Generative models uncertainty estimation

Lucio Anderlini[1], Constantine Chimpoesh[2], Nikita Kazeev[2,3], Agata Shishigina[2]
on behalf of LHCb Simulation Project
[1]*Universita e INFN, Firenze*
[2]*HSE University*
[3]*National University of Singapore, Institute for Functional Intelligent Materials (I-FIM)*

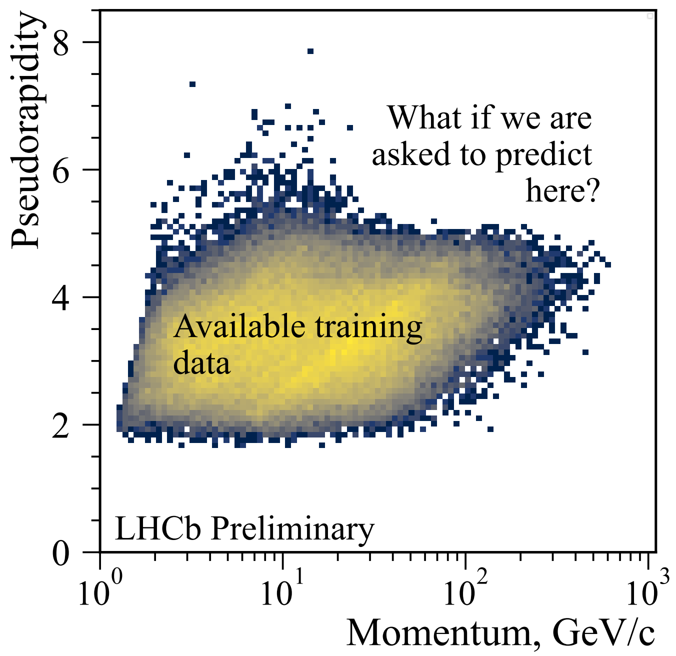
Learning to Discover, 29/04/2022, Institut Pascal Paris-Saclay

[Physics intro] Generative models are used for fast detector simulation and they are not perfect

- Training on MC → max quality is same as MC, needs a lot of CPU
- Training on calibration samples → bias from calibration data selection, parametrisation, limited coverage
- On any data, an ML model is just a heuristic fit

The question we answer:

For a given input, is the model usable?



[Generative models uncertainty estimation, ACAT-2021](#)

ML intro

- Uncertainty for Conditional GANs
- Same goal as for classification & regression, but more complex
  - Model quality can only be measured on samples
    - And even then, comparing two multidimensional distributions is difficult
  - Formalism: uncertainty of a regression is a generative model (pdf), uncertainty of a generative model is a distribution in the function space

Plan

- Model problem (LHCb RICH)
- Methods of uncertainty estimation
- Ensemble Distillation
- Results

