[2022.04.29]

Simulating the LHCb experiment with Generative Models

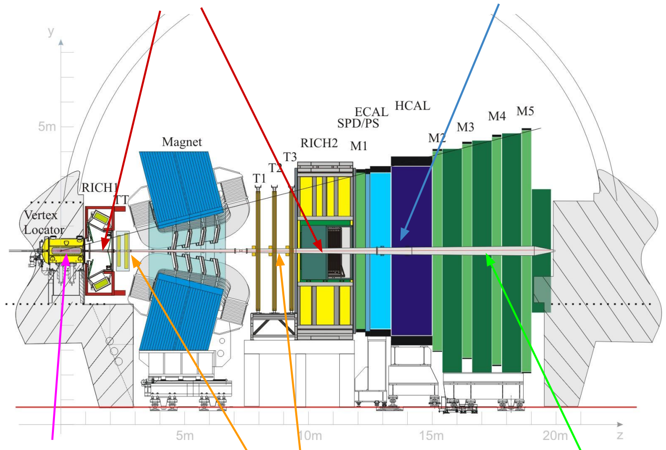
Matteo Barbetti

on behalf of the LHCb Simulation Project

University of Florence

INFN - Firenze

LHCb detector and its upgrades



Detector paper
[JINST 3 (2008) S08005]

Upgrade design
[LHCb-TDR-12]

The **Upgrade I** of the LHCb experiment is currently in commissioning. What's new?

- replacement of readout electronics
- new **fully software** trigger system

The new detector will be able to collect datasets at least **one order of magnitude larger** thanks to an increased instantaneous luminosity (x5) and a more performant selection algorithm (x2).

To match the increase of collected data, **larger** simulated samples and a strategy to **speed-up** their production is **unavoidable**.

fully software trigger system

x 5 instantaneous luminosity

x 2 selection efficiency

x 10 data sample size

Speed-up the simulation production

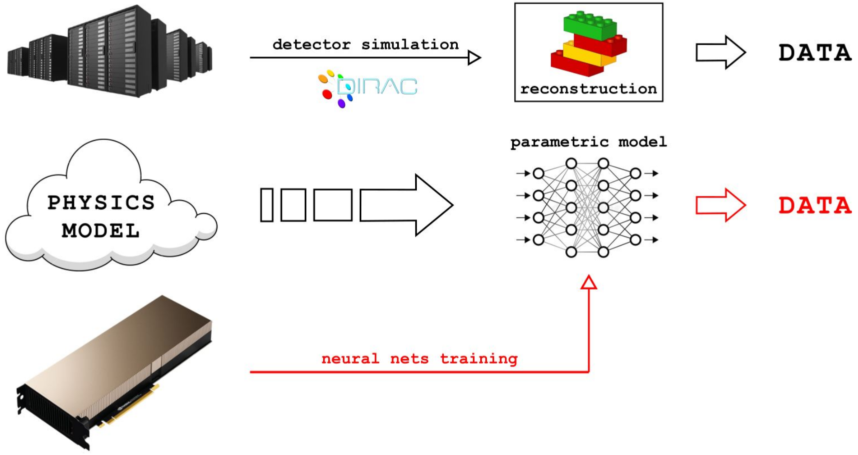
Geant4 consumes the **vast majority** of the *computing resources* intended for simulation

The CPU-greedy **Detailed Simulation** needs *faster solutions* for a large fraction of the samples

Ultra-Fast Simulation skips the simulation of the physics and reconstruction, reducing the computational cost

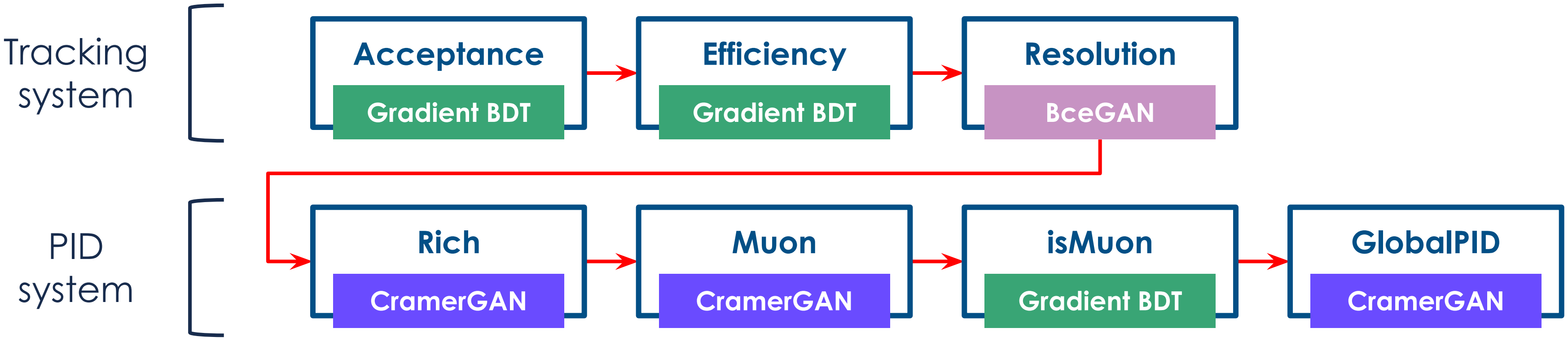
Encoding the *high-level response* of the detector directly within **statistical models**

- Such statistical models can be implemented with neural nets
- Training can be carried out in *adversarial configuration* (namely with GANs)
- **Detection and reconstruction processes** are encoded within neural nets



Ultra-Fast Simulation at LHCb

- In LHCb, Ultra-Fast parameterizations rely on **Gradient Boosted Decision Trees** for *efficiency models* and on **GAN-based neural nets** to model the detector response.
- Where possible, the models are trained directly on **real data**, otherwise they rely on **Detailed Simulation**
- The models are developed to be *separated entities* to be used either as single blocks within the Detailed Simulation or pipelined into a consistent **purely-parametric complete simulation**



Geometrical acceptance

`model  :` **Gradient Boosted Decision Tree**

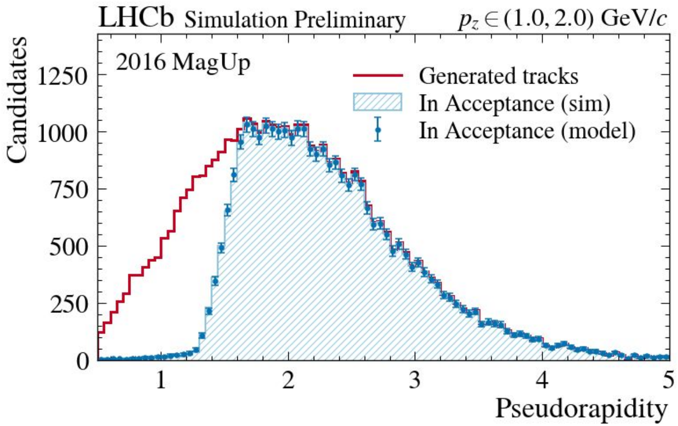
`loss   :` Binary Cross Entropy

`input  :` position and slope of tracks

`output :` in acceptance [True , False]

Training performed on **Detailed Simulation**

The GBDT model well-reproduces the Detailed Simulation distribution of the generated tracks weighting by the **probability** of being in acceptance.



