UNIVERSITÀ DEGLI STUDI FIRENZE

12.06.2020

Techniques for parametric simulation with deep neural networks and implementation for the LHCb experiment at CERN and its future upgrades

Relatore:
Dott. Lucio Anderlini

Candidato:
Matteo Barbetti

Correlatore:
Prof. Piergiulio Lenzi

Scientific background

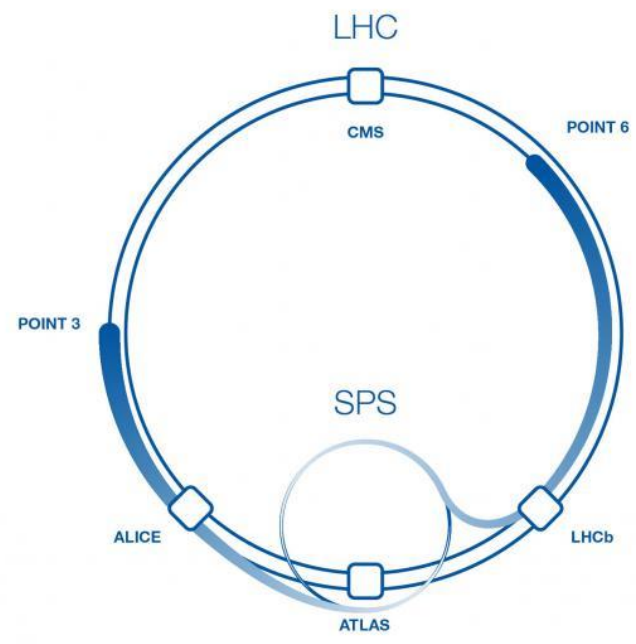
Large Hadron Collider
A few words about LHC

The **Large Hadron Collider** (LHC) is the world's largest and most powerful particle accelerator. Fired up for the first time on 2008, it is the latest addition to CERN's accelerator complex.

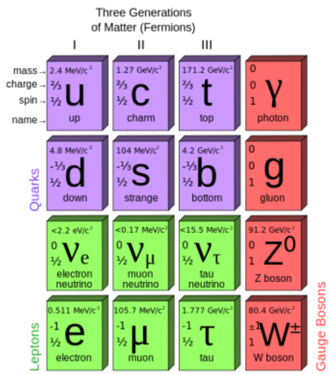
The LHC consists of 27-kilometre ring in which two **high-energy** proton beams (or ion beams) travel in opposite directions at close to the speed of light.

Protons inside the beams are arranged in bunches spaced of 25 ns, and are made to collide corresponding to the positions of **four particle detectors**:

- ALICE
- ATLAS
- CMS
- LHCb



The LHCb experiment
Physics program and detector

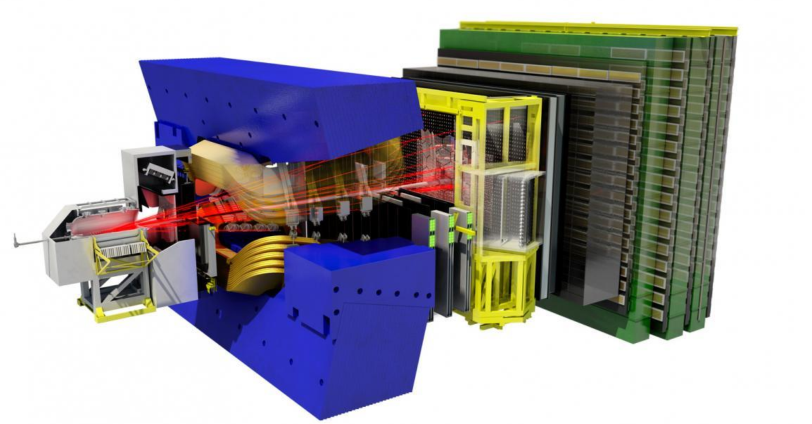


The **Standard Model** divides elementary particles into two families: quarks and leptons. Both the families appear in six different flavours, each of which with different masses and quantum numbers.

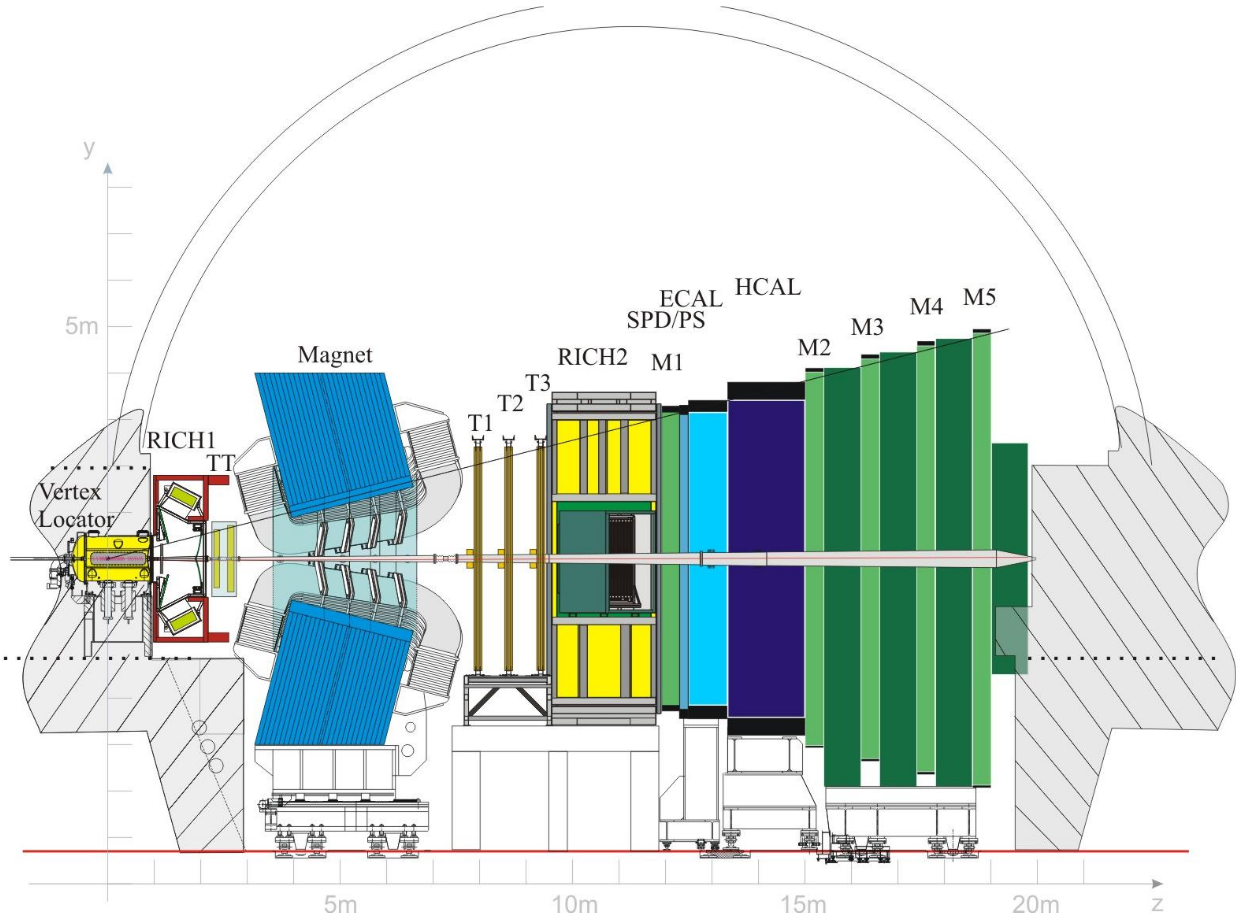
The **Large Hadron Collider beauty** (LHCb) experiment is dedicated to heavy flavour physics, namely the study of heavy quarks (*c* and *b*). Its primary goal is to look for indirect evidence of **New Physics** in *CP*-violation and in rare decays of *b*- and *c*-hadrons.

The LHCb detector is a single-arm spectrometer with a **forward angular** coverage approximately the range 10÷300 mrad. The LHCb sub-detectors can be conceptually divided into:

- Tracking system
- Particle Identification system

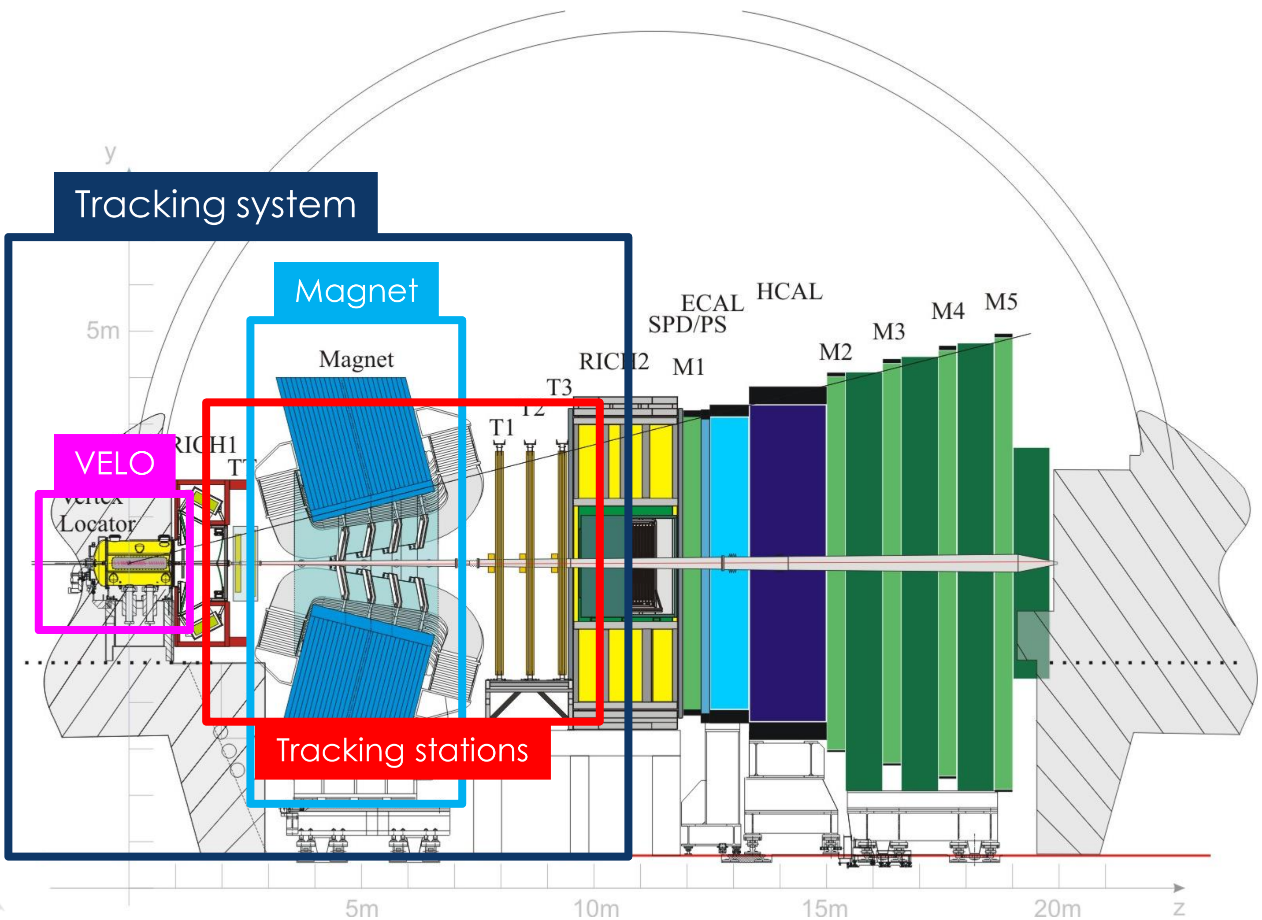


The LHCb experiment
The spectrometer layout

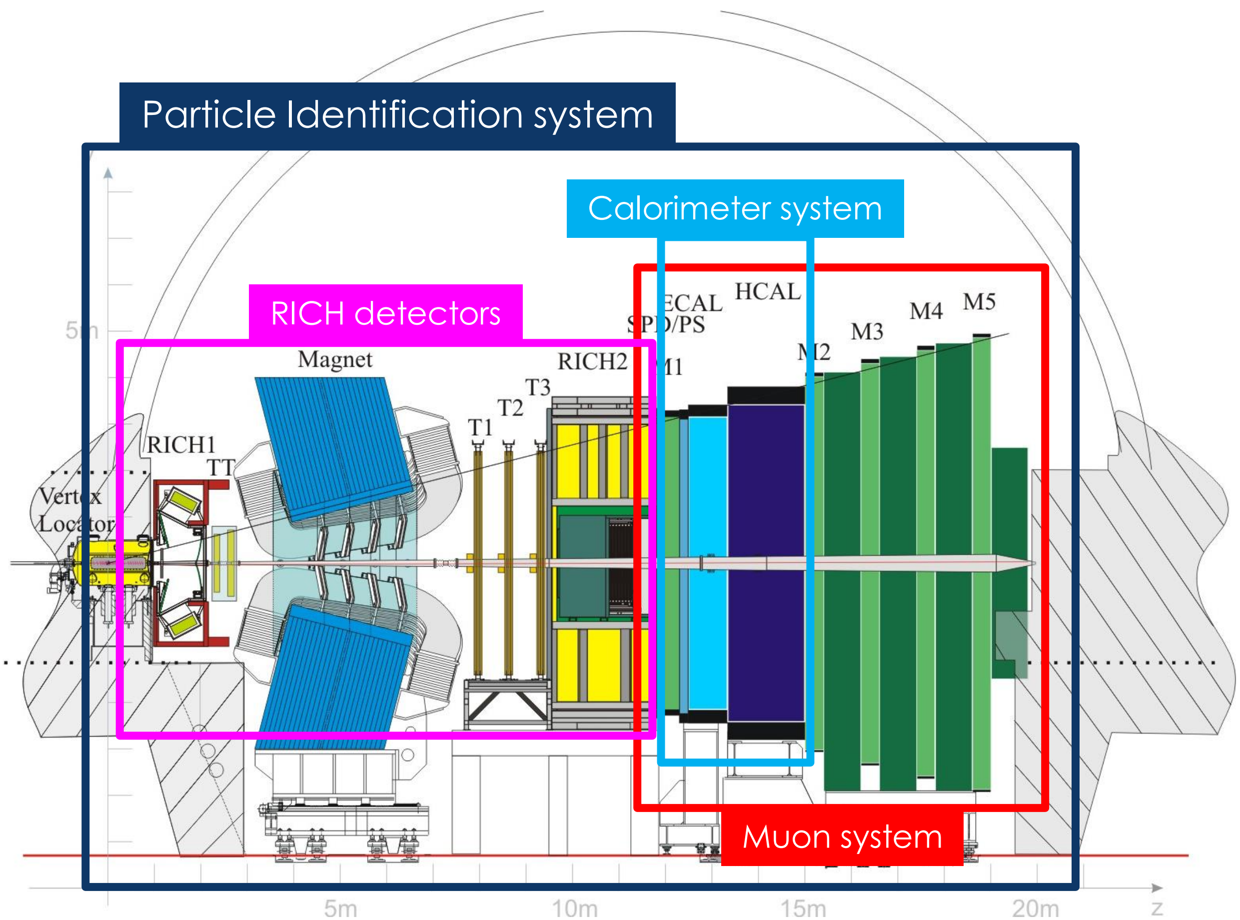


The LHCb experiment

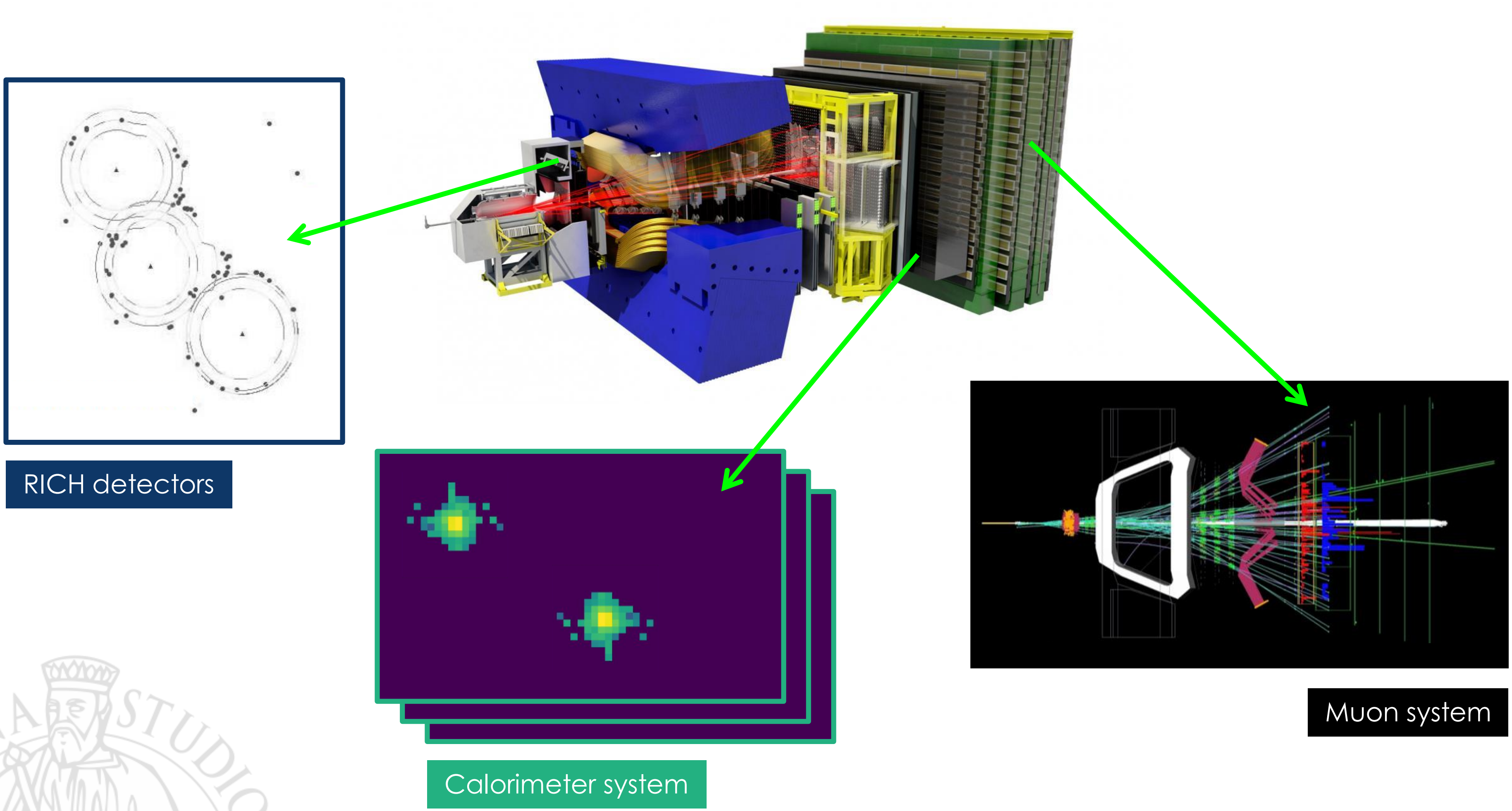
The spectrometer layout



The LHCb experiment
The spectrometer layout

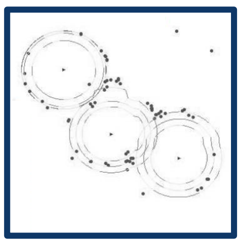


The LHCb experiment
Particle Identification



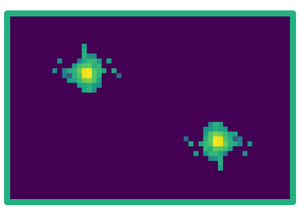
The LHCb experiment
Particle Identification variables

The **PID variables*** result from the combination between tracks and the sub-detectors responses. Exploiting <u>different physics processes</u>, one can compute several **likelihood ratio** (DLL) between particle hypotheses for each reconstructed track.



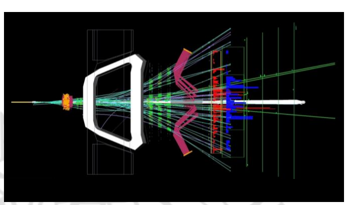
RICH detectors

<u>Better ID:</u> *pions, kaons, protons*
Compare rings expected from the track parameters to hits



Calorimeter system

<u>Better ID:</u> *electrons*
Check consistency of clusters of hits with tracks

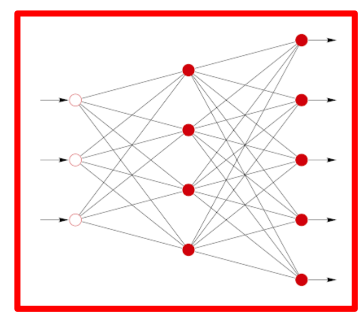


Muon system

<u>Better ID:</u> *muons*
Check consistency of tracks of hits with tracks

$$\mathrm{CombDLL} = \Delta \log \mathcal{L}_{\mathrm{RICH}} + \Delta \log \mathcal{L}_{\mathrm{CALO}} + \Delta \log \mathcal{L}_{\mathrm{MUON}}$$

Combined DLL

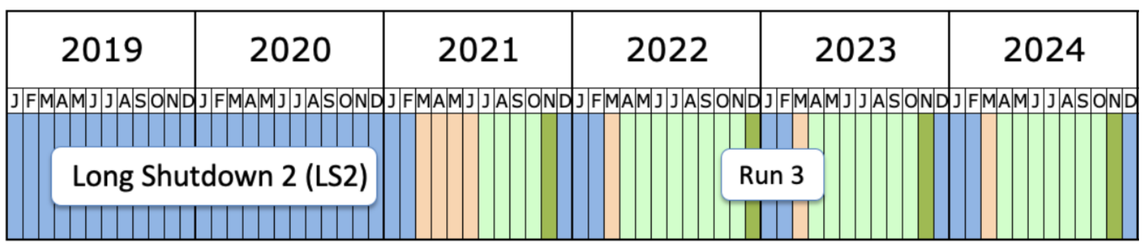


ProbNN

* The PID variables are defined for each long-lived charged particle traversing the detector (e, μ, π, K, p).

The LHCb experiment
Upgrade I

The **Upgrade I** of the LHCb experiment is currently in commissioning, exploiting the stop of data taking for the Long Shutdown 2 (2018-2021). The LHCb Upgrade detector will operate starting from LHC Run 3, and will allow to reach **unprecedent accuracy**.



90% detector channels replaced	**fully** software trigger system	**x 5** instantaneous luminosity	**x 2** selection efficiency	**x 10** data samples size

The LHCb experiment
Simulation

Simulated samples play a key role for the whole development of the upgraded detector. They are also fundamental for **physics analysis**, contributing to the precision of physics measurements.

