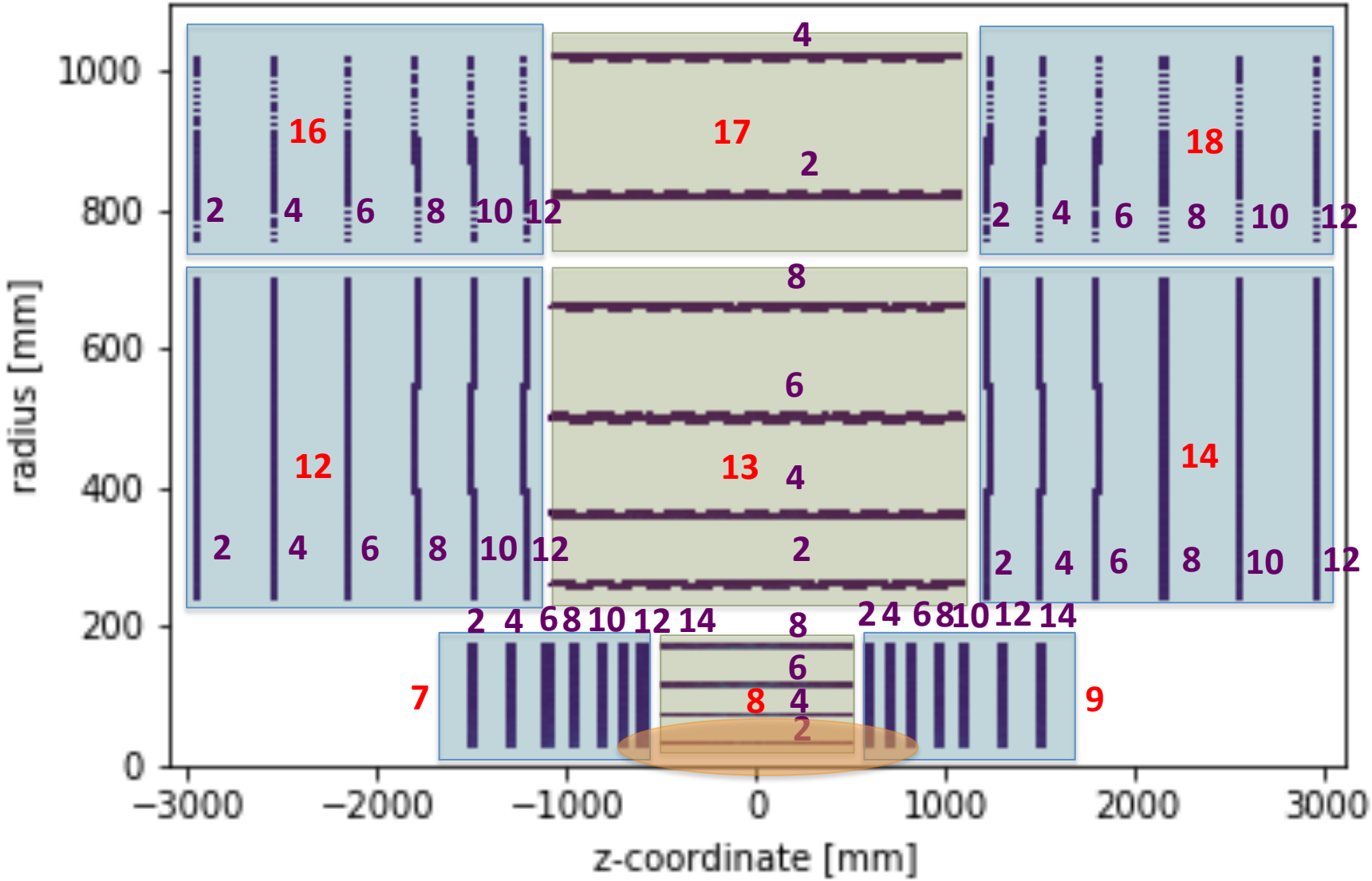
layer_id

Barrel (with horizontal layers). End caps (with vertical layers).

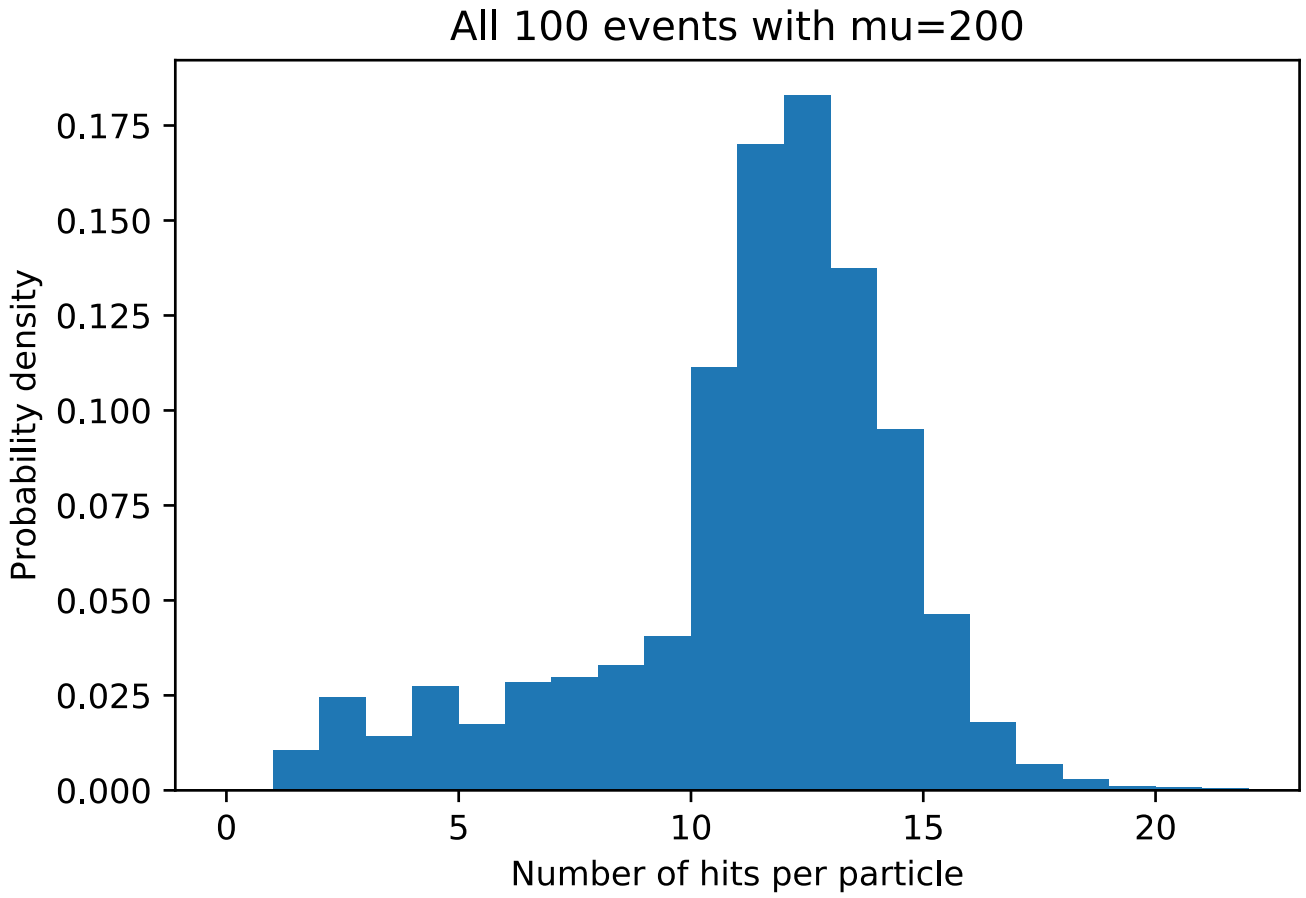
The detector's volume_id in red, and the layer_id in violet.