[LHCb-FIGURE-2022-004]

Tracking Models Training Repo

github.com/landerlini/lb-trksim-train

Tracking efficiency

`model` : **Gradient Boosted Decision Tree**

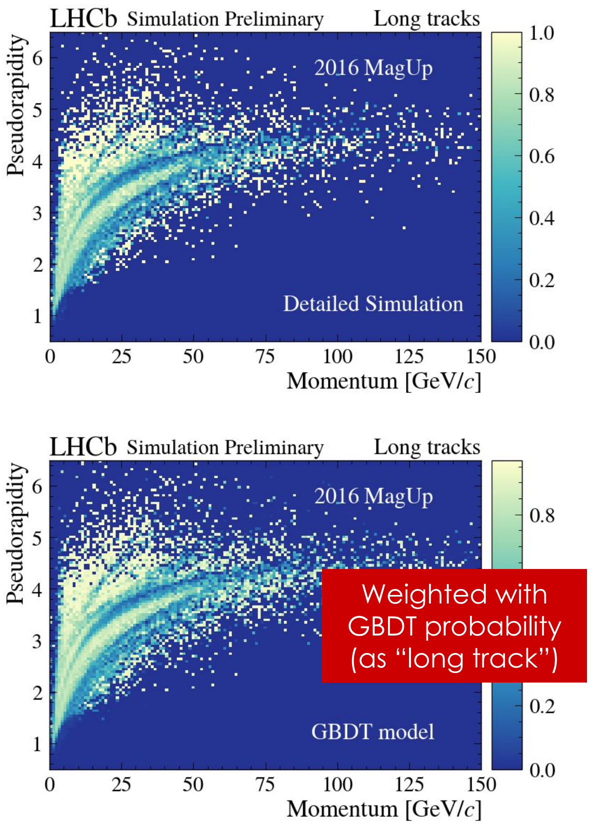
`loss` : Multi-class Cross Entropy

`input` : position and slope of tracks

`output` : track classification as [*long* , *upstream* , *downstream* , *non-reconstructed*]

Training performed on **Detailed Simulation**

The good performance of the GBDT model well-reproduces the **complex structure of shadows** describing the efficiency losses due to the non-trivial material sub-structure of the LHCb detector.



[LHCb-FIGURE-2022-004]

Tracking resolution

`model  :` **Generative Adversarial Networks**

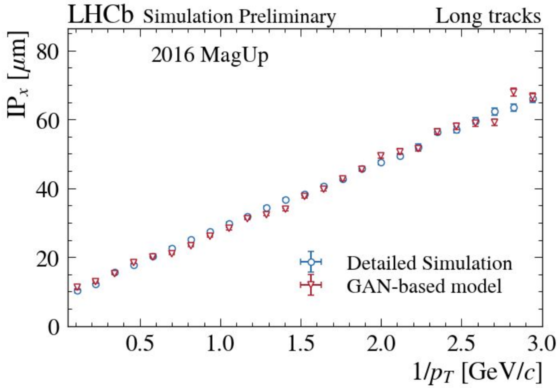
`loss   :` Binary Cross Entropy

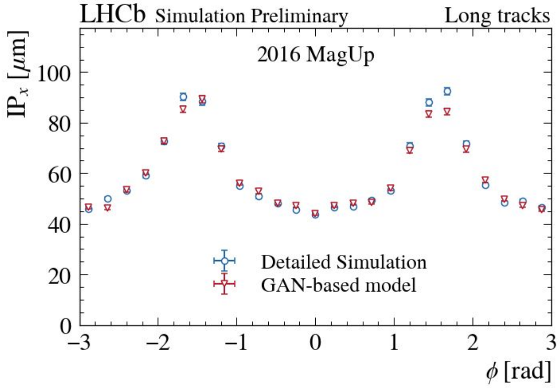
`input  :` position, slope and momentum of tracks

`output :` reconstructed tracks information

Training performed on **Detailed Simulation**

The x-projection of the *Impact Parameter* of tracks originated from the *Primary Vertex* is well-reproduced by the GAN-based model even if **neither the transverse momentum nor the phi angle are used for training**.





[LHCb-FIGURE-2022-004]

PID system: *training details*

To **overcome the typical issues** of GANs training, the parameterization of the LHCb Particle Identification system rely on **CramerGAN**: a stable, reliable and powerful GAN algorithm.

PID models are trained using **Calibration Samples** → Need for removing the **residual background**

- The CramerGANs are used to define **robust base models**, parameterizing both the signal and background components within the Calibration Samples
- The base models are then **fine-tuned** driven by either the *Binary Cross Entropy* or the *Wasserstein distance* as loss function
- The fine-tuning strategies are modified to **statistically subtract the background component** [JINST 14 (2019) P08020]

PID Models Training Repo

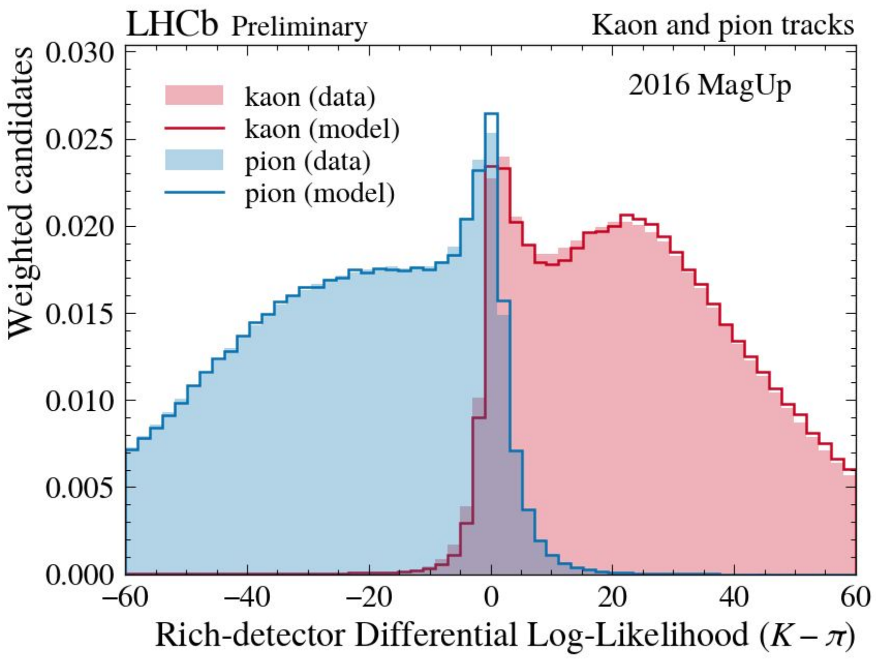
github.com/mbarbetti/lb-pidsim-train

GAN issues [arXiv:1701.04862]

CramerGAN [arXiv:1705.10743]

Calibration samples [LHCb-PUB-2016-020]

Rich detector: *kaon-pion separation*



[LHCb-FIGURE-2022-004]

`model` : **Generative Adversarial Networks**

`loss` : Energy distance (baseline) + BCE / Wasserstein distance (tuning)

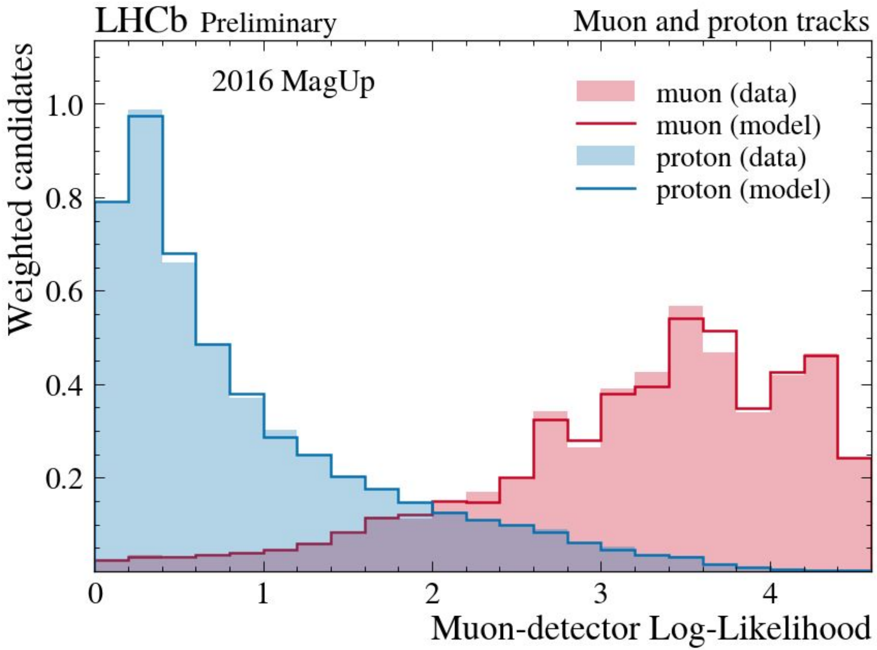
`input` : track kinematic parameters and detector occupancy

`output` : high-level response of the Rich detector

Training performed on **Calibration Samples**

2 neural networks trained in *adversarial configuration* are used to parameterize the **high-level response of the Rich detector** for kaon and pion tracks.

