Image signal selection for trigger

Igor Pains
with Rafael Nóbrega

Contents

1 **Introduction**
- Motivation
- What was done

2 **Algorithms**
- Image level trigger
- Pixel level trigger

3 **Development**
- Fusion simulation
- Model trainings

4 **Results**
- Image level trigger
- Pixel level trigger

5 **Conclusions**
- Conclusions
- Next steps

1. Introduction

Motivation

- One major **challenge** for the **CYGNO experiment** in the long term will be to **store and analyse all the data produced by the detector.**
  - Each run containing **400 images** from Fusion (Quest) needs **~1.36 Gb (2.01 Gb)** to be stored.
  - A **single day of acquisition** may produce **~215 Gb (327 Gb)** of data.

- The motivation of this work was to study algorithms capable of **distinguishing images or regions** containing a **signal of interest** and **background events.**

- An algorithm capable of doing this task was called **image based trigger algorithm.**

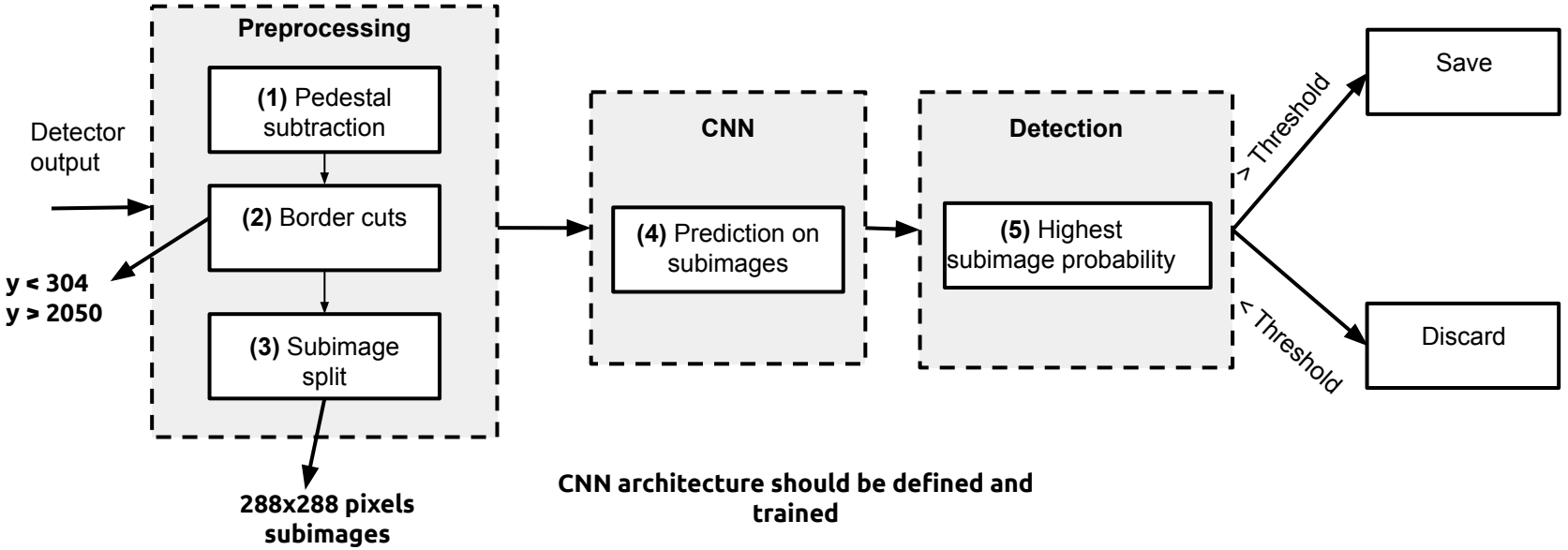
What was done

- Two approaches were proposed:
  - **Image level trigger** using **filtering** and **CNNs.**
  - **Pixel level trigger** using **filtering.**

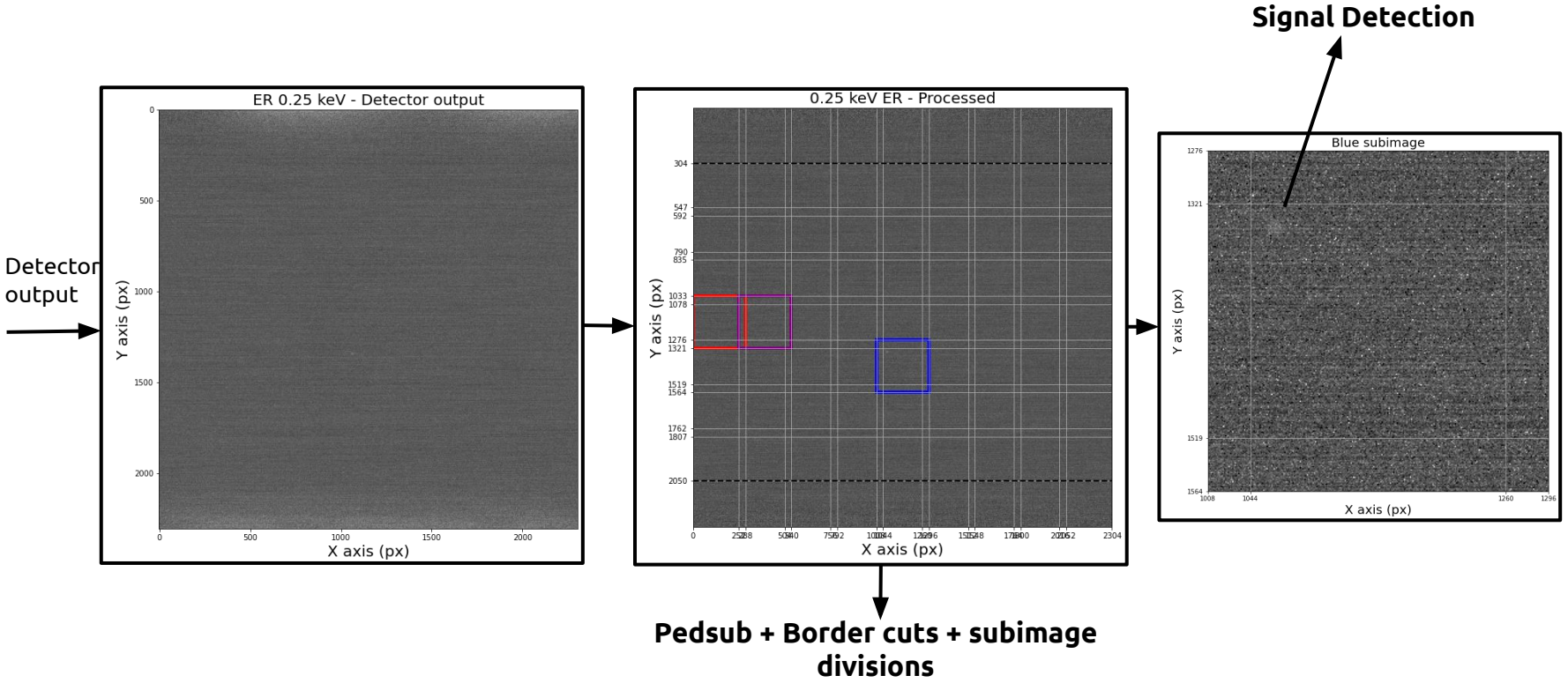
- A performance analysis was done using simulated and real data from Fusion (focused on **low energy** events):
  - Trigger **detection performance**.
  - Reconstruction comparison.
  - **Processing time.**