We want to build a bucket for each hit of volume_id=8 and layer_id=2,

assuming all collisions are simulated at z=0, all particles would first pass by this layer.



Histo counting number of particles forming a given number of hits.
Almost all particles have less than 20 hits => choose bucket size = 20.
Most particles have 12-15 hits => expect that in a given bucket more than half of the hits belong to the same particle.



How are the buckets of hits created for one event?
Group 20 candidate hits together to form a candidate line or track.
(using the Annoy library for nearest neighbors search)

1

- For all hits in the event
- Add their x, y, z coordinates to an Annoy index
- Re-order the index into a tree based on angular closeness

2

- Loop over all the hits that come from the central layer closest to the beam (the seeds of each bucket)
- For this hit find the 20 nearest neighbors (by angle)
- Each bucket has hit candidate to form one line
- Thus candidates to belong to the same original particle

The problem we try to solve using simulated hits and detector, with data offer by LHC experiments for the TrackML challenge.

reco	X	Y	Z	Volume_id	Layer_id	Module_id	
truth							Particle_id

Hard problem: For all reco hits find all reco particles => too hard

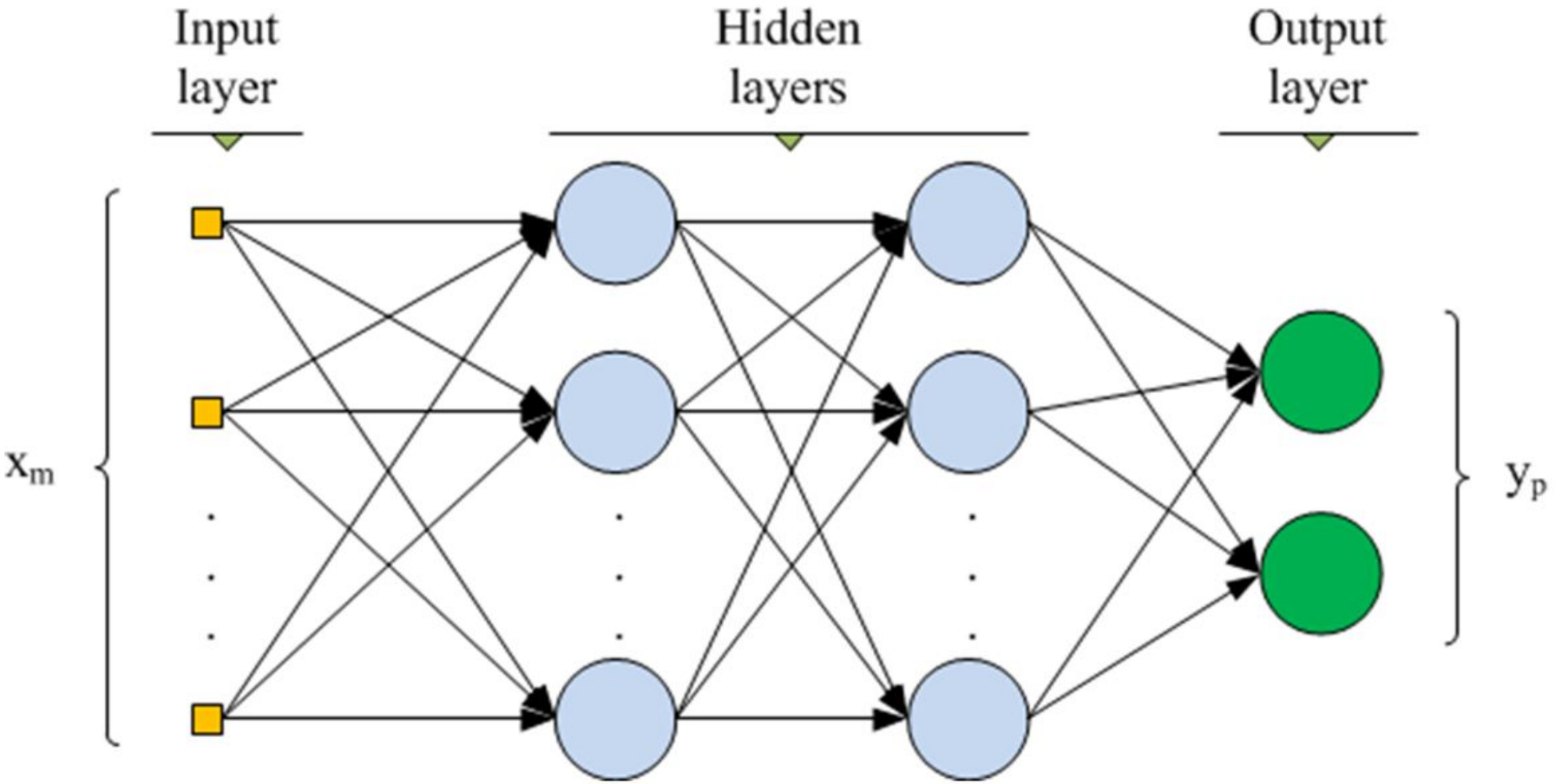
For each hit in “first” layer find 20 nearest hits per direction => bucket => find all particles in bucket => still hard

Our project

For each hit in a bucket find out if the hit belongs to the particle that has the largest number of hits in that bucket.