Muon detector: *muon-proton separation*



[LHCb-FIGURE-2022-004]

`model` : **Generative Adversarial Networks**

`loss` : Energy distance (baseline) + BCE / Wasserstein distance (tuning)

`input` : track kinematic parameters and detector occupancy

`output` : high-level response of the Muon detector

Training performed on **Calibration Samples**

2 neural networks trained in *adversarial configuration* are used to parameterize the **high-level response of the Muon detector** for muon and proton tracks.

isMuon criterion
[JINST 8 (2013) P10020]

Loose Binary Criterion: *isMuon*

`model  :` **Gradient Boosted Decision Tree**

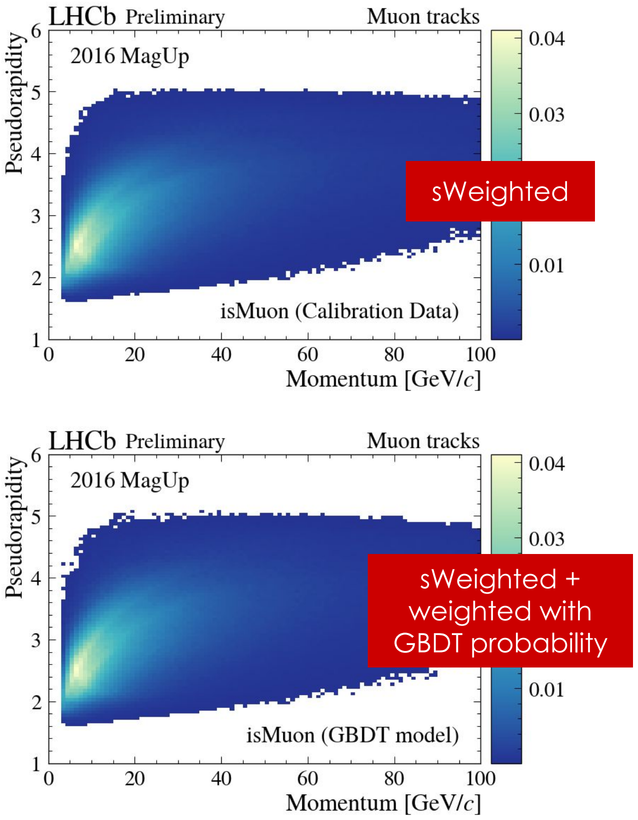
`loss   :` Binary Cross Entropy

`input  :` track kinematic parameters and detector occupancy

`output :` isMuon passed [True , False]

Training performed on **Calibration Samples**

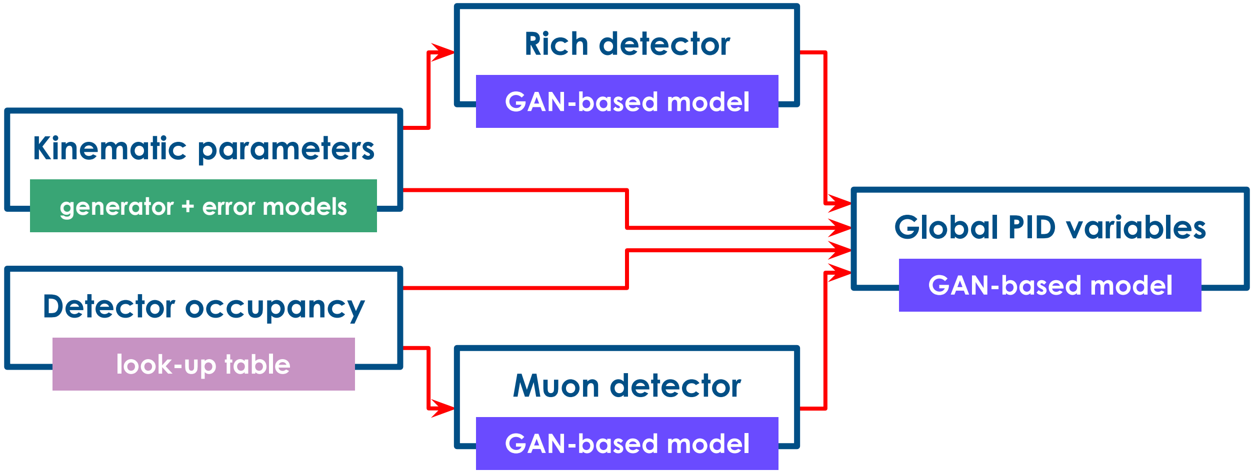
The **residual background** of Calibration Samples is subtracted when training the GBDT. The model well-reproduces the behaviour of the **isMuon criterion** on data.



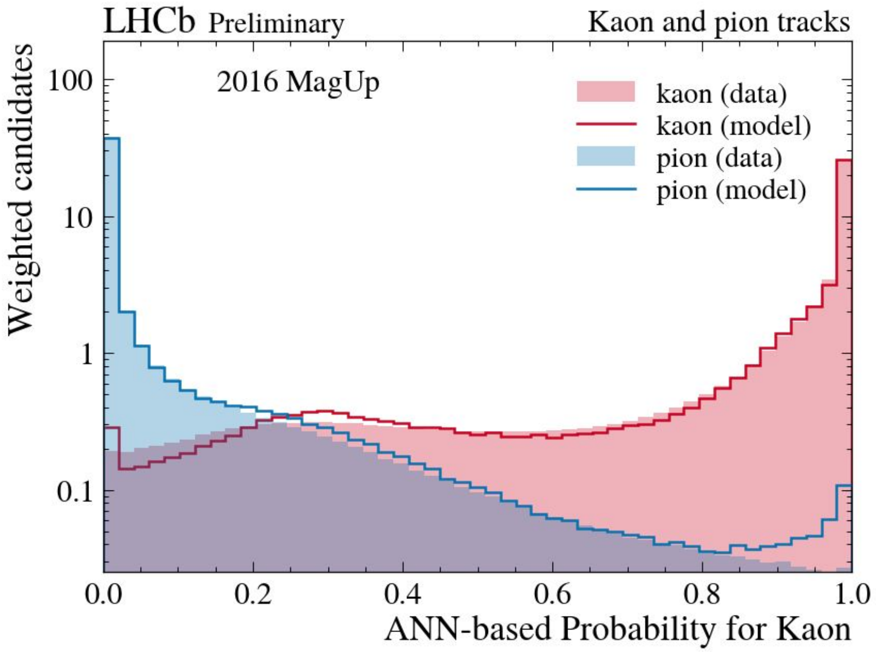
[LHCb-FIGURE-2022-004]

PID system: *stacking generative models*

- The kinematic parameters of the tracks and the detector occupancy information *aren't enough* to correctly parameterize the **Global PID variables**.
- Training a new set of neural networks fed by the **high-level response** of the Rich and Muon detectors allows to parameterize the Global PID variables that can be retrieved in the *inference phase* through a **stack of GANs**.
- The stack of GANs provides the **higher-level response** of the PID system.



Global PID: *kaon-pion separation*



[LHCb-FIGURE-2022-004]

`model` : **Generative Adversarial Networks**

`loss` : Energy distance (baseline) + BCE / Wasserstein distance (tuning)

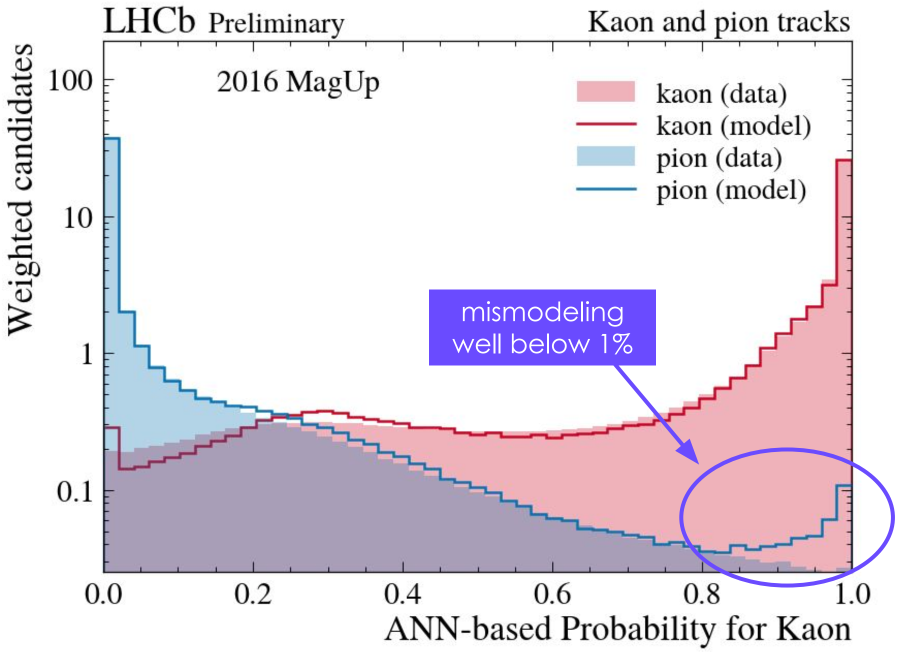
`input` : track kinematic parameters , detector occupancy , isMuon , high-level response of the Rich detector , high-level response of the Muon detector

`output` : Global PID variables

Training performed on **Calibration Samples**

2 neural networks trained in *adversarial configuration* are used to parameterize a global PID variable named **ProbNN** for kaon and pion tracks.

Global PID: *kaon-pion separation*



[LHCb-FIGURE-2022-004]

`model` : **Generative Adversarial Networks**

`loss` : Energy distance (baseline) +
BCE / Wasserstein distance (tuning)

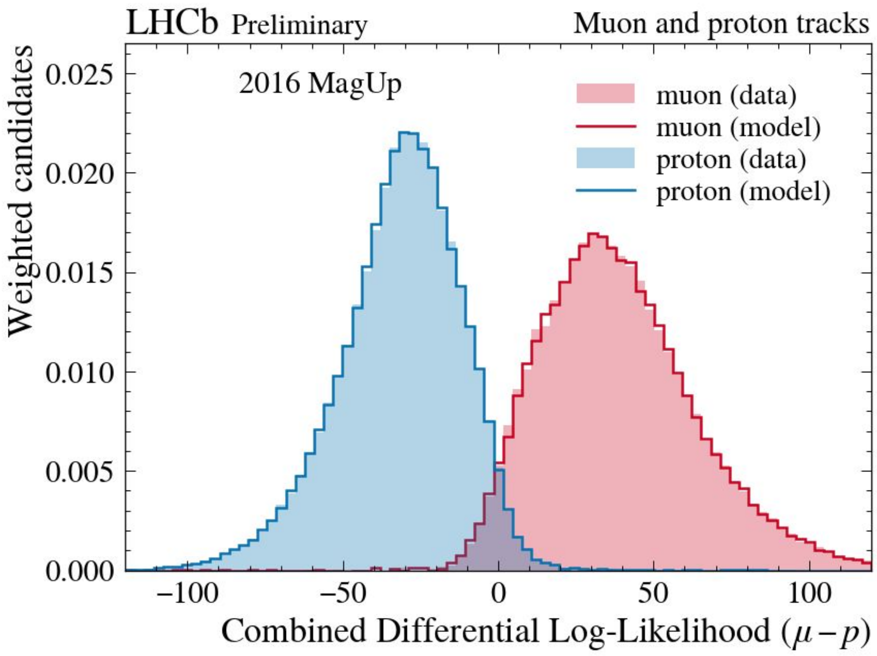
`input` : track kinematic parameters ,
detector occupancy , isMuon ,
high-level response of the Rich detector ,
high-level response of the Muon detector

`output` : Global PID variables

Training performed on **Calibration Samples**

2 neural networks trained in *adversarial configuration* are used to parameterize a global PID variable named **ProbNN** for kaon and pion tracks.

Global PID: *muon-proton separation*



[LHCb-FIGURE-2022-004]

`model` : **Generative Adversarial Networks**

`loss` : Energy distance (baseline) + BCE / Wasserstein distance (tuning)

`input` : track kinematic parameters , detector occupancy , isMuon , high-level response of the Rich detector , high-level response of the Muon detector

`output` : Global PID variables

Training performed on **Calibration Samples**

4 neural networks trained in *adversarial configuration* are used to parameterize various global PID variables shown together in the **Combined Differential Log-Likelihood** for muon versus proton hypothesis.

Summary and outlook

- With the start of Run 3, developing **faster solutions** to produce simulated samples is of key importance.
- The Ultra-Fast Simulation at LHCb consists of **modular components** that can be used as single blocks within the Detailed Simulation or pipelined into a consistent **purely-parametric complete simulation**.
- A **stack of GANs** can be used to effectively parameterize the higher-level response of the PID system.
- Once trained, the models can be integrated within the LHCb simulation software as **shared objects** or easily replaced with new ones [details].

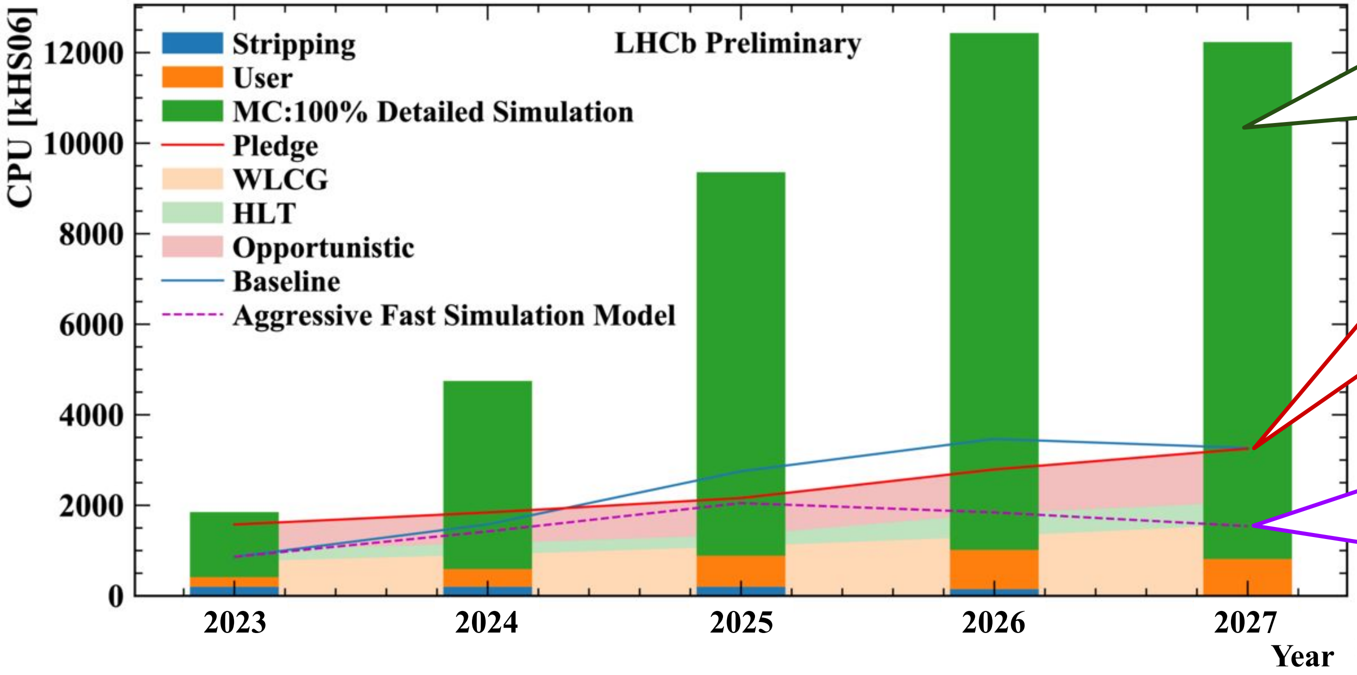
What's next?

