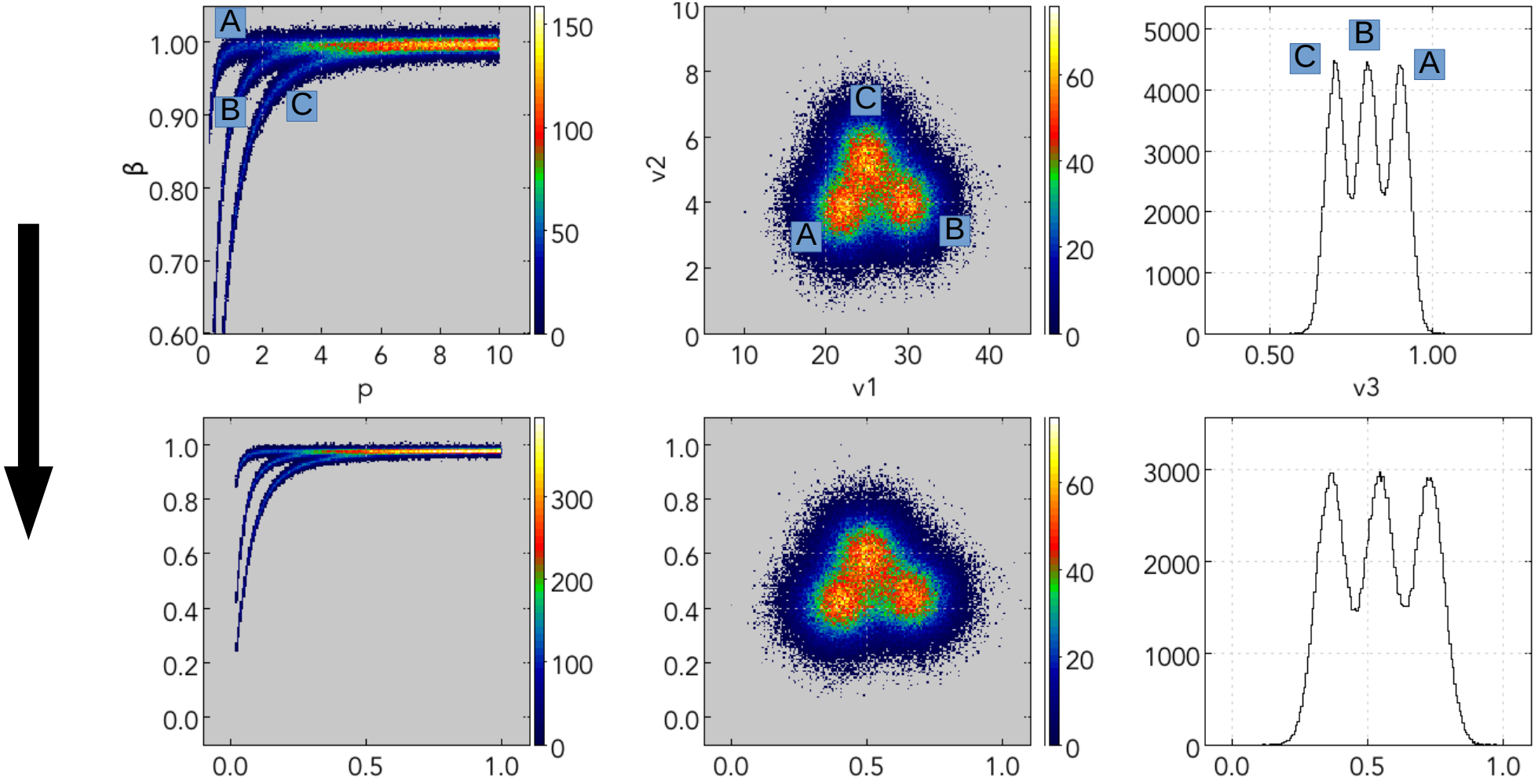
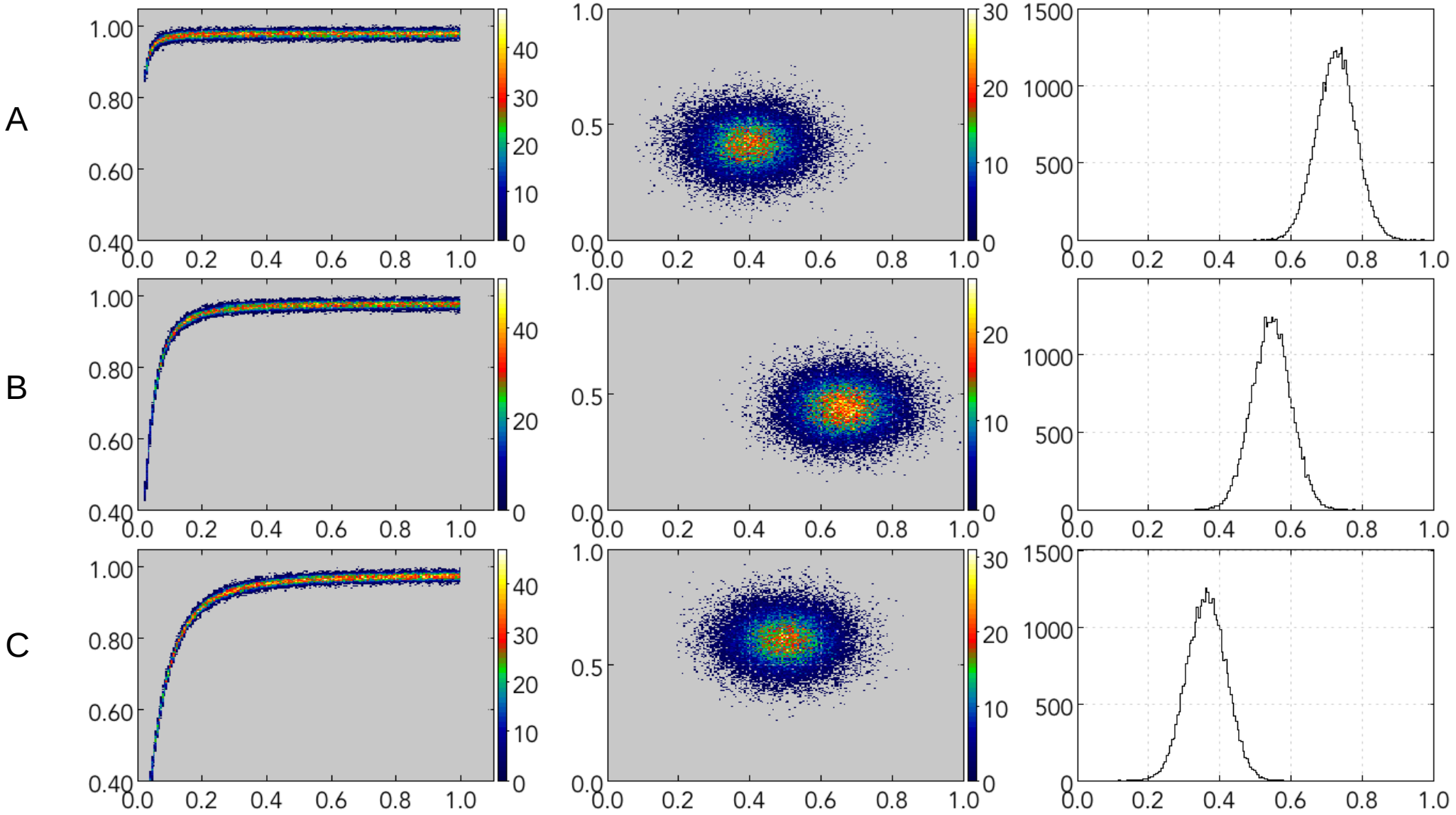
Particle ID with a Neural Network (using coatjava!)