Particle simulation
Gauss

Digitization
Boole

Generation
Pythia

Particles directly producing in ***pp*** **collisions**

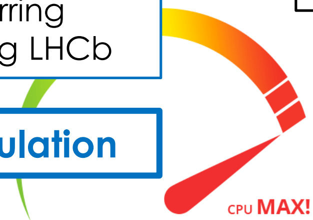
Decay
EvtGen

Heavy flavour decays including dynamics (matrix elements)

Propagation
Geant4

All **radiation-matter interactions** occurring traversing LHCb

Full Simulation

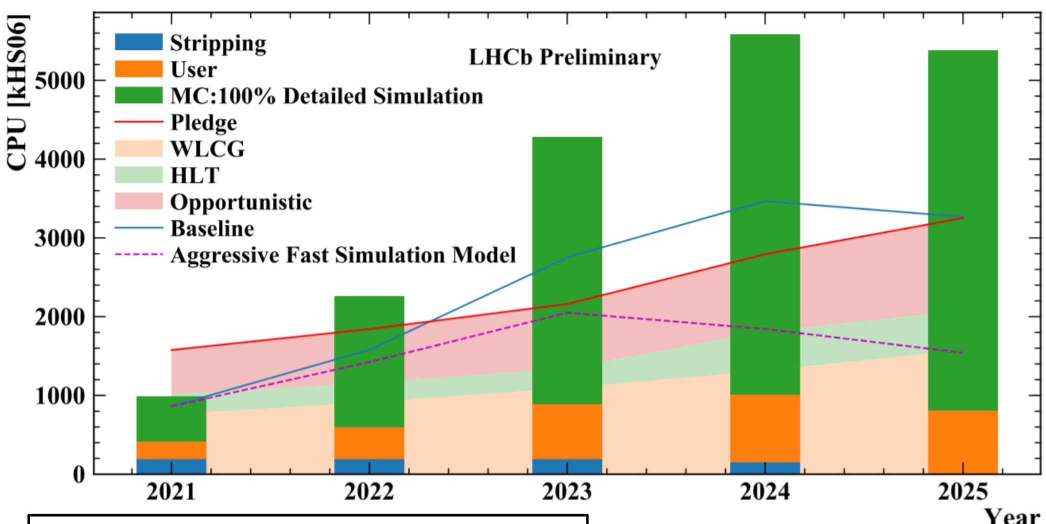


Boole produces a data format equal to the one of true data. The output is then ready to be injected to the rest of the **LHCb software**

The LHCb experiment
Computing requirements

Since the upgraded Trigger will allow to increase the integrated luminosity by **a factor ten**, also the simulation processing needs a similar improvement in order to achieve the highest possible physics accuracy.

x 10
simulated samples size



Full / Fast / Ultra-Fast simulation

Full approach
100% / 0% / 0%

Baseline
40% / 40% / 20%

Aggressive fast model
30% / 50% / 20%

Despite the new Trigger will consume fewer computing resources, pursuing a **Full Simulation** approach is unsustainable with respect to the computing budget available. Adopting faster simulation options will be necessary:

- Fast Simulation
- Ultra-Fast Simulation

The LHCb experiment
Fast Simulation

Simulating the **propagation** of the generated particles through the detector is the hardest step of the entire simulation chain, consuming more than 90% of the event production time.

Faster solutions can be obtained reproducing the experimental setup, and hence the radiation-matter interactions, only partially:

- disabling specific processes (e.g. Cherenkov effect)
- simplifying the geometry (e.g. removing sub-detectors)
- re-using the underlying event (e.g. soft QCD processes)

Similar approaches allow the LHCb Collaboration to meet the demands of Physics Working Group for specific analyses, and are named **Fast Simulation**.

The LHCb experiment
Ultra-Fast Simulation

The main difference between the Full and Fast simulations is that the former computes <u>all</u> the **radiation-matter interactions** within the detector, while the latter <u>only a part</u> of them.

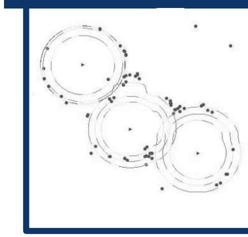
Ultra-Fast Simulation speeds up further the production of simulated samples, <u>renouncing</u> to reproduce the radiation-matter interactions, and <u>parameterizing</u> directly the **high-level** response of each sub-detectors.

The effect of the detector and of the reconstruction algorithms is encoded in <u>parametric</u> formulas or <u>non-parametric</u> predictions.

Among non-parametric solutions, methods based on **Generative Adversarial Networks** (GAN) have proved to be very promising.

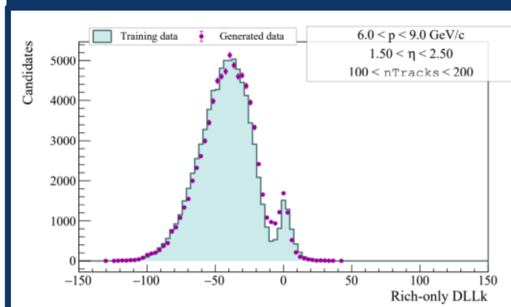
Full Simulation

RICH response



Ultra-Fast Simulation

RICH Differential Log-Likelihood



Generative Adversarial Networks

Generative Adversarial Networks
Introduction

GANs are a powerful class of deep generative models based on the simultaneous training of two neural networks:

- **Generator network** (*G*) that produces synthetic data given some noise source, belonging to the latent space
- **Discriminator network** (*D*) that distinguishes generator's output from true data, representing the reference space

We want that *D* optimally discriminates on the origin of the two samples. Simultaneously the training procedure for *G* is to maximize the probability of *D* making a mistake.

This framework corresponds to a **minimax two-player game**.

I. J. Goodfellow *et al.*, "Generative Adversarial Networks", arXiv:1406.2661.

Generative Adversarial Networks
Minimax two-player game

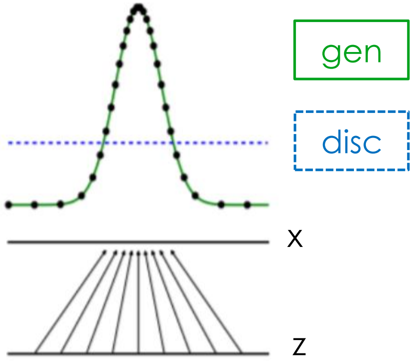
Defining the **loss function** *V(D,G)* as follows

$$V(D,G) = \mathbb{E}_{x\sim P_r}[\log D(x)] + \mathbb{E}_{z\sim P_z}[\log(1 - D(G(z)))]$$

the **minimax game** can be written in this form:

$$\boxed{\min_G \max_D V(D,G)}$$

A unique solution exists, with G recovering the training data distribution and *D* equal to ½ everywhere.



I. J. Goodfellow *et al.*, "Generative Adversarial Networks", arXiv:1406.2661.

Generative Adversarial Networks
Training problems

GANs suffer from many issues, particularly during training:

- generator **collapsing** to produce only a single sample or a small family of very similar samples
- generator and discriminator **oscillating** during training rather than converging to a fixed point
- if **imbalance** between the two neural networks occurs, the system is unable to learn

The training process is driven by the minimax game that represents an optimization problem, and hence it is necessary to compute the **gradient** with respect to the loss function.

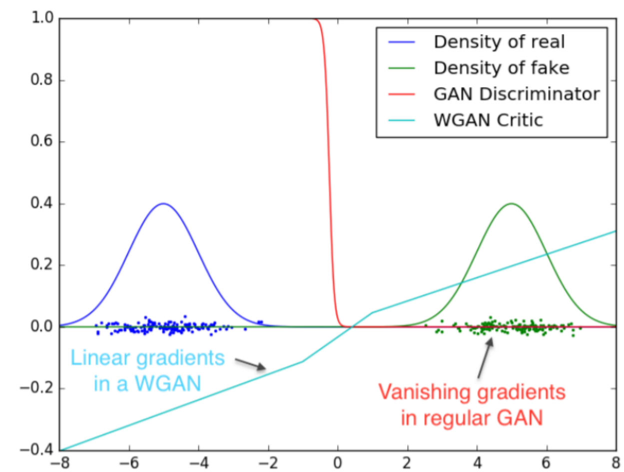
All the drawbacks listed above follow from the saturation of the discriminator: *D* is so good in distinguishing the origin of the two samples that *G* cannot learn anything because of the **vanishing gradient**.

M. Arjovsky and L. Bottou, "Towards Principled Methods for Training Generative Adversarial Networks", arXiv:1701.04862.

Generative Adversarial Networks
Wasserstein GAN

To avoid the vanishing gradient problem, one should change the loss function, preferring a **metric** capable to measure the distance between two distributions.

A similar metric is the **Wasserstein distance** which can provide the generator with information even when the reference space is separated from the generated one (saturation conditions).



Using the Wasserstein distance $W(C,G)$, the discriminator D is replaced by the critic C, a **1-lipschitz function** which prevents from saturation:

$$W(C,G) = \mathbb{E}_{x \sim P_r}[C(x)] - \mathbb{E}_{z \sim P_z}[C(G(z))]$$

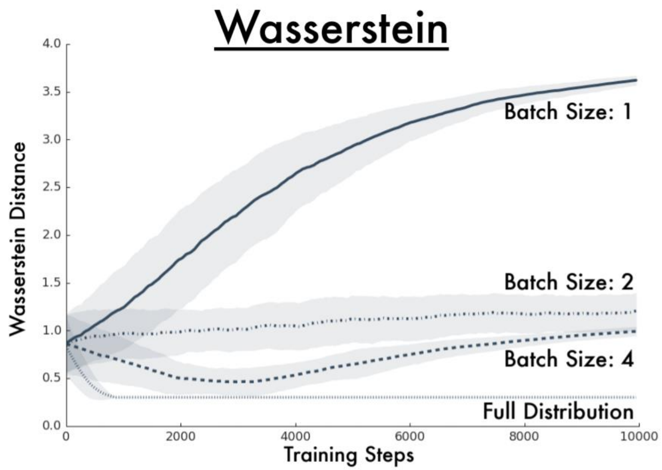
Then, the **minimax game** becomes:

$$\boxed{\min_{G} \max_{C \in \mathcal{C}} W(C,G)}$$

M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN", arXiv:1701.07875.

Generative Adversarial Networks
Unbiased sample gradients

Solving the minimax game requires to compute several gradients. Hence, evaluating such gradients over the entire training sample is <u>inefficient</u> and often <u>impractical</u>. What is typically done is to compute the various gradients over small subsets named **mini-batches**.



A crucial feature in training GAN is that the loss function value <u>does not depend on</u> the batch size chosen to compute gradients: the **unbiased sample gradients** condition.

Despite its good properties, the Wasserstein distance has gradients that <u>depend on</u> the mini-batch choice.

To ensure the unbiased sample gradients condition, one should use the **Cramér distance**, a metric similar to *W(C, G)* but with <u>stricter hypothesis</u> for the critic *C*.

M. G. Bellemare *et al.*, "The Cramer Distance as a Solution to Biased Wasserstein Gradients", arXiv:1705.10743.

Generative Adversarial Networks
Cramér GAN

Using the **energy distance** (multivariate generalization of the Cramér distance) as loss function, one can make <u>more stable</u> the training process, and ensure that gradients are <u>independent</u> of the batch size.

The energy distance is defined as follows

$$\mathcal{E}(f, G) = \mathbb{E}_{x \sim P_r}[f_h(x)] - \mathbb{E}_{z \sim P_z}[f_h(G(z))]$$

Where the critic f_h is a function <u>absolutely continuous</u> with gradient norm **less than one**:

$$f_h(a) = \mathbb{E}_{z' \sim P_z}\left[\, \|h(a) - h(G(z'))\|_2 \,\right] - \mathbb{E}_{x' \sim P_r}\left[\, \|h(a) - \cancelto{\approx 0}{h(x')}\|_2 \,\right]$$

The map h is the output of the <u>discriminator network</u>, one can derive the critic from. Then, the **minimax game** becomes:

$$\boxed{\min_{G} \max_{h \in \mathcal{H}} \mathcal{E}(f, G)}$$

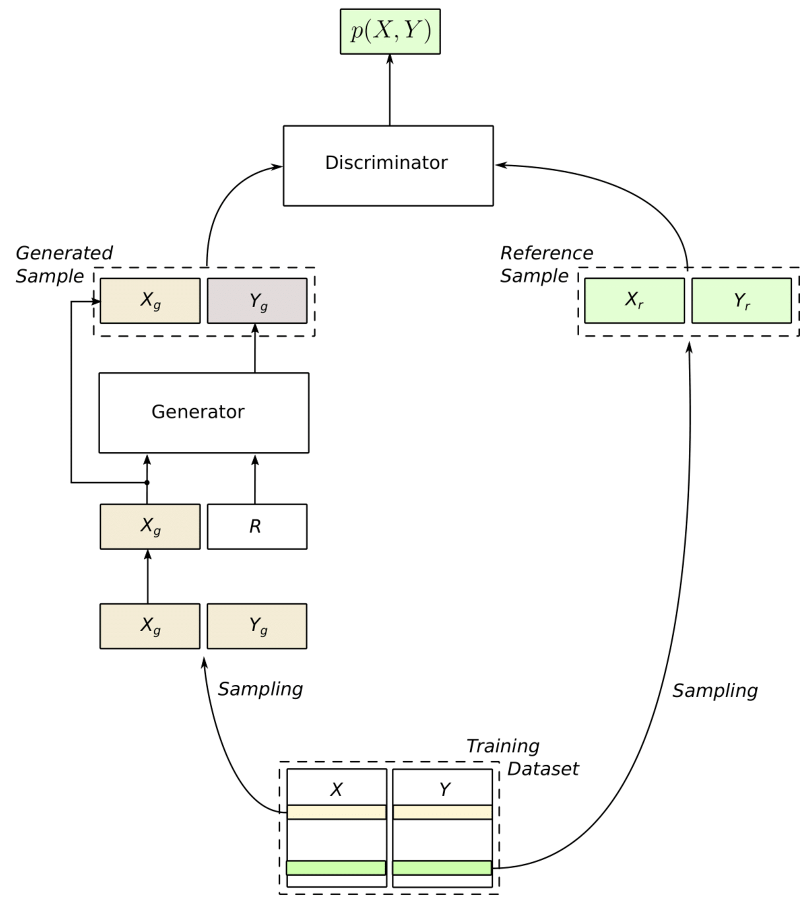
M. G. Bellemare *et al.*, "The Cramer Distance as a Solution to Biased Wasserstein Gradients", arXiv:1705.10743.

Generative Adversarial Networks
Conditional GAN

GANs provide an easy extension to the **conditional form**. Considering a training sample composed by multi-variable elements, we can imagine to split the variables into:

- variables whose distributions are the goal of the generator (*Y*)
- variables that simply conditions the generator outputs (*X*)

The generator task remains that of reproducing synthetic data, but now the **conditional space** is joint to the latent one (noise source *R*) in order to pursue this goal.

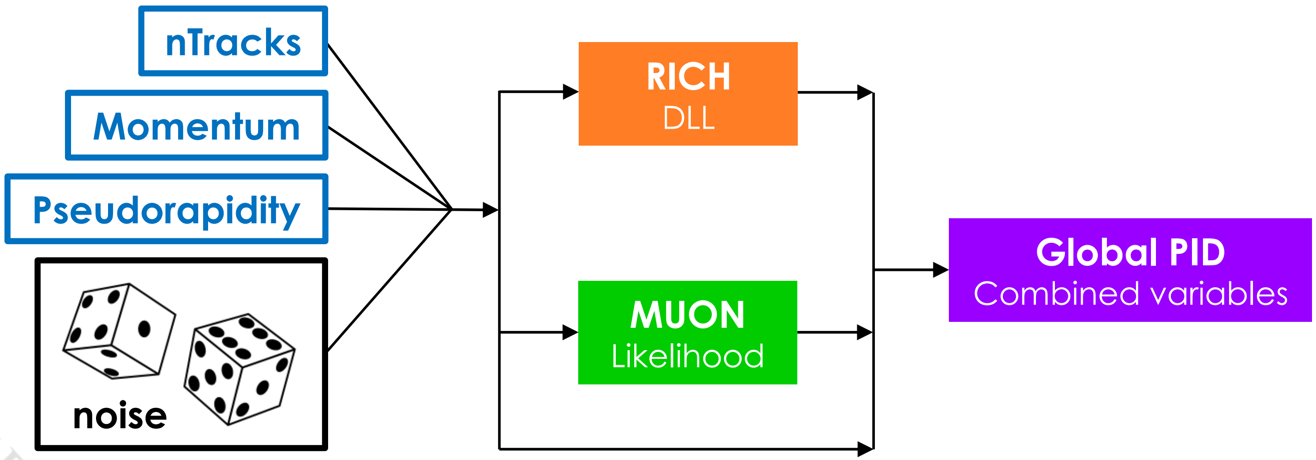


M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets", arXiv:1411.1784.

GANs for Particle Identification
Generative models

The generator network can be used effectively for simulation, modelling the **high-level** response of the Particle Identification system of LHCb.

We expect that the PID response depends on the kinematics of the traversing particles and on the detector occupancy. Hence, this information must be provided to the **generative models** (conditional GANs) in order to produce faithful synthetic sample.



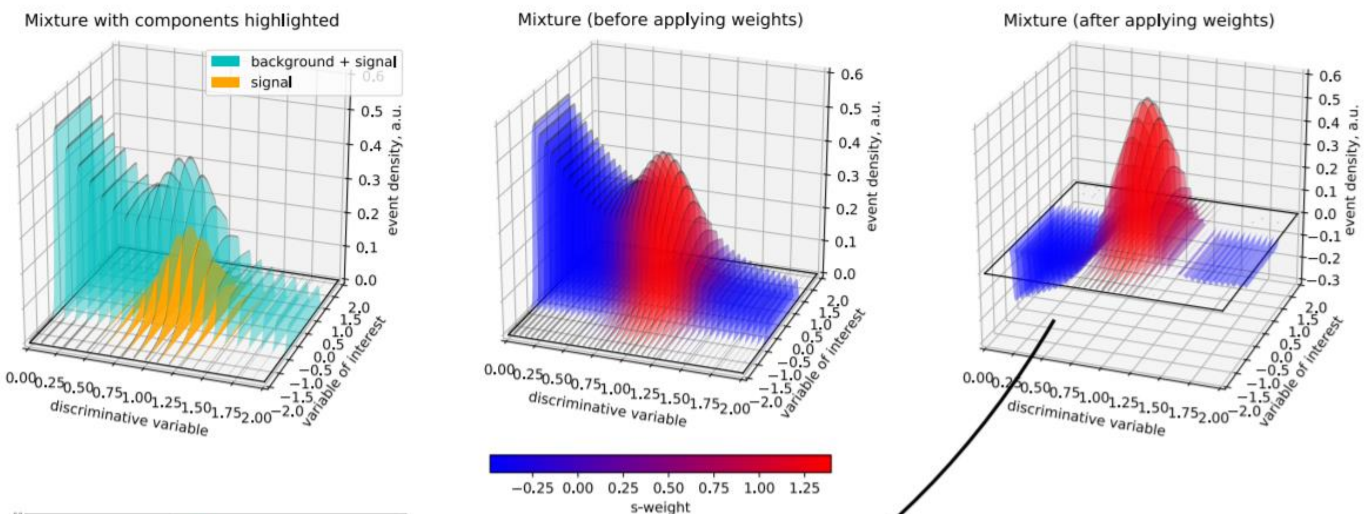
GANs for Particle Identification
Training data

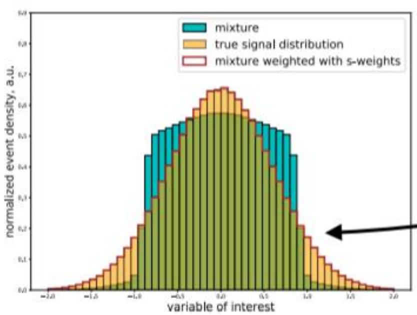
We also expect that each PID sub-detector behaves **differently** for the various species of <u>long-lived particles</u> (μ, π, K, p). Hence, it is necessary to build generative models for each particle in order to parameterize the different behaviours.

Moreover, such generative models should be trained over a data sample containing <u>only</u> the corresponding particle species. To this end, the training procedure is performed using **calibration samples** collected in 2016. Despite the calibration data are selected by <u>exclusive trigger lines</u>, this samples can still have some **residual background** that <u>should be removed</u> to allow the generator to correctly model the PID system.

Lastly, to stabilize the training process, all the generative models should be trained through **Cramér GAN** systems, ensuring <u>non-zero</u> and <u>unbiased</u> gradients.

GANs for Particle Identification
Background subtraction (1/2)





marginalizing out the discriminative variable reconstructs the distribution for the variable of interest

sPlot technique

$$w_n(x_e) = \frac{\sum_{j=1}^{N_s} \mathbf{V}_{nj} \mathrm{f}_j(x_e)}{\sum_{k=1}^{N_s} N_k \mathrm{f}_k(x_e)}$$

M. Pivk and F. R. Le Diberder, "sPlot: A Statistical Tool to Unfold Data Distributions", arXiv:physics/0402083.

GANs for Particle Identification
Background subtraction (2/2)

Consider a dataset characterized by a discriminating variable x and a variable of interest *y*, and composed as the mixture of two components: **signal** and **background**, described by $f_{sig}(x,y)$ and $f_{bkg}(x,y)$.

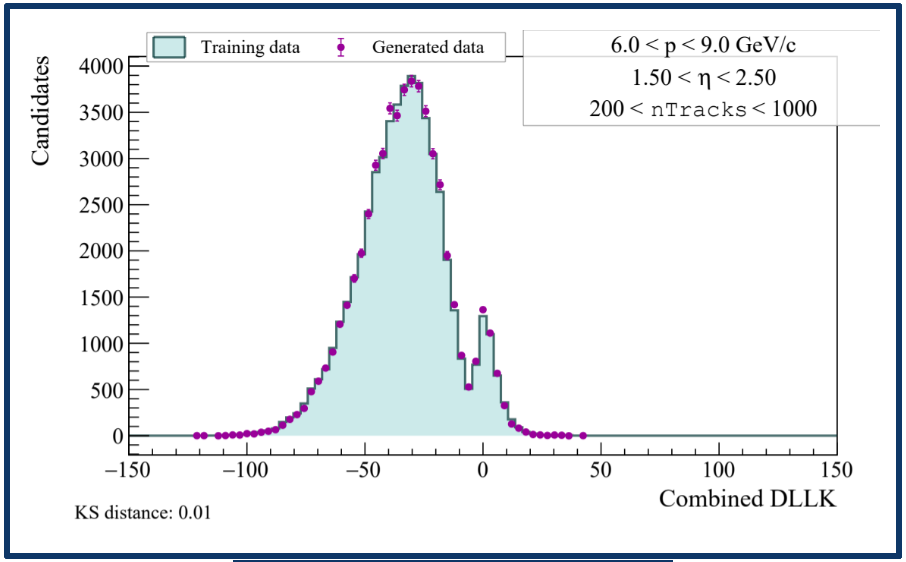
The **sPlot technique** allows to infer the marginal distribution of $f_{sig}(x,y)$ with respect to *y* known the one with respect to x, from which it is possible to extract a set of sWeights.

$$\boxed{\mathbb{E}_{y\sim f^{y}_{\mathrm{sig}}}[\mathcal{F}] = \frac{\sum_i w^{(i)}\mathcal{F}\left(y^{(i)}\right)}{\sum_i w^{(i)}}} \Longrightarrow \boxed{\mathbb{E}_{\substack{y_r\sim P_r^{\mathrm{sig}} \\ y_g,y'_g\sim P_g^{\mathrm{sig}}}}[\mathcal{E}] = \frac{\sum_i w_r^{(i)} w_g^{(i)} {w'}_g^{(i)}\mathcal{E}\left(y_r^{(i)}, y_g^{(i)}, {y'}_g^{(i)}\right)}{\sum_i w_r^{(i)} w_g^{(i)} {w'}_g^{(i)}}}$$

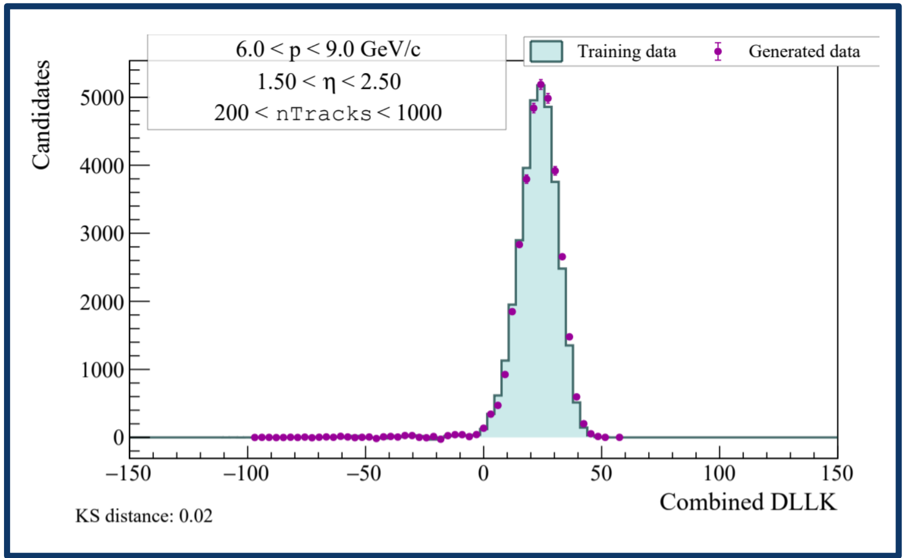
Given a set of contaminated data and a generic expression *F* that depends on the variable of interest, using **sWeights** allows to extract the signal contribution of *F* on average.

