Image signal selection for trigger

Igor Pains

with Rafael Nóbrega

Contents

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** needs **~1.36 Gb** to be stored (Fusion, compressed .mid).
  - A **single day of acquisition** may produce **~266 Gb** of data (Run5 on 26th september).

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

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

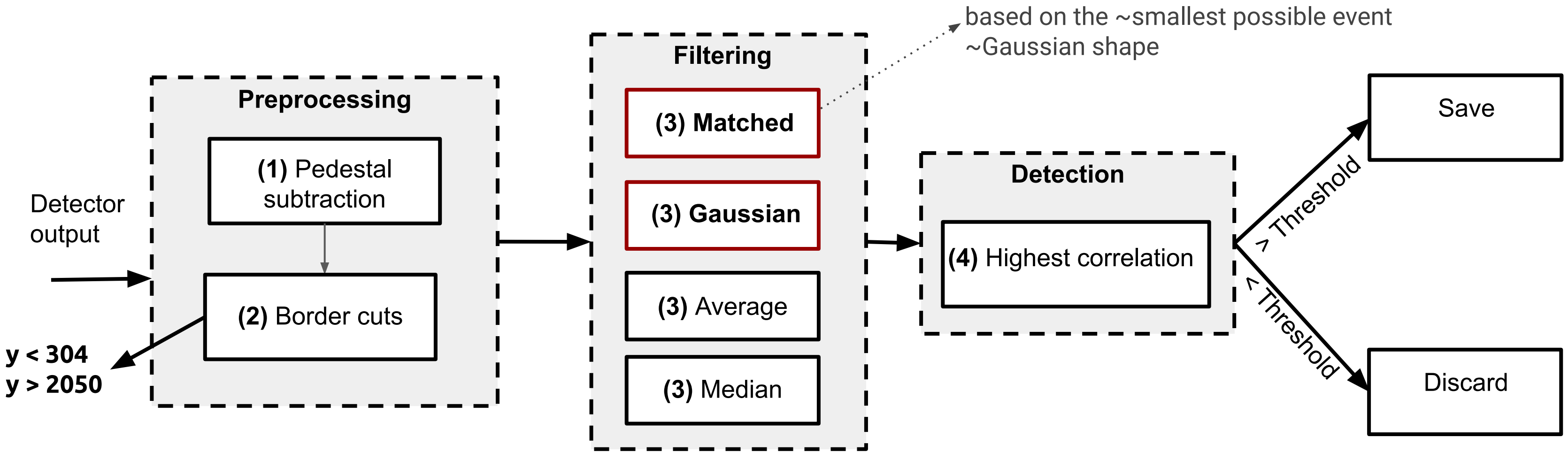
What was done

- Two algorithms proposed:
    - **Filtering** based trigger.
    - **CNN** based trigger.

- A comparison analysis was done using these two algorithms:
    - Trigger **detection performance**.
    - Reconstruction comparison.
    - **Processing time.**

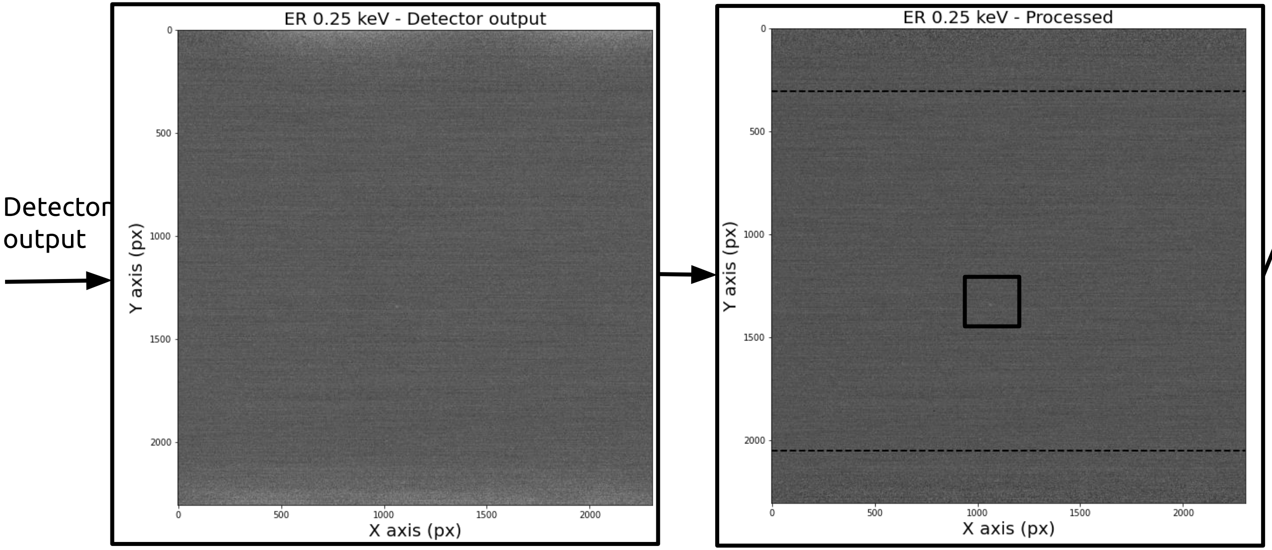
2. Algorithms

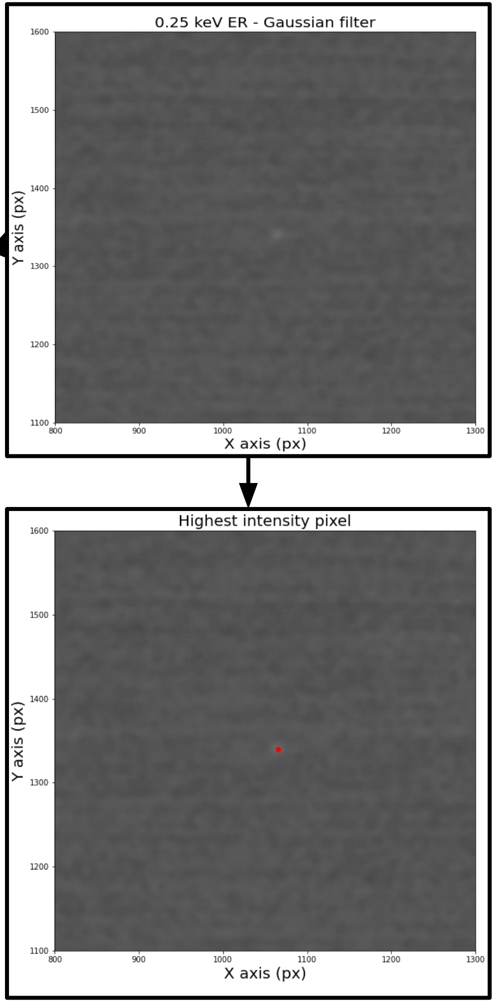
Filtering based trigger



Filter parameters and detection threshold selected based on training data

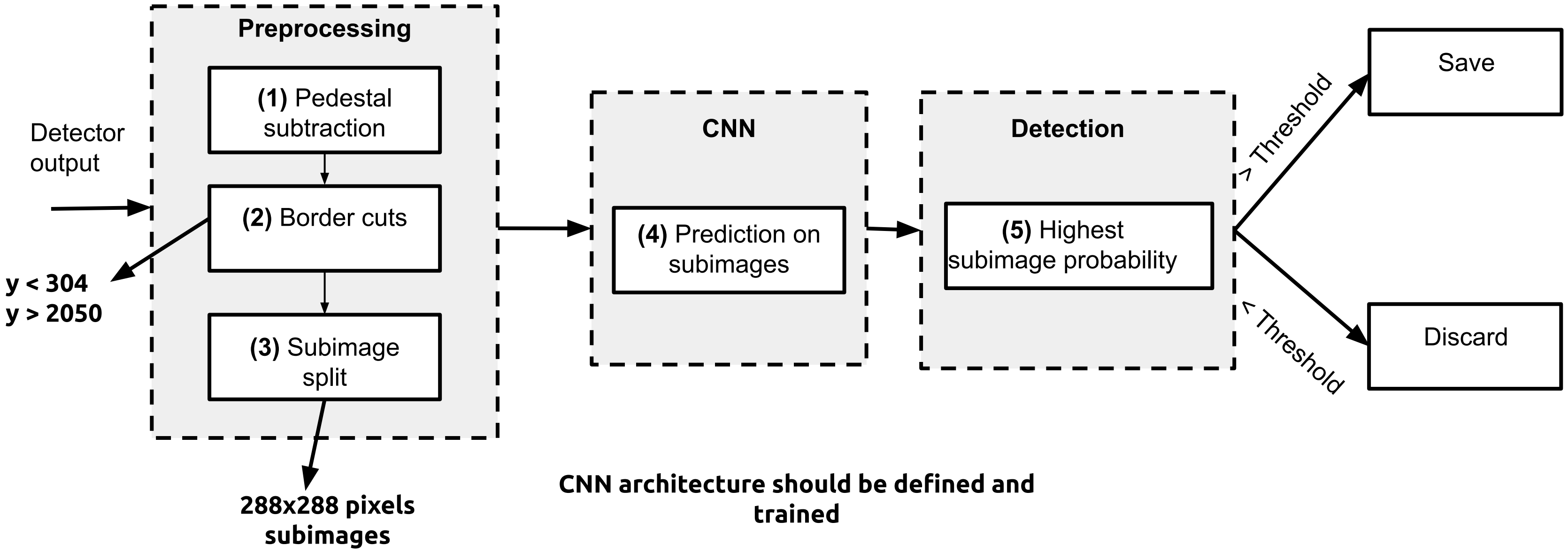
Filtering based trigger





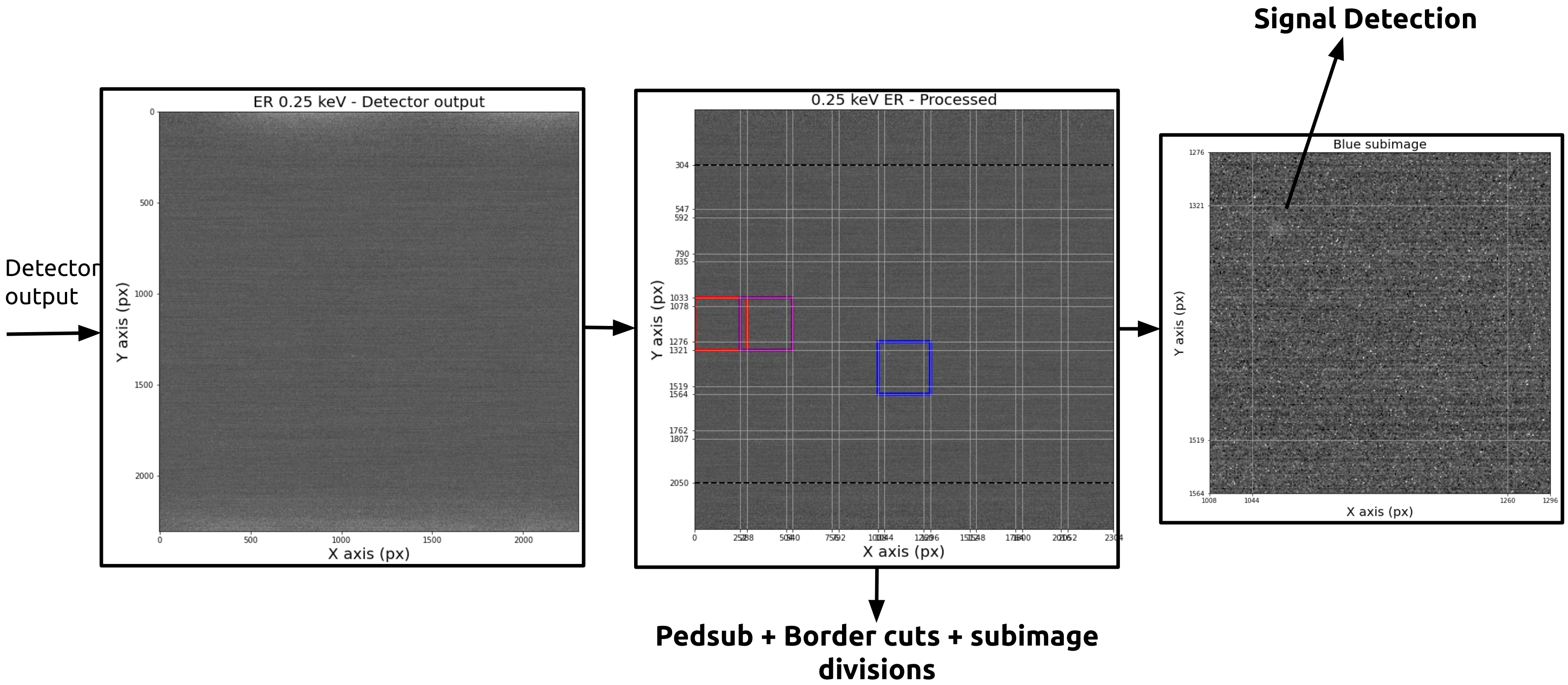
Pedsub + Border cuts

CNN based trigger

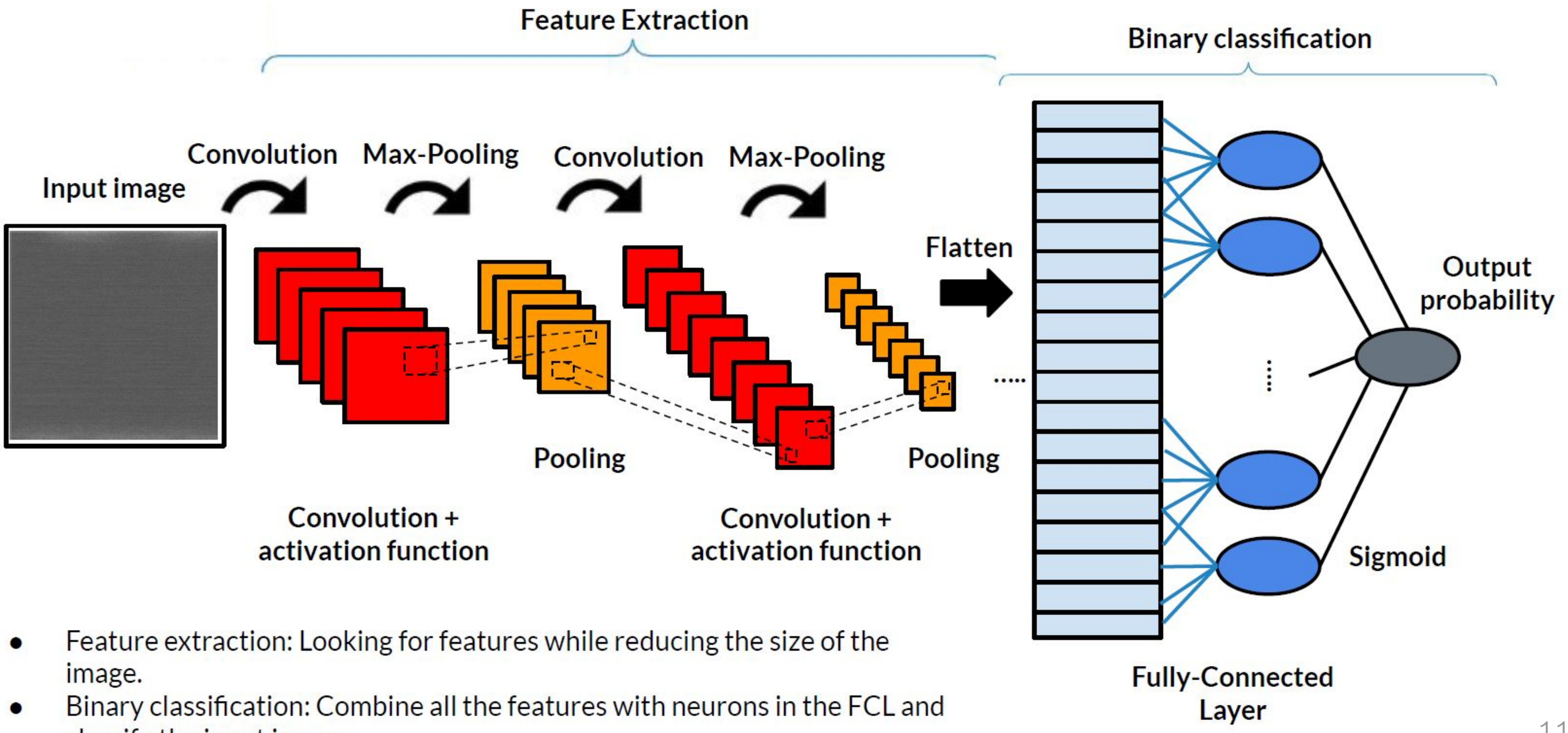


CNN architecture should be defined and trained

CNN based trigger



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.

3. Development

Datasets

- **Training:**
  - Noise dataset: 600 images from pedestals runs (Run4 underground).
  - ER and NR signal simulation: 600 images each containing 0.25-1 keV signals added to pedestal runs (different from noise dataset).

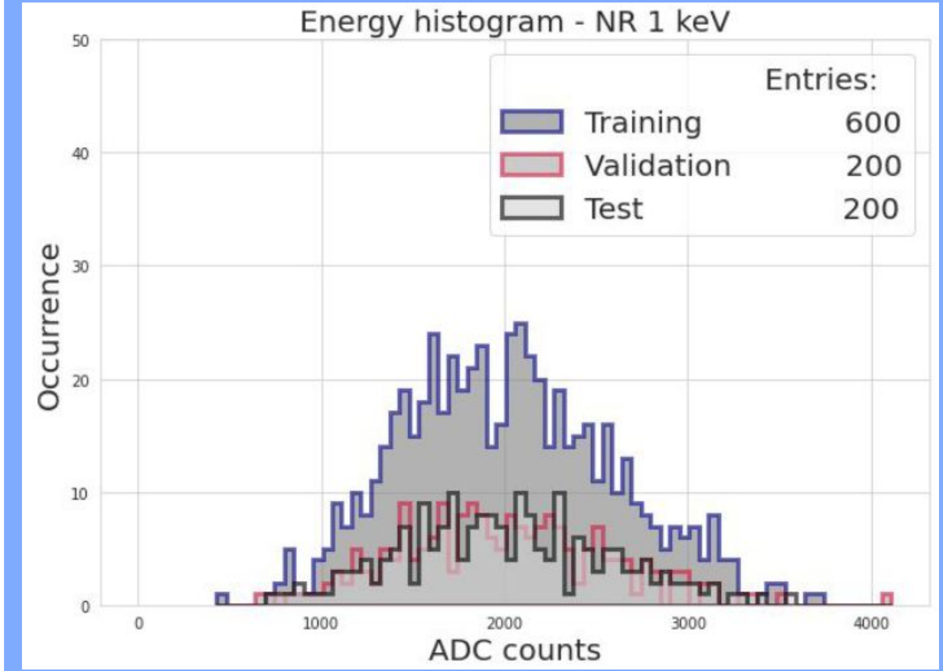
- **Validation:**
  - Noise dataset: 200 images from pedestal runs.
  - ER and NR signal simulation: 200 images each containing 0.25-1 keV signals.

- **Test:**
  - Same configuration as validation.

Datasets

- The signal simulation was **divided** considering the **balance in ADC counts** across the three datasets.

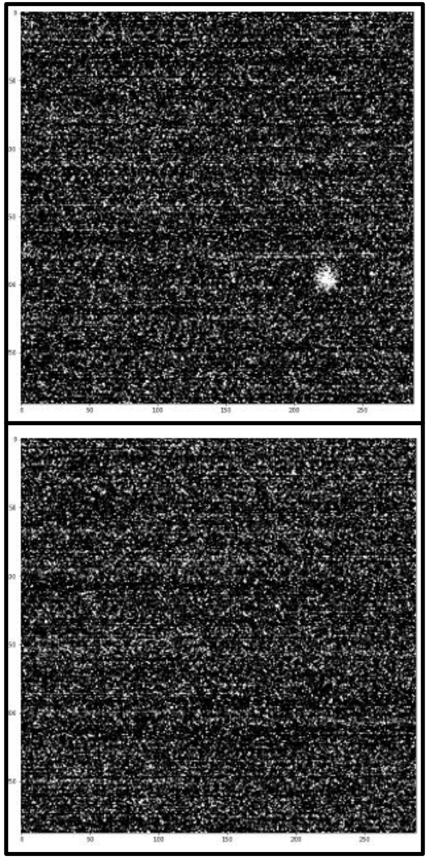
- This prevents the **data split** from **influencing** the results.



CNN training

- Both **ER** and **NR** were used **together** during the **CNN training** using **data augmentation.**
  - 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



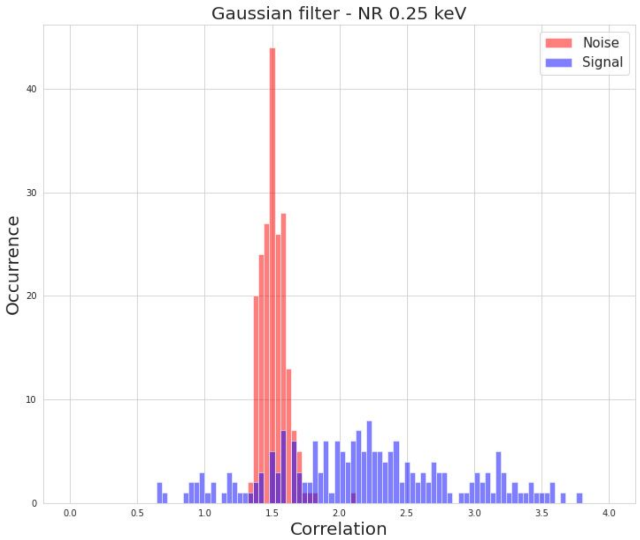
Noise

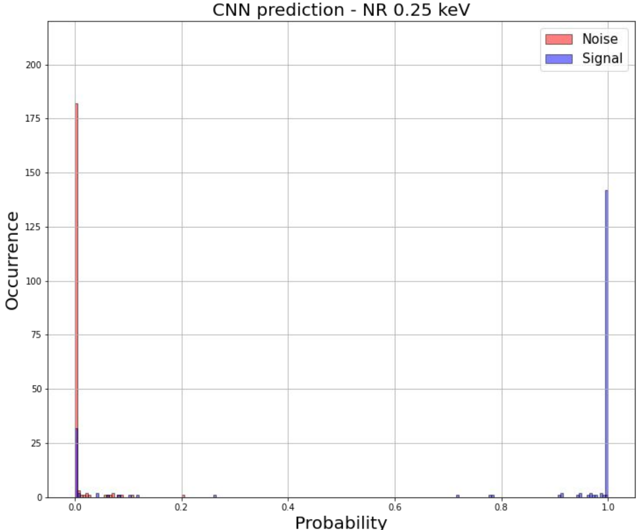
4. Results

Detection performance

- Applying the trigger algorithms on the test dataset results in **two distributions.**
  - The **Gaussian filter** method **output** is a **correlation.**
  - The **CNN output** is a **probability** (more interpretable)

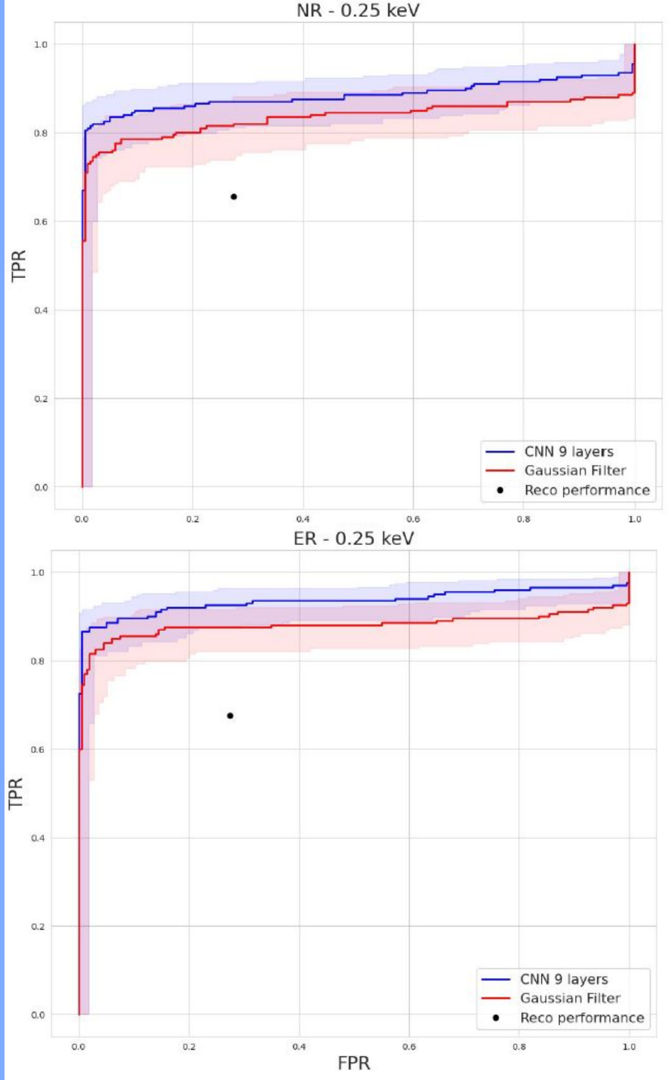
- These distributions may be used on **ROC curves** to evaluate the results.
  - All possible thresholds are used to measure the true positive rate **(TPR)** and false positive rate **(FPR).**
  - **TPR** is analogue to **signal detection.**
  - **FPR** is analogue to **false alarm.**





Detection performance

- The **Gaussian filter** may **detect ~80%** of the **0.25 keV NR** and **ER** events with a **~10% false alarm.**
- The **CNN** may **detect ~80%** of the **0.25 keV NR** and **ER** events with a **~0.5% false alarm.**
- Both methods **outperform** the **reconstruction** in **detecting 0.25 keV** events.
- **All methods** can easily **detect** energies **above 0.5 keV.**

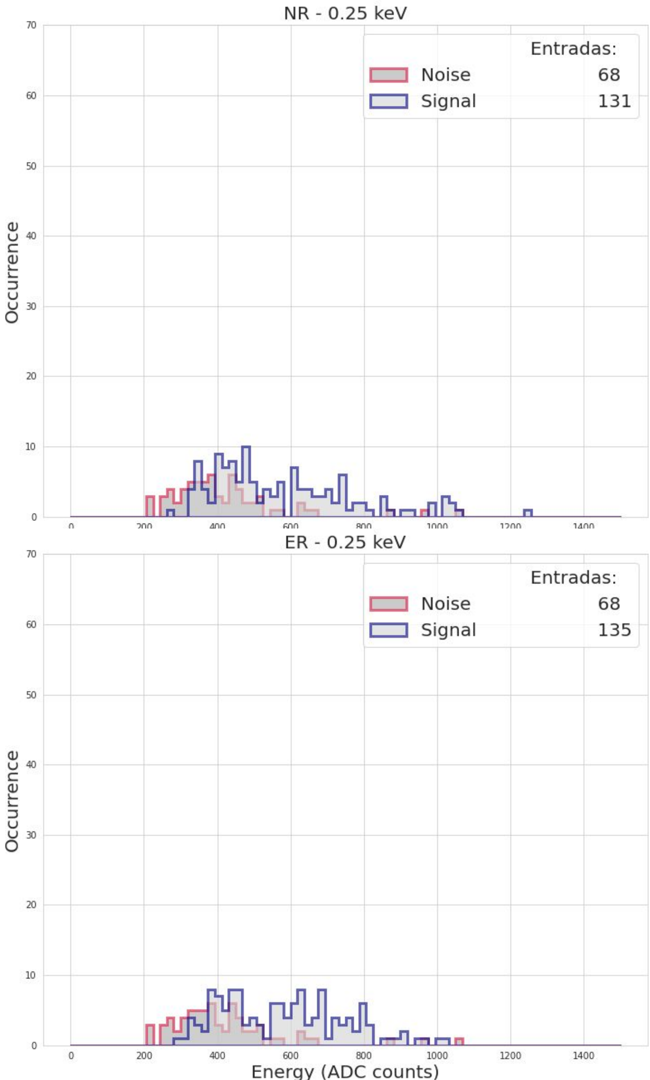


Reconstruction

- The reconstruction code found **68 noise clusters** on **55 images (~27.5% false alarm)** on the test dataset.

- The reconstruction detected **131 NR (~65.5% detection)** and **135 ER (~67.5% detection)** events with **0.25 keV.**

- **All events** found by the **reconstruction** were also **detected** by the **trigger algorithms.**



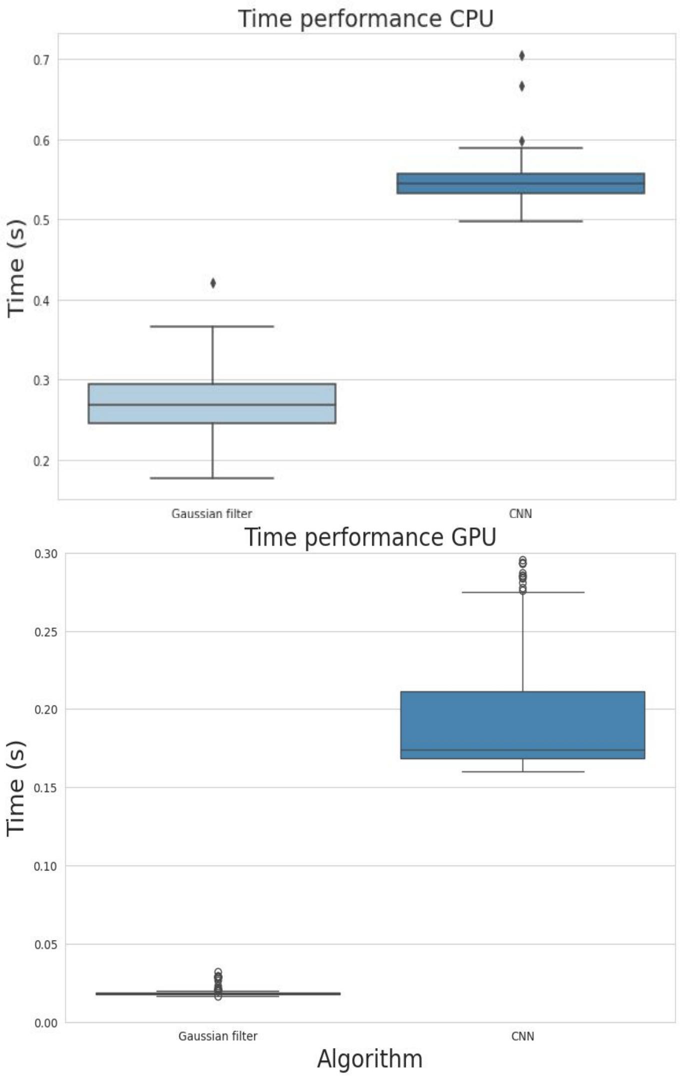
Processing time

- The **Gaussian filter** and **CNN** need **~0.25 and 0.55 seconds** to analyse one image using **CPU**[1] respectively**.**

- The **Gaussian filter** and **CNN** need **~0.02 and 0.2 seconds** to analyse one image using **GPU**[2] respectively**.**

- A **higher detection performance** needs to be **compensated** with a **slower processing time.**

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



5. Conclusions

Conclusions

- The proposed algorithms may **detect ~80%** of the **0.25 keV NR** and **ER simulated events** with a **small false alarm** ratio**.**
    - **Gaussian filter** with **10% false alarm** (**20 out of 200** pedestal images **misclassified**).
    - **CNN** with **0.5% false alarm** (**1 out of 200** pedestal images **misclassified**).

- The proposed algorithms may **detect ~100%** of the events **above 0.5 keV.**

- The **CNN needs a GPU** to have a **proper margin time** to analyse an image considering the current exposure time, whereas the **Gaussian filter** may be implemented with a **CPU.**

- **All the events** detected by the **reconstruction** were easily detected by the **proposed algorithms.**

Next steps

- Study methods to simplify the CNN model (on going).
  - **Bit reduction, weight combination, pruning and vectorization.**

- Test the CNN on the DAQ machine.
  - **GPU: Quadro RTX 5000.**

- Test popular CNN architectures such as AlexNet, GoogleLeNet, Unet with adaptations.

- Three possible approaches for the trigger:
  - Save the entire image.
  - Save subparts of the images.
  - Retrain the CNN to reject also natural radioactivity.

Next steps

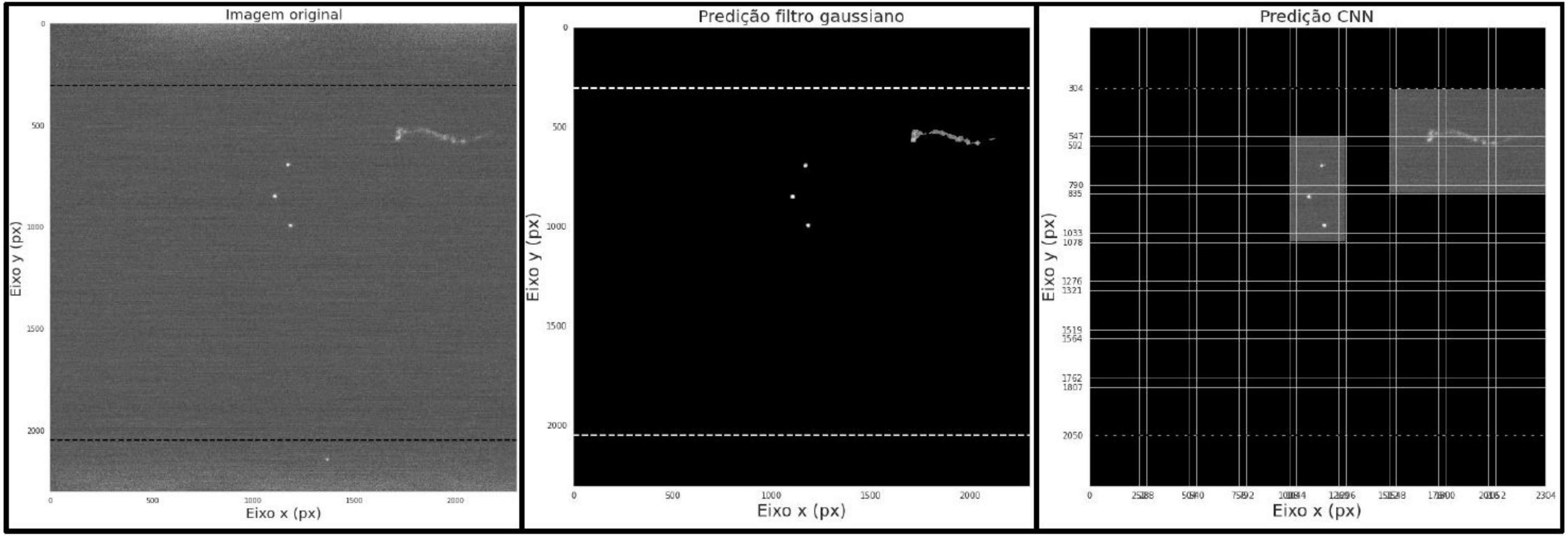


Imagem original
Predição filtro gaussiano
Predição CNN
Eixo x (px)
Eixo y (px)

Thank you

Credits

This is where you give credit to the ones who are part of this project.

Did you like the resources on this template? Get them for **free** at our other websites.

Presentation template by Slidesgo

Icons by Flaticon

Images & infographics by Freepik

Author introduction slide photo created by **katemangostar** - Freepik.com

Big image slide photo created by **jcomp** - Freepik.com

Text & Image slide photo created by **rawpixel.com** - Freepik.com

Text & Image slide photo created by **Freepik**

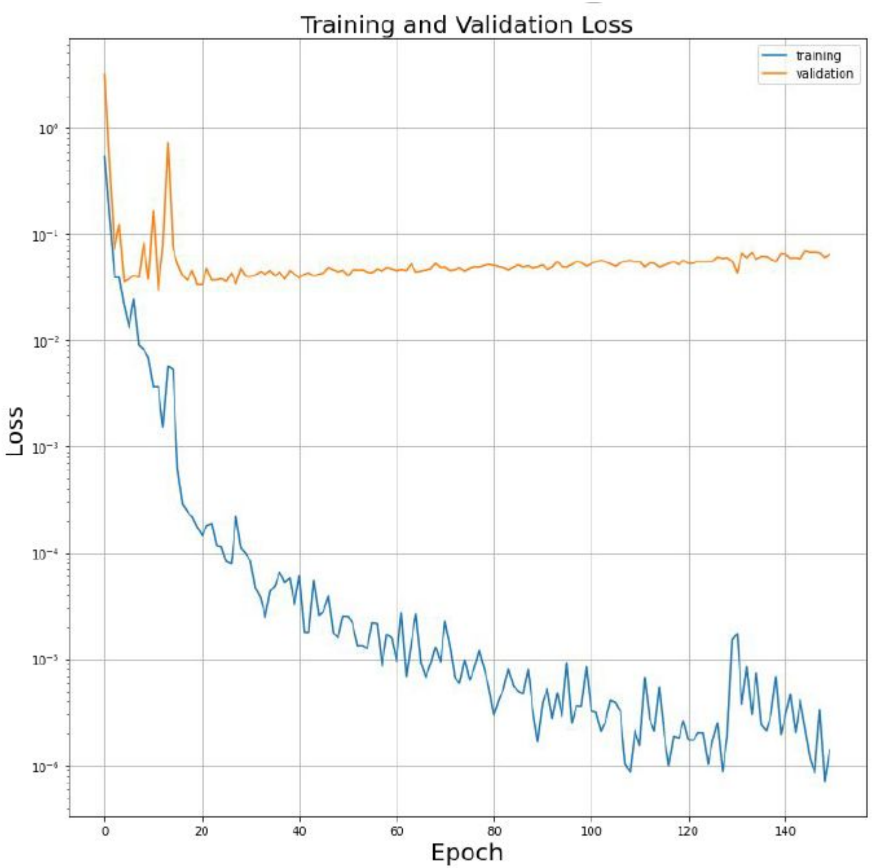
Backup

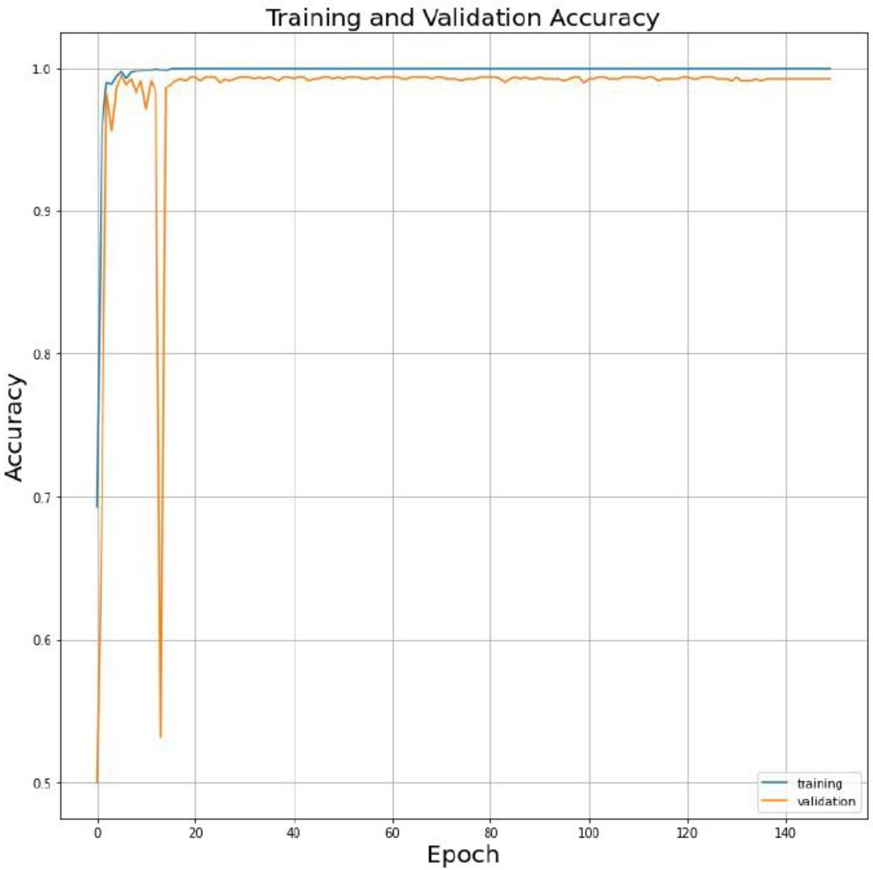
CNN architecture

- 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.

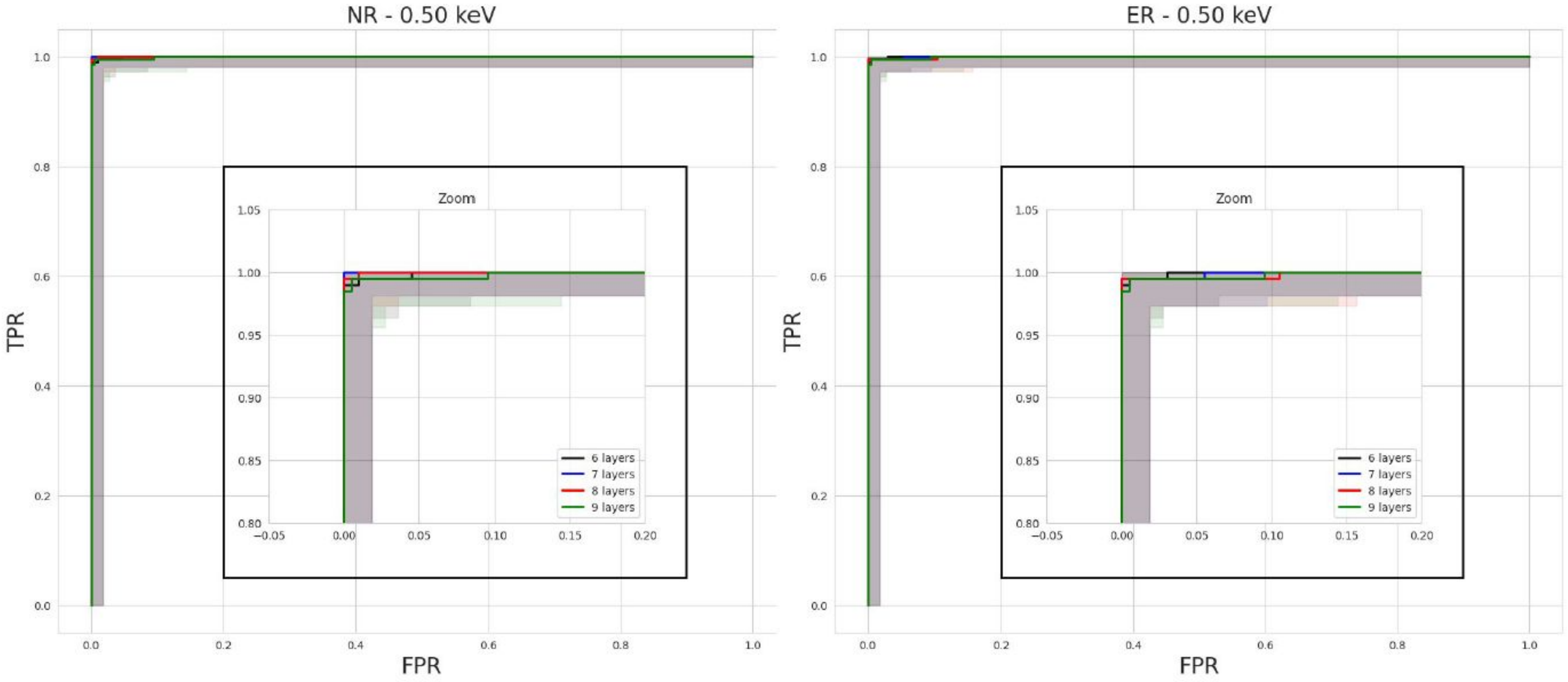
- Four CNN architectures were selected (number of layers from 6 to 9).
  - The bayesian optimization was used during training.
  - The approach is to select a range of possible hyperparameters (number of filters in each conv layer, neurons on dense layer, etc) and the method will find the optimal values.

CNN training





CNN 0.5 keV



NR - 0.50 keV
ER - 0.50 keV
TPR
FPR
Zoom
6 layers
7 layers
8 layers
9 layers