LHCb RICH GAN

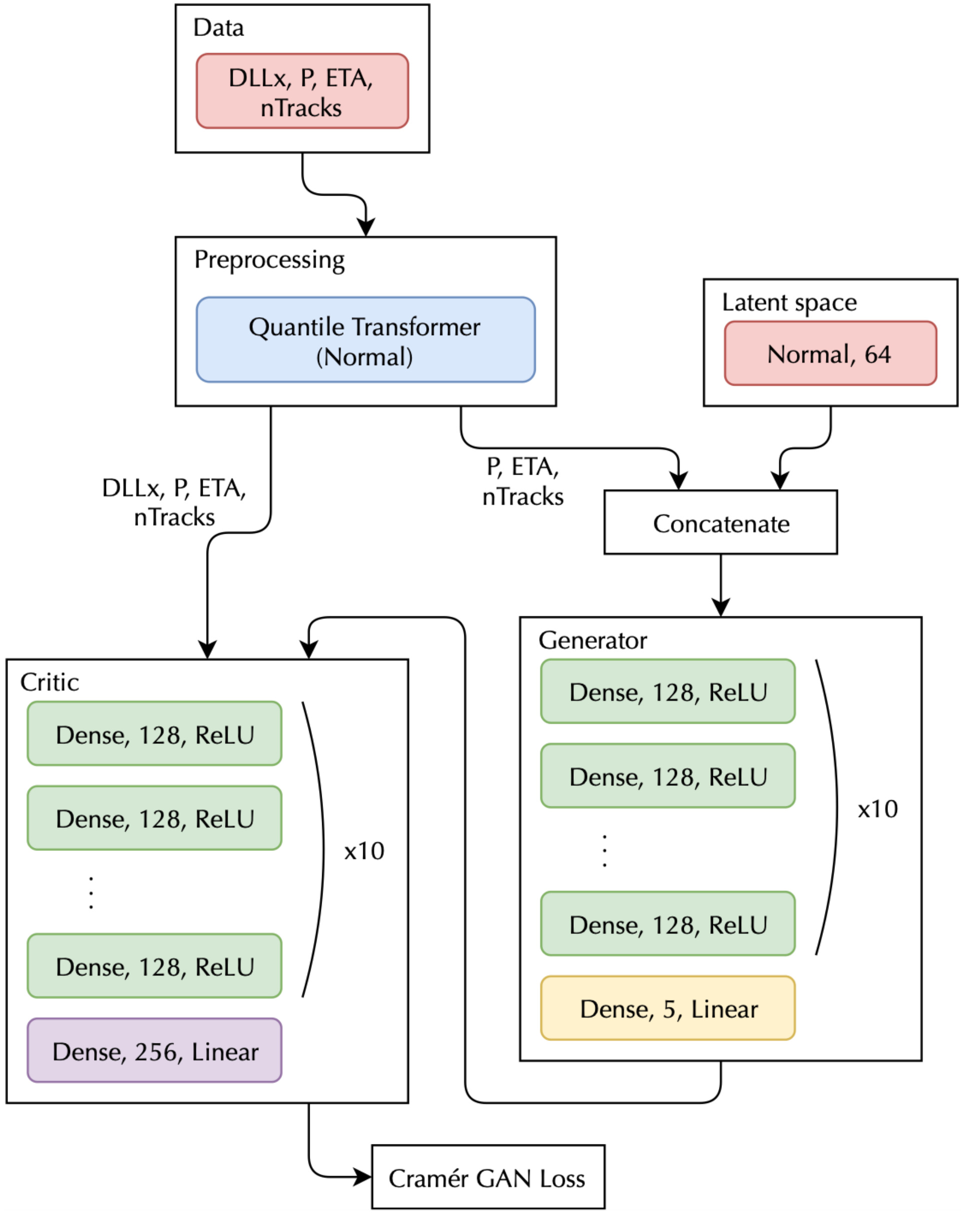
Input: $X \subset \mathbb{R}^3$

- Momentum `P`
- Pseudorapidity `ETA`
- Number of tracks `nTracks`

Output: $Y \subset \mathbb{R}^5$

- Particle type hypotheses delta log-likelihoods `RichDLLx`

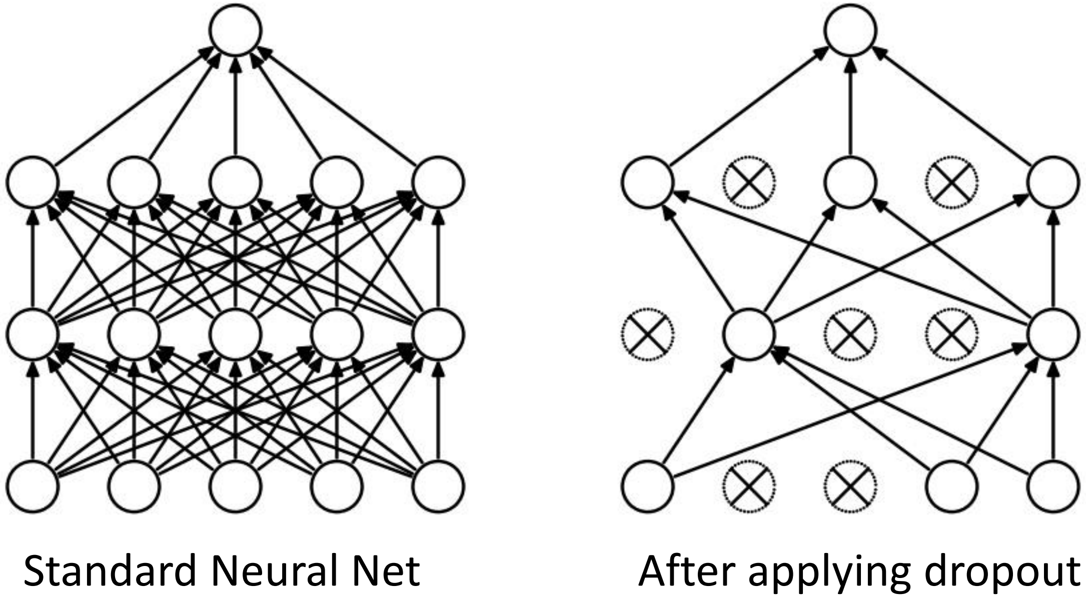
For track i, $x \in \{K, \mu, e, p, \text{below threshold}\}$
$\text{RichDLL} = \log \mathcal{L}(\text{type of } i \text{ is x}) - \log \mathcal{L}(\text{type of } i \text{ is } \pi)$



[Fast Data-Driven Simulation of Cherenkov Detectors Using Generative Adversarial Networks]