A similar strategy allows to train GAN systems over data samples with **residual background**: it is enough to replace the generic expression with the loss function chosen, such as the energy distance ε.

GANs for Particle Identification
Model validation



Pion track candidates

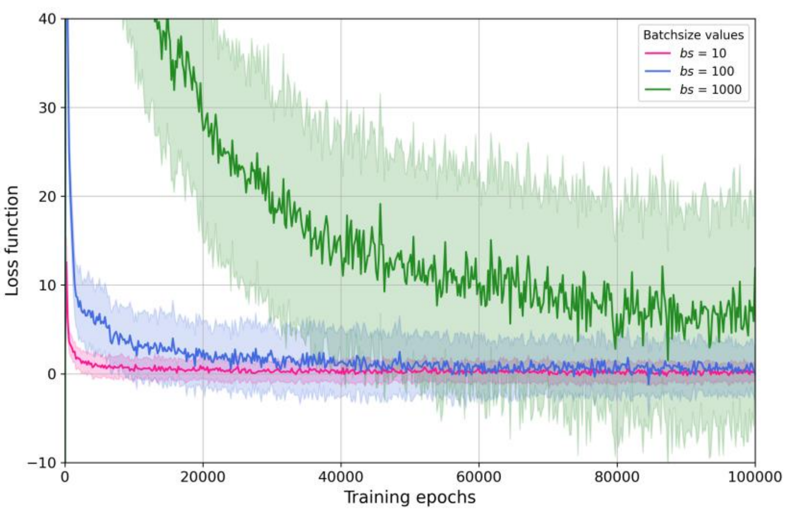


Kaon track candidates

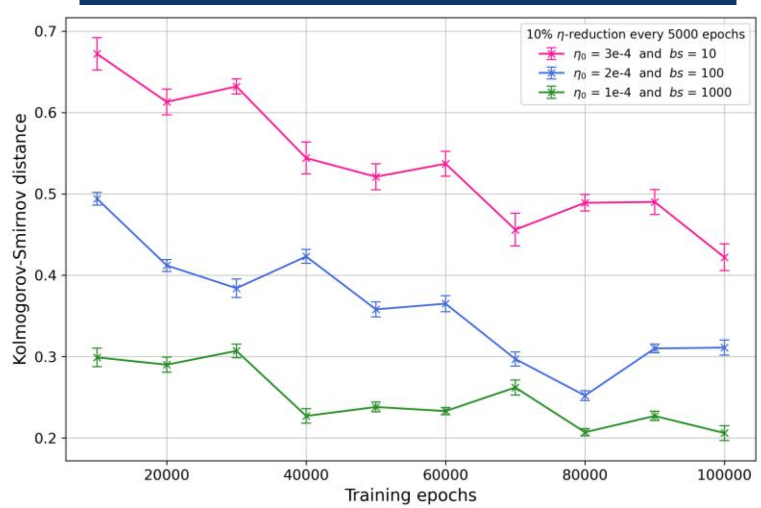
GANs for Particle Identification
Generated sample quality

Even if the loss function measures the distance between the target distributions and the generated ones, it cannot be used to validate the models since its judgement is clouded by the competition between the two players of the **minimax game**.

Introducing a third **independent** player is the solution: the idea is to exploit the output of a **robust** algorithm trained to distinguish the reference data from the synthetic one.



Score values for three different models



Failed: KS = 1.00

Quality of the **Generative Model**

Perfect: KS = 0.00

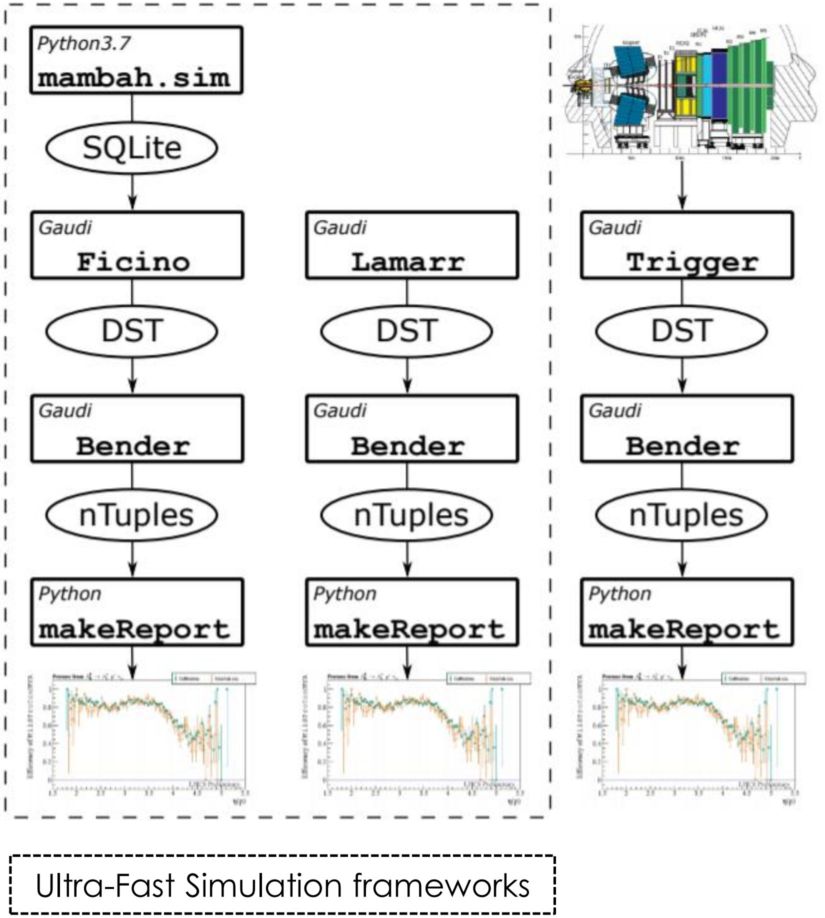
Integration of GAN models within the LHCb Simulation

The LHCb experiment
Ultra-Fast Simulation frameworks

All the simulation software, in order to be used by the Collaboration, should be interfaced with **Gaudi**, a software framework developed to allow running LHCb applications within a highly parallel environment.

Lamarr, the official Ultra-Fast Simulation framework of LHCb, directly produces simulated samples in the correct data format for LHCb applications.

Mambah is a generic framework designed for High Energy Physics applications. It needs an external software (**Ficino**) to convert its databases into LHCb data formats.



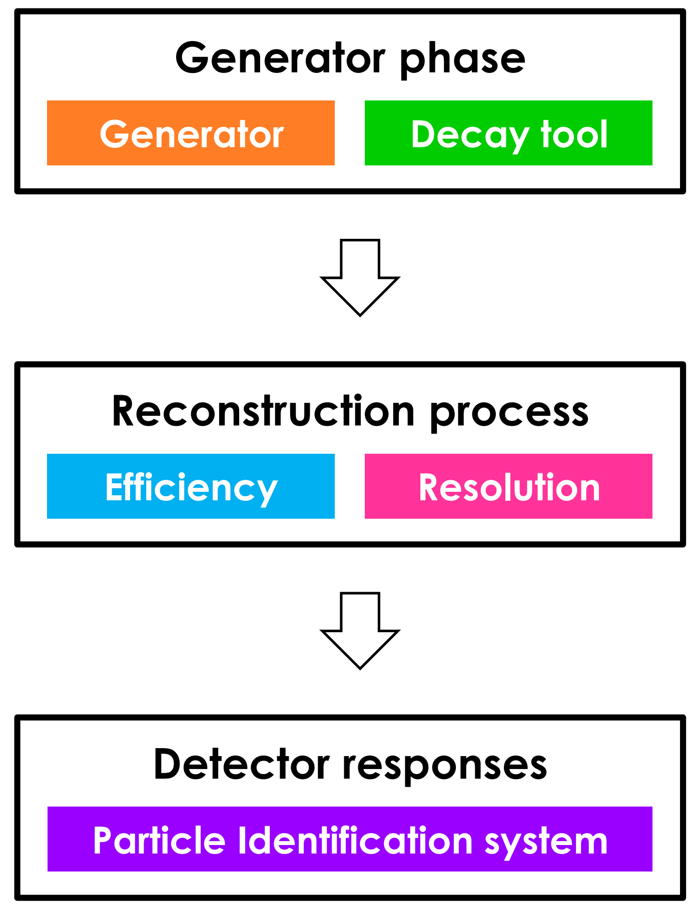
The Mambah framework
The `mambah.sim` package

Mambah is a new <u>python</u> framework designed with a **batch-grained** paradigm and represented within **relational databases**.

Mambah provides a <u>complete set</u> of functions **to access** the particles and the vertices composing the decay trees in the event.

<u>Data organization</u>, <u>database management</u> and <u>algorithm configuration</u> make Mambah the perfect starting point to build an efficient simulation framework named **`mambah.sim`**.

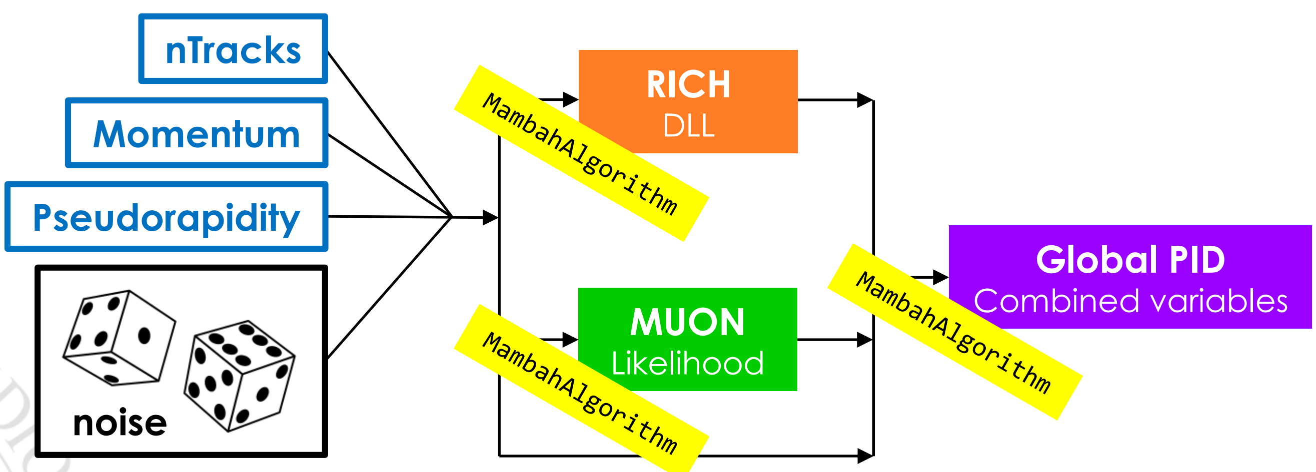
The `mambah.sim` module provides a set of useful classes and tools to simulate the <u>particle decays</u> together with the <u>detector responses</u>.



The Mambah framework
Particle Identification

The reconstruction process provides the **track kinematics parameters** as "expected" from the tracking system. In addition, `mambah.sim` allows to parameterize the **detector occupancy** (nTracks) extracting a random number from the corresponding distribution for the decay $K_s^0 \to \pi^+\pi^-$.

Finally, the high-level PID responses can be obtained feeding the **generative models** previously trained with the available parameters. It should be noted that Mambah provides powerful methods to compute **efficiently** the following chain of neural networks.



The Mambah framework
Framework validation (1/3)

In order to validate the Mambah framework, a decay channel **different** from the one used to train the generative models was chosen:

$$\Lambda_b^0 \to \Lambda_c^+ \mu^- \bar{\nu}_\mu$$
$$\hookrightarrow \Lambda_c^+ \to p K^- \pi^+$$

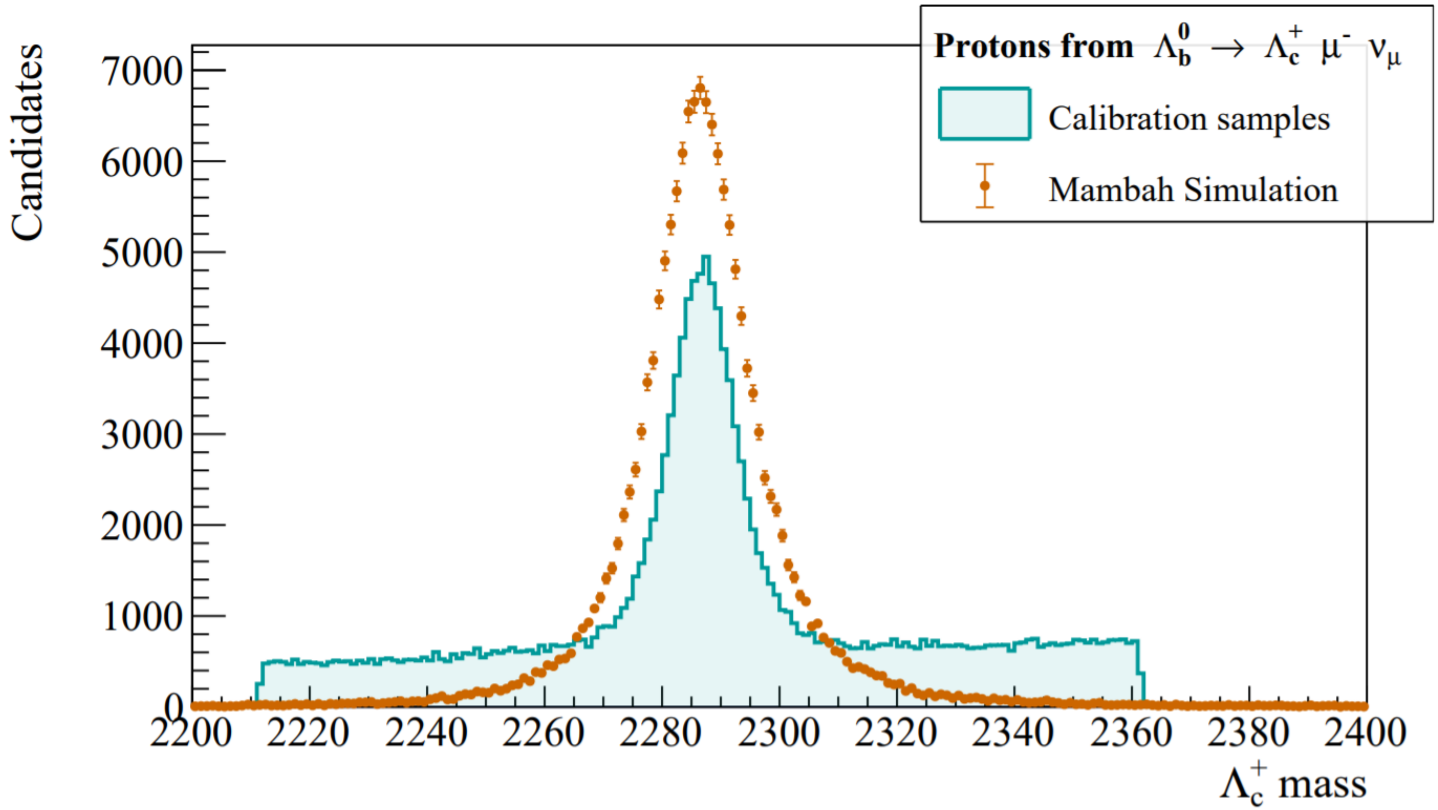
It is a non-trivial decay sensitive to several aspects of the simulation:

- semileptonic decay whose dynamic should be treated by EvtGen
- decay channel contains **all** the charged stable particles that need parameterizations
- decay channel belonging to **calibration data** to select pure samples of proton
- decay channel **different** from the ones used to train GAN systems

	Validation		Training
Proton:	$\Lambda_c^+ \to p K^- \pi^+$	$\neq$	$\Lambda^0 \to p \pi^-$

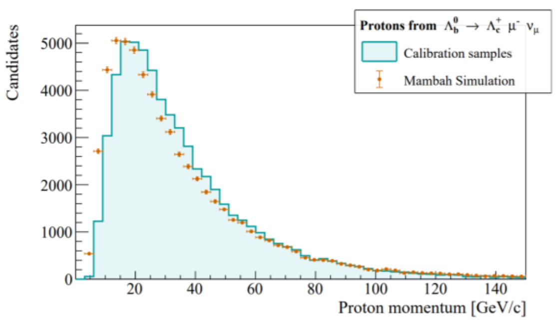
The Mambah framework

Framework validation (2/3)

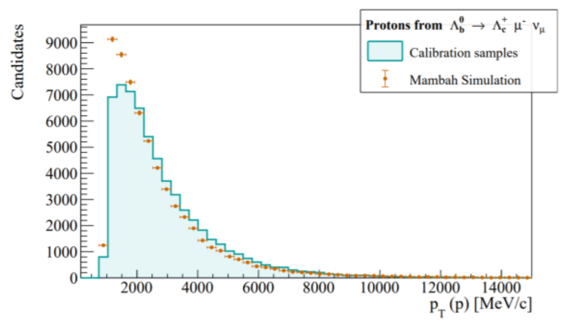


The Mambah framework
Framework validation (3/3)

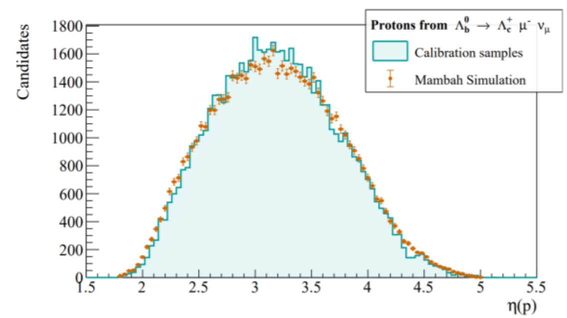
Proton kinematics



Momentum

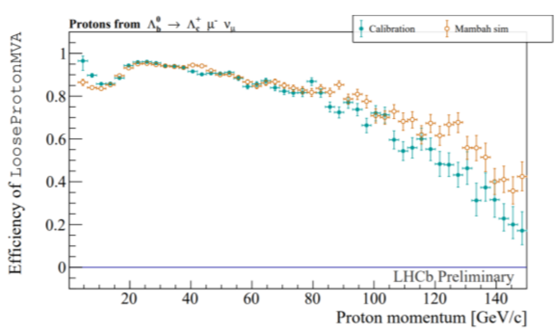


Transverse momentum

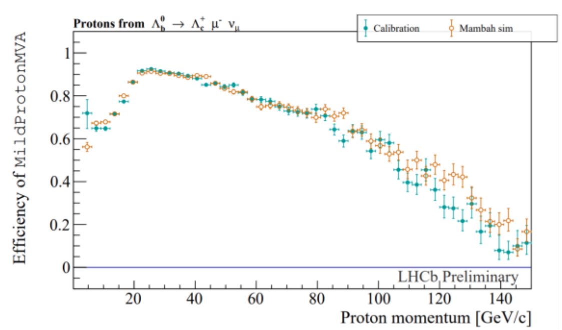


Pseudorapidity

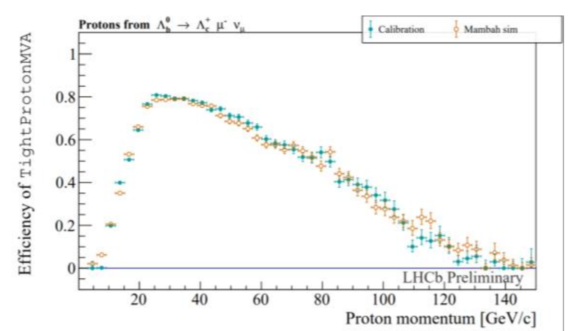
Efficiency of proton PID requirements



Momentum VS
ε [ProbNNp > 0.6]



Momentum VS
ε [ProbNNp > 0.8]



Momentum VS
ε [ProbNNp > 0.95]

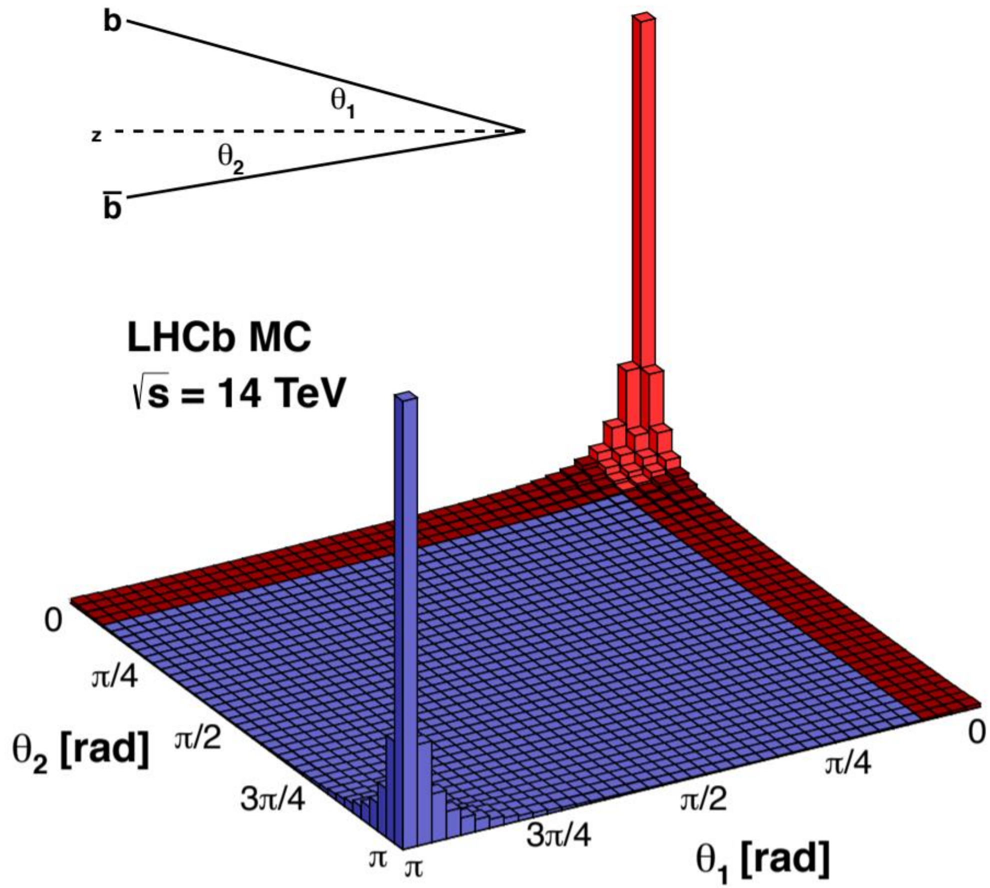
Conclusion

- Starting from Run 3, the LHCb Upgrade detector will increase significantly the collected data, allowing to reach **unprecedent accuracy** in heavy flavour physics studies
- To this end, it is crucial to develop and implement **faster** strategies than the Full Simulation to produce simulated samples
- Among Ultra-Fast Simulation, **GAN systems** have proved to be very promising to parameterize the high-level detector response, especially for the **PID system** of LHCb
- Neural networks trained by a conditional minimax game are able to reproduce effectively the probability distributions of the **Global PID** variables
- These generative models can be used within a new simulation framework named **`mambah.sim`** able to evaluate efficiently several computational graphs and to produce huge simulated samples consuming much less computing resources

