Trigger proposal

Igor Pains
with Rafael Nóbrega

18/07/2024
Analysis & reconstruction meeting

1.
Introduction

Introduction

- Motivation: reduce data to manageable levels by selecting only events of interest, saving storage and processing resources.
  - Each run may need up to 2 Gb to be stored after the compression.
    - ~1 Tb per day considering the current frequency.

Proposal

- Develop algorithms to be tested as online trigger to decide whether to save or not images taken by the detector.
  - Convolution of the image with several kernels: look for high correlation points. Link of the last presentation
  - *on going →* Explore Machine Learning methods (CNN)

2.

Dataset

Simulation

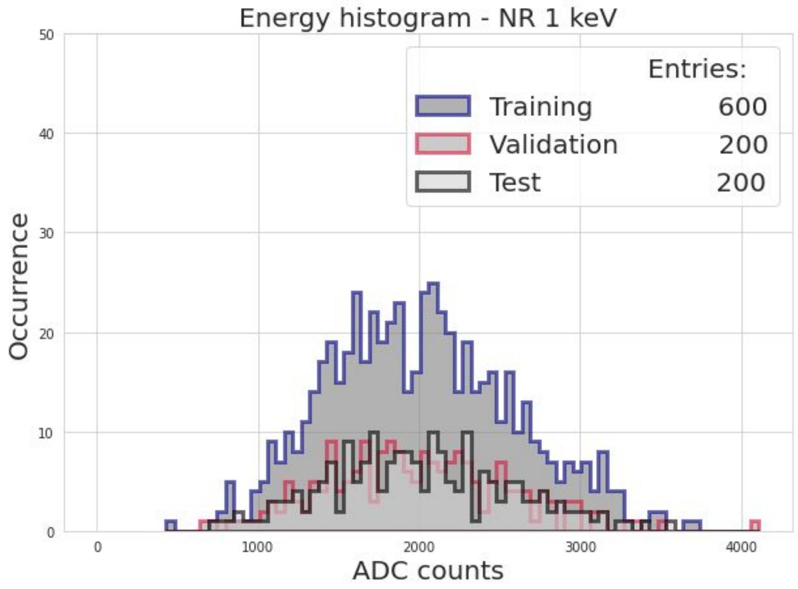
- We started using Pietro’s simulation, which contains:
  - ER events with 1, 3, 6, 10, 30 and 60 keV (1k each)
  - NR events with 1, 3, 6, 10, 30 and 60 keV (1k each)

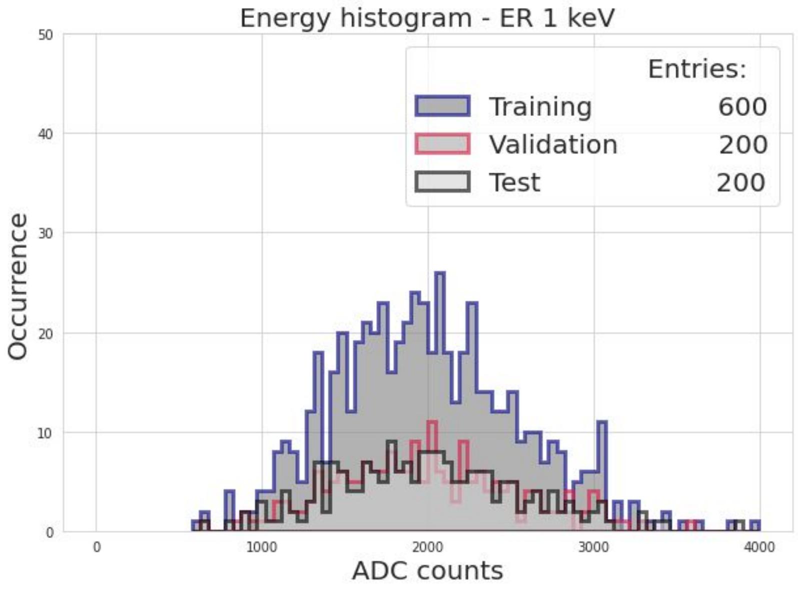
- The 1 keV simulation was used to create smaller energies simulations (0.25 and 0.5 keV).

Datasets

- **Datasets**
  - **Training:**
    - Noise dataset: 600 images from pedestal runs (Run 4 underground).
    - ER and NR signal simulation: 600 images each containing 0.25-1 keV signals added to pedestal runs (different from noise dataset).
  - **Validation:**
    - Noise dataset: 200 images from pedestal runs.
    - ER and NR signal simulation: 200 images each containing 0.25-1 keV signals.
  - **Test:**
    - Same configuration as validation.

Signal split





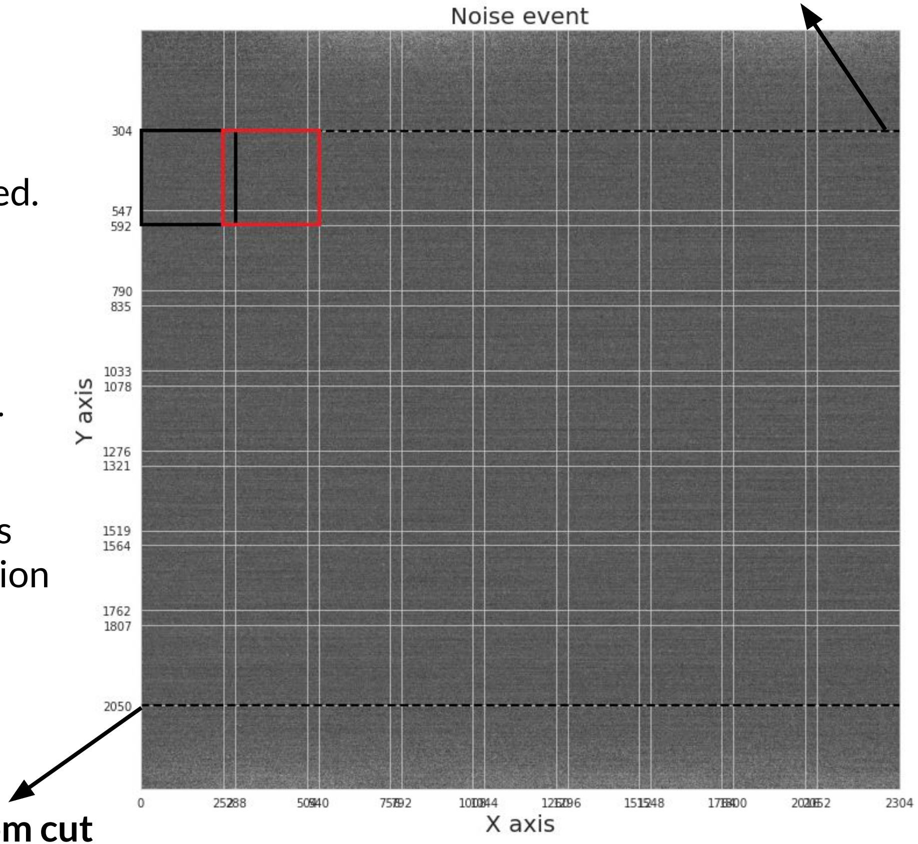
- The signal split was done in a way to maintain the three distributions as similar as possible.

