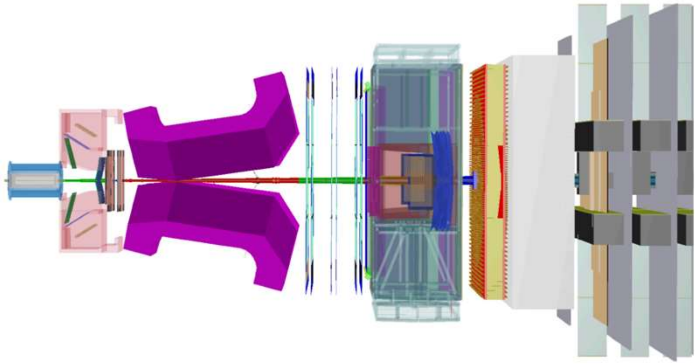
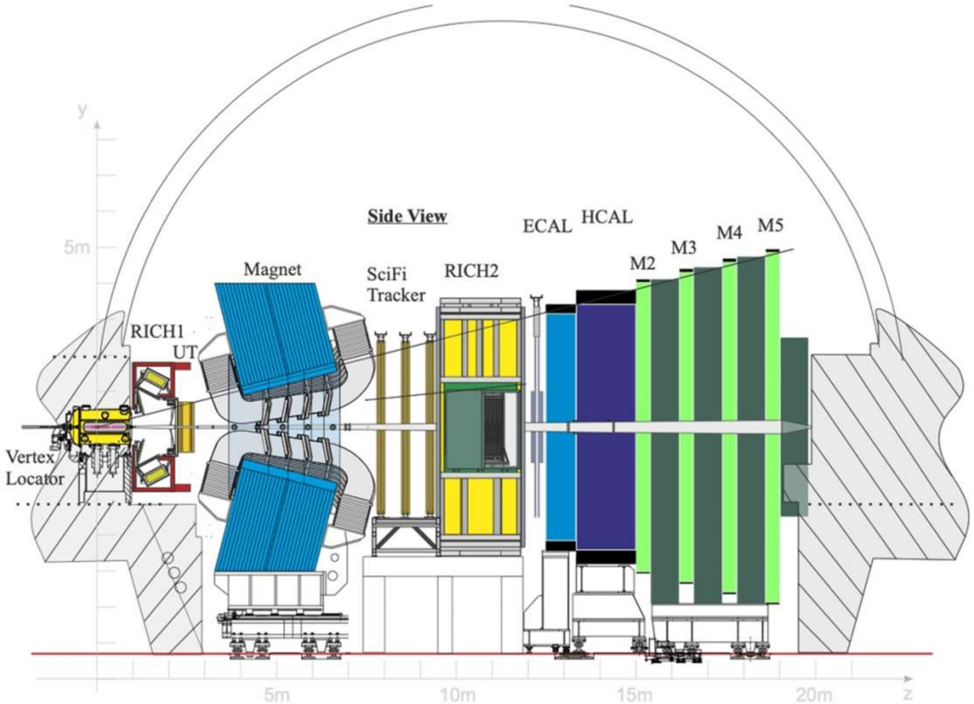


A Machine Learning Approach for Particle Tracking in RICH Detectors

SUPERVISOR: MARTINO BORSATO

CO-SUPERVISOR: MAURIZIO MARTINELLI

DANIELE GHEZZI

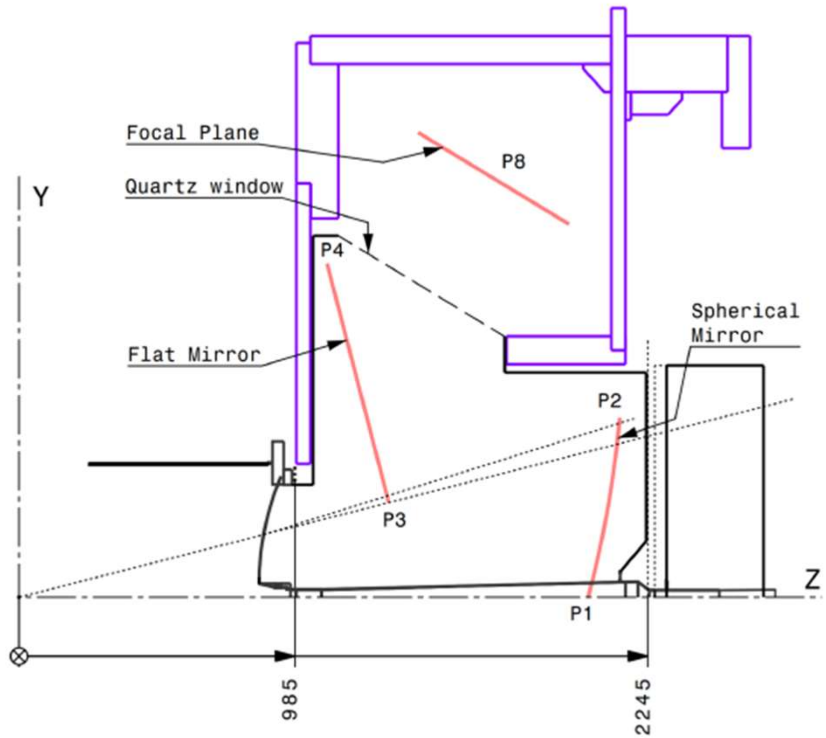


LHCb experiment

- **LHCb** (**Large Hadron Collider beauty**) is one of the four experiments at the Large Hadron Collider (LHC)
- Designed to investigate **matter-antimatter asymmetry** through measurements of **CP violation**
- Features a **single-arm forward geometry**, optimized to study decays involving **charm** (*c*) and **beauty** (*b*) hadrons

RICH detectors

- The **RICH** (**Ring Imaging Cherenkov**) system consists of two detectors: **RICH1** and **RICH2**
- They exploit **Cherenkov Radiation** to perform **Particle Identification** (**PID**) in the second-stage trigger (**HLT2**) by measuring the Cherenkov angle
- Each detector is divided in two identical halves and uses a series of **mirrors** to focus the photons onto a plane of **Multi-Anode Photomultiplier Tubes** (**MaPMTs**)



RICH1 upper structure

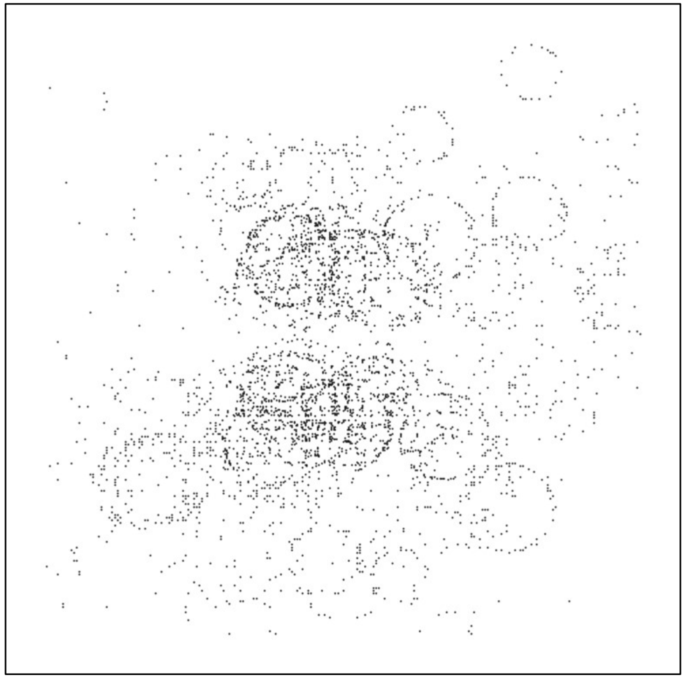
For each pp event, **more than 150 charged particles are produced**, each generating a Cherenkov ring on the MaPMTs plane, resulting in a **complex pattern of overlapping rings** per event.

