Generative Models for the **ultra-fast simulation** of the **LHCb experiment**

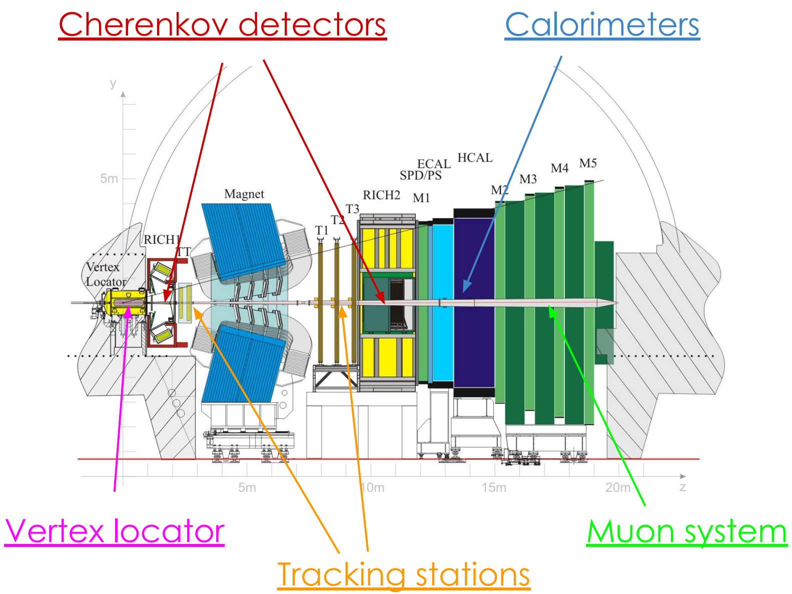
Lucio Anderlini

Istituto Nazionale di Fisica Nucleare – Sezione di Firenze

Istituto Nazionale di Fisica Nucleare
SEZIONE DI FIRENZE

Centro Nazionale di Ricerca in HPC,
Big Data and Quantum Computing

The LHCb experiment 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 **full 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**.

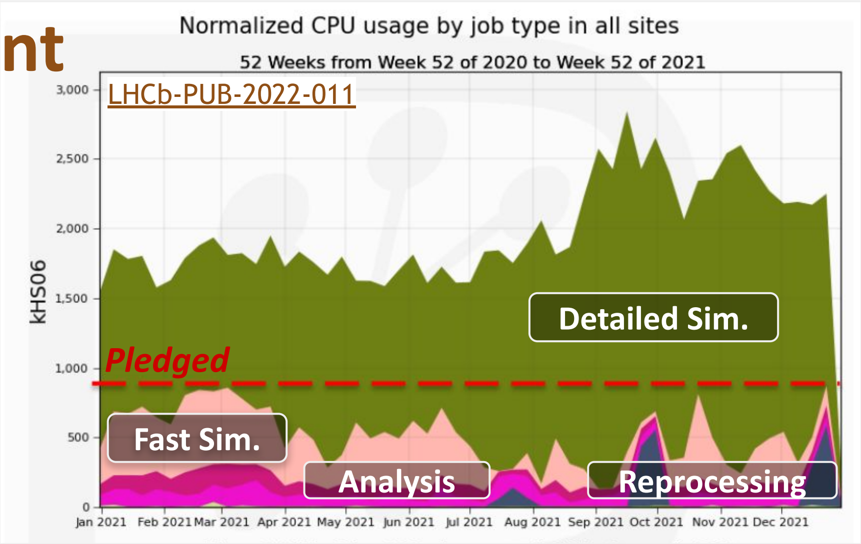
fully software trigger system

x 5 instantaneous luminosity

x 2 selection efficiency

x 10 data sample size

Simulating the LHCb experiment



Detailed

Centralized MC productions. Interactions of particles with detector material is simulated with Geant4, and converted into *hits*.

Same trigger & reconstruction algorithms are used as for real data.

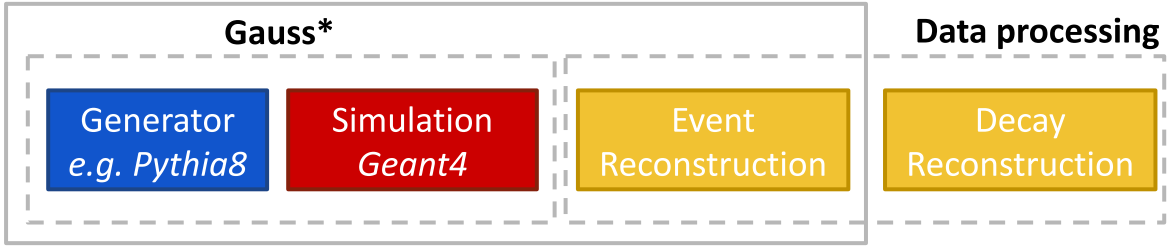
Fast Simulation

Replace parts of the simulation with models, e.*g.*

underlying event → ReDecay
[Eur. Phys. J. C 78 (2018) 1009]

calorimeter deposits → CaloGAN
[arXiv:1812.01319]

Detailed/Fast Simulation

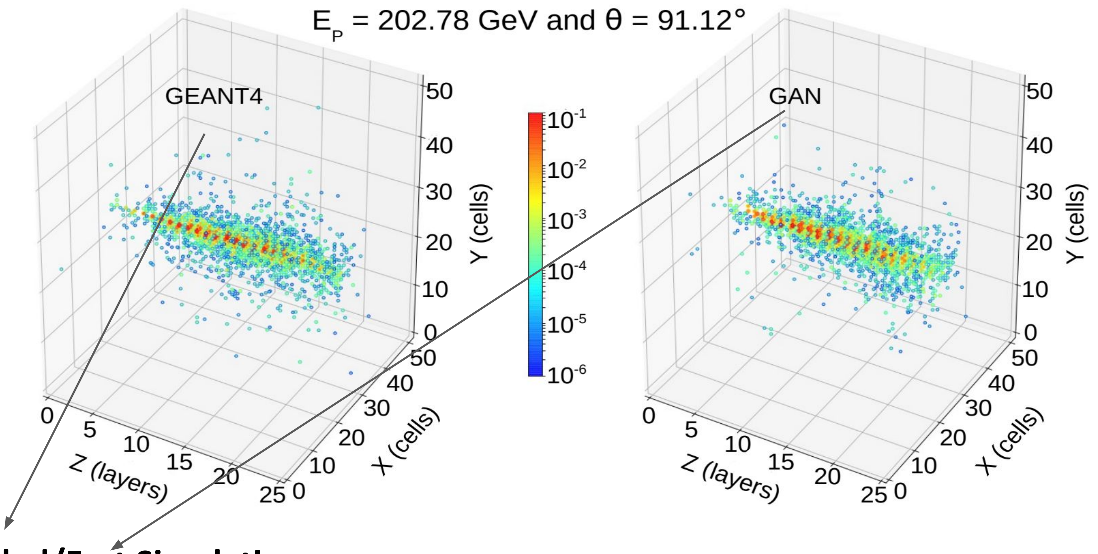


* *Gauss is the LHCb simulation framework based on Gaudi* [J. Phys. Conf. Ser. 331 032023]

Machine Learning in Fast Simulation

Machine Learning models are studied to replace the Geant4 simulation phase as in most other experiments [Chekalina (2018), Khattak (2021)].

With these models the reconstruction step is the same as for real data (and detailed simulation).



Detailed/Fast Simulation

Gauss*

Generator *e.g. Pythia8*

Simulation *Geant4*

Data processing

Event Reconstruction

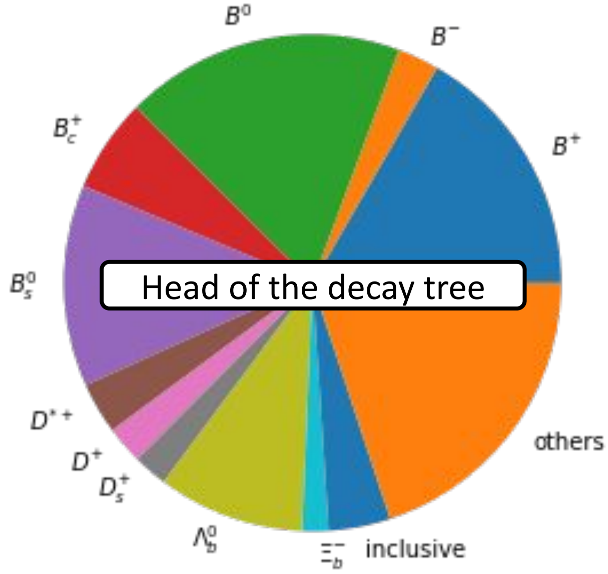
Decay Reconstruction

* *Gauss is the LHCb simulation framework based on Gaudi* [J. Phys. Conf. Ser. 331 032023]

How does LHCb simulate events?

What should we parametrize first?

10^4 MC datasets generated with 2016 nominal conditions

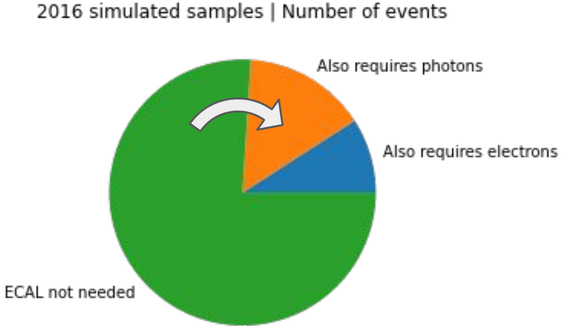


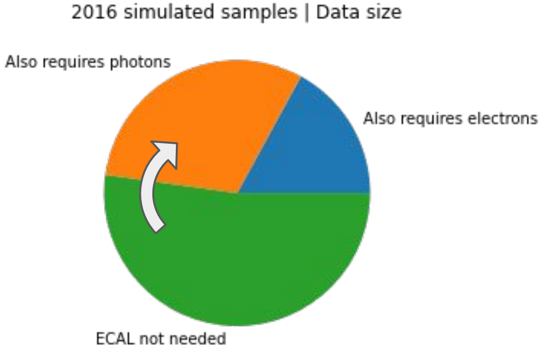
Most simulated decay modes are heavy hadron decays.