Noise

- A 2304x2304 image is too big for a CNN to handle, meaning that a size reduction is needed.

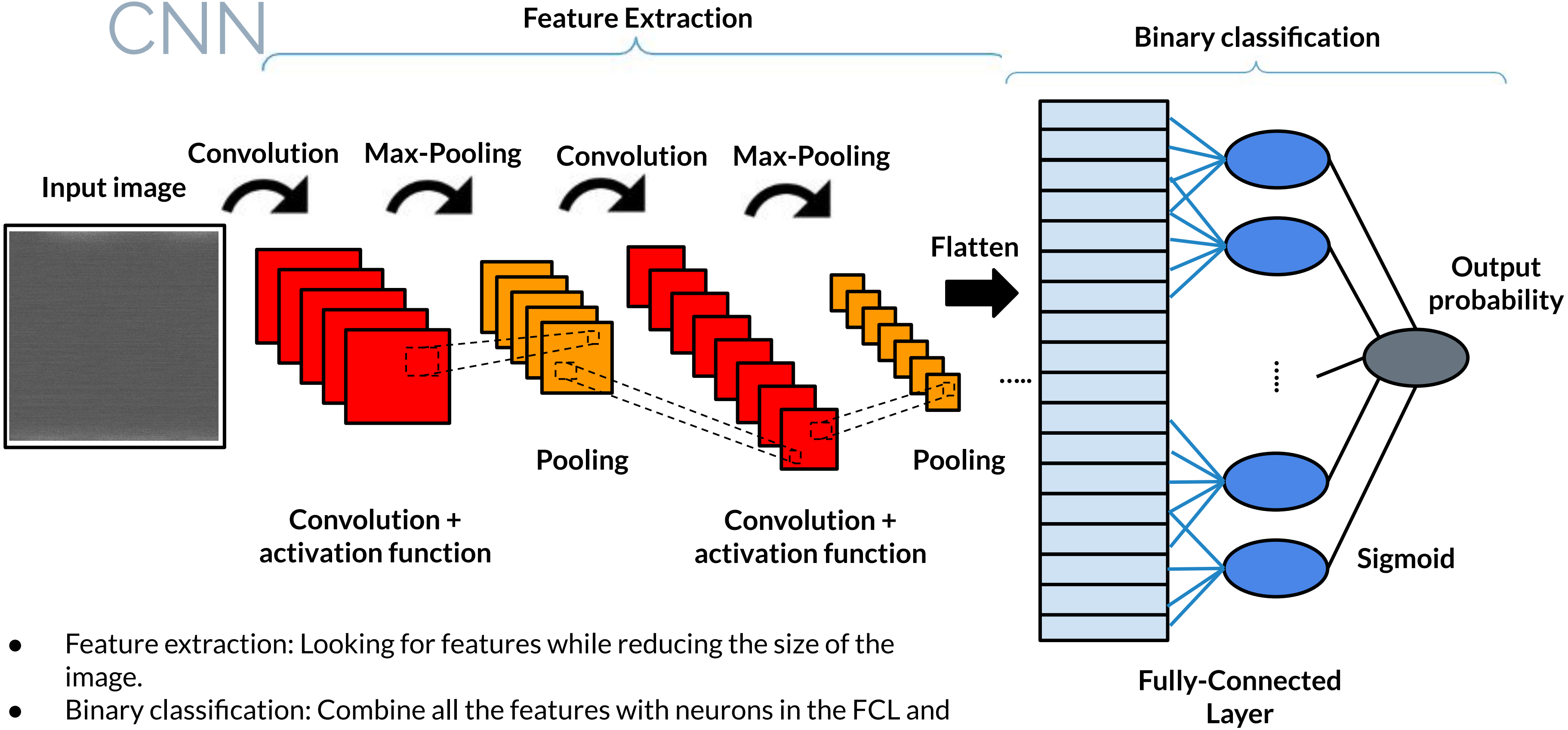
- A possible approach is to send fractionated patches from the original image to the CNN.
    - Tensorflow extract patches function was used.

- The right images were divided into 63 patches with 288x288 pixels each with an overlap region between them.