Sophisticated PID algorithm are used based on HLT1 information.
However, **what if RICH could directly exploit its ring patterns to determine the centers?**

The **ring centers correspond to the particle positions**, providing potential additional points for the **LHCb tracking system**.

The complexity of the ring patterns naturally suggests using **Machine Learning** techniques, such as **Computer Vision**, to identify the centers.

Keypoint detection emerges as the most suitable method for this task!



RICH1 ring pattern from full Monte Carlo

Keypoint Detection

Computer vision task consisting in the **identification and localization of points of interest**

*In our case: **keypoint = ring center***

Regression Approach

The network outputs directly the **keypoints coordinates**

Advantage: computationally efficient, suitable for real time applications

Disadvantage: struggles with noisy and ambiguous input

→ **YOLO model**

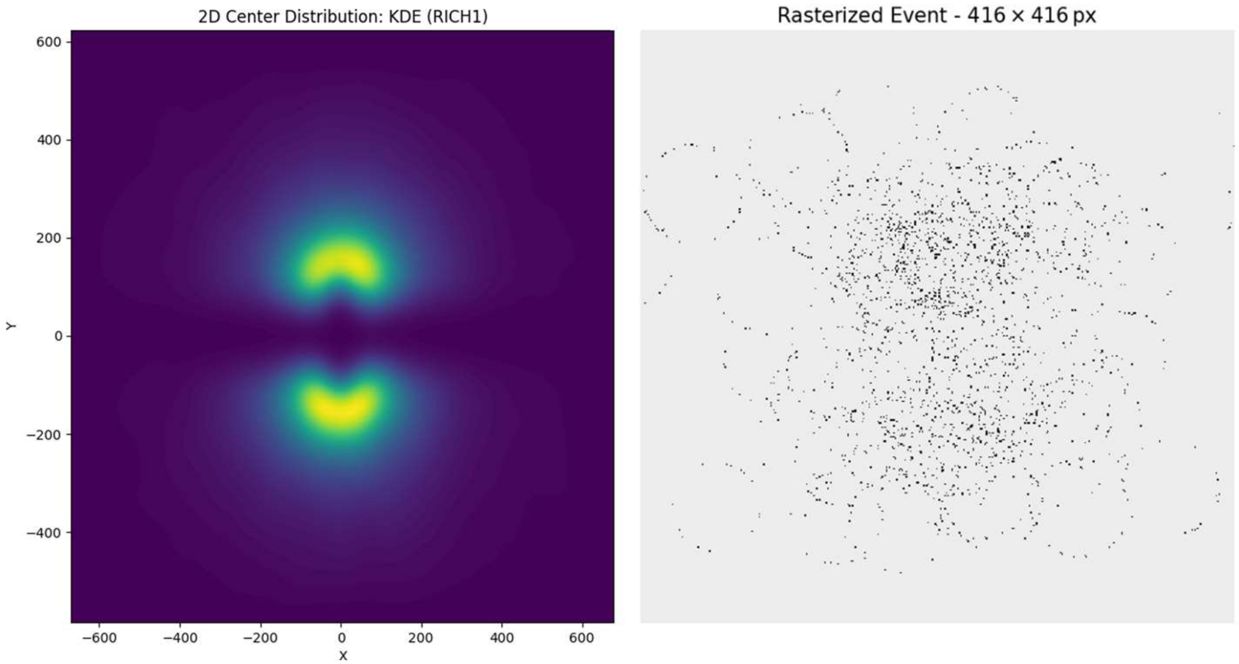
Heatmap Approach

The network outputs a **probability distribution** over the image, whose peaks correspond to the Cherenkov center

Advantage: more robust localization

Disadvantage: the extraction of finite coordinates increases inference time

→ **UNet model**



Two different image size are considered:

1. Each pixel correspond one-to-one to MaPMTs: **416x416 px**
2. Each pixel is sensitive to the detector error: **800x800 px**

The Dataset

Supervised Machine Learning requires a large **labeled dataset for training**: for RICH, the labels come from **Monte Carlo** simulations

→ **Synthetic Cherenkov Generator**

(from Giovanni Laganà work)

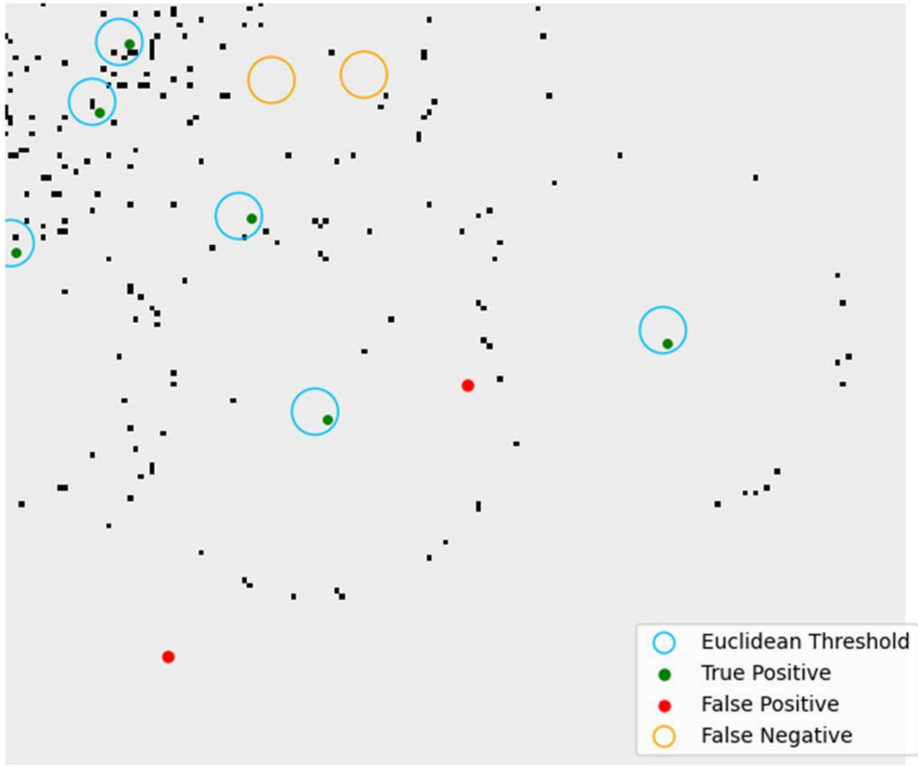
The key physical processes in *pp* collisions and RICH detectors are replicated by **exploiting probabilistic distributions derived from known full Monte Carlo simulations** via Kernel Density Estimation (KDE)

Evaluation Metrics

Quantify the efficiency and accuracy of a model

Euclidean Threshold:

A predicted keypoint is considered correct if its distance from a ground truth keypoint is smaller than the Euclidean threshold



True Positive (TP)

the predicted point is a correct prediction under the chosen threshold.

False Positive (FP)

the predicted point is considered incorrect because it is farther than the chosen threshold from any ground truth point

False Negative (FN)

a ground truth point that has not been matched by any predicted point within the chosen threshold

Evaluation Metrics

We assume an acceptable 1% error (12 mm on the 1200 mm MaPMT plane) on predicted center coordinates

The 800px rasterization achieved overall better performance!

