A fast convolutional neural network for online particle track recognition

N.V. Biesuz[2], R. Bolzonella[1,2], P. Cardarelli[2], E. Calore[2], **V. Cavallini**[1,2], M. Fiorini[1,2], S.F. Schifano[1,2], R. Zese[1]

1. University of Ferrara, Italy
2. INFN - Istituto Nazionale di Fisica Nucleare, Italy

Timepix4

Timepix4 is a **hybrid pixel detector readout ASIC** developed by the Medipix4 Collaboration.
It consists of a matrix of **~230k pixels** with **55 μm** pitch.
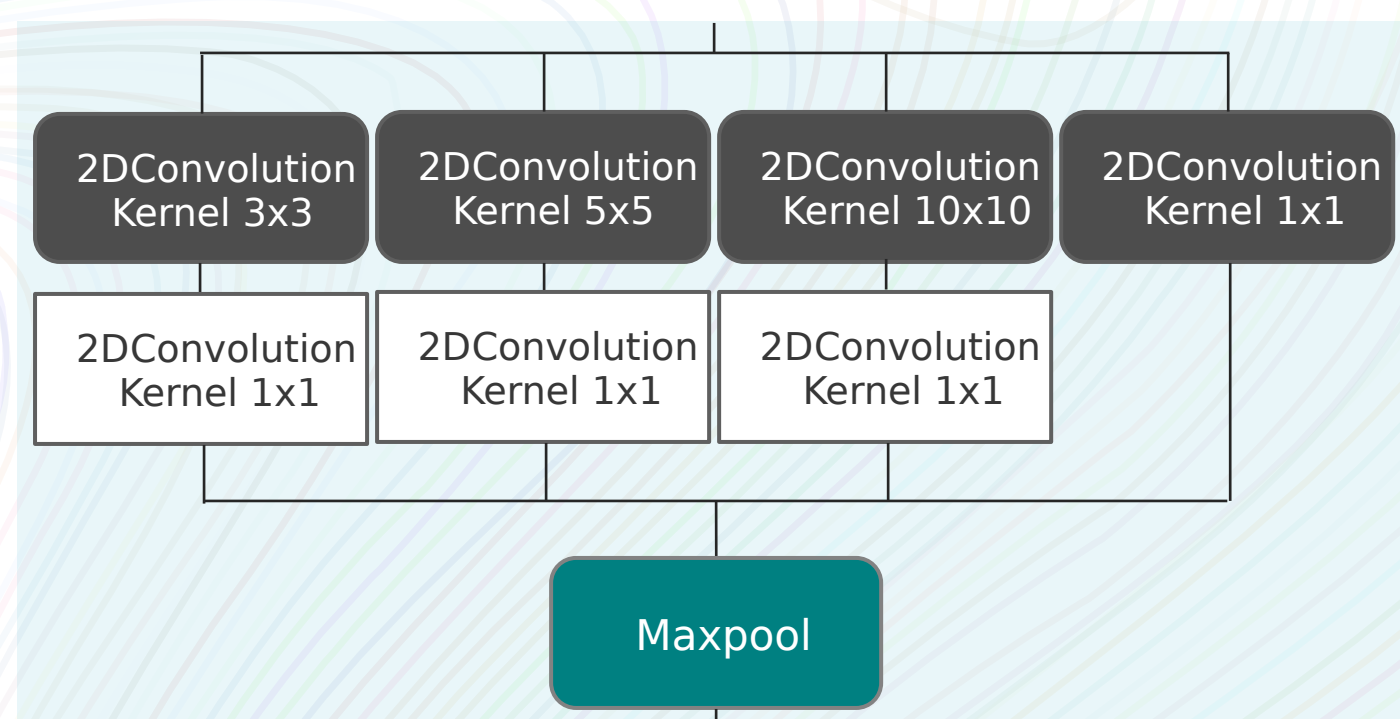Each can measure **time-of-arrival** and **time-over-threshold** when hit.
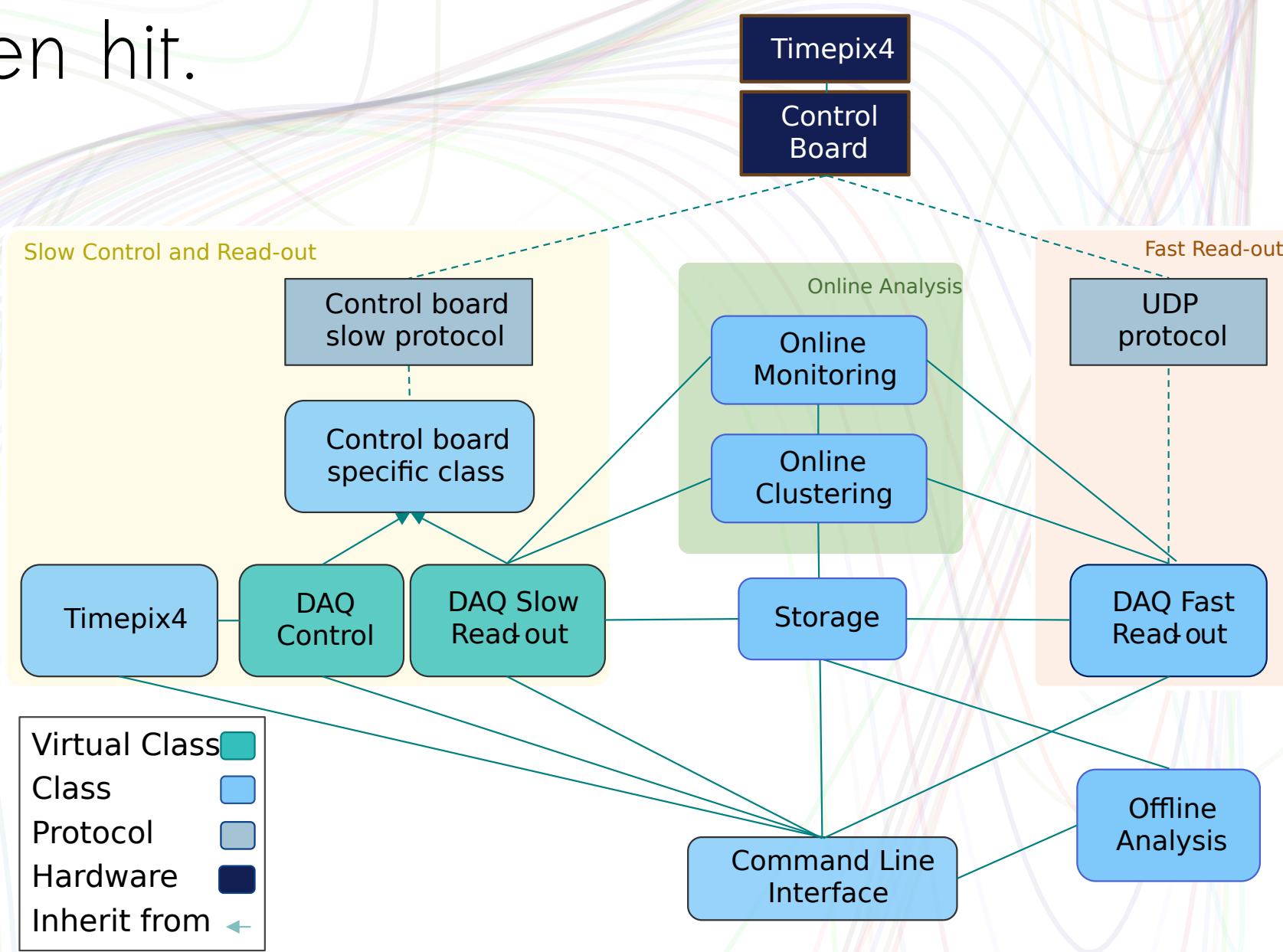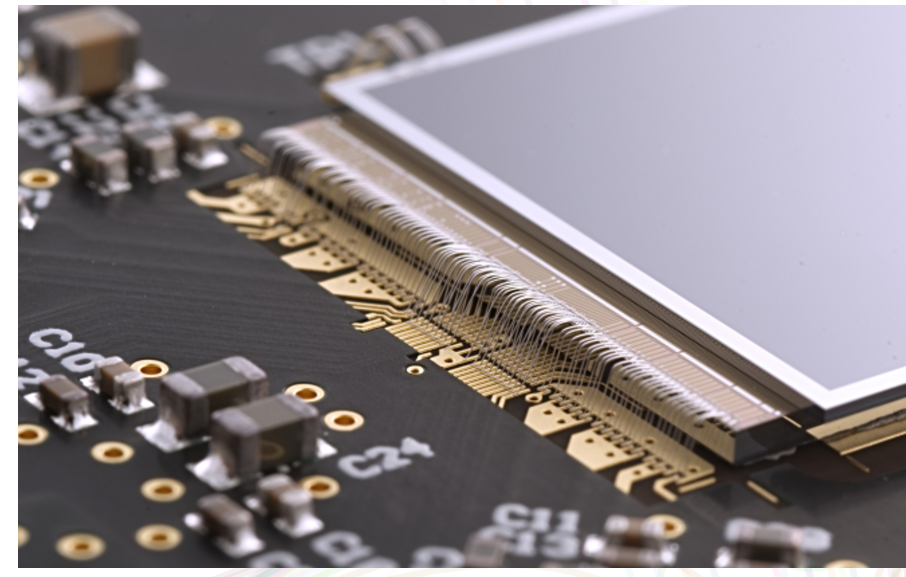
Timepix4 software