The question we try to answer: find the function using a deep neural network (DNN) that for each bucket (of 20 hits) has the following input and output structure:

input	(x,y,z)_1	(x,y,z)_2	(x,y,z)_3	…	(x,y,z)_18	(x,y,z)_19	(x,y,z)_20	60 el
output	1	-1	1	…	-1	1	1	20 el



Neural Network
Training
Predicting
Evaluating the performance

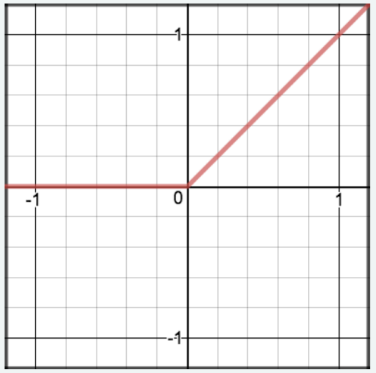
Tested various models, example of two of them. Today plots from last.

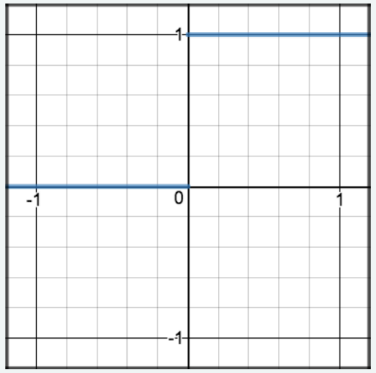
Choice		
Number nodes per input layer	20x(x,y,z)=60	20x(x,y,z)=60
Number nodes per output layer	20	20
Number nodes per hidden layer	40	100
Number hidden layer	5	2
Activation function input	Linear	Linear
Activation function output	tanh	tanh
Activation function hidden	ReLU	eLU
Loss function	hinge	squared hinge
K-fold	50% Train 50% Test	70% Train 30% Test
Optimizer	Adadelta	Adam

Two activation functions compared for hidden layers: RELU and ELU.

RELU: rectified linear units

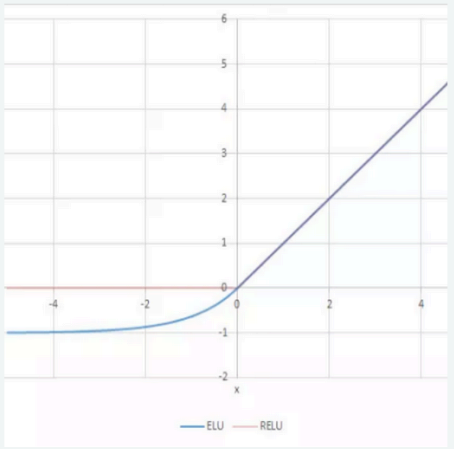
Function	Derivative
$R(z) = \begin{cases} z & z > 0 \\ 0 & z <= 0 \end{cases}$	$R'(z) = \begin{cases} 1 & z > 0 \\ 0 & z < 0 \end{cases}$

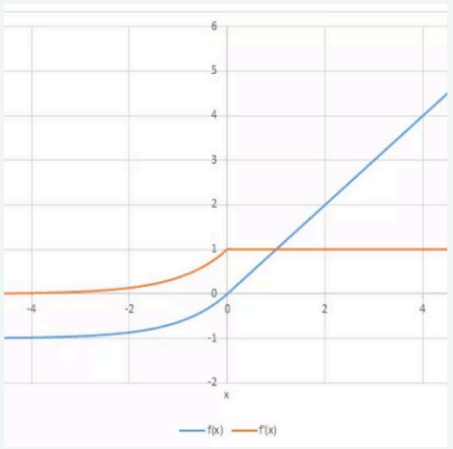




ELU: variation of RELU allowing negative values

Function	Derivative
$R(z) = \begin{cases} z & z > 0 \\ \alpha.(e^z - 1) & z <= 0 \end{cases}$	$R'(z) = \begin{cases} 1 & z > 0 \\ \alpha.e^z & z < 0 \end{cases}$