2. Algorithms

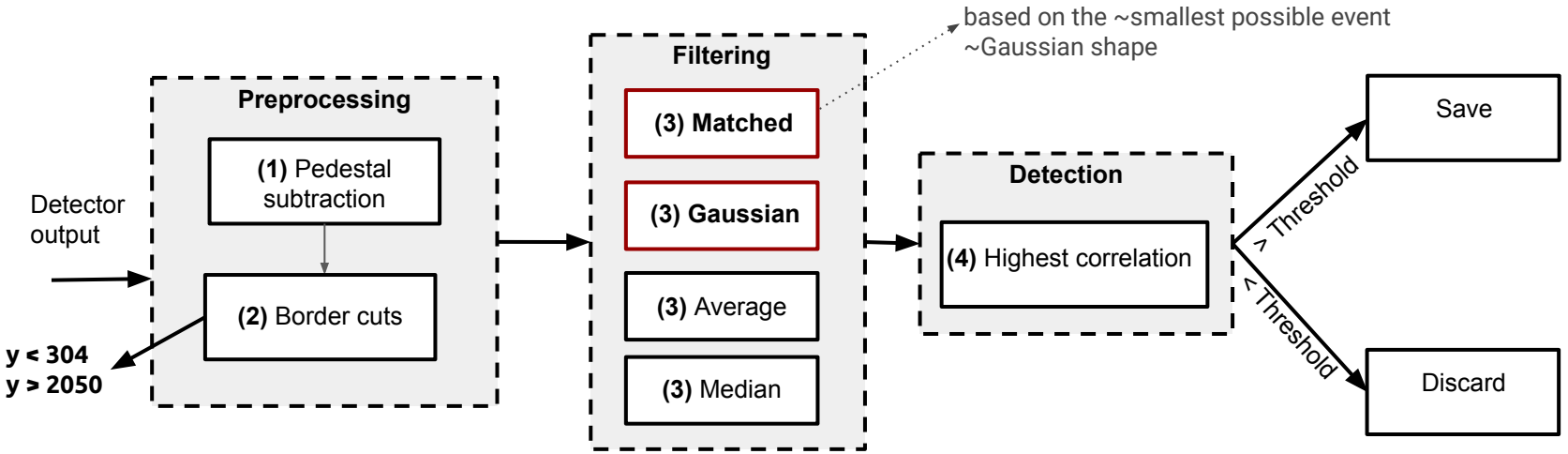
CNN based image level trigger



CNN based image level trigger



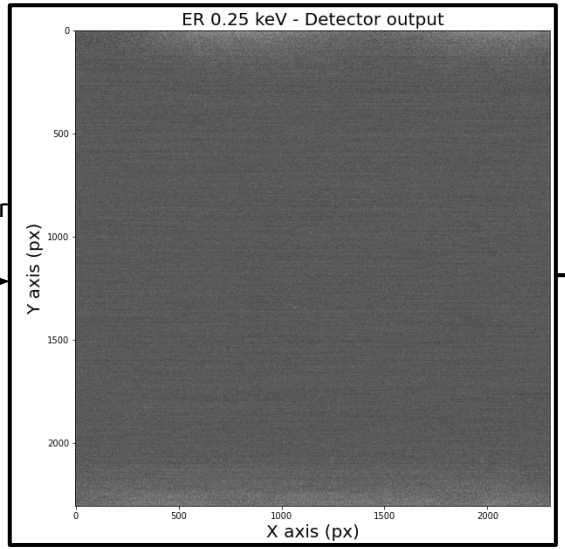
Filtering based image level trigger

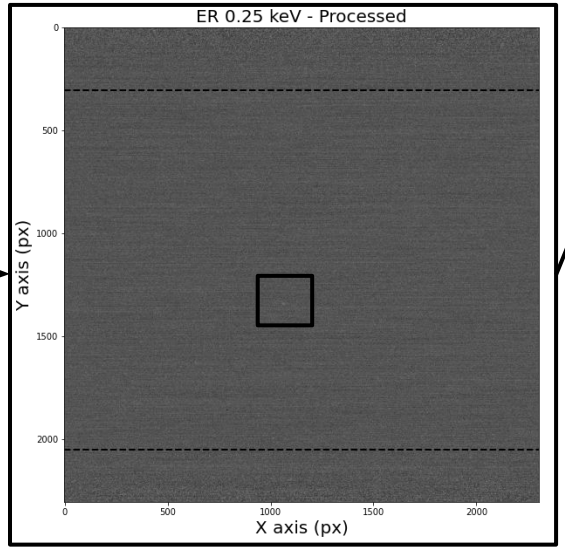


Filter parameters and detection threshold selected based on training data

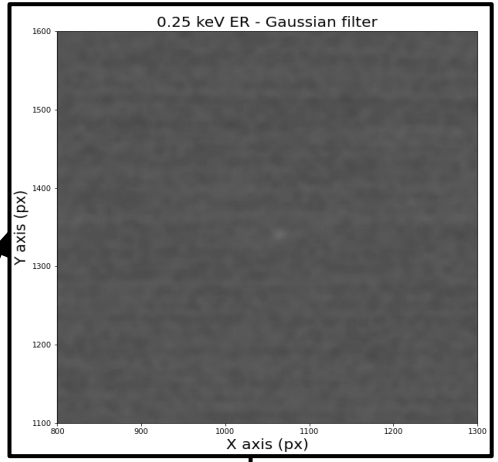
Filtering based image level trigger

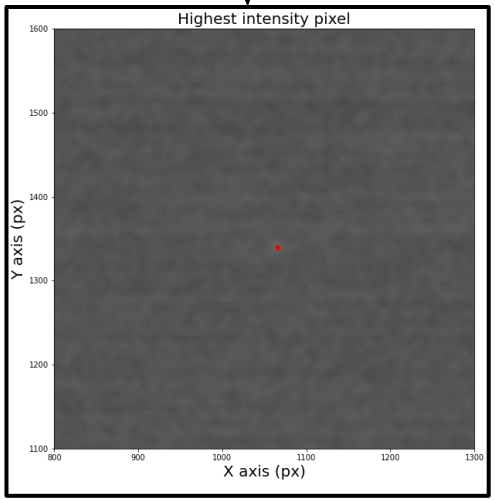
Detector output



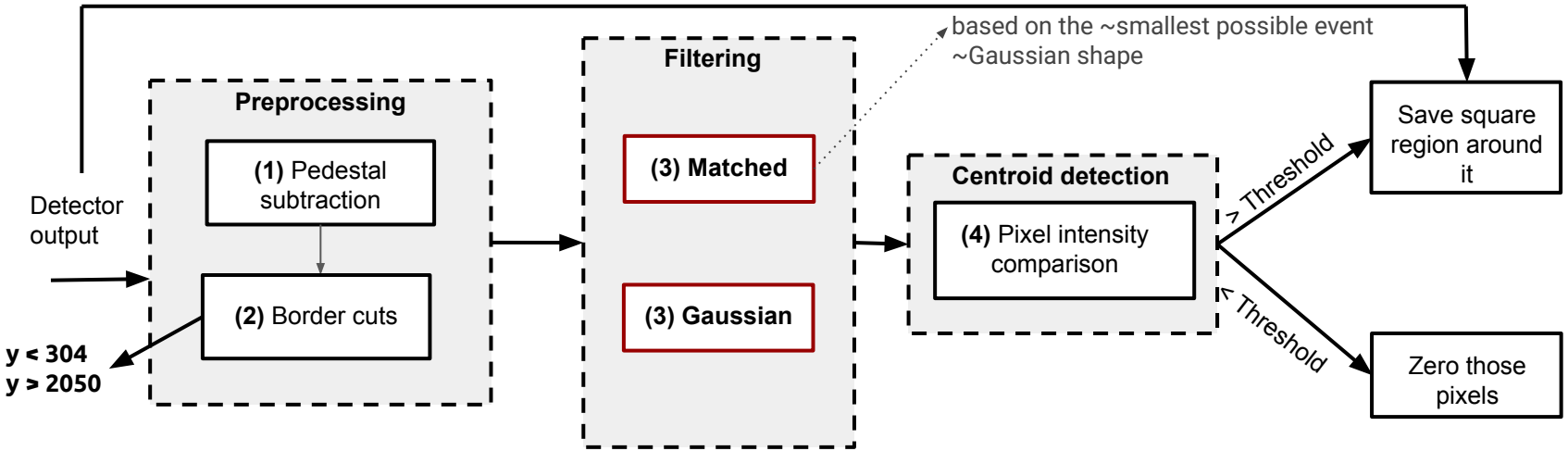


Pedsub + Border cuts





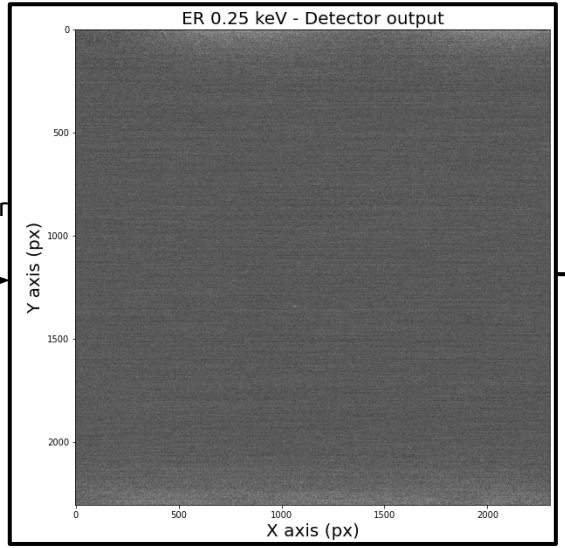
Filtering based pixel level trigger

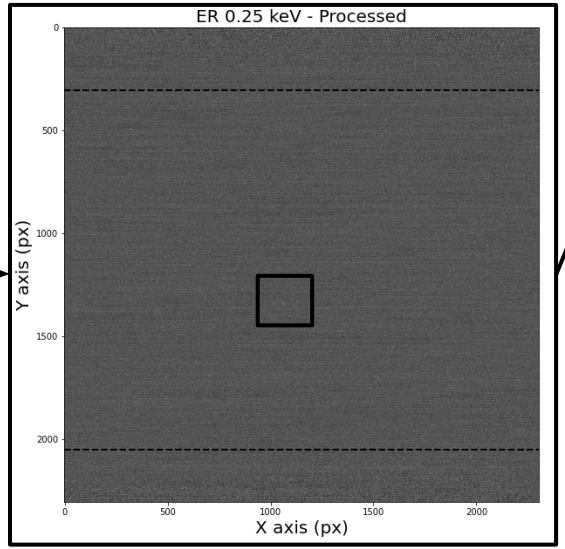


Filter parameters and detection threshold selected based on training data

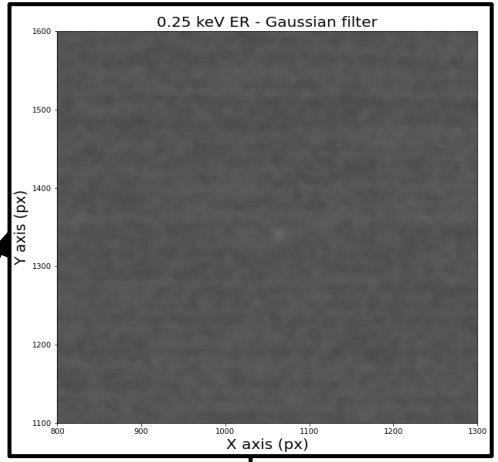
Filtering based pixel level trigger

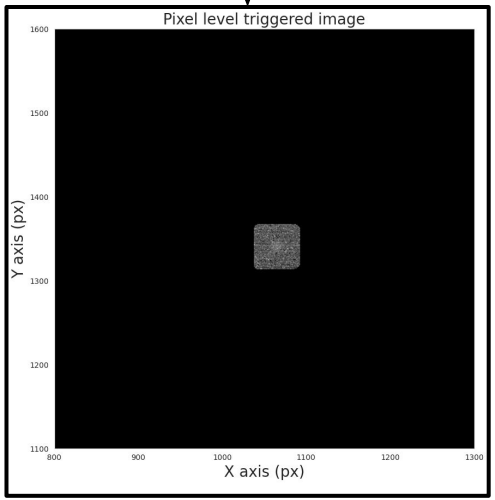
Detector output





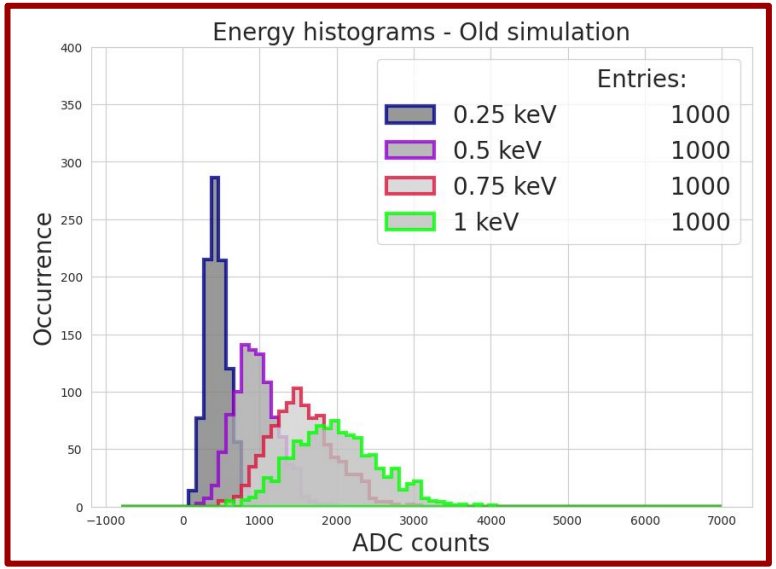
Pedsub + Border cuts

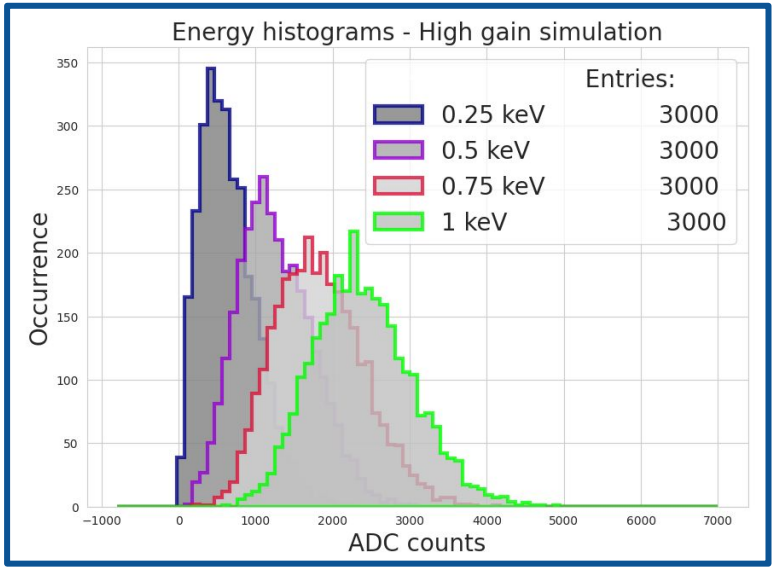




3. Development

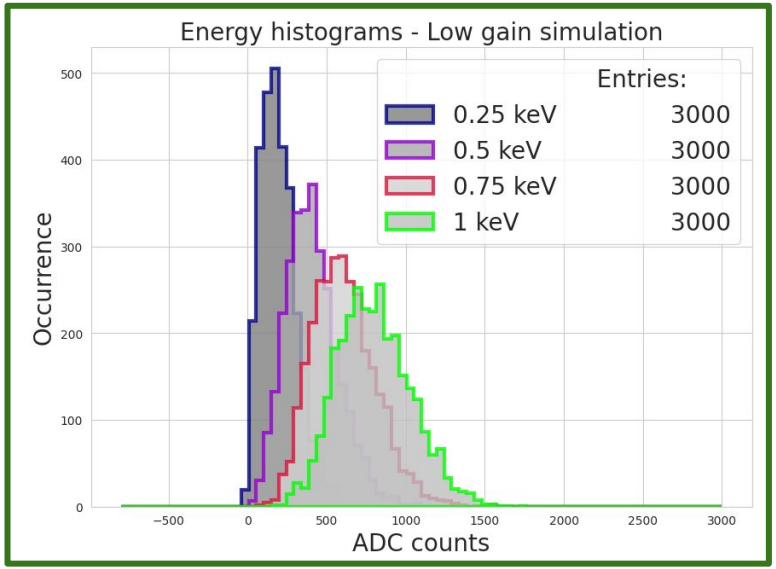
High gain Fusion simulation

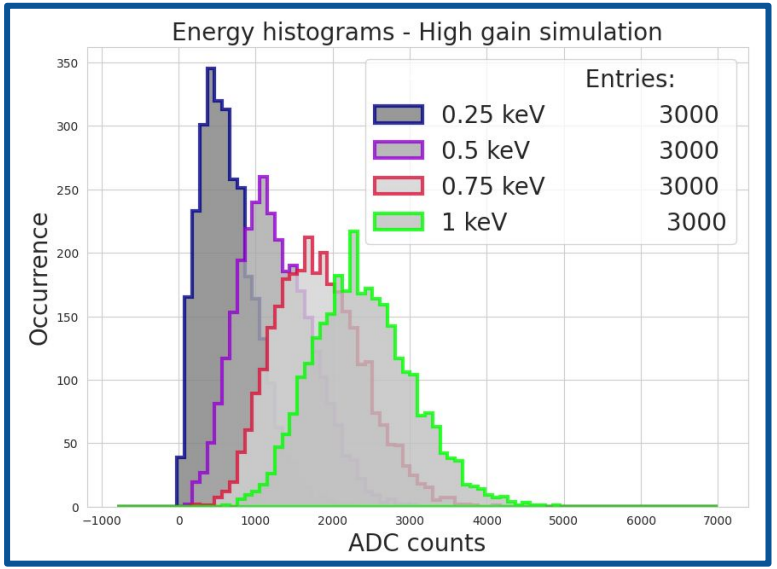




- The new high gain simulation energy in ADC counts is slightly higher and wider.
  - **0.25 keV: [446.17 ± 138.45]** vs **[650.76 ± 367.88]**