- Models of the electromagnetic calorimeter (in progress)
  - shower libraries [EPJ Web Conf. 214 (2019) 02040] or Self-Attention GANs [EPJ Web Conf. 251 (2021) 03043] for the **low-level response**
  - a mixture of generative models and parametric functions for the **high-level response**
- Models for all current and future LHCb datasets

Backup

LHCb computing requirements for Run 3

Updated from [LHCb-FIGURE-2019-018]



CPU resources needed for a simulation scheme only including Geant4-based ***Detailed Simulation***

Available resources according to the planned **funding scheme**

Computing resources required by an excellent ***Fast*** and ***Ultra-Fast Simulation***

Long, upstream and downstream tracks

Upstream tracks

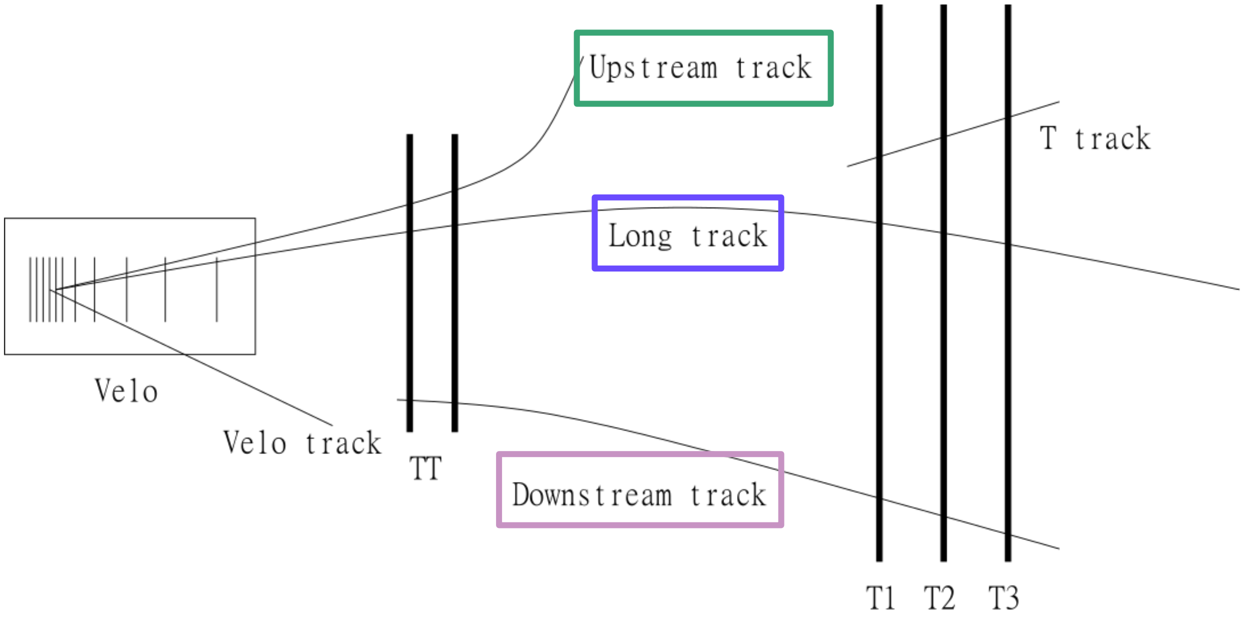
Tracks reconstructed by the tracking stations upstream the magnet.

Long tracks

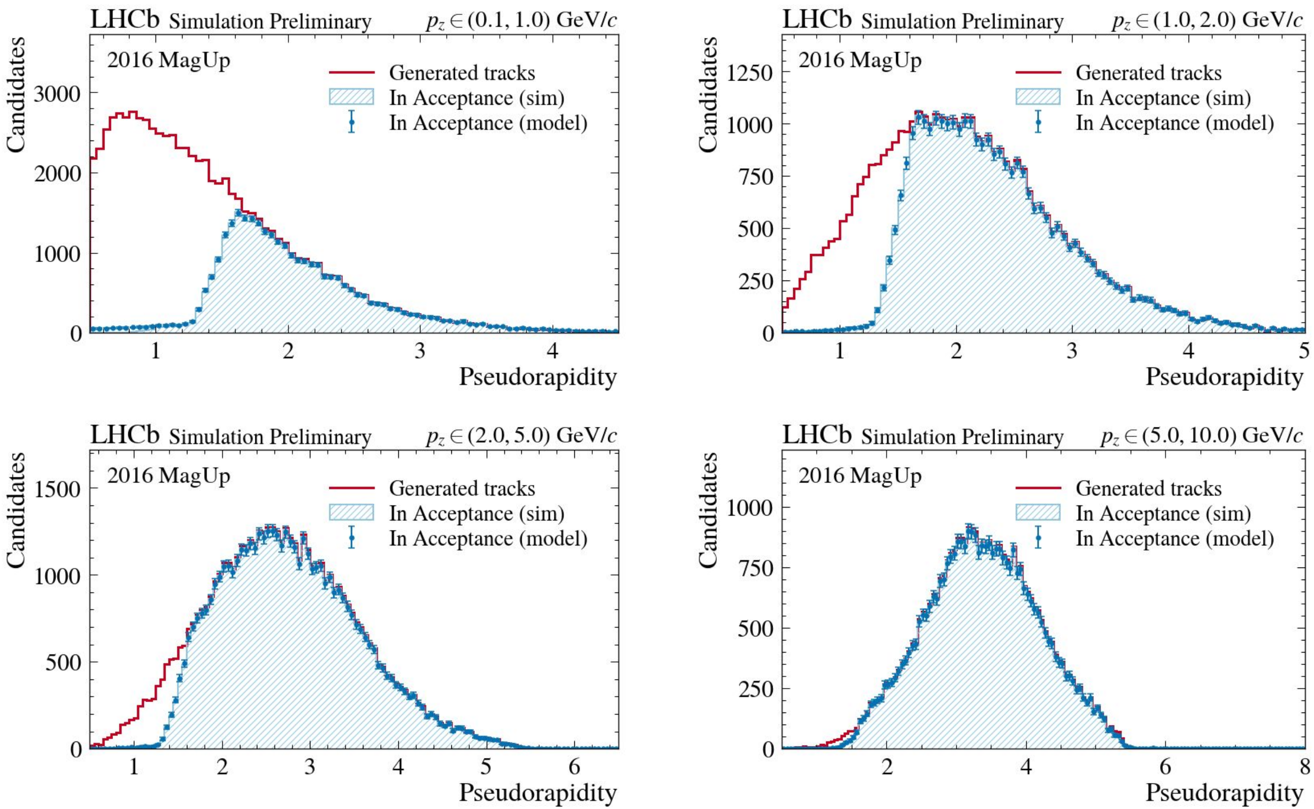
Tracks reconstructed after having traversed the whole detector.