Recap: MC Dropout

- Dropout: Randomly drops units (along with their connections) from the neural network during training
- MC Dropout: dropout is applied both during training and inference



[Dropout: A Simple Way to Prevent Neural Networks from Overfitting]

[Our] Adversarial deep ensembles

Cramér GAN generator loss modification:

$$f(\boldsymbol{y}) = \left\|D(\boldsymbol{y}) - D(\boldsymbol{y}'_g)\right\|_2 - \|D(\boldsymbol{y})\|_2$$

Rewards the model for being different from the ensemble average

$$L_G = f(\boldsymbol{y}_r) - f(\boldsymbol{y}_g) - \alpha \left\|D(\boldsymbol{y}_g) - D(\boldsymbol{y}_{\cup_{\boldsymbol{g}}})\right\|_2$$

$\boldsymbol{y}_{\cup_{\boldsymbol{g}}}$ is a concatenation of the predictions of the ensemble, corresponding to a model with averaged probability density

[Our] Adversarial deep ensembles

$$L_G = f(\boldsymbol{y}_r) - f(\boldsymbol{y}_g) - \alpha \left\| D(\boldsymbol{y}_g) - D(\boldsymbol{y}_{\cup_g}) \right\|_2$$

Training schedule

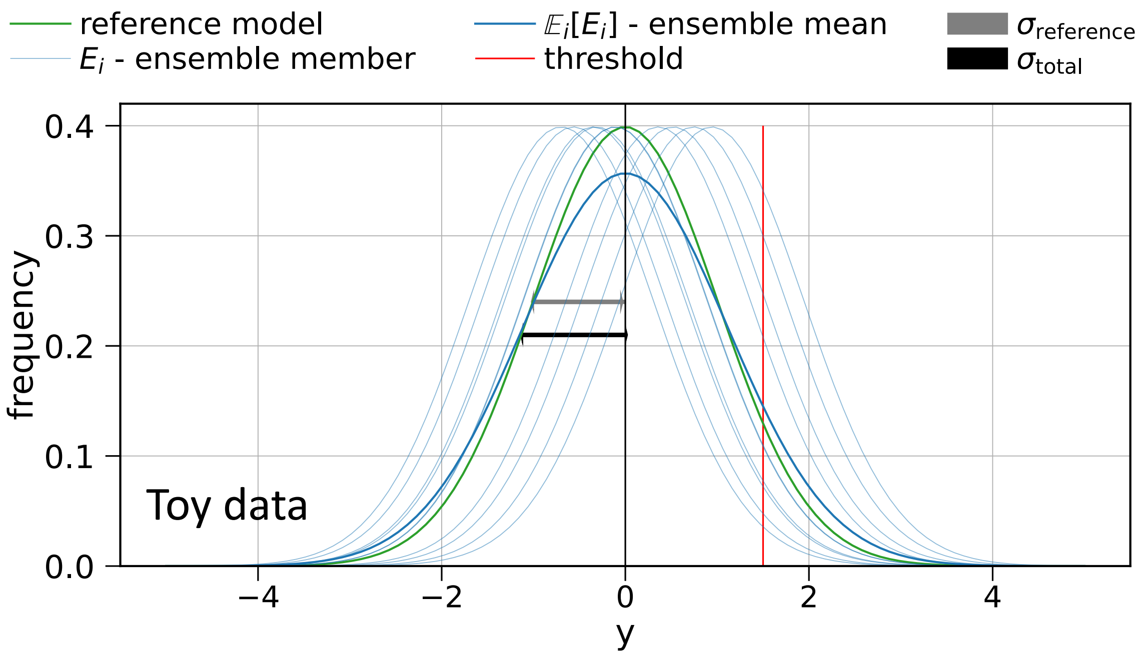
train Cramér GANs with the classic loss ($\alpha = 0$)

- retain the critics weights
- reinitialize the generators with random weights
- set $\alpha > 0$

- train both the generators and the critics with our loss
- decrease α according to a schedule

Ensemble distillation

- **Approximate a computationally complex ensemble with a single lightweight model during inference**
- Assumptions:
  - y is an output variable for track x, $y \sim \mathcal{N}(\mu(x),\ \ \sigma_{\text{reference}}\ (x))$
  - Ensemble has normally-distributed $\mu(x)$
  - $\sigma^2_{\text{reference}}$ - variance of distribution of y for the reference model
  - $\sigma^2_{\text{systematic}}$ - systematic uncertainty of the training procedure