THANK YOU

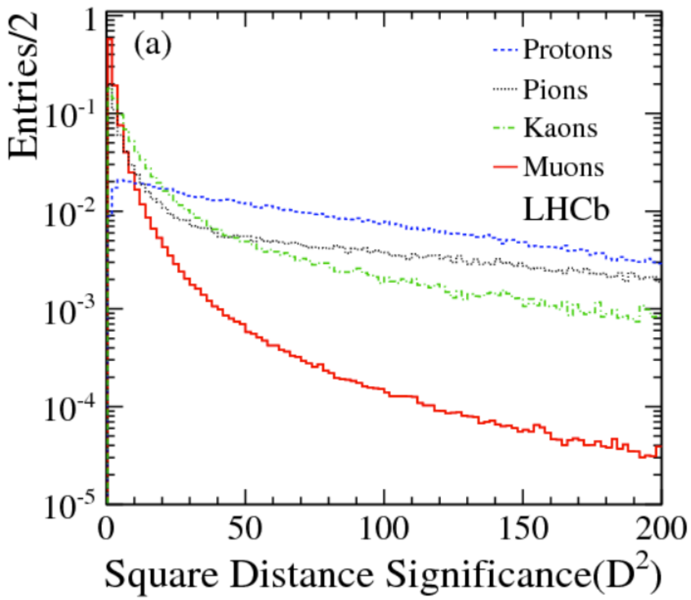
Backup

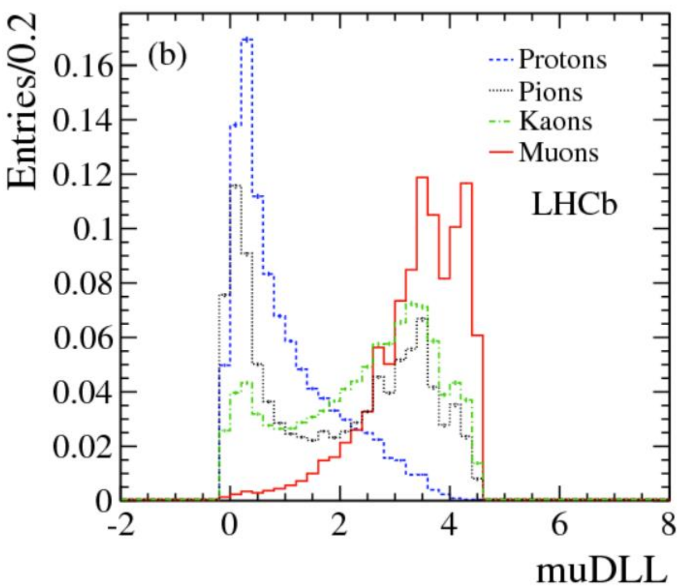
The LHCb experiment
Production angles of b-hadrons



Particle Identification variables
Muon system

$$D^2 = \frac{1}{N}\sum_i \left[\left(\frac{x^{(i)}_{\rm closest} - x^{(i)}_{\rm track}}{pad^{(i)}_x}\right)^2 + \left(\frac{y^{(i)}_{\rm closest} - y^{(i)}_{\rm track}}{pad^{(i)}_y}\right)^2\right]$$





<u>MuonMuLL:</u> the cumulative of the red D^2 distribution

<u>MuonBkgLL:</u> the cumulative of the blue D^2 distribution

$$\text{muDLL} = \log \frac{\texttt{MuonMuLL}}{\texttt{MuonBkgLL}}$$

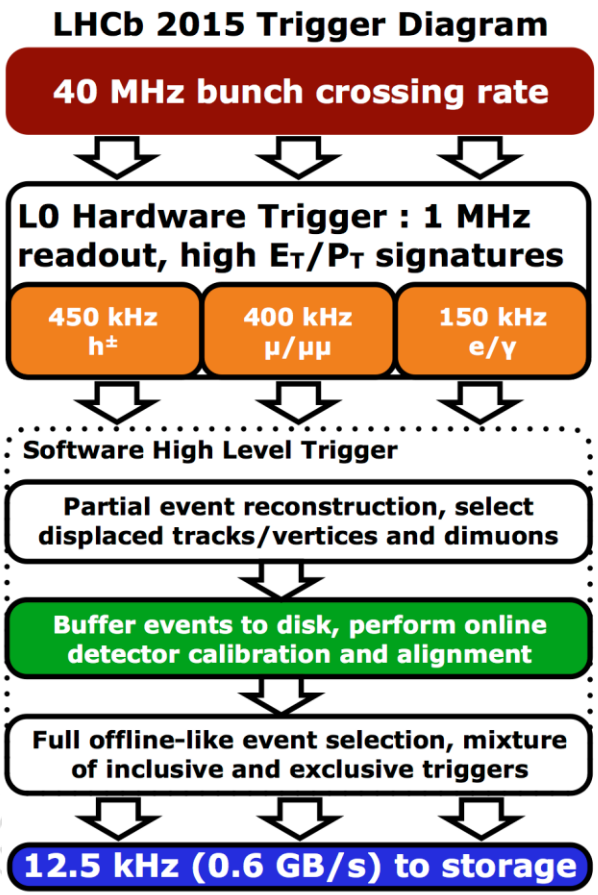
Particle Identification performance
Decay modes of calibration samples

Species	Low momentum	High momentum
$e^{\pm}$	$B^+ \to (J/\psi \to e^+e^-)K^+$	$B^+ \to (J/\psi \to e^+e^-)K^+$
$\mu^{\pm}$	$B^+ \to (J/\psi \to \mu^+\mu^-)K^+$	$J/\psi \to \mu^+\mu^-$
$\pi^{\pm}$	$K_s^0 \to \pi^+\pi^-$	$D^{*+} \to (D^0 \to K^-\pi^+)\pi^+$
$K^{\pm}$	$D_s^+ \to (\phi \to K^+K^-)\pi^+$	$D^{*+} \to (D^0 \to K^-\pi^+)\pi^+$
$p, \bar{p}$	$\Lambda^0 \to p\pi^-$	$\Lambda^0 \to p\pi^-$; $\Lambda_c^+ \to pK^-\pi^+$