The detector will provide very similar *"response"* to, *e.g.*, a *kaon* from either a B^+ or a B_c^+.

We could save a lot of computing resources by parametrizing the **detector response** to that *kaon* and applying it to whatever decay model.

Or, with **Parametric Simulation.**



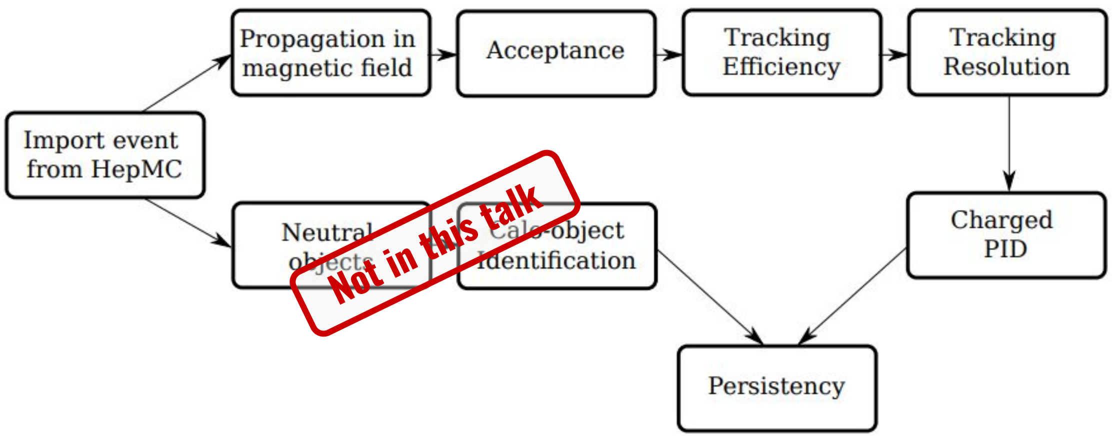


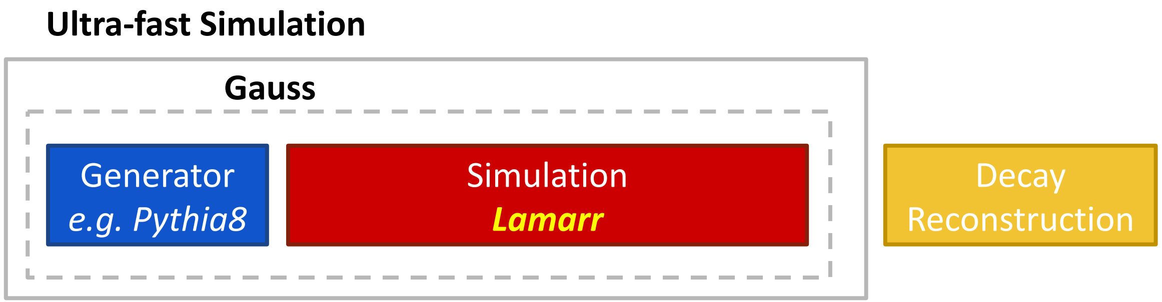
Analyses involving $h^{\pm}$ and $\mu^{\pm}$, only, often **drop simulated raw detector information** immediately.

Lamarr: *a pipeline of parameterizations embedded in Gauss*

Lamarr is a pipeline of **modular parametrizations**, integrated with the LHCb analysis framework:

- compatibility of the same, LHCb-tuned, **generators**
- compatibility with the **distributed computing** middleware (LHCbDirac) and production environment
- producing datasets with same **persistency** format





The models

Machine Learning parametrizations: two families

Efficiencies

Gradient Boosted Decision Trees (GBDTs) trained on simulated data with *Binary* or *Categorical Cross Entropy* to predict the fraction of "*good**" candidates, *i.e.* the "efficiency"of a specific step as a function of generator-level quantities.

- GBDTs are robust and easy to train
- Almost no preprocessing is needed

* either "accepted", "reconstructed", "selected"... depending on the context

Reconstructed quantities

Conditional **Generative Adversarial Networks** trained on either simulated or calibration data.

Various GAN flavours adopted for different parameterizations balancing between accuracy and robustness.

Training is performed on **opportunistic GPU resources** provided to the Collaboration.

Geometrical acceptance

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

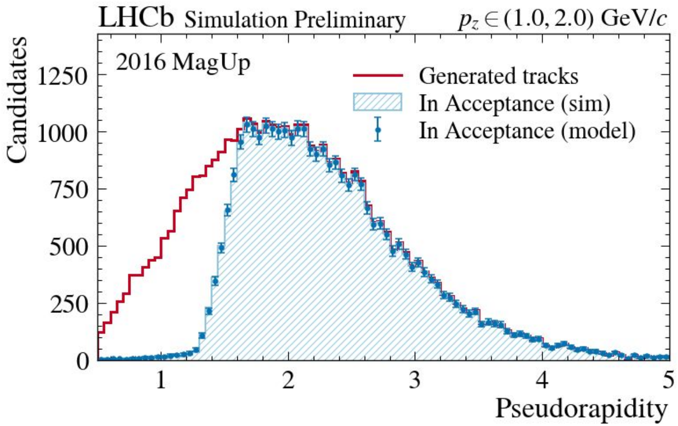
`loss   :` Binary Cross Entropy

`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**

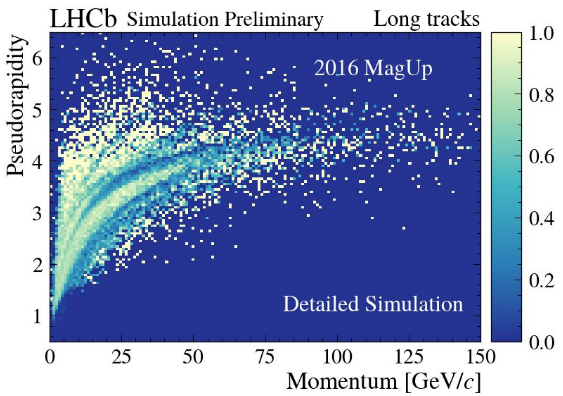
`loss` : Multi-class Cross Entropy

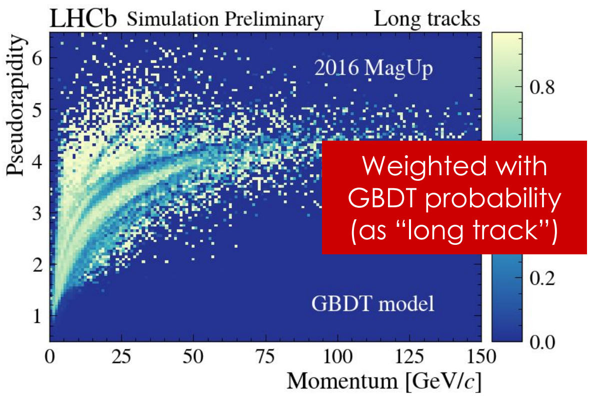
`input` : position and slope of tracks

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

Training performed on **Detailed Simulation**

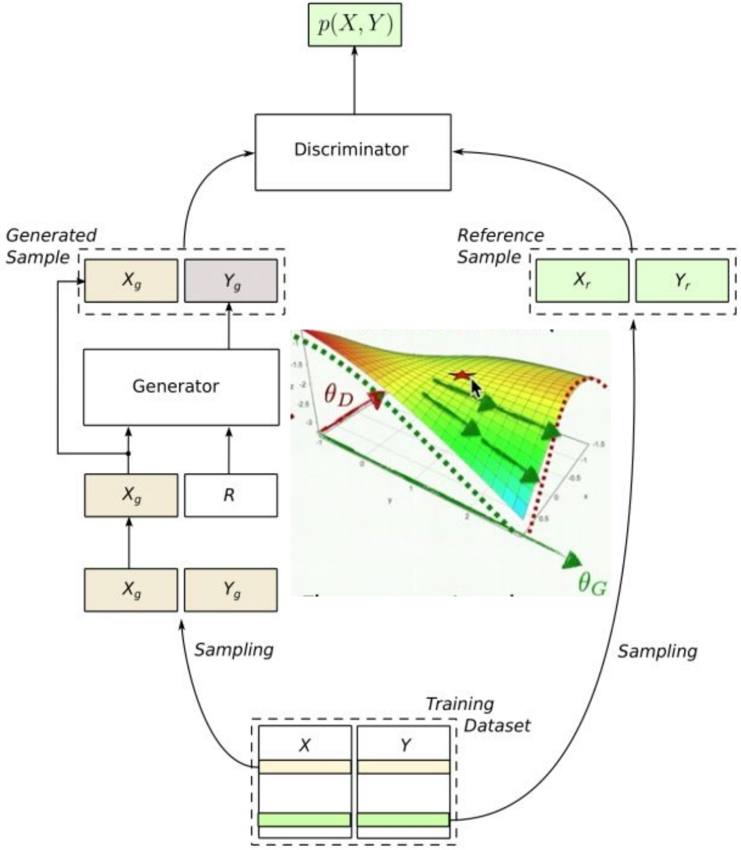
The good performance of the GBDT model well-reproduces the **complex structure of shadows** describing the efficiency losses due to the non-trivial material sub-structure of the LHCb detector.