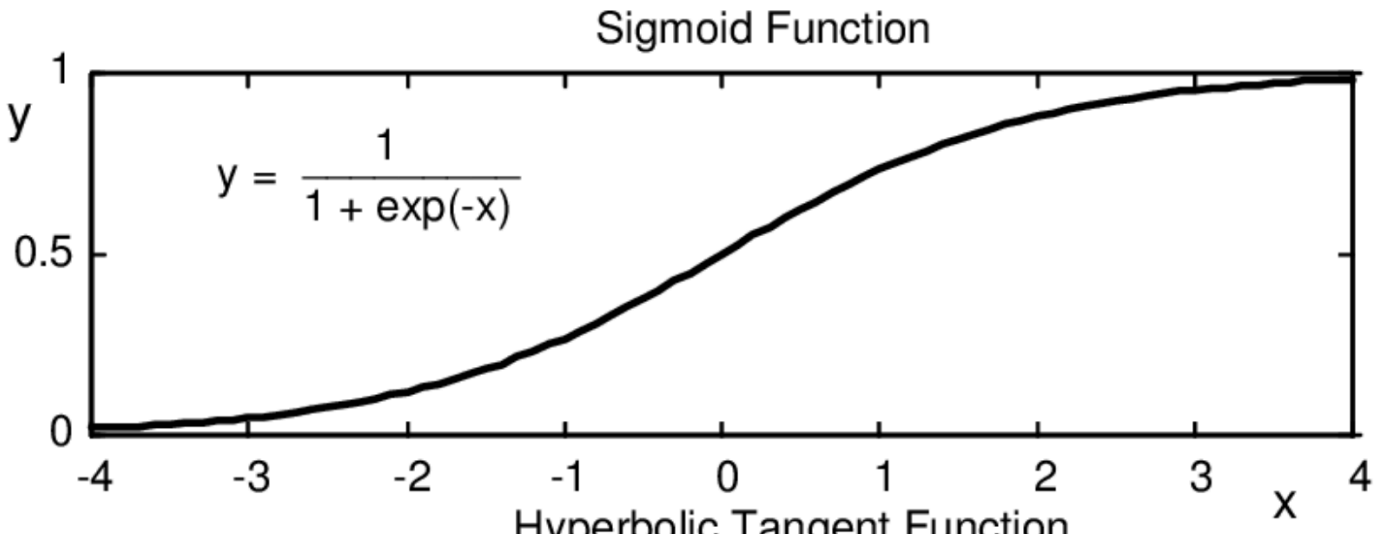




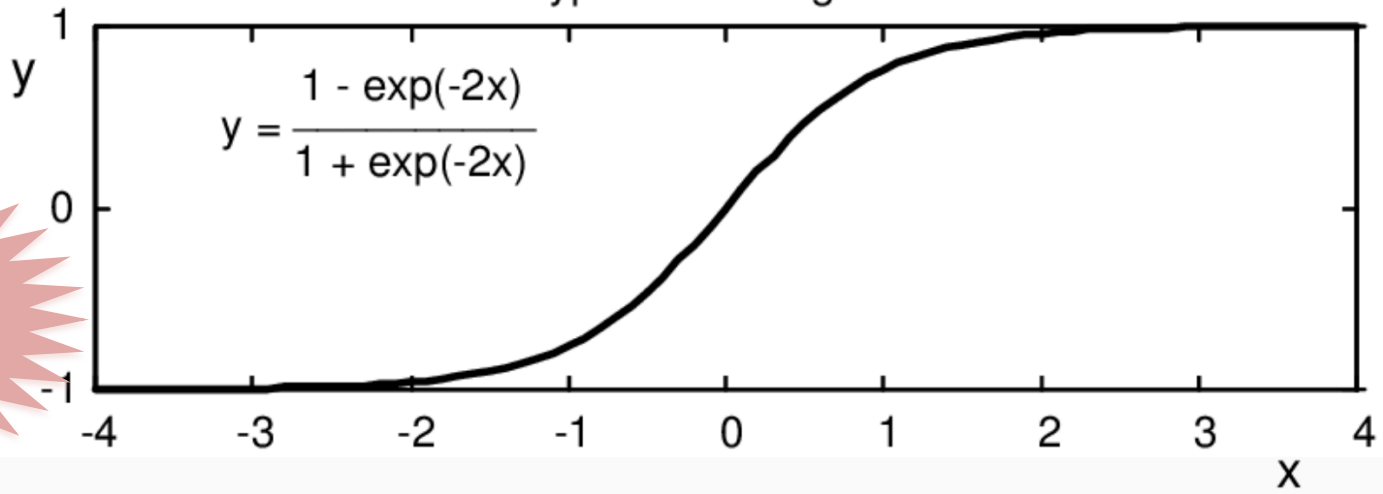
Credit images: https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html

Two activation functions considered for the output layer: sigmoid and hyperbolic tangent.

Sigmoid: [0,1]



Tanh: [-1,1]



Our project

Credit images: https://towardsdatascience.com/complete-guide-of-activation-functions-34076e95d044

Two loss functions considered for the learning method:
Hinge and Squared Hinge, which work with output 1 and -1.

Squared-Hinge:
Sums of 0 or 4

$$L(y, f(x)) = \sum_{bucket} \sum_{hit} max(0, 1 - y * f(x))^2$$

Hinge:
Sums of 0 or 2

$$L(y, f(x)) = \sum_{bucket} \sum_{hit} max(0, 1 - y * f(x))$$

We have several output labels and a classification problem (case B).

Case A: Sum (probabilities of each label)=1 => Softmax loss function => Sigmoid output layer activation function.

Case B: Probabilities of all labels are independent of each other => (Squared) Hinge loss function => Tanh output layer activation function.

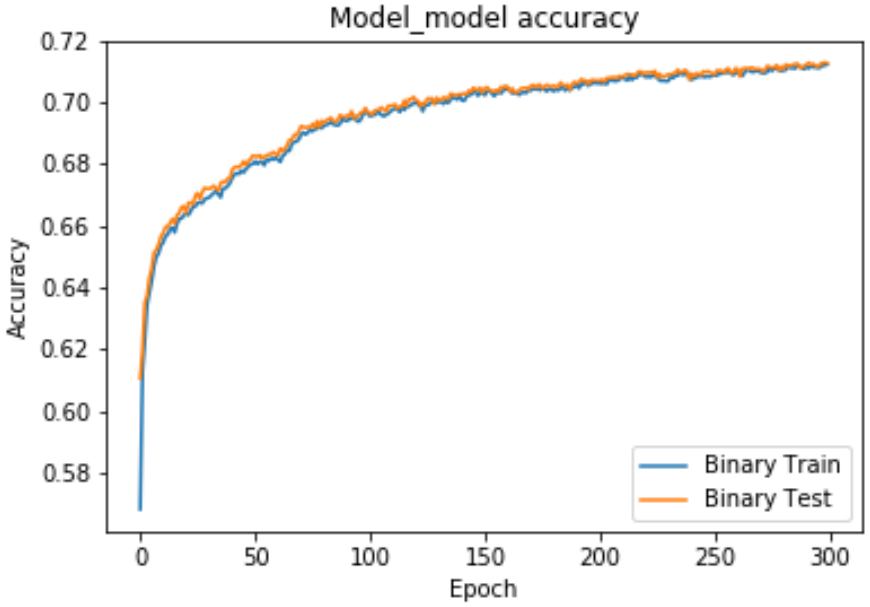
Our project

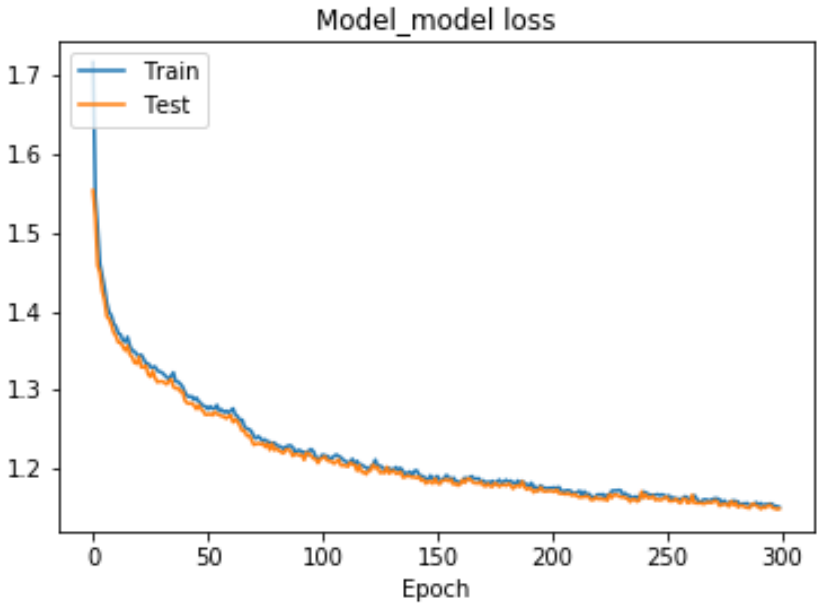
In following plots we split train and test by 70% to 30%,
Running on all 100 events simulated with mu=200,