- **Written in C++**: fast, optimized, low-level, object-oriented.
- **Unique interface**: for control, read-out and online analysis.
- **Flexible Architecture**: designed to accomodate various DAQ systems with minimal changes.
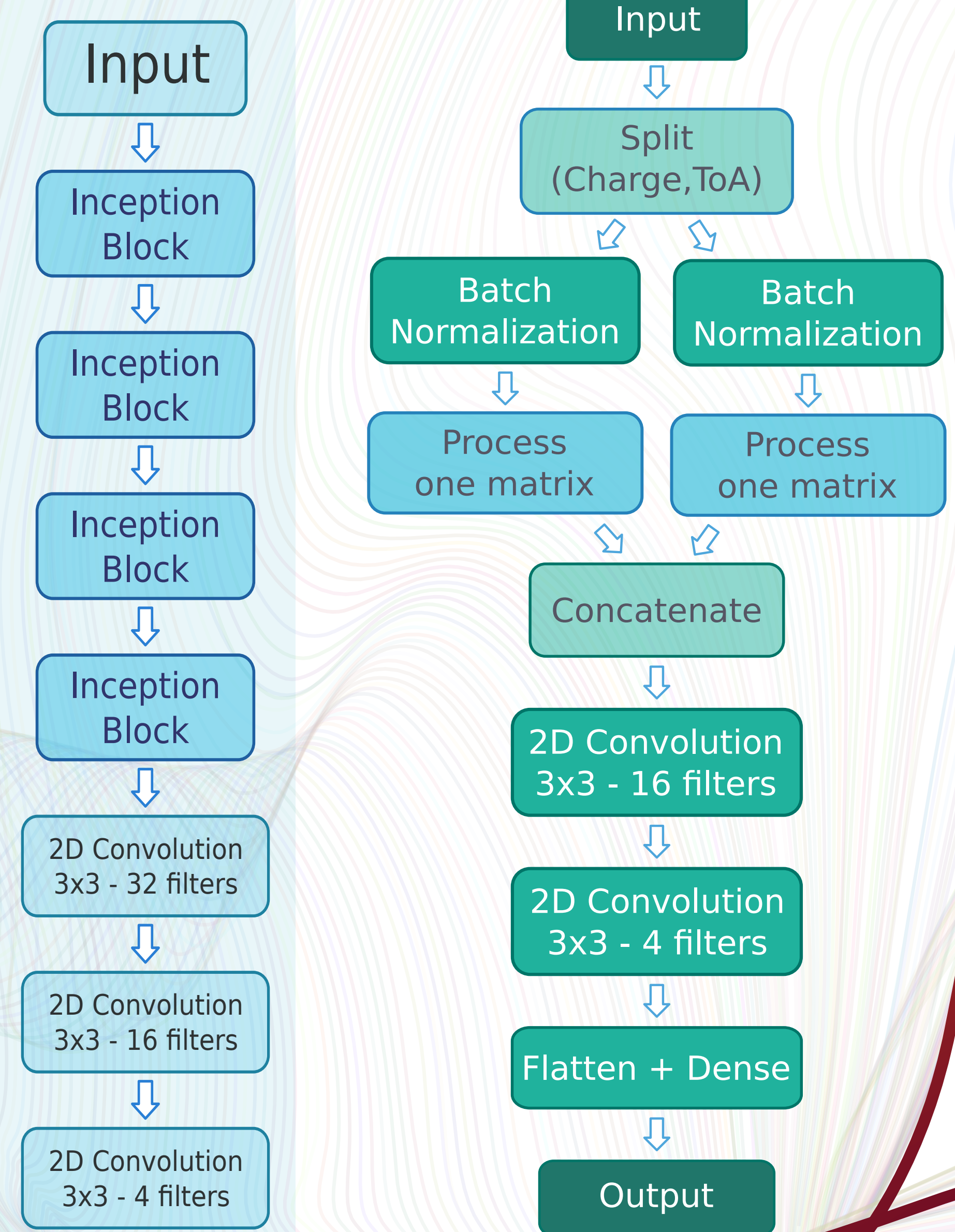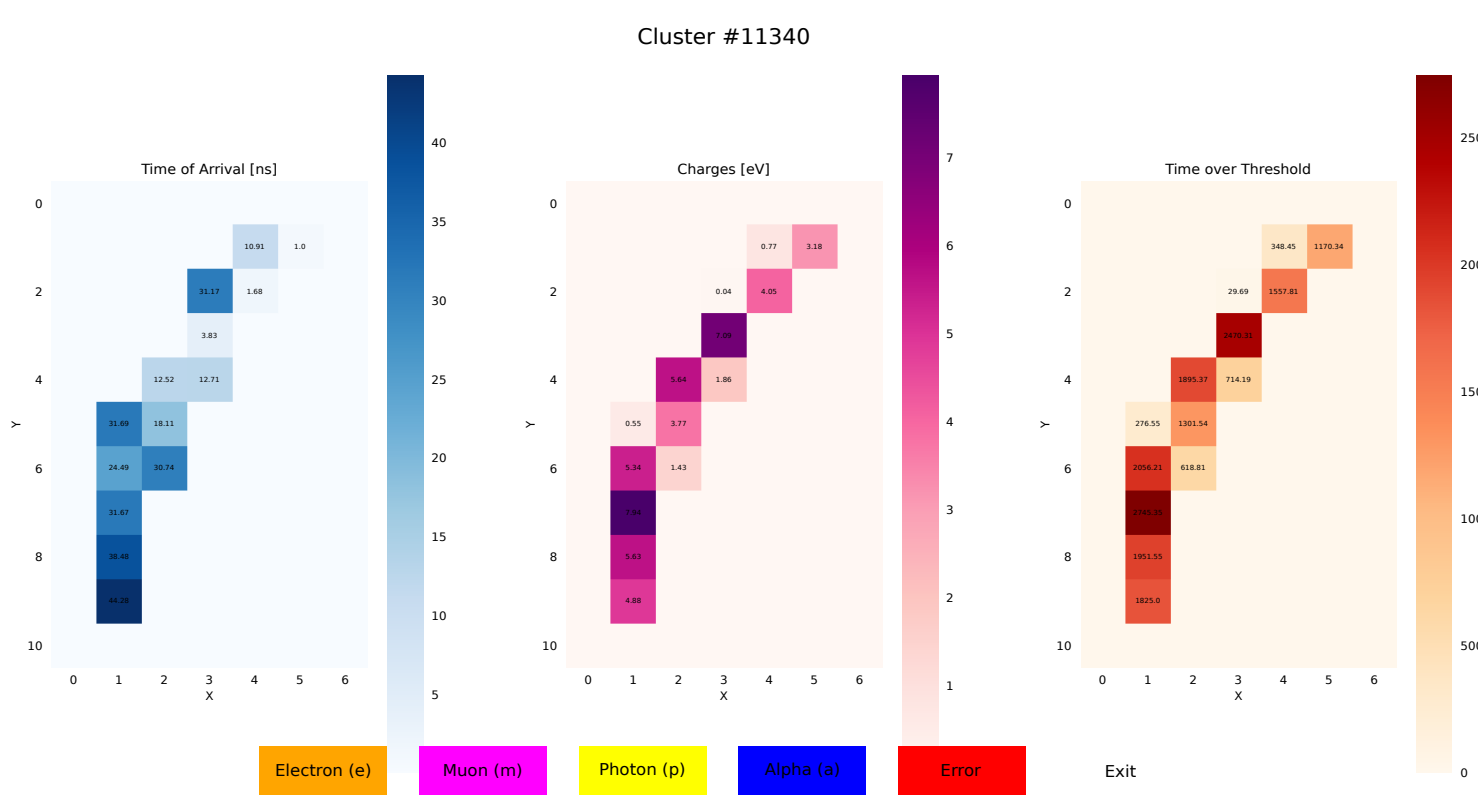- **Open source**