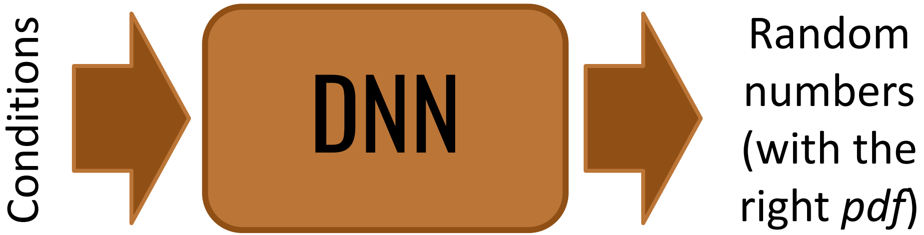




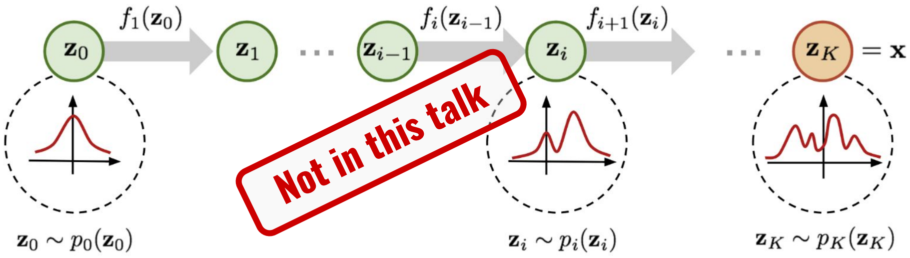
[LHCb-FIGURE-2022-004]

Generative Deep Neural Networks



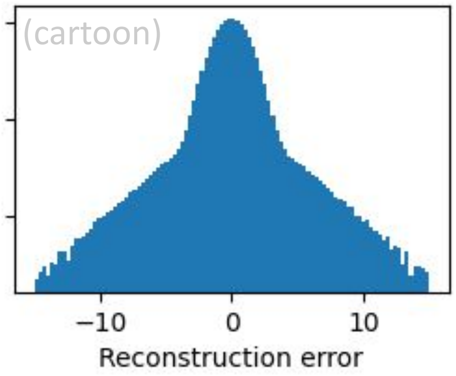


Generative Adversarial Networks (GANs) and **Normalizing Flows** (NFs) are emerging as go-to solutions for building parametrizations for fast simulations.



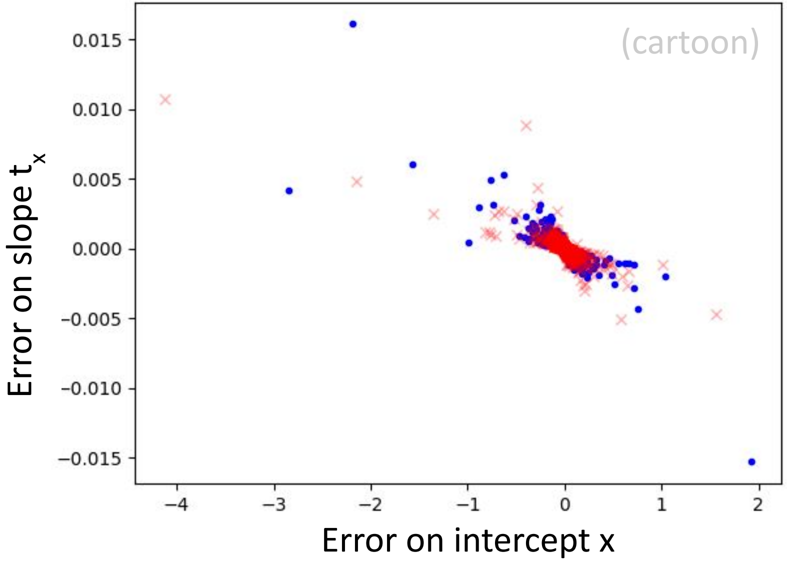
Tracking resolution modeling with a GAN

Generator-level particle features (origin position, momentum coords) → **DNN** → Reconstruction errors on intercept and slopes



Smearing is most often treated by adding Gaussian-distributed noise.

Unfortunately, **error distributions are not Gaussians** and **correlations** between reconstruction errors are not trivial.



Tracking resolution

`model  :` **Generative Adversarial Networks**

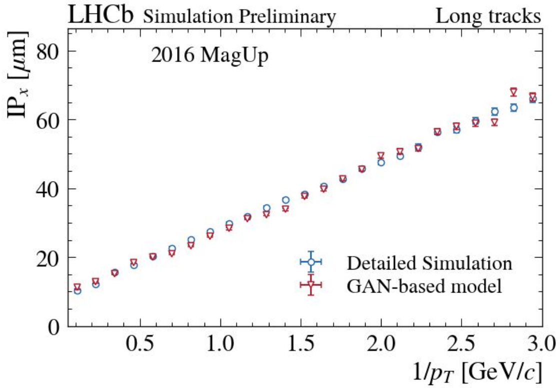
`loss   :` Binary Cross Entropy

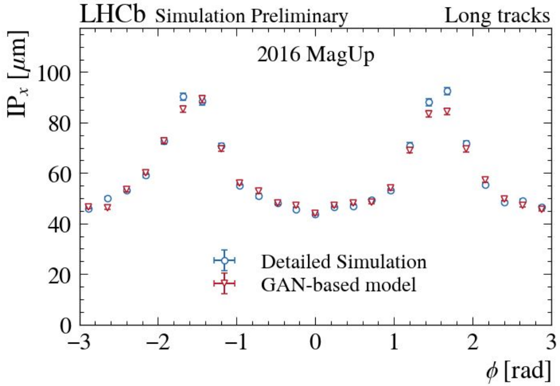
`input  :` position, slope and momentum of tracks

`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]

Training Particle Identification on real data

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]

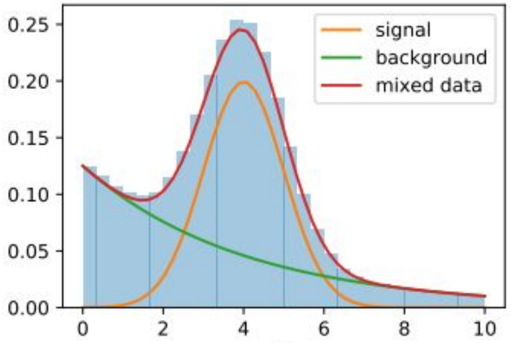
CramerGAN [arXiv:1705.10743]

Calibration samples [LHCb-PUB-2016-020]

Training Particle Identification models on real data

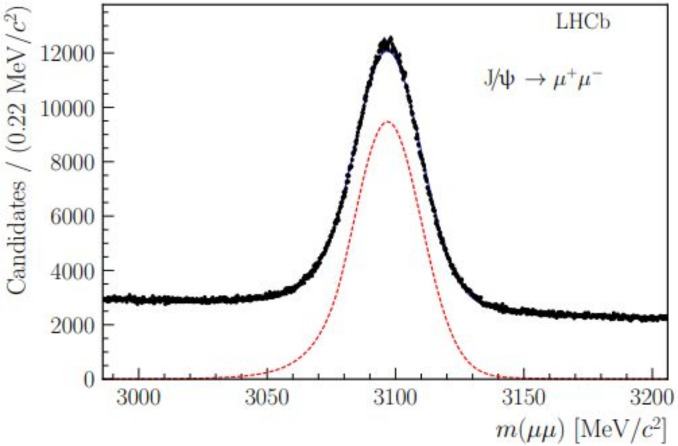
Calibration datasets are obtained selecting special decay modes (enabling Particle Identification with *tag&probe* techniques) with **special trigger lines explicitly avoiding biases on the probe**.

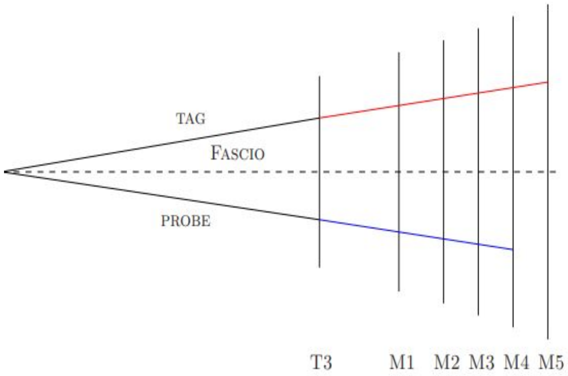
Background is then subtracted with sPlot technique.

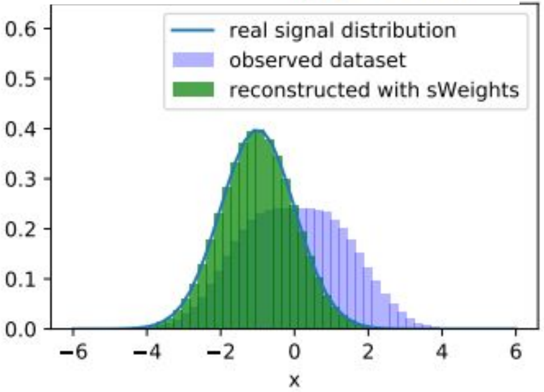


JINST 14 (2019) P08020

$$\text{sWeight}_n(m_e) = \frac{\sum_{j=1}^{N_s} \mathbf{V}_{nj} p_j(m_e)}{\sum_{k=1}^{N_s} N_k p_k(m_e)}$$







Modification to the loss to ignore the background

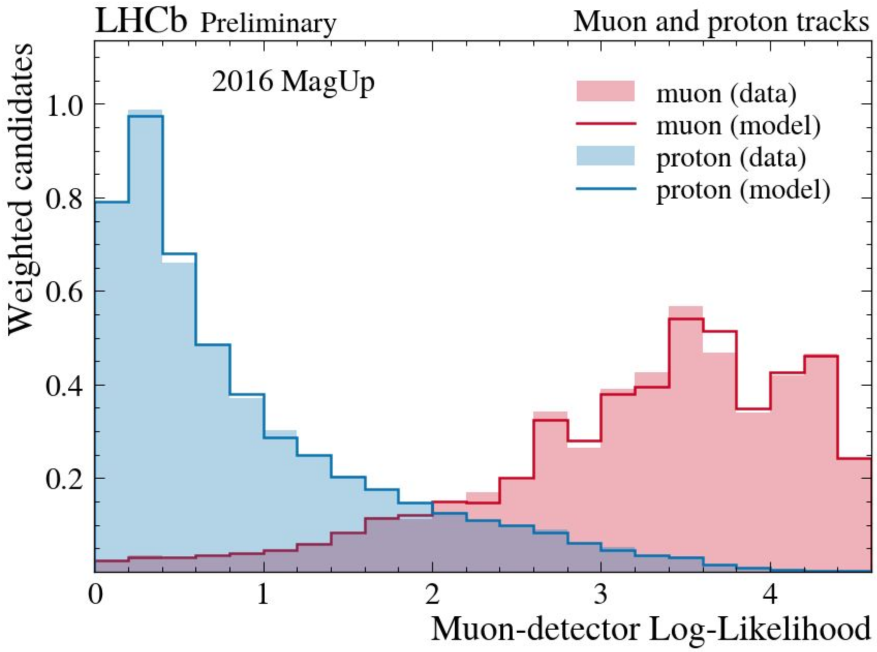
The loss function of the Discriminator is then simply modified to statistically subtract the background contribution.

