[dRICH meeting – September 10th 2025]

Last updates on AI-based distributed data reduction for the dRICH detector

(INFN Sezione di Roma - APE Lab)

Speaker: Cristian Rossi
(cristian.rossi@roma1.infn.it)

dRICH: Data reduction (features)

Online Signal/ Noise discrimination using ML

- **Signal (i.e. Merged Phys Signal + Bkg)**:
  - **Physics Signal:**
    - e.g DIS
  - **Physics Background:**
    - e/p with beam pipe
    - Synchrotron radiation (not included yet)

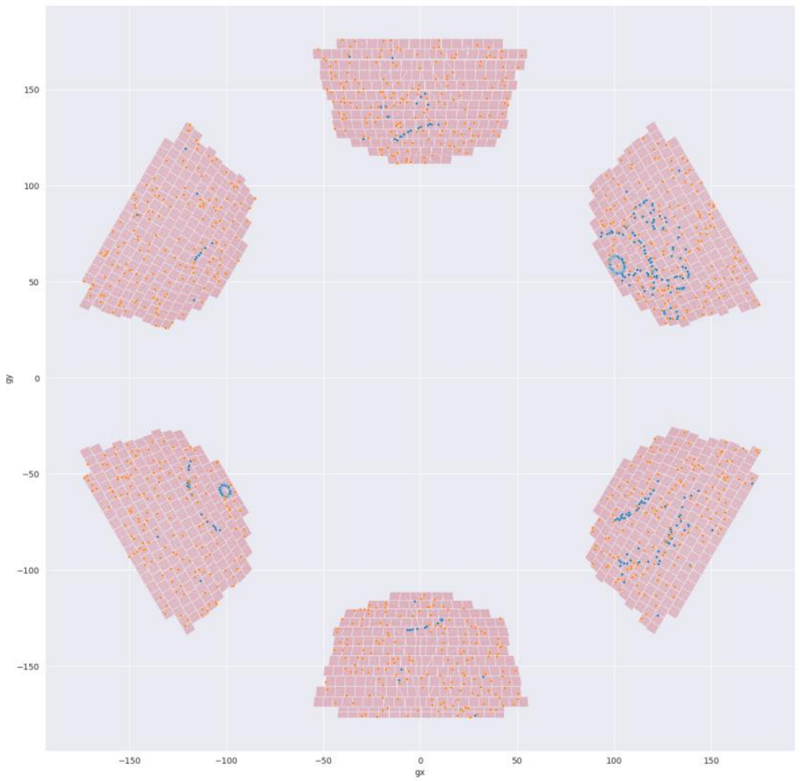
- **SiPM Noise:**
  - Dark count rate (DCR) modelled in the reconstruction stage (*recon.rb* eic-shell method OR PYTHON4NOISE)

ML task:

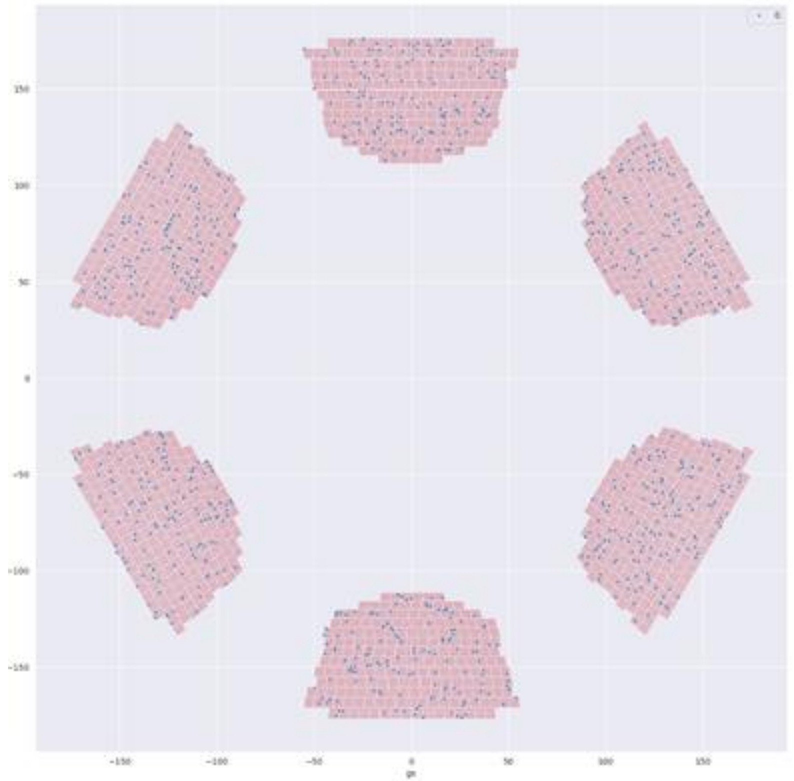
Discriminate between **Noise Only** and **Signal + Noise** events

dRICH DataReduction: Features Def.

Phys Signal+Phys Background+Noise

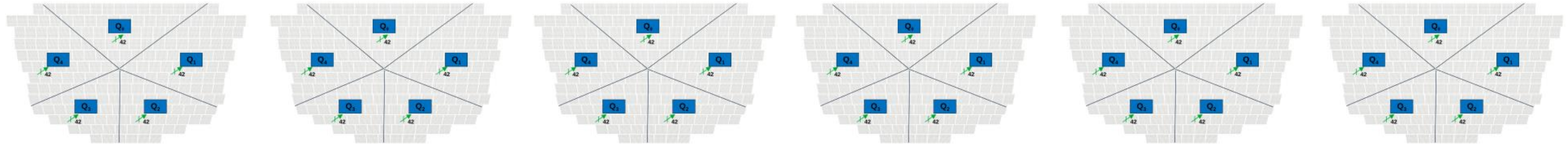


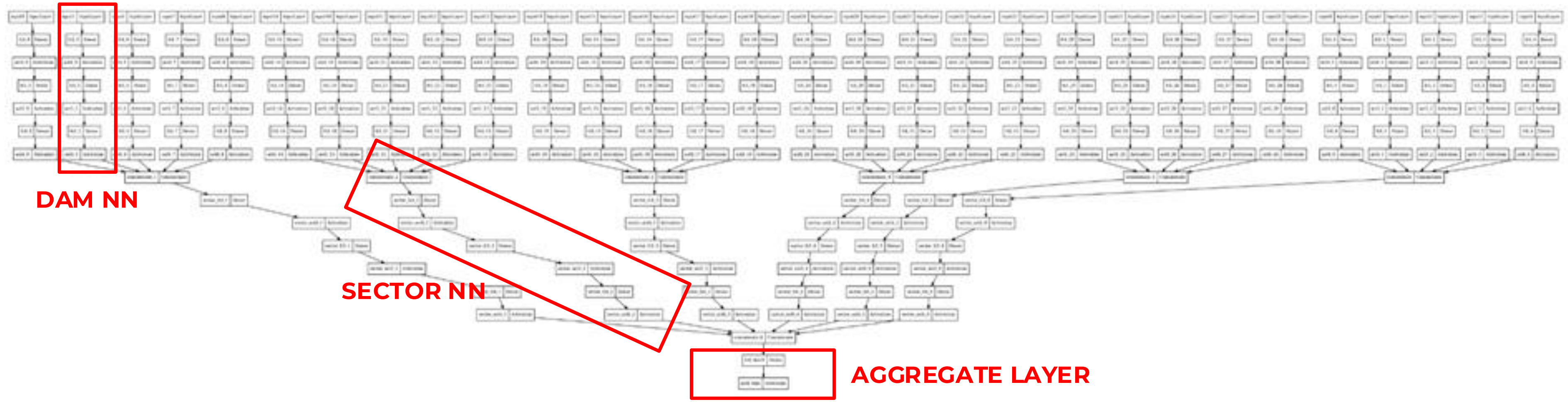
Noise Only



dRICH Data reduction: Tensorflow-Keras Model definition

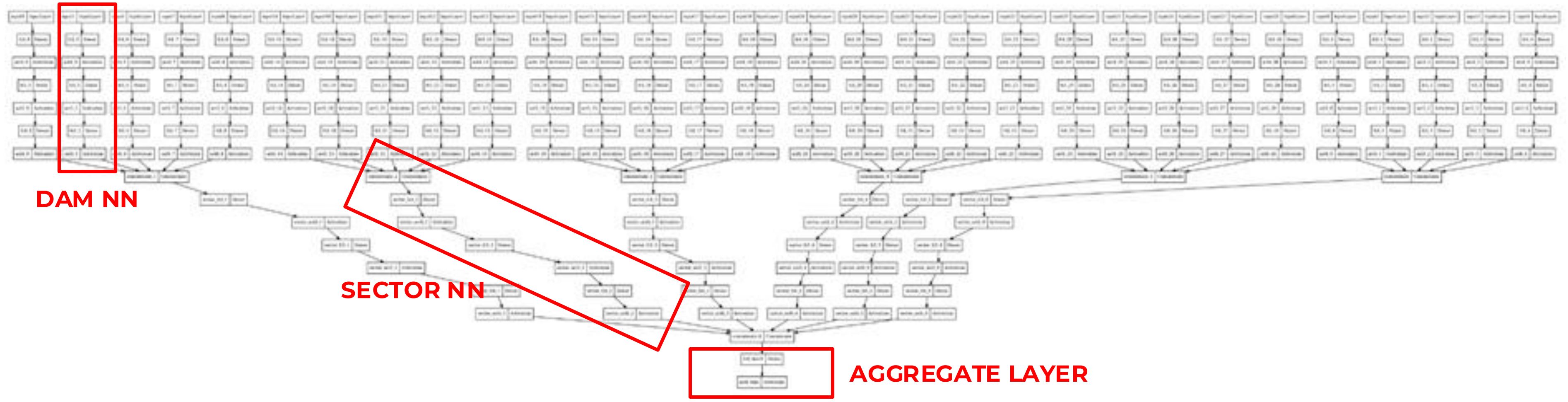
- Coherently with the hardware design composition, we trained **30** (# of subsectors x # of sectors) **parallel MLP networks** to be deployed on 30 DAM FPGAs.





dRICH Data reduction: Tensorflow-Keras Model definition

- Coherently with the hardware design composition, we trained **30** (# of subsectors x # of sectors) **parallel MLP networks** to be deployed on 30 DAM FPGAs.
- These 30 DAM network are then concatenated to feed 6 intermediate model (called **Sector NN**) to be deployed on the TP FPGA. Each Sector NN work on the aggregated information of a single sector (5 DAMs)
- The 6 outputs from Sector NNs are then aggregated and processed in a **lightweight TP NN** (single layer, 5 neurons), deployed on the same TP FPGA

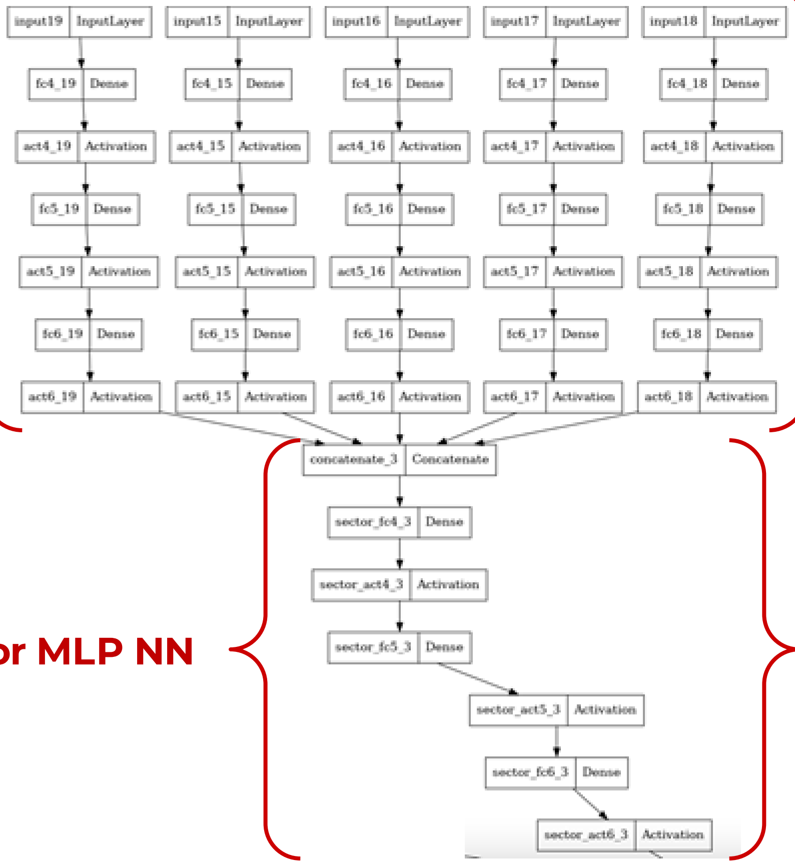


dRICH Data reduction: Tensorflow-Keras Model definition

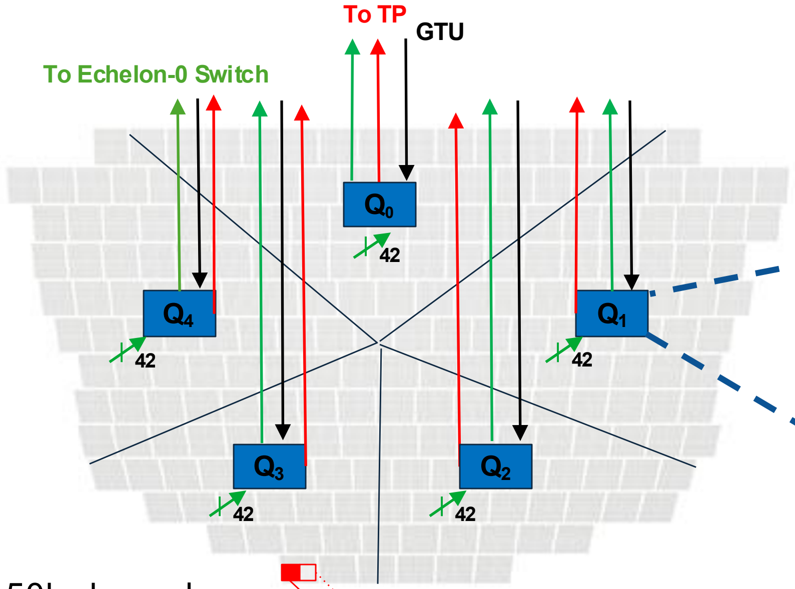
5 MLP DAM NNs (same sector)

For each sector, 5 MLP DAM output (**embedding**) are concatenated and then used to feed the Sector MLP model ⇒ **sector local information** extracted from the incoming data to perform the final prediction

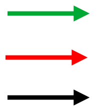
Sector MLP NN

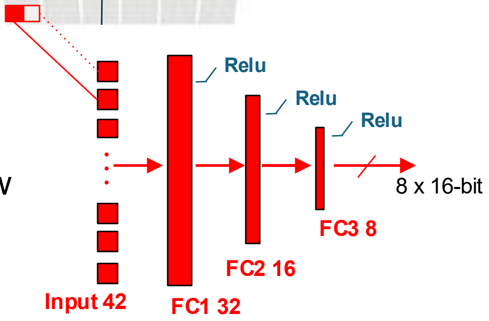


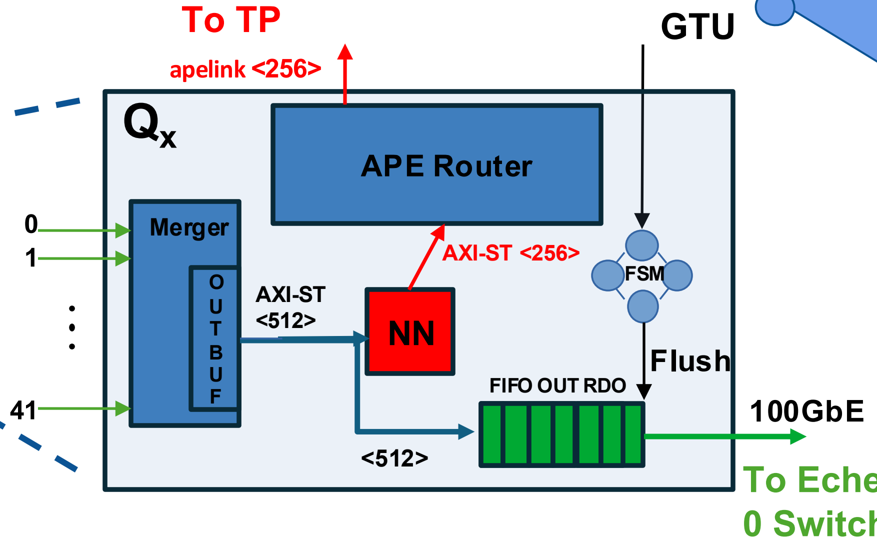
dRICH Data Reduction Stage on FPGA: subsectors' design



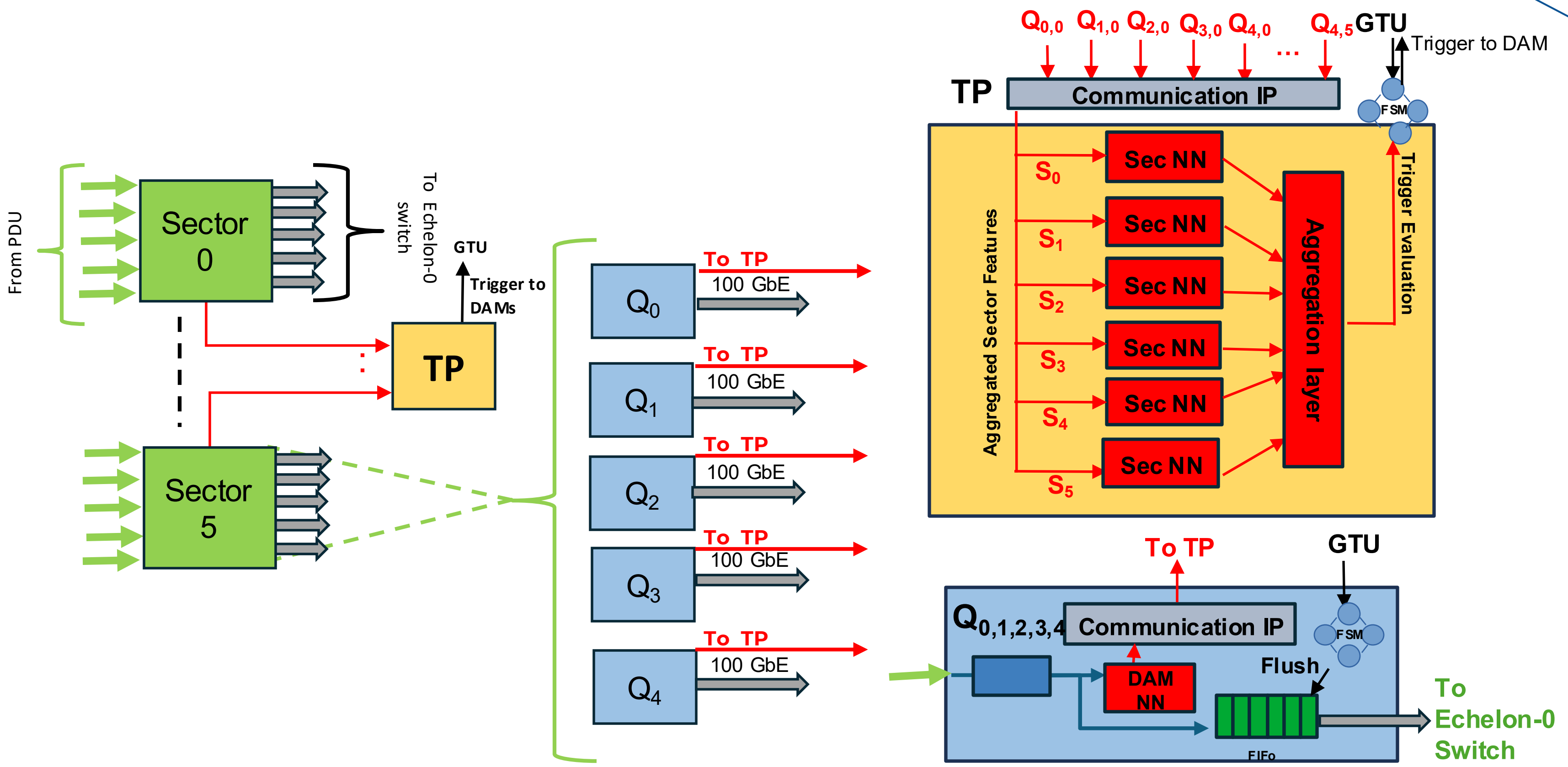
~50k channels per sector





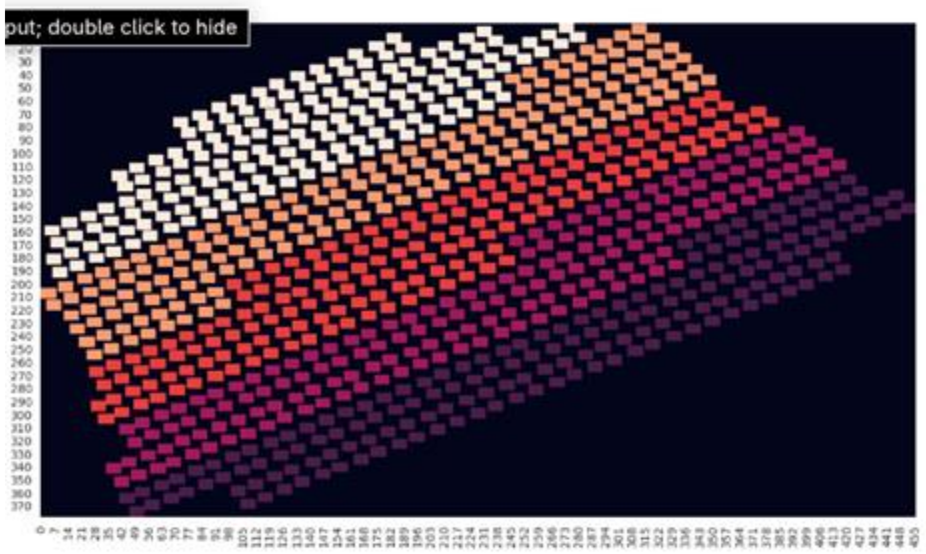


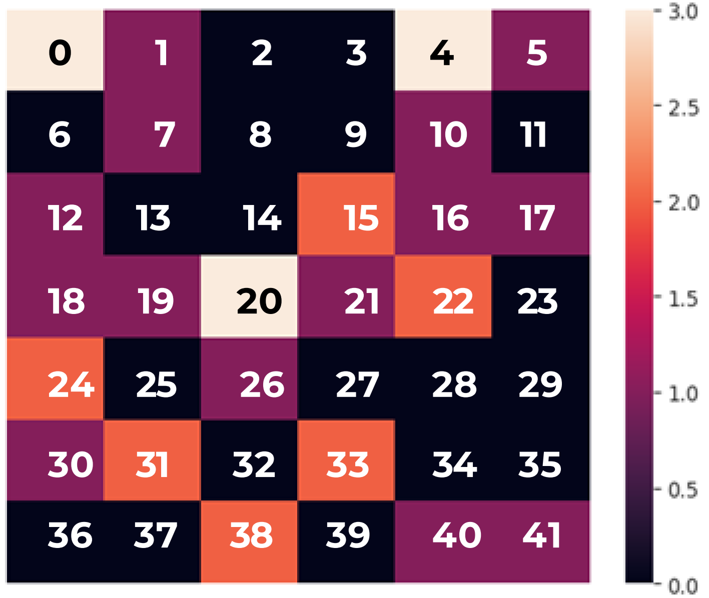
dRICH Data Reduction Stage on FPGA: subsectors' design



dRICH: Data reduction ⇒ Subsectors

- From our design proposal, we indicate **42 input links for each DAM** occurring into the streaming readout data reduction computation.
  ⇒ This number **(42)** is coherent with the number of expected PDUs per subsector (~210/5).
  (“Answer to the Ultimate Question of Life, the Universe, and Everything”)
- Thus, to cope with the realistic composition of the dRICH hardware readout, we decided to take the **information of each PDU as input** for the respective subsector MLP NN model





dRICH Data reduction ⇒ Dataset

Montecarlo Events
(Physics Sig + Physics Bg)

⇓

(GEANT4) Simulation
(ePIC detectors output)

⇓

Reconstruction
(digitization, quantum efficiency, safety factor)

ePIC software framework workflow
(e.g, EICrecon library)

Up to now, we have produced **~1.6M events** to **train** and **test** our ML models ⇒ Various **noise rates** and **noise models** for each generated dataset

Noise-Only Dataset

⇑

+

(Python) Noise Generation
(dRICH SiPMs Dark count)

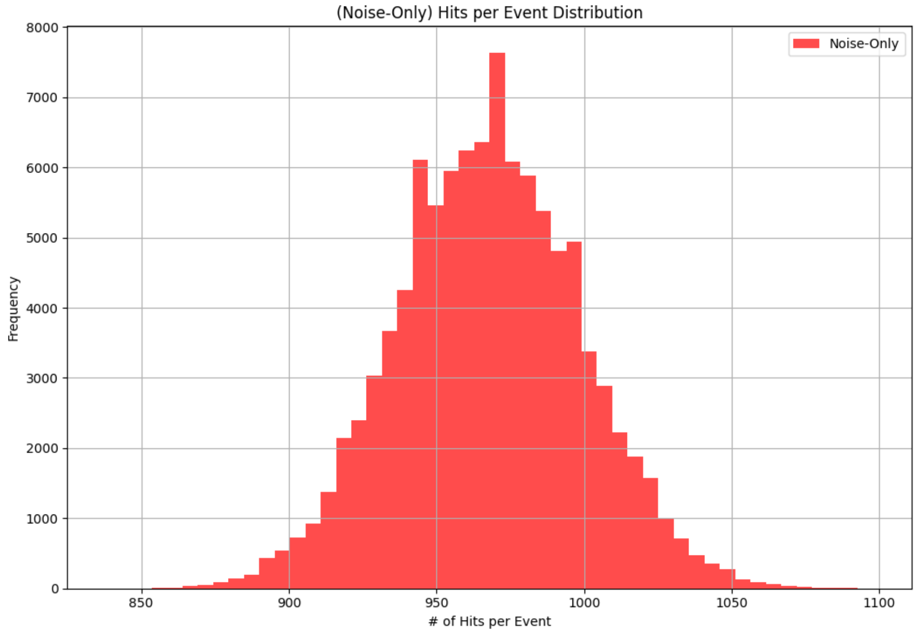
=

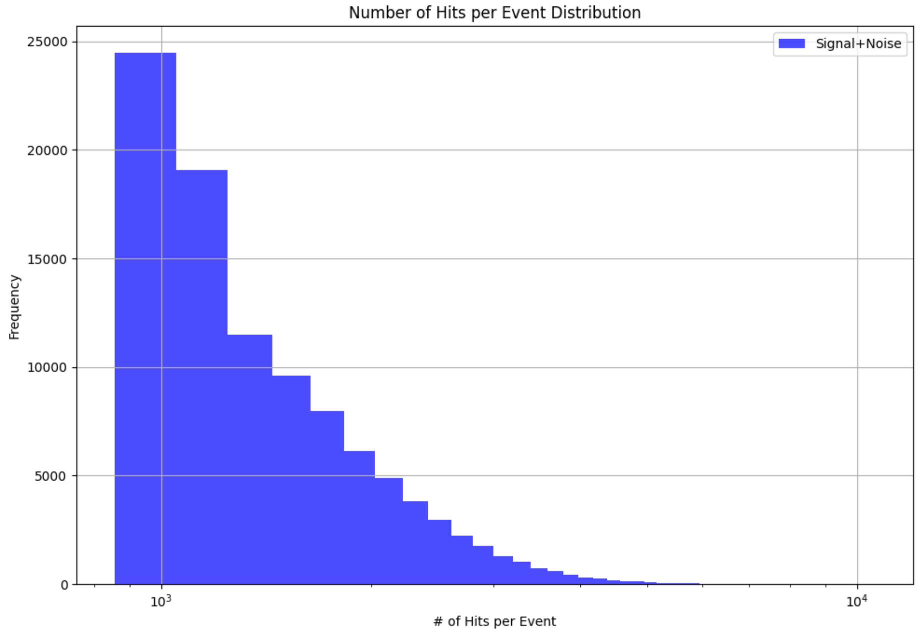
Signal+Noise Dataset

dRICH Data reduction: Noise Distribution

- **Gaussian** dark current SiPM **noise hits distribution**:
  - mean = noiseRate*noiseTimeWindow*NumberOfSiPMsDRICH
  - sigma = 0.1*avg
  - noiseTimeWindow = 10 ns

noiseRate = 300 kHz





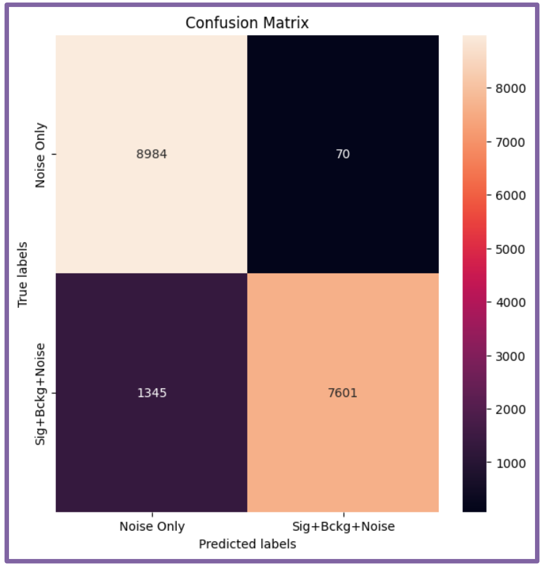
dRICH Data reduction: Tensorflow training and evaluation

- ➔ We trained the 30 MLP DAM models concatenated to the single MLP TP model by using 100k Signal+Noise and 100k Noise Only events.
- ➔ **200k balanced dataset** (90% training set, 8% testing set, 2% validation set) for any of the considered noise hits distribution models, varying their Dark Count Rate parameter:

  - ◆ **Gaussian Noise Hits Distribution model:**
    - noiseRate = **25 kHz**, timeWindow = 10ns;
    - noiseRate = **50 kHz**, timeWindow = 10ns;
    - noiseRate = **100 kHz**, timeWindow = 10ns;
    - noiseRate = **150 kHz**, timeWindow = 10ns;
    - noiseRate = **200 kHz**, timeWindow = 10ns;
    - noiseRate = **300 kHz**, timeWindow = 10ns;

NN Model performance (100 KHz & 10ns)

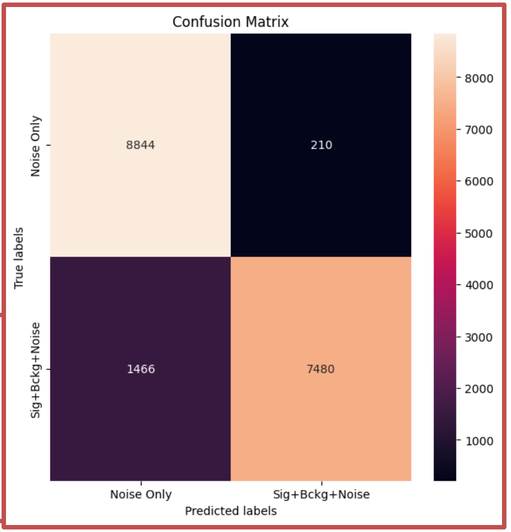
Keras model



- ❑ Accuracy = (TP+TN)/(TP+TN+FP+FN) = 0.921
- ❑ Purity = TP/(TP+FP) = 0.870
- ❑ Recall = TP/(TP+FN) = 0.992

Model Quantization

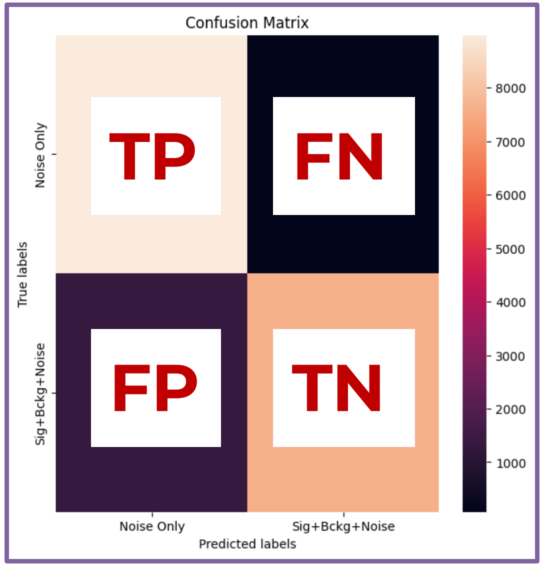
- Inputs, Activations: fixed point<16,6>
- Weights, Biases: fixed point<8,1>



- ❑ Accuracy = (TP+TN)/(TP+TN+FP+FN) = 0.906
- ❑ Purity = TP/(TP+FP) = 0.858
- ❑ Recall = TP/(TP+FN) = 0.977

NN Model performance (100 KHz & 10ns)

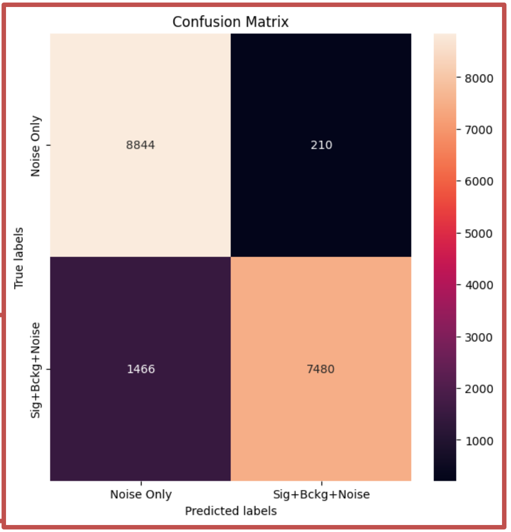
Keras model



- ❑ Accuracy = (TP+TN)/(TP+TN+FP+FN) = 0.921
- ❑ Purity = TP/(TP+FP) = 0.870
- ❑ Recall = TP/(TP+FN) = 0.992

Model Quantization

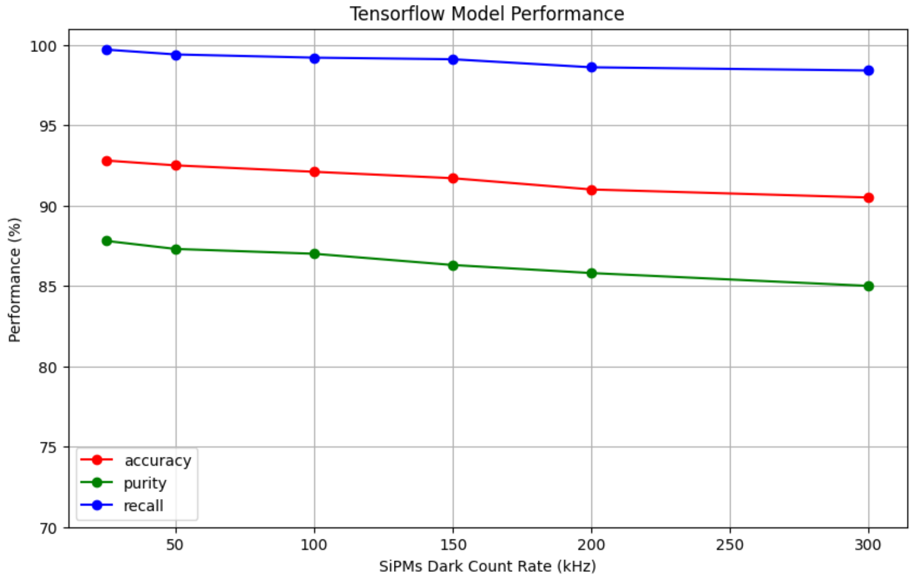
- Inputs, Activations: fixed point<16,6>
- Weights, Biases: fixed point<8,1>

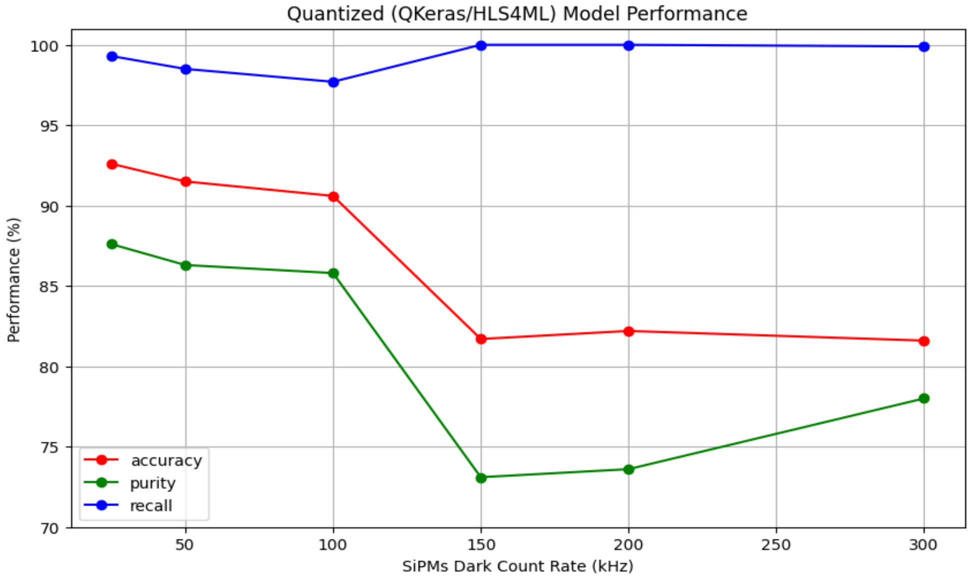


- ❑ Accuracy = (TP+TN)/(TP+TN+FP+FN) = 0.906
- ❑ Purity = TP/(TP+FP) = 0.858
- ❑ Recall = TP/(TP+FN) = 0.977

NN Model performance scaling

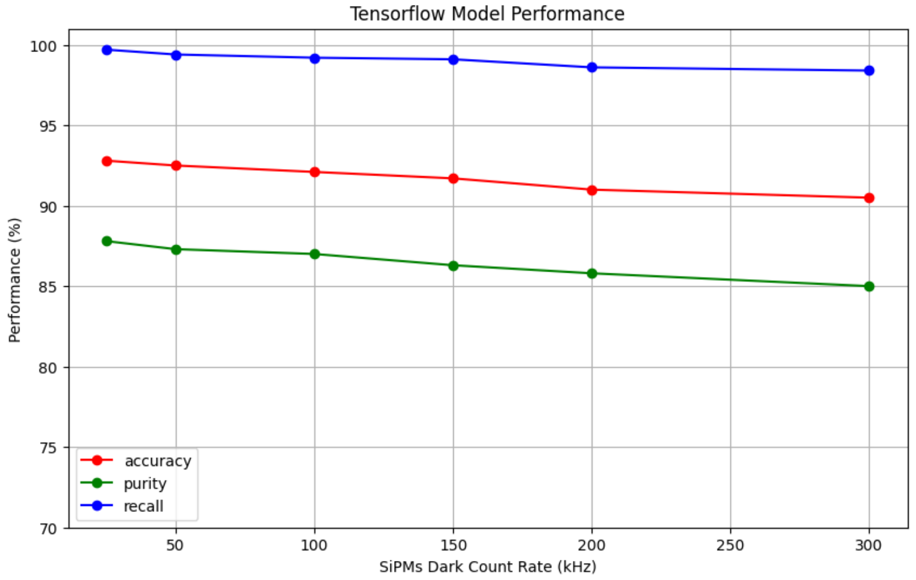
We noticed a drop of prediction performance with increasing dark count rate (e.g. increasing number of noise hits per event), but still *purity > 85%* for noisiest case (DCR = 300 kHz).
As expected, prediction performance drop after quantization step

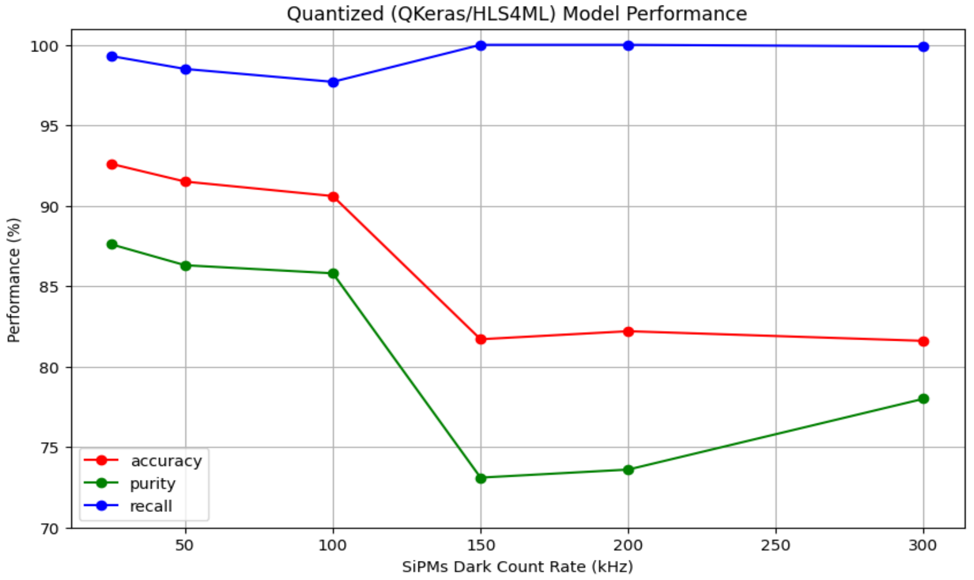




NN Model performance scaling

We noticed a drop of prediction performance with increasing dark count rate (e.g. increasing number of noise hits per event), but still *purity > 85%* for noisiest case (DCR = 300 kHz).
As expected, prediction performance drop after quantization step



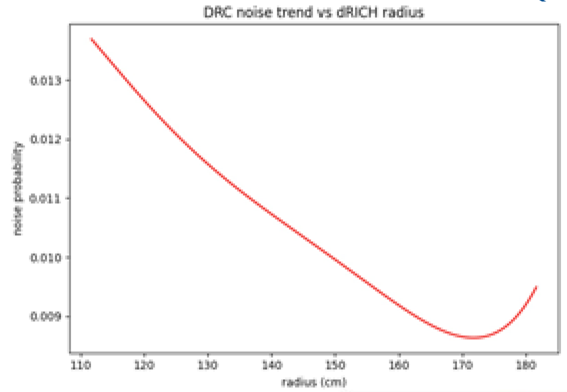


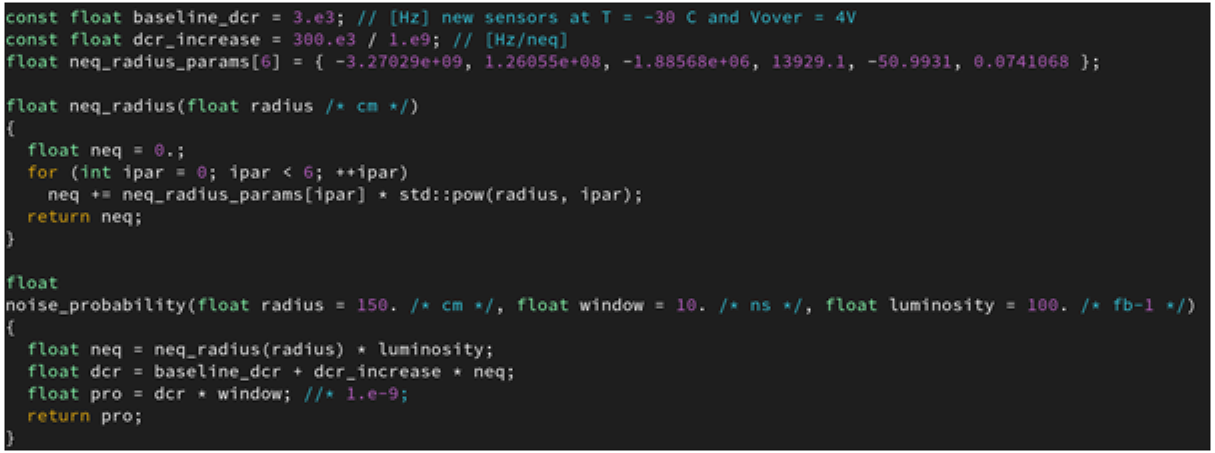
==> WHY NOT 99%??

dRICH Data reduction: Noise Distribution

- Dark current SiPM **noise hits distribution,** obtained by introducing Dark Count probability of single dRICH SiPM with a dependence on its **radial distance from the detector z-axis** and on the **integrated luminosity**
  ⇒ Implemented in EICRecon digitization step (new flag to enable new model noise)

(R. Preghenella's contribution)

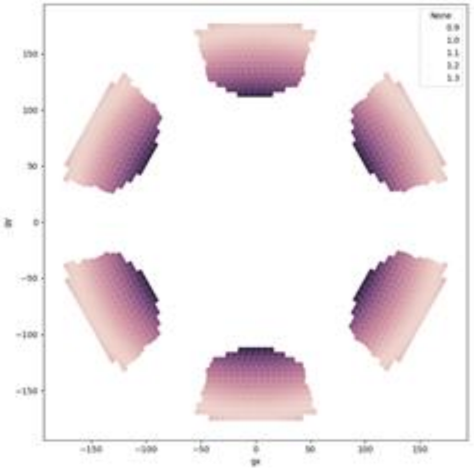


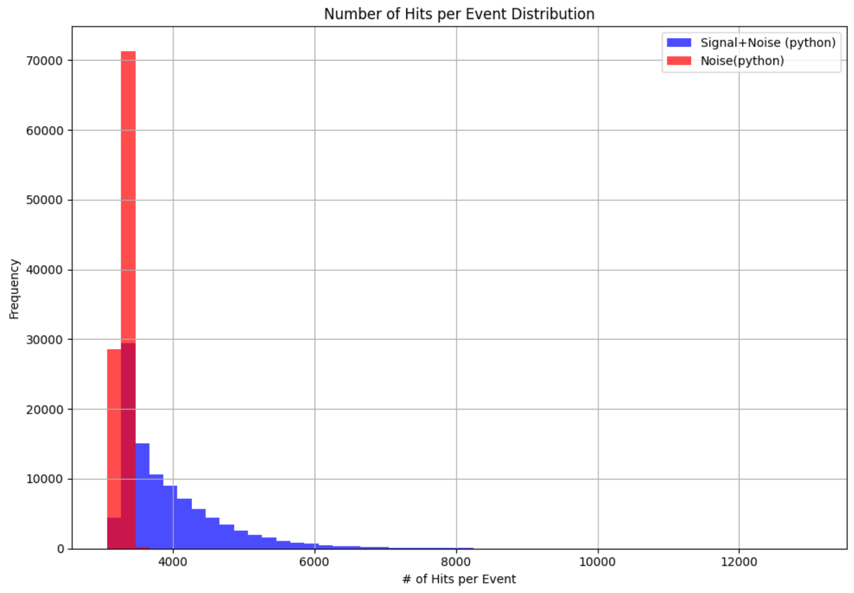


```
const float baseline_dcr = 3.e3; // [Hz] new sensors at T = -30 C and Vover = 4V
const float dcr_increase = 300.e3 / 1.e9; // [Hz/neq]
float neq_radius_params[6] = { -3.27029e+09, 1.26055e+08, -1.88568e+06, 13929.1, -50.9931, 0.0741068 };

float neq_radius(float radius /* cm */)
{
  float neq = 0.;
  for (int ipar = 0; ipar < 6; ++ipar)
    neq += neq_radius_params[ipar] * std::pow(radius, ipar);
  return neq;
}

float
noise_probability(float radius = 150. /* cm */, float window = 10. /* ns */, float luminosity = 100. /* fb-1 */)
{
  float neq = neq_radius(radius) * luminosity;
  float dcr = baseline_dcr + dcr_increase * neq;
  float pro = dcr * window; //* 1.e-9;
  return pro;
}
```

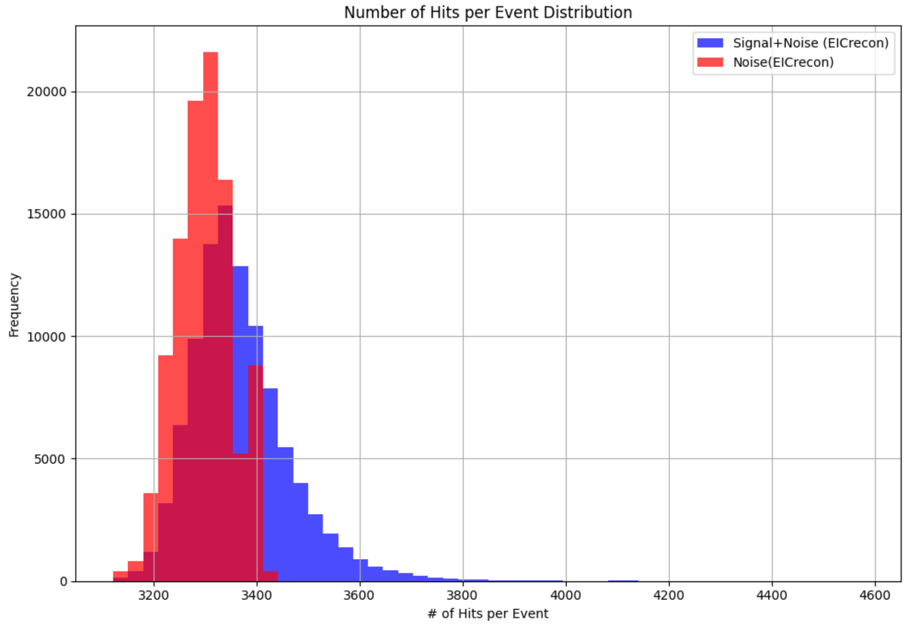




Performance >80% (@100fb-1)

Eic-shell version=25.06

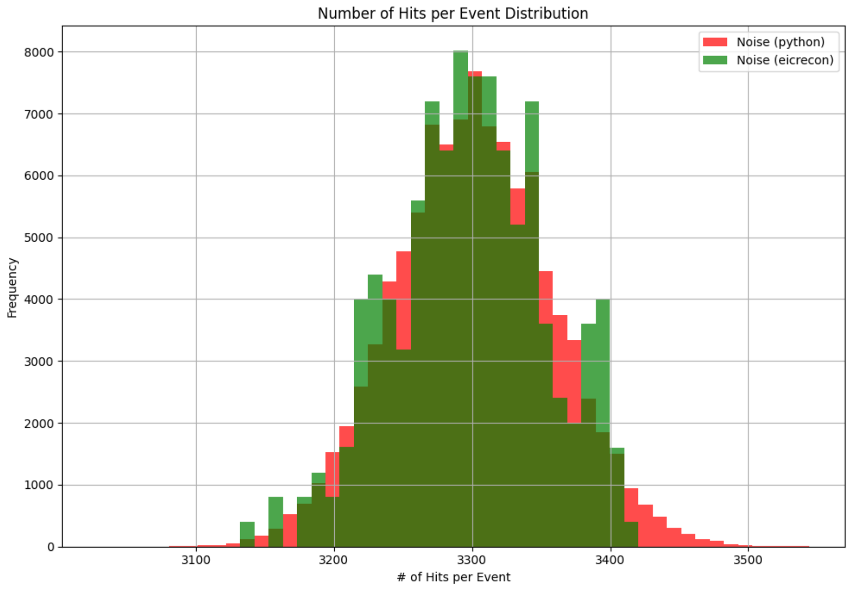
Noise: Python



Performance ~99% (@100fb-1)

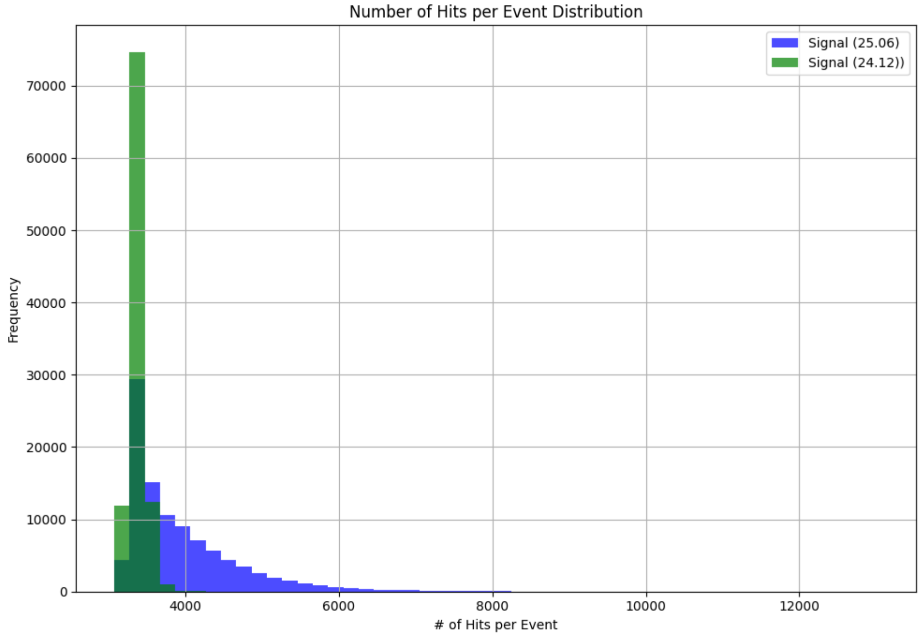
Eic-shell version=24.12

Noise: EICrecon



Noise comparison (@100fb-1)

==> same distro mean



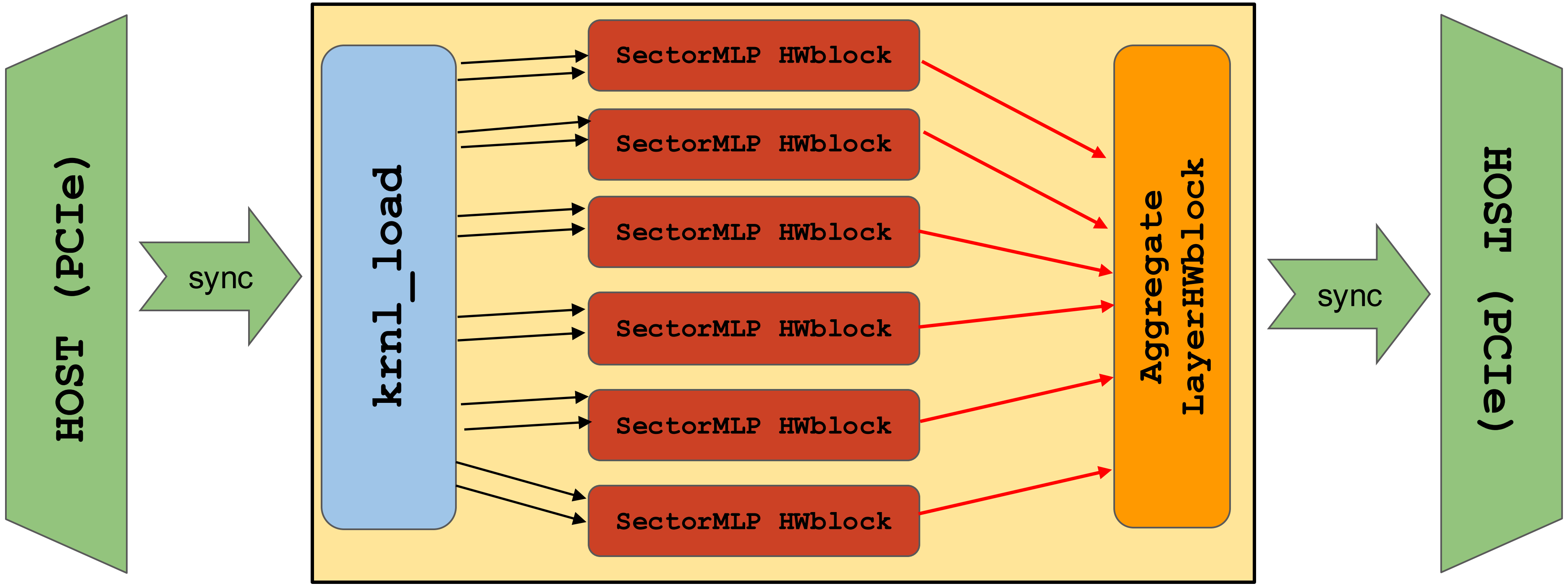
Signal Comparison (@100fb-1)

Eic-shell version=24.12 vs 25.06

==> same starting MC files

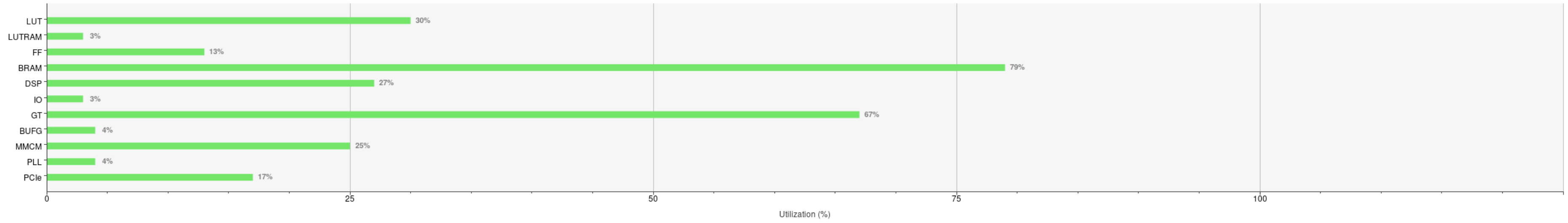
dRICH Data reduction stage on FPGA: HLS4ML ⇒ HW implementation

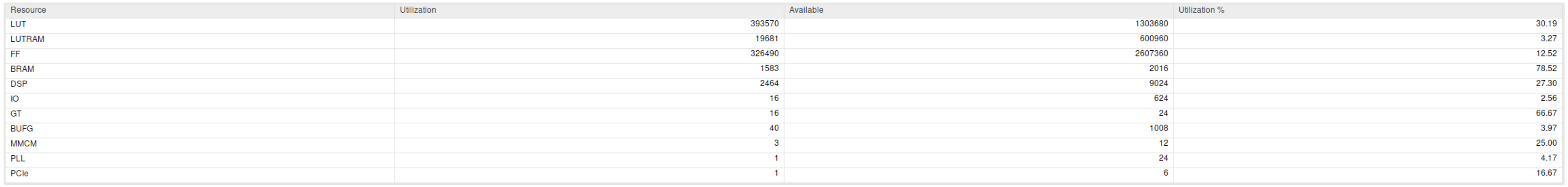
→ To validate the correct implementation of the **TP NN model (6 Sector MLP + Aggregate Layer) HLS4ML blocks** and evaluate system's performance, we decide to design an HW toy-model to prove the correct behaviour of the firmware on our Xilinx Alveo U280.



dRICH Data reduction: HLS4ML ⇒ (FPGA) HW Synthesis

From **reports** after Vitis synthesis, current TP NN design (6 Sector TP NN + Aggregation MLP Layer) correctly fits into the available HW resources (of test Xilinx Alveo U280 board)
⇒ **high BRAM utilization** due to allocation of 6 different sets of weights and biases
⇒ **occupation percentage** to take into account when moving to the target HW (FELIX: Xilinx Versal Prime

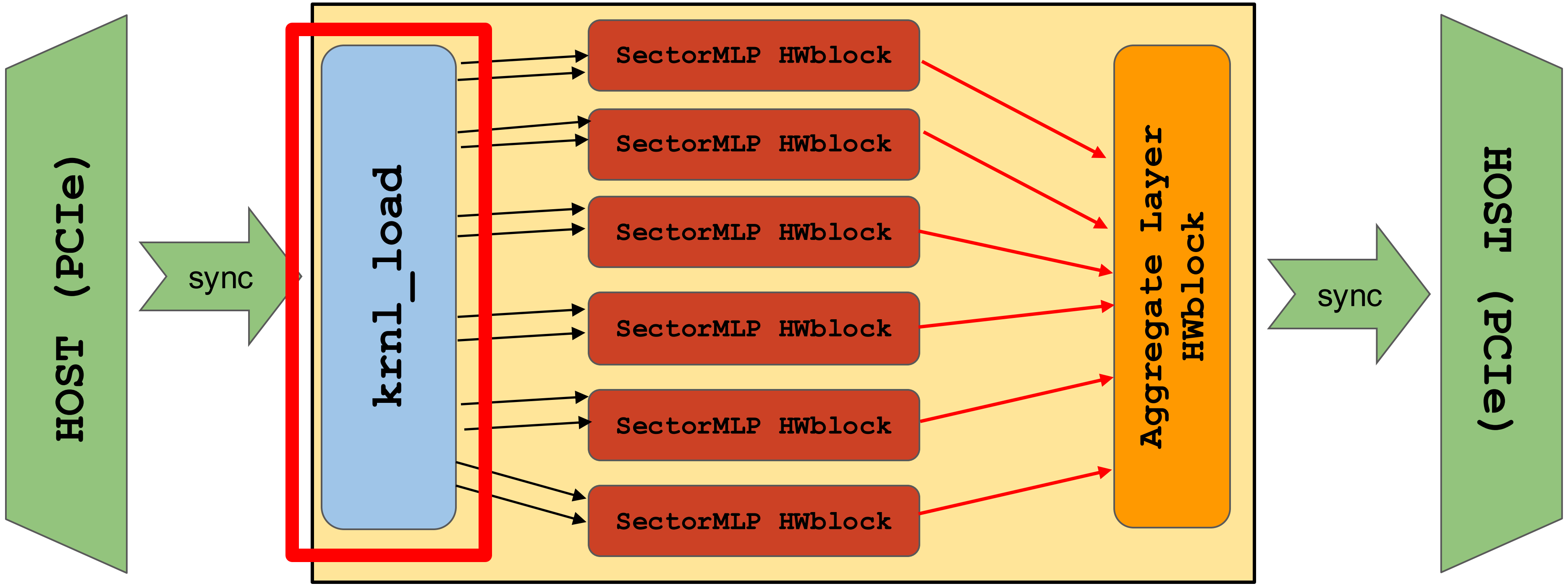




Resource	Utilization	Available	Utilization %
LUT	393570	1303680	30.19
LUTRAM	19681	600960	3.27
FF	326490	2607360	12.52
BRAM	1583	2016	78.52
DSP	2464	9024	27.30
IO	16	624	2.56
GT	16	24	66.67
BUFG	40	1008	3.97
MMCM	3	12	25.00
PLL	1	24	4.17
PCIe	1	6	16.67

dRICH Data reduction stage on FPGA: HLS4ML ⇒ HW implementation

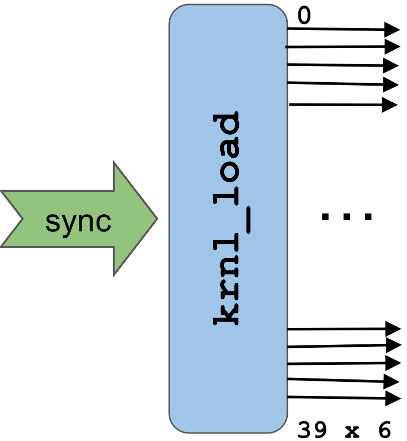
- To validate the correct implementation of the **TP NN model (6 Sector MLP + Aggregate Layer) HLS4ML blocks** and evaluate system's performance, we decide to design an HW toy-model to prove the correct behaviour of the firmware on our Xilinx Alveo U280.

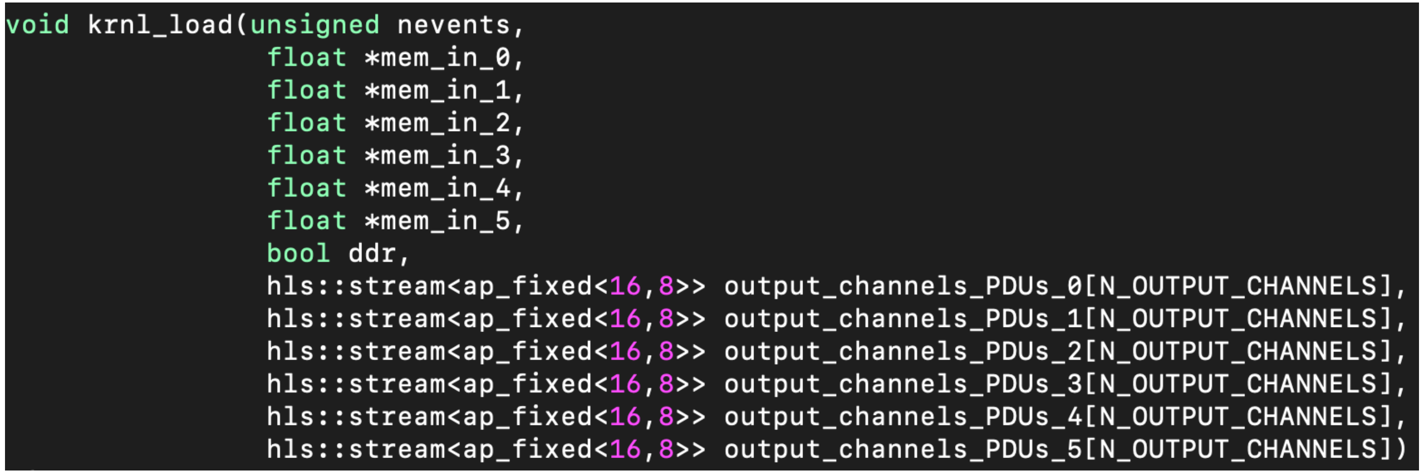


dRICH Data reduction stage on FPGA: HLS4ML ⇒ HW implementation

- **krnl_load** is connected to the Host CPU via PCIe bus, allowing to load events data on the FPGA DDR. Corresponding input data are sent to each of the 6 **Sector MLP blocks** through 40 input hls::stream<ap_fixed<16,8>>.

- By disabling the *ddr* kernel flag, **krnl_load** can send through the system few events data (O(10)) already loaded on the FPGA BRAM during firmware synthesis. In this way, **throughput** measurements can be performed without **DDR reading bottleneck**

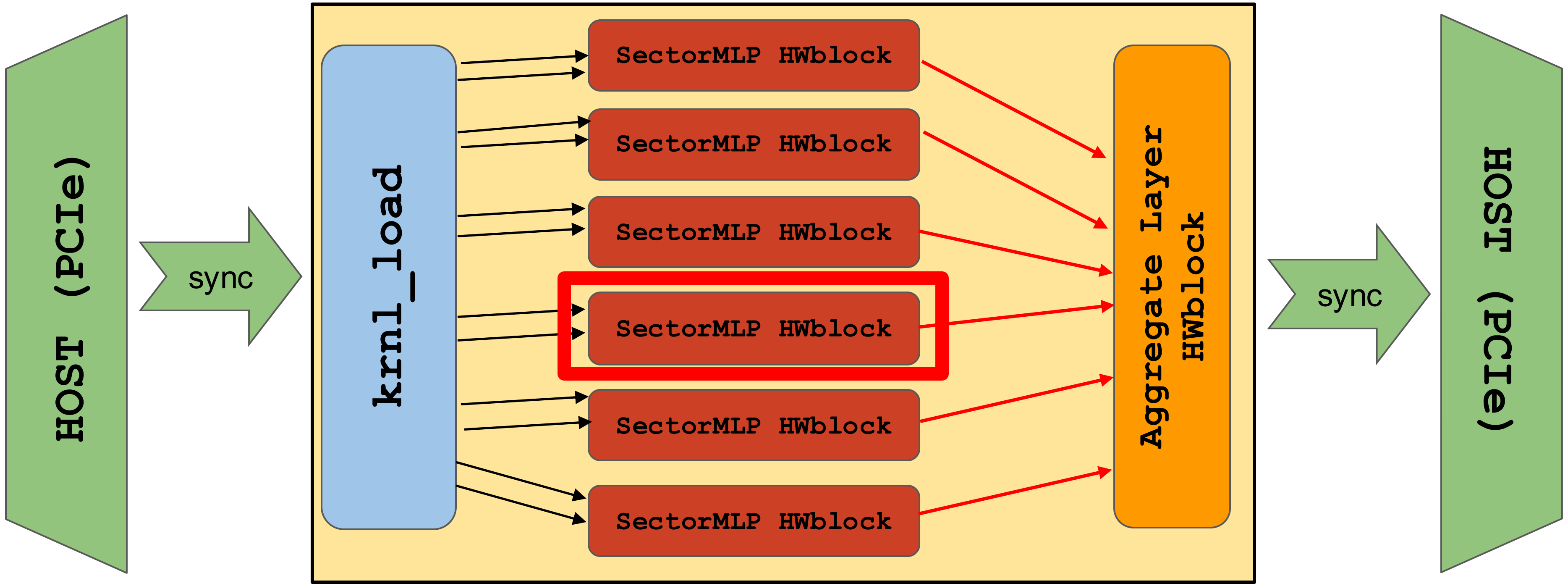




```
void krnl_load(unsigned nevents,
                float *mem_in_0,
                float *mem_in_1,
                float *mem_in_2,
                float *mem_in_3,
                float *mem_in_4,
                float *mem_in_5,
                bool ddr,
                hls::stream<ap_fixed<16,8>> output_channels_PDUs_0[N_OUTPUT_CHANNELS],
                hls::stream<ap_fixed<16,8>> output_channels_PDUs_1[N_OUTPUT_CHANNELS],
                hls::stream<ap_fixed<16,8>> output_channels_PDUs_2[N_OUTPUT_CHANNELS],
                hls::stream<ap_fixed<16,8>> output_channels_PDUs_3[N_OUTPUT_CHANNELS],
                hls::stream<ap_fixed<16,8>> output_channels_PDUs_4[N_OUTPUT_CHANNELS],
                hls::stream<ap_fixed<16,8>> output_channels_PDUs_5[N_OUTPUT_CHANNELS])
```

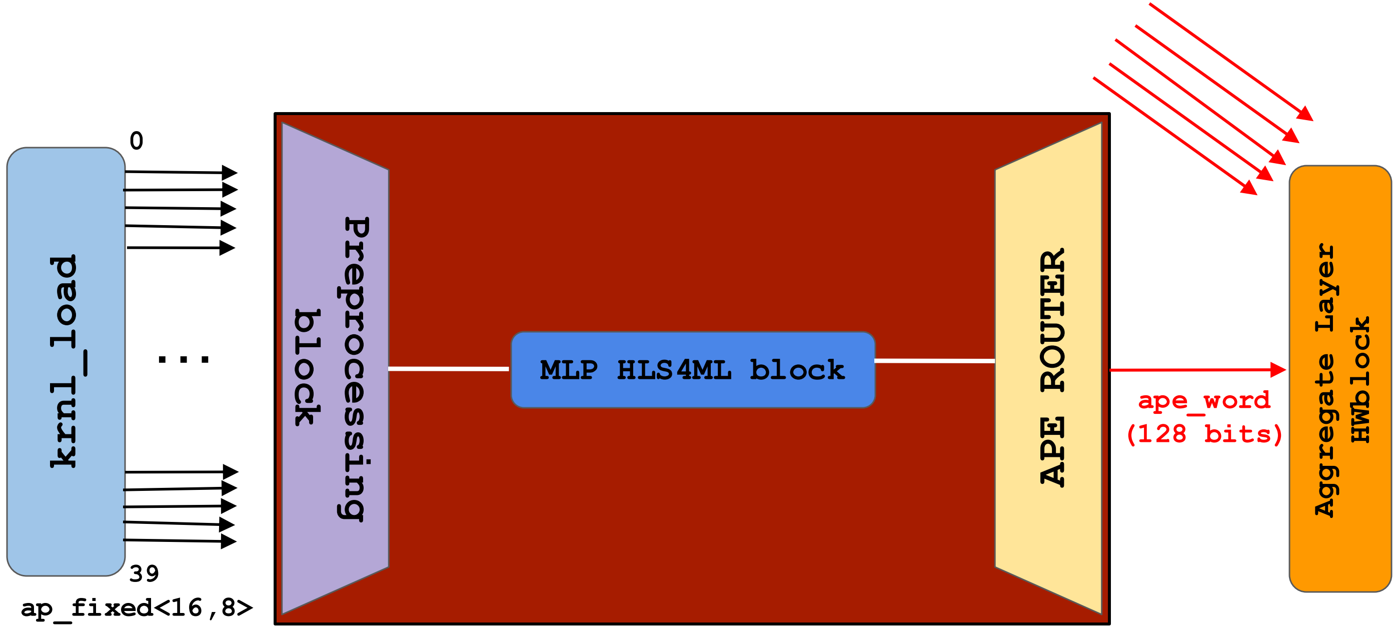
dRICH Data reduction stage on FPGA: HLS4ML ⇒ HW implementation

→ To validate the correct implementation of the **TP NN model (6 Sector MLP + Aggregate Layer) HLS4ML blocks** and evaluate system's performance, we decide to design an HW toy-model to prove the correct behaviour of the firmware on our Xilinx Alveo U280.



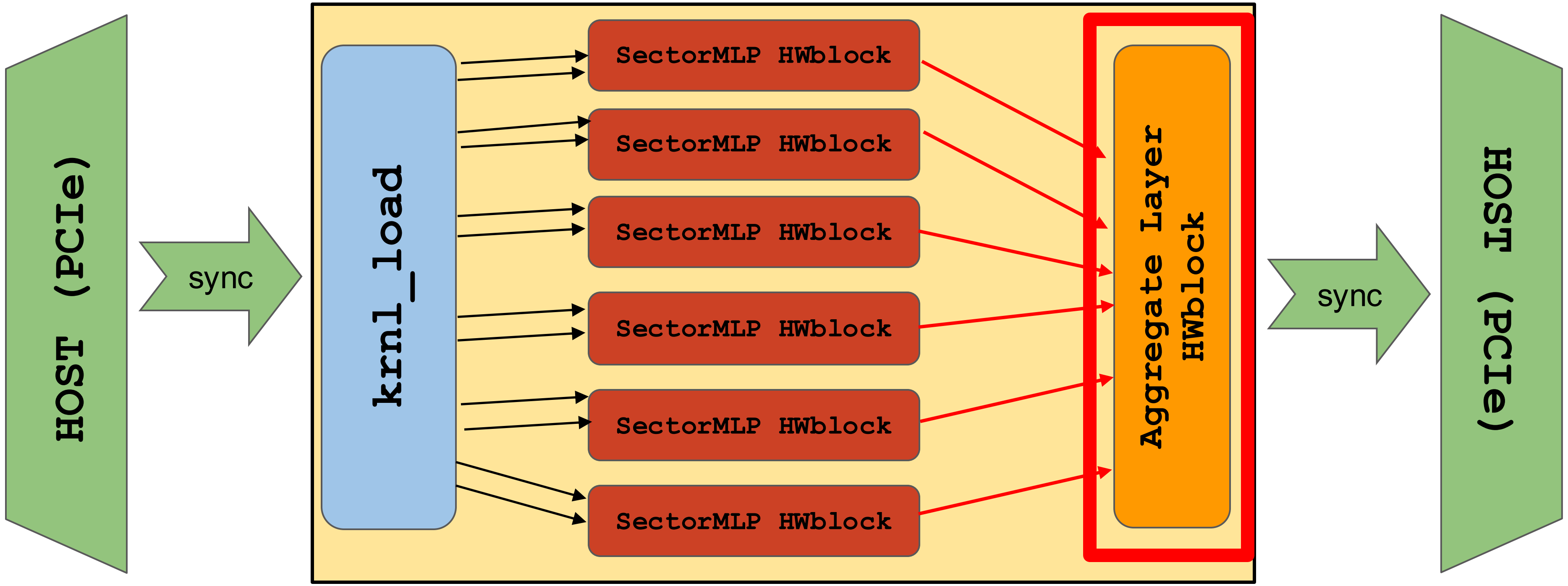
dRICH Data reduction stage on FPGA: HLS4ML ⇒ HW implementation

- The 40 input hls::stream<ap_fixed<16,8>> are connected to the **preprocessing block**, which merges the whole set of input in order to feed the **MLP HLS4ML block**. Here, the HW NN computes its output by using ap_fixed<8,0> weights and biases. The output, composed by 4 features, are then merged into a single *ape_word* of 128bits and then sent through the network via the **APEIRON Switch**



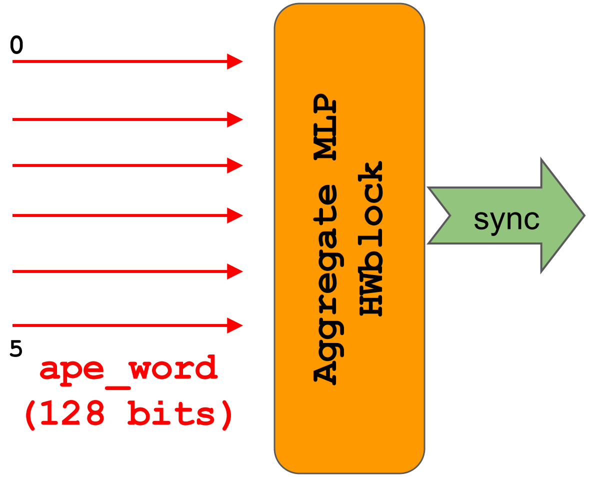
dRICH Data reduction stage on FPGA: HLS4ML ⇒ HW implementation

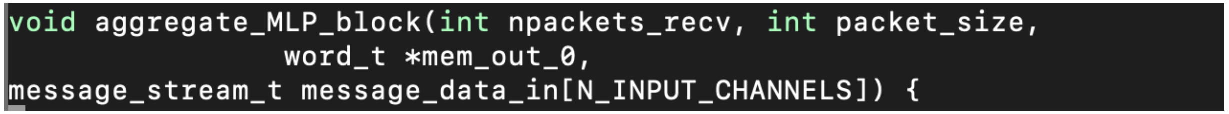
→ To validate the correct implementation of the **TP NN model (6 Sector MLP + Aggregate Layer) HLS4ML blocks** and evaluate system's performance, we decide to design an HW toy-model to prove the correct behaviour of the firmware on our Xilinx Alveo U280.



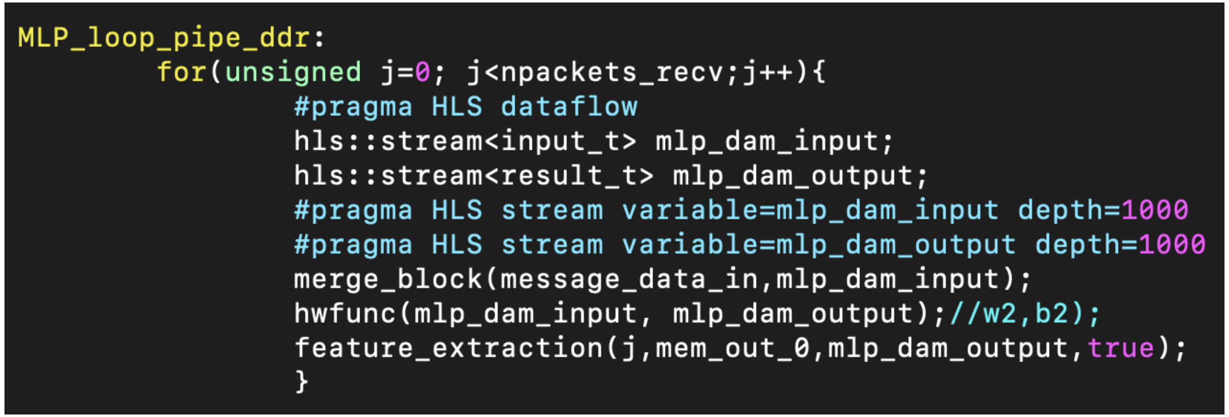
dRICH Data reduction stage on FPGA: HLS4ML ⇒ HW implementation

➔ **Aggregate MLP HWblock** receives as input 6 *ape_word* from the 6 **Sector MLP blocks**, each containing 4 features corresponding to the information extracted from a single dRICH sector. Here, incoming data are merged to feed the last MLP layer of the NN model, which finally computes the prediction. This last output is then loaded back to the Host CPU via PCIe in order to compare prediction with the true label of the processed event





```
void aggregate_MLP_block(int npackets_recv, int packet_size,
                word_t *mem_out_0,
message_stream_t message_data_in[N_INPUT_CHANNELS]) {
```



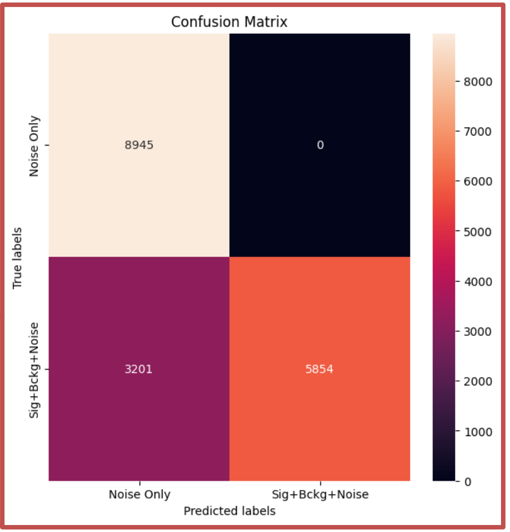
```
MLP_loop_pipe_ddr:
        for(unsigned j=0; j<npackets_recv;j++){
                #pragma HLS dataflow
                hls::stream<input_t> mlp_dam_input;
                hls::stream<result_t> mlp_dam_output;
                #pragma HLS stream variable=mlp_dam_input depth=1000
                #pragma HLS stream variable=mlp_dam_output depth=1000
                merge_block(message_data_in,mlp_dam_input);
                hwfunc(mlp_dam_input, mlp_dam_output);//w2,b2);
                feature_extraction(j,mem_out_0,mlp_dam_output,true);
                }
```

HLS4ML FPGA performance (200kHz & 10ns)

- ❑ Throughput (DDR) = 2.065 MHz
  ➔ instantiation interval II~97 cycles (@200 MHz)
- ❑ Throughput (BRAM) = 10.867 MHz
  ➔ instantiation interval II~19 cycles (@200 MHz)

Model Quantization

- Inputs, Activations: fixed point<16,6>
- Weights, Biases: fixed point<8,1>



- ❑ Accuracy = (TP+TN)/(TP+TN+FP+FN) = 0.822
- ❑ Purity = TP/(TP+FP) = 0.736
- ❑ Recall = TP/(TP+FN) = 1.000

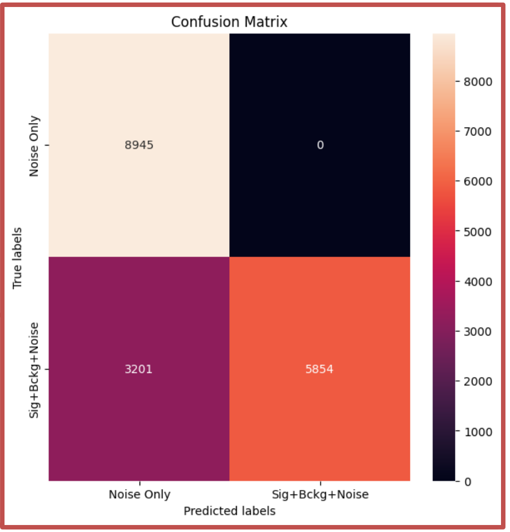
HLS4ML FPGA performance (200kHz & 10ns)

- ❑ Throughput (DDR) = 2.065 MHz
  ➔ instantiation interval II~97 cycles (@200 MHz)
- ❑ Throughput (BRAM) = 10.867 MHz
  ➔ instantiation interval II~19 cycles (@200 MHz)

Throughput issue! ==> evaluation ongoing on whole HW system instantiation interval!

Model Quantization

- Inputs, Activations: fixed point<16,6>
- Weights, Biases: fixed point<8,1>



- ❑ Accuracy = (TP+TN)/(TP+TN+FP+FN) = 0.822
- ❑ Purity = TP/(TP+FP) = 0.736
- ❑ Recall = TP/(TP+FN) = 1.000

Conclusions

- Optimization of current multi NN model performance in terms of **accuracy/purity/recall (ML parameters)** and **resources/throughput (HW implementation)** has been performed.
  ⇒ new studies for prediction' **purity enhancement** and better **quantization step** ongoing
  ⇒ different NN design under investigation

- Current Sector TP NN model under complete HW validation:
  ⇒ Xilinx Versal design for "realistic" FELIX implementation ongoing

- Deployment of the HW FPGA DAM+TPP NN model on our testbed is ongoing
  ⇒ test for the interconnection with the DAM NN **(throughput issue to be solved)**

Thanks for your attention!

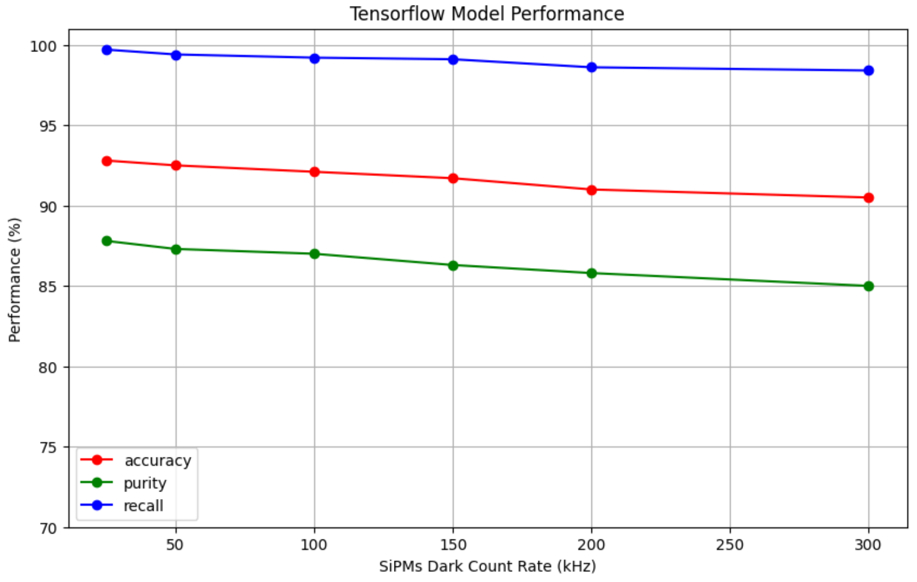
Contacts:

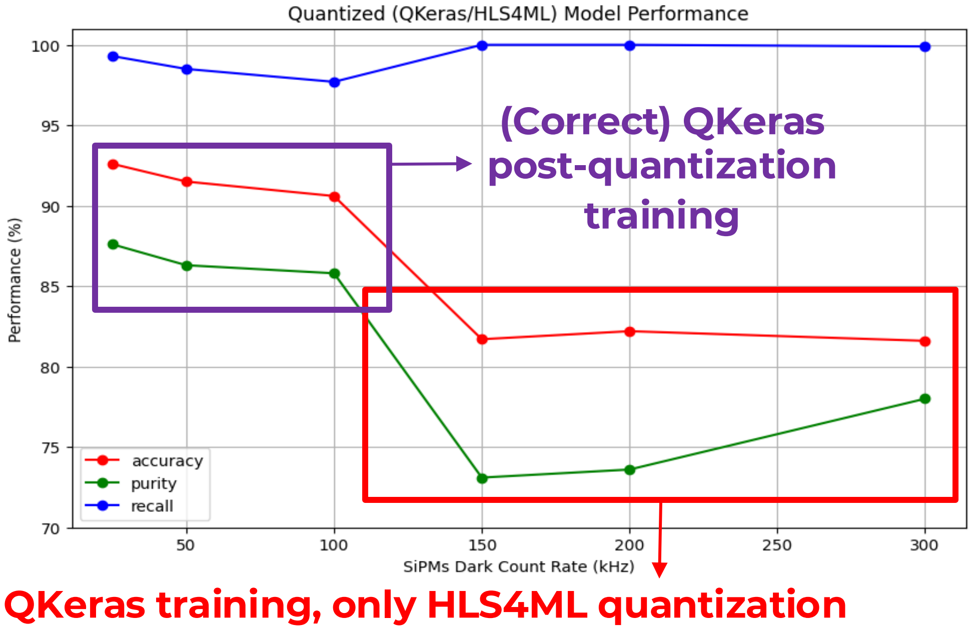
- cristian.rossi@roma1.infn.it
- alessandro.lonardo@roma1.infn.it
- https://apegate.roma1.infn.it

BACKUP SLIDES

NN Model performance scaling

We noticed a drop of prediction performance with increasing dark count rate (e.g. increasing number of noise hits per event), but still *purity > 85%* for noisiest case (DCR = 300 kHz).
As expected, prediction performance drop after quantization step



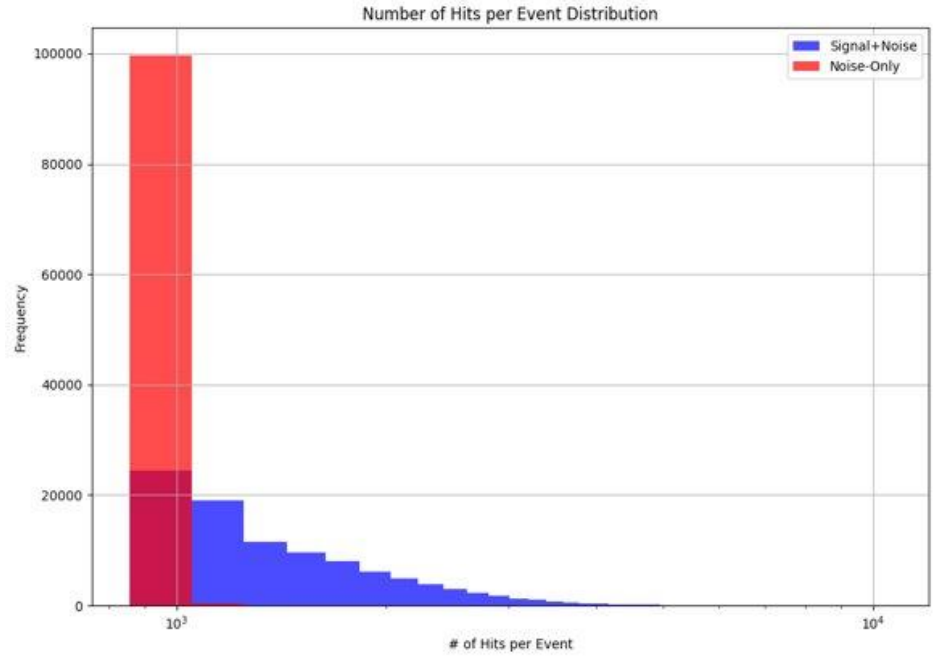


No QKeras training, only HLS4ML quantization

dRICH Data reduction: Noise Distribution

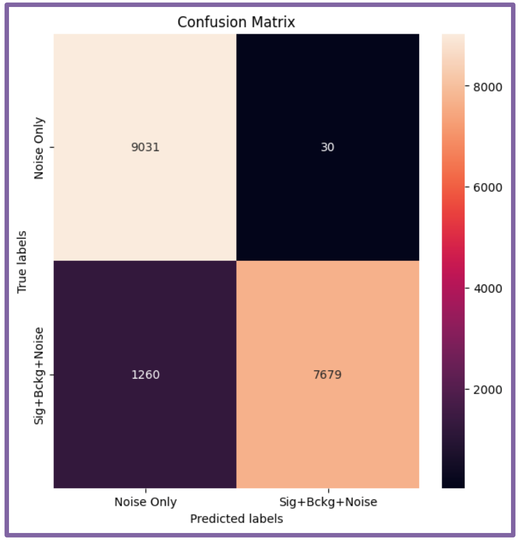
- **Gaussian** dark current SiPM **noise hits distribution**:
  - mean = noiseRate*noiseTimeWindow*NumberOfSiPMsDRICH
  - sigma = 0.1*avg
  - noiseTimeWindow = 10 ns

noiseRate = 300 kHz



NN Model performance (25 KHz & 10ns)

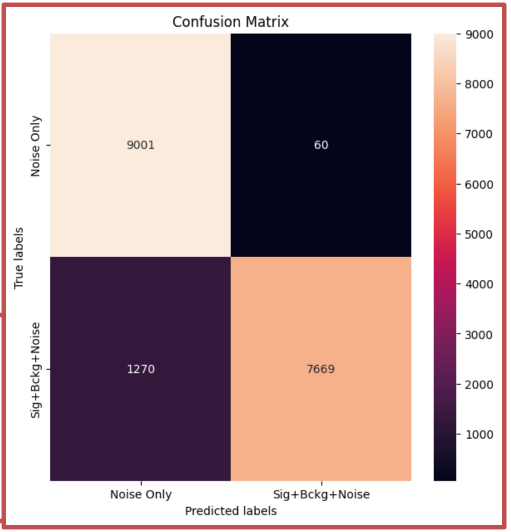
Keras model



- ❏ Accuracy = (TP+TN)/(TP+TN+FP+FN) = 0.928
- ❏ Purity = TP/(TP+FP) = 0.878
- ❏ Recall = TP/(TP+FN) = 0.997

Model Quantization

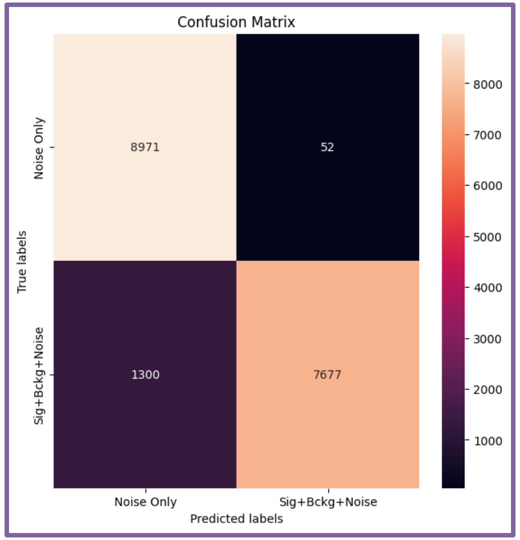
- Inputs, Activations: fixed point<16,6>
- Weights, Biases: fixed point<8,1>



- ❏ Accuracy = (TP+TN)/(TP+TN+FP+FN) = 0.926
- ❏ Purity = TP/(TP+FP) = 0.876
- ❏ Recall = TP/(TP+FN) = 0.993

NN Model performance (50 KHz & 10ns)

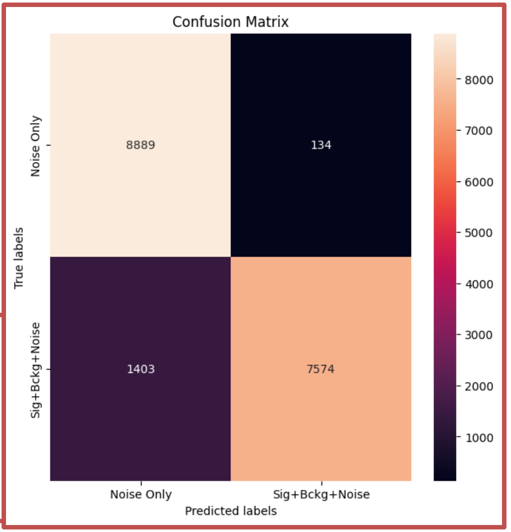
Keras model



- ❑ Accuracy = (TP+TN)/(TP+TN+FP+FN) = 0.925
- ❑ Purity = TP/(TP+FP) = 0.873
- ❑ Recall = TP/(TP+FN) = 0.994

Model Quantization

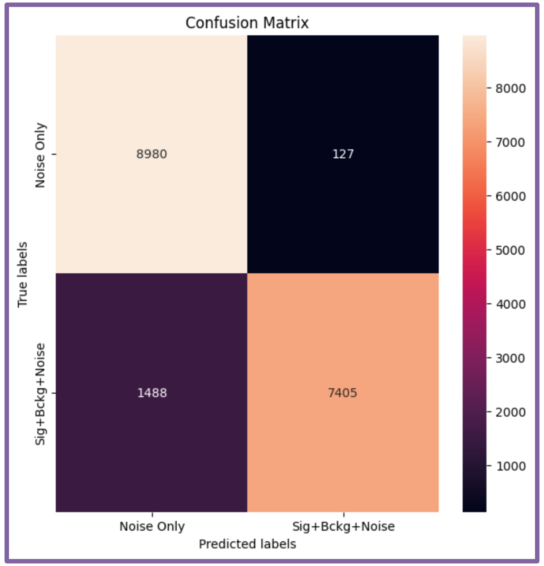
- Inputs, Activations: fixed point<16,6>
- Weights, Biases: fixed point<8,1>



- ❑ Accuracy = (TP+TN)/(TP+TN+FP+FN) = 0.915
- ❑ Purity = TP/(TP+FP) = 0.863
- ❑ Recall = TP/(TP+FN) = 0.985

NN Model performance (200 KHz & 10ns)

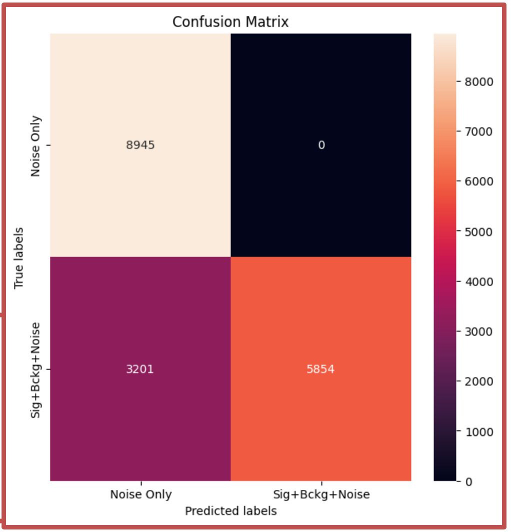
Keras model



- ❑ Accuracy = (TP+TN)/(TP+TN+FP+FN) = 0.910
- ❑ Purity = TP/(TP+FP) = 0.858
- ❑ Recall = TP/(TP+FN) = 0.986

Model Quantization

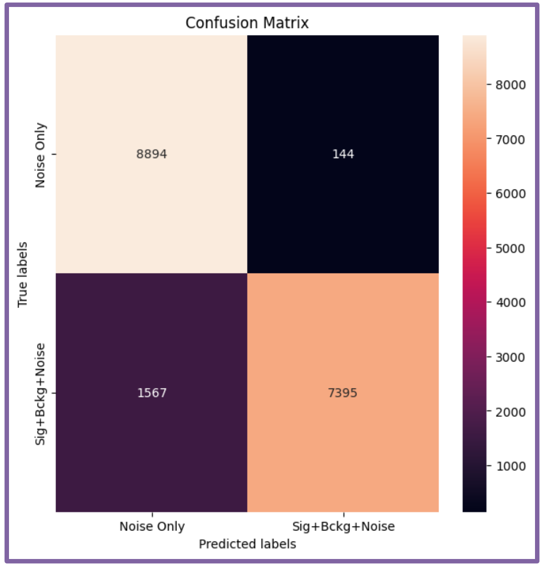
- Inputs, Activations: fixed point<16,6>
- Weights, Biases: fixed point<8,1>



- ❑ Accuracy = (TP+TN)/(TP+TN+FP+FN) = 0.822
- ❑ Purity = TP/(TP+FP) = 0.736
- ❑ Recall = TP/(TP+FN) = 1.000

NN Model performance (300 KHz & 10ns)

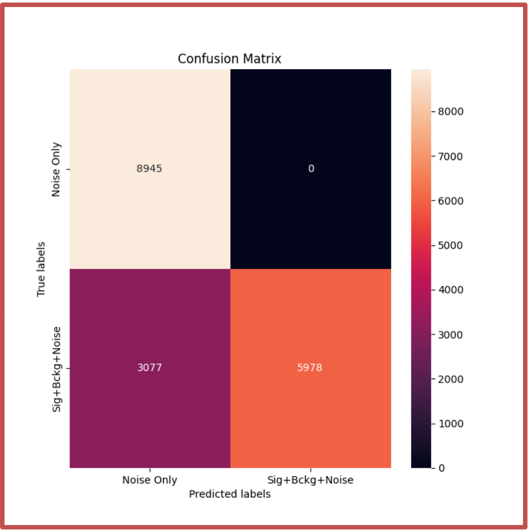
Keras model



- ❑ Accuracy = (TP+TN)/(TP+TN+FP+FN) = 0.905
- ❑ Purity = TP/(TP+FP) = 0.850
- ❑ Recall = TP/(TP+FN) = 0.984

Model Quantization

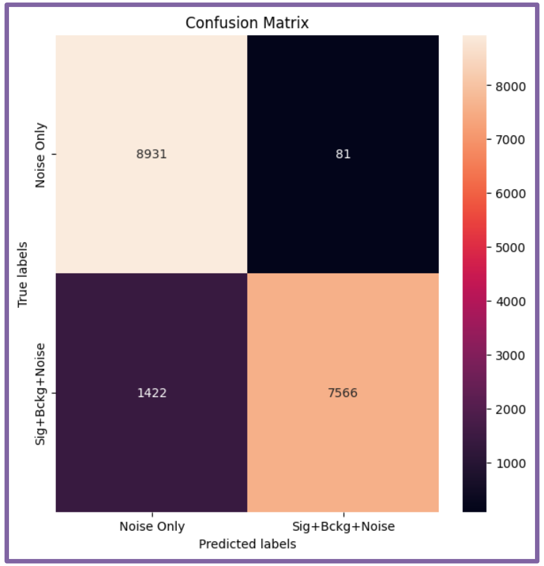
- Inputs, Activations: fixed point<16,6>
- Weights, Biases: fixed point<8,1>



- ❑ Accuracy = (TP+TN)/(TP+TN+FP+FN) = 0.829
- ❑ Purity = TP/(TP+FP) = 0.744
- ❑ Recall = TP/(TP+FN) = 1.000

NN Model performance (150 KHz & 10ns)

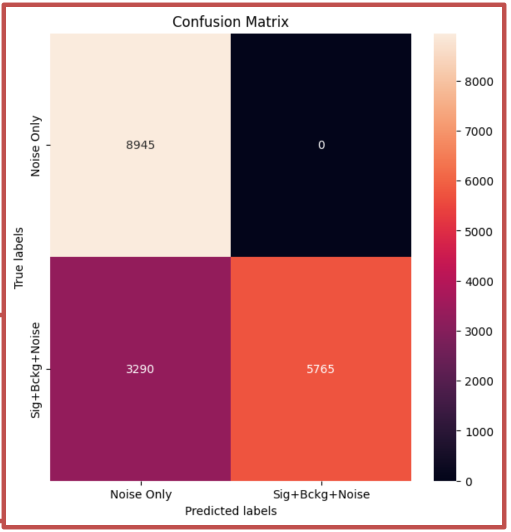
Keras model



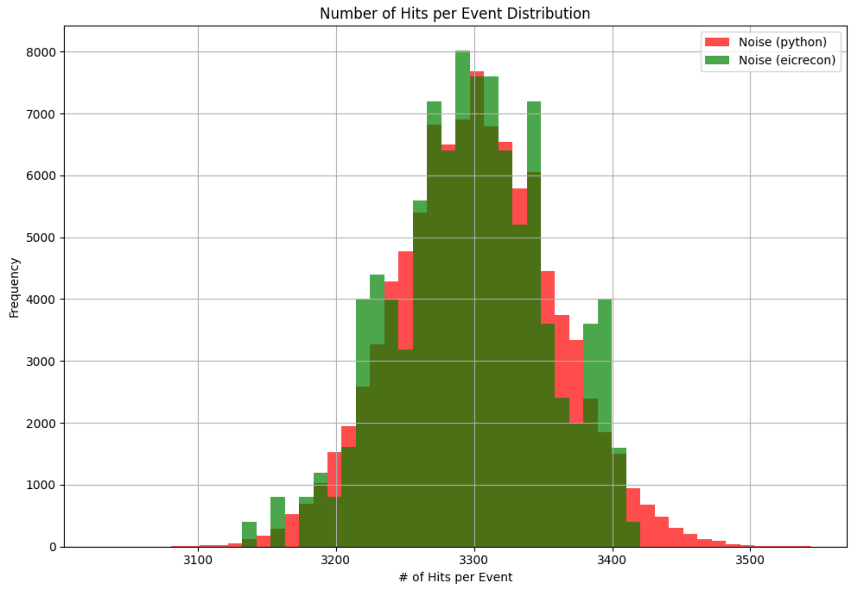
- ❑ Accuracy = (TP+TN)/(TP+TN+FP+FN) = 0.917
- ❑ Purity = TP/(TP+FP) = 0.863
- ❑ Recall = TP/(TP+FN) = 0.991

Model Quantization

- Inputs, Activations: fixed point<16,6>
- Weights, Biases: fixed point<8,1>

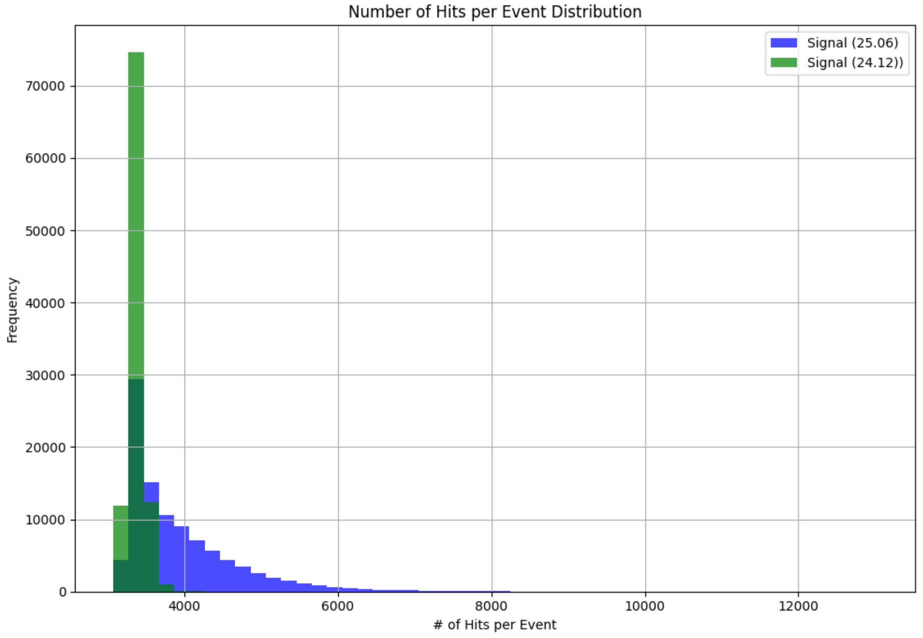


- ❑ Accuracy = (TP+TN)/(TP+TN+FP+FN) = 0.817
- ❑ Purity = TP/(TP+FP) = 0.731
- ❑ Recall = TP/(TP+FN) = 1.000



Noise comparison (@100fb-1)

==> same distro mean



Signal Comparison (@100fb-1)

Eic-shell version=24.12 vs 25.06

==> same starting MC files