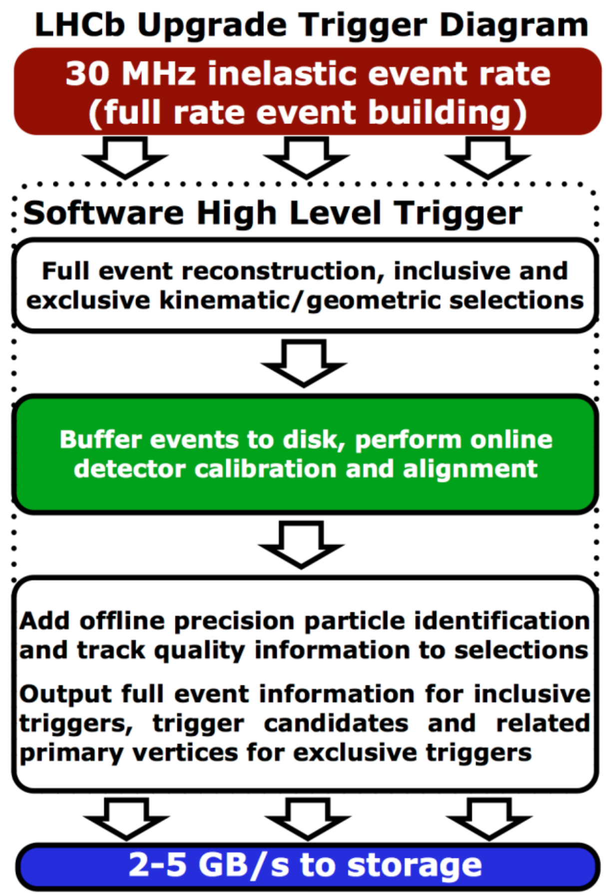
The LHCb experiment

The old and new Trigger

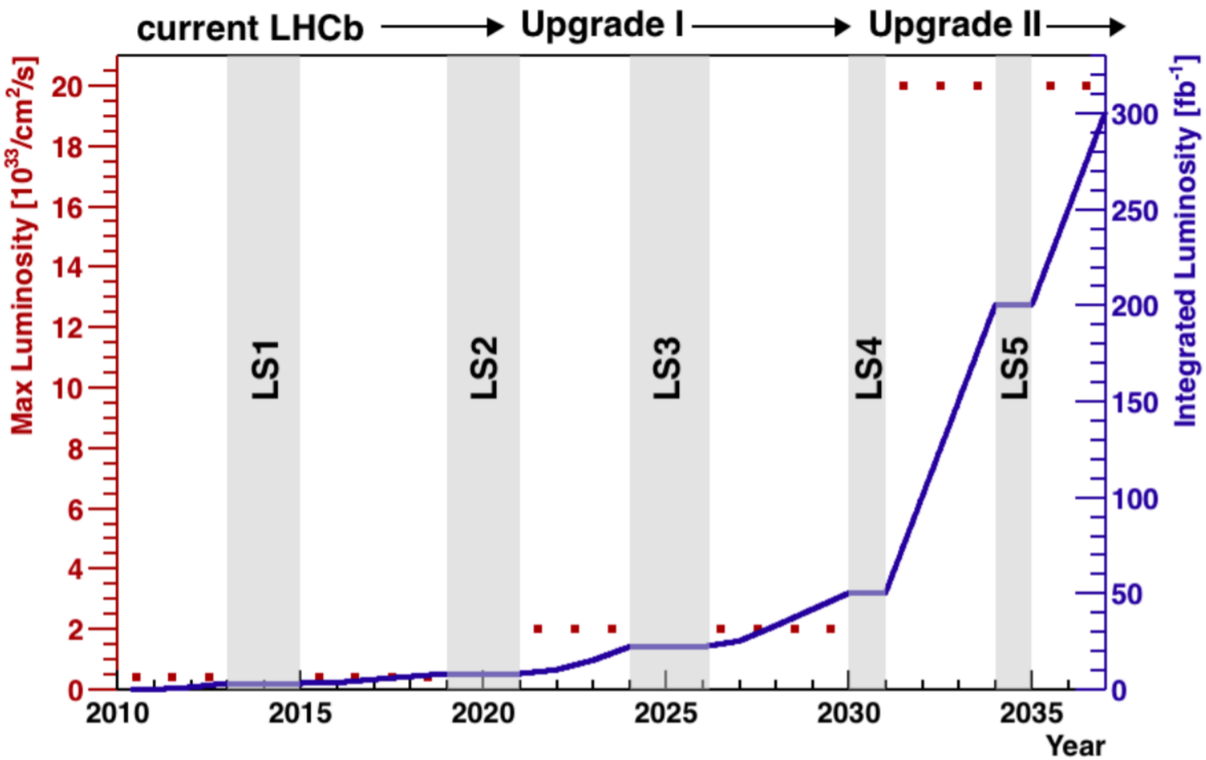
LHC Run 2



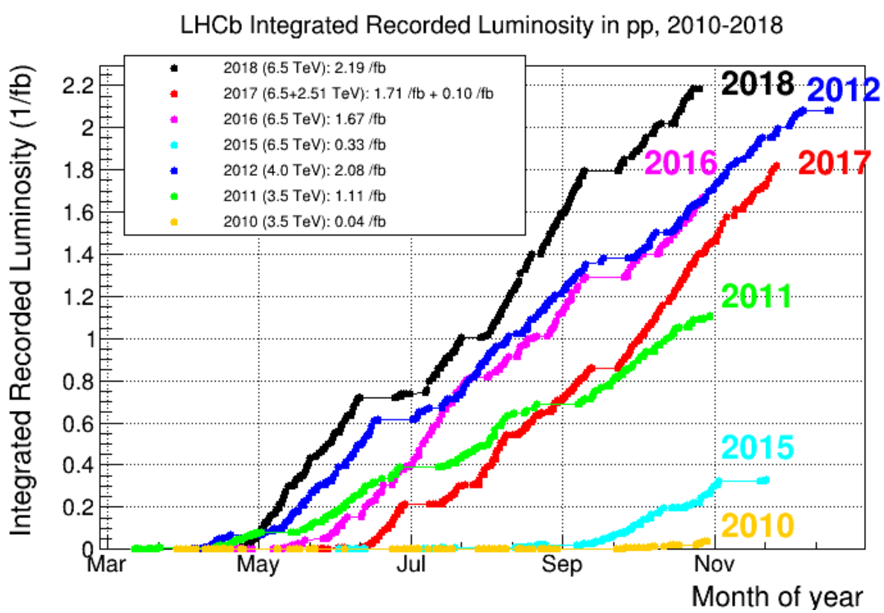
LHC Run 3

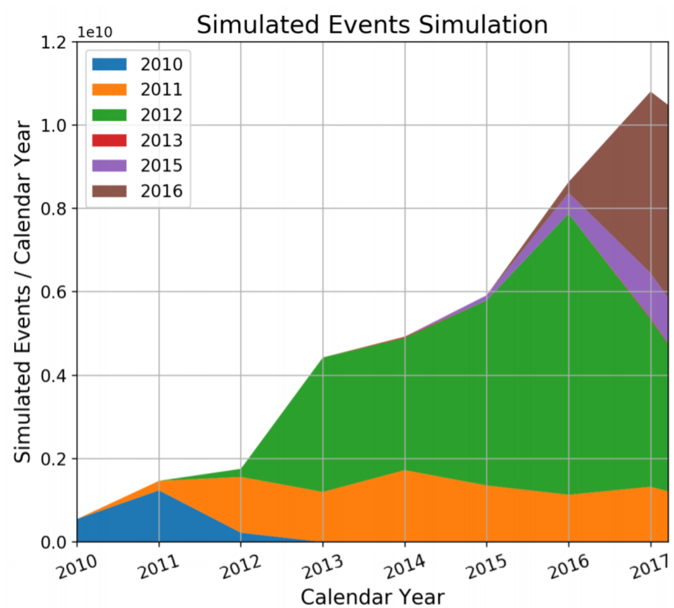


The LHCb experiment
Upgrades

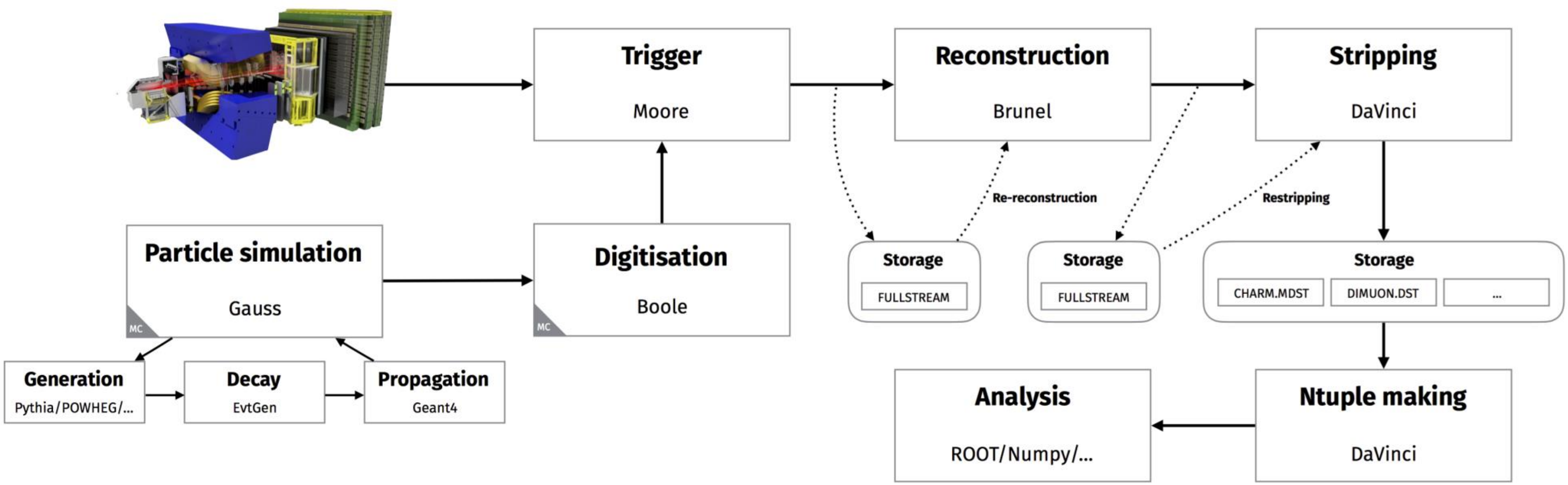


The LHCb experiment
Size of simulated samples

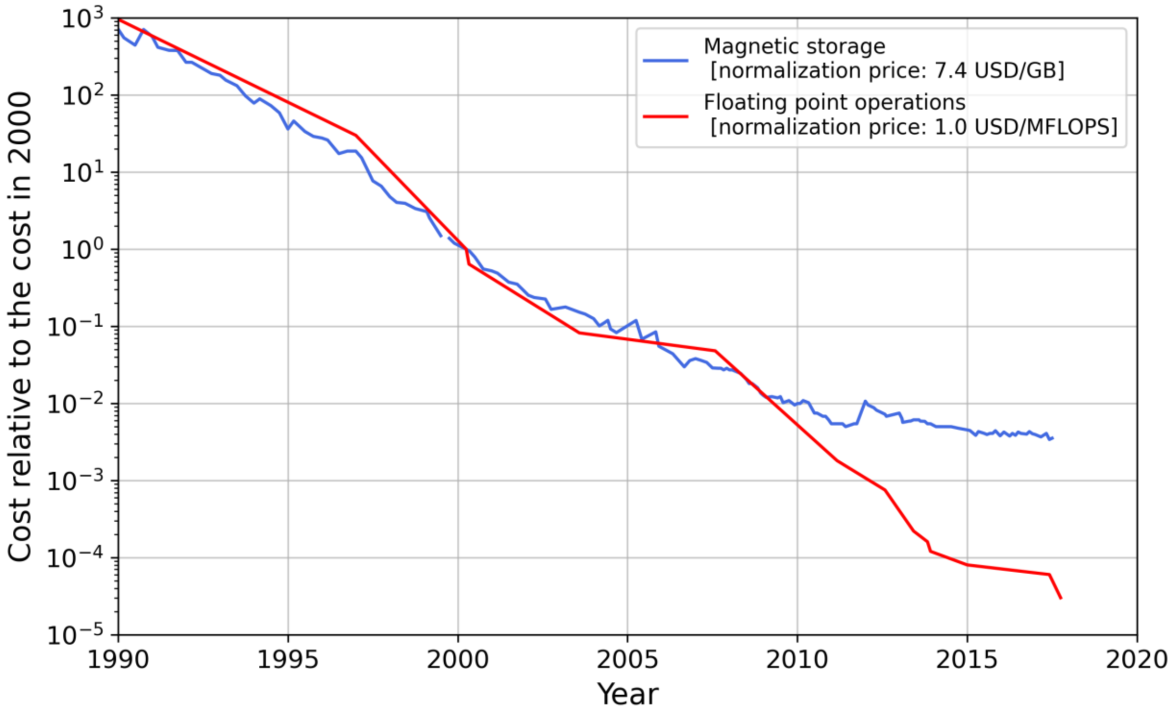




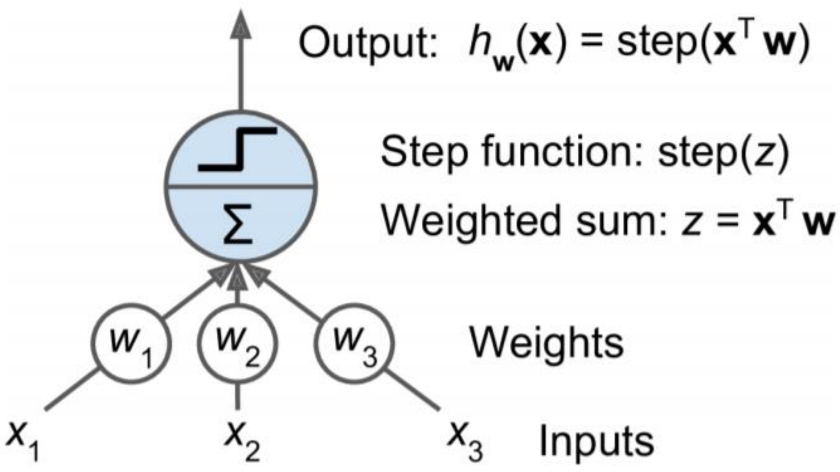
The LHCb experiment
Data flow

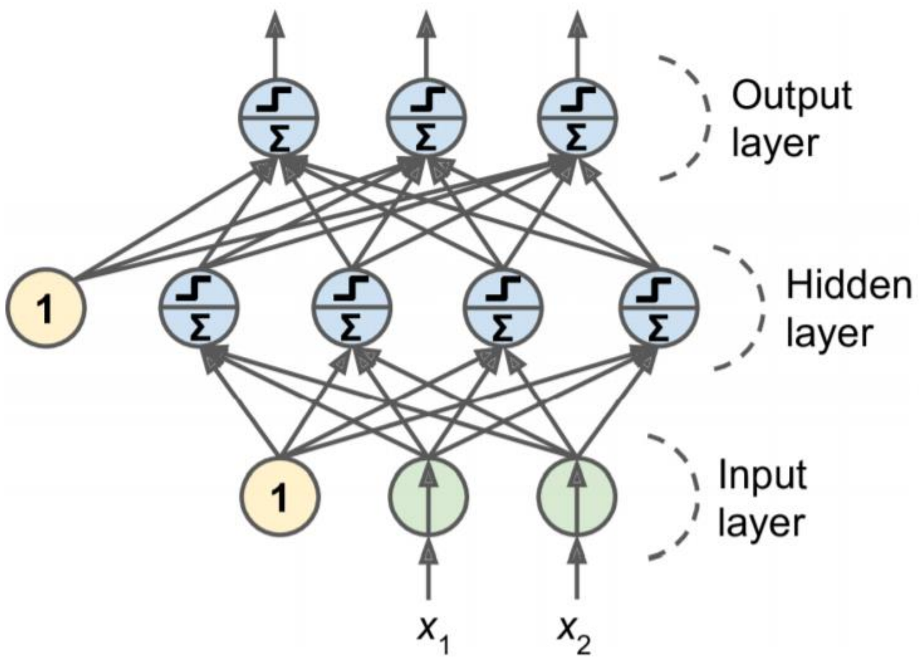


The LHCb experiment
The new simulation paradigm

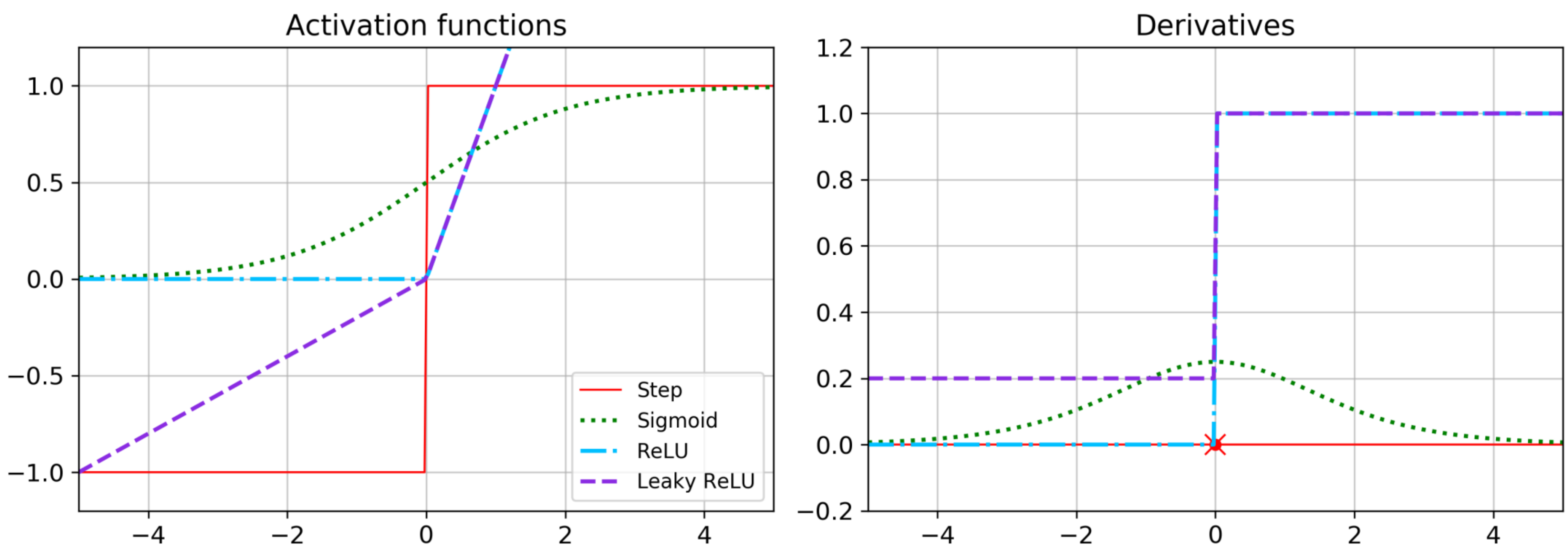


Deep Learning
Perceptron and Multilayer Perceptron





Deep Learning
Activation functions



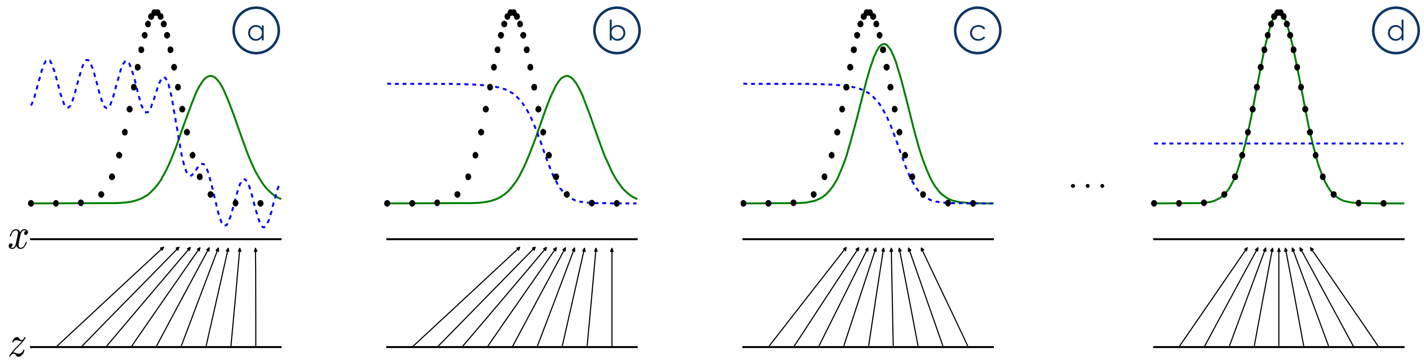
Step: $\mathrm{step}(z) = \mathrm{sign}(z)$

Sigmoid: $\sigma(z) = 1/(1 - \exp(-z))$

ReLU: $f(z) = \max(0, z)$

Leaky ReLU: $f(z; \alpha) = \max(\alpha z, z)$

Generative Adversarial Networks
Pedagogical explanation



a) Minimax game near convergence: P_g is similar to P_r and D is a partially accurate classifier.
b) The D network is trained to **discriminate** samples from data, converging to optimality.
c) After an update of G, gradient of D has driven $G(z)$ to flow to region that are more likely to be classified as data.
d) After several steps of training, they will reach a point at witch both cannot improve because the discriminator is **unable** to differentiate between the two distributions.

I. J. Goodfellow *et al.*, "Generative Adversarial Networks", arXiv:1406.2661.

Generative Adversarial Networks
Optimal discriminator

Solving the minimax game with respect to *D*, we obtain

$$\max_D V(D,G) = V(D^*,G)$$

where *D** indicates the optimal discriminator:

$$D^*(x) = \frac{p_r(x)}{p_r(x) + p_g(x)}$$

It's easily to demonstrate that *V(D*,G)* is related to the **Jensen-Shannon divergence**, as follows

$$V(D^*,G) = -\log 4 + 2 \cdot JS(P_r \| P_g)$$

Then, the minimax game corresponds to minimize the JS divergence:

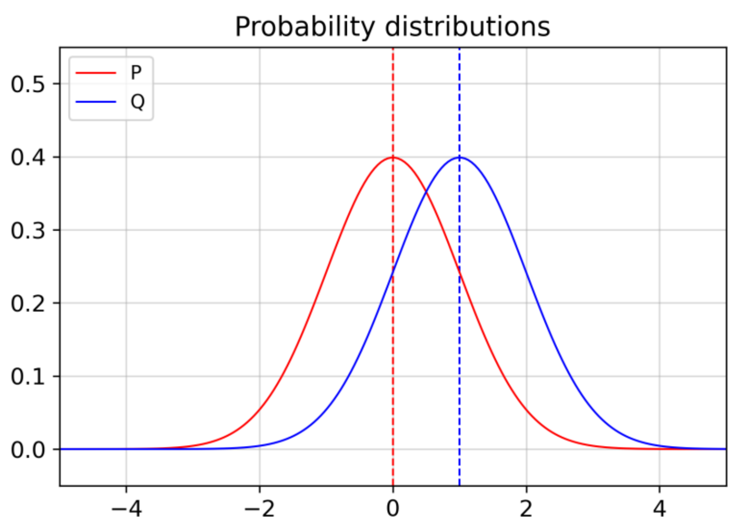
$$\boxed{\min_G \max_D V(D,G) = \min_G JS(P_r \| P_g)}$$

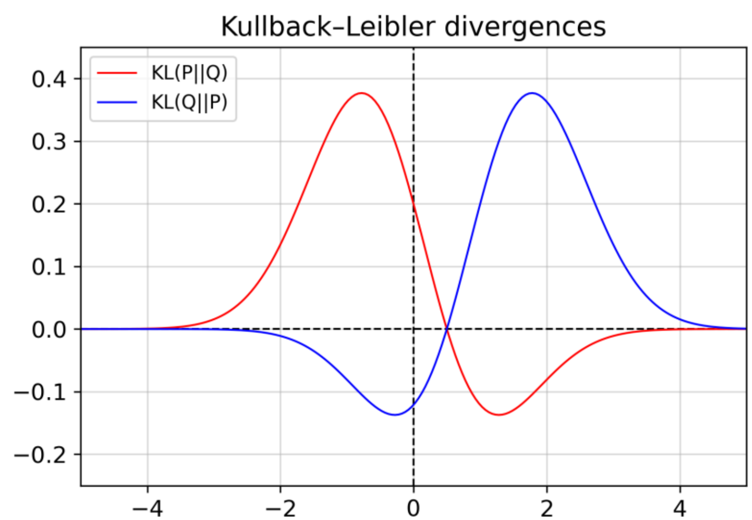
I. J. Goodfellow *et al.*, "Generative Adversarial Networks", arXiv:1406.2661.

Loss function
Kullback-Leibler divergence

The **Kullback–Leibler divergence** (also called **relative entropy**) is a measure of how one probability distribution is different from a second, reference probability distribution.

$$KL(P\|Q) = \mathbb{E}_{x\sim P}[\log P(x)] - \mathbb{E}_{x\sim P}[\log Q(x)]$$

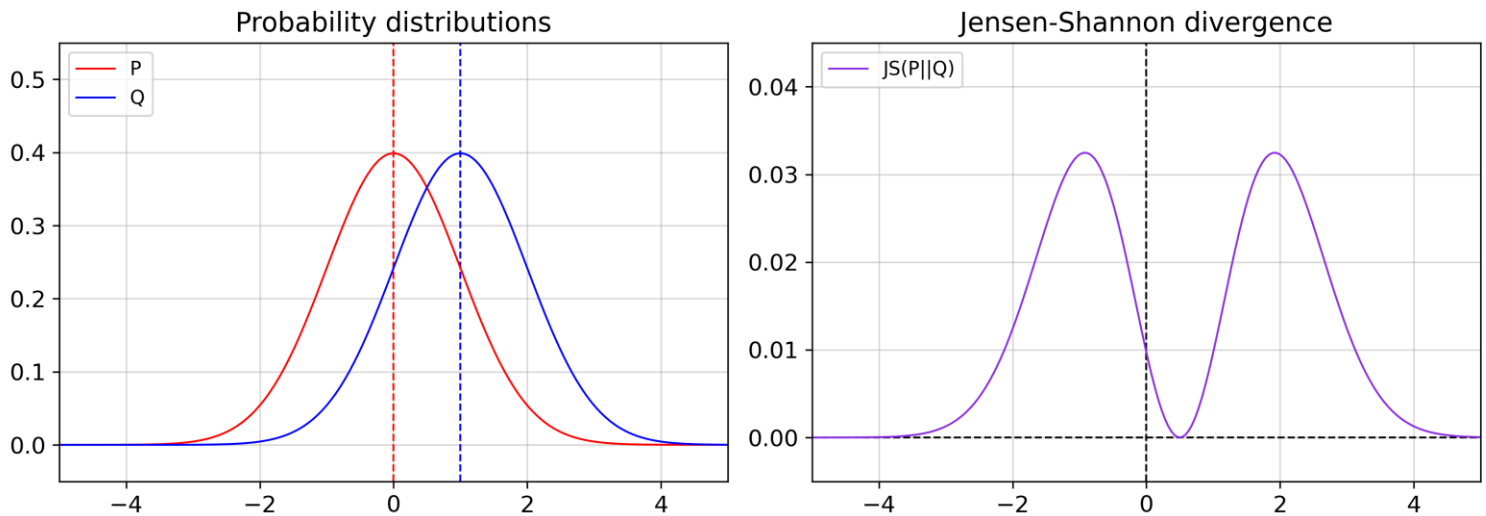




Loss function
Jensen-Shannon divergence

The **Jensen–Shannon divergence** is a method of measuring the <u>similarity</u> between two probability distributions.

$$\boxed{JS(P\|Q) = \frac{1}{2}KL\left(P\middle\|\frac{P+Q}{2}\right) + \frac{1}{2}KL\left(Q\middle\|\frac{P+Q}{2}\right)}$$



Generative Adversarial Networks
Perfect discriminator

Empirically, if we train D till convergence, the JS divergence between P_r and P_g is maxed out. The only way this can happen is if the supports of distributions are **disjoint** or lie in **low dimensional** manifolds. In these hypothesis we can demonstrate that a perfect discriminator always exists.

PERFECT DISCRIMINATOR

$$\boxed{\begin{array}{l} D : \mathcal{X} \to [0,1] \\ P_r[D(x) = 1] = 1 \\ P_g[D(x) = 0] = 1 \end{array}}$$

A perfect discriminator has **zero gradient** almost everywhere on the union of sets containing P_r and P_g supports.

M. Arjovsky and L. Bottou, "Towards Principled Methods for Training Generative Adversarial Networks", arXiv:1701.04862.

Generative Adversarial Networks
Vanishing gradient

Typically, the divergences which GANs minimize are not continuous with respect to generator's parameters θ. This allows the existence of the perfect discriminator D^* for which the gradient on the generator **vanishes**. If we consider an approximation D that distances ε from D^*, we can prove what follows:

$$\boxed{\lim_{\|D-D^*\|\to 0} \nabla_\theta \mathbb{E}_{z\sim P_z}[\log(1-D(G_\theta(z)))] = 0}$$

As our discriminator gets better, the gradient of the generator vanishes. In other words, either our updates to the discriminator will be inaccurate, or they will vanish.

M. Arjovsky and L. Bottou, "Towards Principled Methods for Training Generative Adversarial Networks", arXiv:1701.04862.

Generative Adversarial Networks

Noise insertion

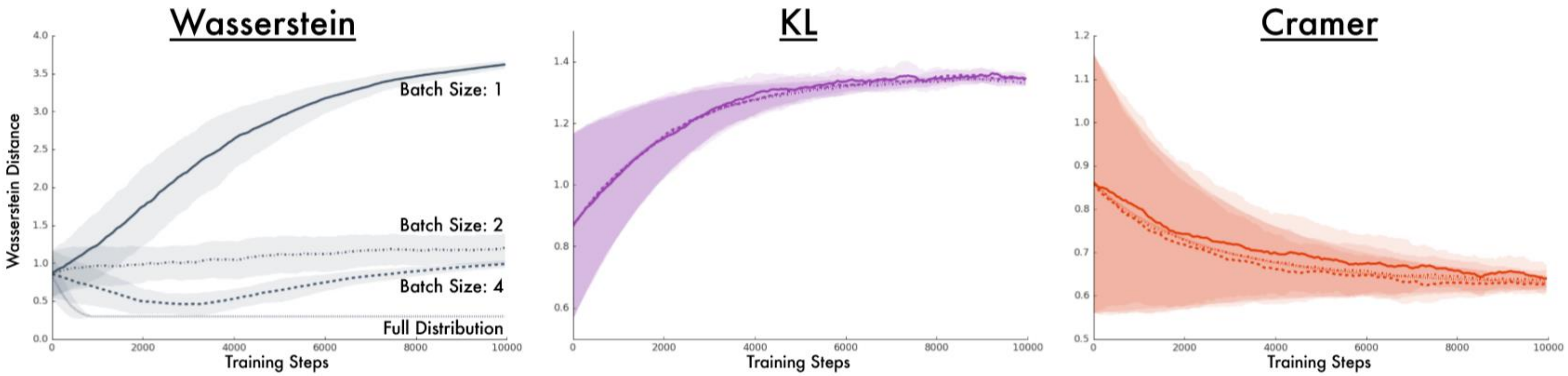
There is something we can do to break our gradient problem: adding continuous **noise** to both discriminator and generator. This move allows to learn thanks to **non-zero gradient** of the generator. However, it's now proportional to the gradient of noisy JS divergence:

$$\boxed{\mathbb{E}_{z\sim P_z,\varepsilon'}\left[\nabla_\theta \log(1 - D^*_\varepsilon(G_\theta(z)+\varepsilon'))\right] = 2\cdot\nabla_\theta JS\left(P_{r+\varepsilon}\|P_{g+\varepsilon}\right)}$$

This variant of JS divergence measures a similarity between the two noisy distribution and isn't an intrinsic measure of P_r and P_g. Luckily, using **Wasserstein metric** we can solve this problem.

M. Arjovsky and L. Bottou, "Towards Principled Methods for Training Generative Adversarial Networks", arXiv:1701.04862.

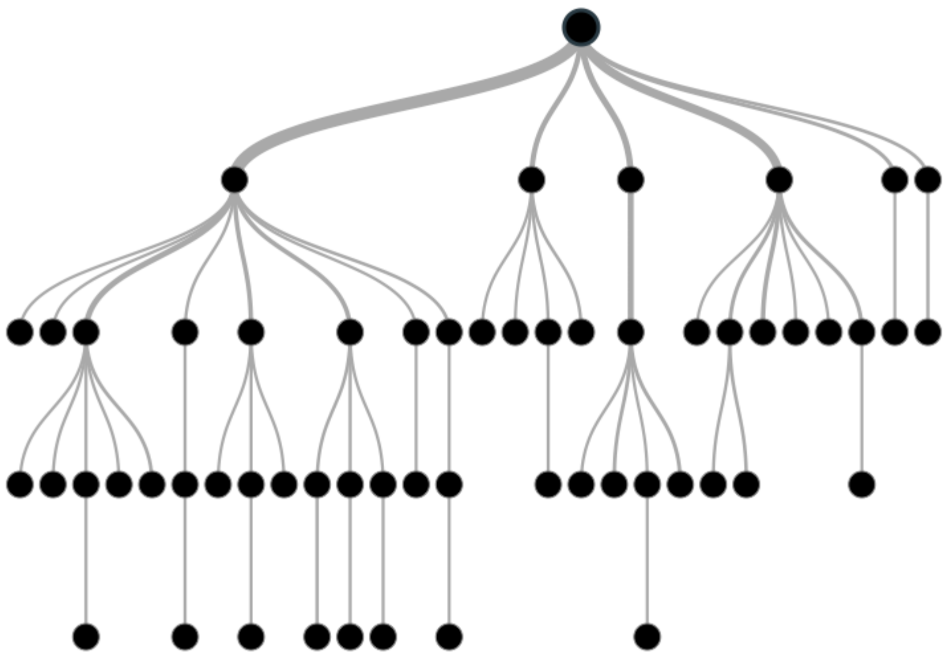
Generative Adversarial Networks
Unbiased sample gradients



M. G. Bellemare *et al.*, "The Cramer Distance as a Solution to Biased Wasserstein Gradients", arXiv:1705.10743.

Matteo Barbetti
12.06.2020

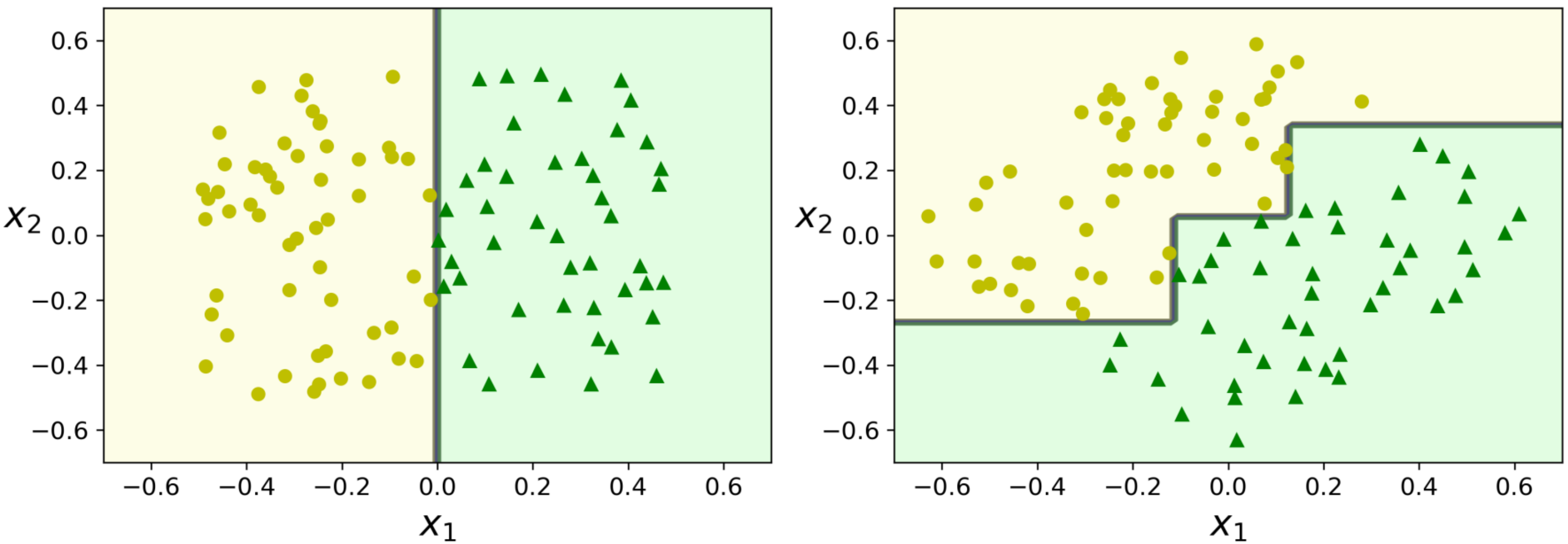
GANs for Particle Identification
Decision Trees (1/2)



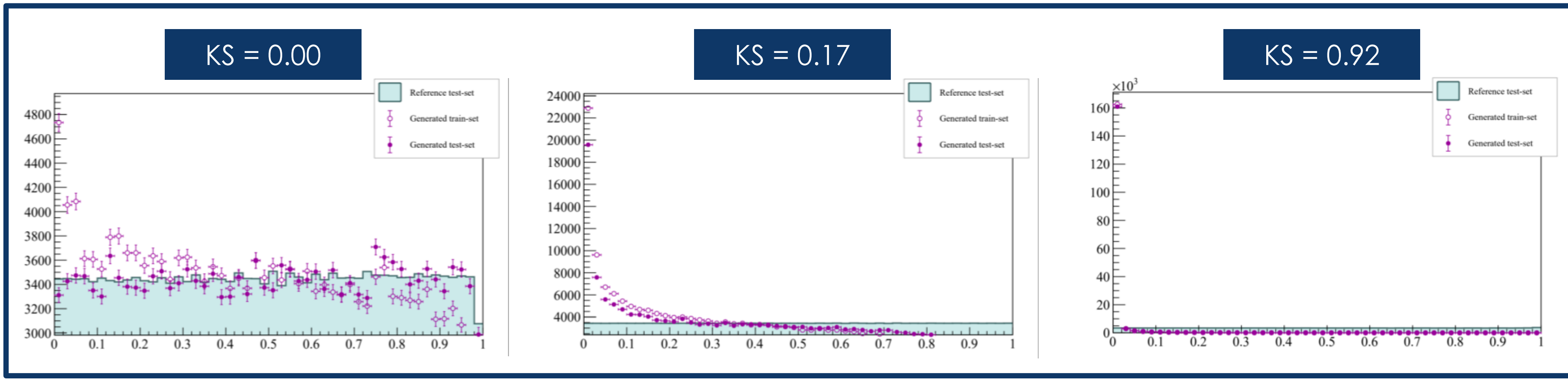
Regularization

```
max_depth = 5
max_leaf_nodes = 14
```

GANs for Particle Identification
Decision Trees (2/2)