Precision (P):

it measures the proportion of predicted point which are correct under the chosen threshold

$$P = \frac{TP}{TP + FP}$$

Recall (R):

it is the ratio of ground truth points correctly detected within the threshold

$$R = \frac{TP}{TP + FN}$$

F1 Score:

it is the harmonic mean between precision and recall, used to make an overall model evaluation

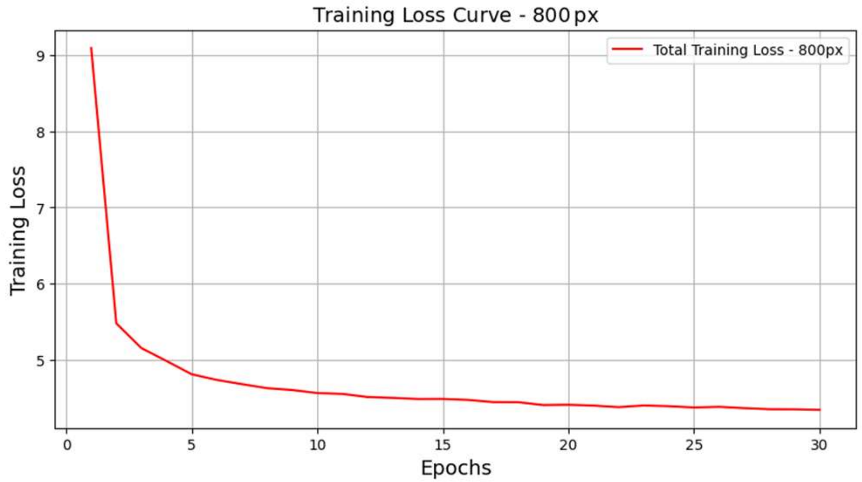
$$\mathrm{F1} = 2 \cdot \frac{P \cdot R}{P + R}$$

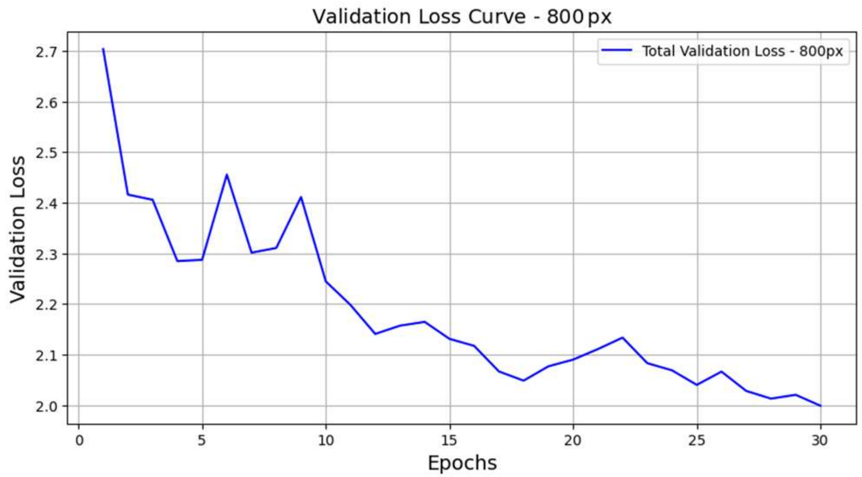
Mean Inference Time:

It is the mean time required by a trained model to generate predictions on a single event

YOLOv11-Pose model

- ***Pose*** model is **YOLO keypoint detection network** focused on detecting and classifying human body parts, but it can be adapted to RICH centers detection
- It detects **keypoints** and also object **bounding boxes**
- YOLO returns for each predicted keypoint a **confidence score**, which represents the probability that a true keypoint is present at the corresponding predicted coordinates

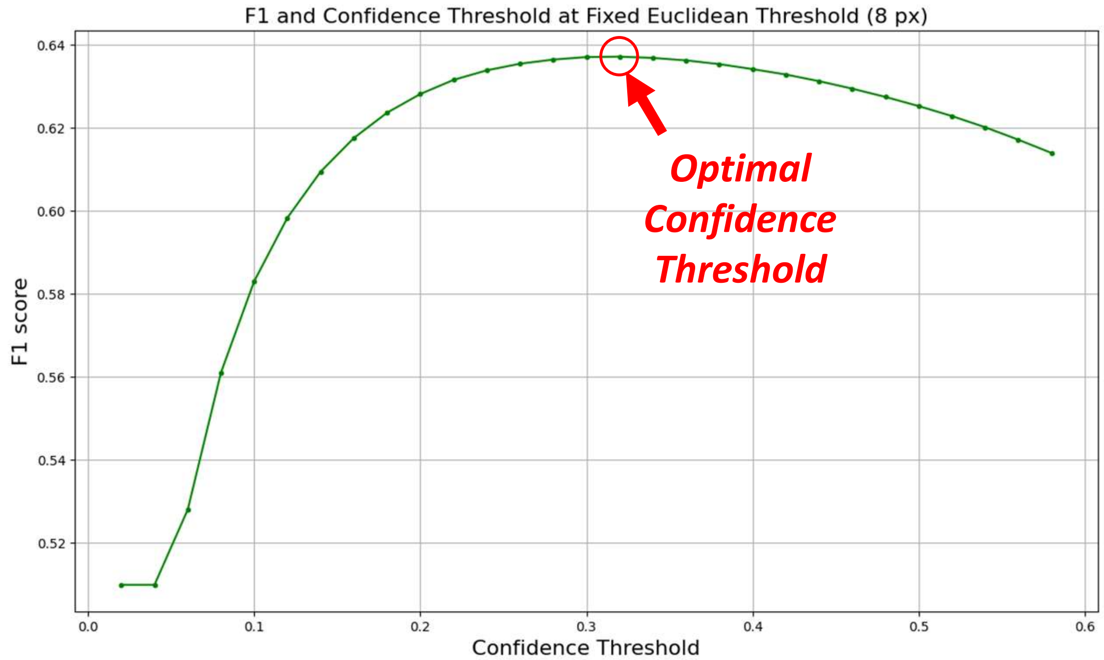




25000 synthetic events generated: 90% training, 10% validation, with 150–175 rings per event.

Confidence Threshold:
minimum confidence value above which a predicted center is accepted as true

We choose the confidence threshold that maximizes the F1 score



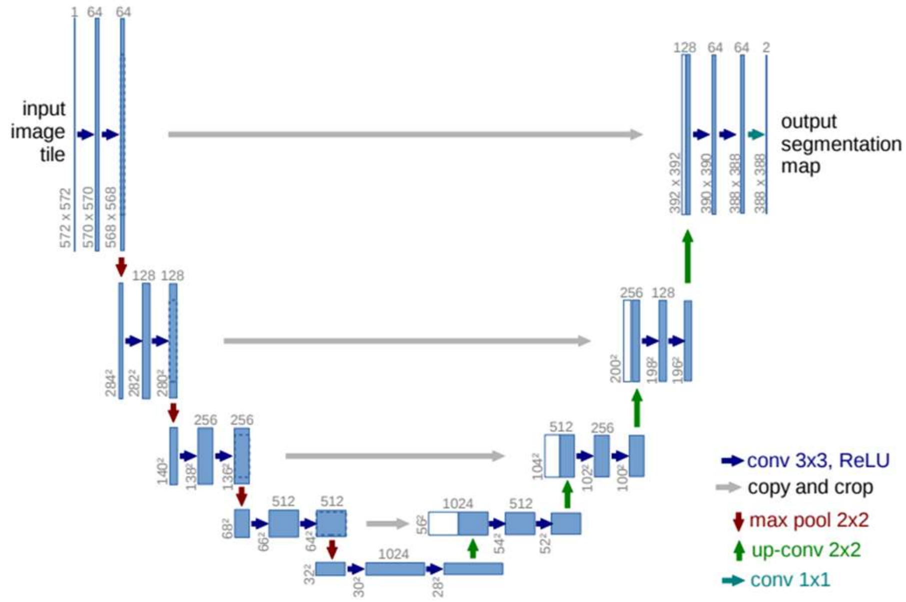
- **80.5% of the predicted centers are correct** within a 12mm error margin
- The recall shows that **the model identifies only 52.7% of the true ring centers**, highlighting a significant limitation in effectiveness
- Inference time was estimated on an NVIDIA A6000 GPU