For example for the binary cross-entropy,

$$\mathcal{L} = -\sum_i {}_s\mathcal{W}_i \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right]$$

Techniques to stabilize the GAN training when using negative weights were studied in more detail in [Borisyaka (2019)]

Muon detector: *muon-proton separation*



[LHCb-FIGURE-2022-004]

`model` : **Generative Adversarial Networks**

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

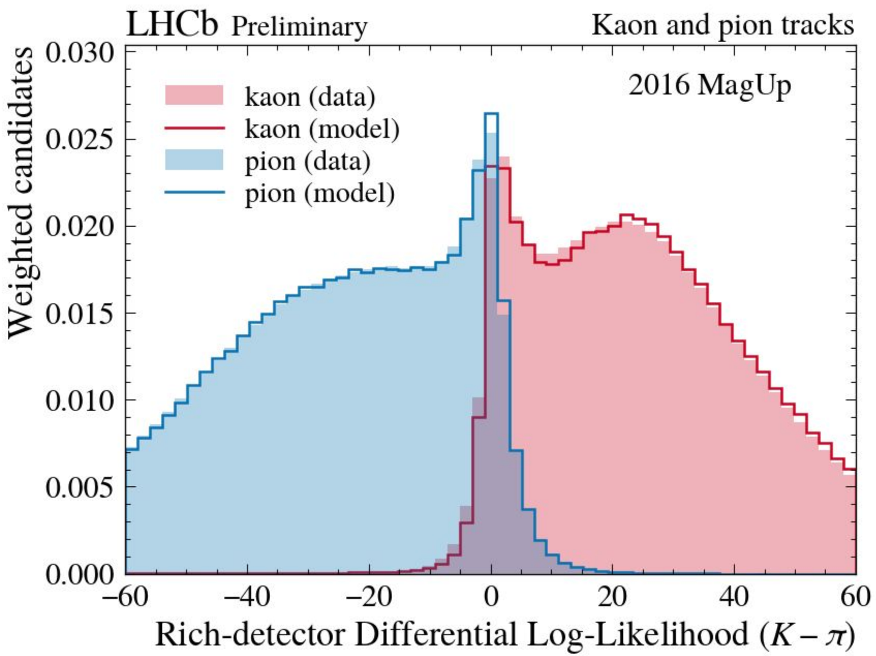
`input` : track kinematic parameters and
detector occupancy

`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.

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.

Loose Binary Muon Identification Criterion: *isMuon*

isMuon criterion
[JINST 8 (2013) P10020]

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

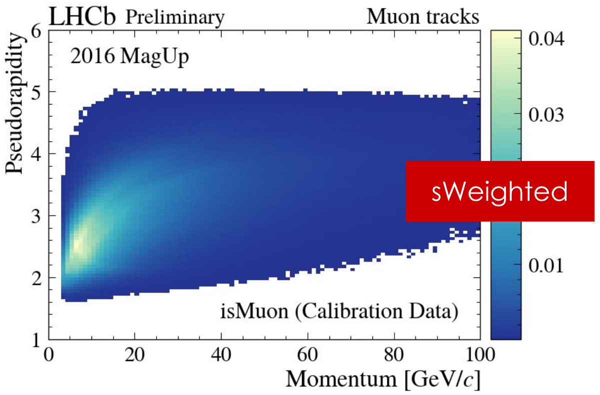
`loss   :` Binary Cross Entropy

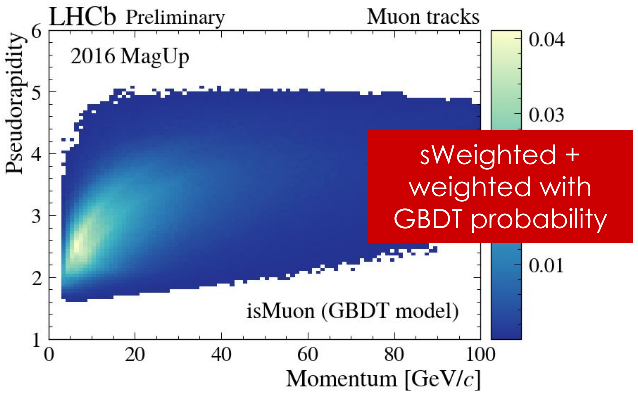
`input  :` track kinematic parameters and detector occupancy

`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.

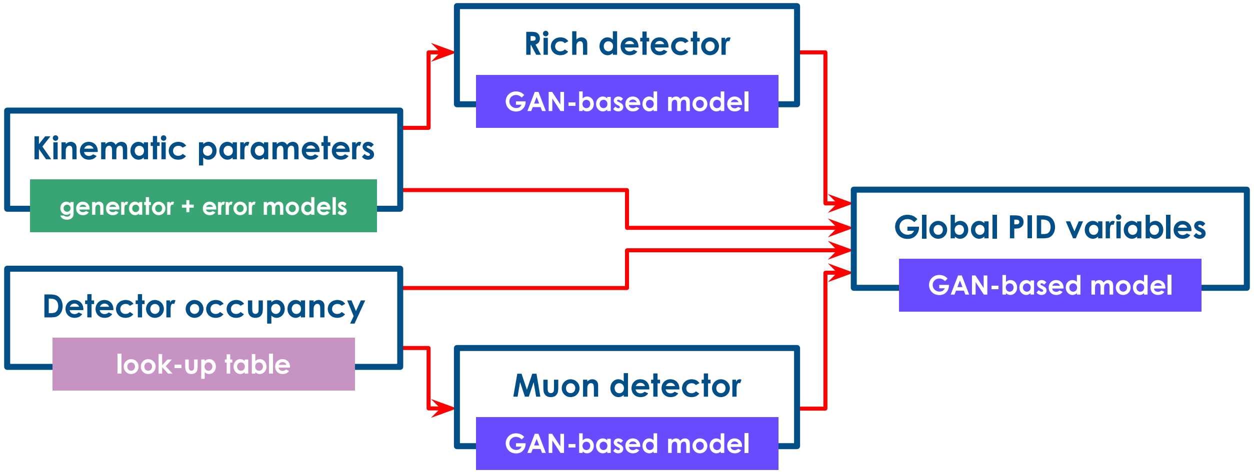




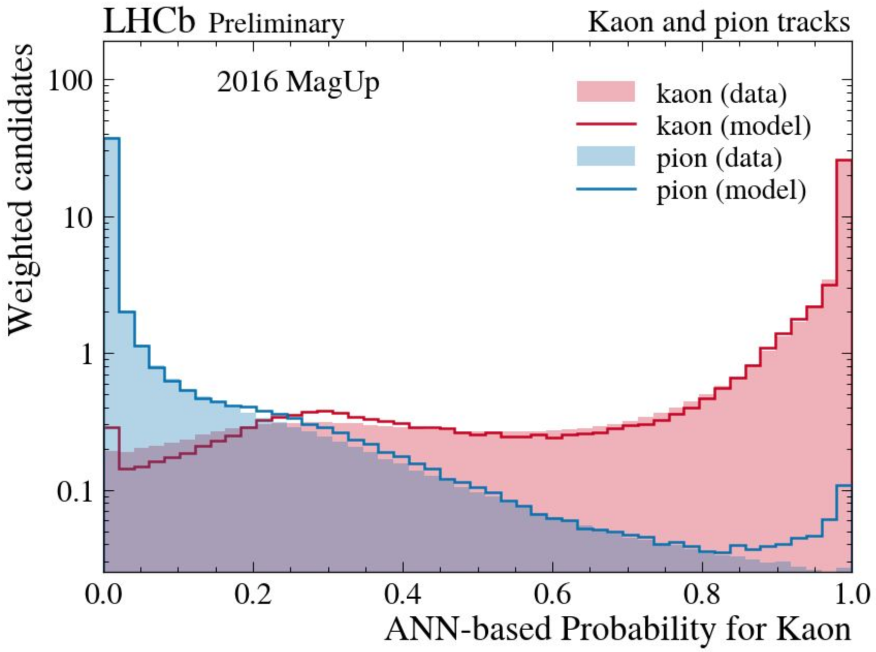
[LHCb-FIGURE-2022-004]

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)