  - **0.5 keV: [946.93 ± 278.46]** vs **[1256.11 ± 479.44]**
  - **0.75 keV: [1546.18 ± 418.33]** vs **[1824.22 ± 570.07]**
  - **1 keV: [1992.36 ± 556.59]** vs **[2386.86 ± 639.05]**

Low gain Fusion simulation





- The **low gain** simulation energy in **ADC counts** is **~3 times smaller** than **high gain**.
  - **0.25 keV: [200.89 ± 116.91]** vs **[650.76 ± 367.88]**
  - **0.5 keV: [408.99 ± 166.39]** vs **[1256.11 ± 479.44]**
  - **0.75 keV: [609.78 ± 201.59]** vs **[1824.22 ± 570.07]**
  - **1 keV: [801.70 ± 230.01]** vs **[2386.86 ± 639.05]**

- It is **expected** to be **harder** to **detect** these **signals**.

Models training

- Simulation were split into **training**, **validation** and **testing** datasets (60%-20%-20%).
  - **Pedestal images** were selected from **Run5** to be used as **noise** and to be **combined** with the **simulation.**

- A **scan** using **different filter configurations** (kernel size, sigma, etc) was done using the **training** and **validation datasets.**
  - Each **configuration** generates **two distributions: highest correlation** on **signal** and **noise.**
  - The **configuration** with the **highest AUC** from the **ROC curves** is selected (**better discrimination** between **distributions**).