Img Size	Precision	Recall	F1 Score	# of Predicted KP	Inference Time (ms)
800px	0.805	0.527	0.637	103.89 ± 0.06	13.541 ± 0.011

Mean metrics computed over 2500 validation events

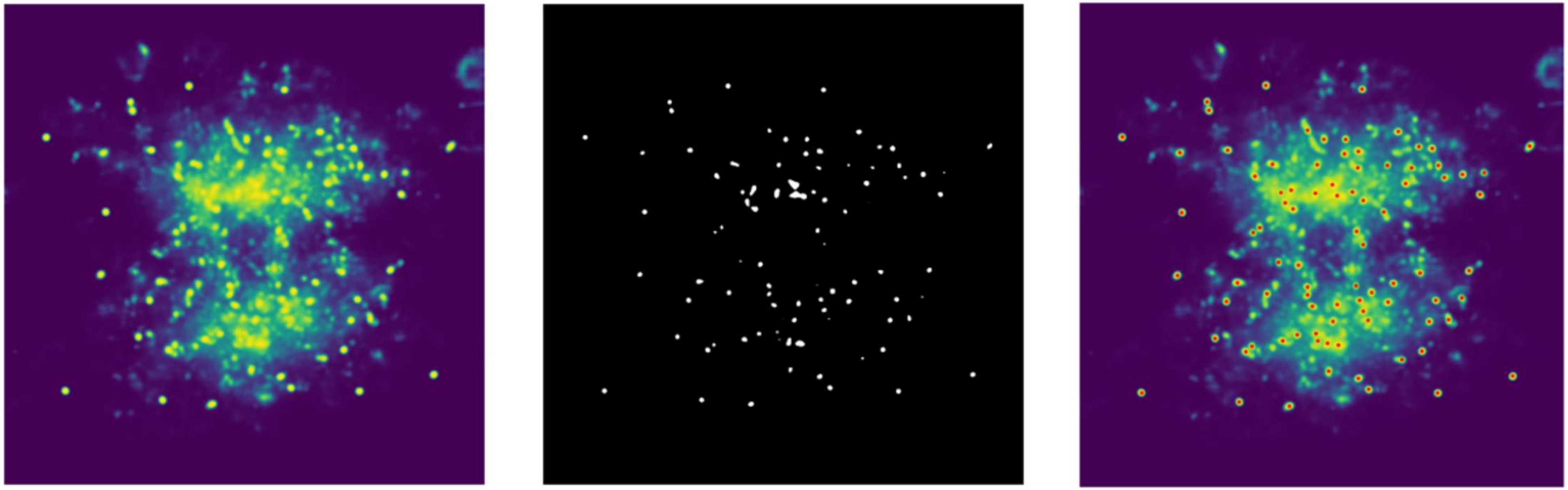
UNet model

- **UNet** is a convolutional neural network that outputs a **probability heatmap**
- It features a **symmetrical encoder-decoder architecture** with skip connections, built in *Pytorch*
- Center inference involves two steps:
  1. **Prediction of the heatmap** by the network
  2. **Extraction of local maxima** from the heatmap to obtain the **coordinates** of the centers
- The UNet architecture integrates **Soft Attention gates** within its skip connections



UNet model requires **keypoint extraction** from the heatmap to obtain the center coordinates:

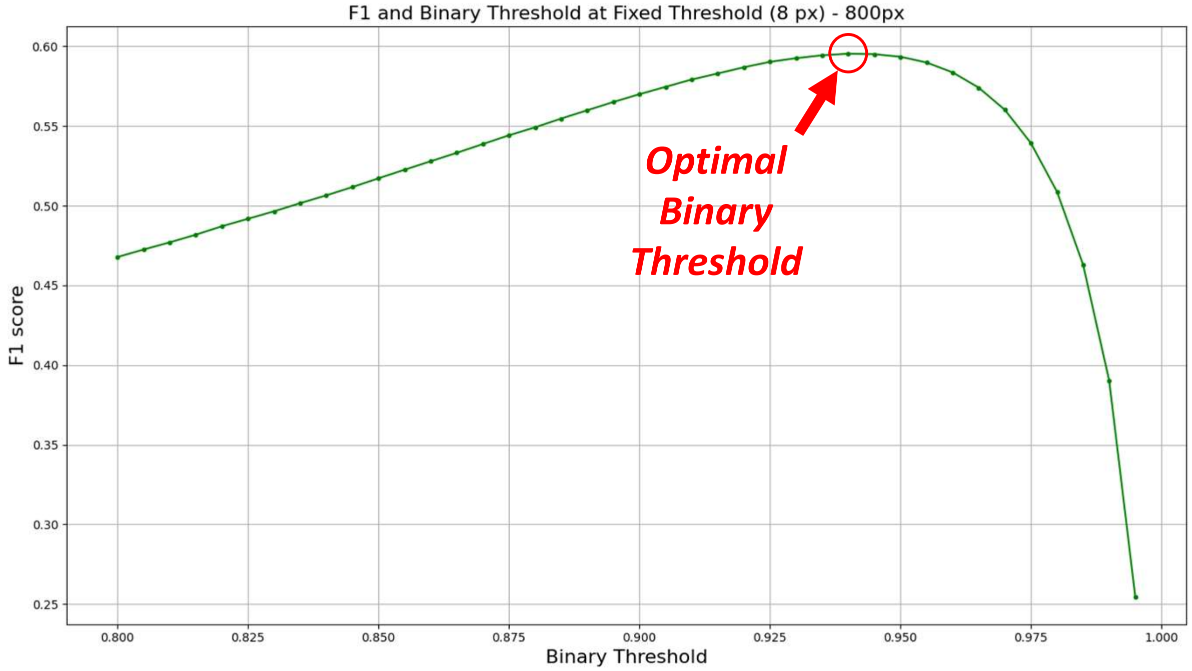
1. **Binary map generation**: *If pixel.value > binary_threshold: pixel.value = 1, else pixel.value = 0*
2. **Center estimation:** centroid of each connected component is used as the center coordinate



The UNet model has been trained on the same YOLO dataset with 416px and 800px rasterization

Binary Threshold:
minimum probability above which a peak is considered part of a connected component