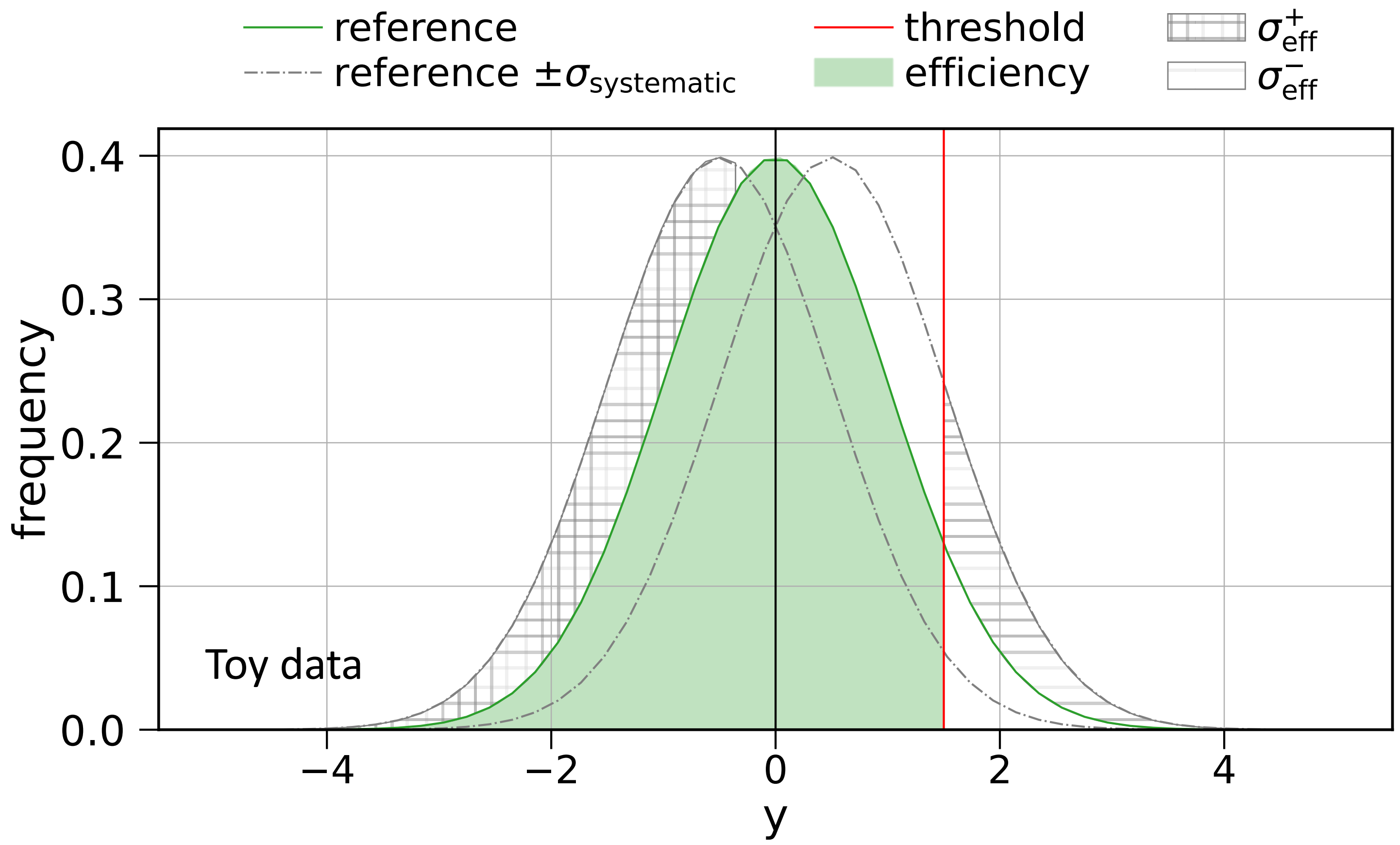
$$\sigma_{\text{total}} = \sqrt{\sigma^2_{\text{reference}} + \sigma^2_{\text{systematic}}}$$

Uncertainty estimation with(out) ensembles

$$\sigma_{\mathrm{systematic}} = \sqrt{\frac{1}{2}\left(\mathbb{E}_{\mathrm{ens}}[(y^{(1)} - y^{(2)})^2] - \mathbb{E}_{\mathrm{ref}}[(y^{(1)} - y^{(2)})^2]\right)}$$

- $\mathbb{E}_{\mathrm{ens}}$ and $\mathbb{E}_{\mathrm{ref}}$ - the average operators computed across data produced by ensemble model or reference models
- $y^{(1)}$ and $y^{(2)}$ - independently sampled examples from the corresponding model
- Train a regression to approximate $\sigma_{\mathrm{systematic}}$ from the ensemble, thus allowing uncertainty computation with just a single model
- **The training doesn't use true labels, therefore is not restricted to the available data**