Structure

Based on inception blocks:

The network takes in input a **50x50x2 matrix**, that is splitted into two 50x50 matrixes.
Each 2D matrix is **individually processed** and the results are *concatenated* together.
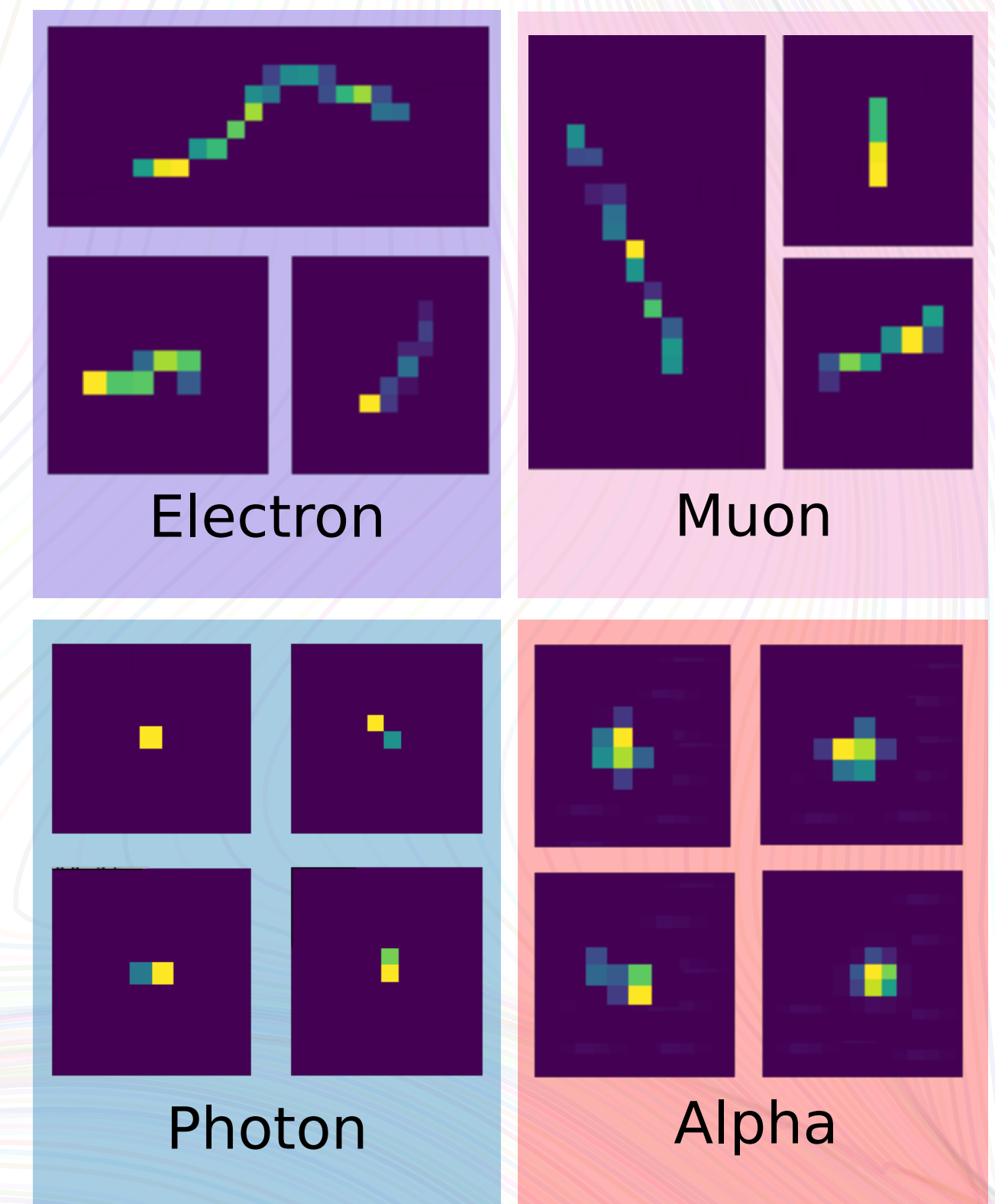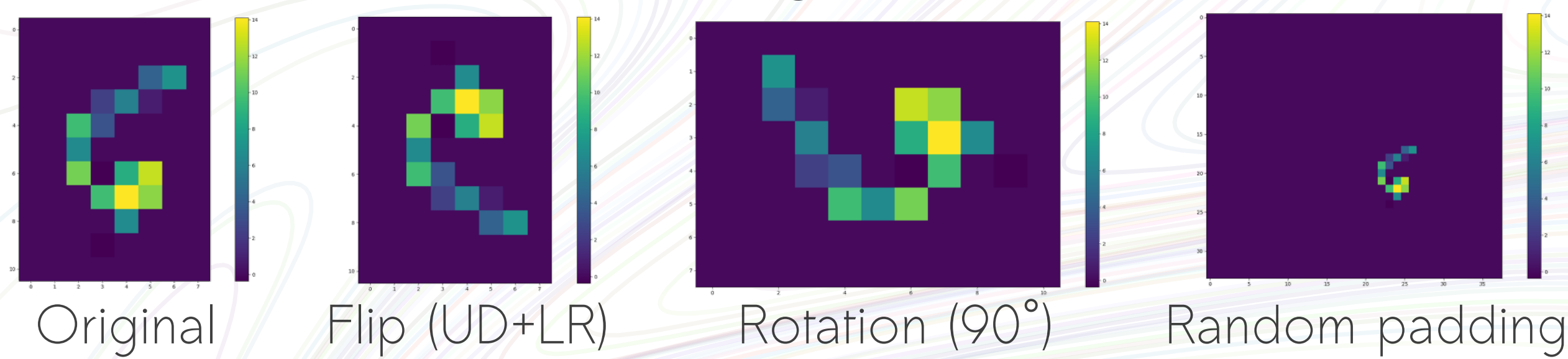
Network to process one matrix:

Dataset

Manually labelled dataset with interactive Python script

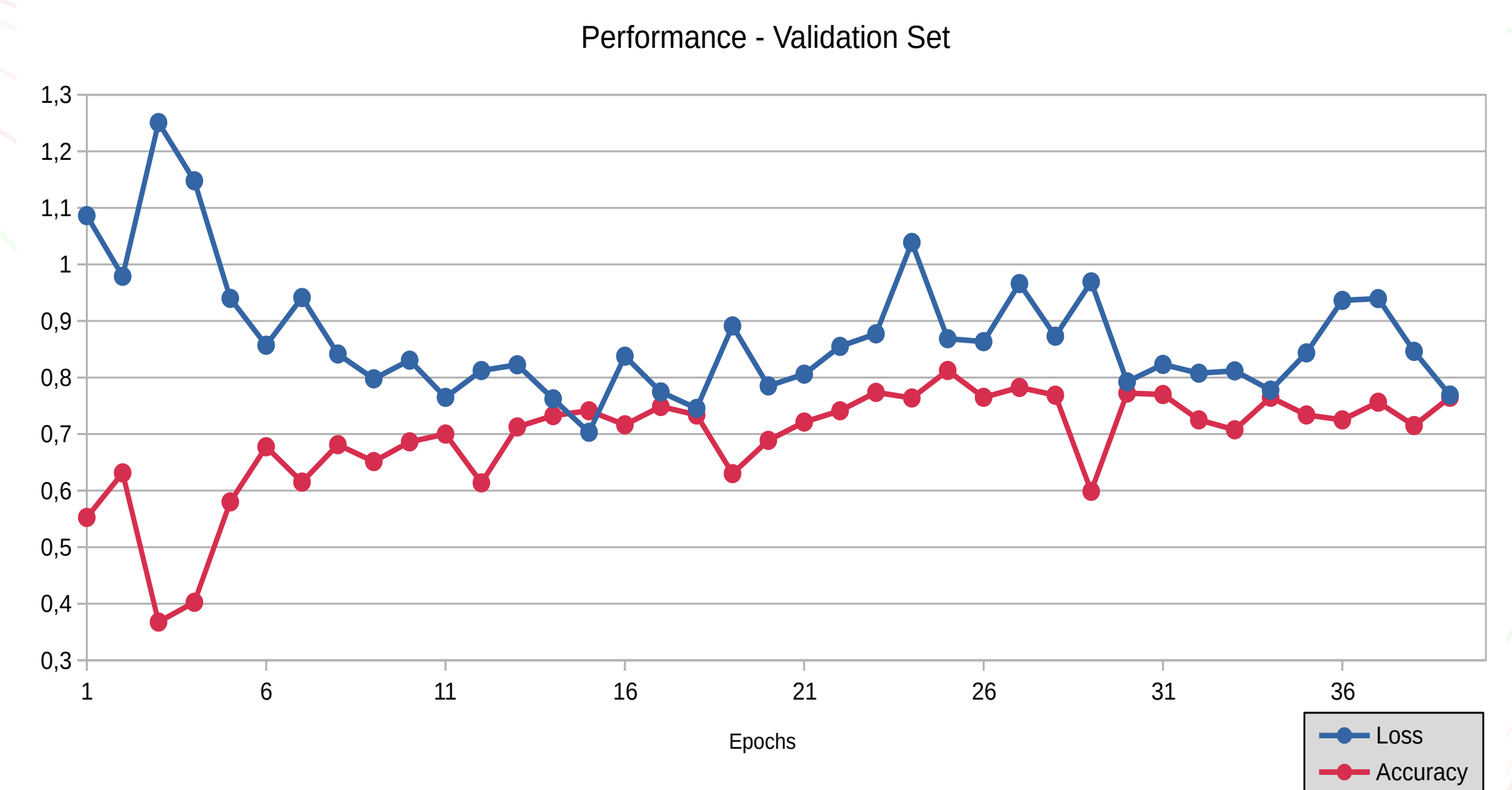**Total clusters**: 4000
- **Validation set**: 800
- **Training set**: 3200

Dataset augmentation

Original | Flip (UD+LR) | Rotation (90°) | Random padding

Performance

Confusion Matrix Validation Set

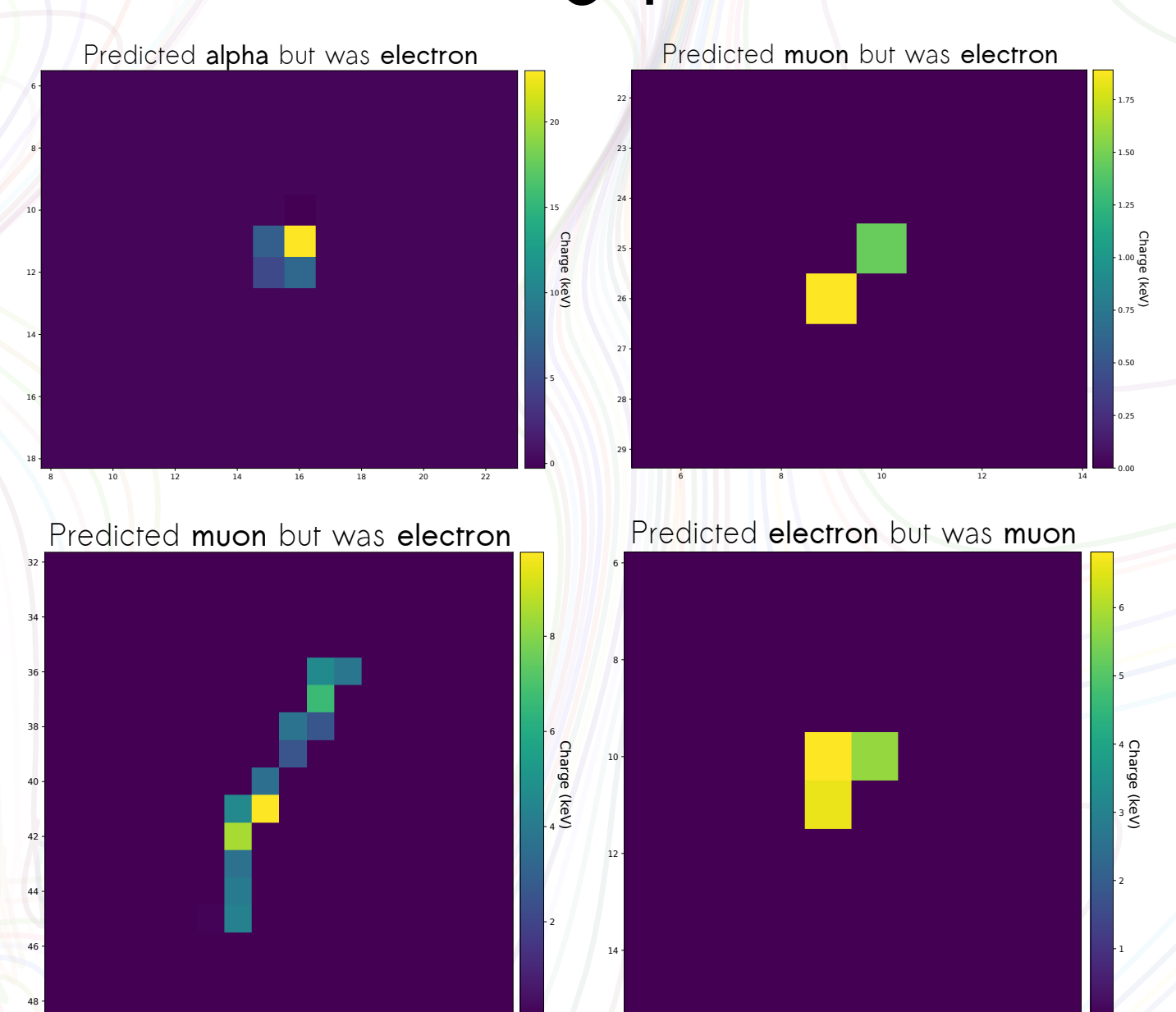| Truth	Alpha	Electron	Muon	Photon
Photon	0.00%	0.83%	5.83%	93.33%
Muon	6.25%	30.47%	63.28%	0.00%
Electron	6.56%	72.50%	15.94%	5.00%
Alpha	96.43%	3.57%	0.00%	0.00%

Predicted

Performance - Validation Set

- **Training Time**: ~8 s / epoch (batch size: 32)
- **Recognition time**: ~20 ms / cluster
- **Best Accuracy**: 81.25% in 25 epochs