Comparing same model with 300 epochs:
Split by events, 70-30: 10x (Tr, Tr, Tr, Tr, Tr, Tr, Tr), 10x (Te, Te, Te)

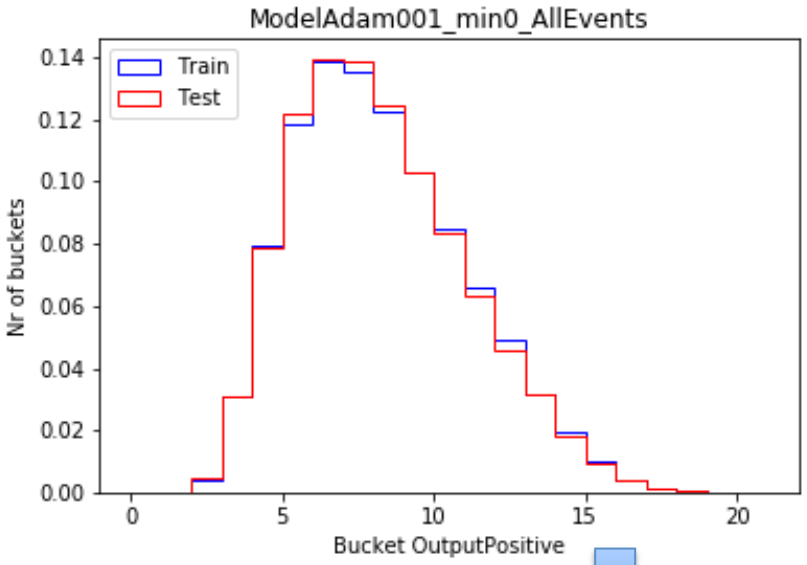
- input: used only one bucket in three to avoid overlapping hits in neighboring buckets

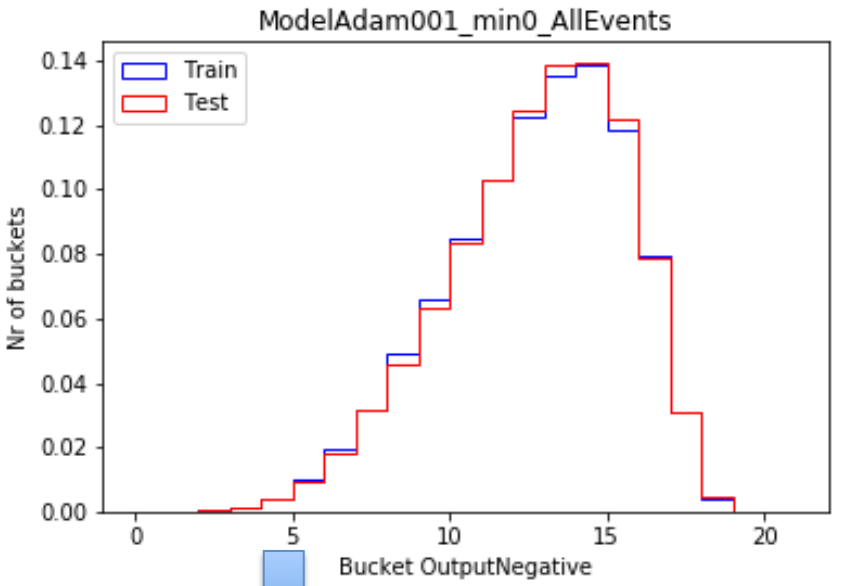
Metrics by default => Accuracy and Loss: Both look good.
Train and Test are similar => no overtraining.
Test is a bit worse than Train, as expected.



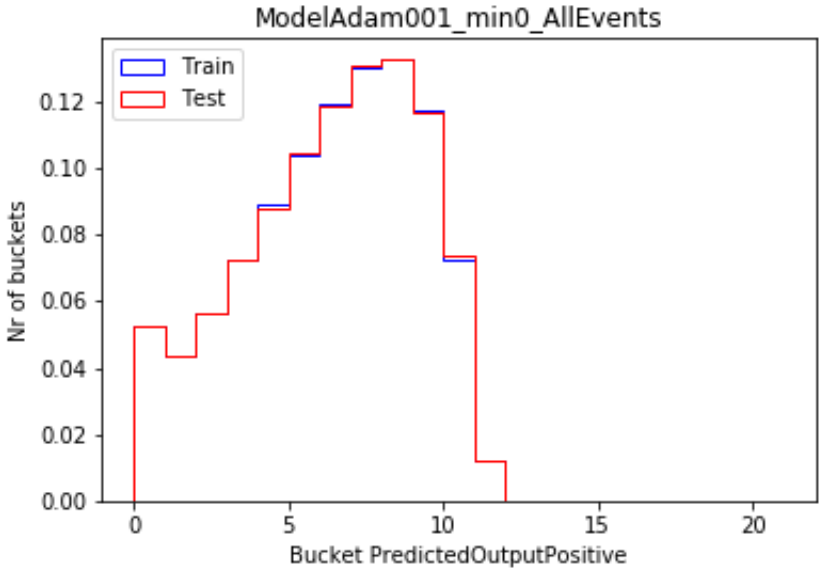


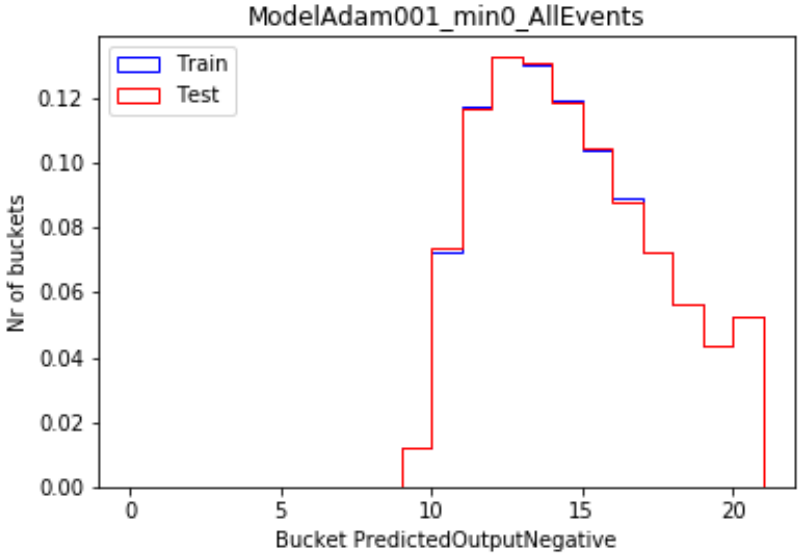
Output vs Predicted Output, for both positive and negative.