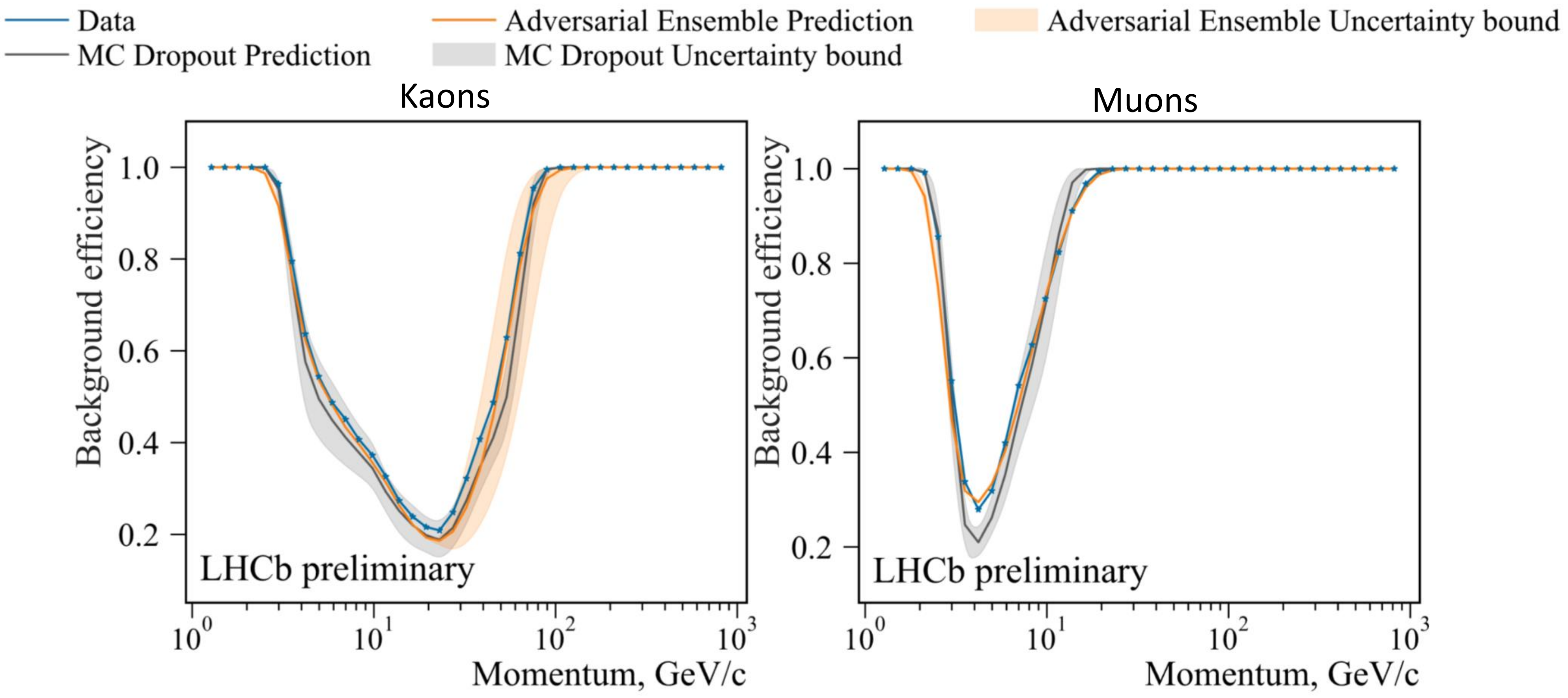
Computing efficiency uncertainty with $\sigma_{\text{systematic}}$



$$\text{Cut efficiency} = \text{refernce efficiency}^{+\sigma^{+}_{\text{eff}}}_{-\sigma^{-}_{\text{eff}}}$$

Efficiency with uncertainty, uniform train/test split

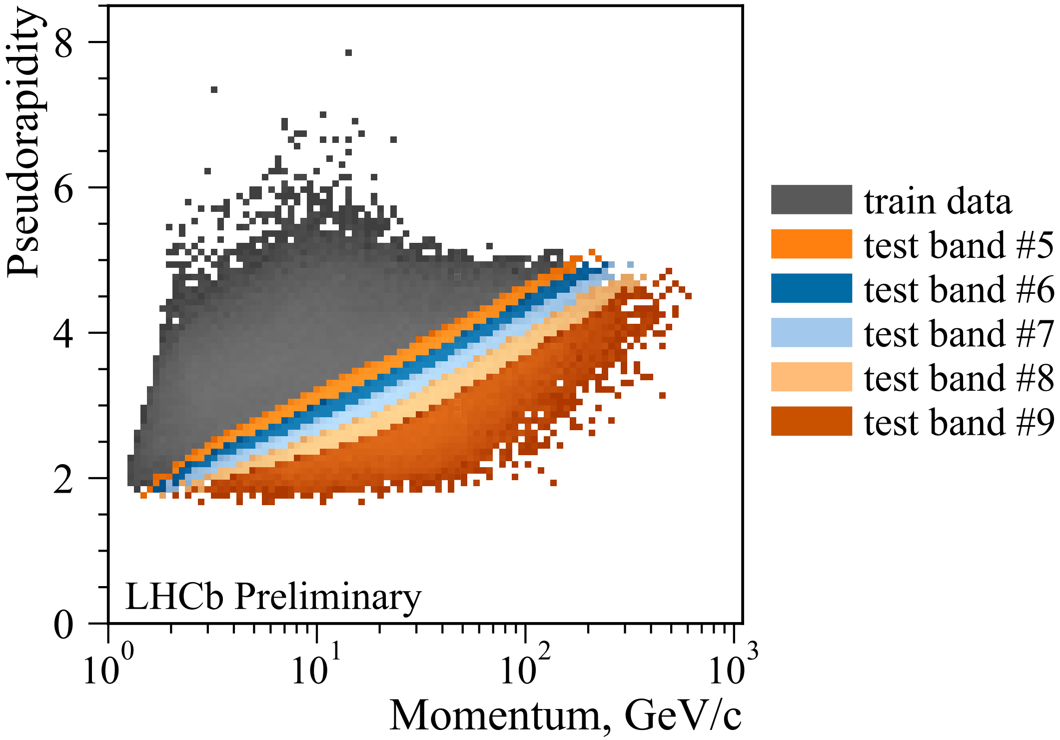
[Generative models uncertainty estimation, ACAT-2021]



Pion efficiency at 90% overall signal efficiency as a function of momentum. The data are uniformly split into training and testing parts

For most of the bins, efficiency on the test data lies inside the error bounds of the efficiency of the model

Extrapolation scan

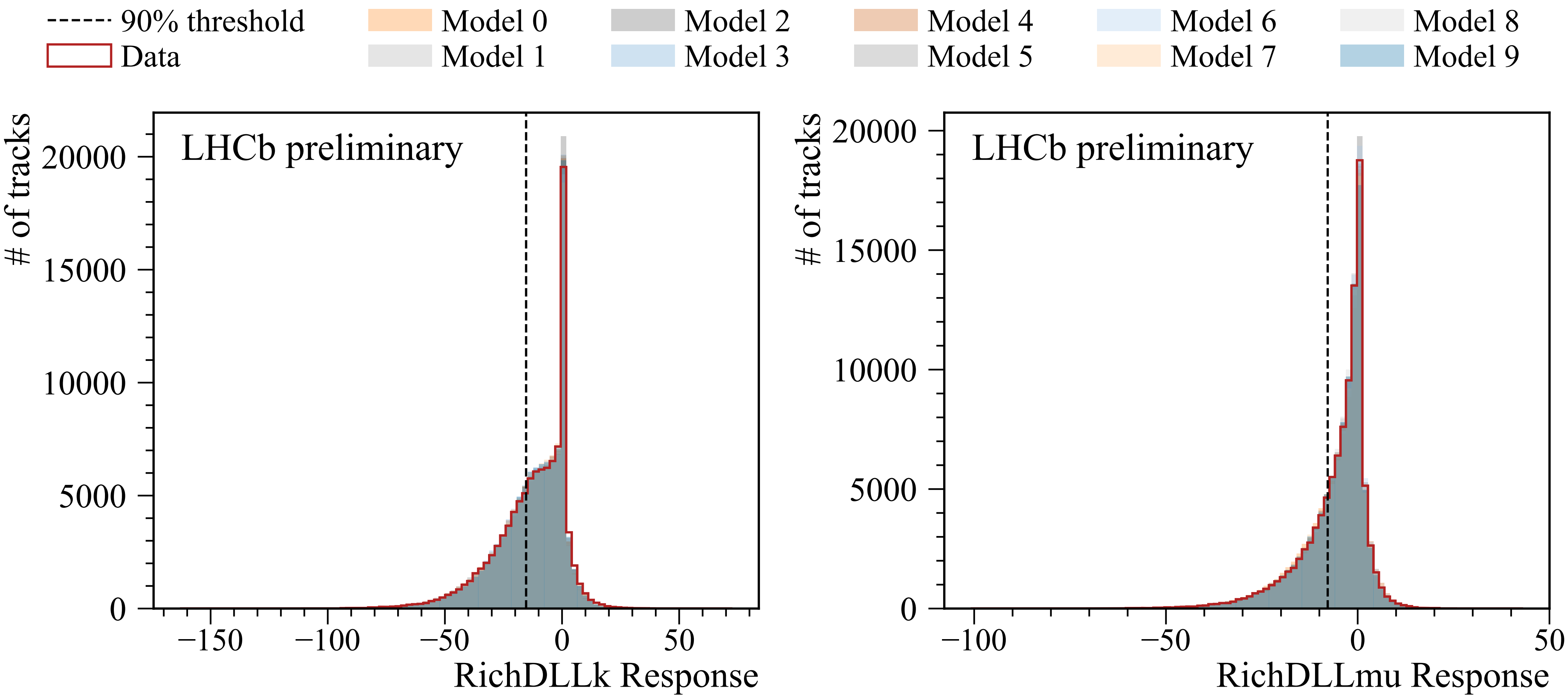


[Generative models uncertainty estimation, ACAT-2021]

- The dataset is split into train and test subsets by a line in equal proportions.
- Each subset is divided into bands, where one band contains the same number of samples

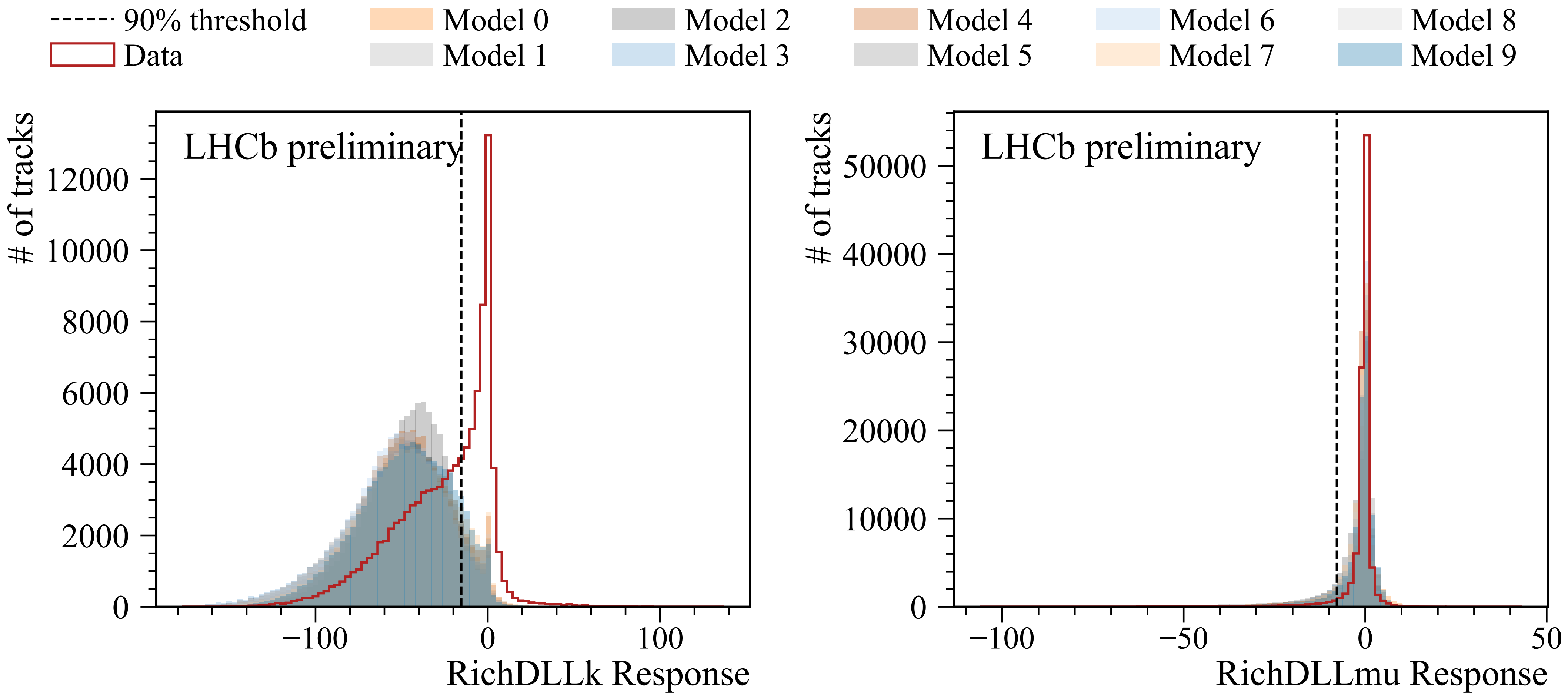
Distribution of the RichDLLs in a region with training data