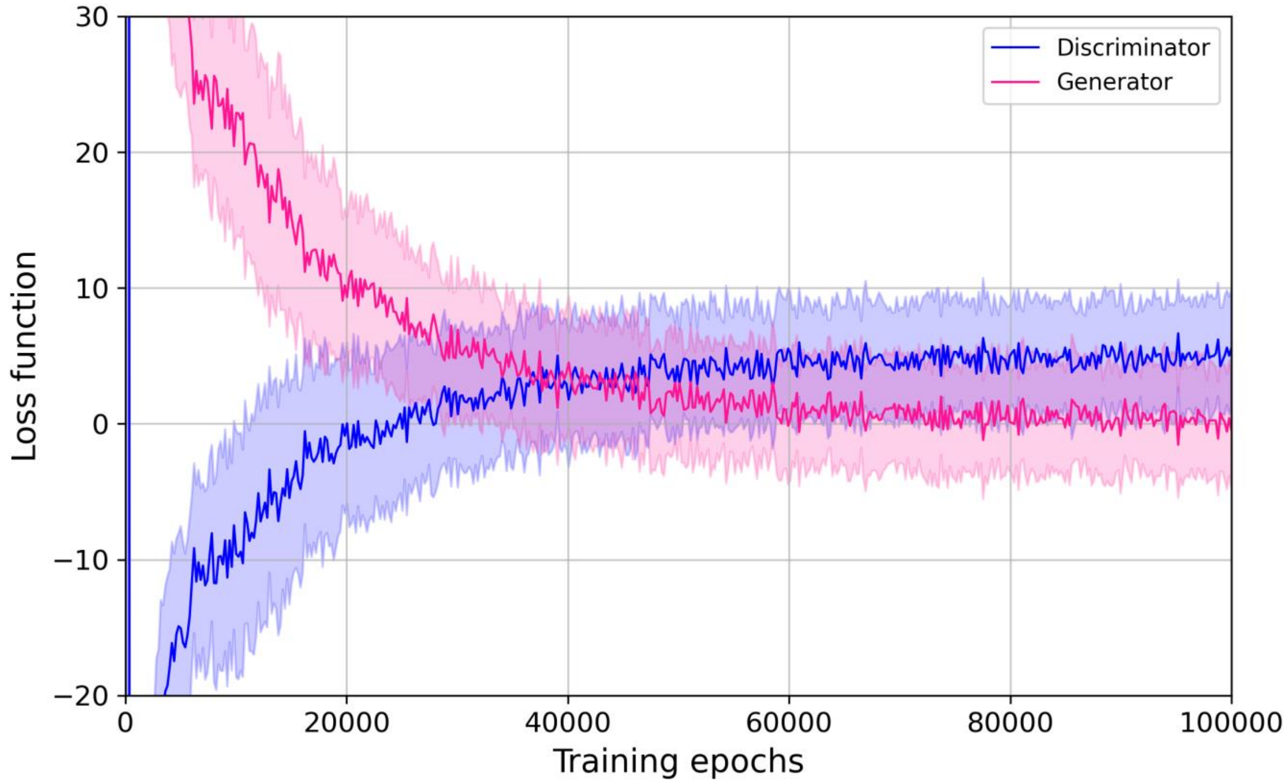


GANs for Particle Identification
Scoring method



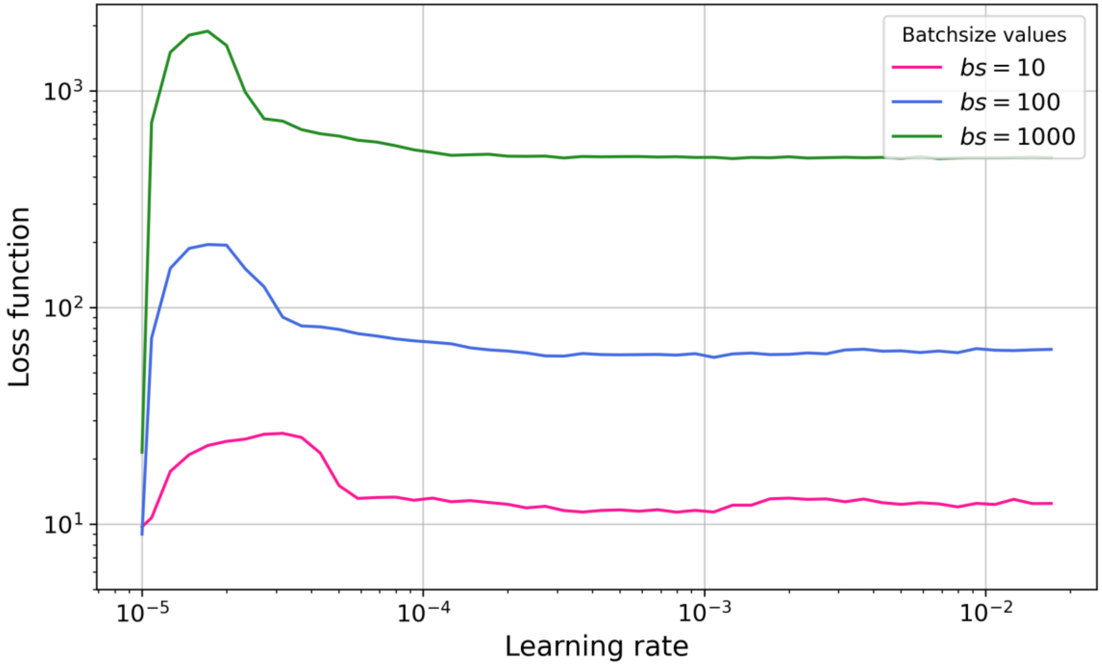
$$D_{KS} = \max_{p_{\mathrm{ref}} \in [0,1]} \left| R(p_{\mathrm{ref}}) - G(p_{\mathrm{ref}}) \right|$$

GANs for Particle Identification
Learning curves

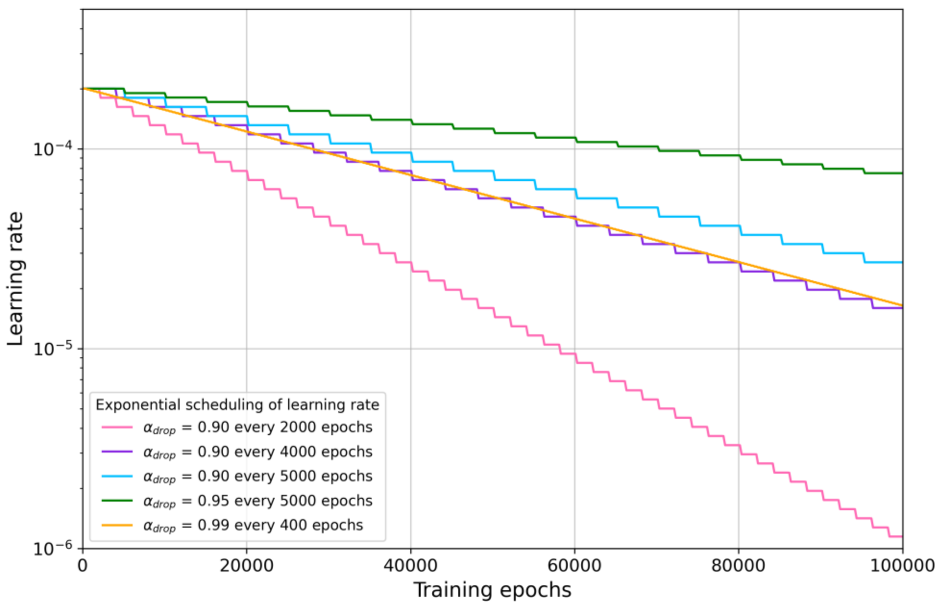


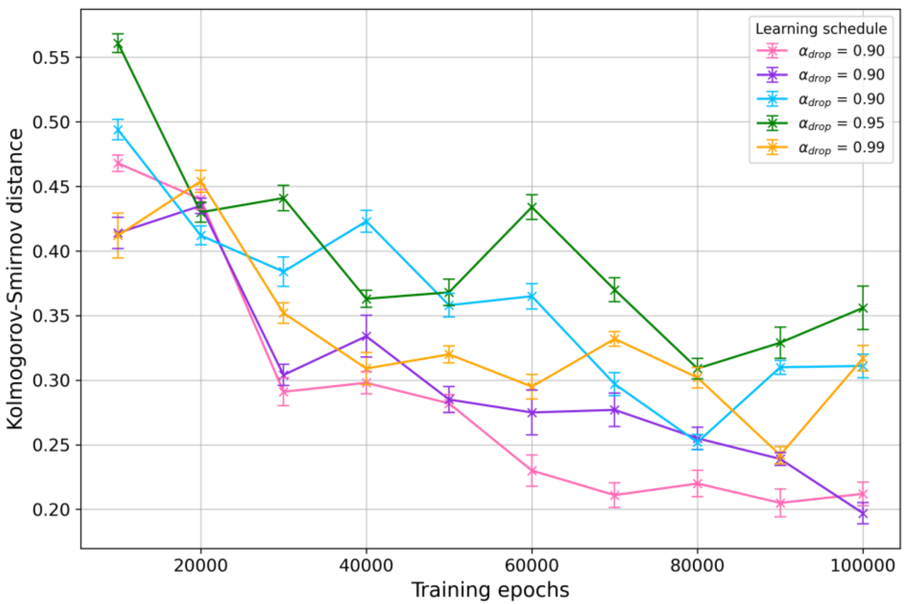
GANs for Particle Identification
Learning rate tuning (1/2)

$$\boxed{\theta_t \leftarrow \theta_{t-1} - \eta \, \nabla \mathcal{L}(\theta)}$$



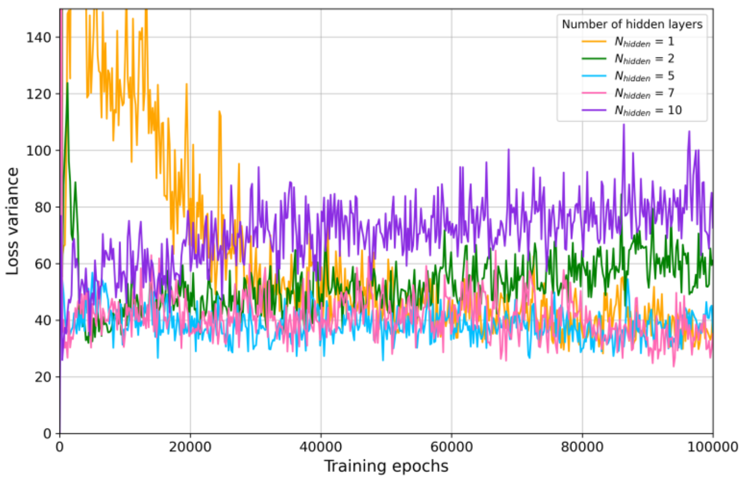
GANs for Particle Identification
Learning rate tuning (2/2)

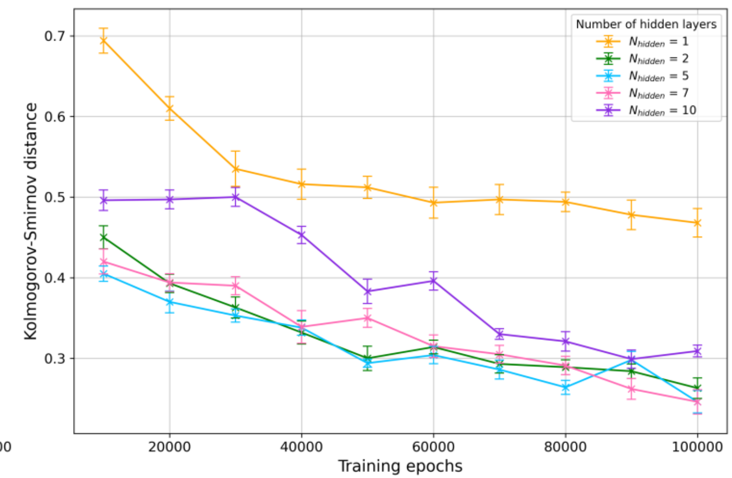


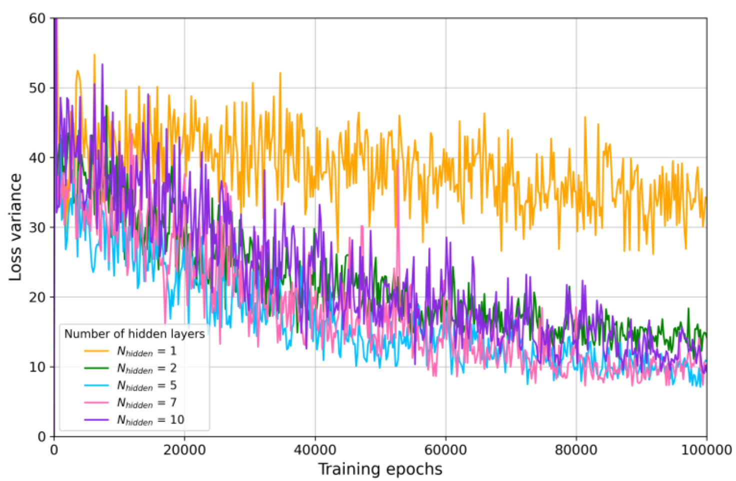


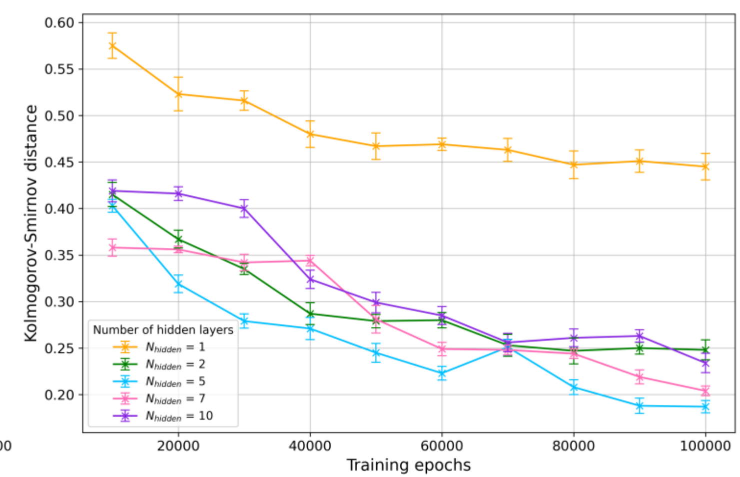
GANs for Particle Identification

Data pre-processing



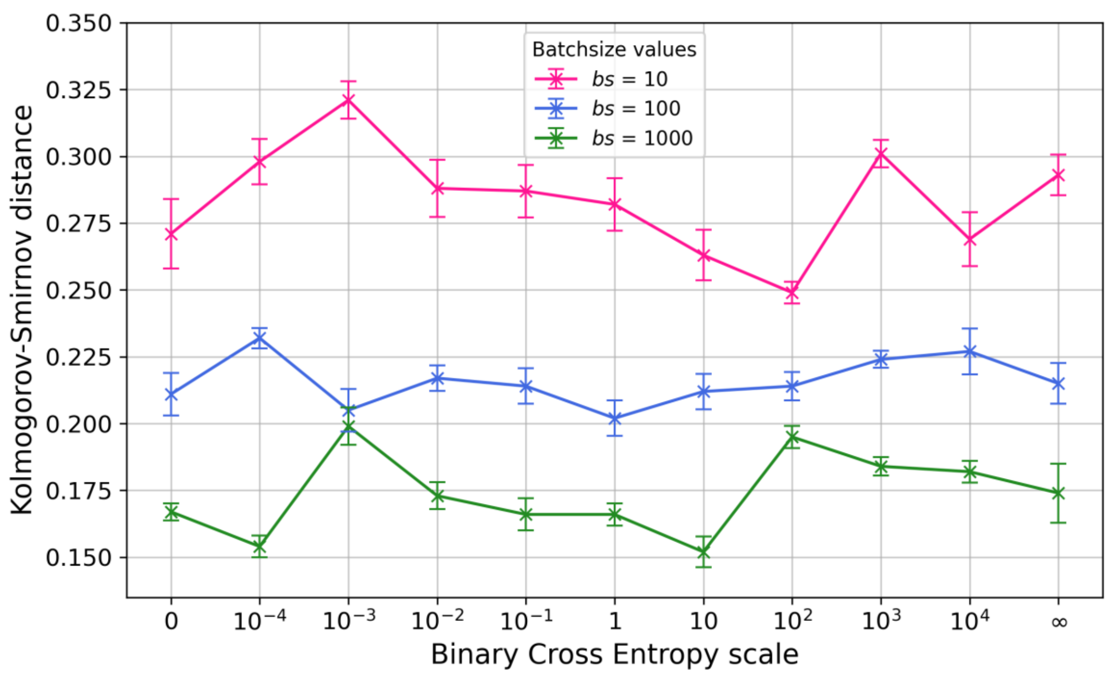




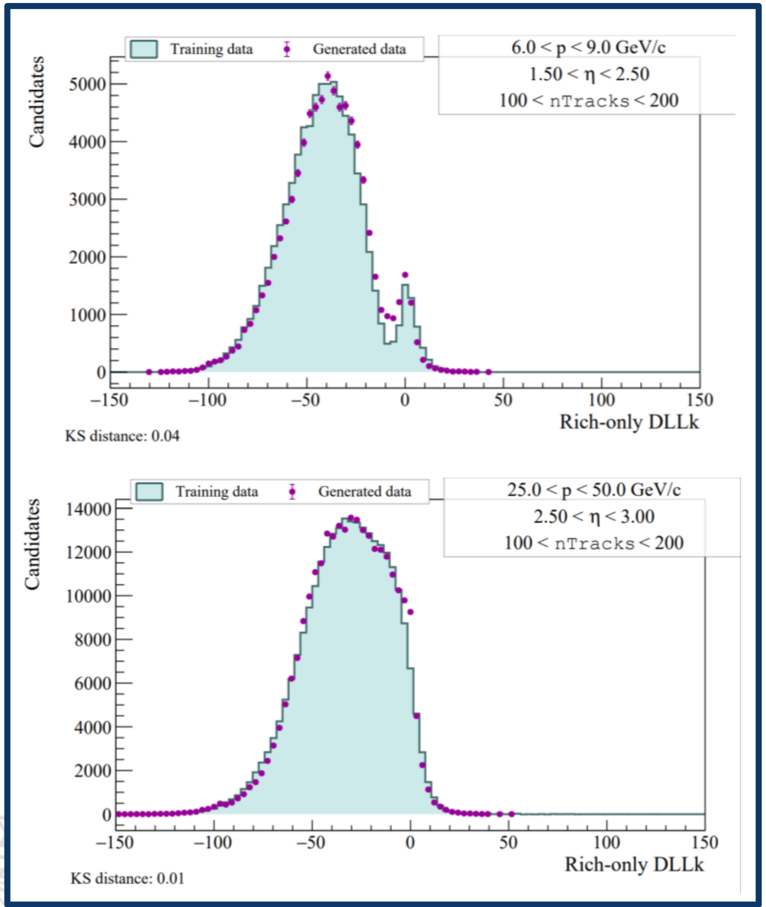


GANs for Particle Identification
Cramér distance VS cross-entropy

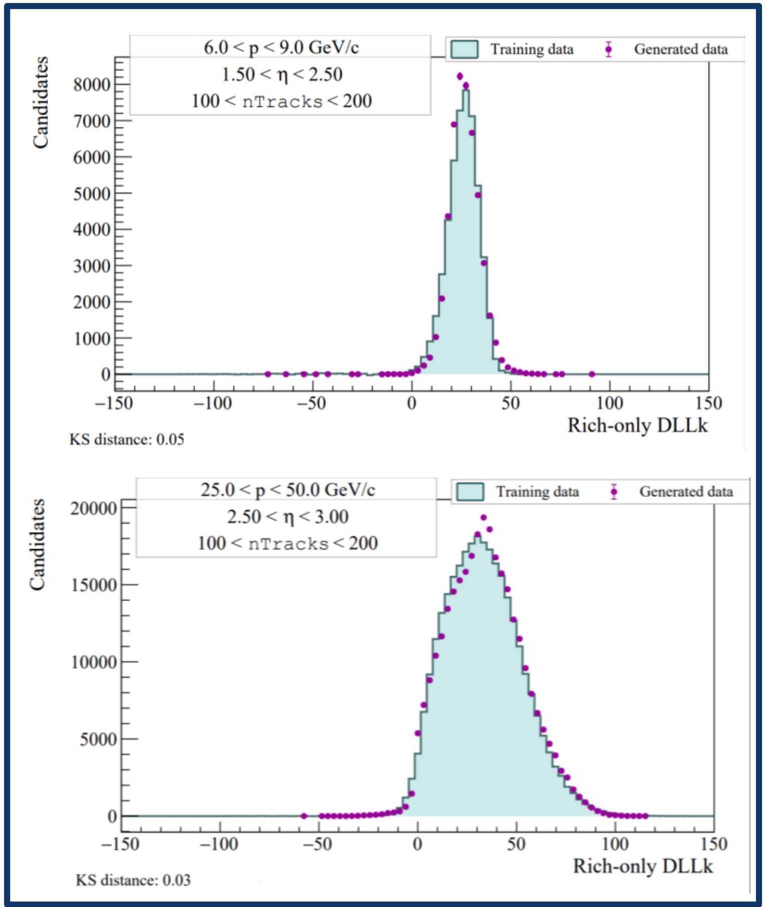
$$\boxed{\mathcal{L} = \mathcal{L}_{\mathrm{cramer}} + \xi\, \mathcal{L}_{\mathrm{bce}}}$$



GANs for Particle Identification
Model validation (1/7)

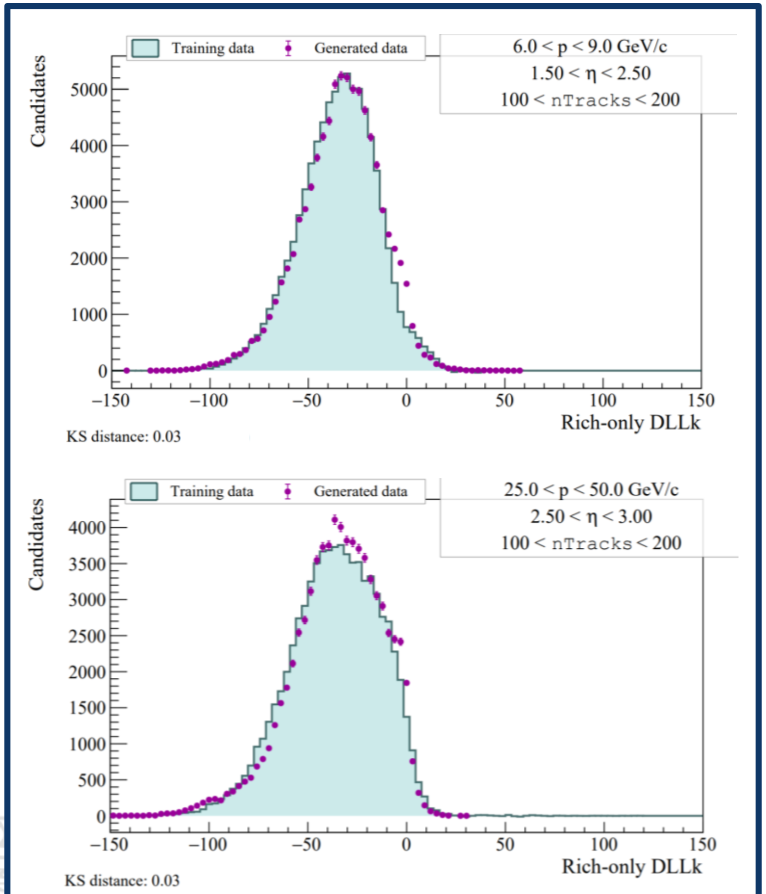


Pion track

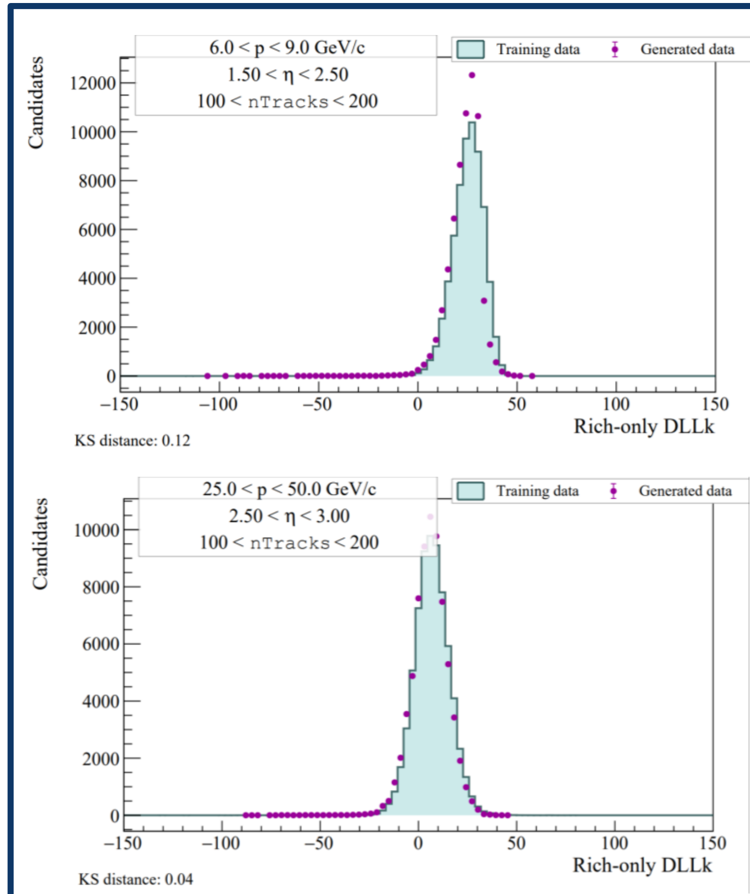


Kaon track

GANs for Particle Identification
Model validation (2/7)