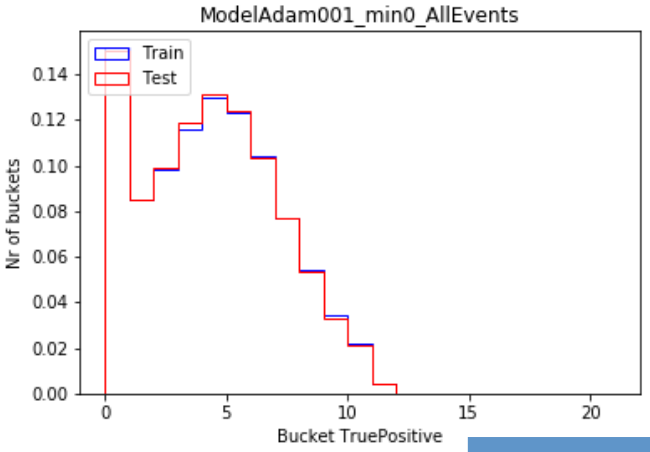


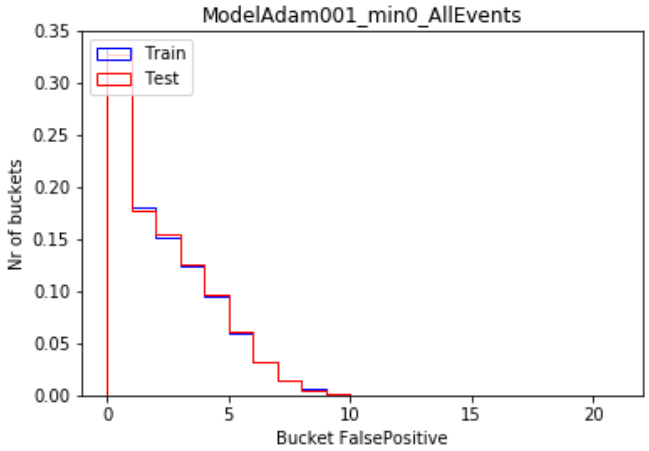
Predict after train.



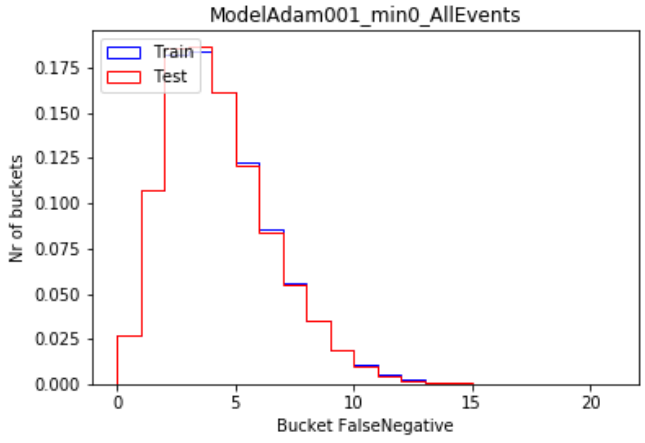


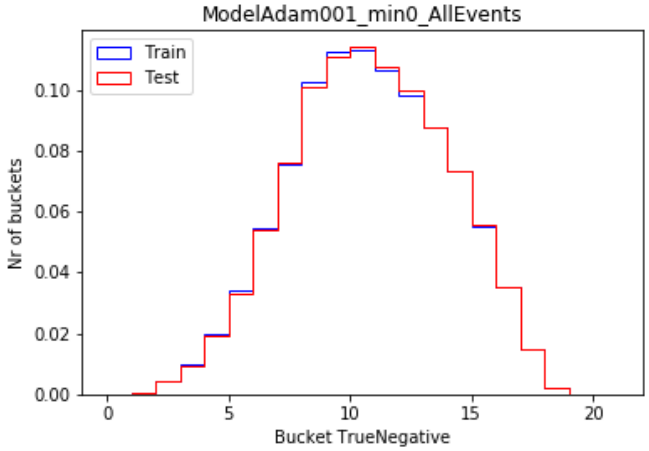
Confusion matrix.



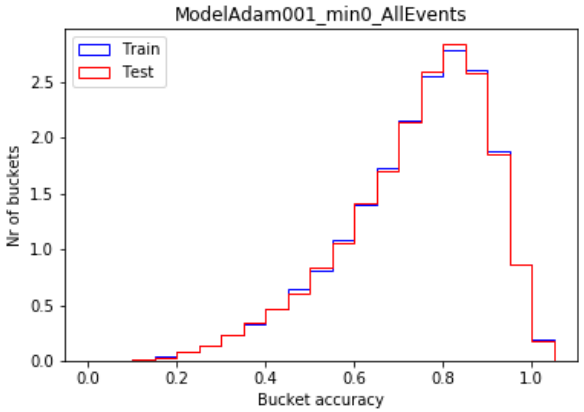


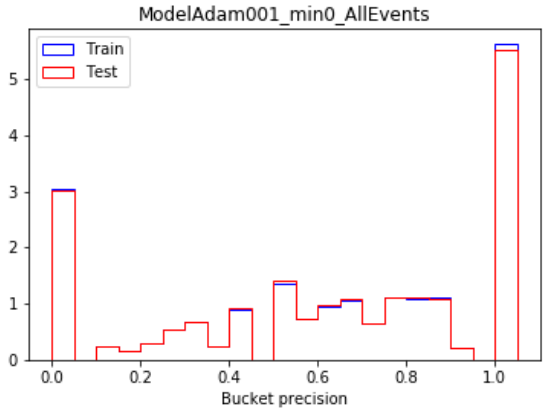
Confusion Matrix		
	True Positive 15%	False Positive 8%
	False Negative 23%	True Negative 54%