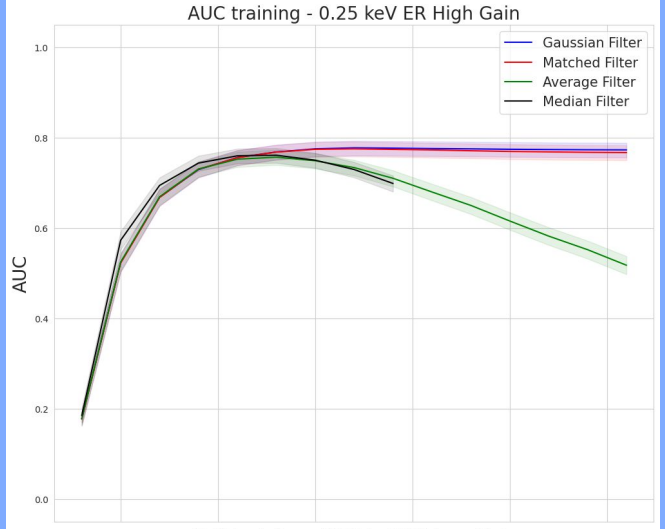
- Several **CNN architectures** were **trained** using the **training** and **validation datasets.**
  - After each **epoch** the **CNN updates** the **weights** of its architecture to **minimize** the **loss** between its **prediction** and **true labels.**

4. Results

Image level trigger

- The **same training procedure** for the **filtering** methods was used with the **new data.**
  - **Gaussian** and **matched filter** still provide the **best results.**
  - **Kernel sizes** of **17 (HG)** and **21 (LG)** were selected.

- The **CNN models trained** with the **old simulation** were used as a **start point.**
  - **New training** will be done for the **Quest** in the **future.**



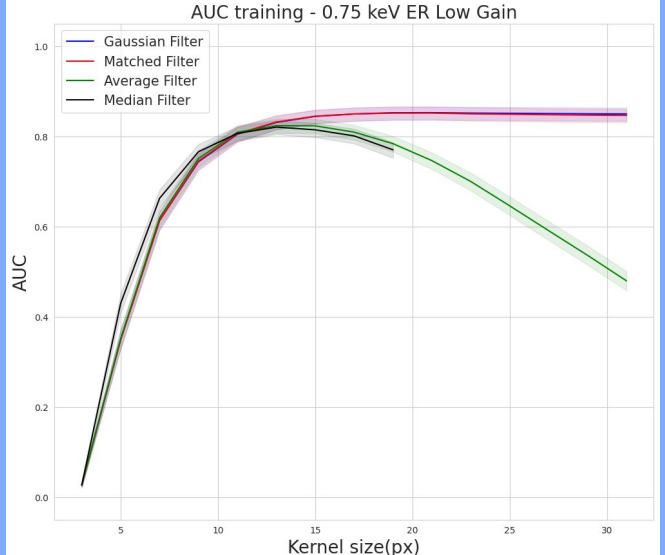
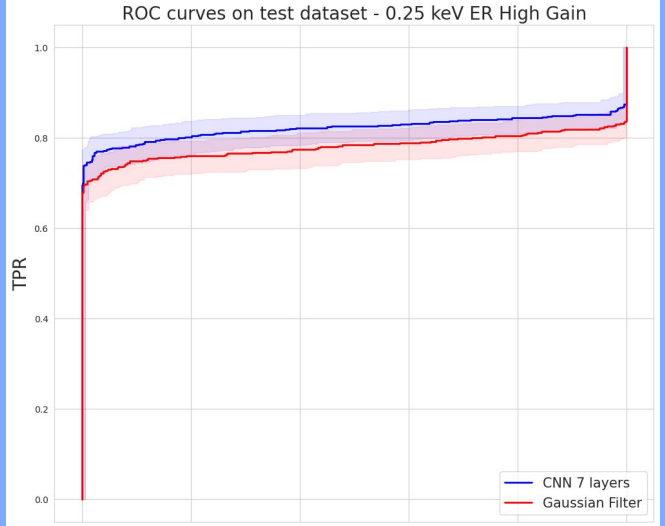
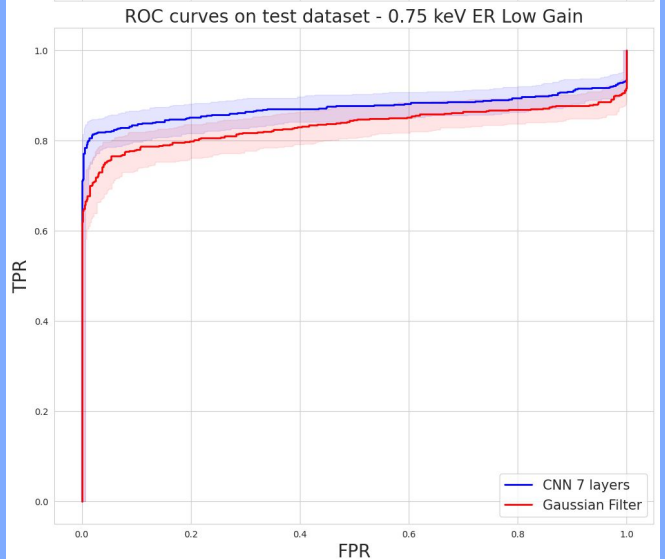


Image level trigger

- The **new high gain simulation** showed **slightly worse results** compared to the **old** one.
  - It has more **low ADC counts events** due to its higher **std**.
  - **Gaussian Filter:** ~75% signal detection with 10 % false alarm.
  - **CNN:** ~74% signal detection with 0.5% false alarm.

- The **low gain 0.75 keV ER** shows **similar** results compared to the **0.25 keV ER** from the old simulation.
  - **Gaussian Filter:** ~78% signal detection with 10% false alarm.
  - **CNN:** ~78% signal detection with 0.5% false alarm.

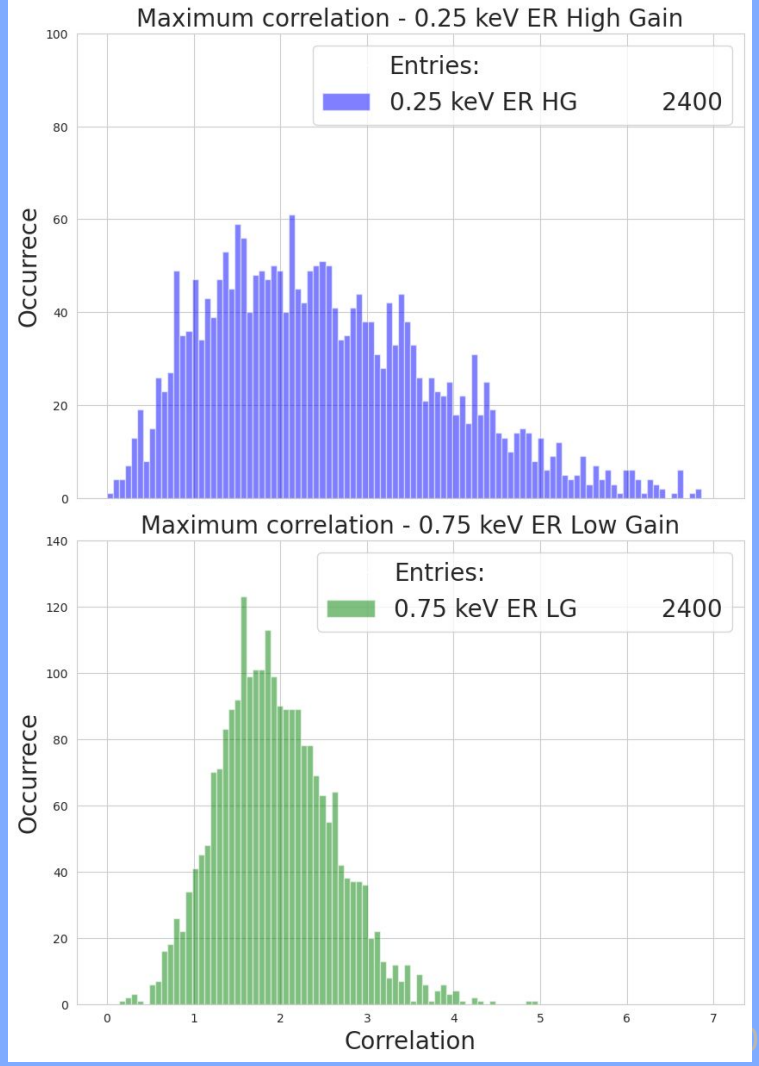




Pixel level trigger

- The **pixel level trigger** needs **two parameters**: **threshold** (centroid detection) and **radius** (save region).
  - The **threshold** should **detect** as much as possible of the **signal** while **discarding noise.**

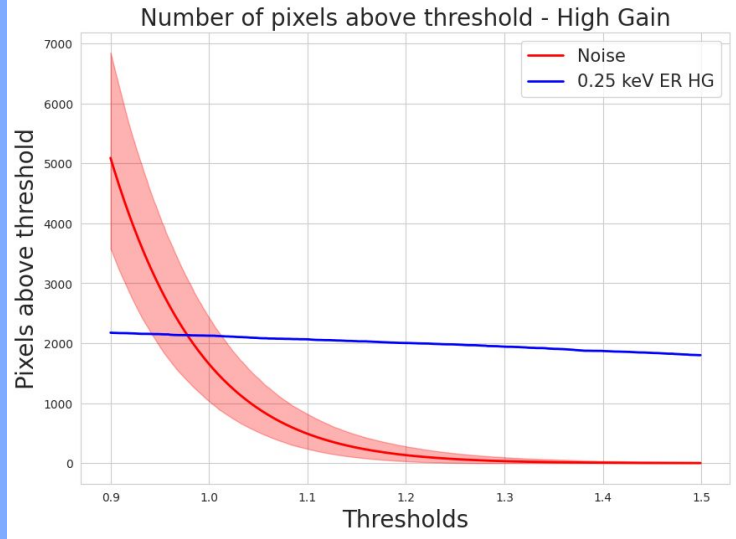
- **Possible ranges** of **thresholds** were selected to see the number of **noise centroids** detected on **noise dataset.**
  - **High gain:** [0.9, 1.5]
  - **Low gain:** [0.8, 1.2]

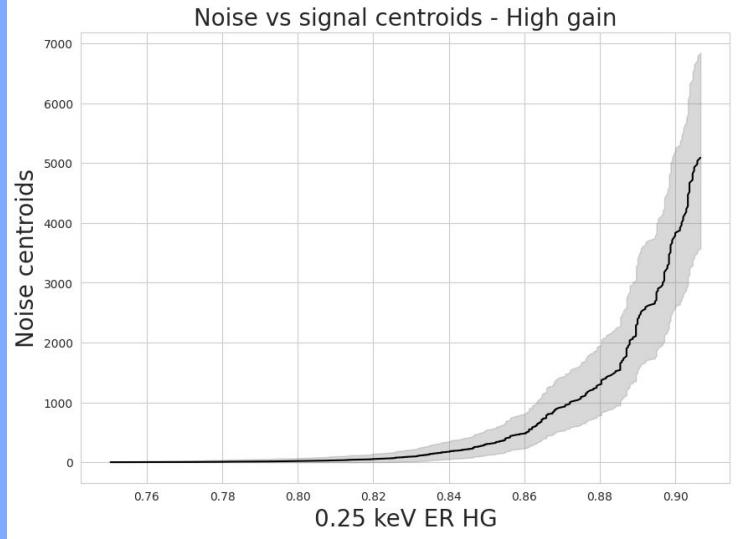


Pixel level trigger

- The **number** of **noise centroids** detected **start high** and **rapidly decreases** as the **threshold increases**.
  - **5000 centroids** could **retain 2 million pixels** in the image with a **radius** of **20** for example.

- The **lower plot** gives a better look at the possible i**deal threshold**: the **knee**.
  - **High gain threshold: 1.123**
  - **Low gain threshold: 0.988**

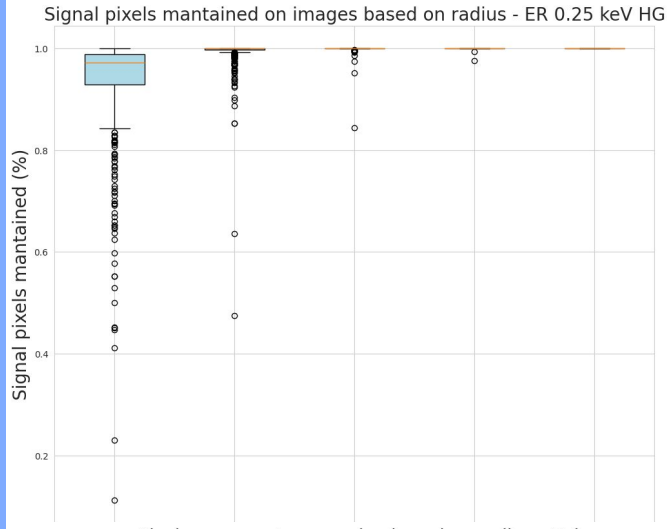


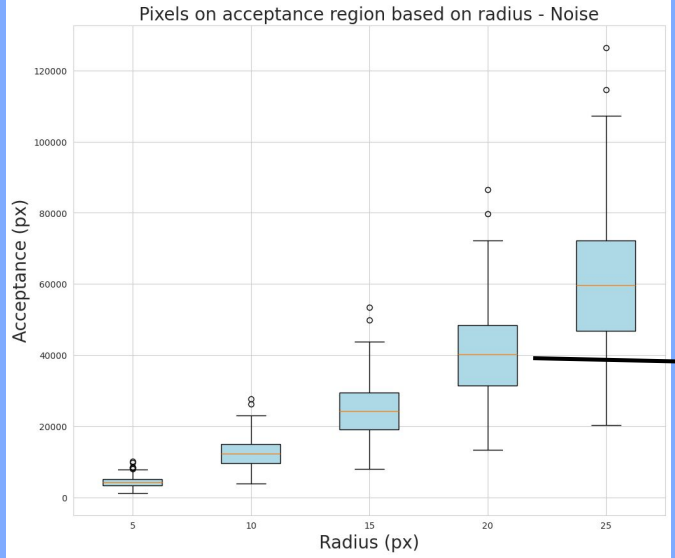


Pixel level trigger HG

- The algorithm was applied on the **HG** data to select the **radius**.
  - A **radius** of **20** is **enough** to **detect almost all** the **pixels** of the **signal detected.**
  - **~86%** of the **0.25 keV ER HG** were **detected.**

- The test dataset was saved after the algorithm was applied.
  - Original full images: 7.1 GB
  - Images as root histograms: 229 MB
  - Images as sparse arrays (scipy): 185.7 MB
  - Images as sparse arrays (root): 188.5 MB



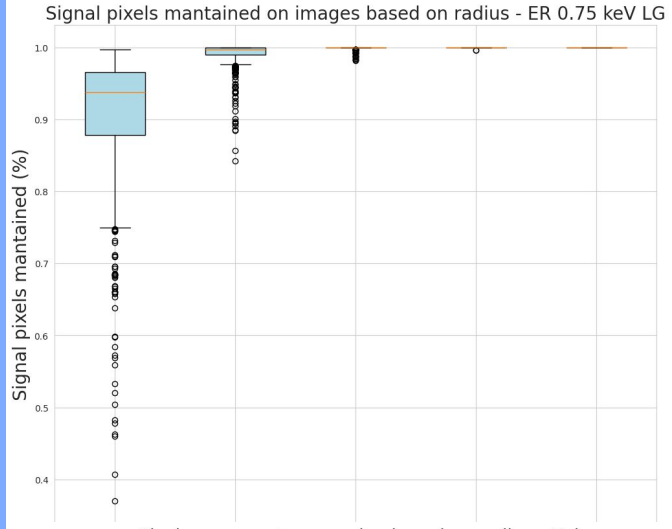


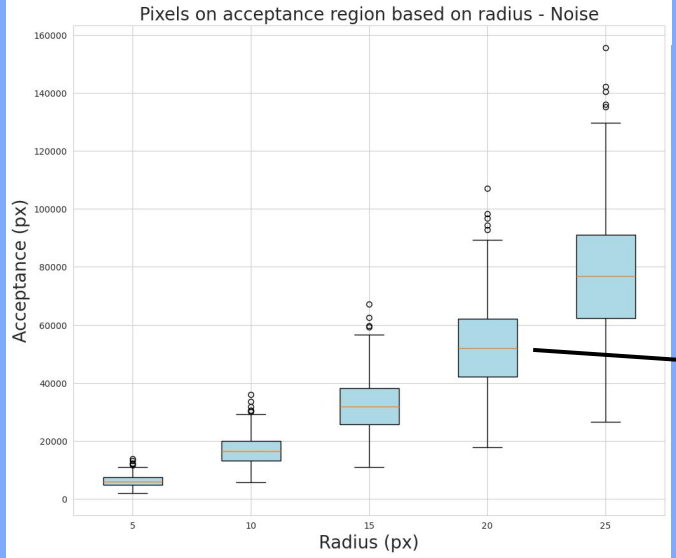
~0.7% of image pixels

Pixel level trigger LG

- The algorithm was applied on the **LG** data to select the **radius**.
  - A **radius** of **20** is **enough** to **detect almost all** the **pixels** of the **signal detected.**
  - **~93%** of the **0.75 keV ER LG** were **detected.**

- The test dataset was saved after the algorithm was applied.
  - Original full images: 7.1 GB
  - Images as root histograms: 284 MB
  - Images as sparse arrays (scipy): 237.6 MB
  - Images as sparse arrays (root): 241.3 MB



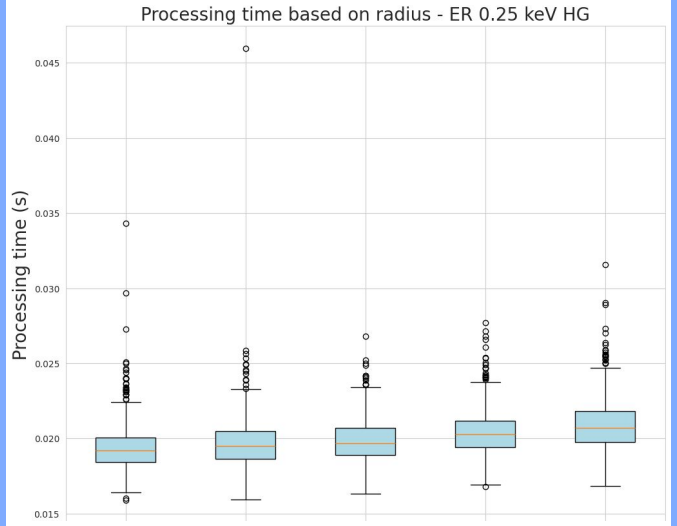


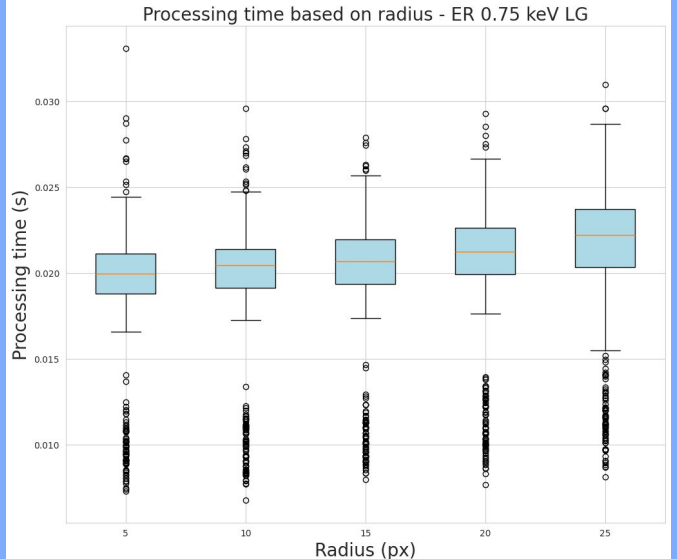
~0.9% of image pixels

Processing time

- The **algorithm** needs **~20 ms (CPU[1])** to **zero all** the **pixels not around centroids.**
  - It highly **depends** on the **number** of **centroids** present on the images.

- Considering that **gaussian filter** needs **~200 ms** and **~20 ms** on **CPU[1]** and **GPU[2]** respectively, the algorithm is still **quite fast.**

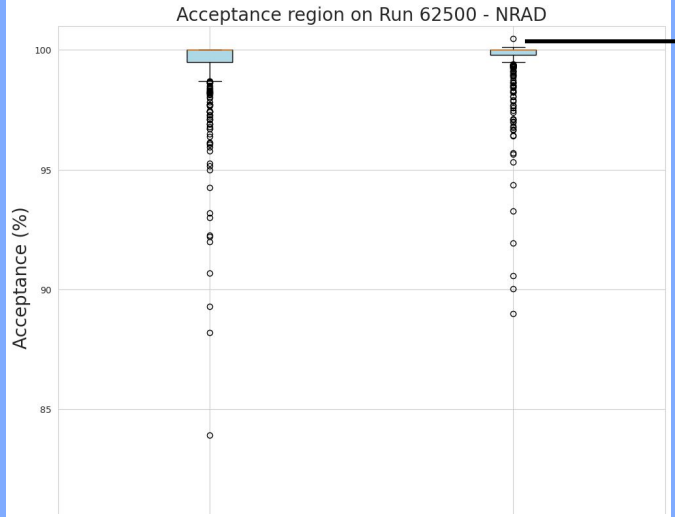


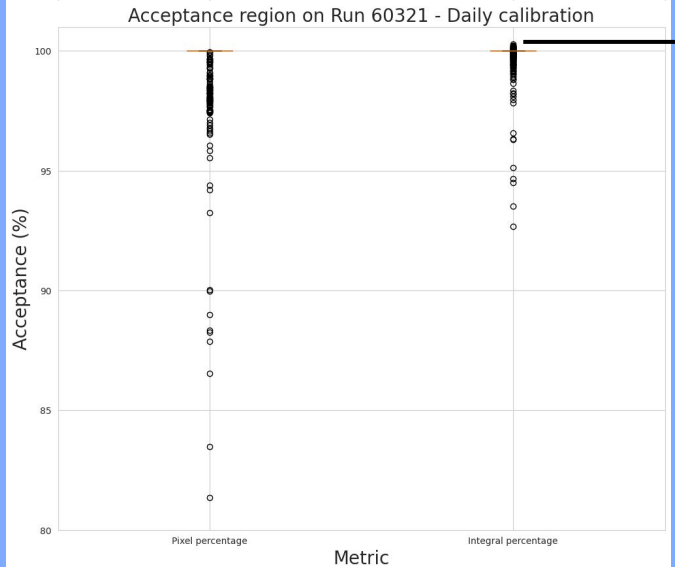


[1]CPU: Notebook01 cloud
[2]GPU: Tesla T4 google collab

Real data results (HG)

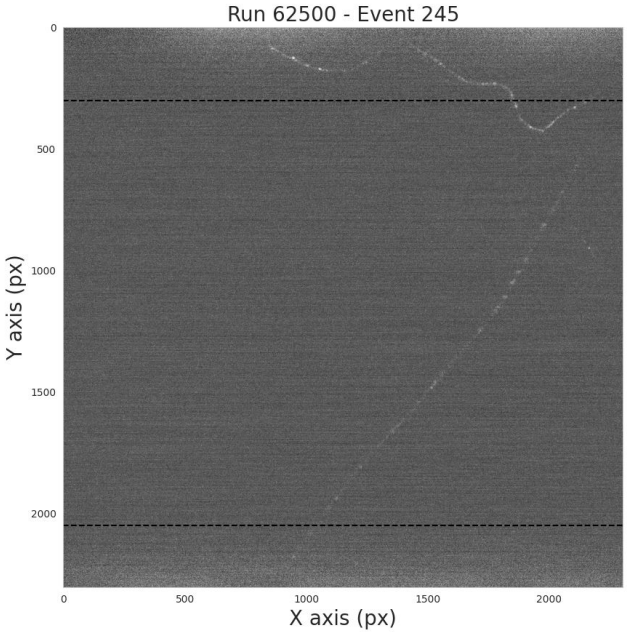
- **Real data** was used to **estimate** the **algorithm performance** (**clusters** with more than **2k ADCs** as **target**)
- **NRAD:** (**99.34±0.08**)% **pixels** and (**99.58 ±0.06**)% **integral** maintained.
- **Daily calibration step2:** (**99.83±0.03**)% **pixels** and (**99.96±0.01**)% **integral** maintained.