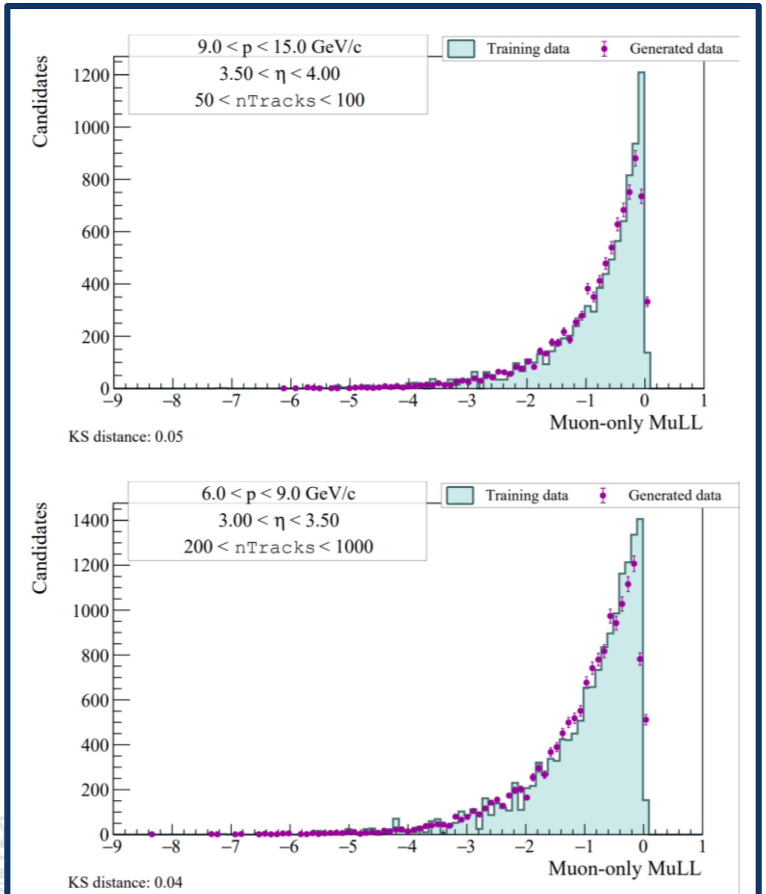
Muon track



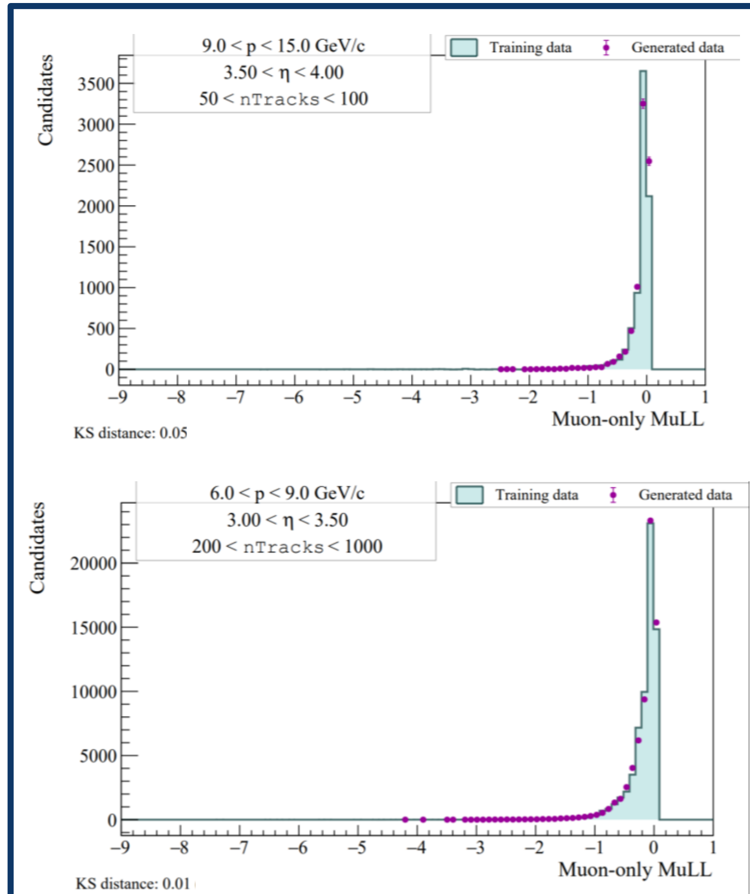
Proton track

GANs for Particle Identification

Model validation (3/7)

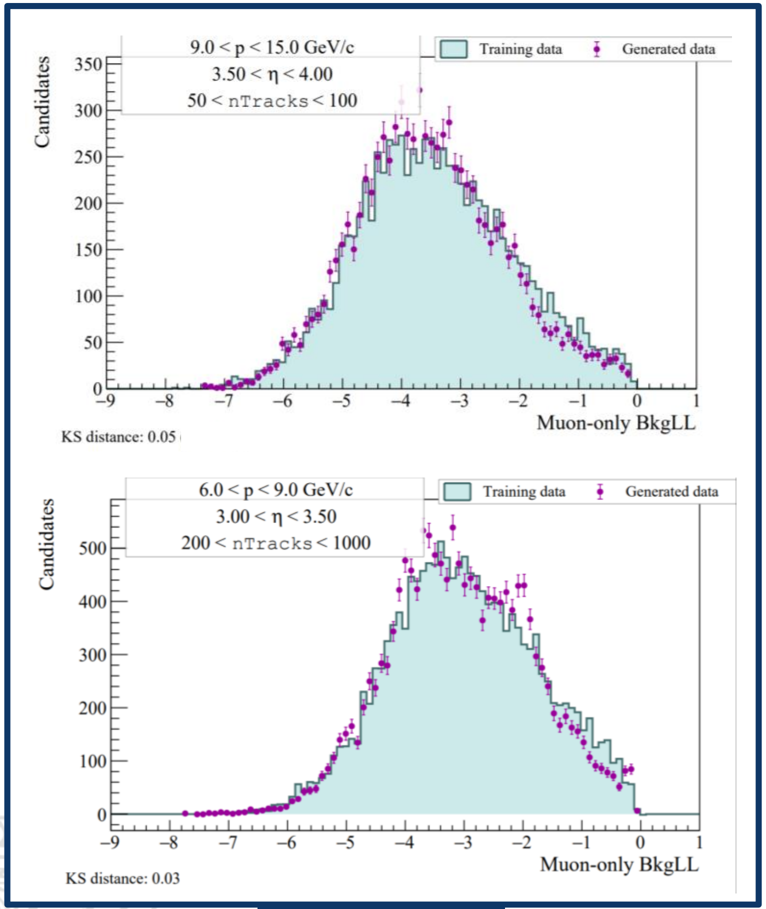


Muon track

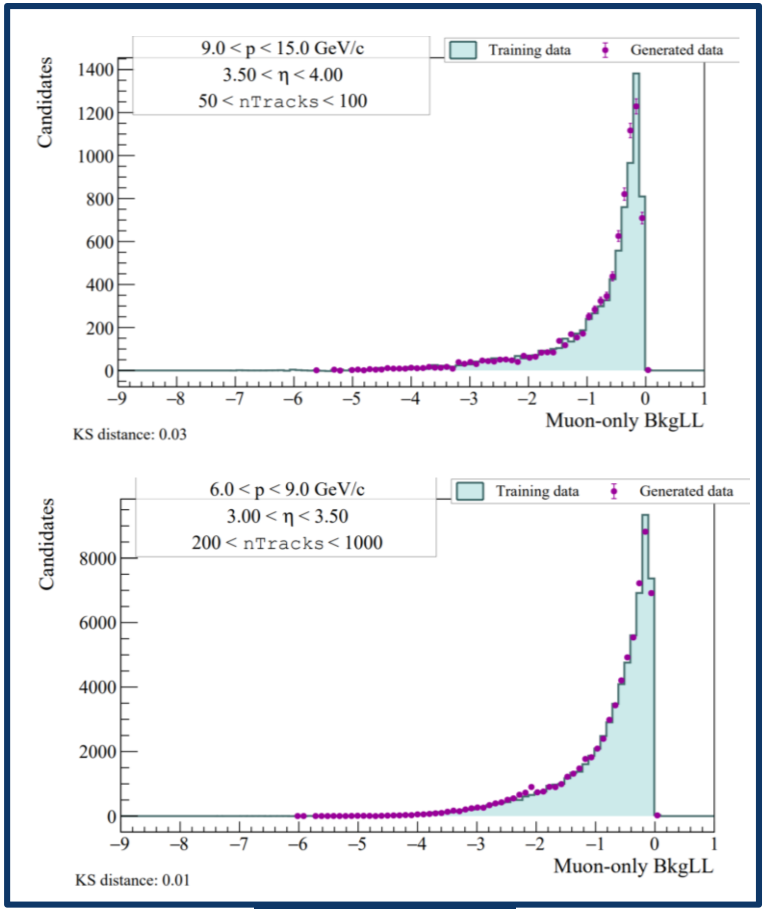


Proton track

GANs for Particle Identification
Model validation (4/7)

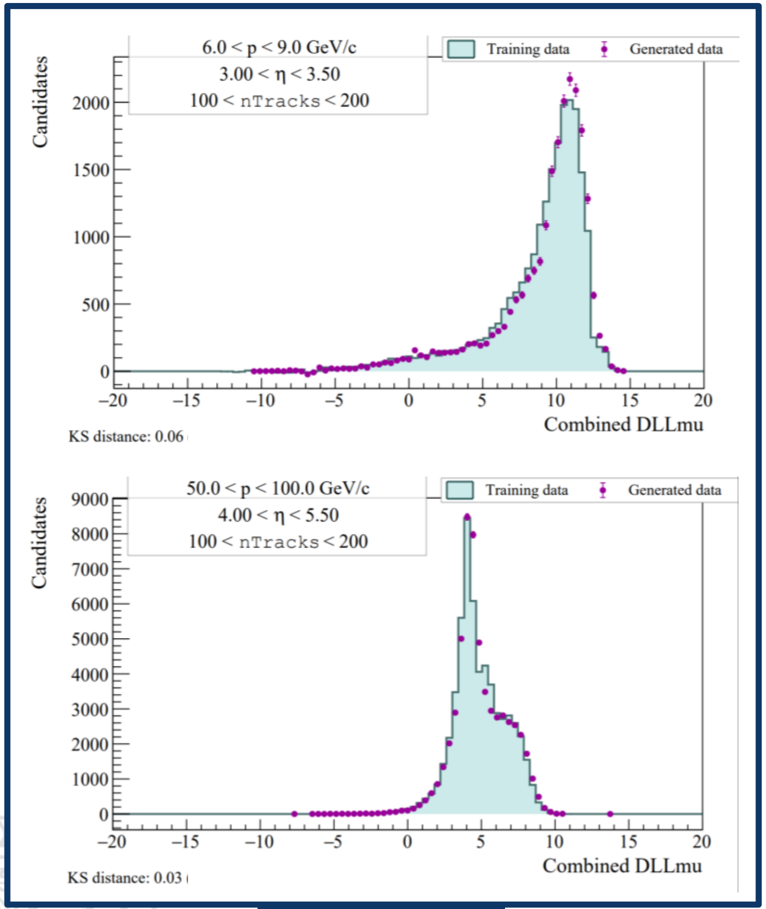


Muon track

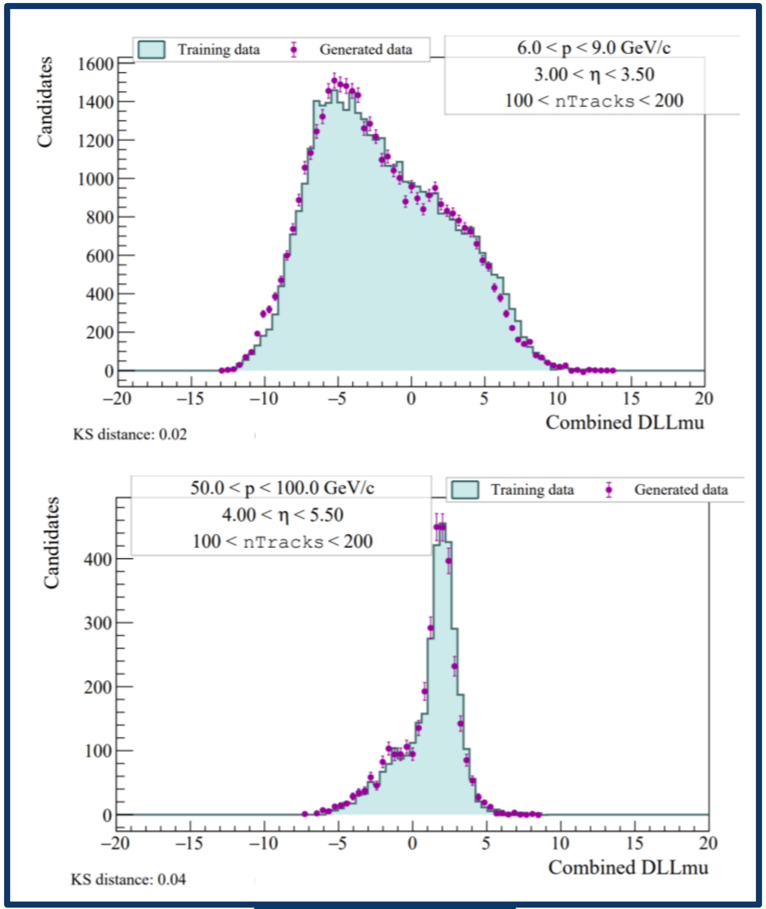


Proton track

GANs for Particle Identification
Model validation (5/7)



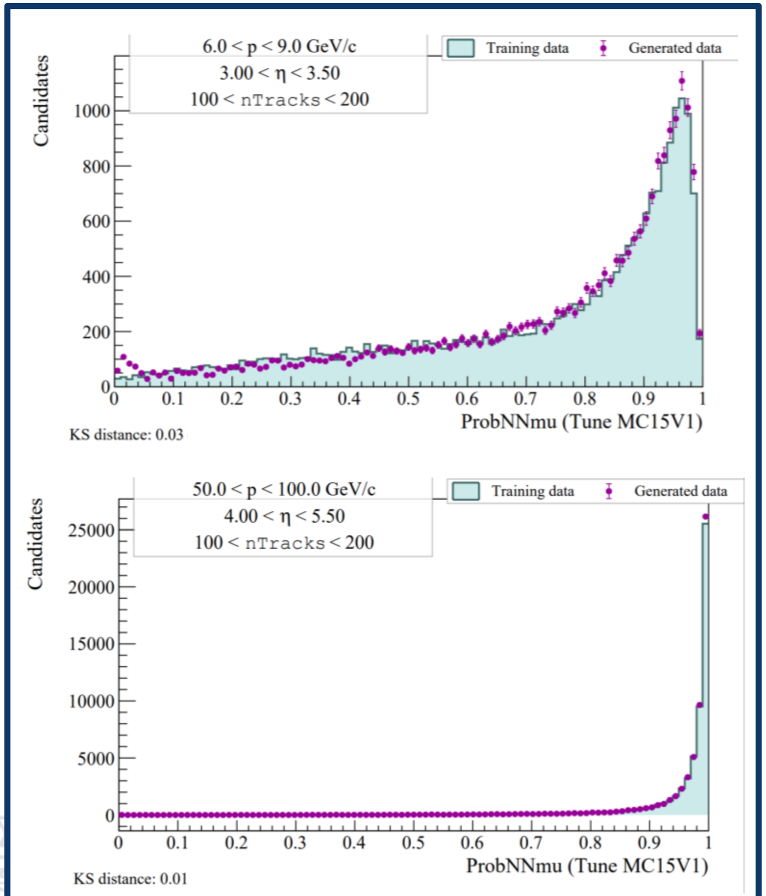
Muon track



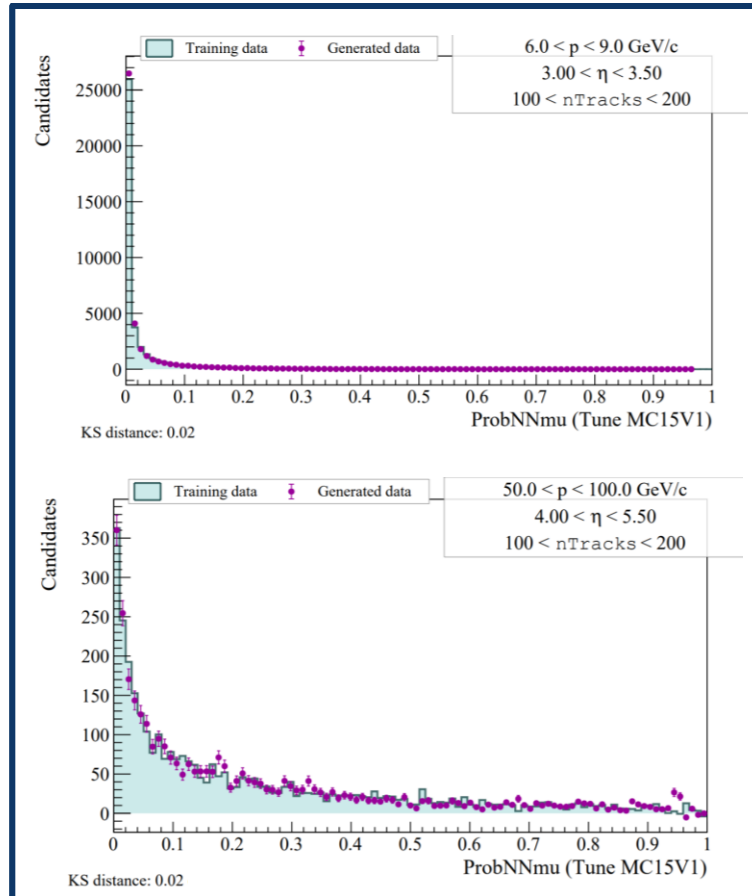
Proton track

GANs for Particle Identification

Model validation (6/7)

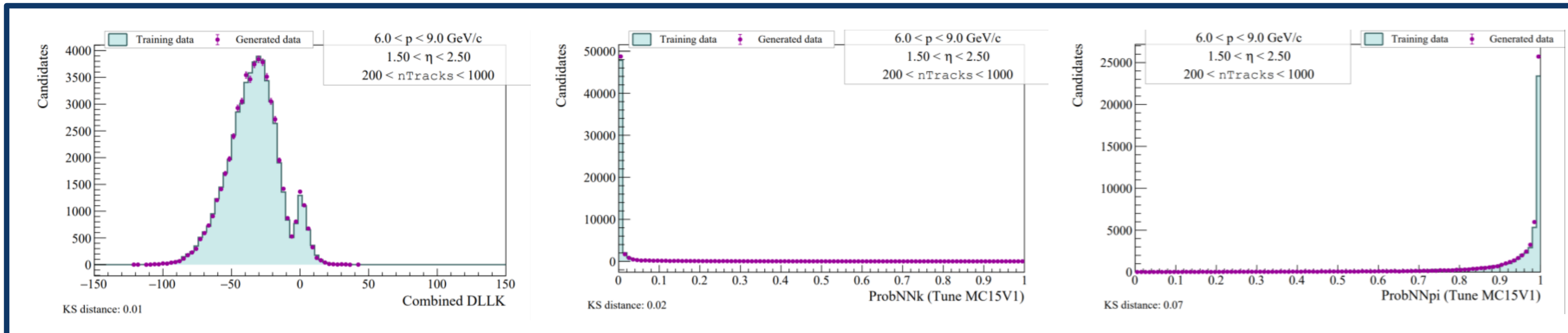


Muon track

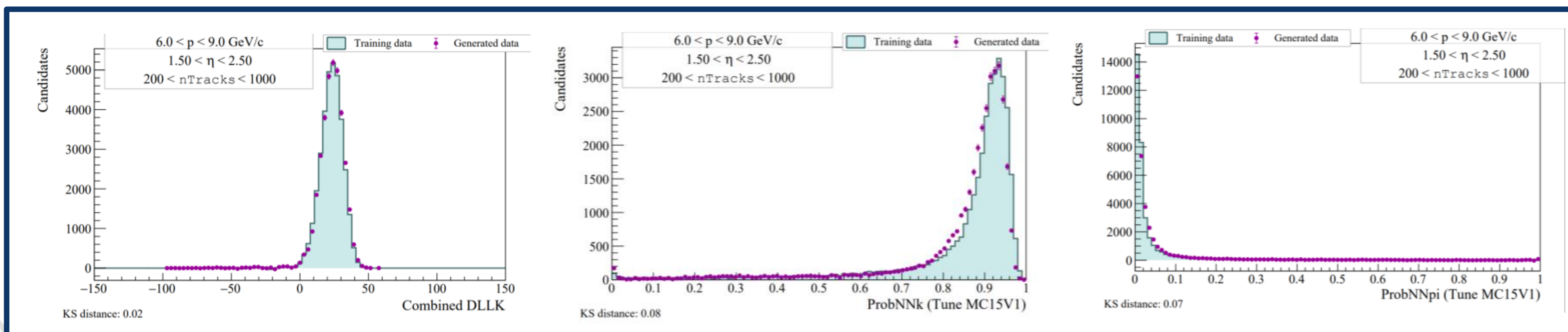


Proton track

GANs for Particle Identification
Model validation (7/7)



Pion track



Kaon track

The Mambah framework
EventStore example

Batch 1				
events	protoparts	vertices	mcParticles	mcVertices
_index	_index	_index	_index	_index
batch	batch	batch	batch	batch
nTracks	event	event	event	event
produced	prodVertex	mother	prodVertex	mother
...	decayVtx	daugh1	decayVtx	decayVtx
	truth	daugh2	reconstr	daugh2
	...	...	...	...

Batch 2				
events	protoparts	vertices	mcParticles	mcVertices
_index	_index	_index	_index	_index
batch	batch	batch	batch	batch
nTracks	event	event	event	event
produced	prodVertex	mother	prodVertex	mother
...	decayVertex	daugh1	decayVertex	daugh1
	truth	daugh2	reconstr	daugh2
	...	...	...	...

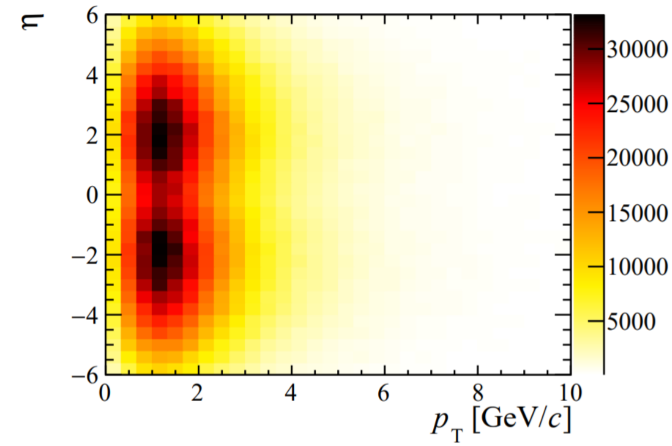
The Mambah framework
Generation phase

Generator

The first step of the generation phase is to produce particles that include heavy and resonant states, never directly detectable.