Generative models uncertainty estimation, ACAT-2021



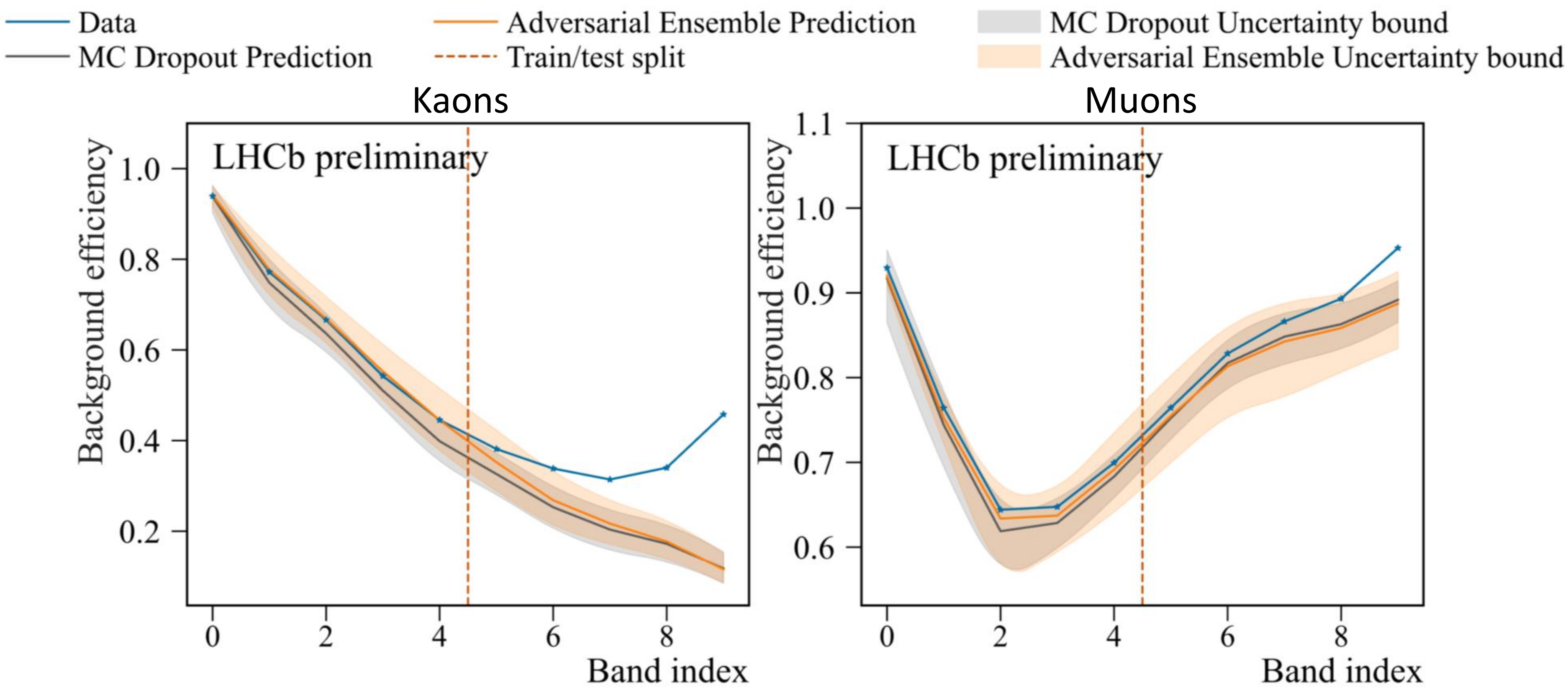
Distribution of the RichDLLs in a region without training data

Generative models uncertainty estimation, ACAT-2021



Efficiency with uncertainty, extrapolation scan

Generative models uncertainty estimation, ACAT-2021



Pion efficiency at 90% overall signal efficiency as a function of band index. Bands 0-4 are training parts 5-9 are testing parts

The uncertainly increases while getting further from the training region. However, the uncertainty does not increase sufficiently to account for the discrepancy in the furthest test regions

Instead of conclusion

- With some adaptation, ensemble and Bayesian methods can be used for generative model uncertainty estimation
- Ideally we want to get an uncertainty estimate to be plugged into an analysis
- Realistically, we are at the stage "can we use this model with these input data?"
- We propose an uncertainty estimation method, and evaluate it on LHCb RICH
  - For most of the bins, efficiency on the test data lies inside the error bounds of the efficiency of the model
  - In the extrapolation case, the uncertainly increases while getting further from the training region, but not sufficiently to account for the discrepancy in the furthest test regions

Thanks!

Backup

LHCb RICH fast simulation. Figure of merit

1. RichDLL common use is classification – tracks are filtered by a condition RichDLLx > threshold
2. We choose a threshold for RichDLLx so that 90% of tracks with type x pass it
3. Of course, not only particles of type x pass the selection, but there are also false positives
4. We plot the pion **efficiency**: the fraction of pions for which RichDLLx > threshold
5. Ideally, it should match for data and GAN