Downstream tracks

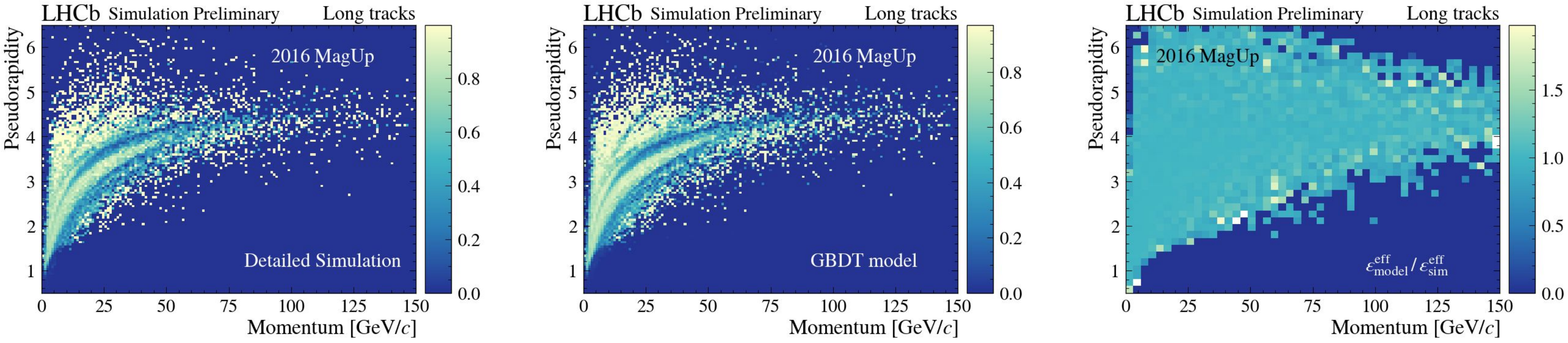
Tracks reconstructed by the tracking stations, but no hits found in the VELO.



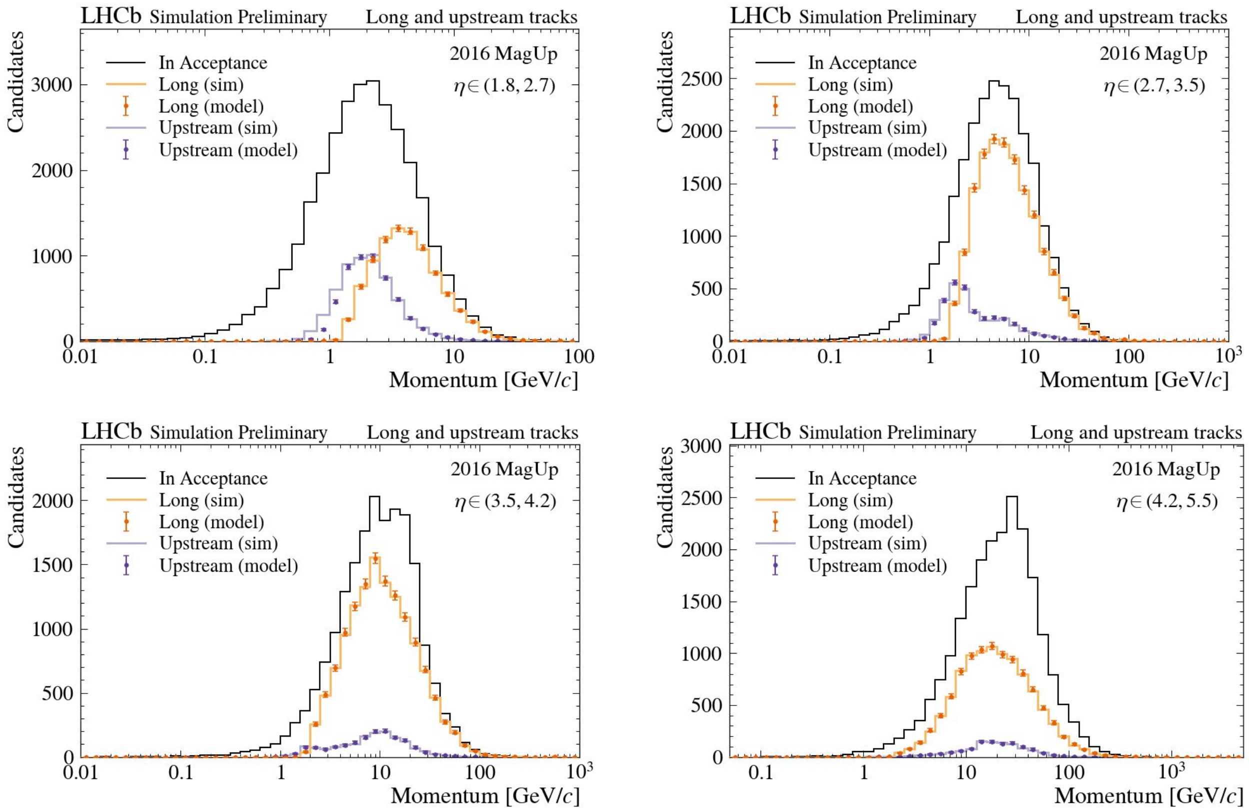
LHCb-FIGURE-2022-004: *Geometrical Acceptance*



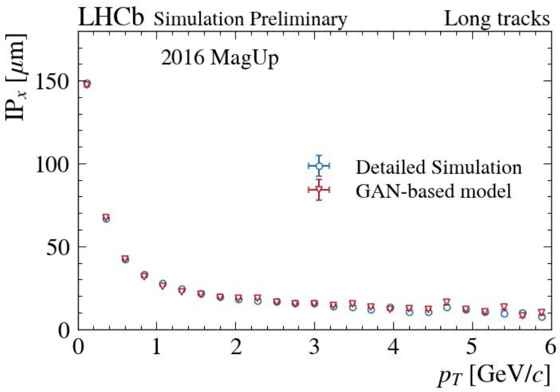
LHCb-FIGURE-2022-004: *Tracking Efficiency* (1/2)

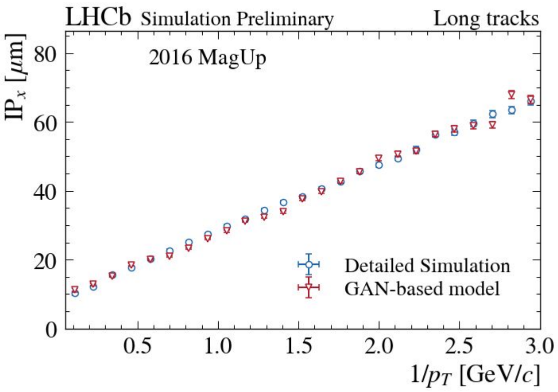


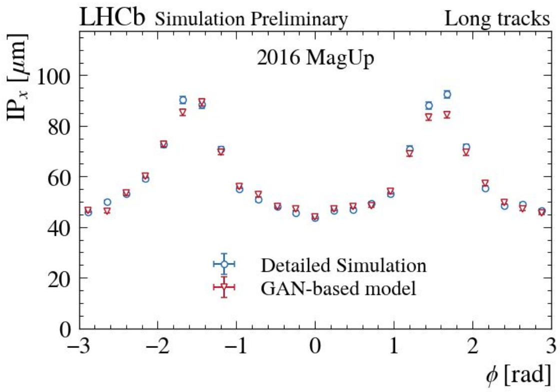
LHCb-FIGURE-2022-004: *Tracking Efficiency (2/2)*



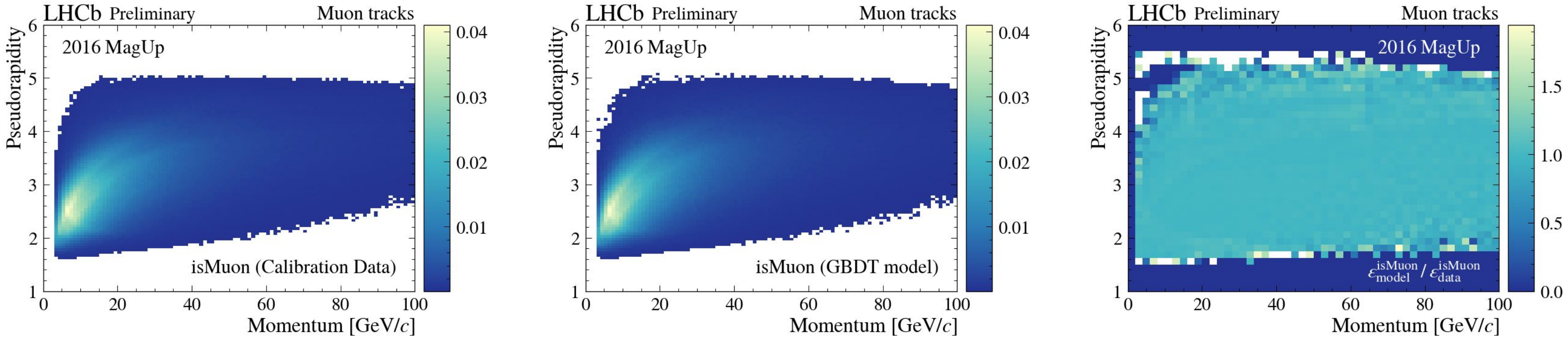
LHCb-FIGURE-2022-004: *Tracking Resolution*