Toy model: 3 particles – A, B, C – with 5 observables – p, beta, v1, v2, v3.

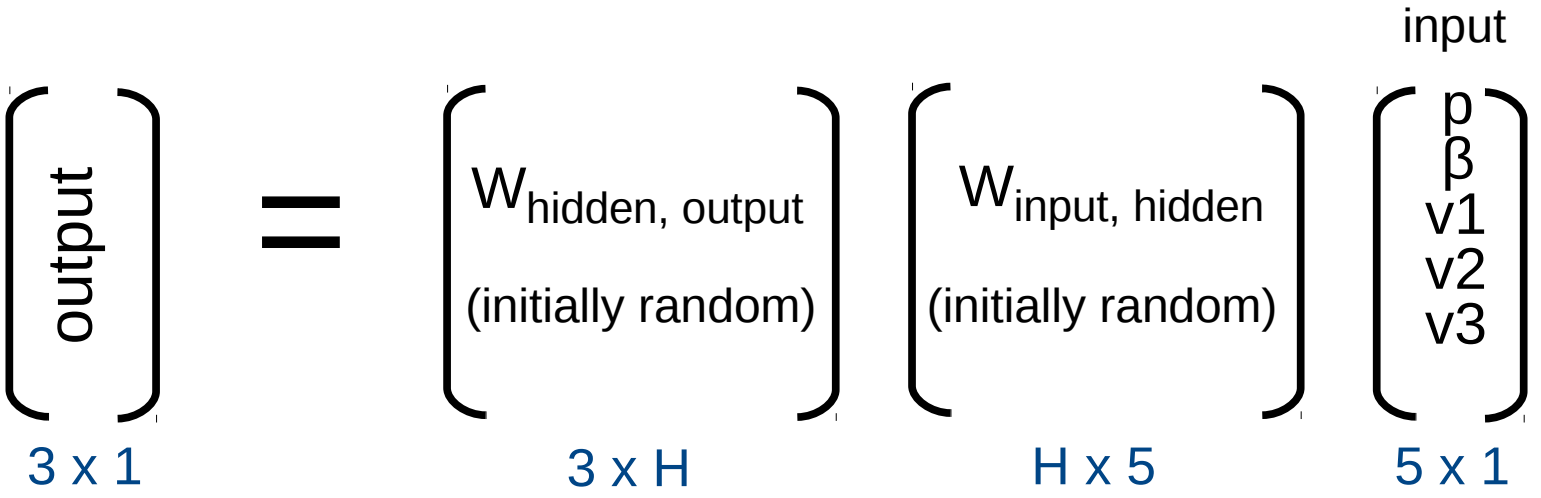


* Quantities are shifted and re-scaled so range is about 0.01 – 1.00

Particle ID with a Neural Network – Training Data



Particle ID with a Neural Network – Training

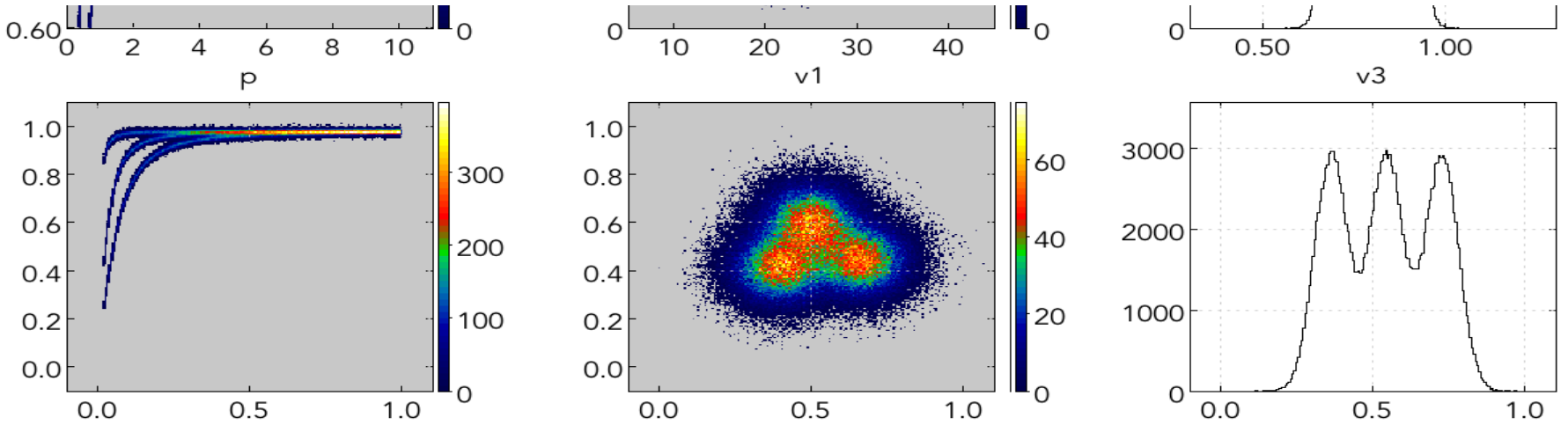


H = number of hidden nodes (on the order of 100).
("Activation function" not shown.)

The output is compared to the desired output – either (1, 0, 0) for A, (0, 1, 0) for B, or (0, 0, 1) for C* – the error then goes into an algorithm used to tune the W matrices. The algorithm has an adjustable parameter called the "learning rate." Many training iterations are needed before the network is reliable.

* zeros are bad for neural networks, so 0.01 is actually used.

Particle ID with a Neural Network – Test Data



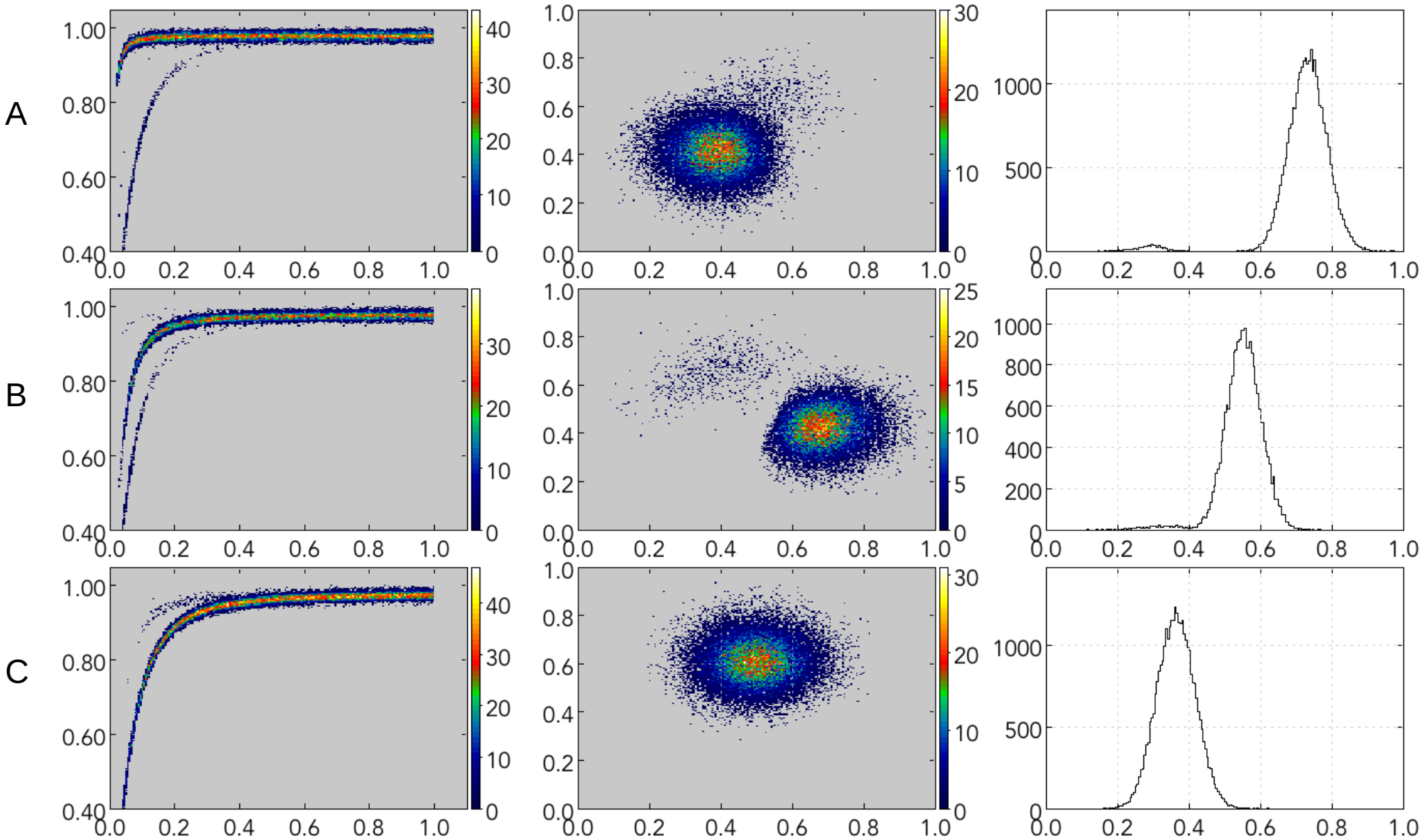
Ask the trained network to identify each particle as either A, B, or C. Each output will look something like this:

$$\begin{pmatrix} 0.977 \\ 0.081 \\ 0.012 \end{pmatrix}$$

Largest value is the networks best guess. This example corresponds to particle A.

Particle ID with a Neural Network – Results

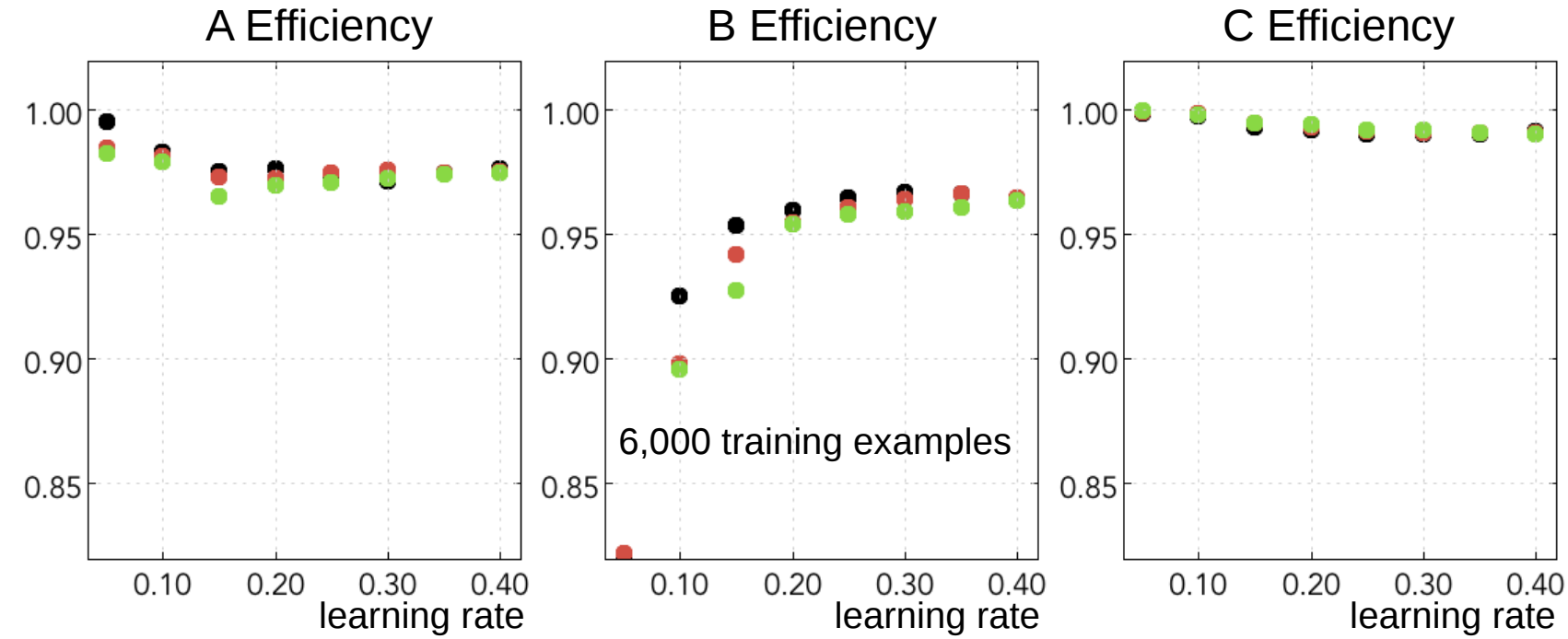
First try, no optimization

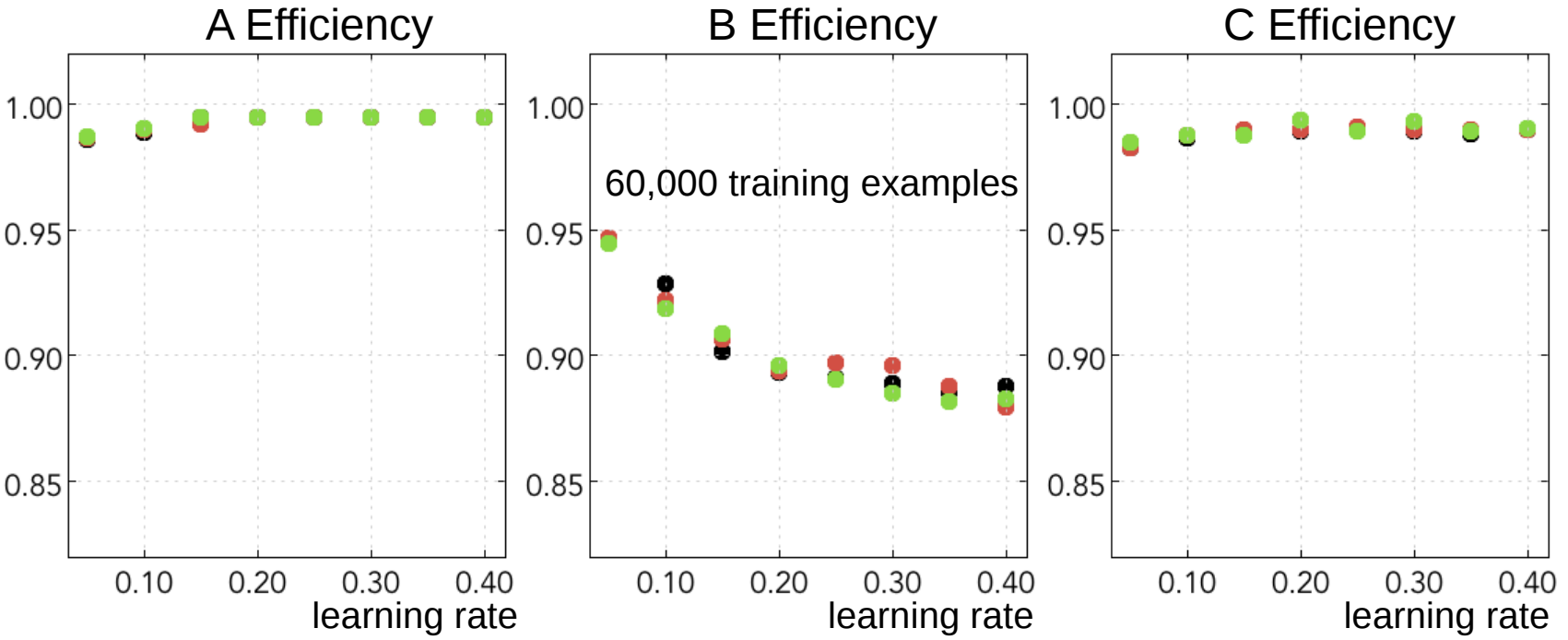


100 hidden nodes, 0.3 learning rate

Particle ID with a Neural Network – Optimization

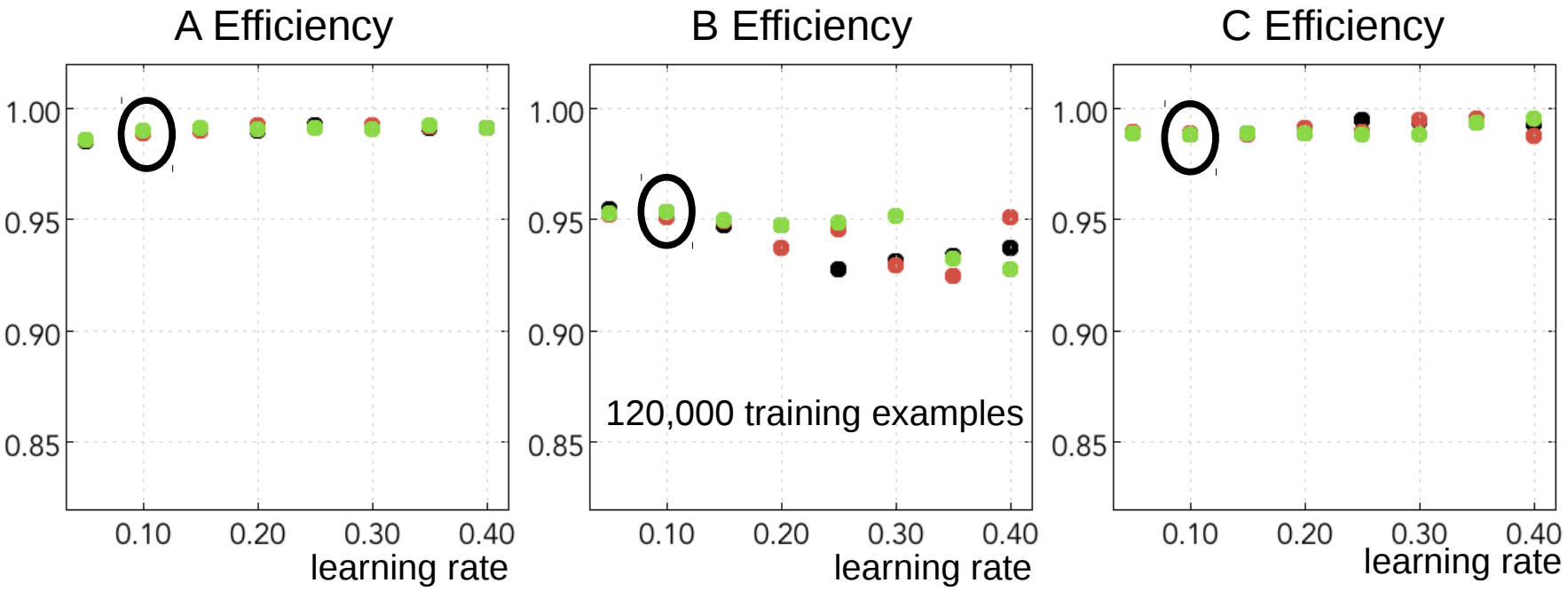
H = 50
H = 100
H = 150



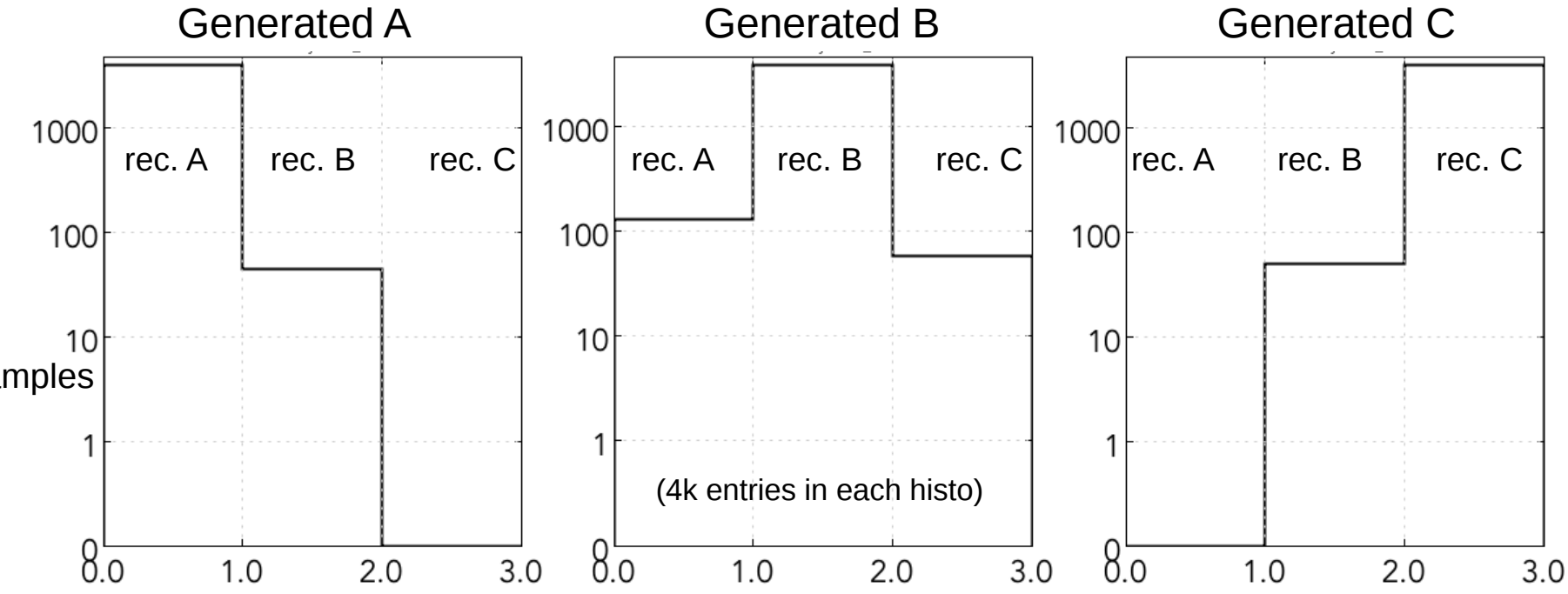


Particle ID with a Neural Network – Optimization

H = 50
H = 100
H = 150



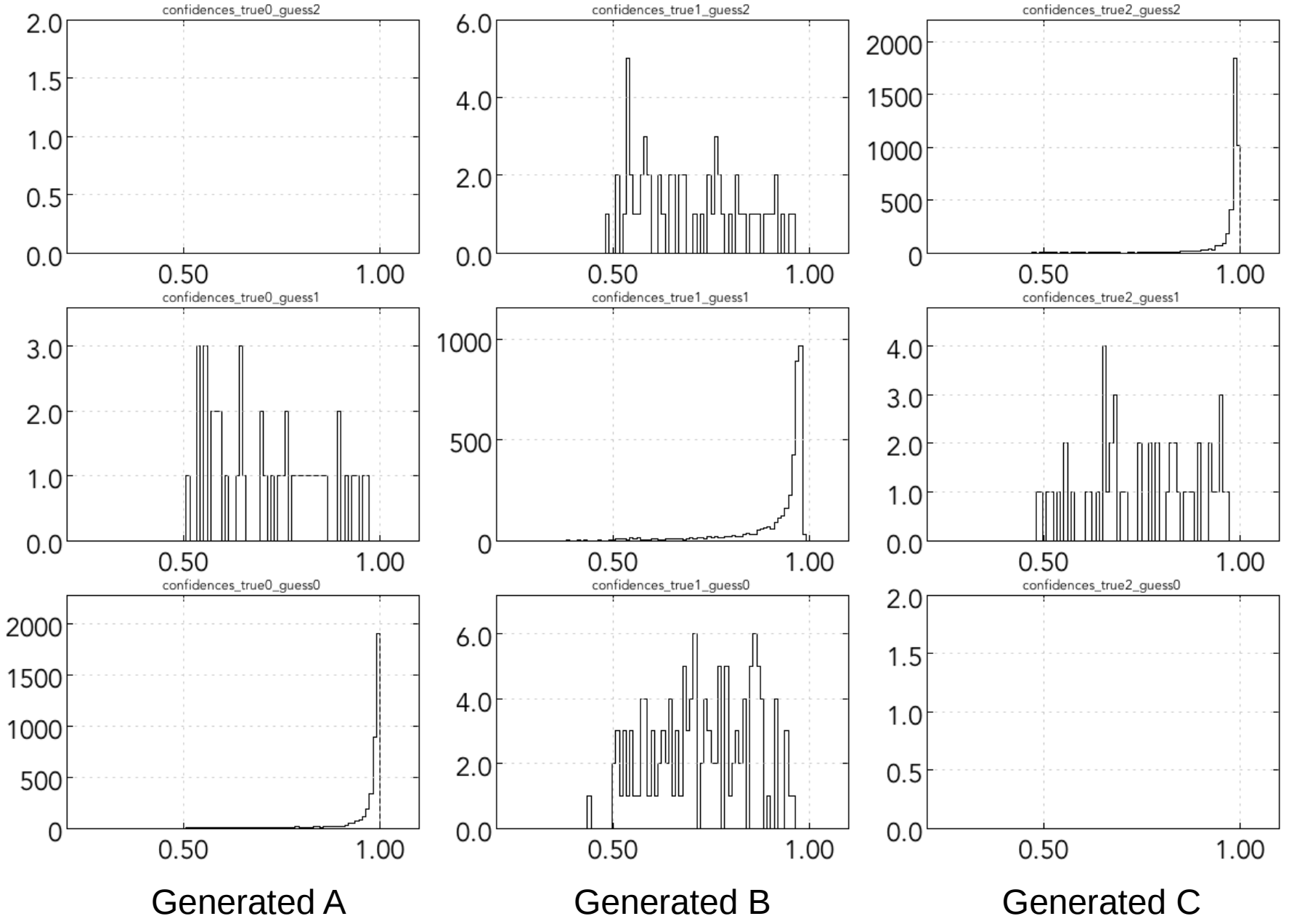
H = 50
L = 0.1
120k training examples



Particle ID with a Neural Network – Optimization

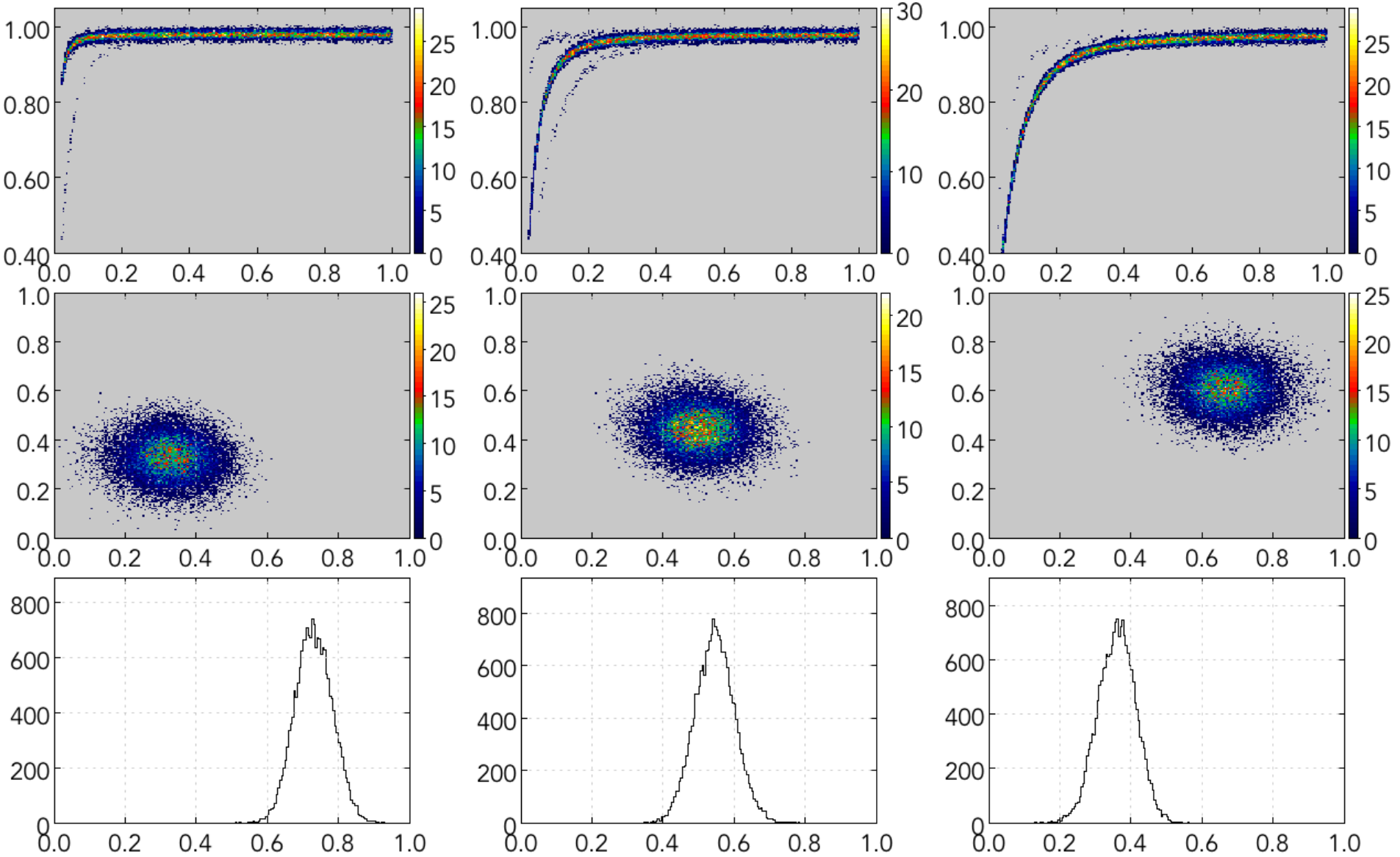
H = 50
L = 0.1
120k training examples

"confidence" values (the maximum value of the output vector)



Particle ID with a Neural Network – Optimized Results

H = 50
L = 0.1
120k training examples
0.7 confidence cut



Particle ID with a Neural Network – Optimized Results

H = 50
L = 0.1
120k training examples
0.85 confidence cut