3.
CNN

CNN



- 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

- After some tests, the best CNN had the following architecture:
  - Feature extraction: Six combinations of convolutional layers (3x3 kernels) with ReLu activation, MaxPooling layers (3x3 kernels), dropout layers.
  - Binary classification: Fully connected layer with 512 neurons with ReLu activation, dropout layer, batch normalization and output layer with sigmoid activation.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 288, 288, 32)      320
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 144, 144, 32)      0
_________________________________________________________________
dropout_14 (Dropout)         (None, 144, 144, 32)      0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 144, 144, 32)      9248
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 72, 72, 32)        0
_________________________________________________________________
dropout_15 (Dropout)         (None, 72, 72, 32)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 72, 72, 32)        9248
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 36, 36, 32)        0
_________________________________________________________________
dropout_16 (Dropout)         (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 36, 36, 32)        9248
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 18, 18, 32)        0
_________________________________________________________________
dropout_17 (Dropout)         (None, 18, 18, 32)        0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 18, 18, 32)        9248
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 6, 6, 32)          0
_________________________________________________________________
dropout_18 (Dropout)         (None, 6, 6, 32)          0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 6, 6, 128)         36992
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 2, 2, 128)         0
_________________________________________________________________
dropout_19 (Dropout)         (None, 2, 2, 128)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_4 (Dense)              (None, 512)               262656
_________________________________________________________________
batch_normalization_2 (Batch (None, 512)               2048
_________________________________________________________________
dropout_20 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 513
=================================================================
Total params: 339,521
Trainable params: 338,497
Non-trainable params: 1,024
_________________________________________________________________
```

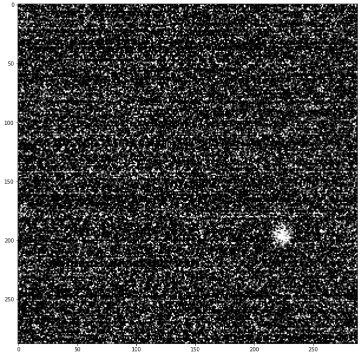
CNN training

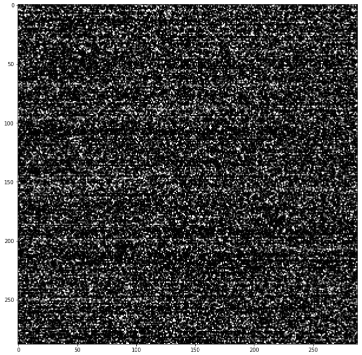
- Both ER and NR were used together during the CNN training.
  - The signal was randomly rotated and placed in a position among the noise.

- 4800 images with 288x288 pixels were used on CNN training and 1600 on validation.
  - Every signal from the split was used twice.
  - The noise patch used was always different.

- The best result was achieved by using 0.5 keV signals on training.
  - 0.25 keVs signals generally led to overfitting.

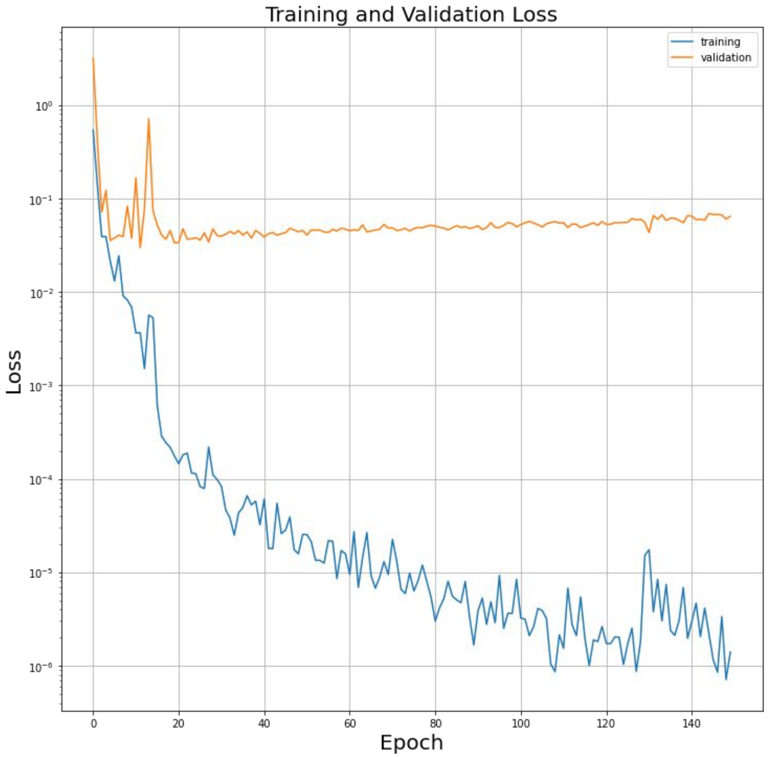
Signal image

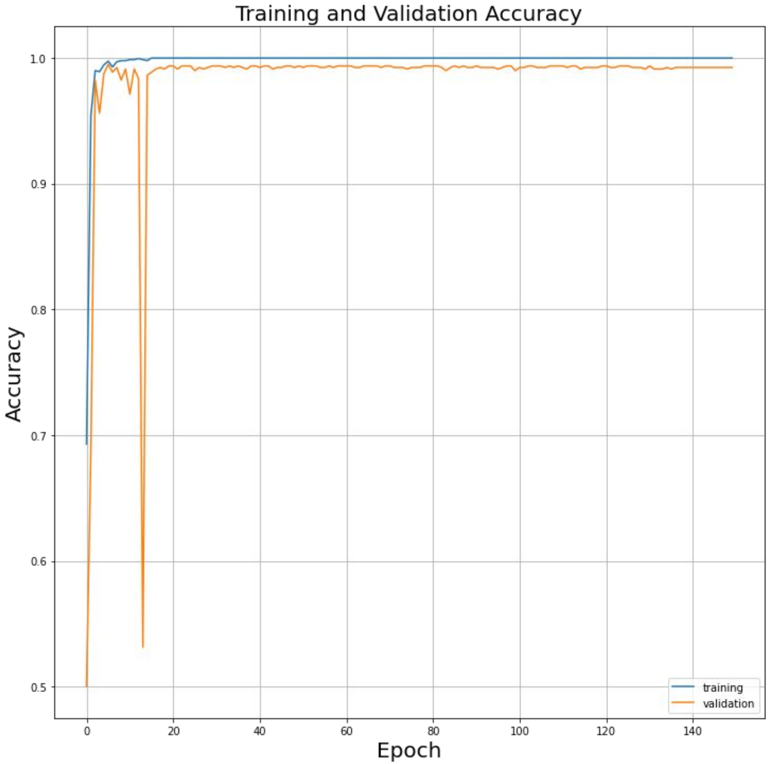




Noise image

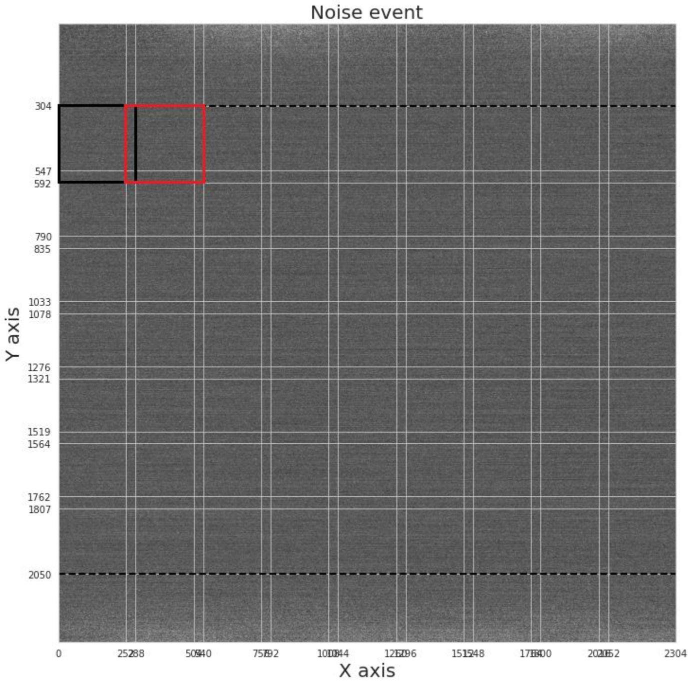
CNN training



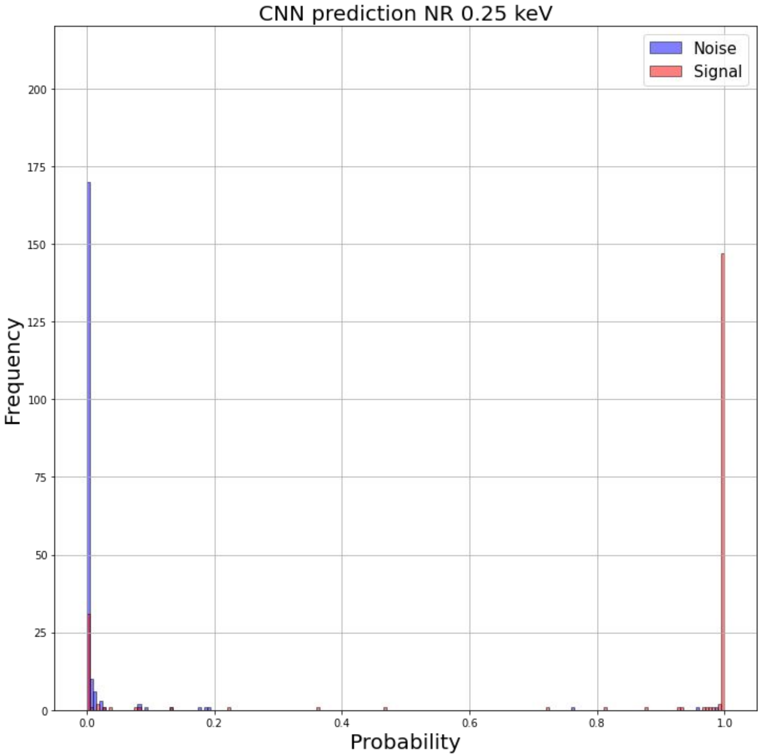


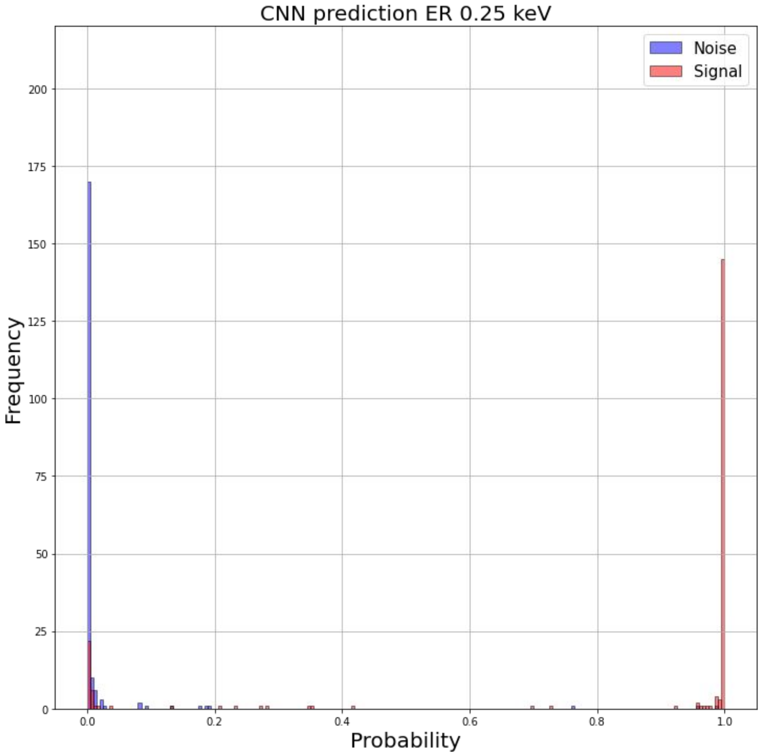
CNN test

- Since the actual image has 2304x2304 pixels, the test should be performed in way to use all that information.
  - The highest prediction of the CNN on each one of the 63 patches from noise images is stored.
  - The highest prediction on the CNN on each one of the patches that contain an information of the signal is stored.
- This procedure was used on the 400 images separated for test.
  - ER and NR were tested separately to see the CNN performance.
  - 0.25 and 0.5 keV signals were used for test.

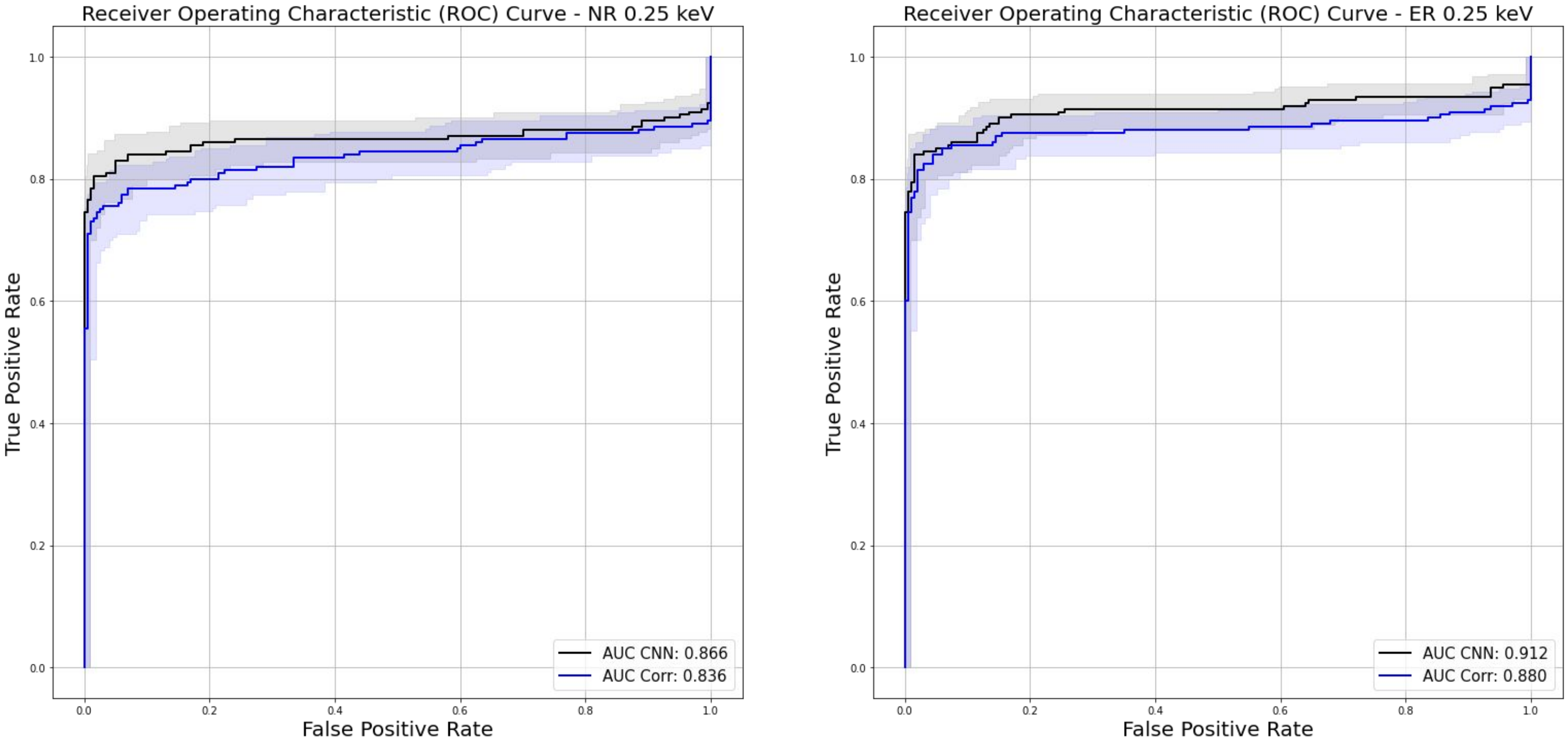


CNN predictions 0.25 keV

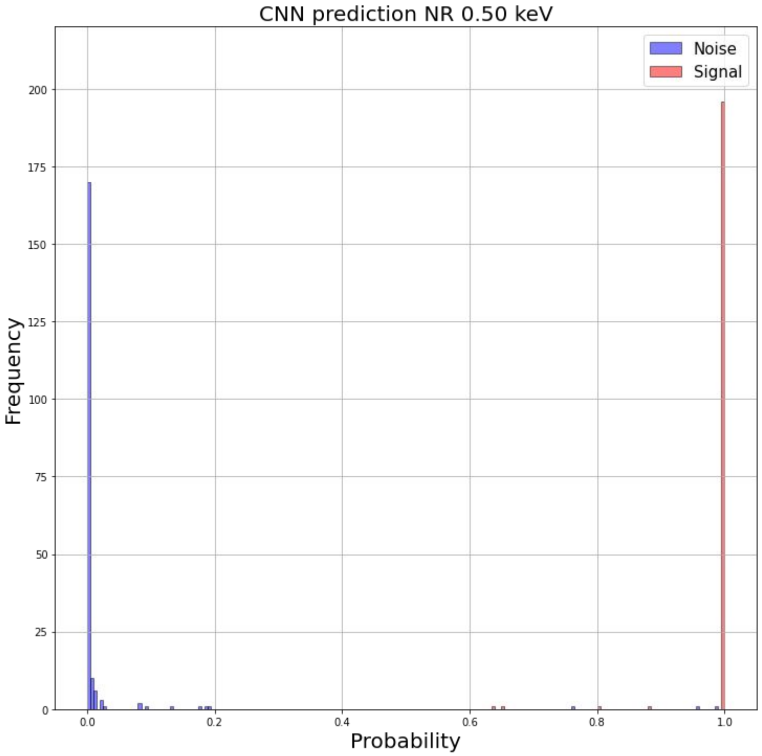


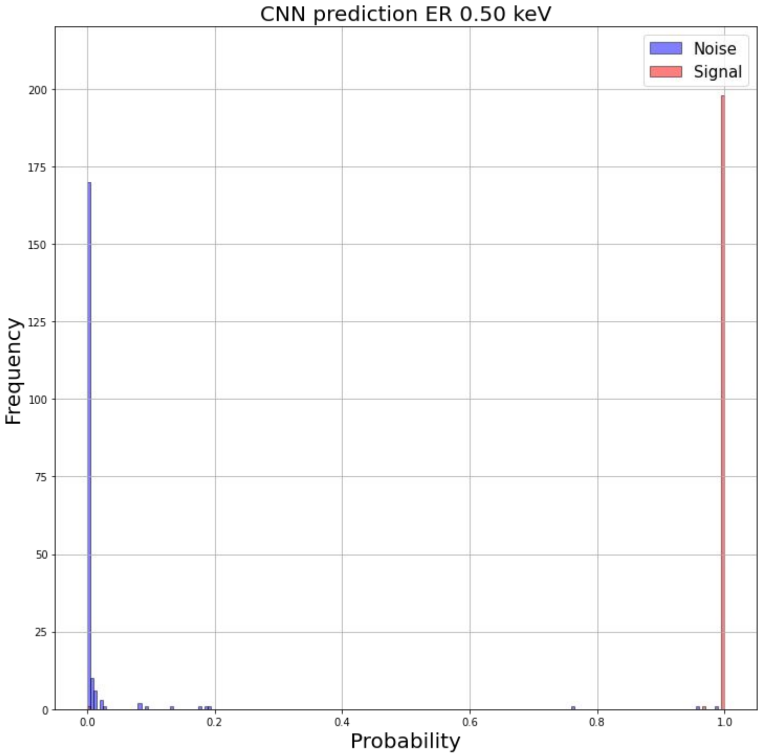


CNN ROC 0.25 keV

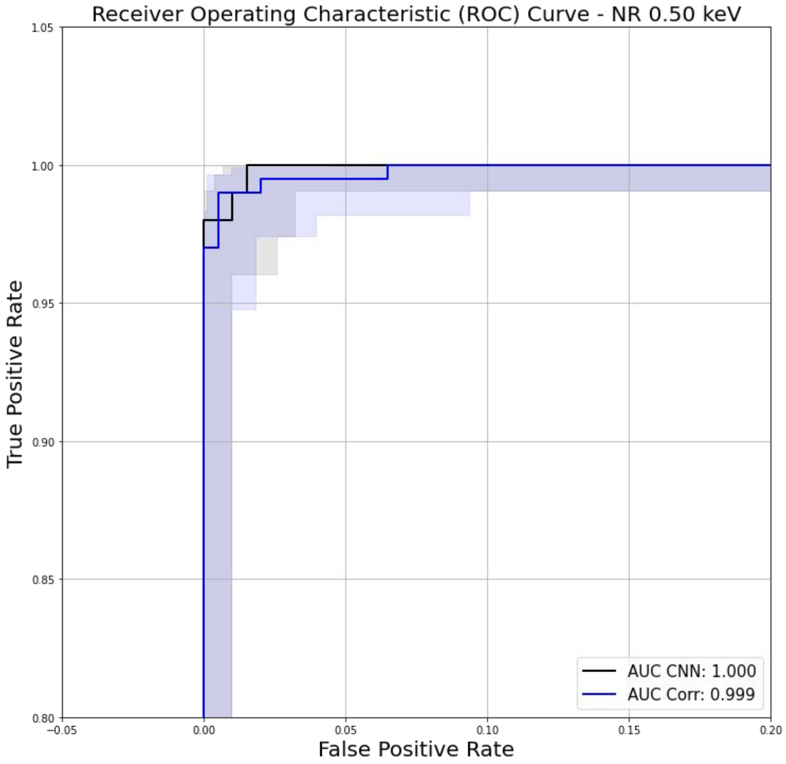


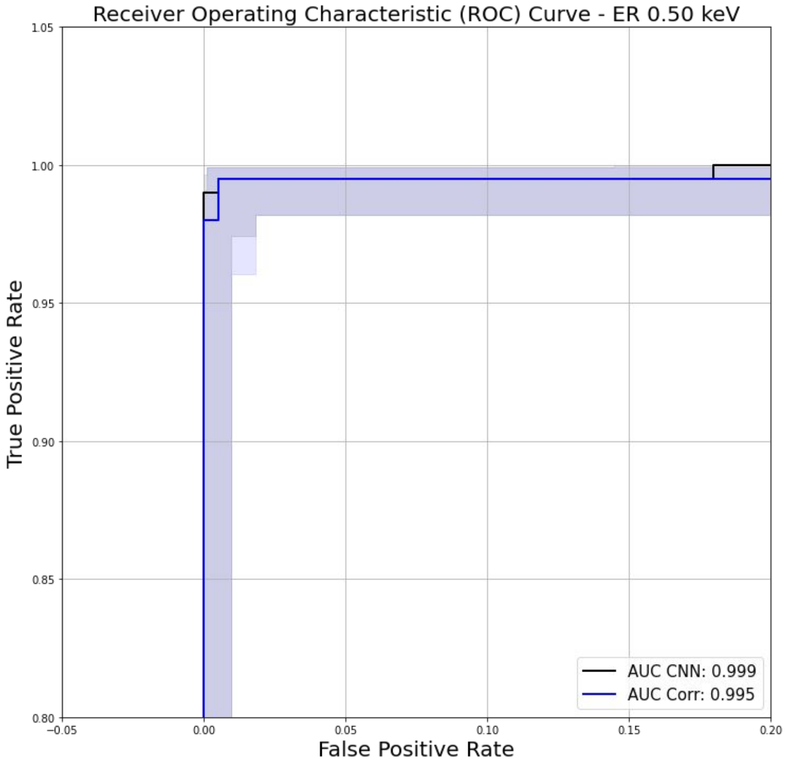
CNN predictions 0.5 keV



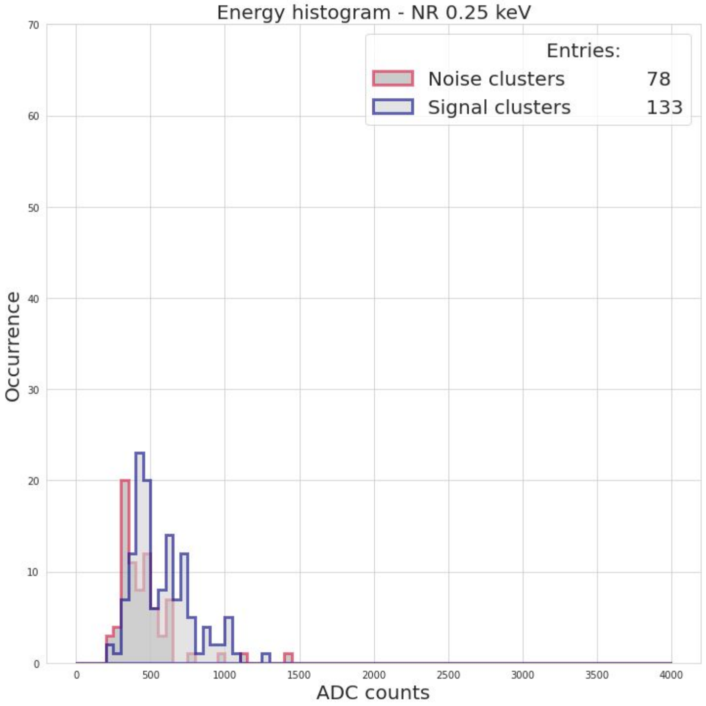


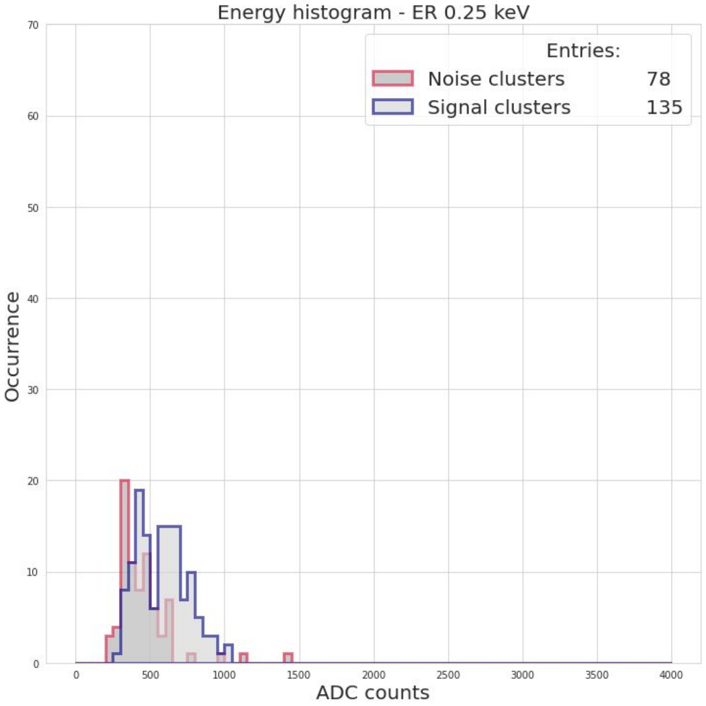
CNN ROC 0.5 keV





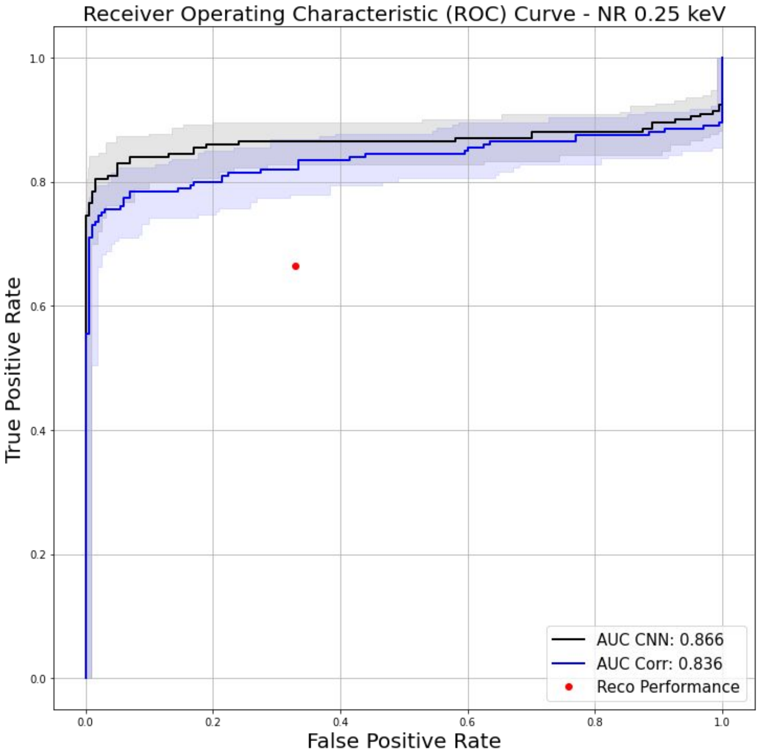
Reconstruction 0.25 keV

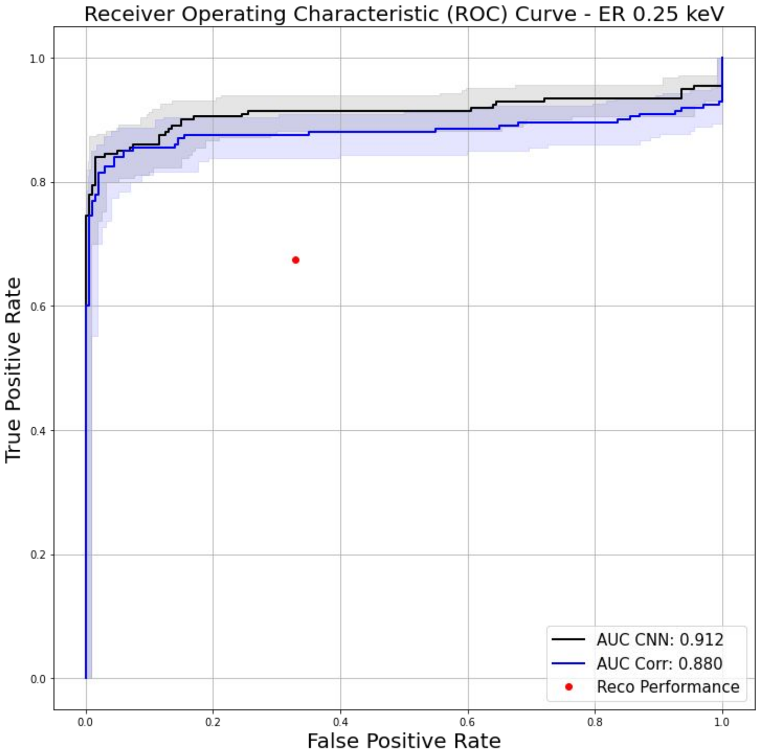




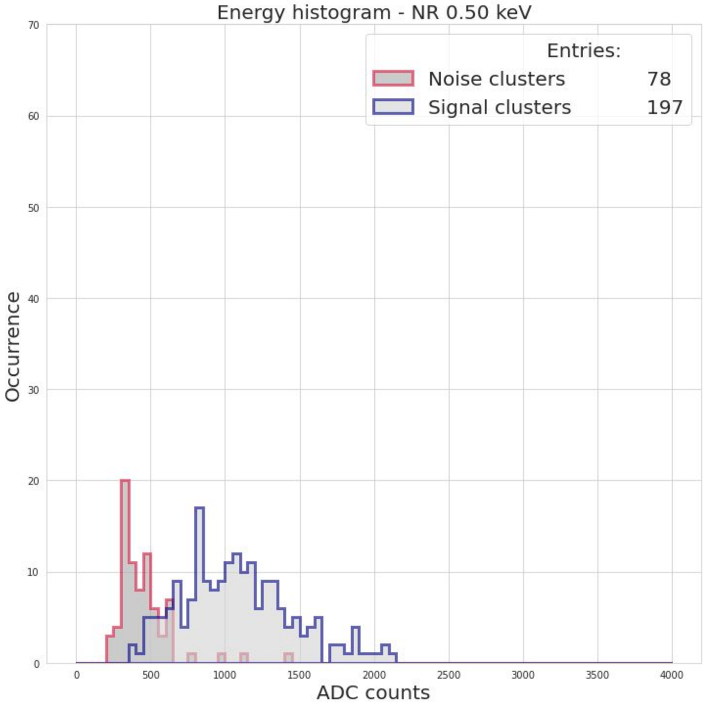
The reconstruction found noise clusters on 66 events (33% false alarm) and detected ~135 signals (67.5% signal detection).

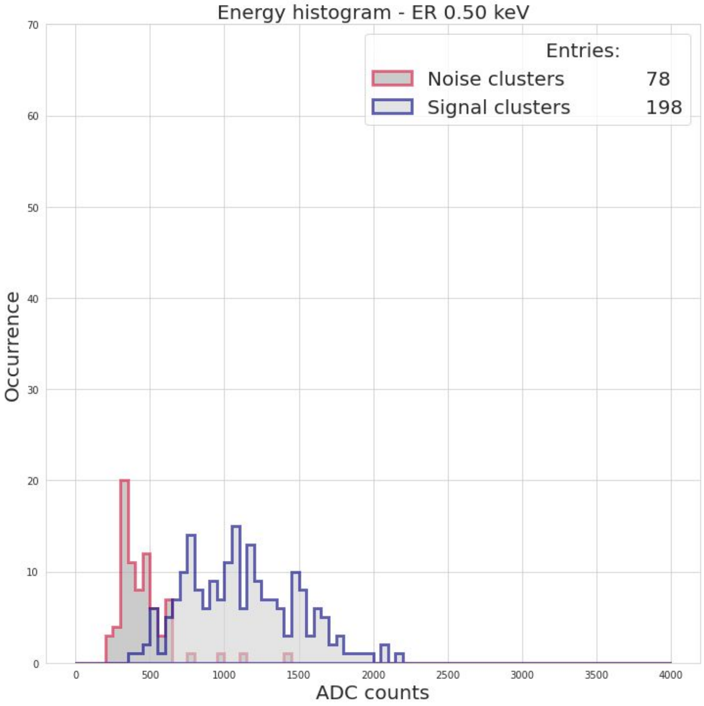
CNN ROC 0.25 keV vs Reco





Reconstruction 0.5 keV



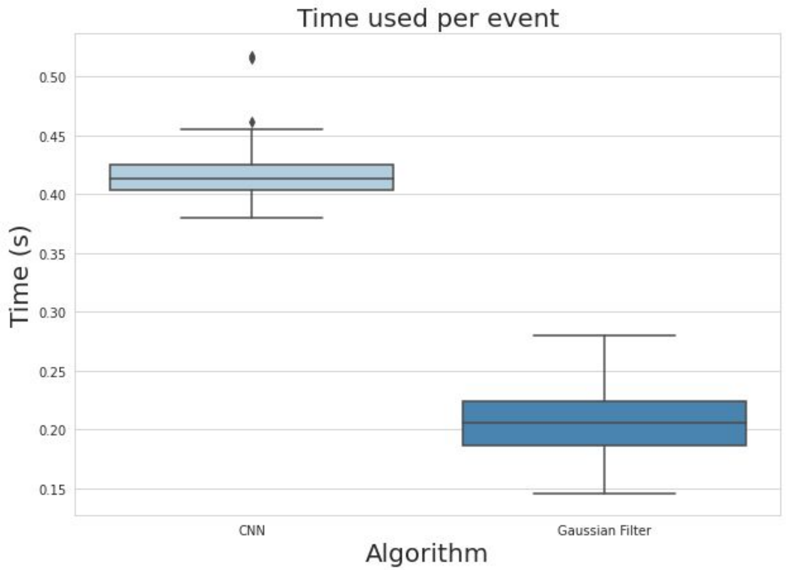


The reconstruction could detect almost all of the signal clusters with 0.5 keV

Time analysis

- Gaussian filter method is slightly faster than the CNN.
  - The CNN method times also consider the time needed to split the image into patches.
  - Gaussian Filter uses the scipy function.

- These times do not consider the time needed to load the image.



4.
Conclusion

Conclusion

▷ The CNN was able to achieve good results comparing to the reconstruction and correlation methods.

▷ It may be optimized with a GPU to make the training and it’s prediction faster.

Next steps

- Train the CNN with different patch sizes and see if the performance in detection and time is increased.

- See how much a GPU may fast up the prediction.

Thanks!