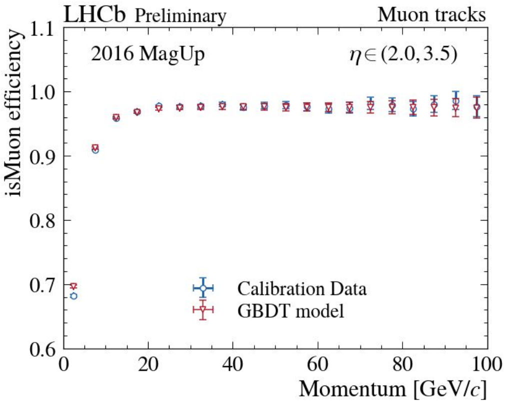


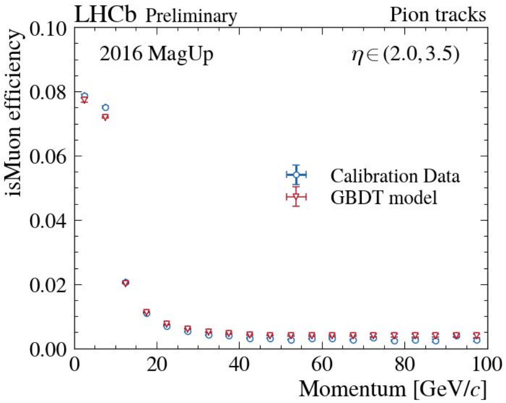


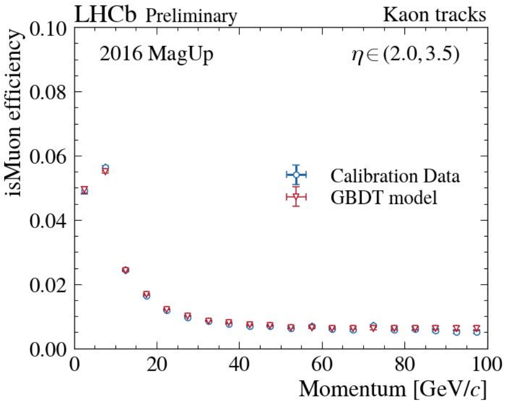
LHCb-FIGURE-2022-004: *isMuon* (1/2)

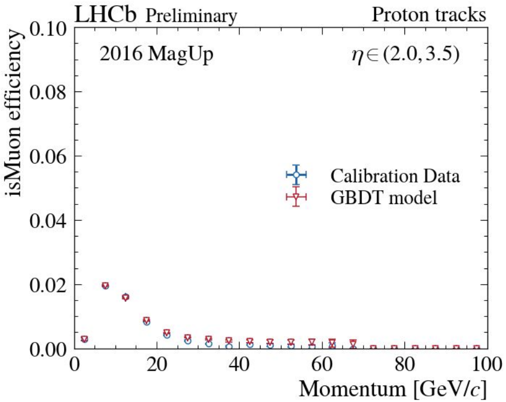


LHCb-FIGURE-2022-004: *isMuon* (2/2)

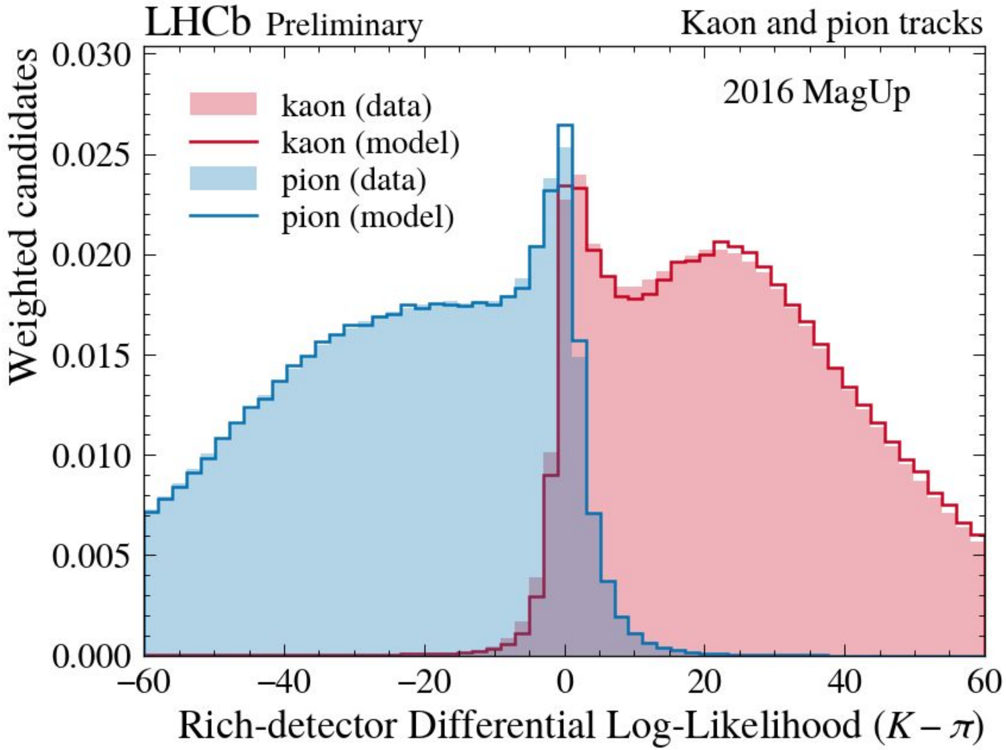




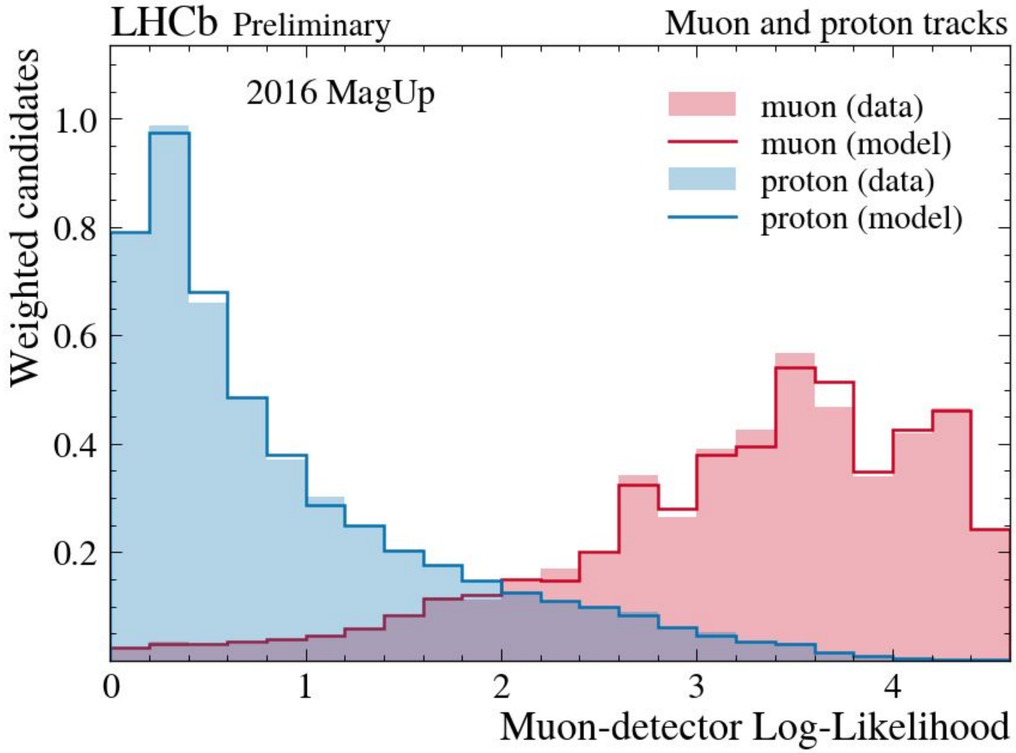




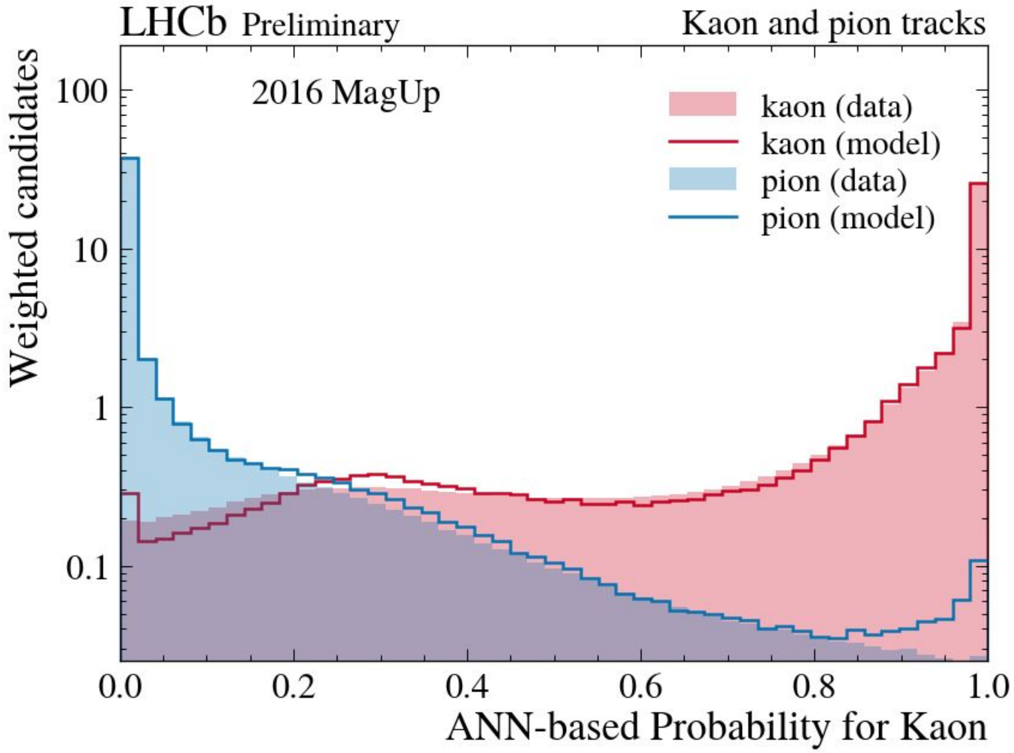
LHCb-FIGURE-2022-004: *Rich system*



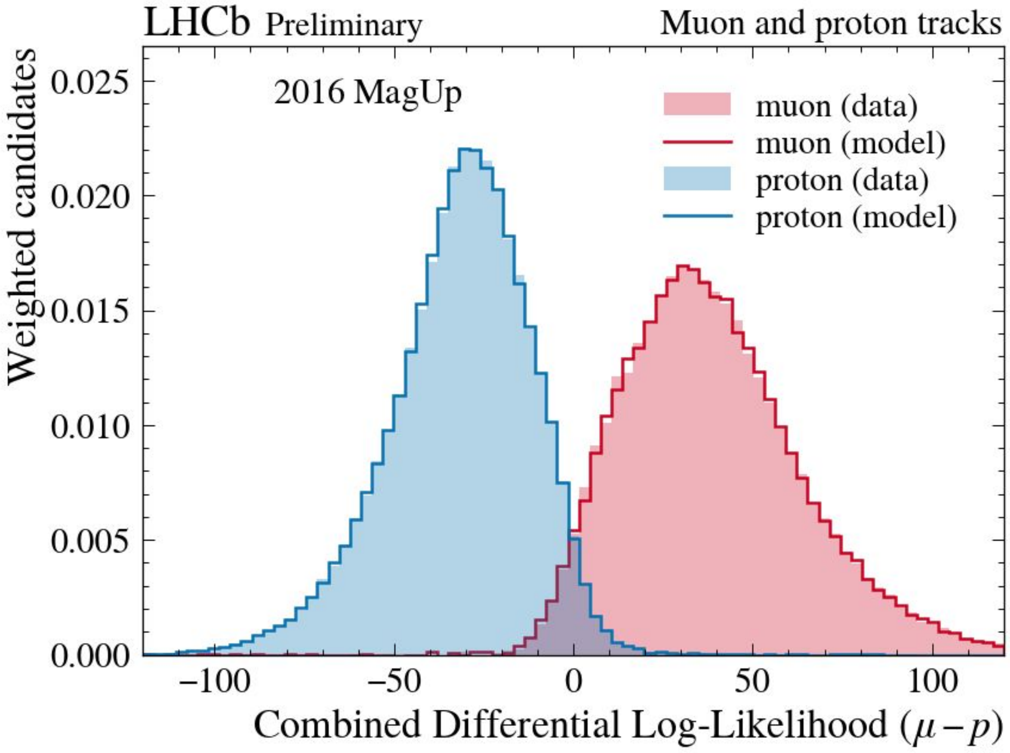
LHCb-FIGURE-2022-004: *Muon system*



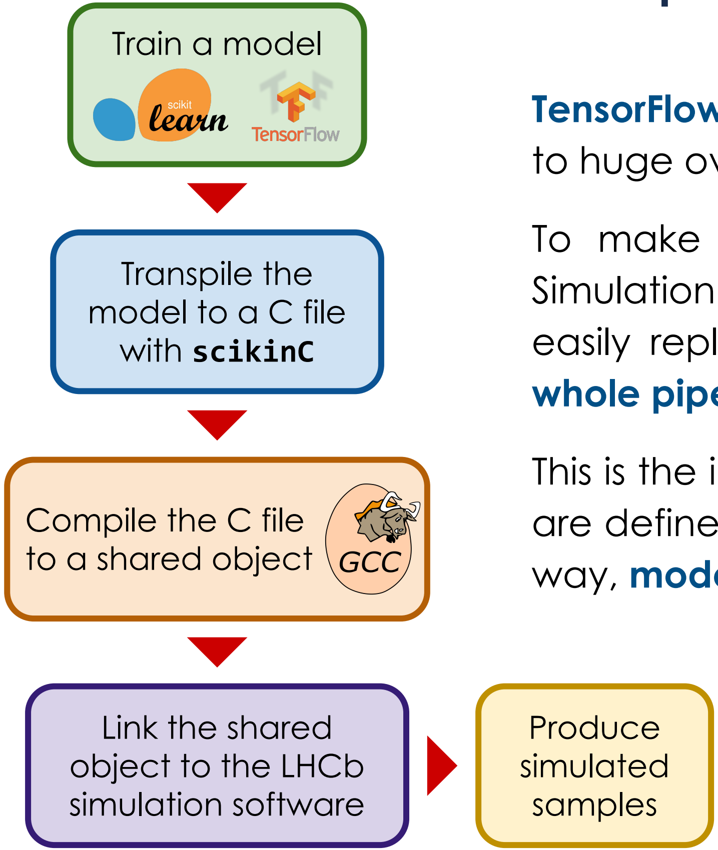
LHCb-FIGURE-2022-004: *Global Particle Identification*



LHCb-FIGURE-2022-004: *Global Muon Identification*



Deployment within the LHCb simulation software



TensorFlow and **ONNX** use their own thread scheduler that can lead to huge overhead for HEP applications.

To make the most from the **modular logic** of LHCb Ultra-Fast Simulation, we are interested in a deployment tool that allow to easily replace a specific parameterization, **without recompile the whole pipeline**.

This is the idea behind the **`scikinC`** tool where the ML-based models are defined to be dynamically linked to the main application. In this way, **models can be developed and released independently**.

`scikinC` tool
[presentation]

Deployment Tool Repo

github.com/landerlini/scikinC