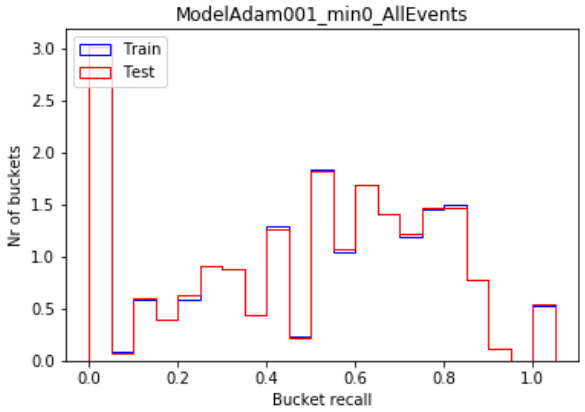


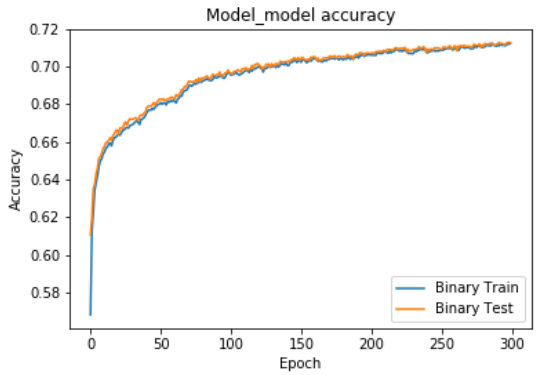


Accuracy	Precision	Recall
$\frac{TP+TN}{TP+FP+FN+TN}$	$\frac{TP}{TP+FP}$	$\frac{TP}{TP+FN}$



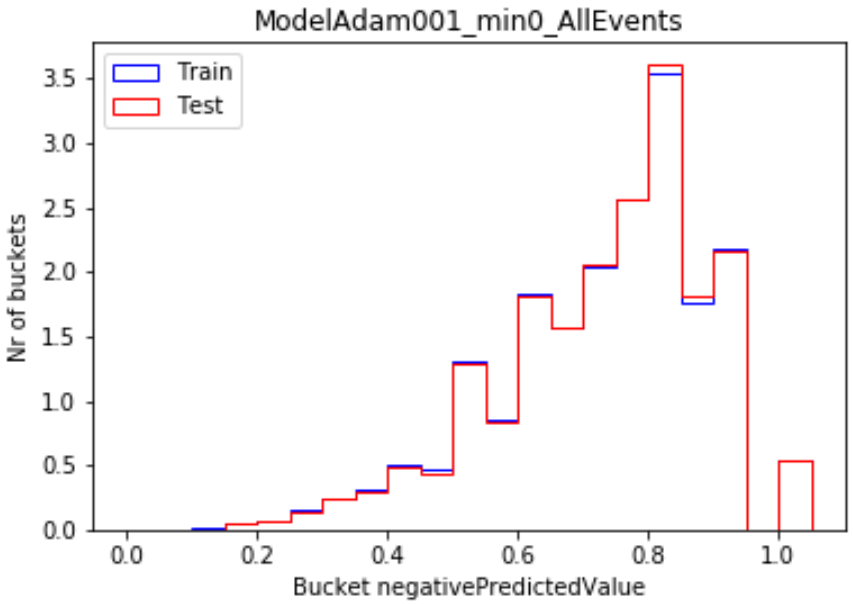


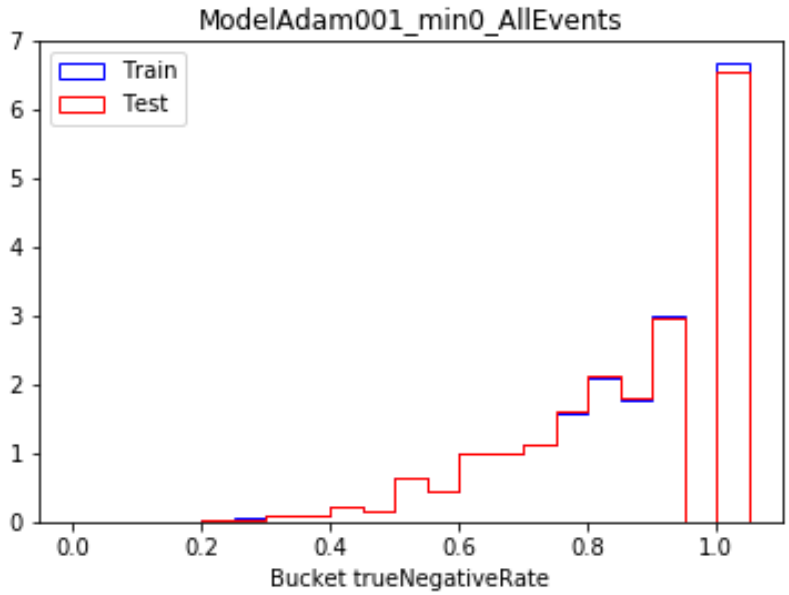




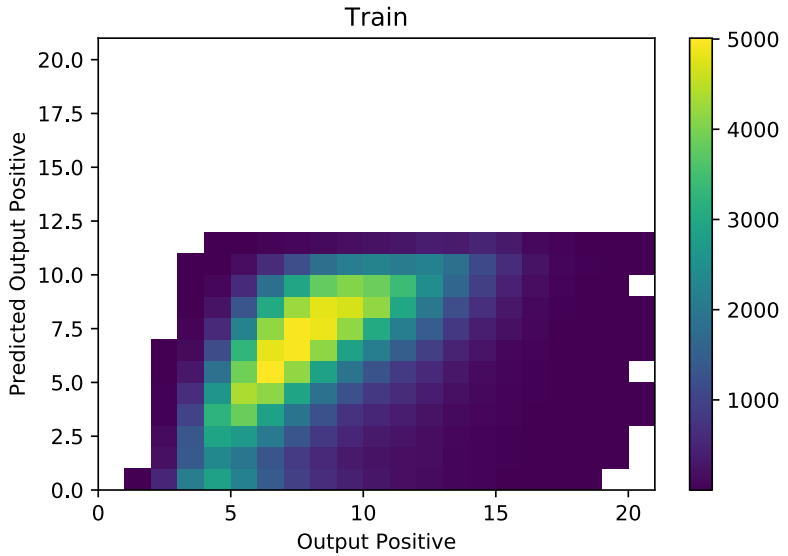
Precision and recall for the negative output

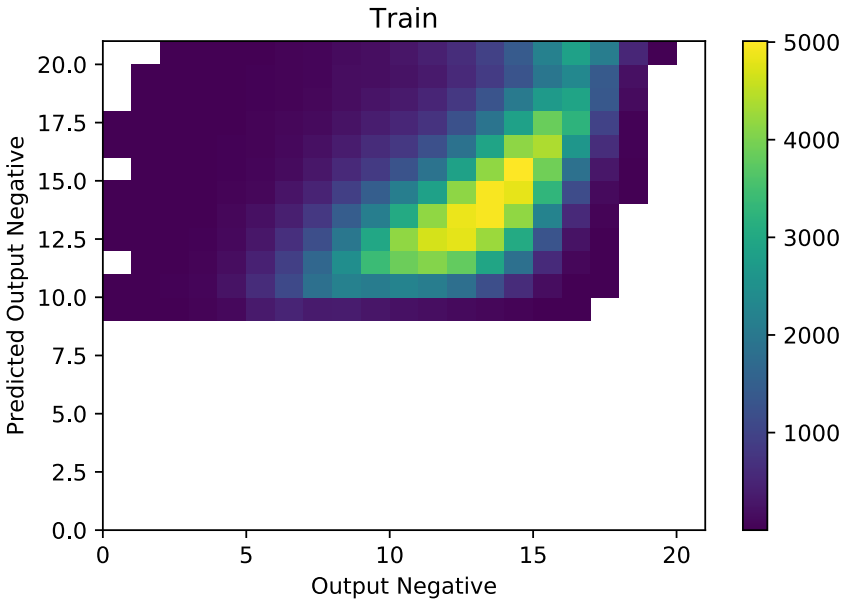
Negative predicted value	True negative rate
$\frac{TN}{TN+FN}$	$\frac{TN}{TN+FP}$

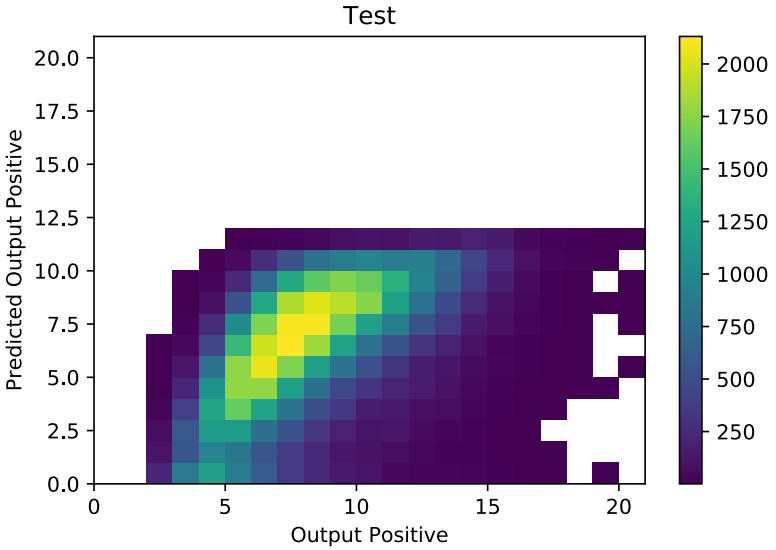


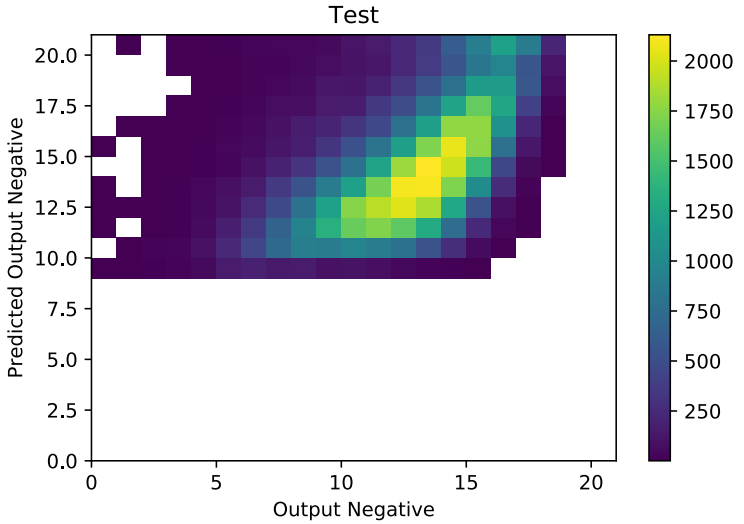


2D Histos Output vs PredictedOutput
Left: Positive, Right: Negative. Top: Train, Bottom: Test



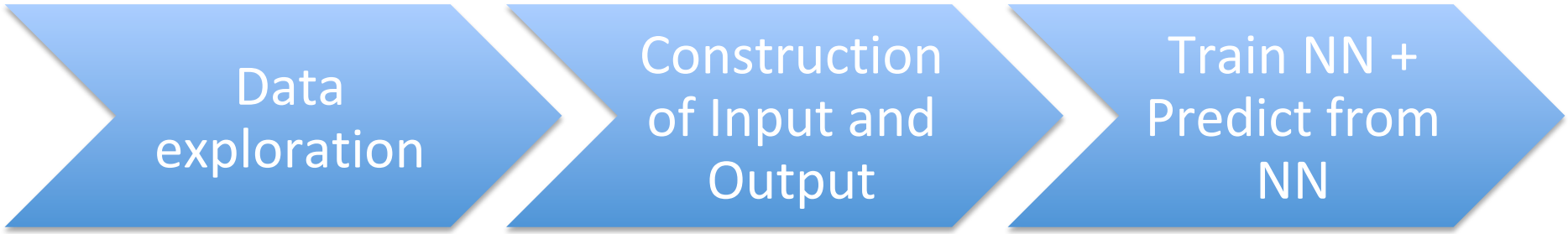






Conclusion:

Our task: For each hit in a bucket find out (DNN) if the hit belongs to the particle that has the largest number of hits in that bucket.



The code describing the best model so far:
two hidden layers, eLU, squared-hinge.

Next steps: Improve further the model,
Study more figures of merit.