To date, `mambah.sim` implements only the **particle-gun** approach, namely it produces a single heavy flavour hadron (beauty or charm) per event according to predefined kinematic distributions.

Charmed Hadrons



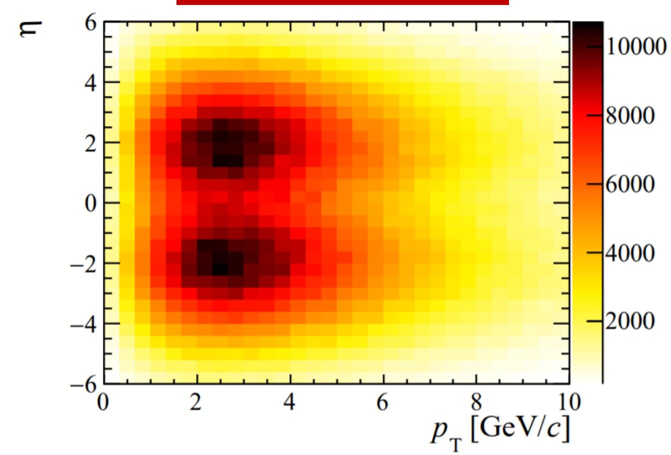
Decay tool

The second step of the generation phase is to simulate the decay chain until long-lived final states.

The `mambah.sim` module implements two models to describe the sequence of decays:

- **`zfit/phasespace`**, a lightning fast package to simulate phase space decays
- **EvtGen**, a celebrated package to simulate the physics of heavy flavour decays

Beauty Hadrons



The Mambah framework
Reconstruction process

Efficiency model

The particles produced at the generator-level are stored within the **Monte Carlo databases** together with all the kinematic information.

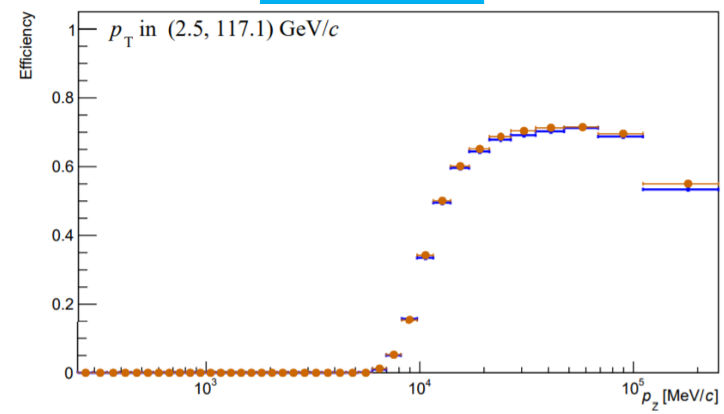
Among all the particles, only the long-lived ones are passed through a filter function (a Mambah tool) parameterizing the reconstruction efficiency. The particles survived the efficiency selection are store within the **reconstruction databases**.

The efficiency correction is modelled by a **trained neural network** taking as inputs the momentum components and the origin vertex coordinates.

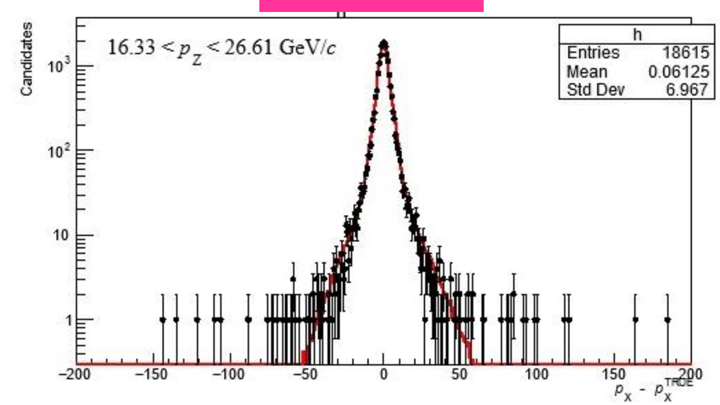
Resolution effects

The reconstructed particles are passed through another Mambah tool which performs the **smearing** of the track momentum components. The smearing function is modelled by a **trained neural network** taking as inputs the momentum components.

Efficiency

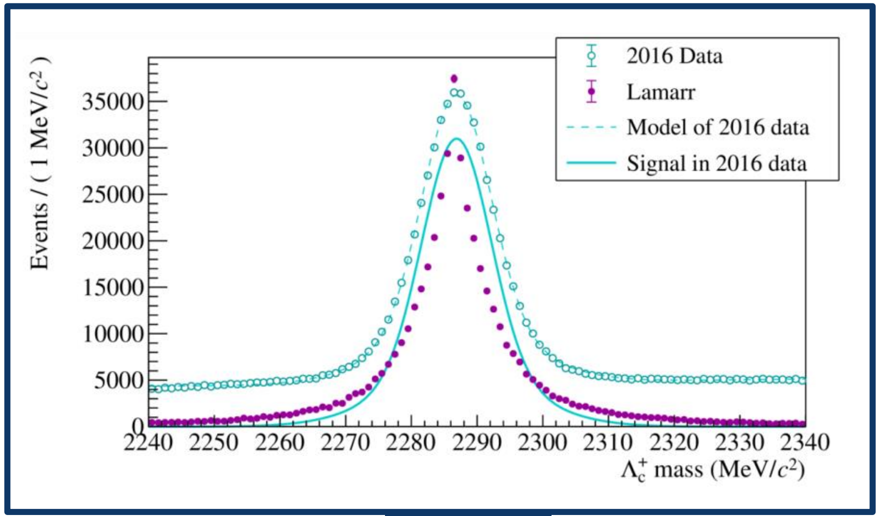


Resolution

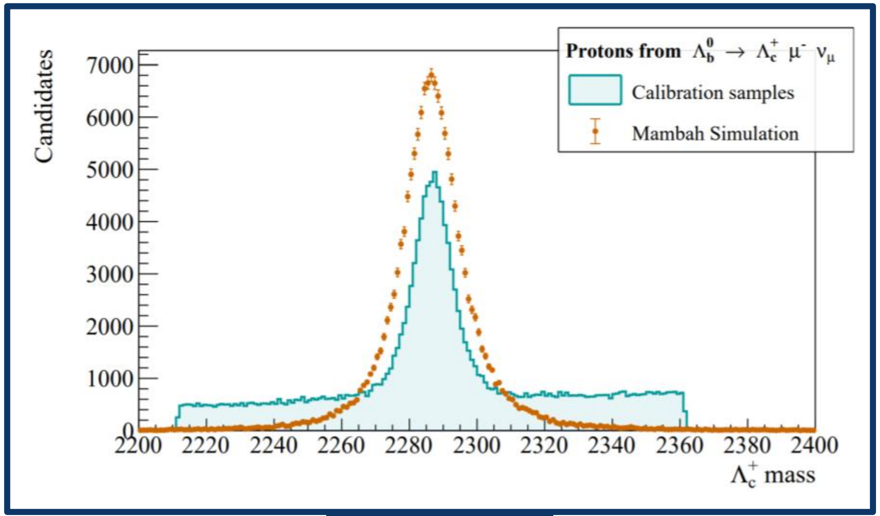


Simulation frameworks

Lamarr VS `mamba.sim` (1/3)

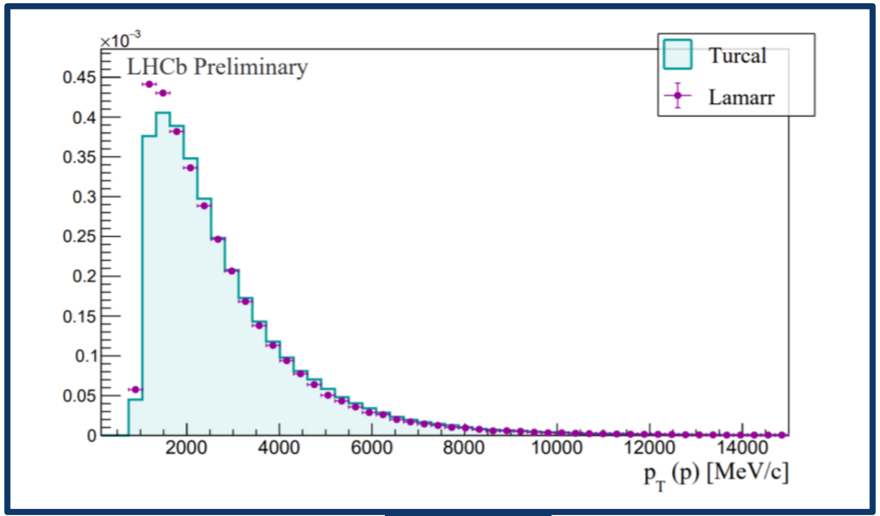


Lamarr

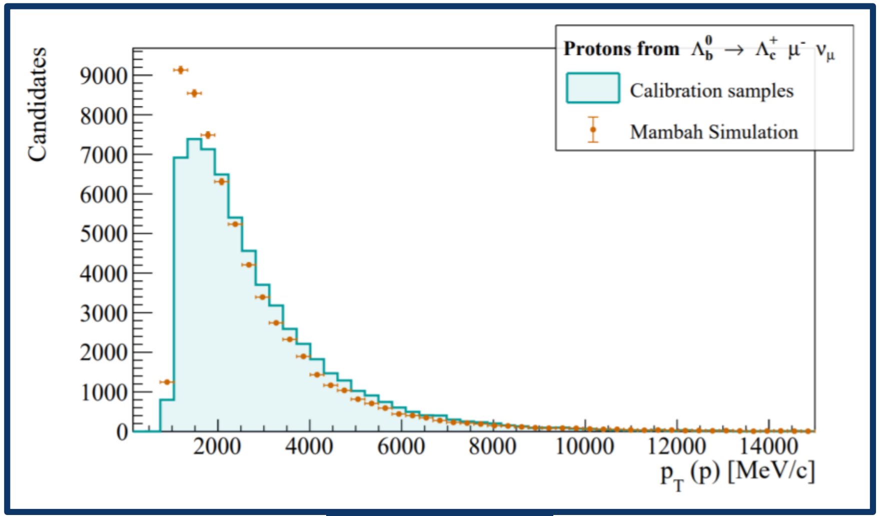


`mamba.sim`

Simulation frameworks
Lamarr VS `mamba.sim` (2/3)

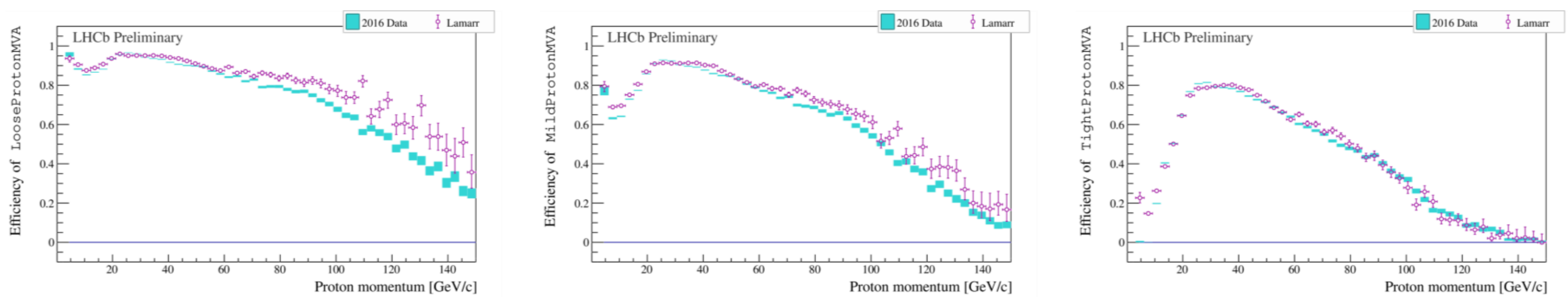


Lamarr

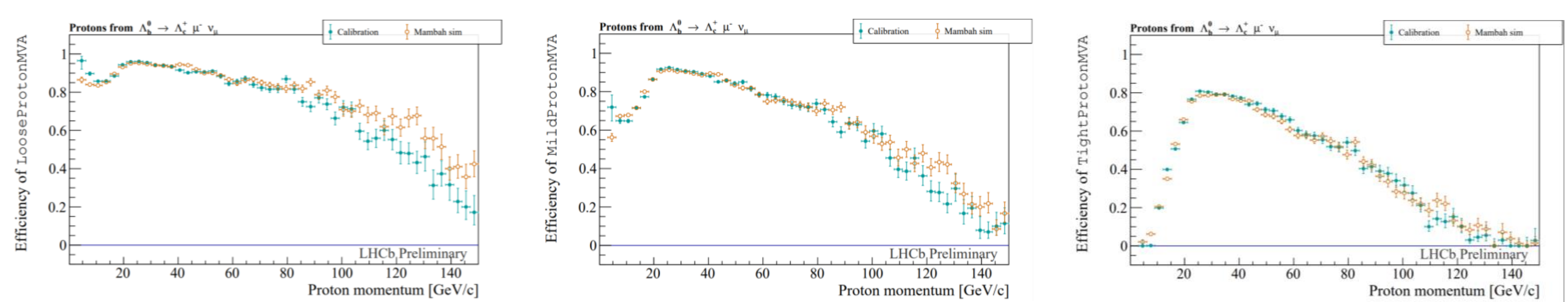


`mamba.sim`

Simulation frameworks
Lamarr VS `mamba.sim` (3/3)

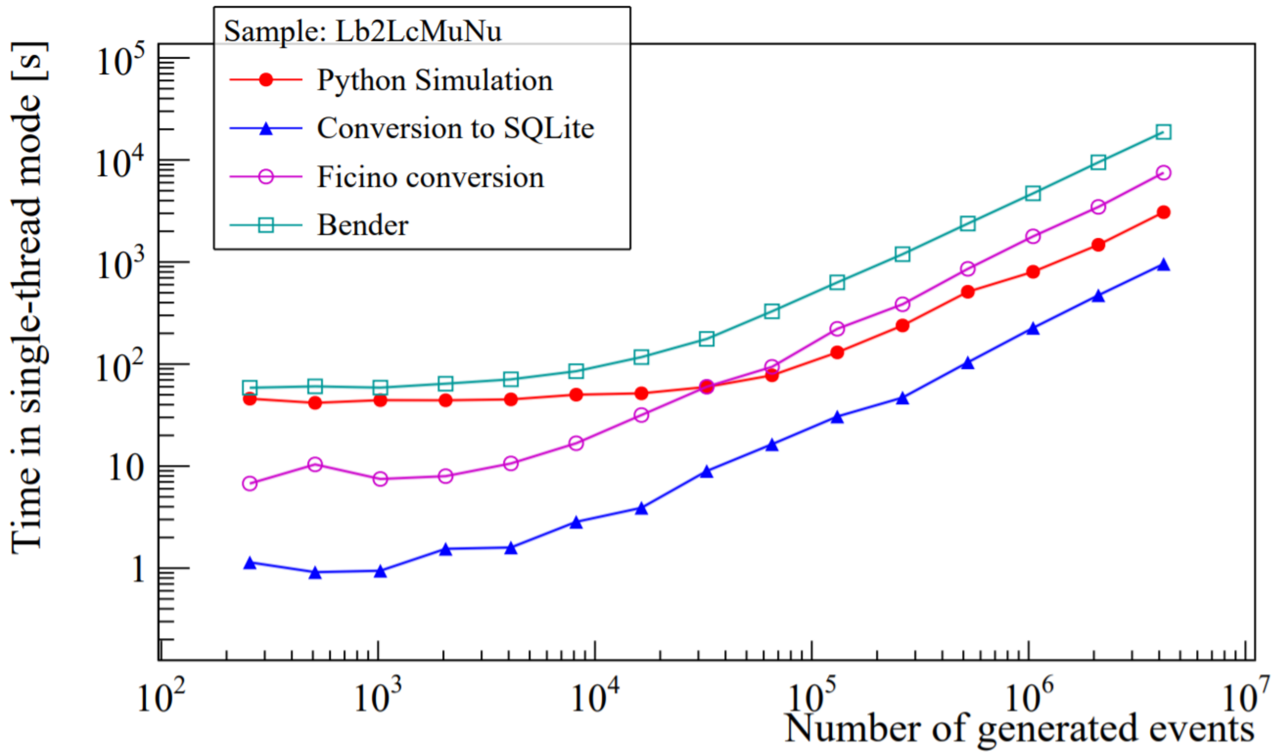


Lamarr

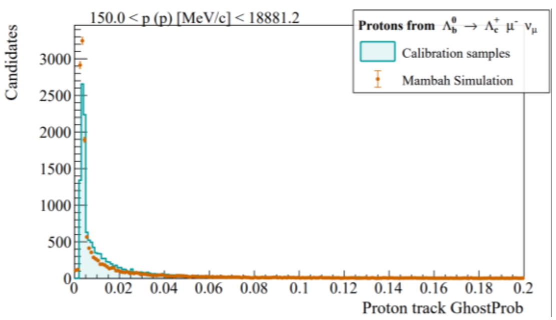


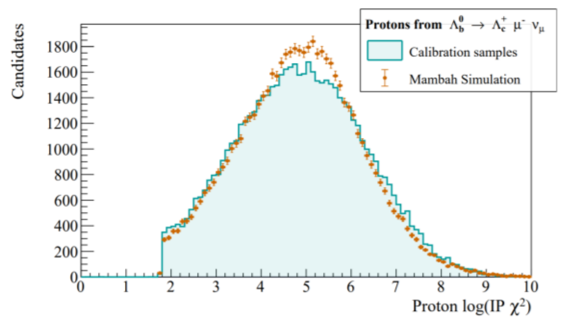
`mamba.sim`

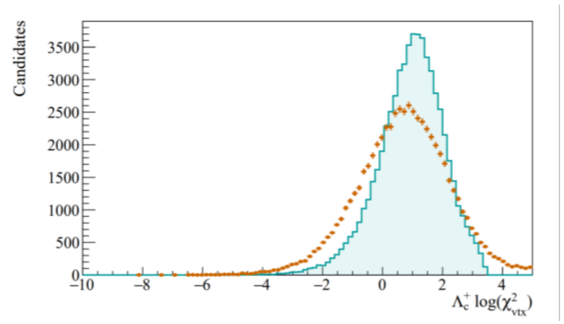
The Mambah framework
Single-thread CPU cost



The Mambah framework
Framework validation (1/2)







ghostProbability

Probability computed by a neural network that the tracks are obtained from a **random combination** of hits in the tracker rather than to a real particle depositing energy in the detector.

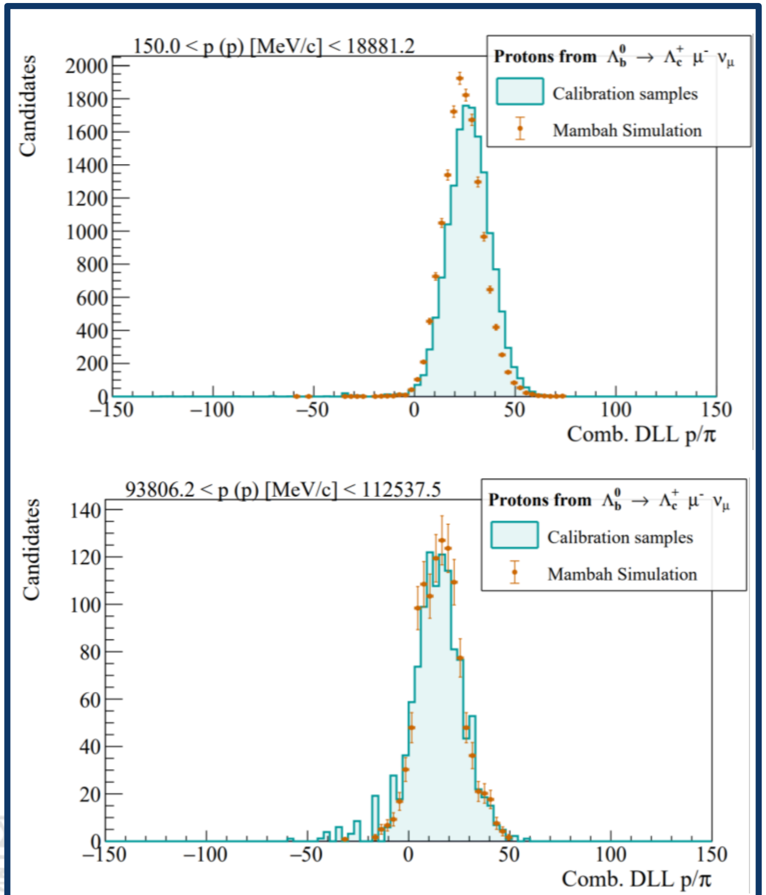
Impact Parameter

Distance from the **primary vertex** to the reconstructed momentum of daughters: in this case, the proton impact parameter is reported and, as expected, it is **inconsistent** with the primary vertex.

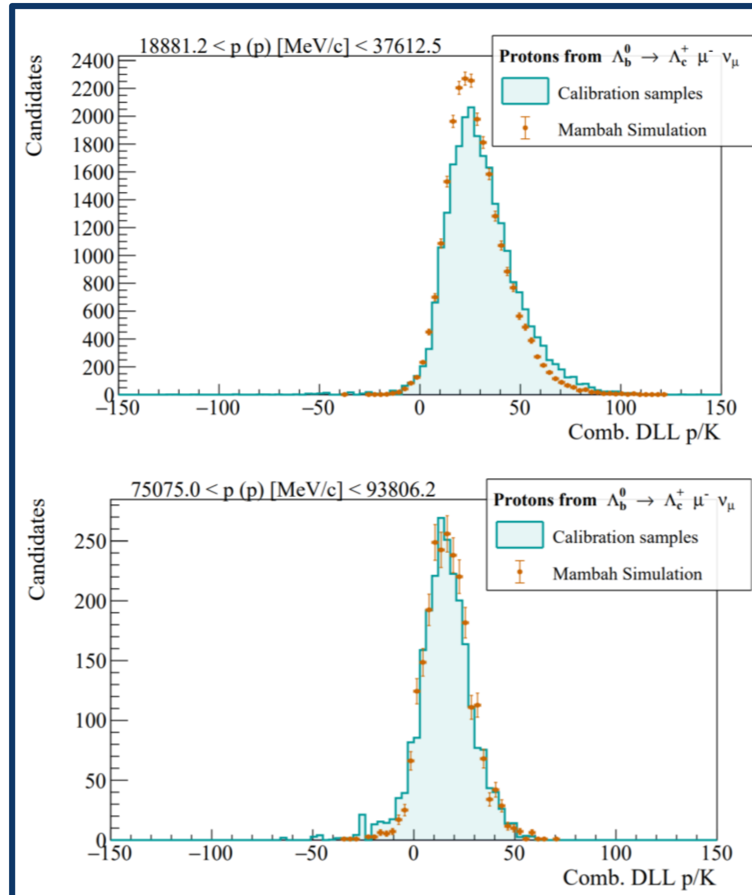
Covariance matrix

Measure of the goodness of track parameters: in this case, the covariance matrix is parameterized as a function of the **momentum only** that is unable, of reproducing data distribution.

The Mambah framework
Framework validation (2/2)



Pion track



Kaon track