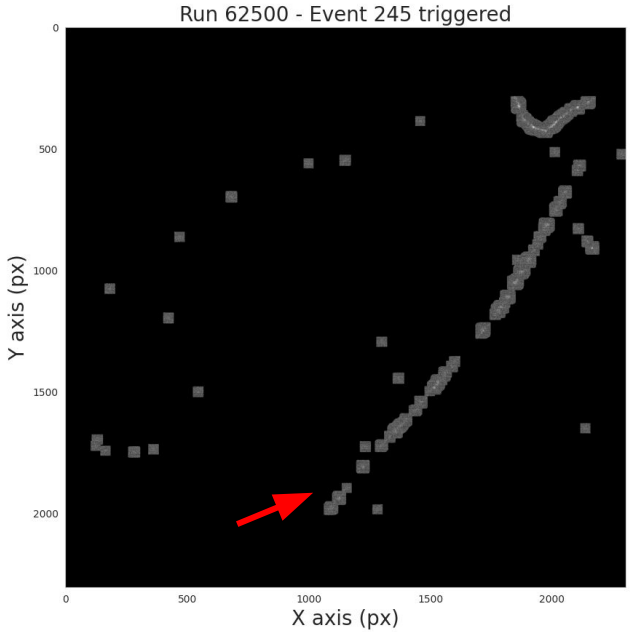


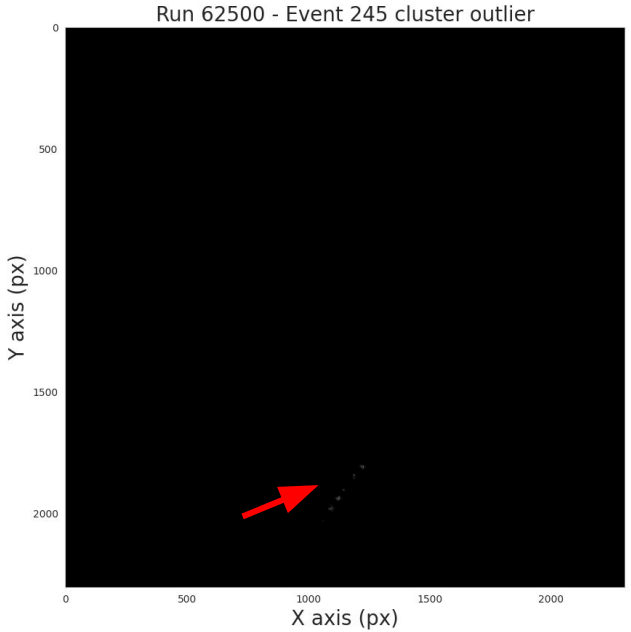


Caused by pedestal subtracted pixels with negative intensity

Worst case scenarios NRAD

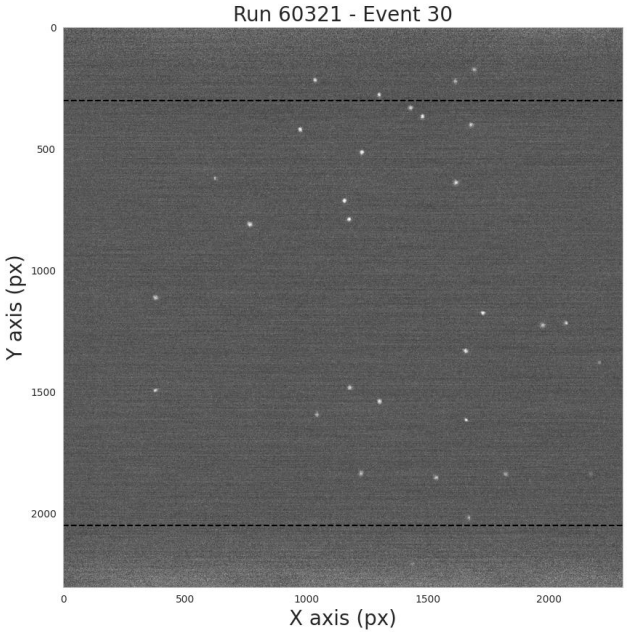


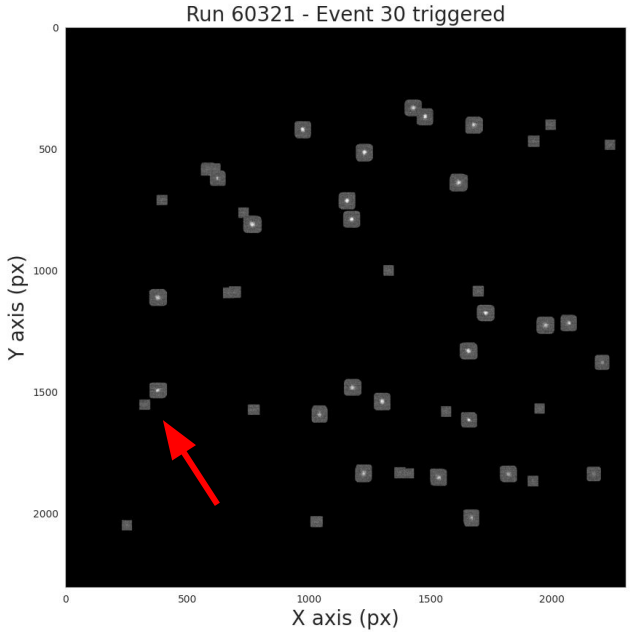


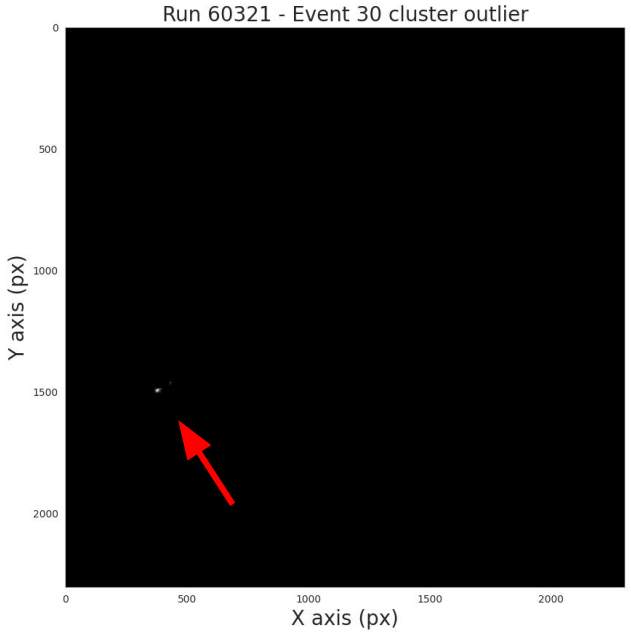


- The **pixel level trigger** maintained **83.92%** and **88.99%** of the **pixels** and **integral** of this **cluster** respectively.

Worst case scenarios Fe55







- The **pixel level trigger** maintained **78.58%** and **94.66%** of the **pixels** and **integral** of this **cluster** respectively.

5. Conclusions

Conclusions

- The analysis using the **new simulation** showed **consistent results** and **proves** that the methods are **resilient**.

- The **pixel level trigger** has a **good potential** to **efficiently save memory** on the **data acquisition**.
  - **Memory reduction factor** of **~40x** on **low energy** simulation.

- It may **detect ~86%** of **0.25 keV (HG)** and **~93%** of **0.75 keV (LG) ER** simulated events.
  - Simulated **ER (HG)** with energies above **3 keV** show **100% efficiency.**
  - **Real data** with more than **2k ADCs counts** are **always detected** maintaining more than **99%** of the **pixels** and **integral**.

Next steps

- The **focus** from now on will be on **Quest** data.
  - Recently started to **optimize digitization parameters** for this **camera.**

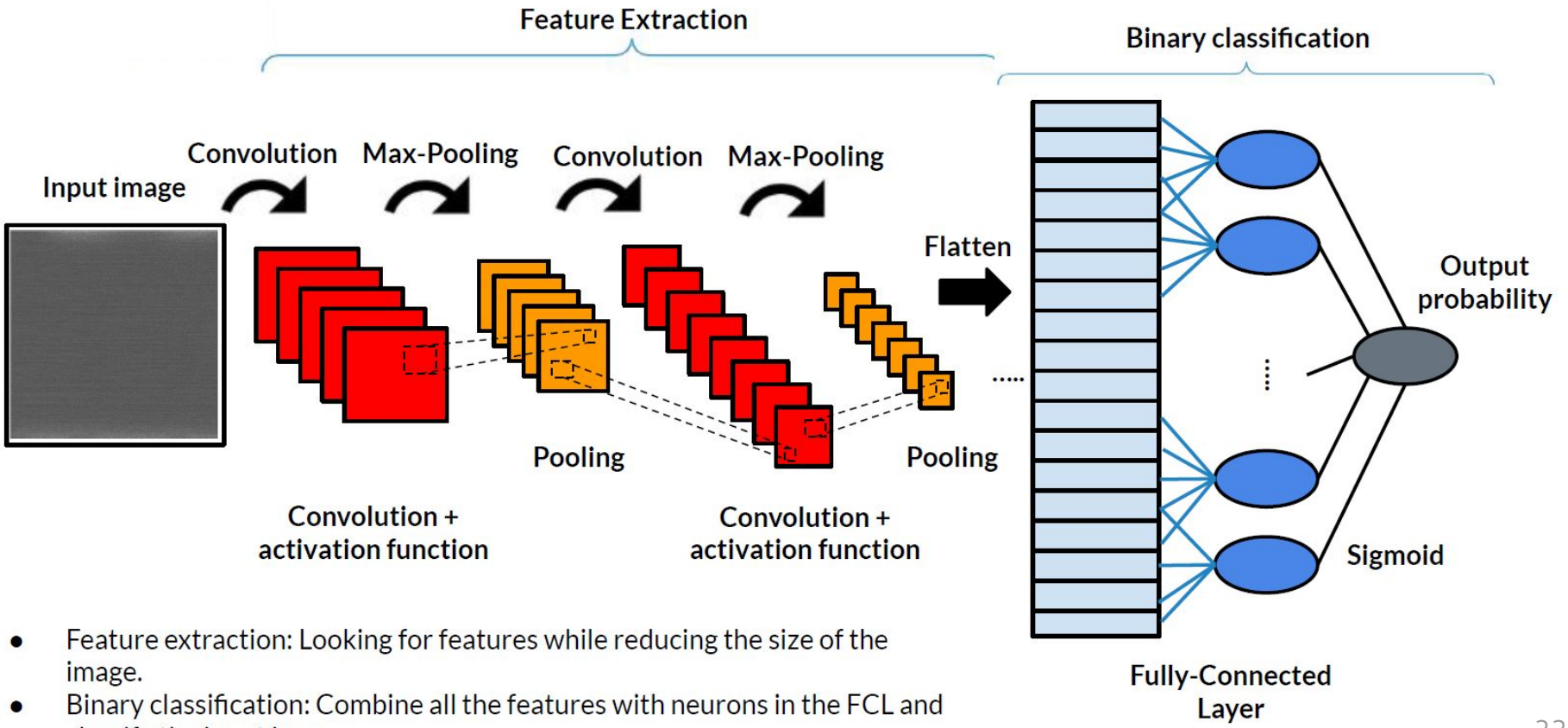
- **Retrain** the **models** with the **new data.**

- **Test** the **algorithms** on a **data taking** next year.

Thank you

Backup

CNN architecture



- Feature extraction: Looking for features while reducing the size of the image.
- Binary classification: Combine all the features with neurons in the FCL and classify the input image.

CNN architecture

- The input shape of the CNN limits the number of convolutional and max-pooling layers that can be used.
  - An image with 288 ($2^5.3^2$) pixels may use up to **7 (5+2)** layers with regular max-pooling.
  - Custom max-pooling layers may be used to increase the number of layers up to 9.

- Four CNN architectures were selected (number of layers from 6 to 9).
  - The bayesian optimization was used during training.
  - The approach is to select a range of possible hyperparameters (number of filters in each conv layer, neurons on dense layer, etc) and the method will find the optimal values.

CNN training