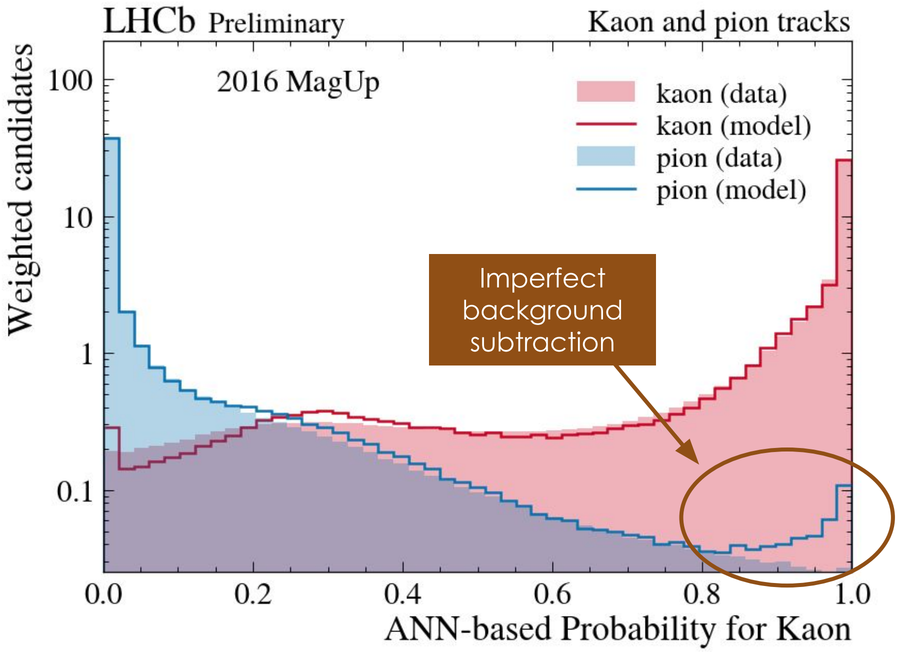
`input` : track kinematic parameters ,
detector occupancy , isMuon ,
high-level response of the Rich detector ,
high-level response of the Muon detector

`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)

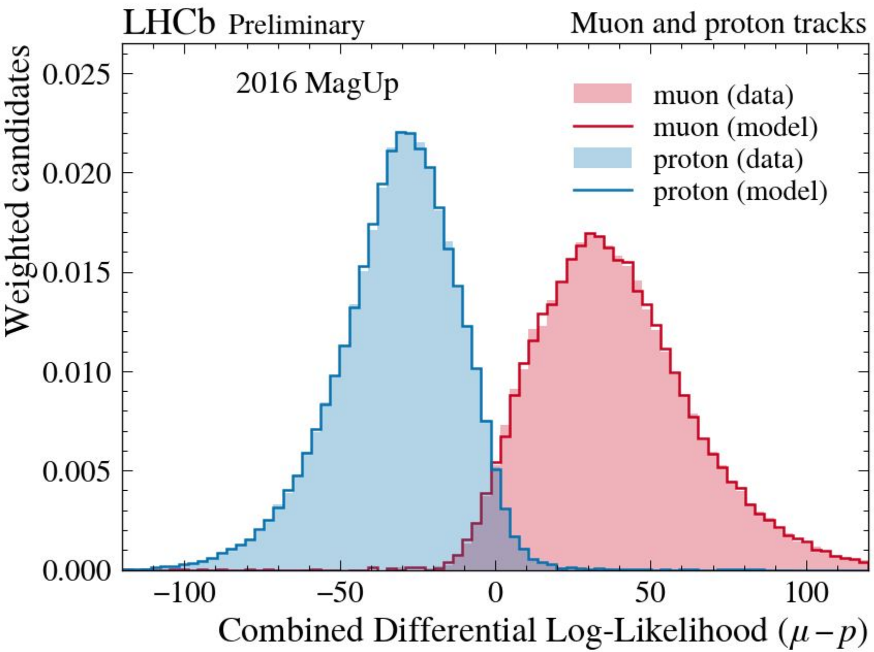
`input` : track kinematic parameters ,
detector occupancy , isMuon ,
high-level response of the Rich detector ,
high-level response of the Muon detector

`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 Particle Identification: *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**

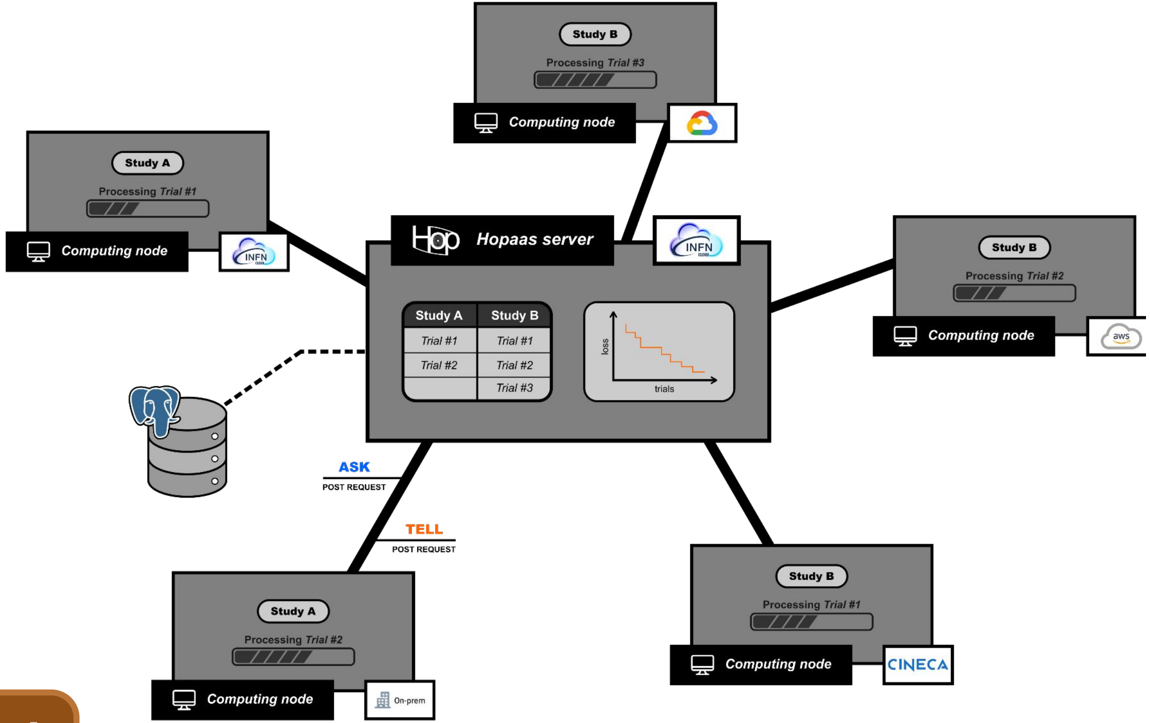
4 neural networks trained in *adversarial configuration* are used to parameterize various global PID variables shown together in the **Combined Differential Log-Likelihood** for muon versus proton hypothesis.

Distributed HPO

Distributed Hyper Parameter Optimization

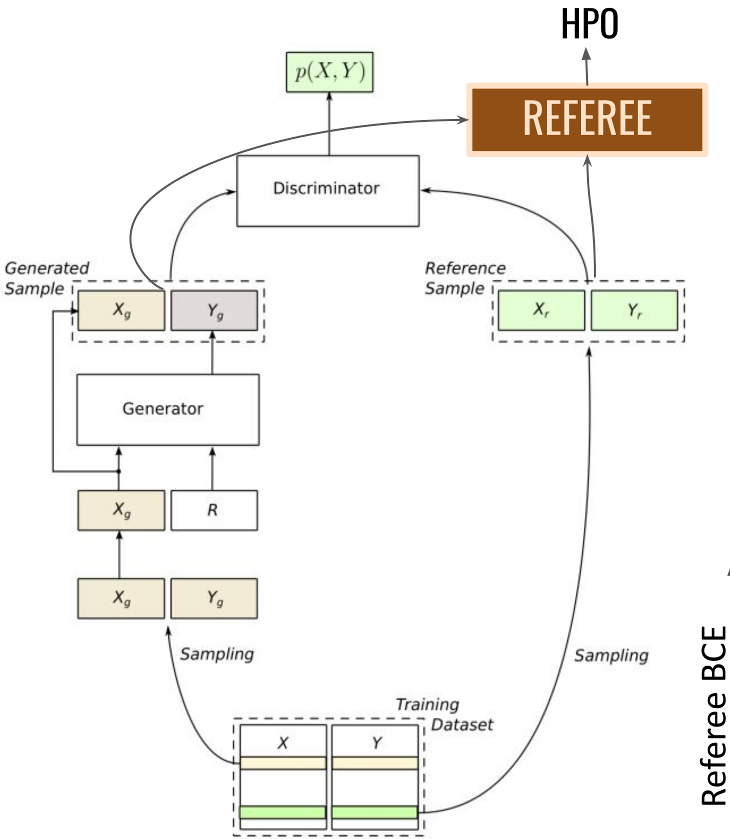
Training GANs benefits from **massive hyperparameter optimization campaigns**.

To enable using opportunistic resources we need a **centralized service for managing HPO campaigns**, independent of the resource provider.