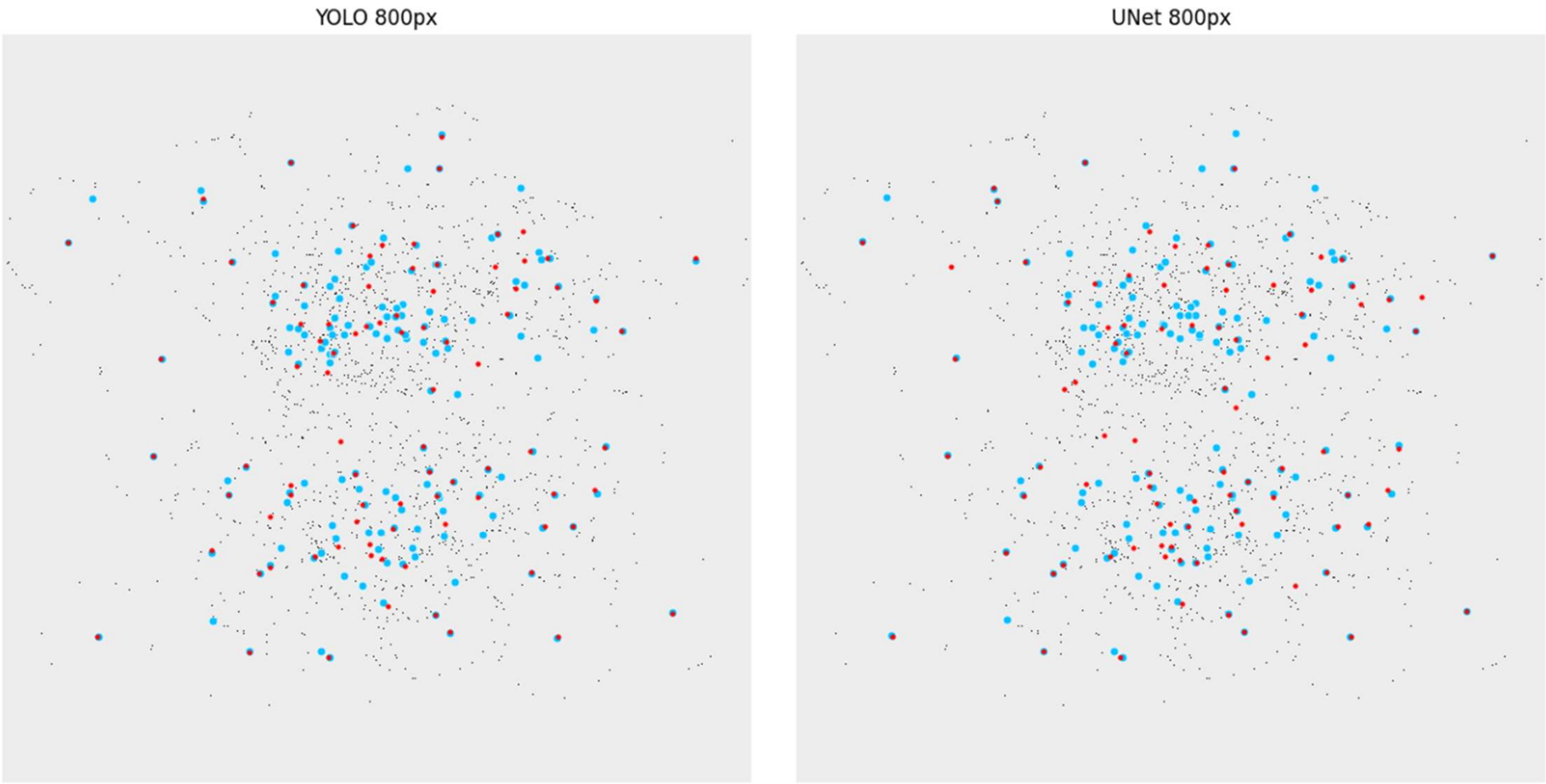
We choose the binary threshold that maximizes the F1 score



- UNet model shows **lower precision and recall** compared to the YOLO model
- Mean inference time is about **four times longer** than YOLO one (NVIDIA A6000 GPU)

Img Size	Precision	Recall	F1 Score	# of Predicted KP	Inference Time (ms)
800 px	0.784	0.483	0.597	100.10 ± 0.16	50.504 ± 0.266

Mean metrics computed over 2500 validation events



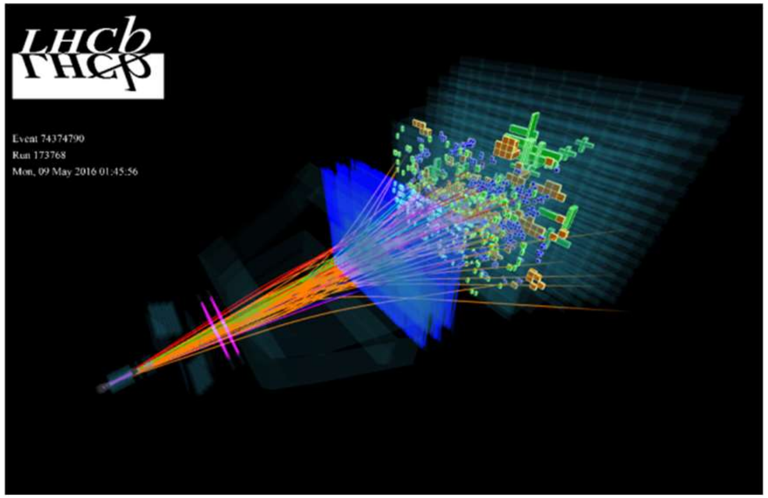
Example of an event inferred by the two models: red points indicate the predicted ring centers, while cyan points represent the truth.

Outlook

- **Regression** approach is **more efficient** than **heatmap** approach
- Both models **struggle** in reconstructing Cherenkov centers **in regions with high ring density**

In future works:

1. **New model architectures**
2. **Parallelization** and **Edge Machine Learning**
3. **Validation** on LHCb **simulations** and **real data**
4. **RICH** for **time-stamping tracks**



BACKUP SLIDES

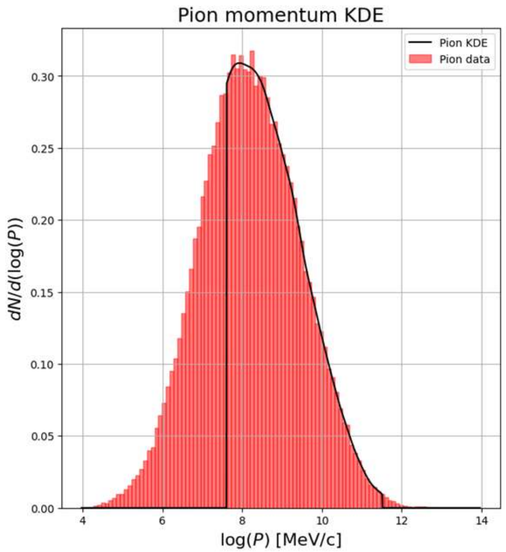
Synthetic Cherenkov Generator

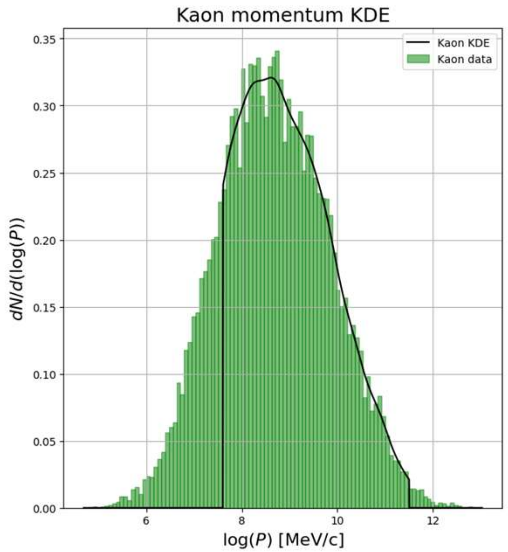
The generator reproduces the main characteristics of real data by exploiting Monte Carlo distributions:

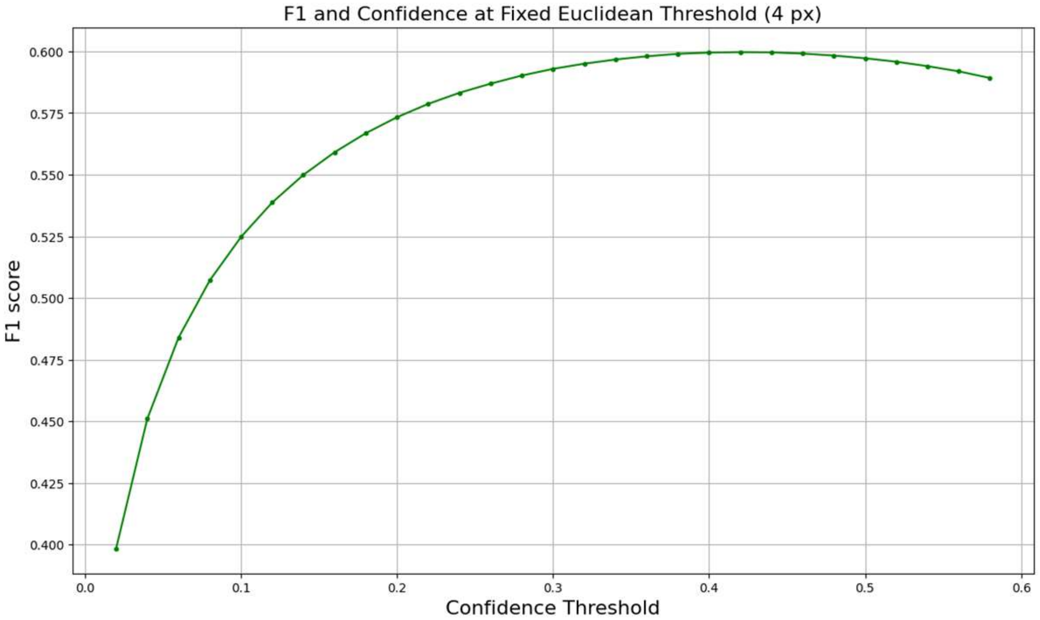
- Cherenkov angle dependece on momentum
- Number of photon hits generated according to Frank-Tamm formula
- Cherenkov angle resolution of 0.8mrad (RICH1)

Generation steps:

1. Particle type selection
2. Momentum sampling
3. Compute Cherenkov angle
4. Compute ring radius
5. Compute the number of photon hits
6. Center sampling
7. Ring generation with radial smearing



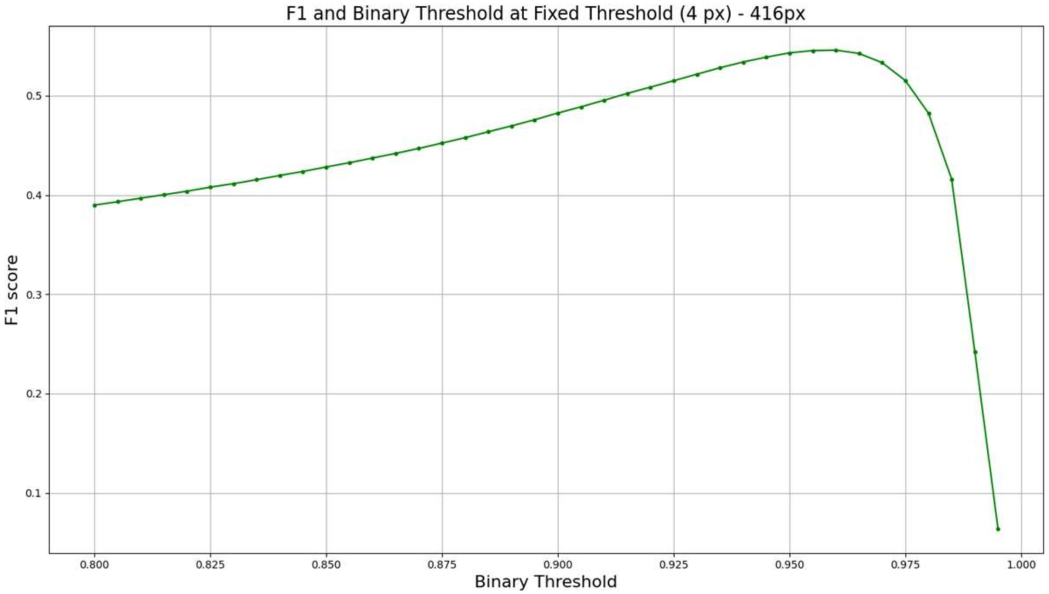




Img Size	Precision	Recall	F1 Score	# of Predicted KP	Inference Time (ms)
416px	0.815	0.475	0.600	92.46 ± 0.06	12.693 ± 0.019

Mean metrics computed over 2500 validation events

YOLO - 416px Rasterization

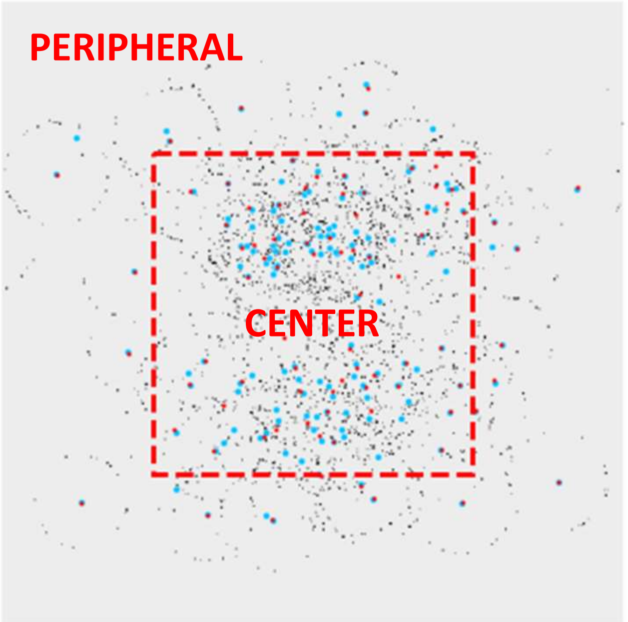


Img Size	Precision	Recall	F1 Score	# of Predicted KP	Inference Time (ms)
416px	0.905	0.390	0.544	69.95 ± 0.11	20.041 ± 0.369

Mean metrics computed over 2500 validation events

UNet – 416px Rasterization

Central and Peripheral Metrics



Img Size and Region	Precision	Recall	F1 Score	# of Predicted KP	# of GT KP
416px center	0.703	0.370	0.485	66.46 ± 0.03	126.27 ± 0.04
416px peripheral	0.950	0.792	0.864	30.15 ± 0.01	36.17 ± 0.01
800px center	0.765	0.430	0.550	70.98 ± 0.03	126.27 ± 0.04
800px peripheral	0.885	0.767	0.821	31.35 ± 0.01	36.17 ± 0.01

YOLO

Img Size and Region	Precision	Recall	F1 Score	# of Predicted KP	# of GT KP
416px center	0.869	0.303	0.449	43.95 ± 0.03	126.27 ± 0.04
416px peripheral	0.952	0.687	0.796	26.00 ± 0.01	36.17 ± 0.01
800px center	0.743	0.411	0.529	69.72 ± 0.03	126.27 ± 0.04
800px peripheral	0.867	0.723	0.787	30.39 ± 0.01	36.17 ± 0.01

UNet

YOLO – Loss Function

Standard YOLOv11-Pose Loss: $\mathscr{L}_{\text{total}} = 7.5\,\mathscr{L}_{\text{box}} + 12.0\,\mathscr{L}_{\text{pose}} + 2.0\,\mathscr{L}_{\text{obj}} + 0.5\,\mathscr{L}_{\text{cls}} + 1.5\,\mathscr{L}_{\text{dfl}}$

RICH YOLOv11-Pose Loss: $\mathscr{L}_{\text{total}} = 3.0\,\mathscr{L}_{\text{box}} + 25.0\,\mathscr{L}_{\text{pose}} + 6.0\,\mathscr{L}_{\text{obj}} + 0.5\,\mathscr{L}_{\text{cls}} + 0.6\,\mathscr{L}_{\text{dfl}}$

Box Loss: $\mathscr{L}_{\text{box}} = 1 - \text{IoU} + \frac{\rho^2(B_p, B_{gt})}{d^2} + \alpha v$ where ρ is the box center distance, d the diagonal that encloses both boxes and αv measures the difference between the box ratio

Distribution Focal Loss: refines the prediction of bounding box coordinates by avoiding direct rounding to discrete pixels

Classification Loss: based on Cross Entropy, it is useless since the class is unique

Pose Loss: loss function related to keypoint detection, uses Mean Absolute Error (L1) and Mean Squared Error (L2)

$$\mathscr{L}_{\text{pose}} = \frac{1}{K} \sum_{k=1}^{K} v_k \, \text{SmoothL1}(P_k, T_k) \qquad \text{SmoothL1}(P,T) = \begin{cases} 0.5 \cdot \text{L2}(P,T) & \text{if } \text{L1} < \beta \\ \text{L1}(P,T) - 0.5 \cdot \beta & \text{otherwise} \end{cases}$$

where $\text{L1}(P,T) = |x_P - x_T| + |y_P - y_T|$ and $\text{L2}(P,T) = (x_P - x_T)^2 + (y_P - y_T)^2$

Objectness Loss: measures the confidence of the model that an object is present in a given grid cell. The loss is computed over three different grids and it is based on Binary Cross Entropy:

$$\mathscr{L}_{\text{obj}} = \frac{1}{N_{cells}} \sum_{i=1}^{N_{cells}} \text{BCE}(p_{obj}^{(i)}, y_{obj}^{(i)})$$

where $\text{BCE}(p,y) = -[y \log p + (1-y) \log(1-p)]$

UNet – Loss Function

UNet Total Loss: $\mathscr{L} = 4.0\,\mathscr{L}_{\text{BCE}} + 3.0\,\mathscr{L}_{\text{SL1}} + 3.0\,\mathscr{L}_{\text{Dice}}$

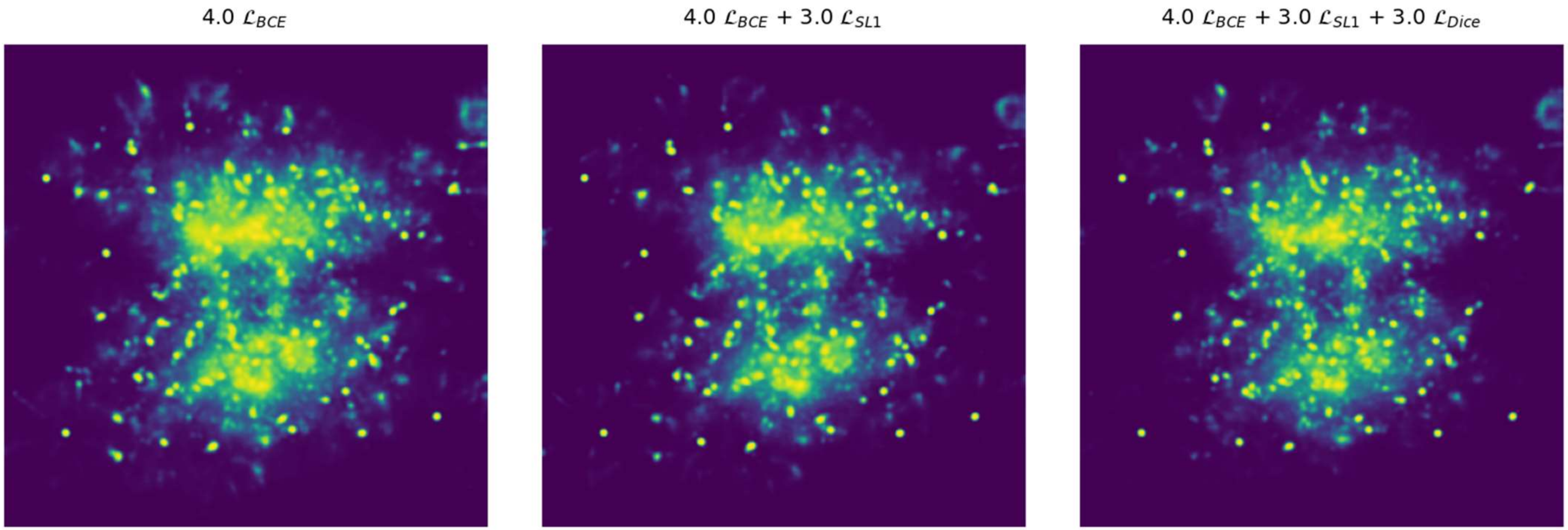
Binary Cross Entropy with Logits Unit: it is a numerically stable version of BCE which involves logits z

$$\mathscr{L}_{\text{BCE}} = \frac{1}{N}\sum_{i=1}^{N}\left[\left(1+(w_{\text{pos}}-1)y_i\right)\max(z_i,0) - y_i w_{\text{pos}} z_i + \left(1+(w_{\text{pos}}-1)y_i\right)\log\left(1+e^{-|z_i|}\right)\right]$$

SmoothL1 Loss: $\mathscr{L}_{SL1}$ same as YOLO Pose Loss

Dice Loss: measure the overlap between predicted P and ground truth G images

$$\mathscr{L}_{\text{Dice}} = \frac{1}{N}\sum_{j=1}^{N}\left(1-\text{Dice}(P^{(j)},G^{(j)})\right) \qquad \text{where} \qquad \text{Dice}(P^{(j)},G^{(j)}) = \frac{2\sum_i P_i^{(j)} G_i^{(j)}}{\sum_i P_i^{(j)} + \sum_i G_i^{(j)}}$$



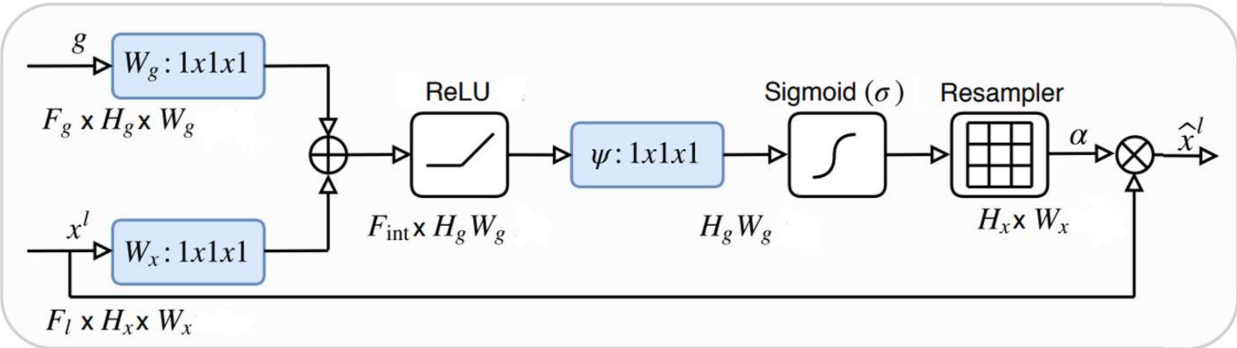
Example of the same inferred image obtained with UNet models trained using different loss functions on the same dataset. The BCE loss captures the overall heatmap, while SL1 and Dice losses refine the predictions by improving precision and suppressing background noise

Soft Attention Gates

The **Attention Mechanism** enhances neural networks by allowing them to selectively **focus on the most informative parts** of the input features while suppressing less relevant regions

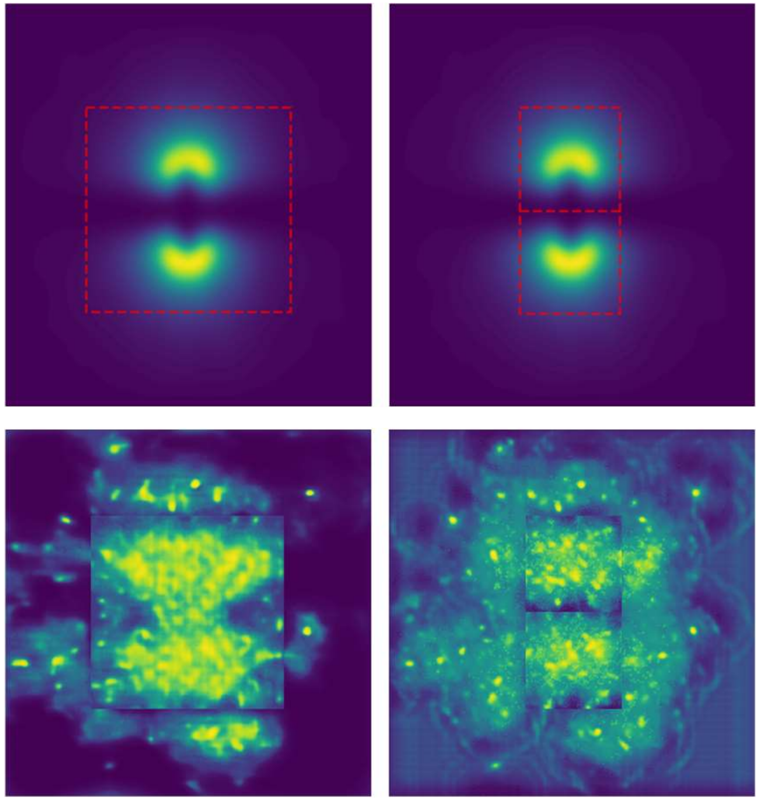
Soft Attention Gates:

- **Weight the feature maps from skip connections**, highlighting the regions most relevant for accurate reconstruction in the decoder
- The mechanism is **differentiable** and the weights are optimized through backpropagation



Patch-based Hard Attention

- **Patch-based Hard Attention:** the image is divided into fixed subregions processed independently
- **Training:** separate losses are computed for each patch and combined with predefined weights, enabling the network to learn distinct representations for each region
- **Motivation:** this strategy is a natural choice given the high ring density in the central parts of the images
- **Outcome:** after extensive experimentation, this approach proved to be ineffective

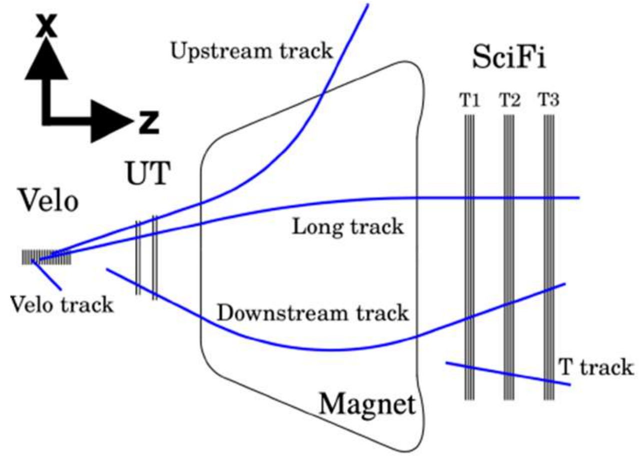


Particle Tracking and Identification

Particle identification (PID) with Cherenkov **rings relies on tracking system**

$$\cos\theta_C = \frac{1}{\beta n} \quad\Longrightarrow\quad \beta = \frac{pc}{\sqrt{(pc)^2 + (mc^2)^2}} \quad\Longrightarrow\quad m = \frac{p}{c}\sqrt{n^2\cos^2\theta_C - 1}$$

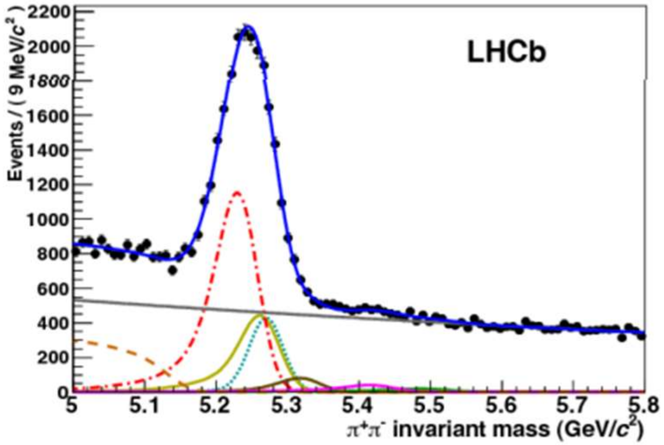
The **momentum** of charged particles is **reconstructed** directly **by fitting** their trajectories through the tracking detectors and the magnetic field in the first-stage trigger HLT1
However, **the mass formula cannot be applied directly in the harsh conditions of a real experiment** due to finite detector resolution, multiple scattering and material interaction, magnetic field inhomogeneities, and background and spurious photon hits

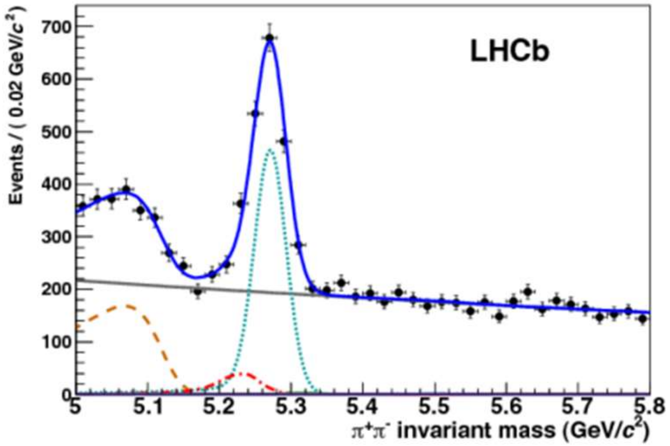


RICH PID

RICH Particle Identification (PID) is computed in HLT2:

1. **Cherenkov hits measurement**: each charged particle produces *N* photons, whose angular distribution depends on particle type and momentum (provided by HLT1)
2. **Ring selections**: only rings with associated centers from HLT1 tracking are considered
3. **Likelihood calculation**: for each particle hypothesis *h*, a likelihood comparing photon angular positions with Monte Carlo expectations is computed
4. **Log-likelihood differences (DLL):** final identification is based on the difference of log-likelihoods between hypothesis. Larger DLL means that the track is more compatible with one particle type than the other





PID removes spurious events, highlighting those of interest: total events are shown in the top plot, while the selected relevant events appear in the bottom plot