Results

Natural radioactivity acquisition

Red: Alpha particle
Orange: Electron
Blue: Muon
Green: Photon

Raw hits are clusterized into tracks with a deterministic algorithm and then labelled using the neural network.
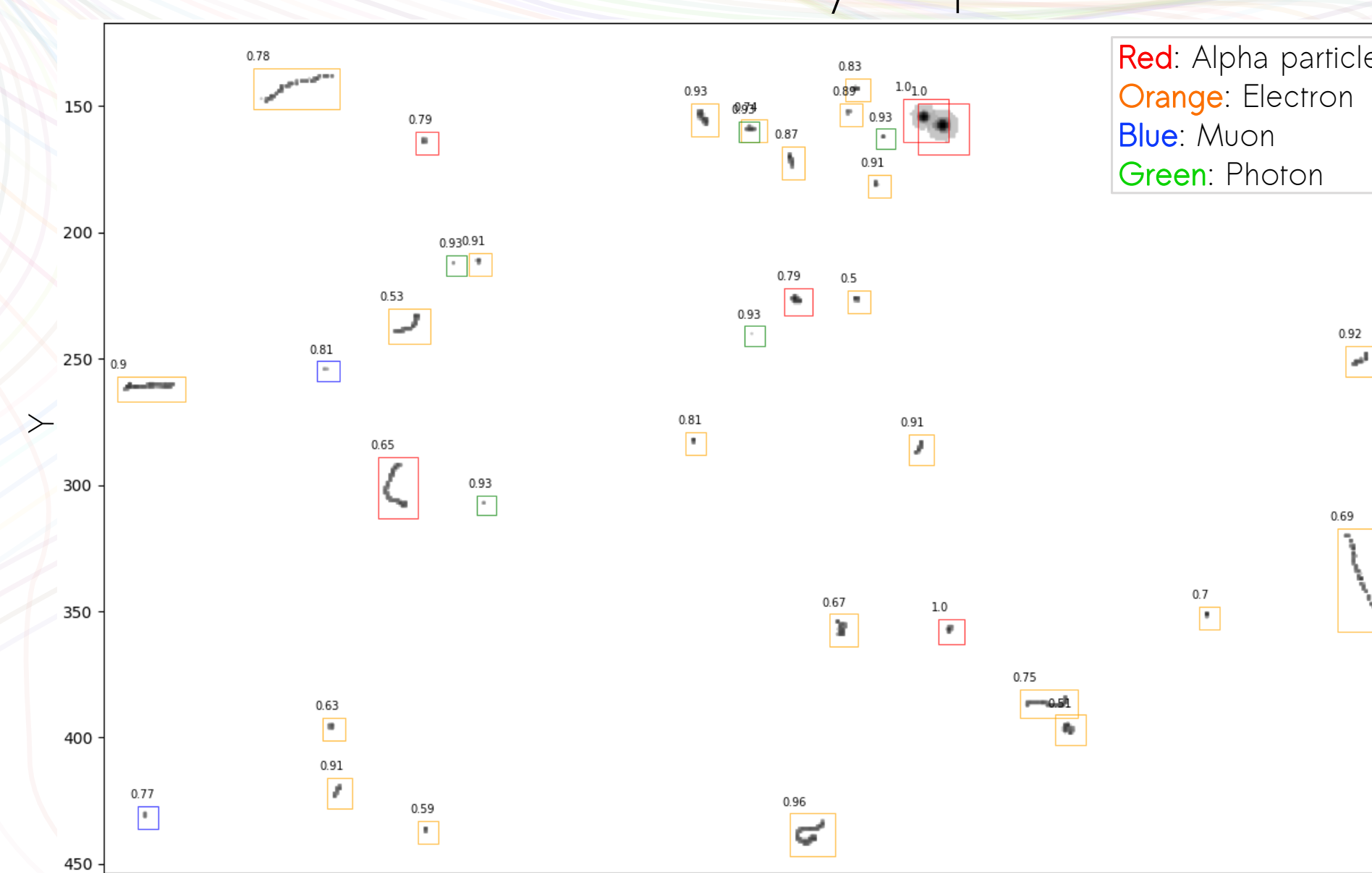
Some wrong predictions

Future Work

- **Integration in Timepix4 software**: connected to online and offline clustering and analysis.
- **New dataset**: use a simulation to generate new labelled clusters.
- **More parameters**: add floating paramers for each cluster, such as total energy, roundness, linearity, ...
- **Implementation on FPGA**: using VitisAI and hls4ml

Acknowledgements

Contact:
viola.cavallini@cern.ch