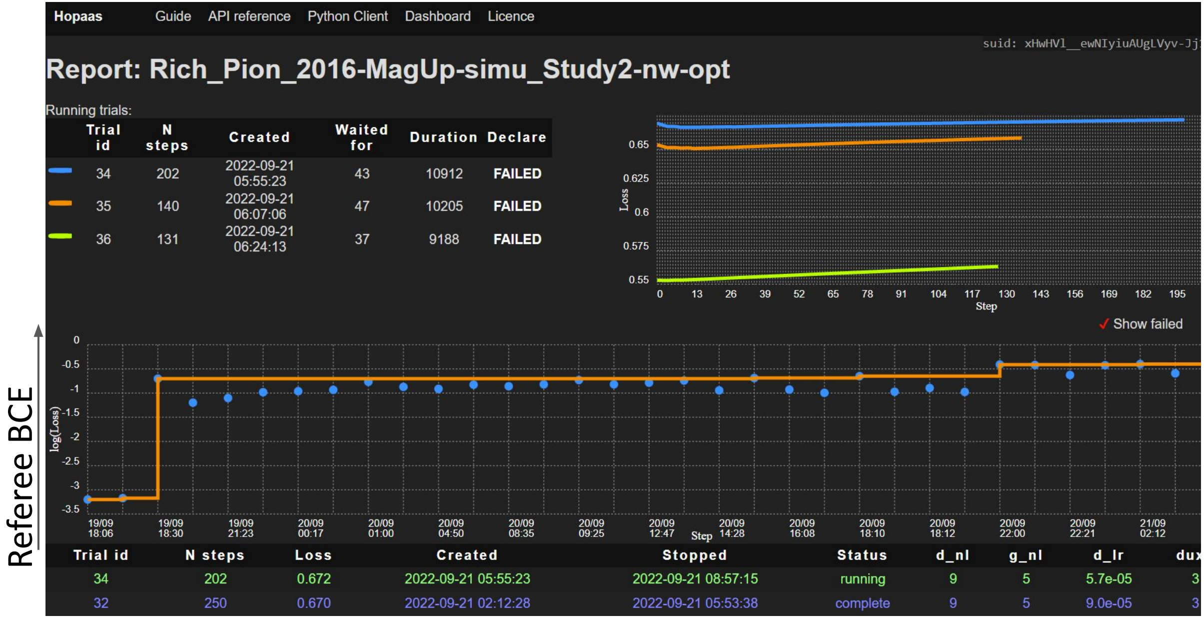


Web-based service hosted by INFN Cloud accessed through REST APIs

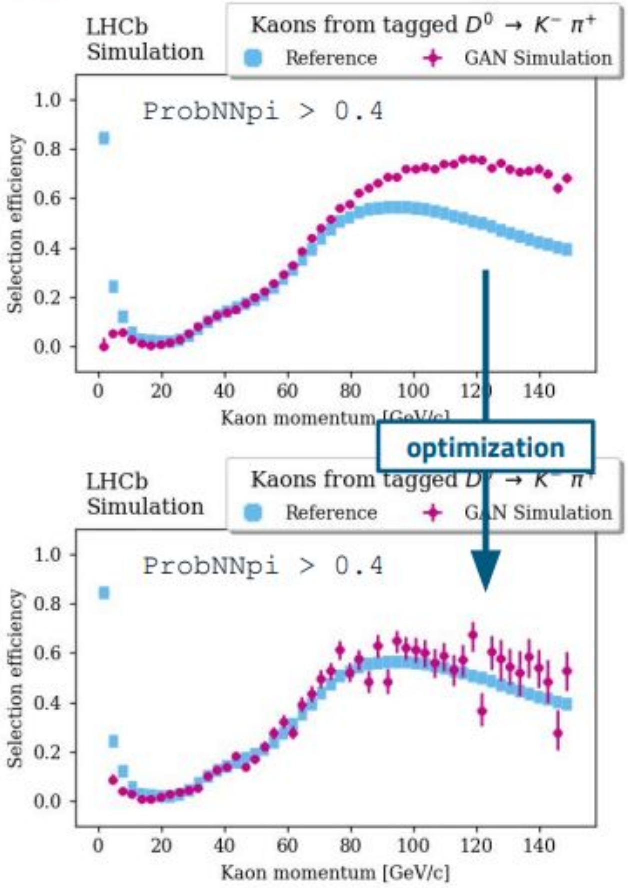
Referee network and dashboard



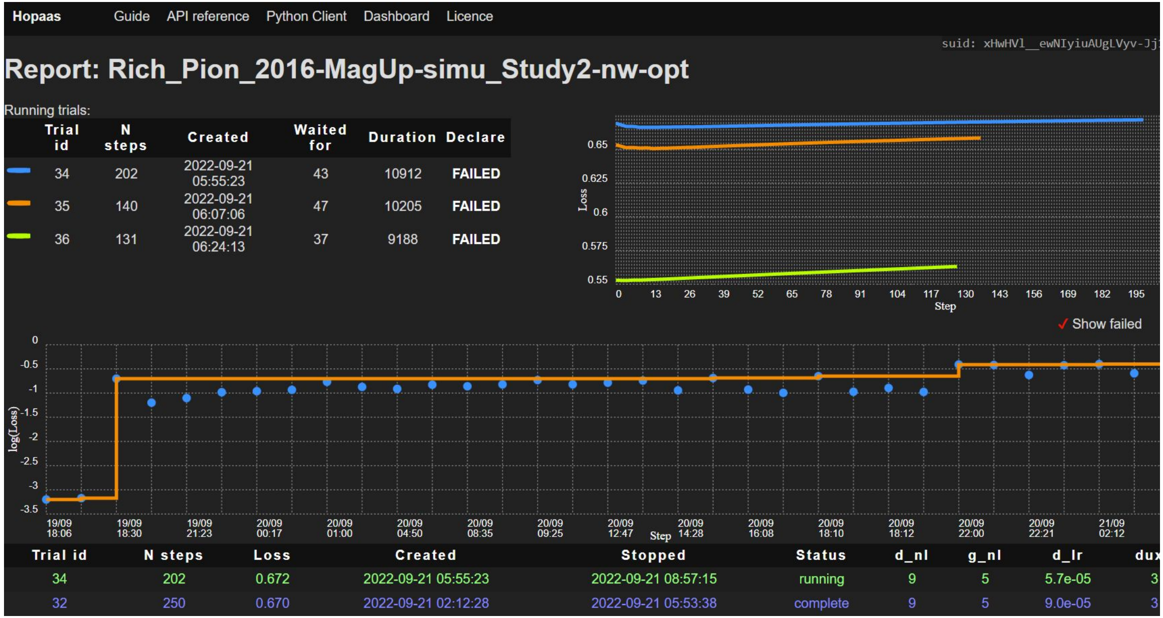
We use a **referee network**, architecturally equivalent to the discriminator, but evolving independently and **never used to inform the generator**, only for HPO.



Effects of the HPO on PID efficiencies



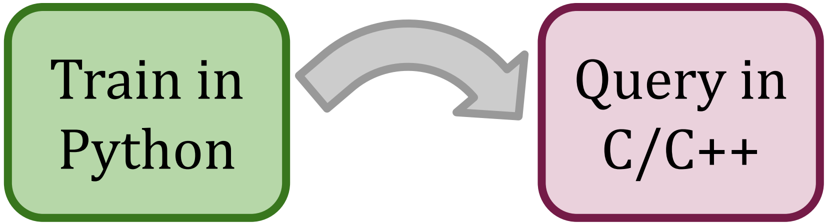
HPO is observed to be particularly beneficial to improve modeling on the tails of the condition distributions.



Deployment in HEP C++ applications

Deploying trained models in Gauss

Using trained ML models in C++ applications is wider and more general issue.



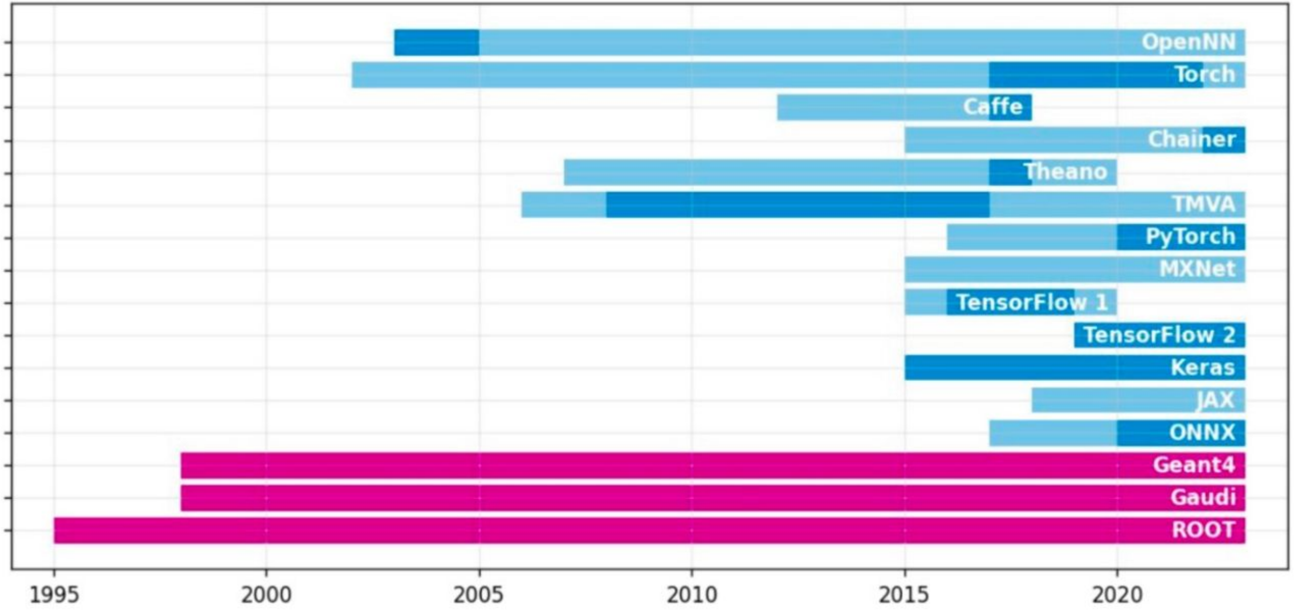
Several options for deployment exist, but come with some practical limitation.
For example,

- ➢ Require **external dependencies** sometimes difficult to integrate in the build system of large HEP applications
- ➢ Expect vectorized inputs introducing **overhead for branched flows**, as for example Geant4-based simulations
- ➢ Introduce limits in the interplay between the **preprocessing** and **algorithmic** steps
- ➢ Often require **compiling with the framework** large part of the algorithm.

Choosing a DNN framework to rely on

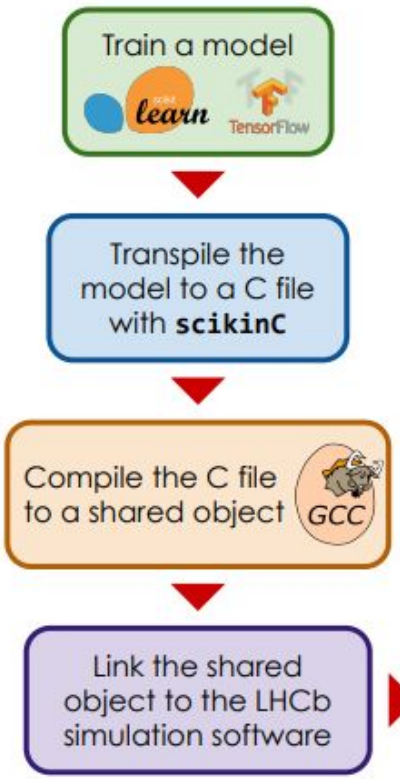
Frameworks and standards to define and deploy DNN in C++ applications have **change a lot during the last years.**

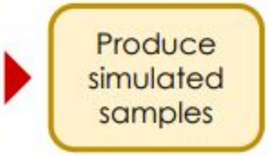
Choosing one for an application we would like to keep using in production in O(10) years **would be a bet**.



Also, multithreading is used differently (and possibly inconsistently) by HEP frameworks and ML frameworks.

Transpiling approach: `scikinC` *and* `keras2c`





For a seamless integration of the trained parameterizations in the LHCb simulation framework models have to be applied to each single particle → **thousands of independent calls per event**.

Even a small latency (*e.g. context switching*) wastes unacceptable amount of CPU resources .

We **transpile our models in C** and compile them to binaries, **dynamically linked** at runtime.

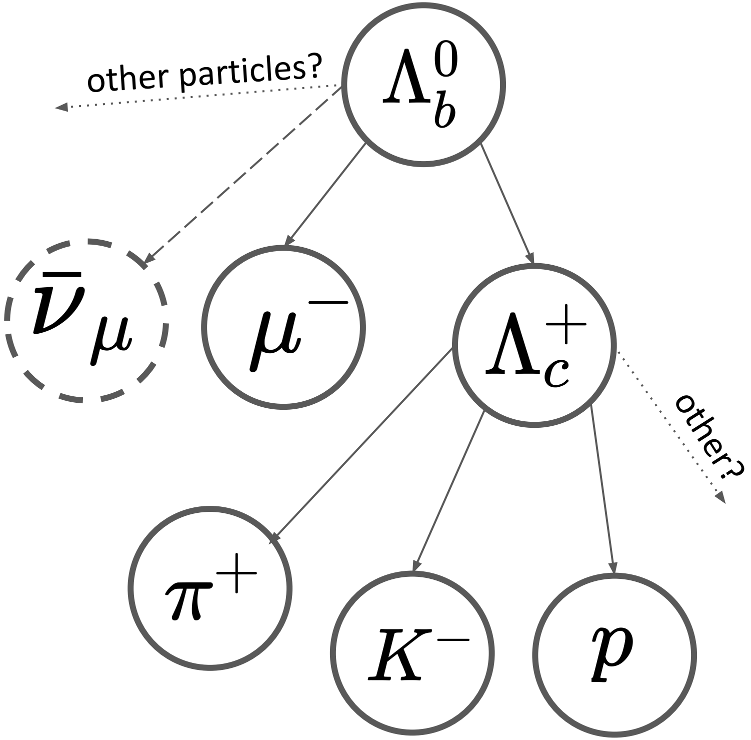
LHCb tool: **scikinC** [PoS(CompTools2021)034]

Possible partial migration to **keras2c** [J.Eng.App.AI, (2021) 104182]

Physics validation

Lamarr validation: $\Lambda_b^0 \to \Lambda_c^+ \mu^- \bar{\nu}_\mu$ with $\Lambda_c^+ \to p K^- \pi^+$

- Abundant decay in LHCb, widely studied to measure, *e.g.*, beauty baryon production
  - *e.g. see* JHEP 10 (2021) 060, PHYS. REV. D100 (2019) 032001, PHYS. REV. D96 (2017) 112005, …
- It is part of the Particle Identification Calibration samples [EPJ TI 2019 6:1];
- It is described by a complex decay model including many feed-down modes;
- It provides examples for **muons**, **pions**, **kaons** and **protons** in a single decay mode.

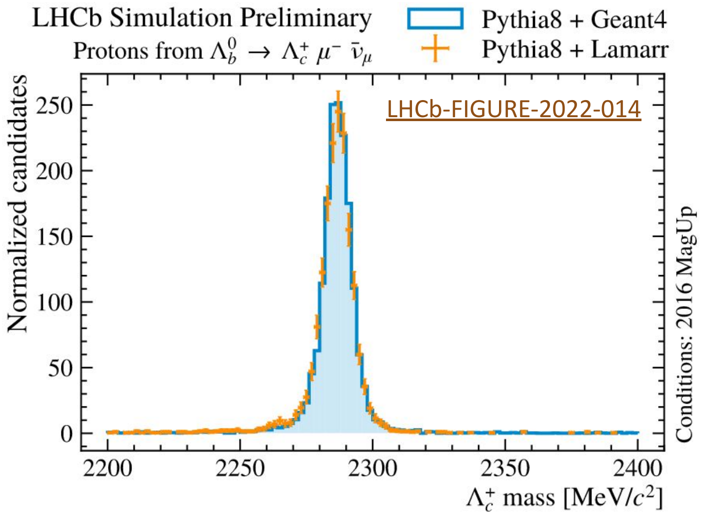


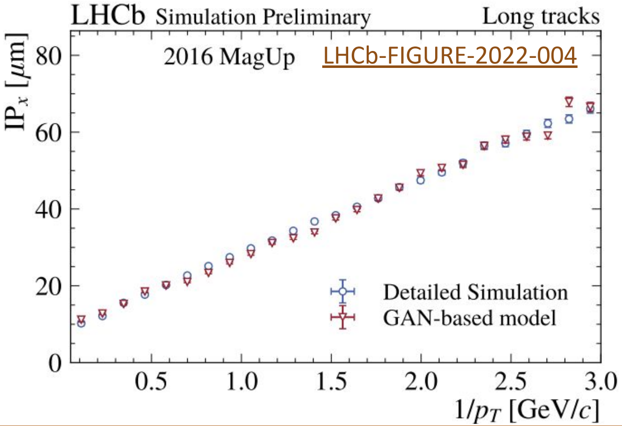
The training of the models is based on a cocktail of heavy flavour decays, where $\Lambda_b^0 \to \Lambda_c^+ \mu^- X$ represents a negligible fraction.

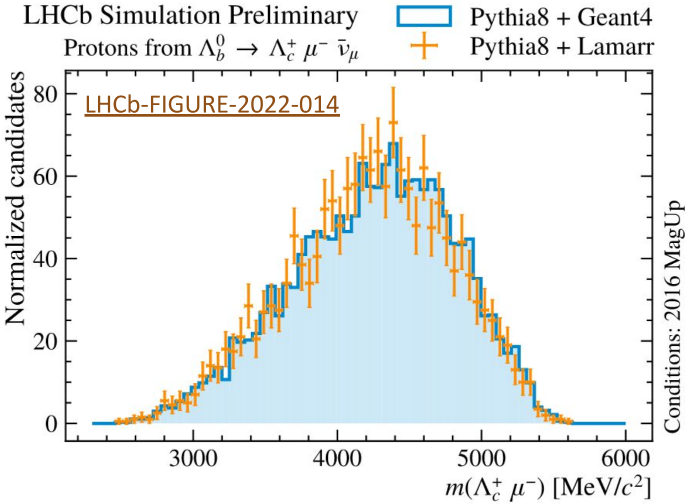
Track smearing

The momentum and point of closest approach to the beams of the generated particles **get smeared**: a GAN predicts effects as *multiple scattering*, imperfections of alignment, calibration…

Reconstructed masses and **impact parameters** are then computed on the smeared quantities.





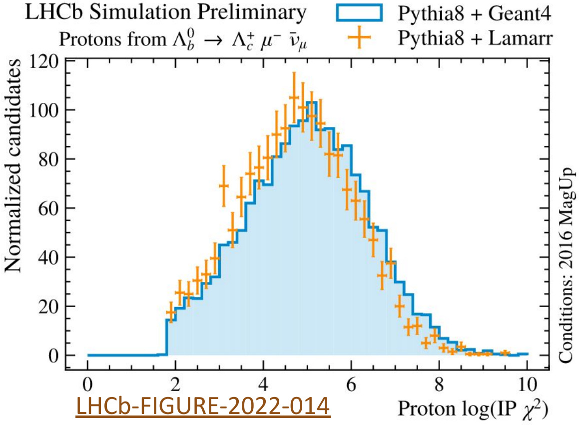


Tracking uncertainties

A GAN is used to predict the **uncertainties associated to the track reconstruction**.

Track uncertainties are crucial in LHCb to define the **consistency of trajectories with vertices**.

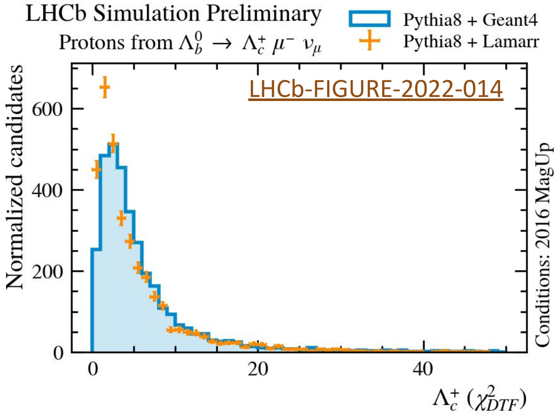
For example, the **impact parameter χ^2** is a measure of inconsistency of a trajectory with a PV.



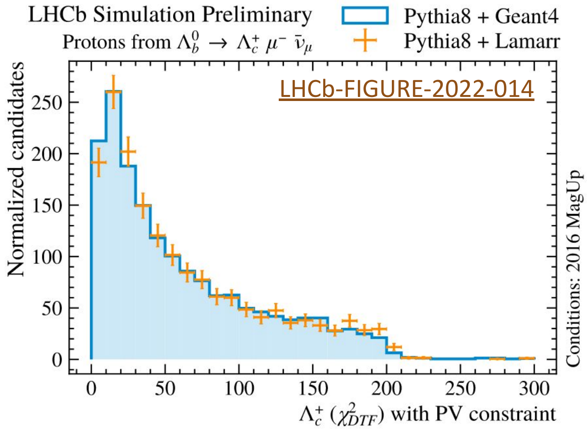
LHCb-FIGURE-2022-014

The whole decay tree can be fitted at once with a ***Decay Tree Fitter*** [NIMA 552 (2005) 566]

Constraining the Λ_c^+ to be produced in the PV, the χ^2 explodes.

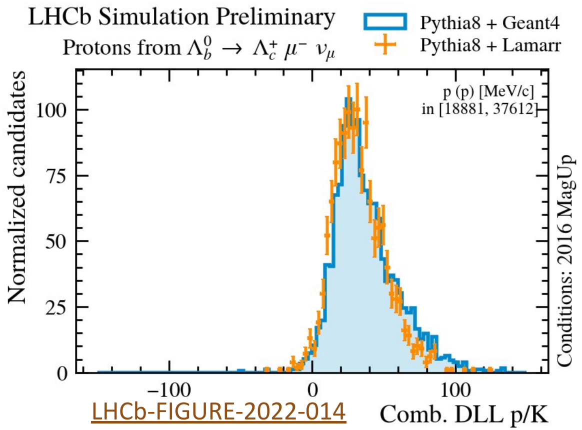


Include PV constraint

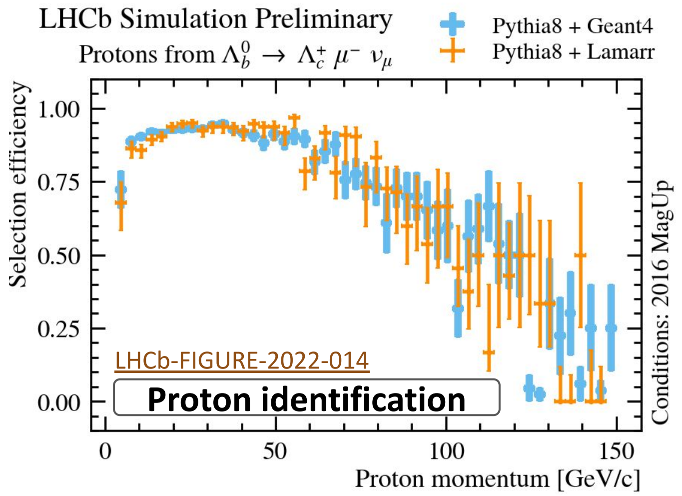


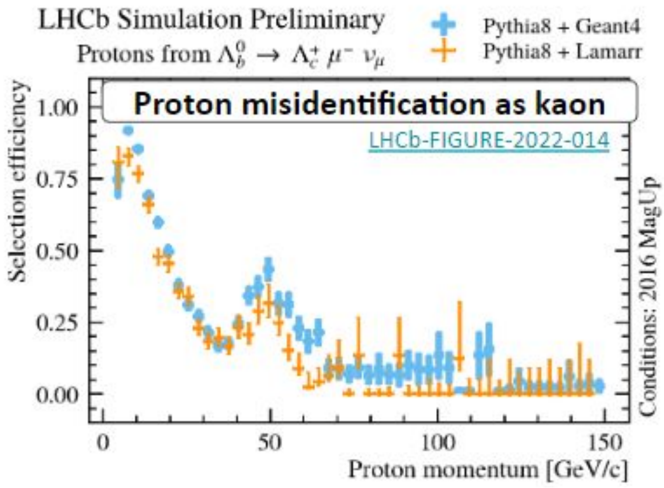
Proton identification

Lamarr simulates the distribution of the detector response. Analysts often inject the detector response in some analysis-specific classifier.



Here, we define cuts to visualize the ability of the trained models to describe the **dependence of the detector response on occupancy and kinematics**.





One more word on timing

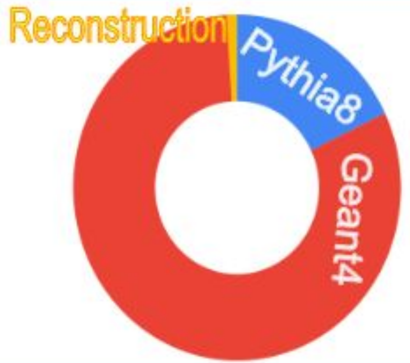
Comparing the normalized CPU spent for Geant4-based and Lamarr simulations of $\Lambda_b^0 \to \Lambda_c^+ \mu^- X$ decays we estimate a CPU reduction of 98.3 % for the ***Simulation* phase**.

Generation of *b*-baryons is exceptionally expensive: here Pythia 8 largely dominates the CPU consumption.

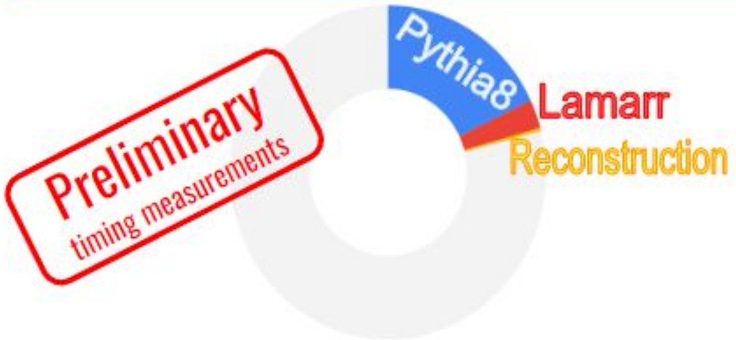
Generation of *b*-mesons requires 5% of less of the overall Simulation time.

Repeating the exercise **on minimum bias, CPU reduction exceeds 99%.**

Detailed simulation: Pythia8 + Geant4
1M events @ 2.5 kHS06.s/event ≃ 80 HS06.y



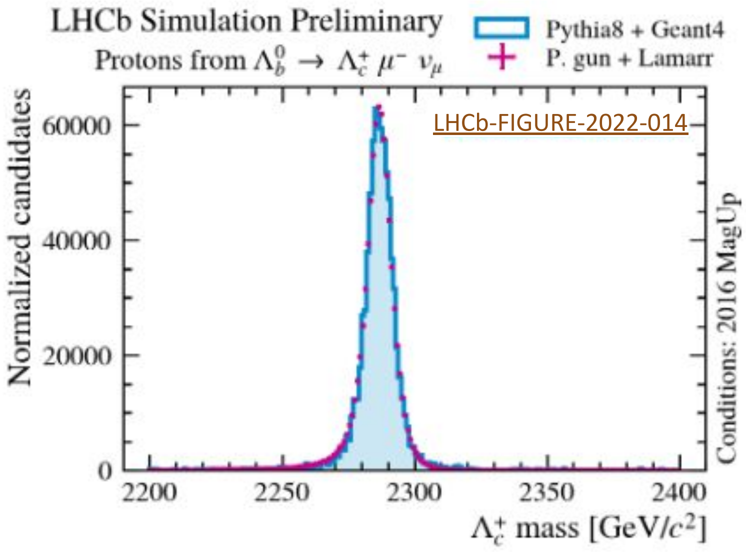
Ultra-fast simulation: Pythia8 + Lamarr
1 M events @ 0.5 kHS06.s/event ≃15 HS06.y

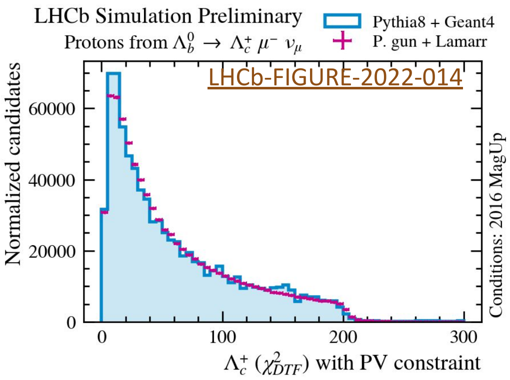


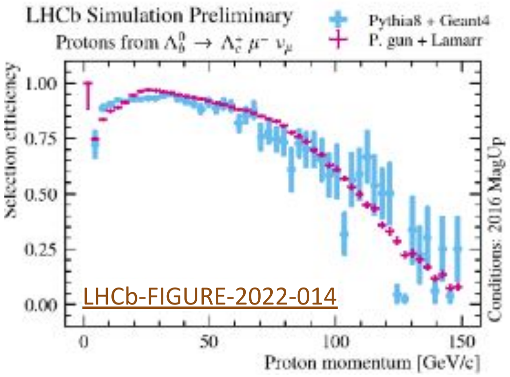
Saving more with Particle Guns

Detector occupancy is parametrized: one can achieve similar performance by **only simulating the signal particles** (*i.e.* with *Particle Guns*).

Production spectra are generated once-for-all with Pythia8 and then sampled.







Detailed simulation: Pythia8 + Geant4
1M events @ 2.5 kHS06.s/event ≃ 80 HS06.y

Ultra-fast simulation: Pythia8 + Lamarr
1 M events @ 0.5 kHS06.s/event ≃ 15 HS06.y

Ultra-fast simulation: Particle Gun + Lamarr
100 M events @ 1 HS06.s/event ≃ 4 HS06.y

Conclusion and outlook

Technology Readiness Level & Limitations

- 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

Conclusion

Private productions are currently run on the **WLCG** and could be made standard, centralized productions easily.

We are now **tuning models to compromise between accuracy and CPU performance**, focusing on 2016 datataking conditions. We plan to extend to 2015, 2017 and 2018 soon. Run1 support may come later.

Lamarr will never replace Detailed Simulation, but may provide soon a precious tool to **design selection** strategies, **train multivariate classifiers**, study kinematic-induced **correlation effects** in the analysis-level quantities, or in general when **theoretical uncertainties** on the decay model are large.

As of today, these use-cases are mostly covered with *Detailed Simulation*.

Acknowledgements

ICSC
Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

This work is partially supported by ICSC – Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing, funded by European Union – NextGenerationEU.

Finanziato dall'Unione europea
NextGenerationEU

Ministero dell'Università e della Ricerca

Italiadomani
PIANO NAZIONALE DI RIPRESA E RESILIENZA

ICSC
Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing