Machine Learning in LHCb

OSAF Topical Lectures, 9 June 2022
Jacco de Vries

Overview

- Various real-life ML examples in LHCb
- Practical scenarios, tips, WIP
- Some physics, reco algorithms
  (and shameless advertisement) along the way
- Disclaimer: will use some internal material
- Please ask questions!

LHCb experiment

- Tracking, vertexing

  $\delta p/p$ = 0.5-1.0 %

  δt = 45 fs

- PID

  95% Kaon ID

  @ 5% pion misID

- 3-stage Trigger

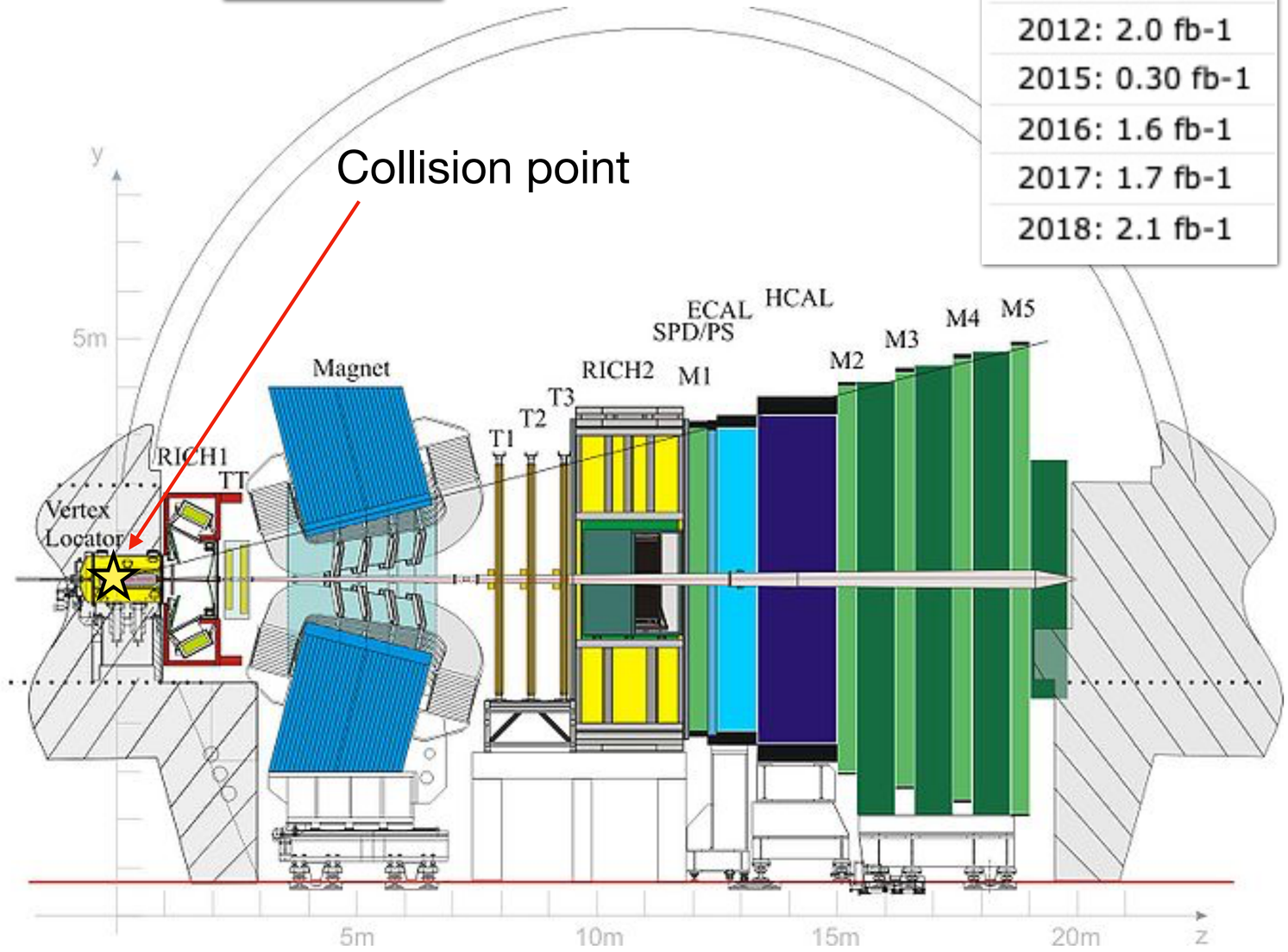
  40 MHz —> 12.5 kHz (run2)

  ε ~ 90% for dimuon

  ε ~ 30% for multibody hadron

$2 < \eta < 5$

2011: 1.0 fb-1
2012: 2.0 fb-1
2015: 0.30 fb-1
2016: 1.6 fb-1
2017: 1.7 fb-1
2018: 2.1 fb-1



LHCb experiment

- Tracking, vertexing

  $\delta p/p$ = 0.5-1.0 %

  δt = 45 fs

- PID

  95% Kaon ID

  @ 5% pion misID

- Trigger

  40 MHz —> 12.5 kHz (run2)

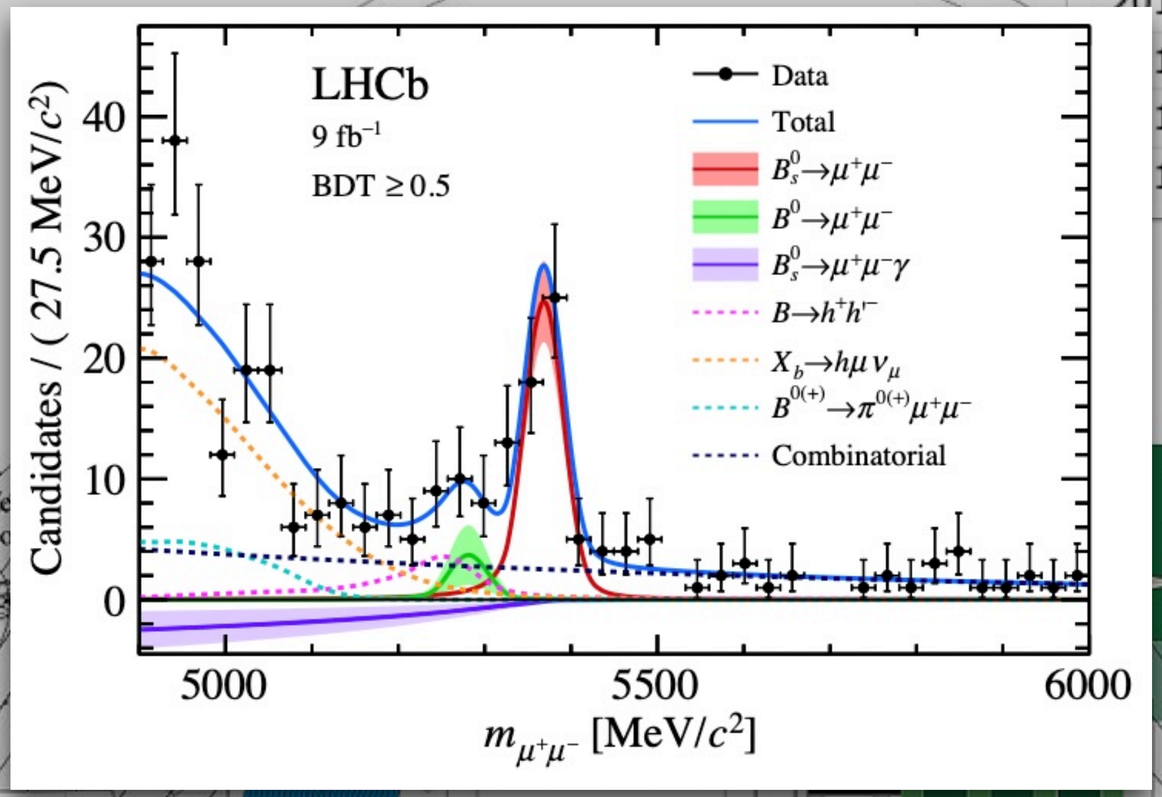
  ε ~ 90% for dimuon

  ε ~ 30% for multibody hadron

$2 < \eta < 5$

2011: 1.0 fb-1
2012: 2.0 fb-1
2015: 0.30 fb-1
16: 1.6 fb-1
17: 1.7 fb-1
18: 2.1 fb-1

https://doi.org/10.1103/PhysRevLett.128.041801



LHCb experiment

- Tracking, vertexing

  $\delta p/p$ = 0.5-1.0 %

  δt = 45 fs

- PID

  95% Kaon ID

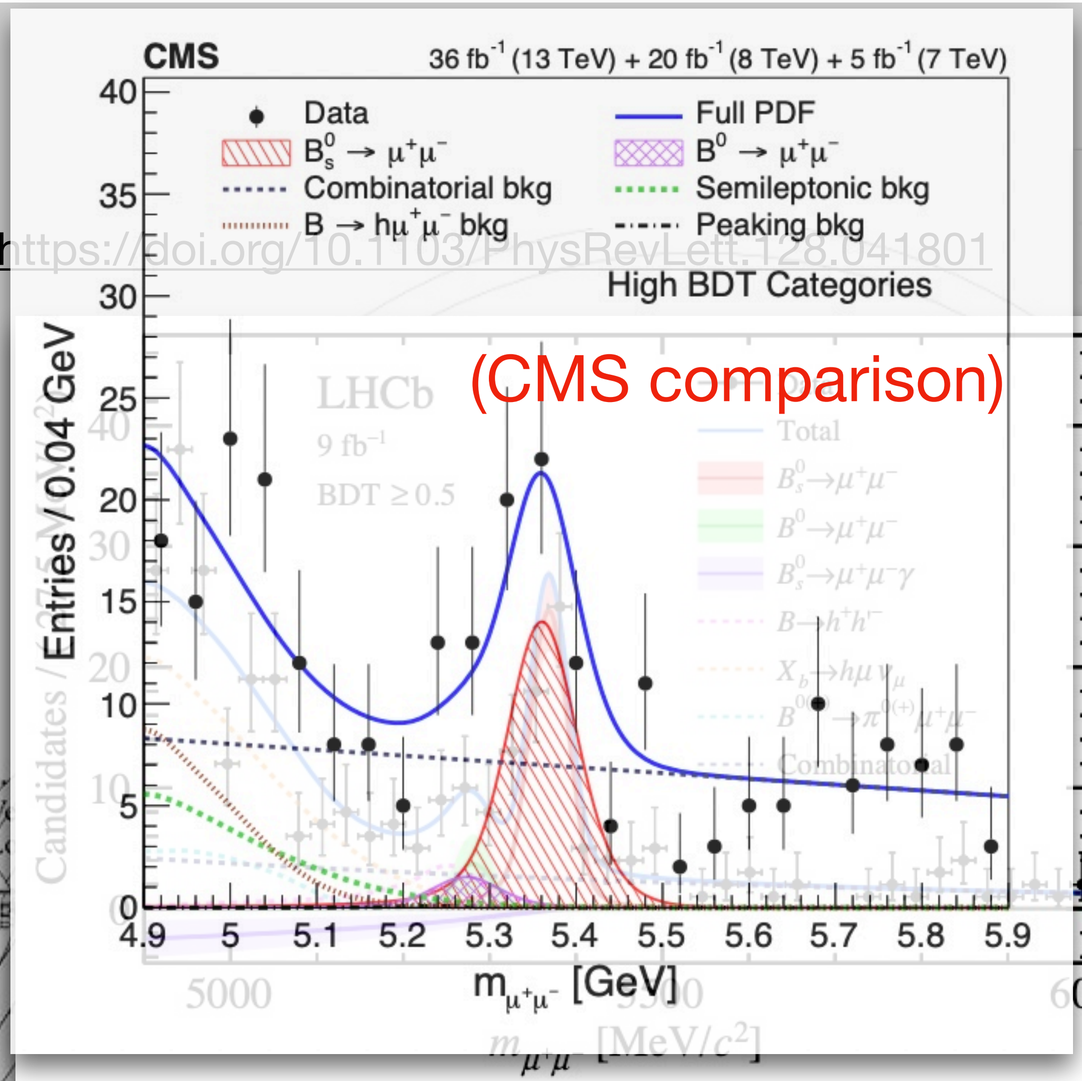
  @ 5% pion misID

- Trigger

  40 MHz —> 12.5 kHz (run2)

  ε ~ 90% for dimuon

  ε ~ 30% for multibody hadron



(CMS comparison)

$2 < \eta < 5$

2011: 1.0 fb-1
2012: 2.0 fb-1
2015: 0.30 fb-1
16: 1.6 fb-1
17: 1.7 fb-1
18: 2.1 fb-1

LHCb experiment

- Tracking, vertexing

  $\delta p/p$ = 0.5-1.0 %

  δt = 45 fs

- PID

  95% Kaon ID

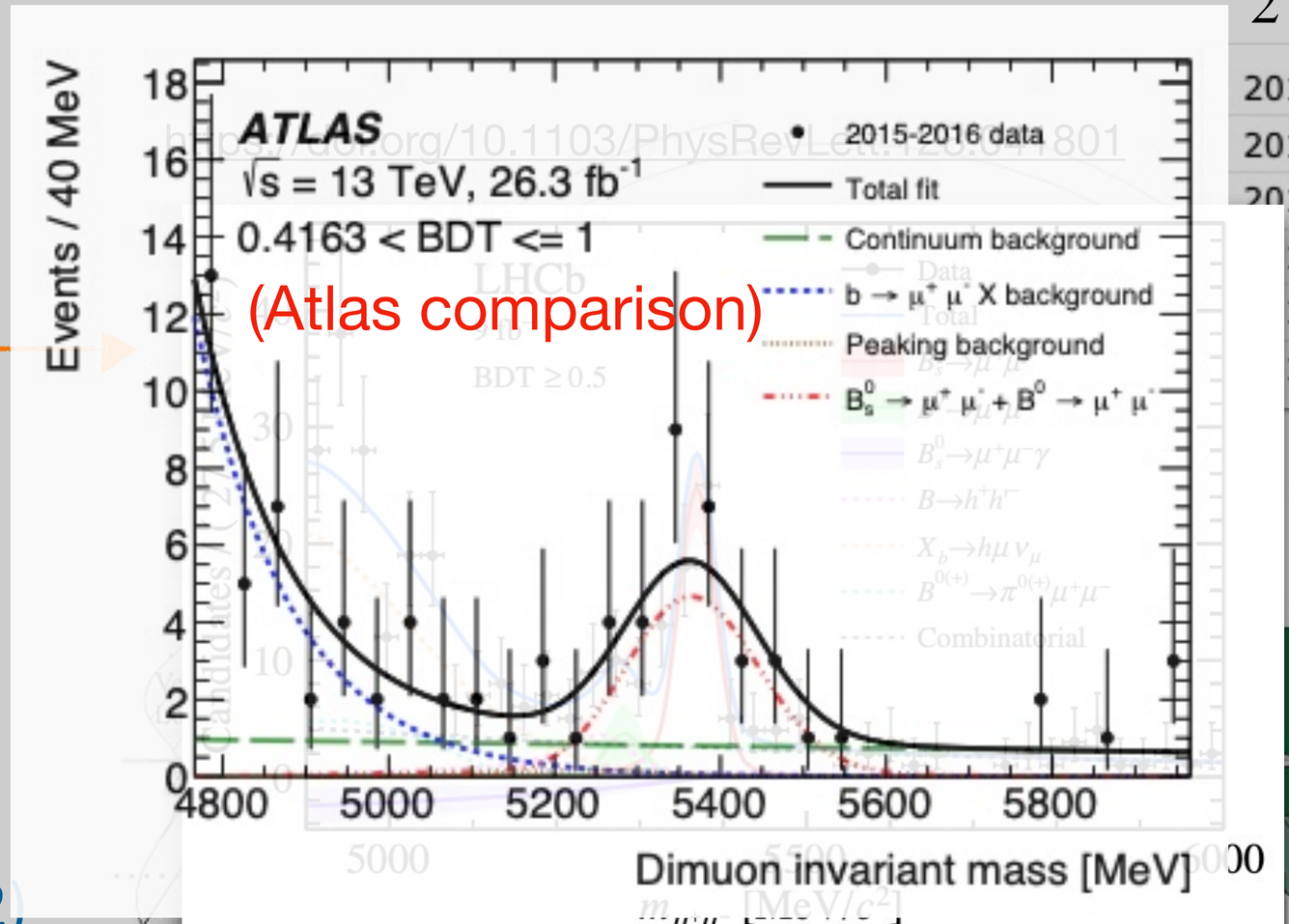
  @ 5% pion misID

- Trigger

  40 MHz —> 12.5 kHz (run2)

  ε ~ 90% for dimuon

  ε ~ 30% for multibody hadron



$2 < \eta < 5$

2011: 1.0 fb-1
2012: 2.0 fb-1
2015: 0.30 fb-1
16: 1.6 fb-1
17: 1.7 fb-1
18: 2.1 fb-1

LHCb experiment

- Tracking, vertexing

  $\delta p/p$ = 0.5-1.0 %

  δt = 45 fs

- PID

  95% Kaon ID

  @ 5% pion misID

- Trigger

  40 MHz —> 12.5 kHz (run2)

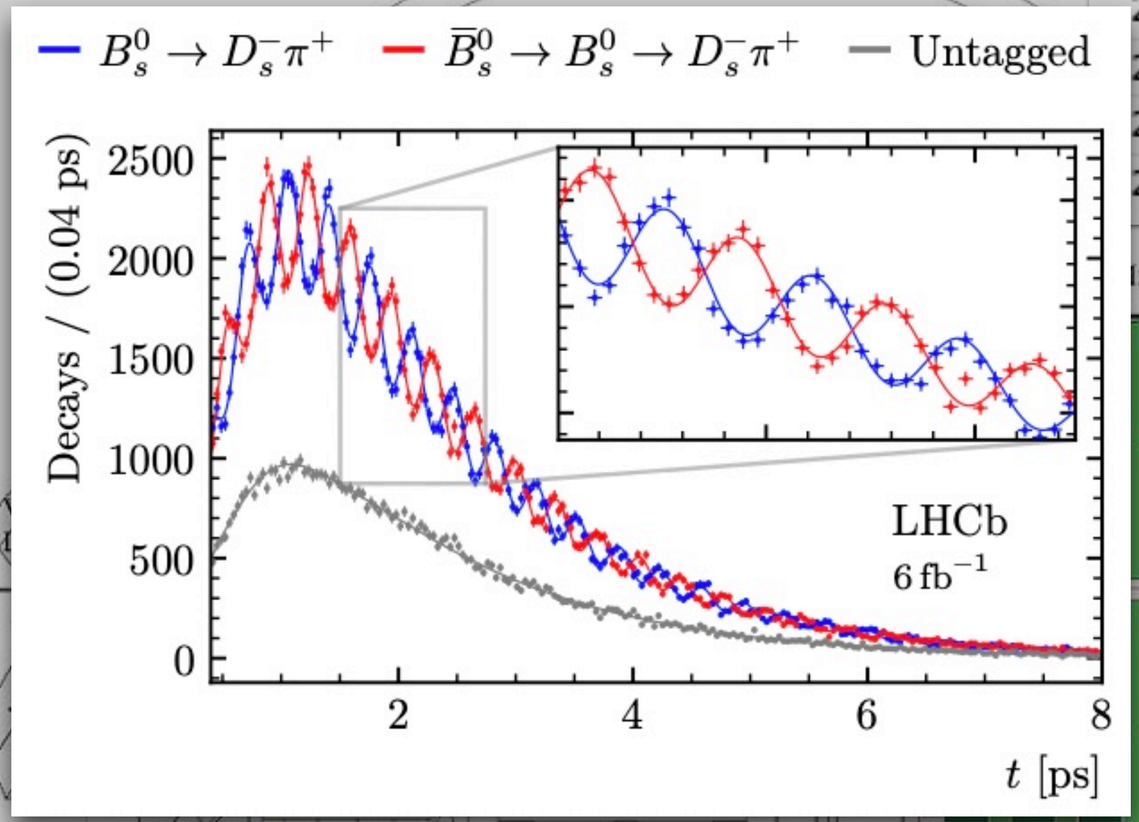
  ε ~ 90% for dimuon

  ε ~ 30% for multibody hadron

$2 < \eta < 5$

2011: 1.0 fb-1
2012: 2.0 fb-1
2015: 0.30 fb-1
2016: 1.6 fb-1
2017: 1.7 fb-1
2018: 2.1 fb-1

https://www.nature.com/articles/s41567-021-01394-x



LHCb experiment

$2 < \eta < 5$

2011: 1.0 fb-1
2012: 2.0 fb-1
2015: 0.30 fb-1
b-1
b-1
b-1

https://indico.cern.ch/event/1137298/

- Tracking, vertexing

  $\delta p/p$ = 0.5-1.0 %

  δt = 45 fs

- PID

  95% Kaon ID

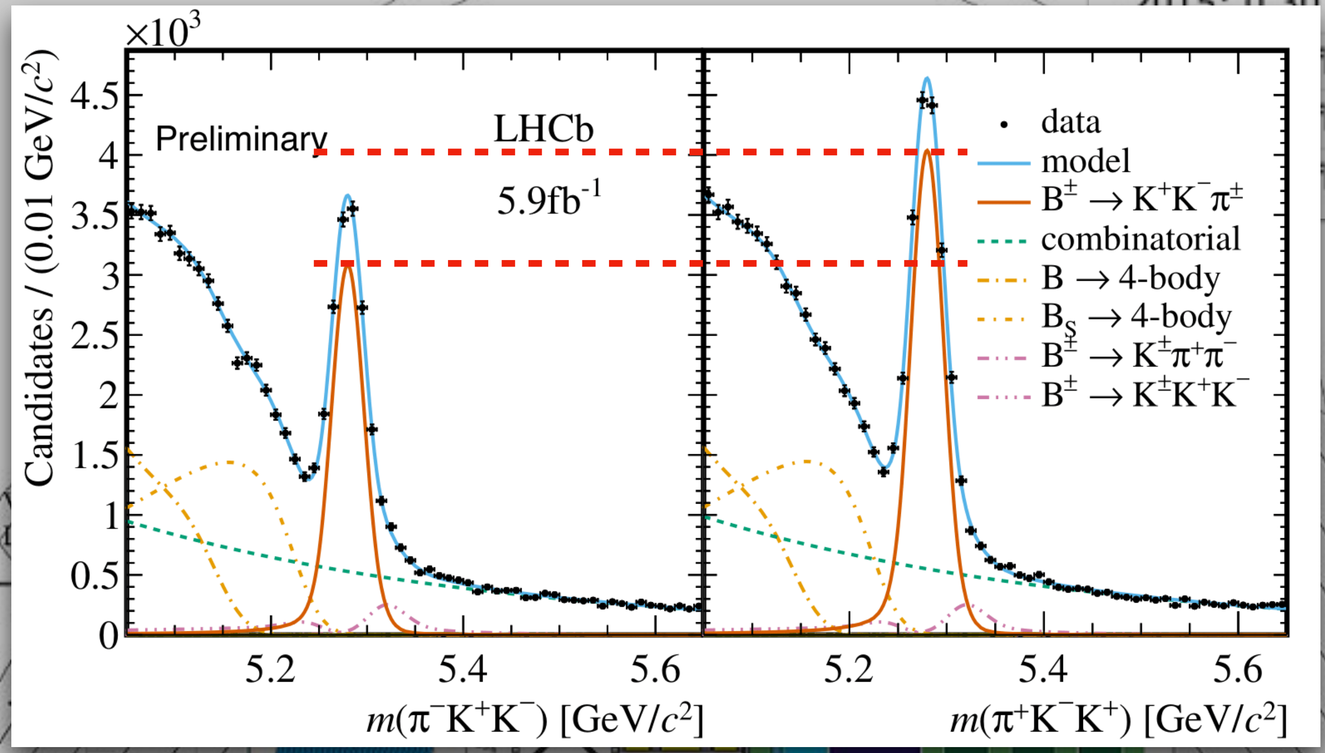
  @ 5% pion misID

- Trigger

  40 MHz —> 12.5 kHz (run2)

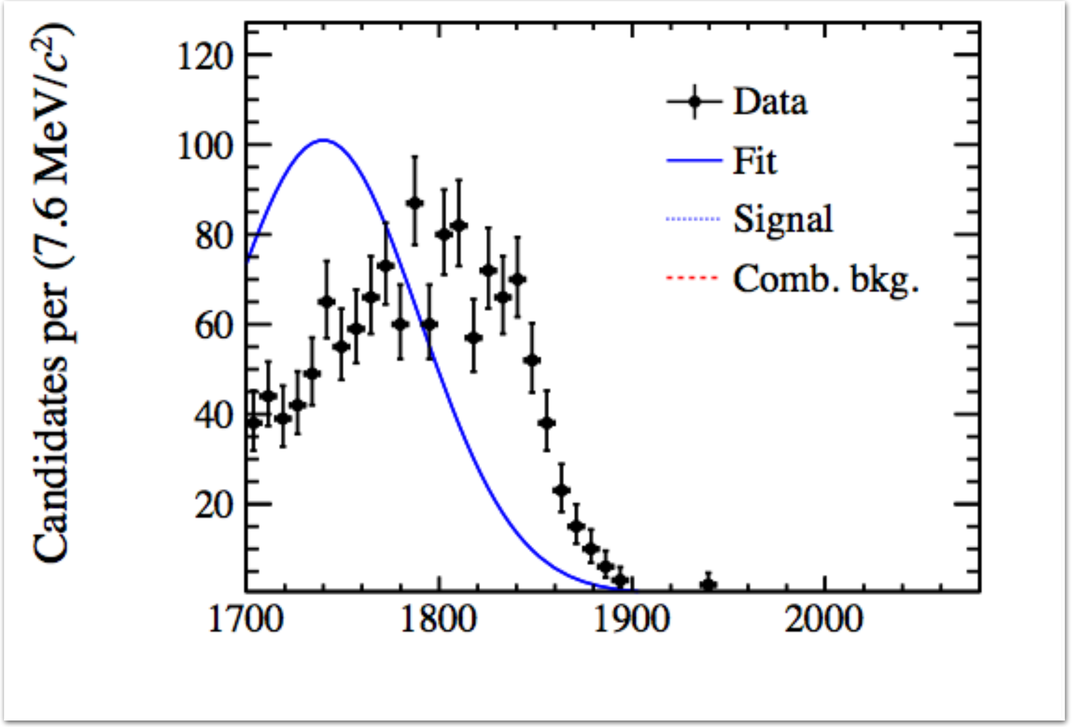
  ε ~ 90% for dimuon

  ε ~ 30% for multibody hadron



Machine Learning in LHCb

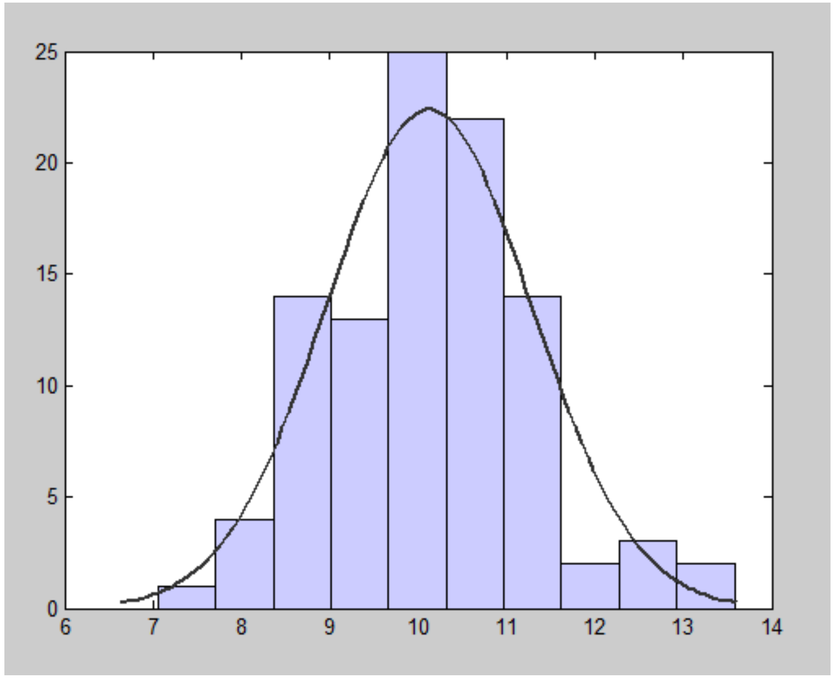
Fitting data



Fitting data

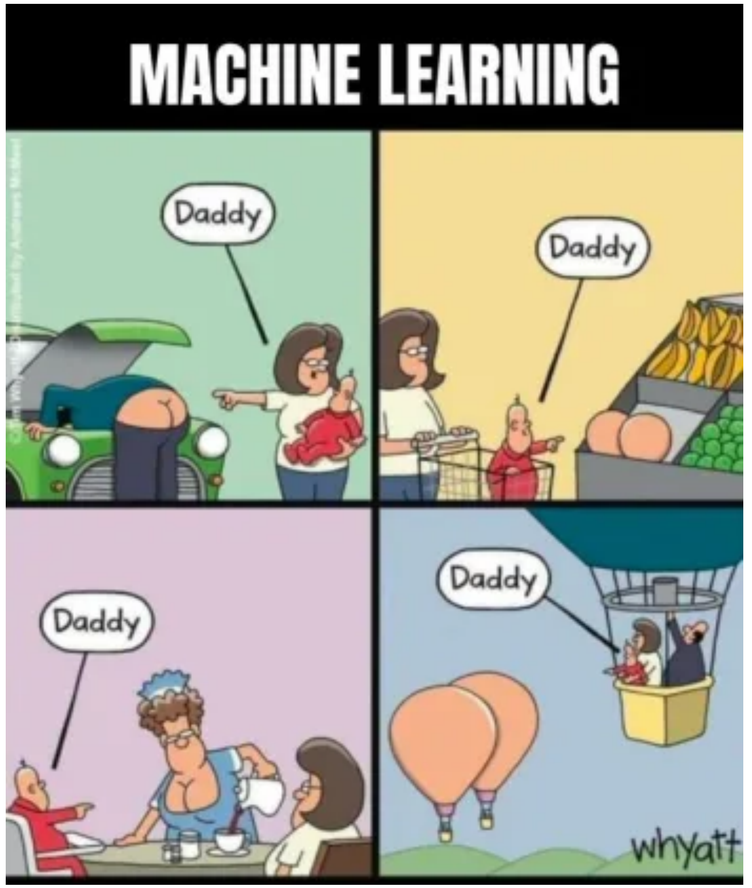
- Optimisation problem: find best set of parameters of model given data
- Data: histogram bin values $\{x_i, y_i\}$
- Model: Gaussian $f(x,\theta)$, where θ = {mean, width, norm}
- Loss function: $\chi^2 = \sum_i (y_i - f(x_i))^2$, 'goodness of fit'
- Learning: minuit2/migrad (based on gradient descent), updates free parameters ('weights') θ
- Stops when some criterium is met (Estimated Distance to Minimum < threshold)

—> Why is this (not) 'Machine Learning'?



Machine Learning in LHCb

Classification



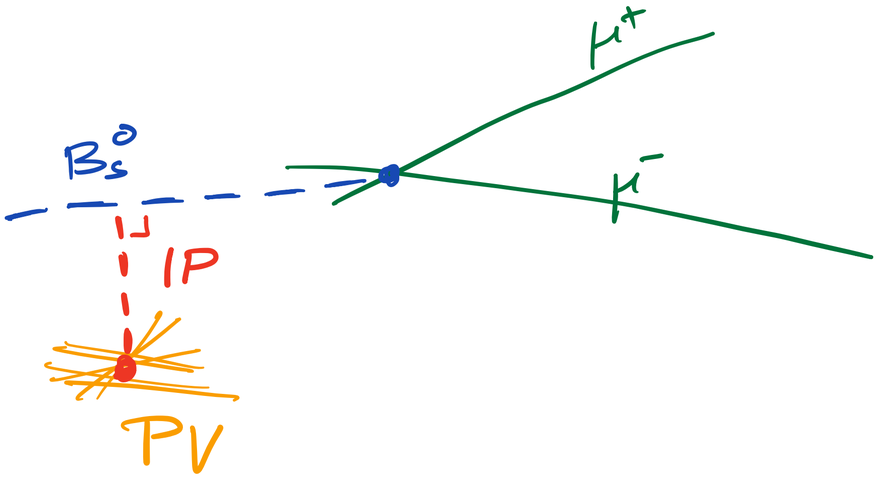
Signal vs background

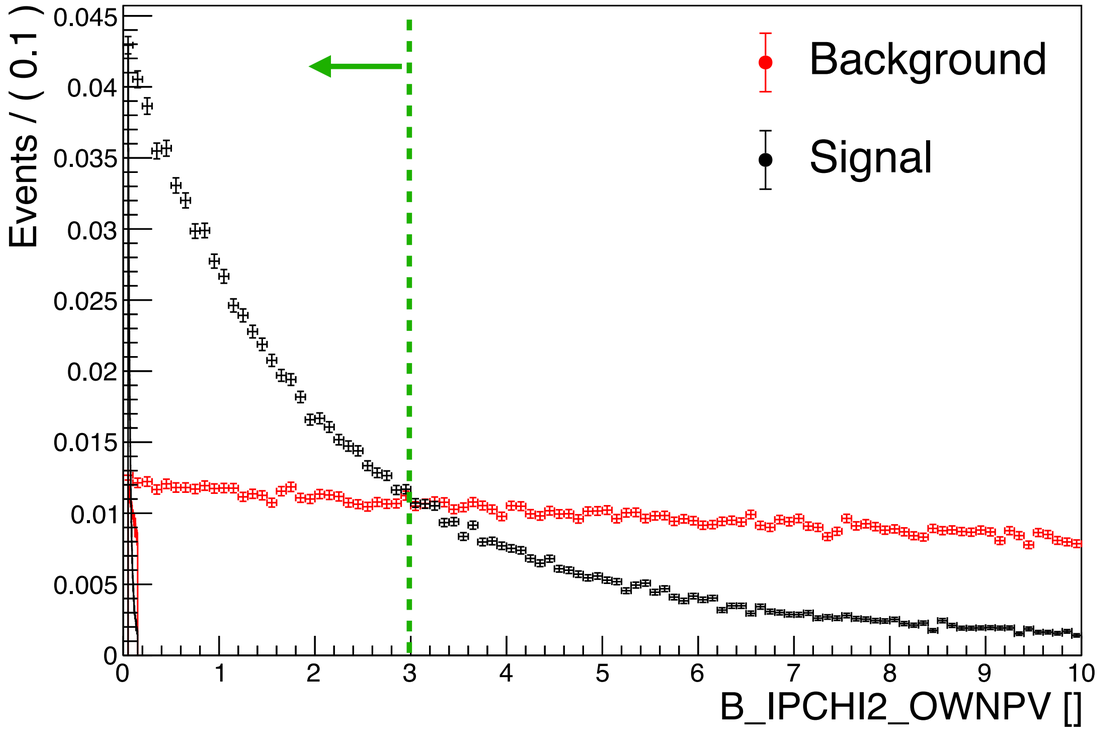
- Traditional multiple single-variable cuts, to improve S/B

Signal vs background

- Traditional multiple single-variable cuts, to improve S/B

  ✅ benefit: easy interpretable.





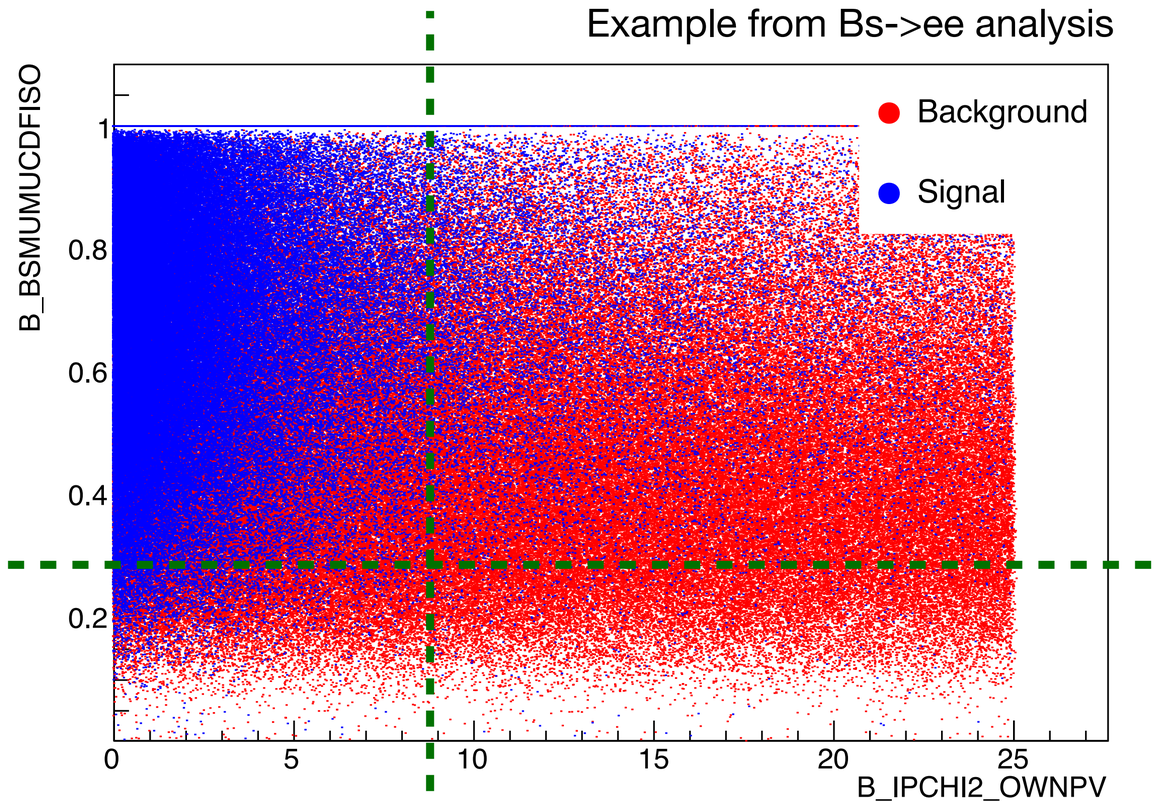
Signal vs background

- Traditional multiple single-variable cuts, to improve S/B

✅ benefit: easy interpretable.

❌ downside: no use of correlations

—> only ‘rectangular’ selections



Signal vs background

- Traditional multiple single-variable cuts, to improve S/B

  ✅ benefit: easy interpretable.

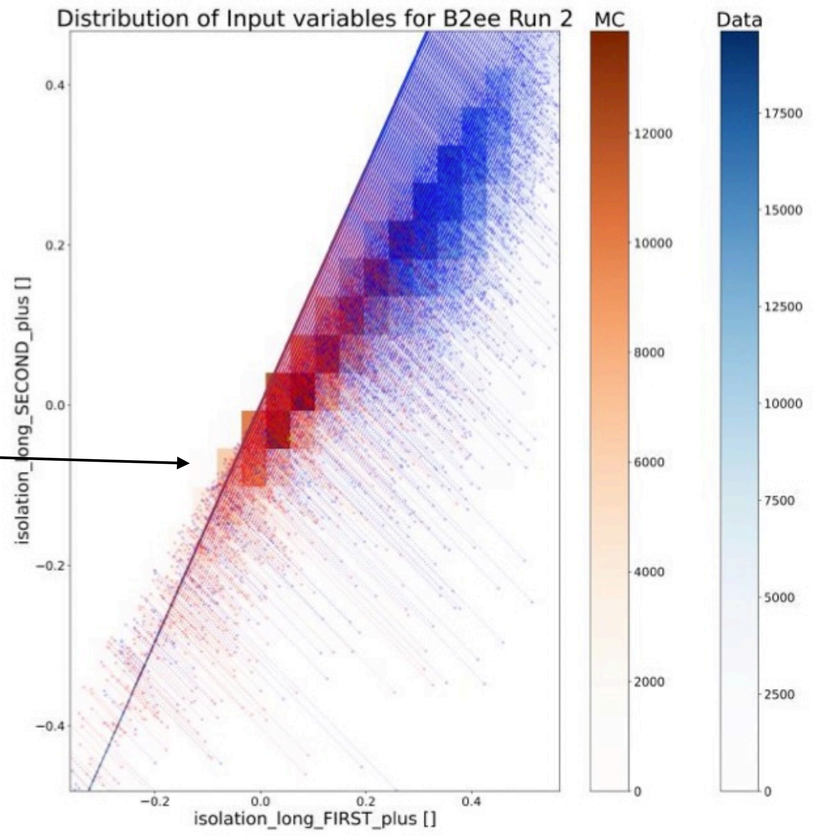
  ❌ downside: no use of correlations

- Some multivariate classifier options:

  —> Fischer discriminant

  —> (Boosted) Decision Tree

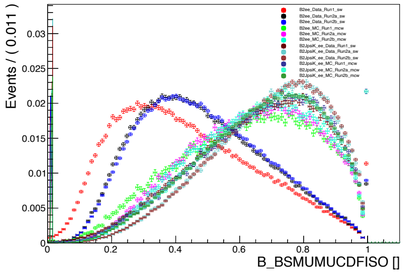
  —> Neural network

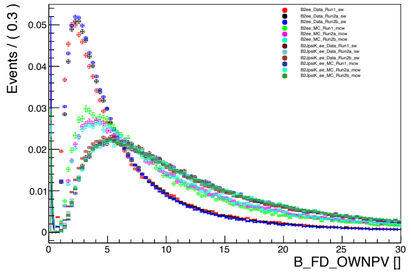


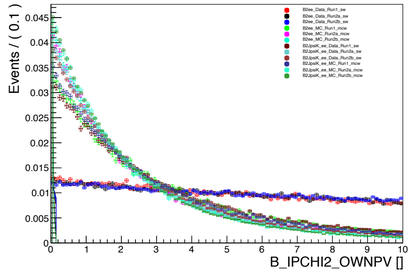
- Often combine input to form a *single* output variable, which holds maximal discriminating power between classes

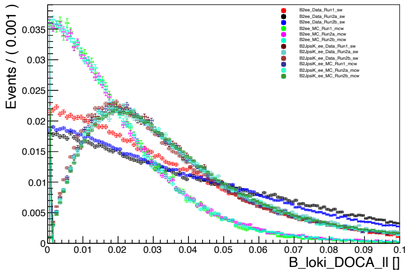
Signal vs background

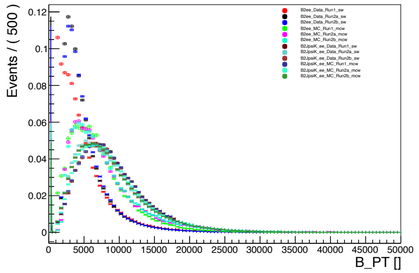
- BDT example in $B_s^0 \to e^+e^-$ analysis

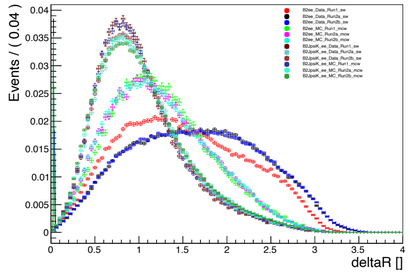


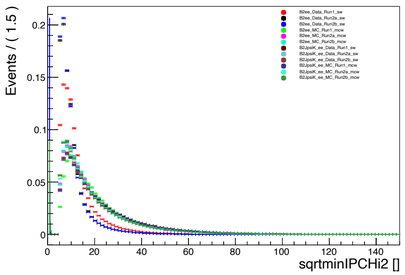


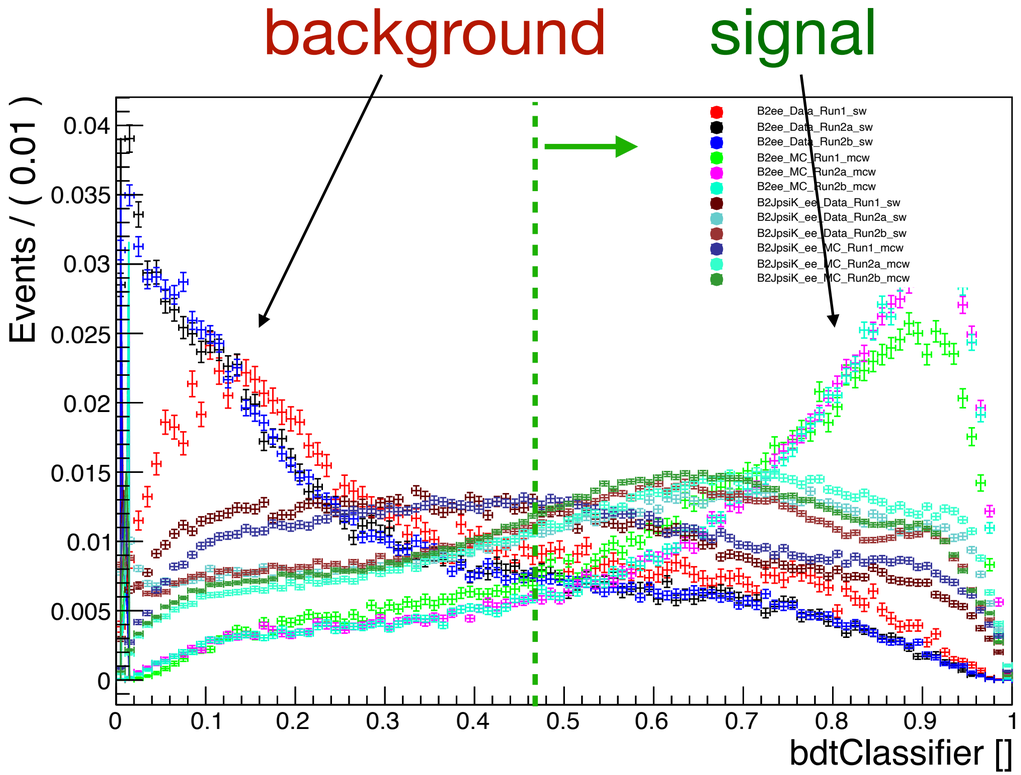












Maastricht University

S/B Classification - considerations

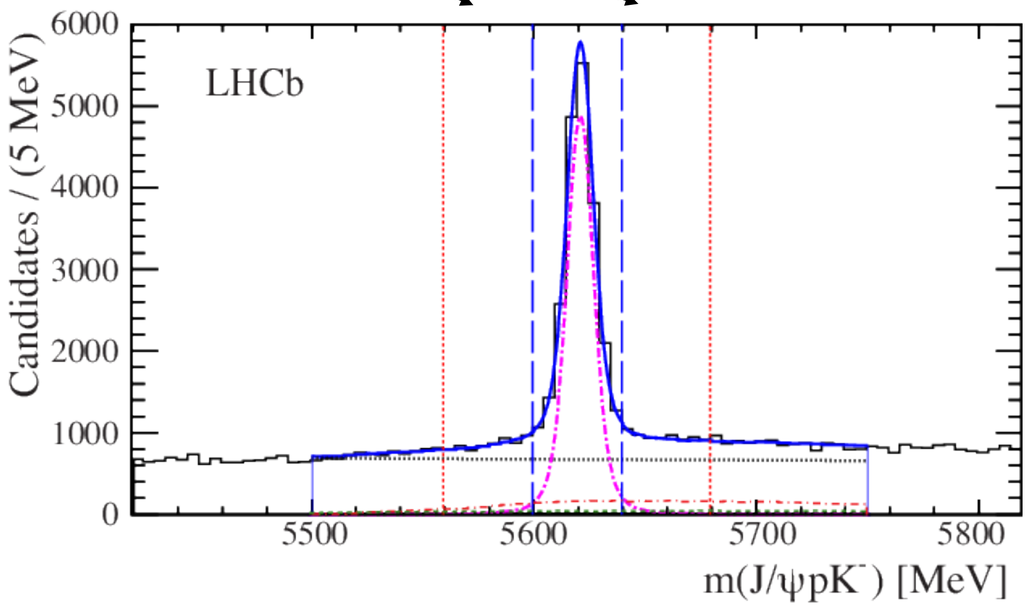
- What samples to use for training? *“What do you want to achieve?”*

Common in LHCb: signal over combinatorial background

→ Signal: simulation (MC) or background-subtracted data?

→ Background: inclusive MC (bb —> $\mu\mu$X) or ‘sidebands’ in data?

→ MC-data differences? [Q: is this bad?]



S/B Classification - considerations

- What samples to use for training? *“What do you want to achieve?”*

Common in LHCb: signal over combinatorial background

→ Signal: simulation (MC) or background-subtracted data?

→ Background: inclusive MC (bb —> $\mu\mu$X) or ‘sidebands’ in data?

→ MC-data differences? [Q: is this bad?]

[A: for training - only suboptimal. For efficiency determination: bad!]

S/B Classification - considerations

- What samples to use for training? *"What do you want to achieve?"*

Common in LHCb: signal over combinatorial background

→ Signal: simulation (MC) or background-subtracted data?

→ Background: inclusive MC (bb —> $\mu\mu$X) or 'sidebands' in data?

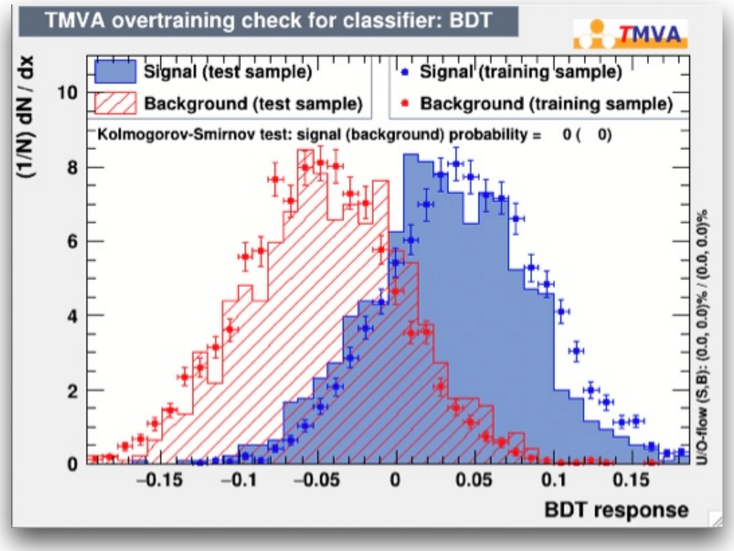
→ MC-data differences? [Q: is this bad?]

[A: for training - only suboptimal. For efficiency determination: bad!]

→ Overtraining: k-folding?

→ Calibration sample? [e.g. $B^+ \to J/\psi K^+$ for $B_s^0 \to \mu^+\mu^-$]

→ Cut on BDT output value, or bin in it for maximal sensitivity?



S/B Classification - considerations

- What

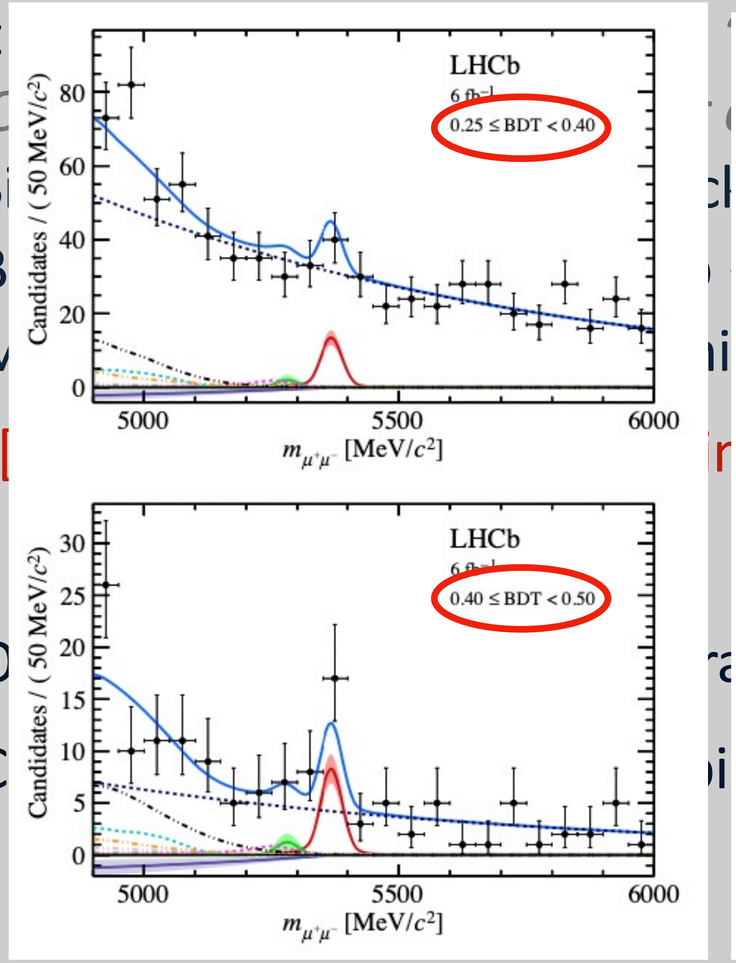
—> S

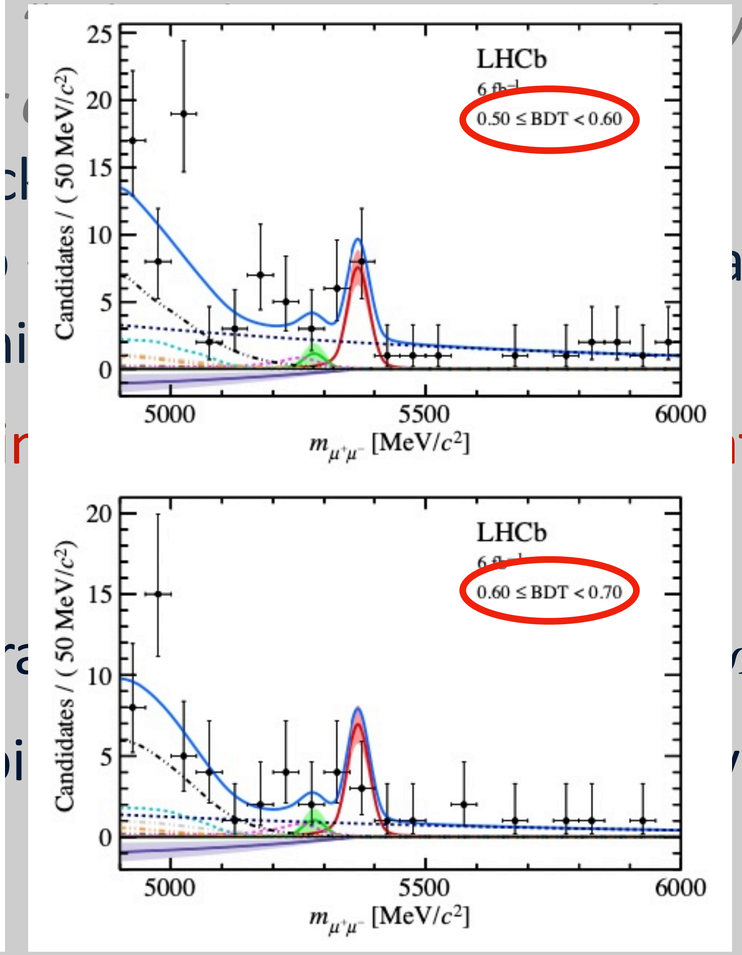
—> B

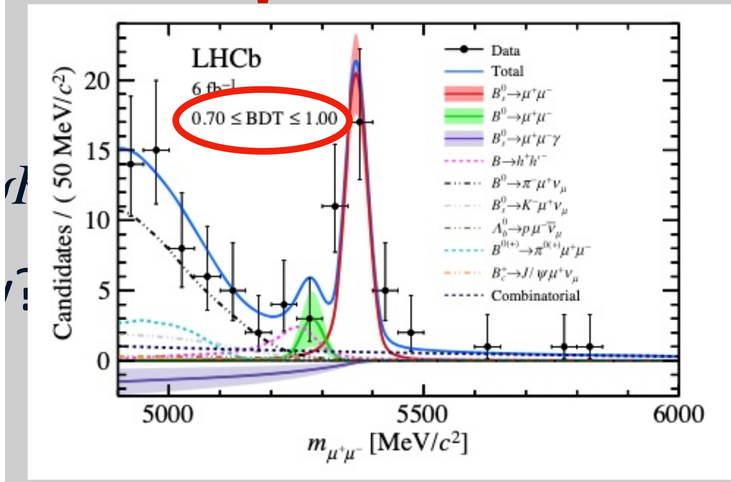
—> M

—> O

—> C



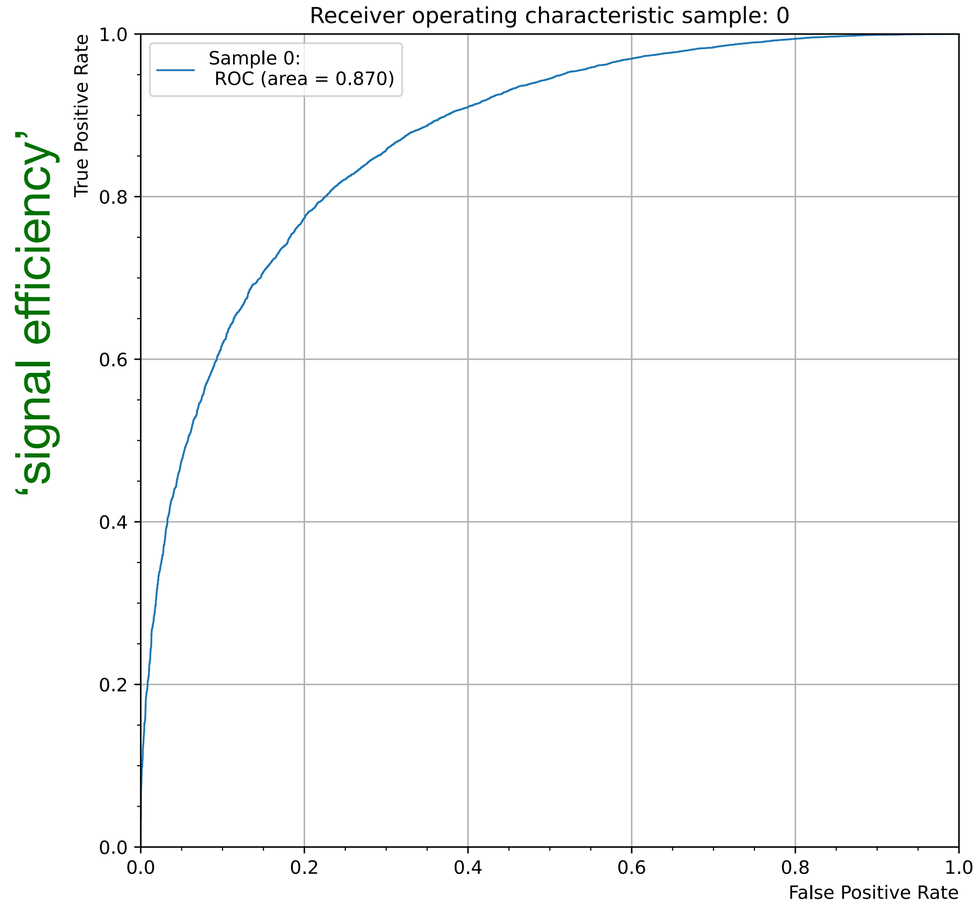




https://journals.aps.org/prd/abstract/10.1103/PhysRevD.105.012010

S/B Classification - considerations

- Compare performance of classifiers: signal efficiency vs background rejection (ROC)

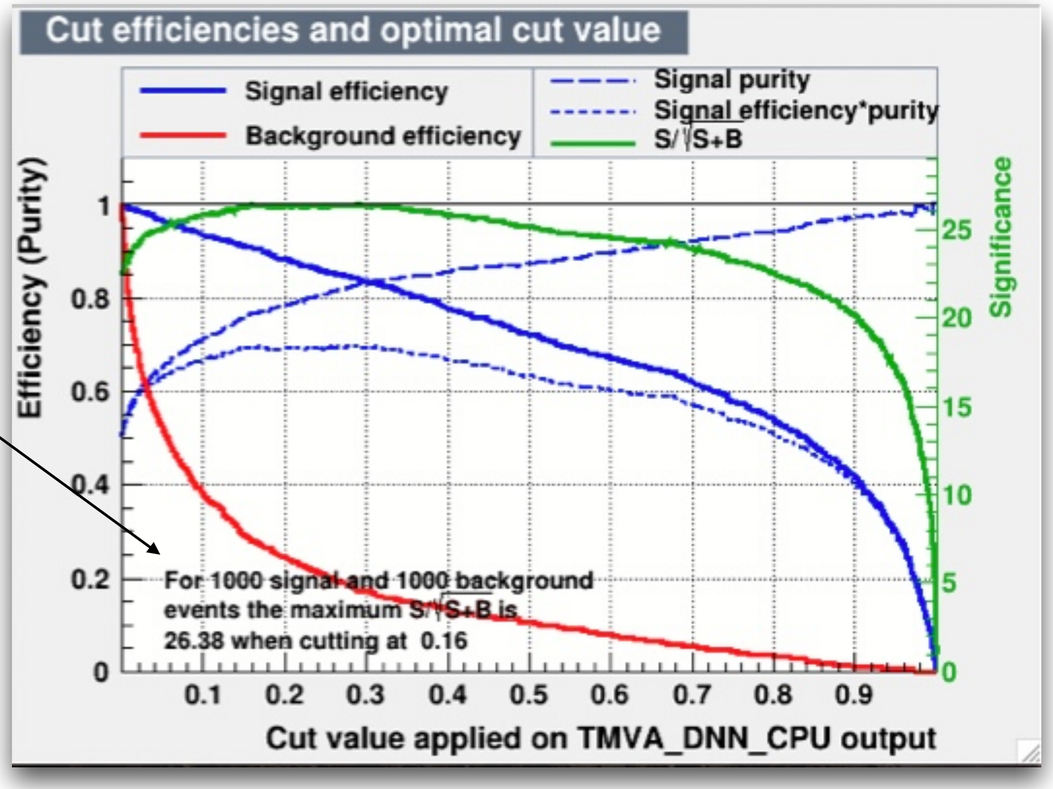


S/B Classification - considerations

- Compare performance of classifiers: signal efficiency vs background rejection (ROC)
- Choosing the optimal cut point: various ‘Figure of Merit’ (FoM):

$\rightarrow S/B$? $S/\sqrt{(S+B)}$?

(note: S,B are yields)



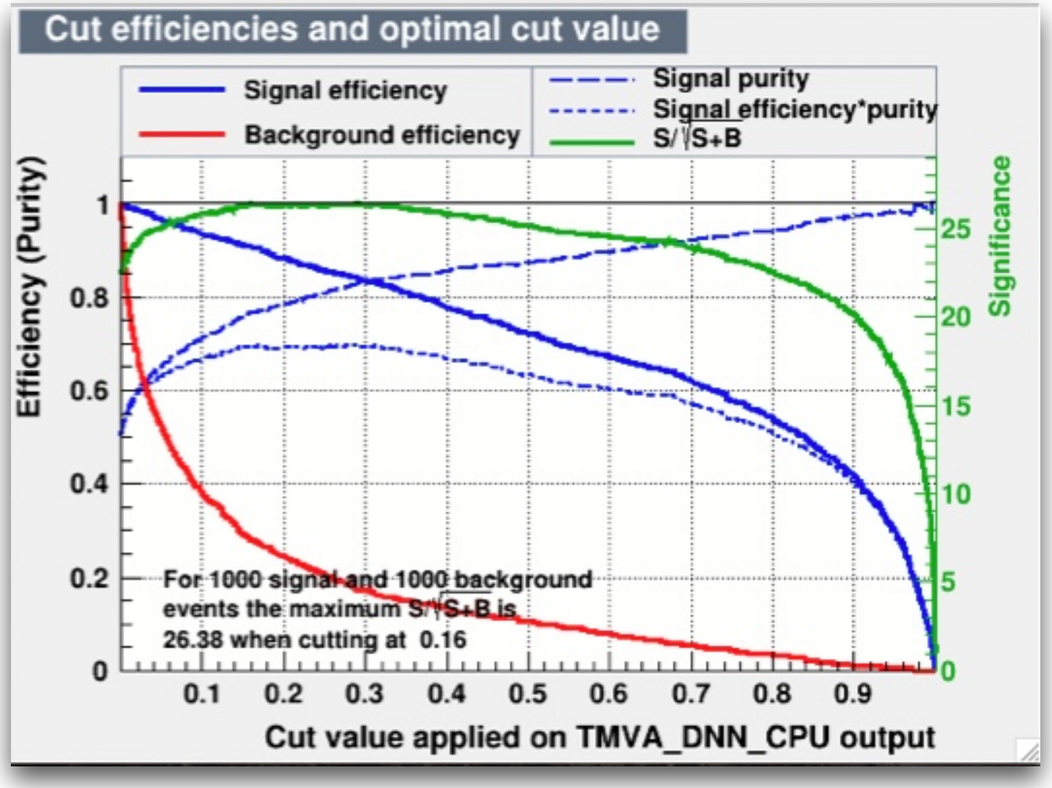
S/B Classification - considerations

- Compare performance of classifiers: signal efficiency vs background rejection (ROC)
- Choosing the optimal cut point: various ‘Figure of Merit’ (FoM):

→ S/B ? $S/\sqrt{(S+B)}$? (note: S,B are yields)

→ ‘Punzi’ FoM:
- independent of expected signal yield
- based on (poisson) errors,
- optimise for significance *a*

$$\frac{\epsilon(t)}{a/2+\sqrt{B(t)}}$$



https://inspirehep.net/literature/634798

High-level trigger

J. Phys.: Conf. Ser. 664 (2015) 082025

- Hlt2 trigger in Run2: 2,3,4 body 'Topological lines'

  → Generically capture 'B hadron' decay signatures

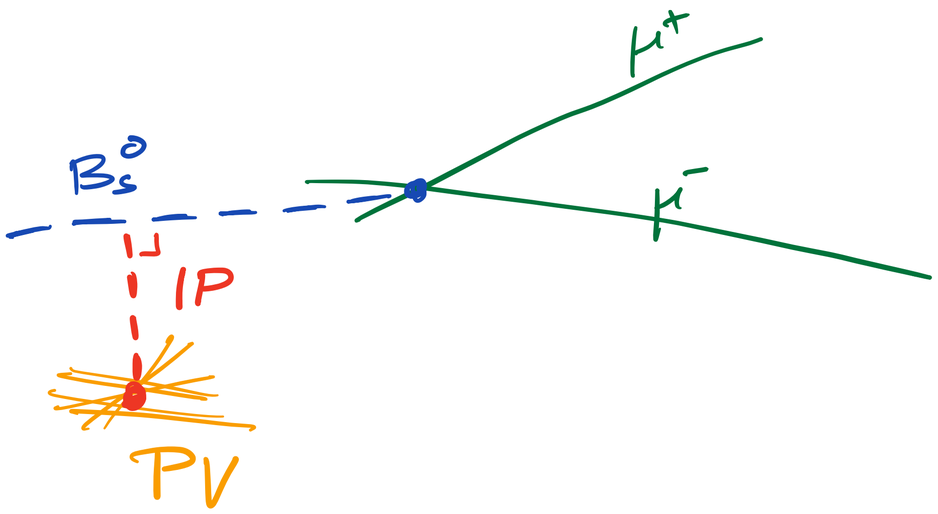


Table 3. HLT topological line description.

Track preselections:	
	$PT > 200$ MeV
	$IP_{\chi^2} > 4$
	$track_{\chi^2}/ndof < 2.5$
SV preselections:	
	$vertex_{\chi^2} < 10$
	$1 < mcor < 10$ GeV
	$2 < \eta < 5$ (PV to SV)
	$N(\text{tracks with } IP_{\chi^2} < 16) < 2$
Analysis variables:	
	n, mcor, sum PT, $vertex_{\chi^2}$, η, $FD_{\chi 2}$, min PT, IP_{χ^2}, N(tracks), $N(\text{tracks with } IP_{\chi^2} < 16)$
Output rate:	2-4 kHz

https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.117.061803

High-level trigger

J. Phys.: Conf. Ser. 664 (2015) 082025

- Hlt2 trigger in Run2: 2,3,4 body 'Topological lines'

  ⟶ Generically capture 'B hadron' decay signatures

  ⟶ Improvement over non-BBDT 'topo' lines used in Run1: **50-80%**

  ⟶ Inference needs to be 'fast' —> 'Bonsai' BDT (pruning forest, discretise inputs)

  ⟶ Majority of LHCb papers makes use of physics from these lines!

Table 3. HLT topological line description.

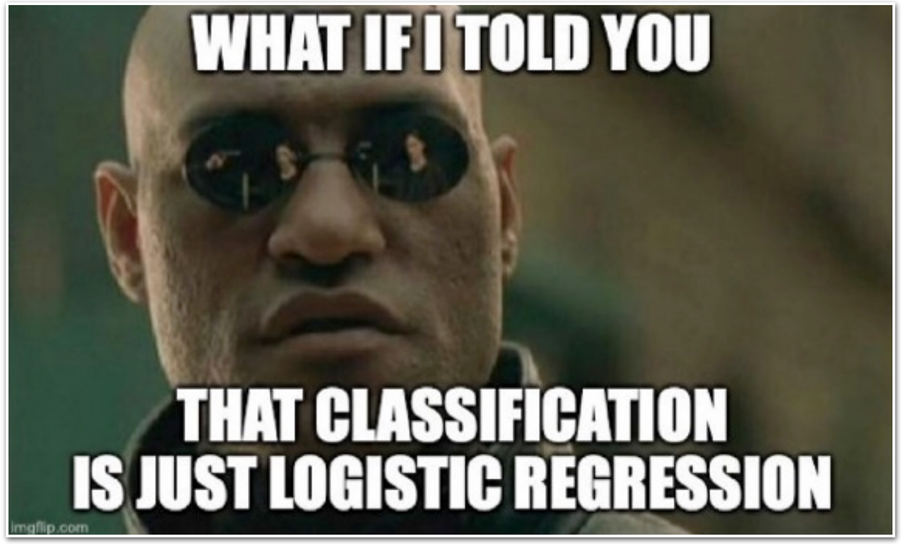
Track preselections:	
	$PT > 200$ MeV
	$IP_{\chi^2} > 4$
	$track_{\chi^2}/ndof < 2.5$
SV preselections:	
	$vertex_{\chi^2} < 10$
	$1 < mcor < 10$ GeV
	$2 < \eta < 5$ (PV to SV)
	$N(\text{tracks with } IP_{\chi^2} < 16) < 2$
Analysis variables:	
	n, mcor, sum PT, $vertex_{\chi^2}$, η, FD_{χ_2}, min PT, IP_{χ^2}, N(tracks), $N(\text{tracks with } IP_{\chi^2} < 16)$
Output rate:	2-4 kHz

hadronic calorimeter and a muon system. Online event selection is performed by a two-stage trigger. For this analysis, the first (hardware) stage selects muons in the muon system; the second (software) stage applies a full event reconstruction. Here the events are first selected by the presence of the muon or one of the hadrons from the D_s^- decay, after which a combination of the decay products is required to be consistent with the topological signature of a b-hadron decay. Simulated events are produced using the software described in Refs. [13–17].

https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.117.061803

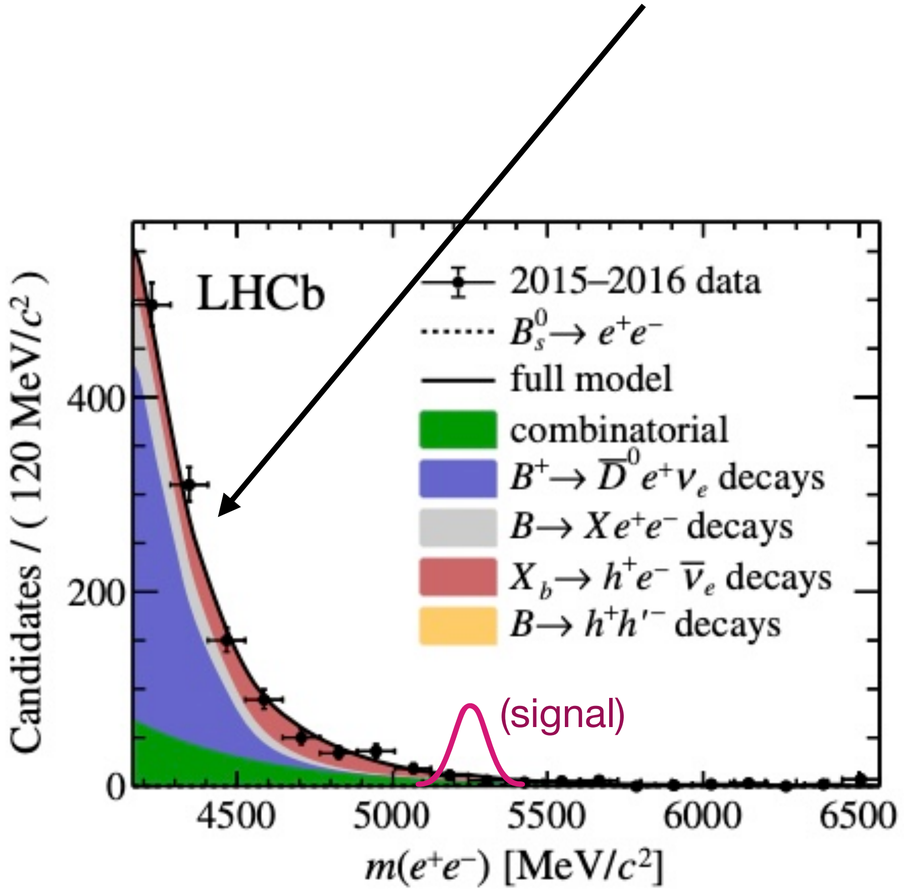
Machine Learning in LHCb

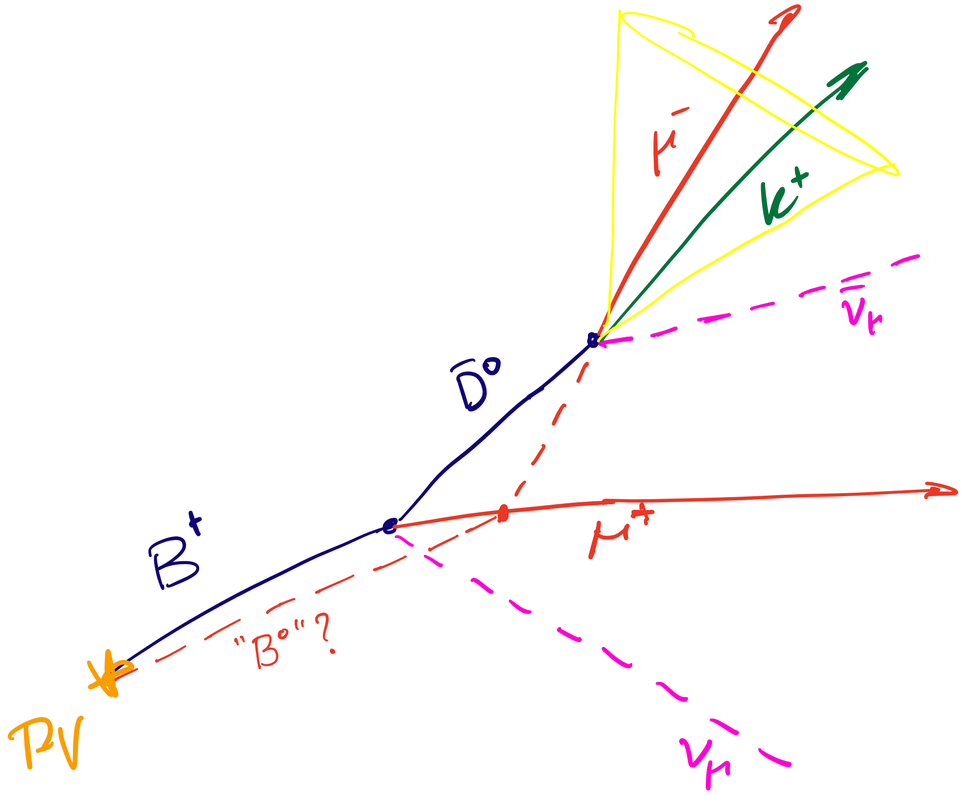
Regression



Isolation variables

- Partially reconstructed background in mass window



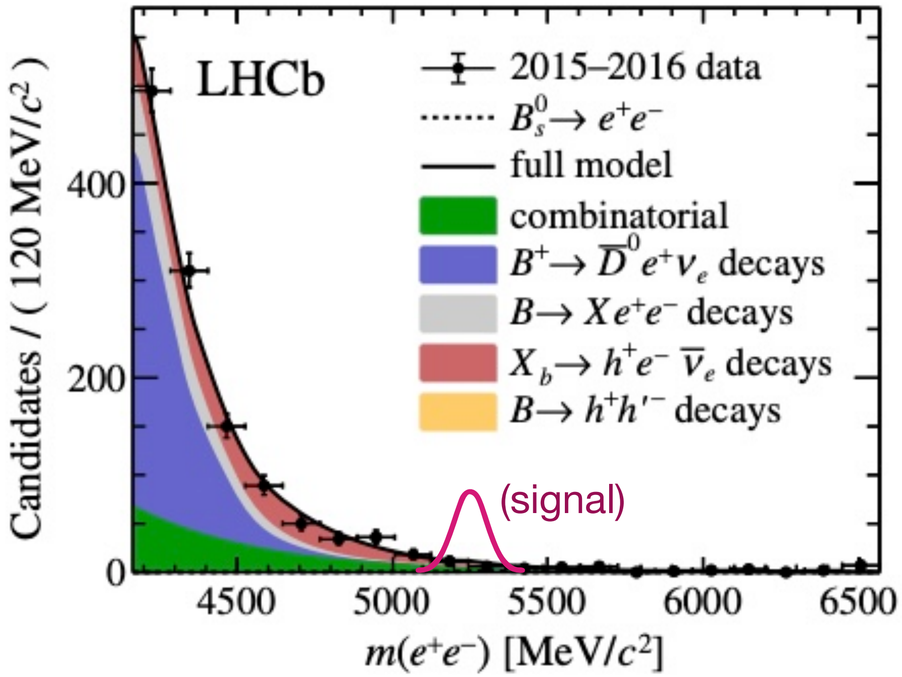


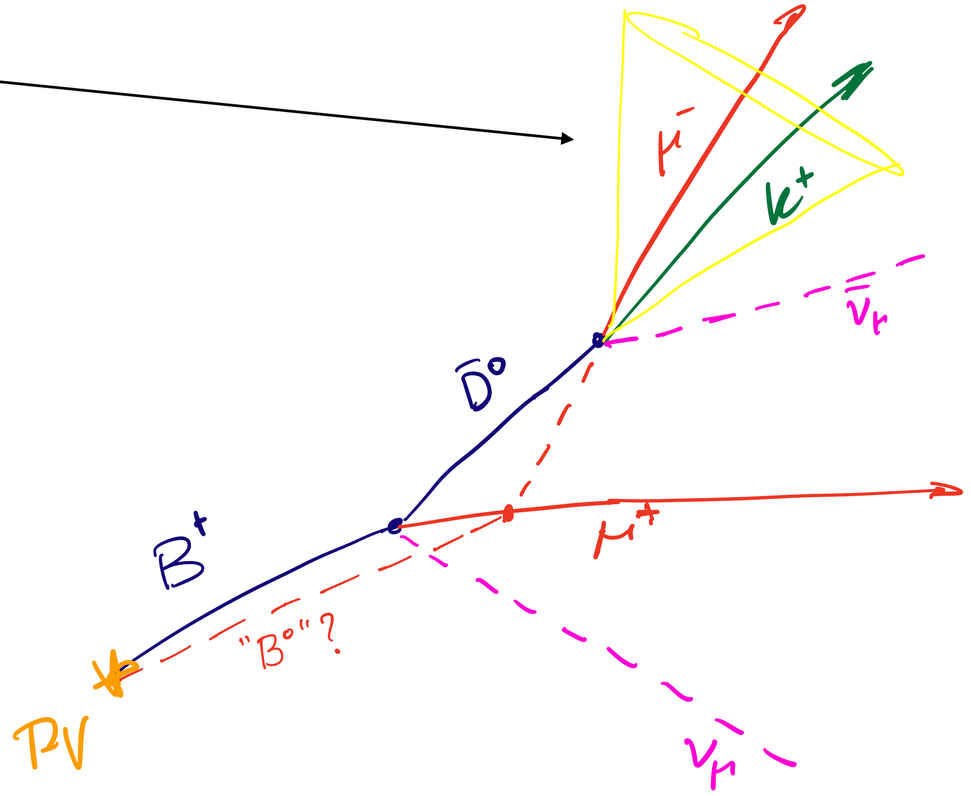
https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.124.211802

Isolation variables

- Partially reconstructed background in mass window

- Obtain information on ‘stand alone-ness’ of track of interest





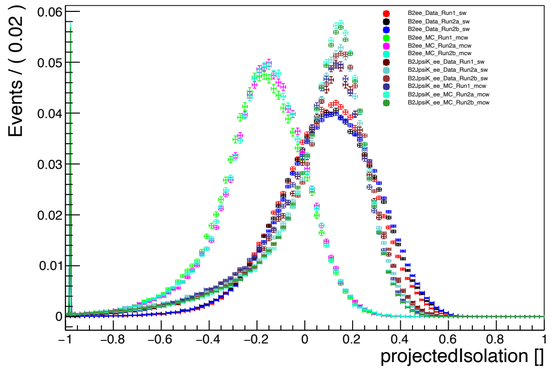
https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.124.211802

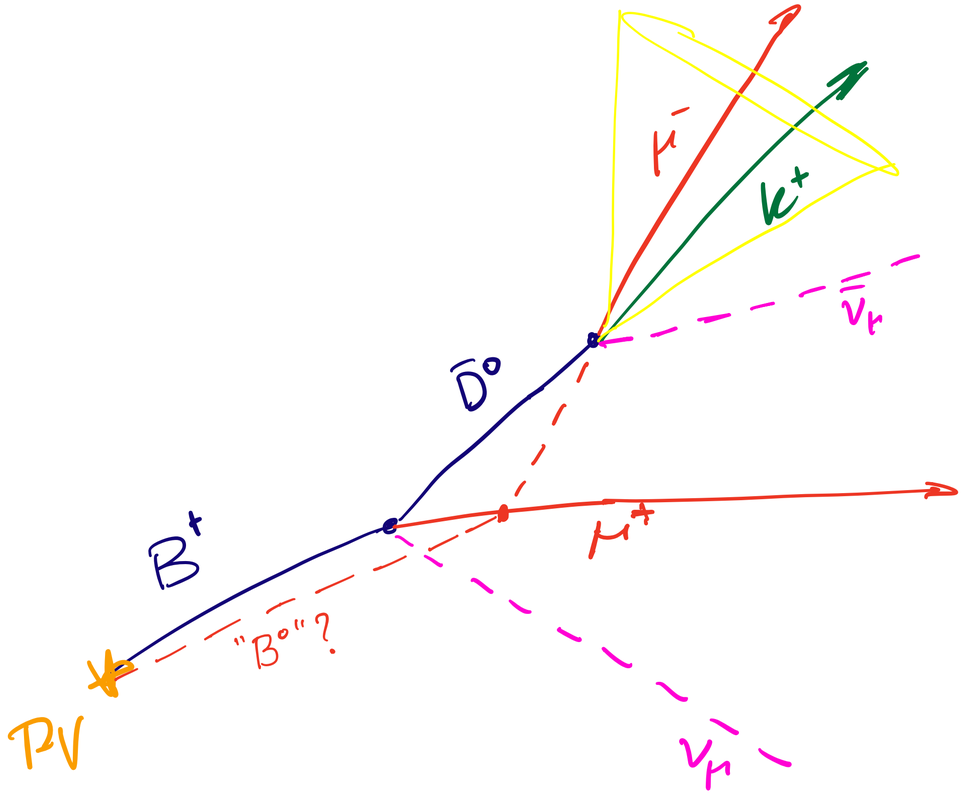
Isolation variables

- Partially reconstructed background in mass window

- Obtain information on ‘stand alone-ness’ of track of interest

- p_T, p, angles of other found tracks w.r.t. main track:
  → input for BDT
  → Use for further processing (or cut)
  → Efficiency depends on e.g. event ‘occupancy’





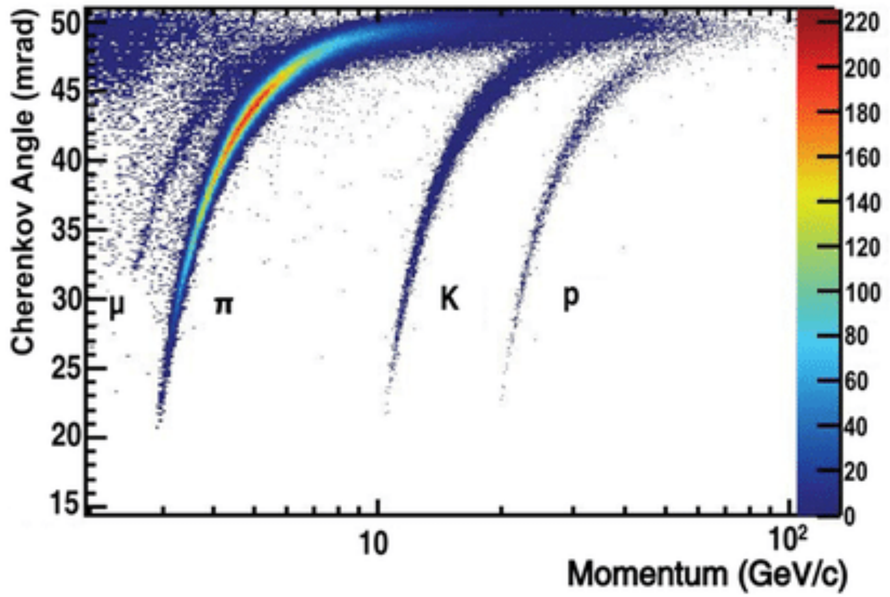
https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.124.211802

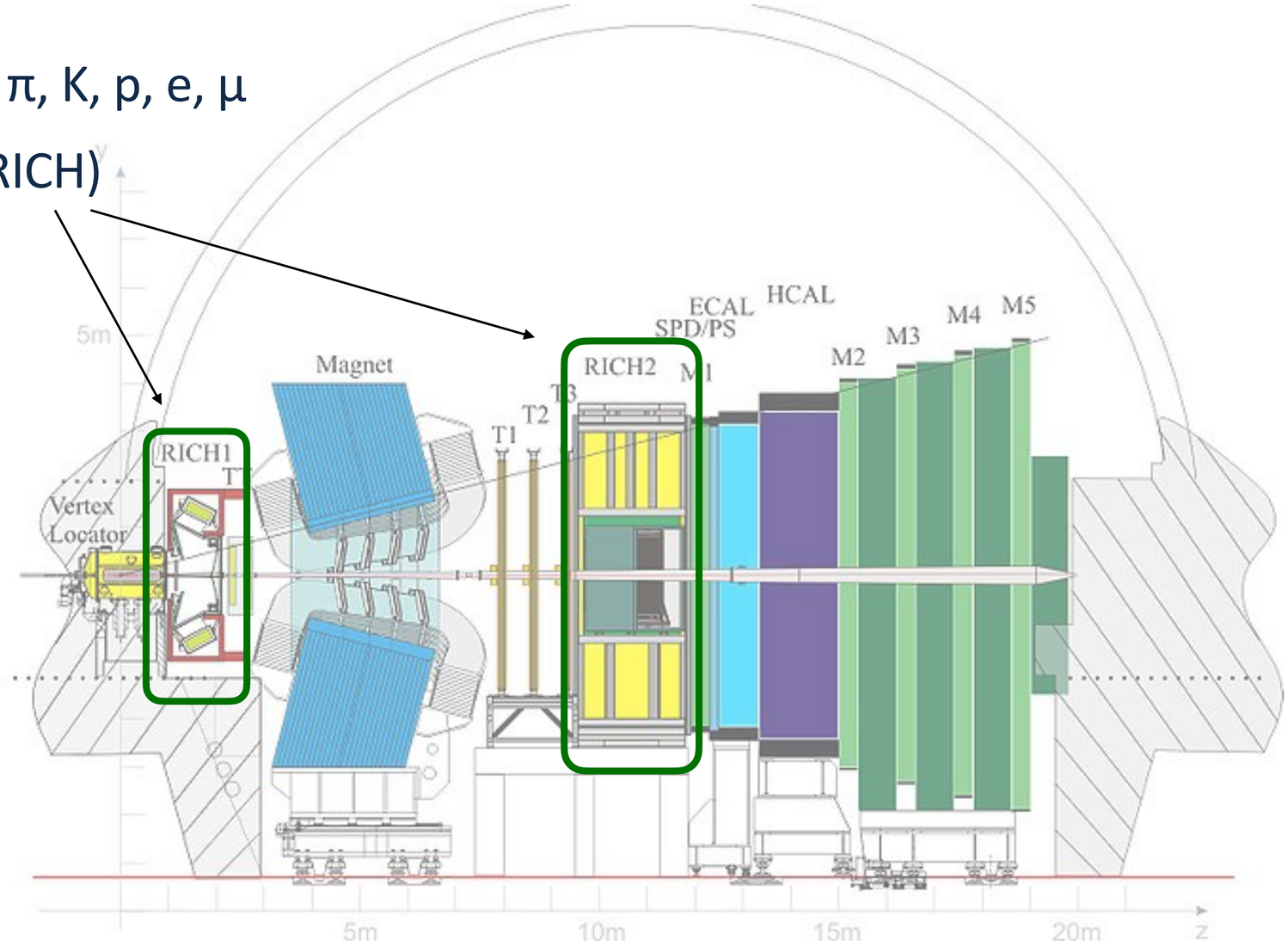
(charged) Particle ID in LHCb

- Identify particles (→ obtain mass) of π, K, p, e, μ
- Ring Imaging Cherenkov Detectors (RICH)

→ Velocity, $p = \gamma m v$

$$\cos\theta_c = \frac{c}{nv}$$





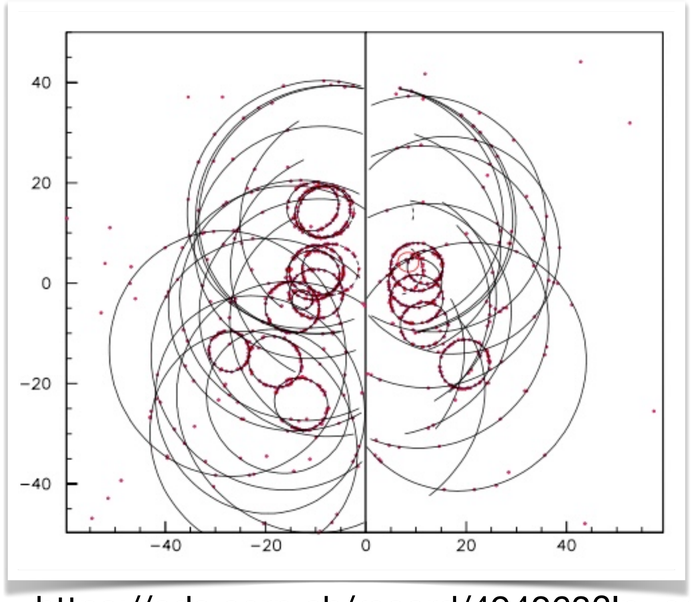
https://www.researchgate.net/publication/352081244_Particle_Identification_at_FCC-ee

(charged) Particle ID in LHCb

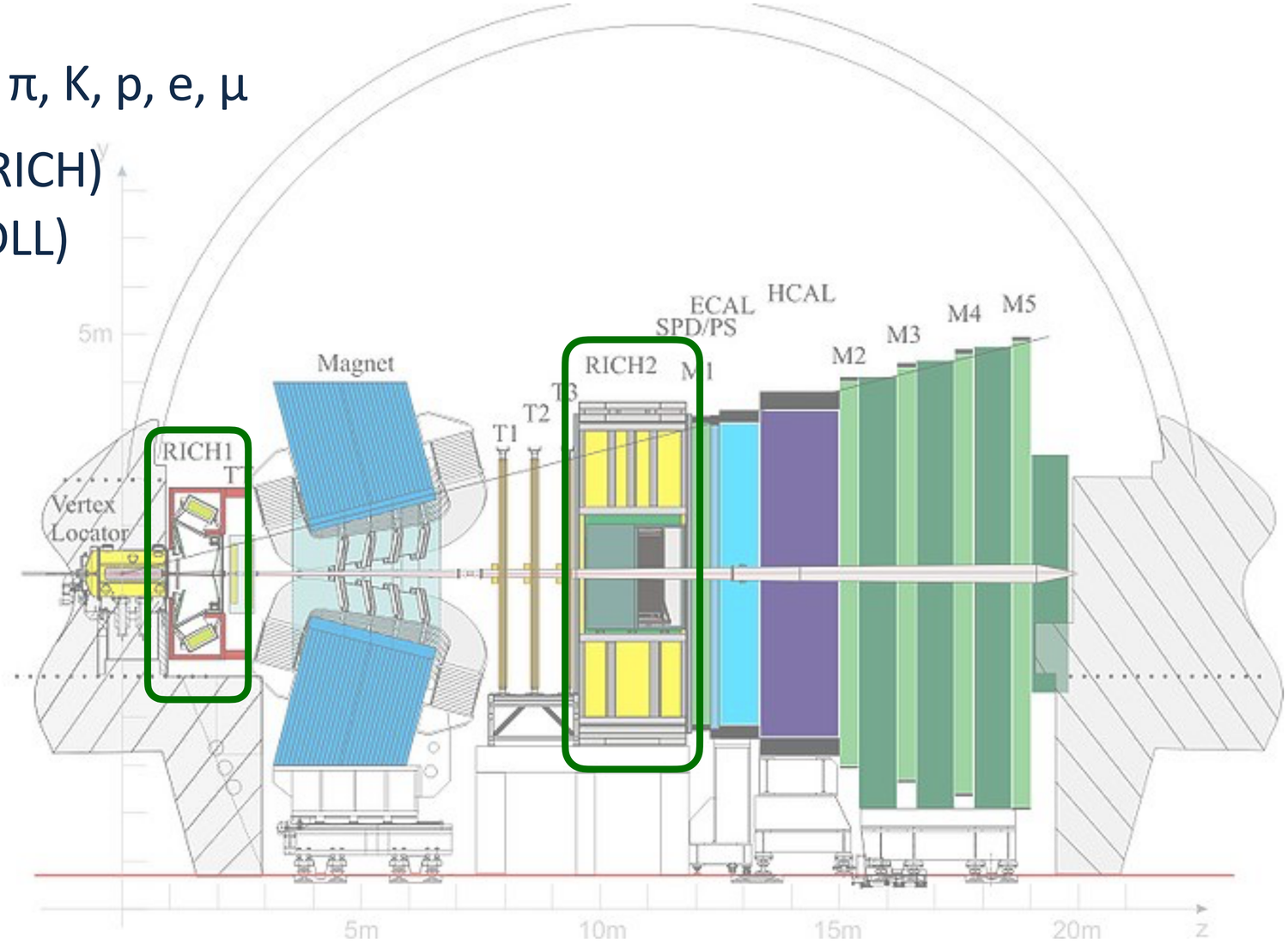
- Identify particles (→ obtain mass) of π, K, p, e, μ

- Ring Imaging Cherenkov Detectors (RICH)

  → In reality: Construct likelihoods (DLL)
  under particle hypotheses

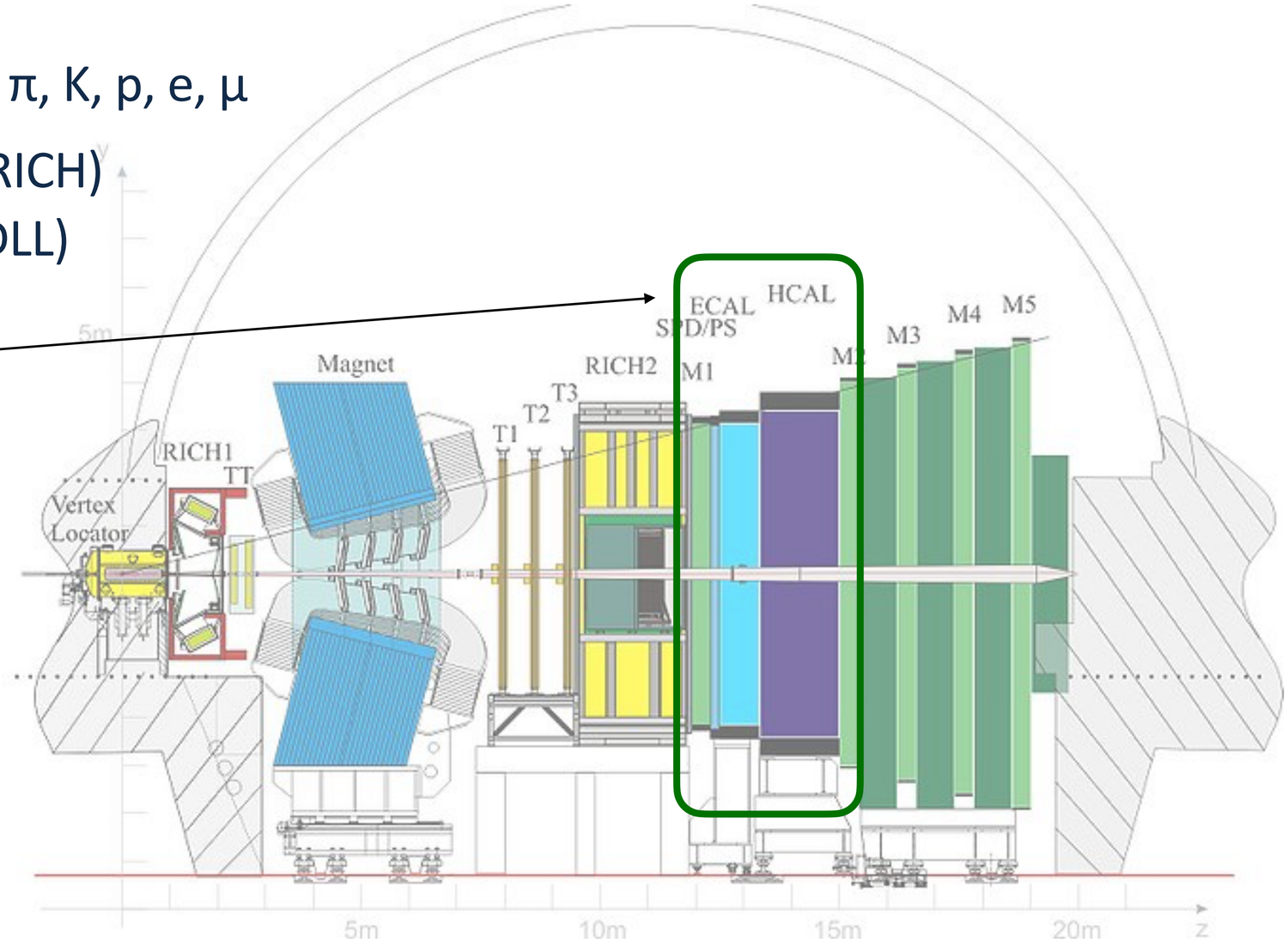


https://cds.cern.ch/record/494263?ln=en



(charged) Particle ID in LHCb

- Identify particles (→ obtain mass) of π, K, p, e, μ
- Ring Imaging Cherenkov Detectors (RICH)
  → In reality: Construct likelihoods (DLL)
  under particle hypotheses
- Calorimeters
  → Energy, $E^2 = p^2 + m^2$
  → electron: $E_{ECAL}/p \sim 1$



(charged) Particle ID in LHCb

- Identify particles (→ obtain mass) of π, K, p, e, μ

- Ring Imaging Cherenkov Detectors (RICH)

  → In reality: Construct likelihoods (DLL)
  under particle hypotheses

- Calorimeters

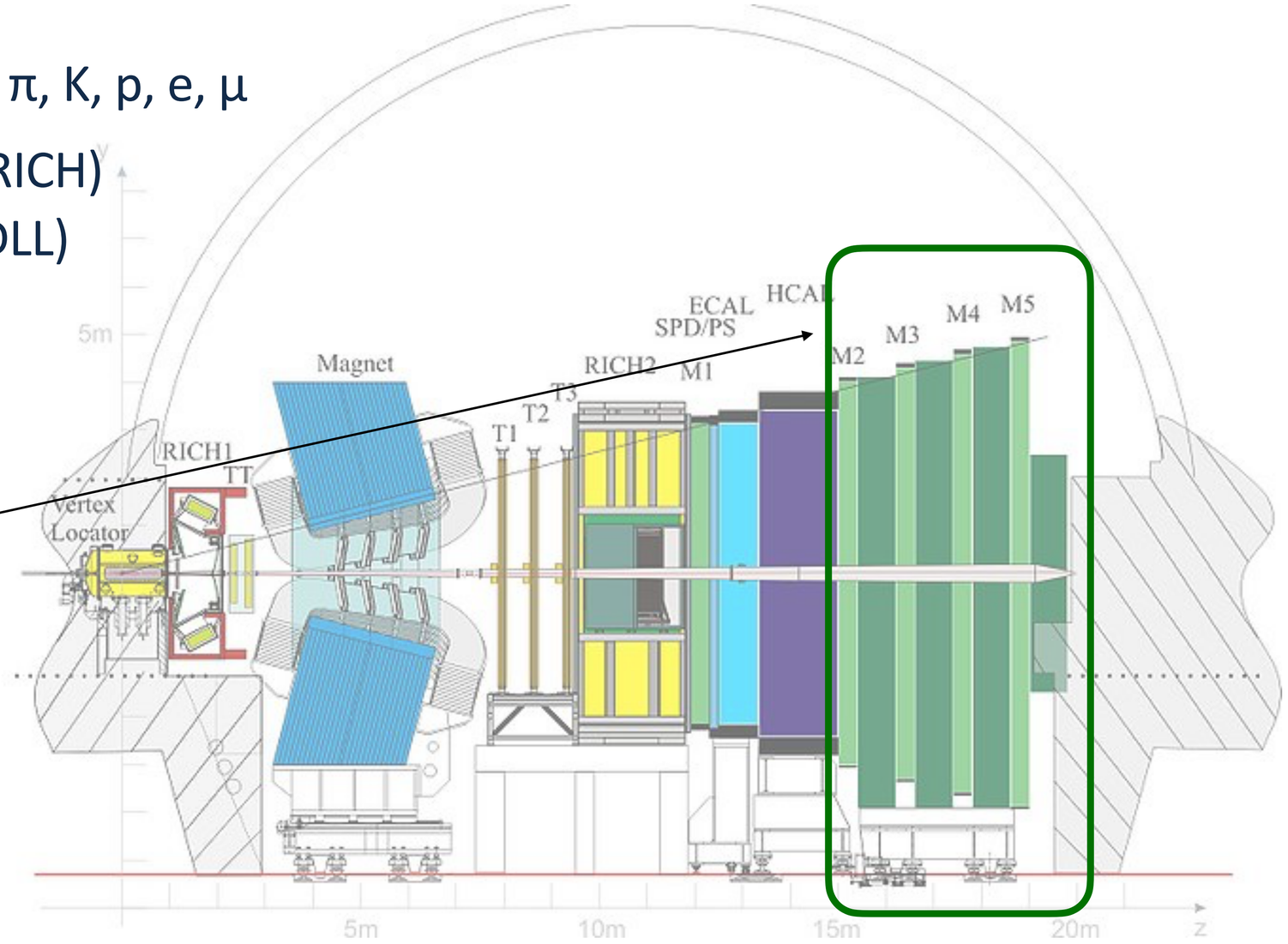
  → Energy, $E^2 = p^2 + m^2$

  → electron: $E_{ECAL}/p \sim 1$

- Muon system

  → mostly muons pass through

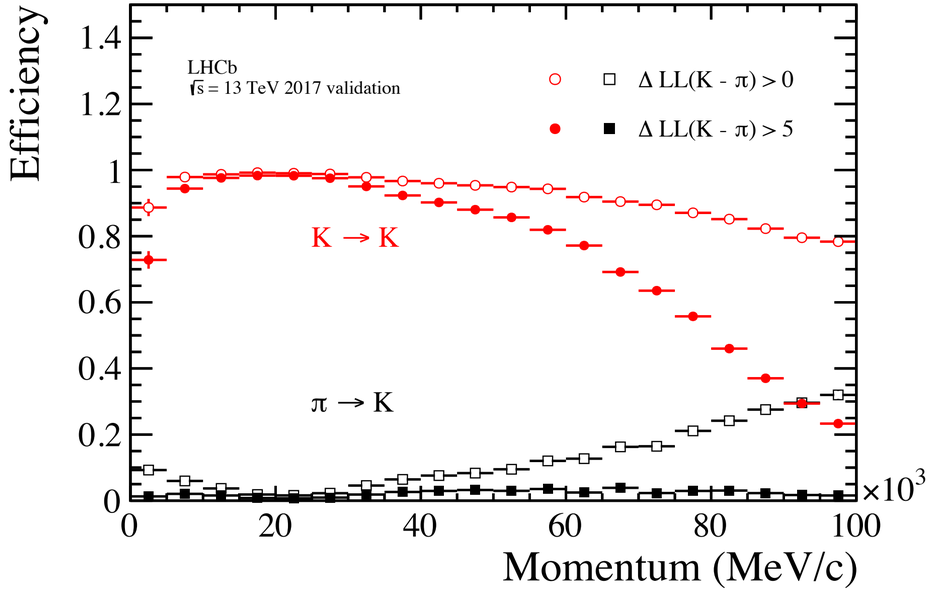
  → track extrapolation
  & hit finding



(charged) Particle ID in LHCb

- global DLL obtained by adding likelihoods (DLL) of all subsystems
  → e.g. Kaon ID: particle DLL$(K - \pi) > 5$

- What about correlations between systems?
  (→ e.g. muon hits make it very likely to not be a Kaon)



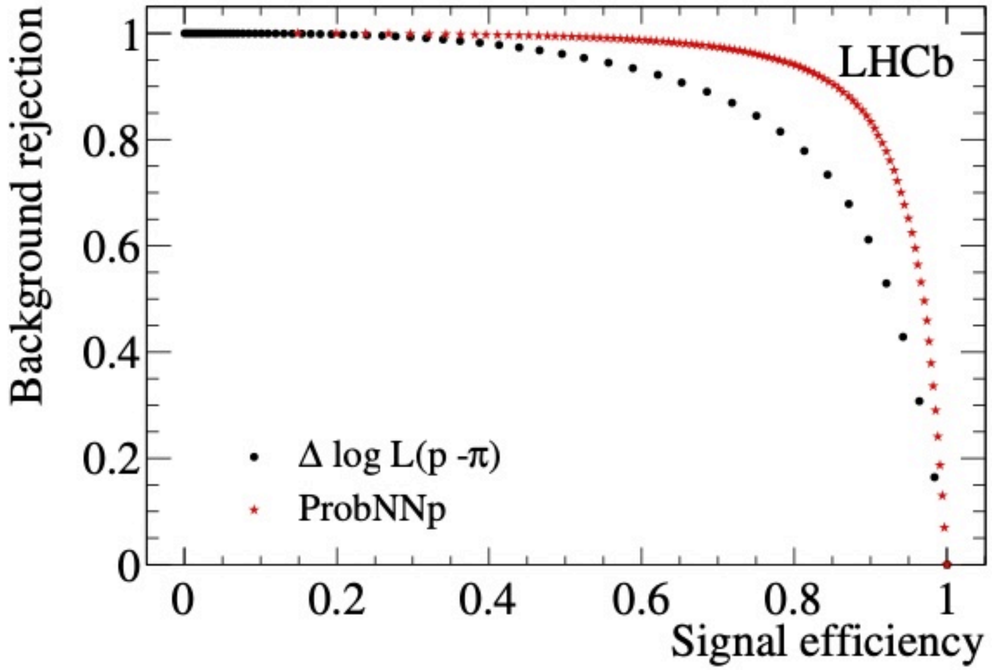
https://inspirehep.net/literature/1841673

(charged) Particle ID in LHCb

- global DLL obtained by adding likelihoods (DLL) of all subsystems
  → e.g. Kaon ID: particle DLL$(K - \pi) > 5$

- What about correlations between systems?
  (→ e.g. muon hits make it very likely to not be a Kaon)

- Neural-net based approach: ('probNN')
  → Use 'basic' subdetector information:
  p, pT, likelihoods, distance to z-axis, hasHits, cherenkov thresholds, calo cluster chi2, …



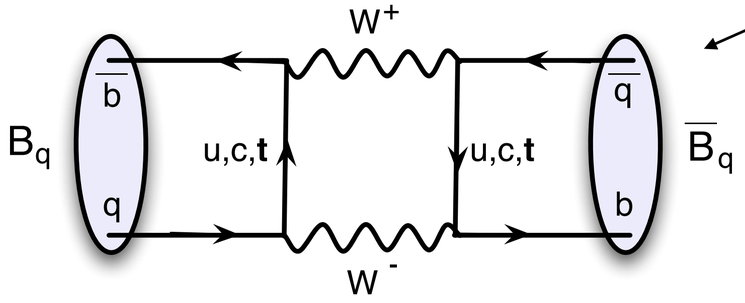
https://iopscience.iop.org/article/10.1088/1742-6596/1085/4/042038/pdf

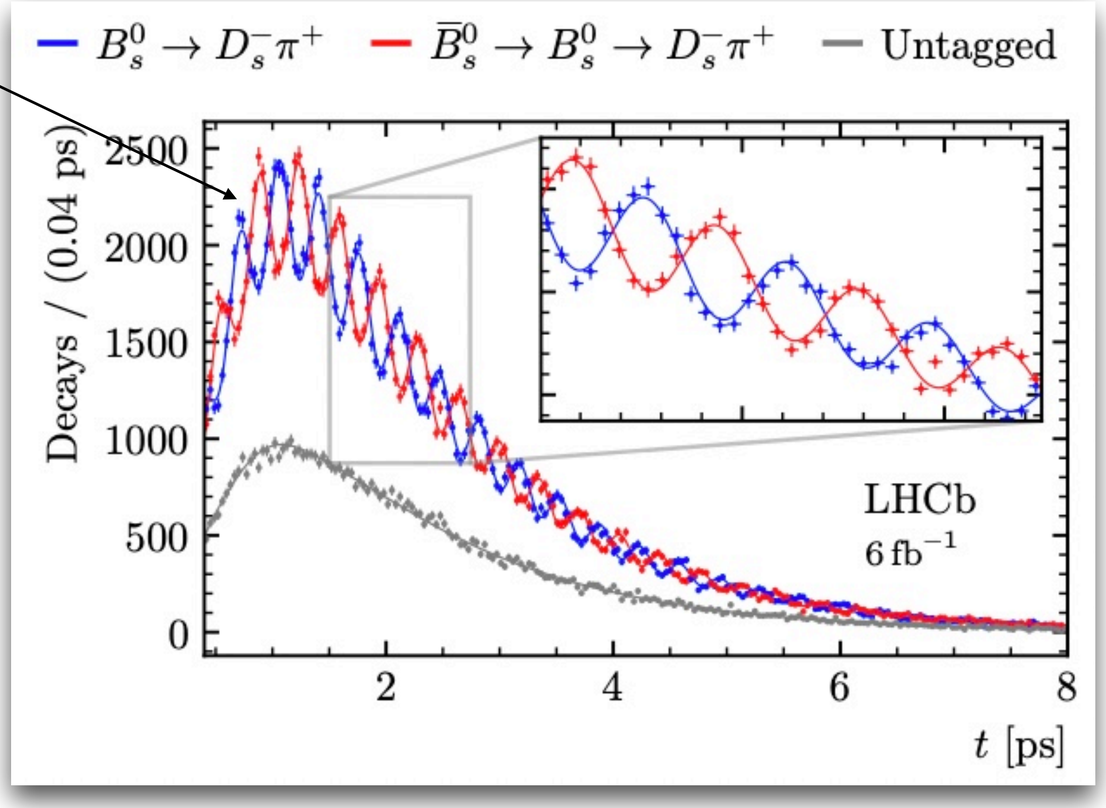
https://gitlab.cern.ch/lhcb/Rec/-/blob/run2-patches/Rec/ChargedProtoANNPID/src/ChargedProtoANNPIDCommonBase.h

B flavour tagging

- For CP violation analyses: need to know if particle was <u>created</u> as B_s^0 or $\bar{B}_s^0$

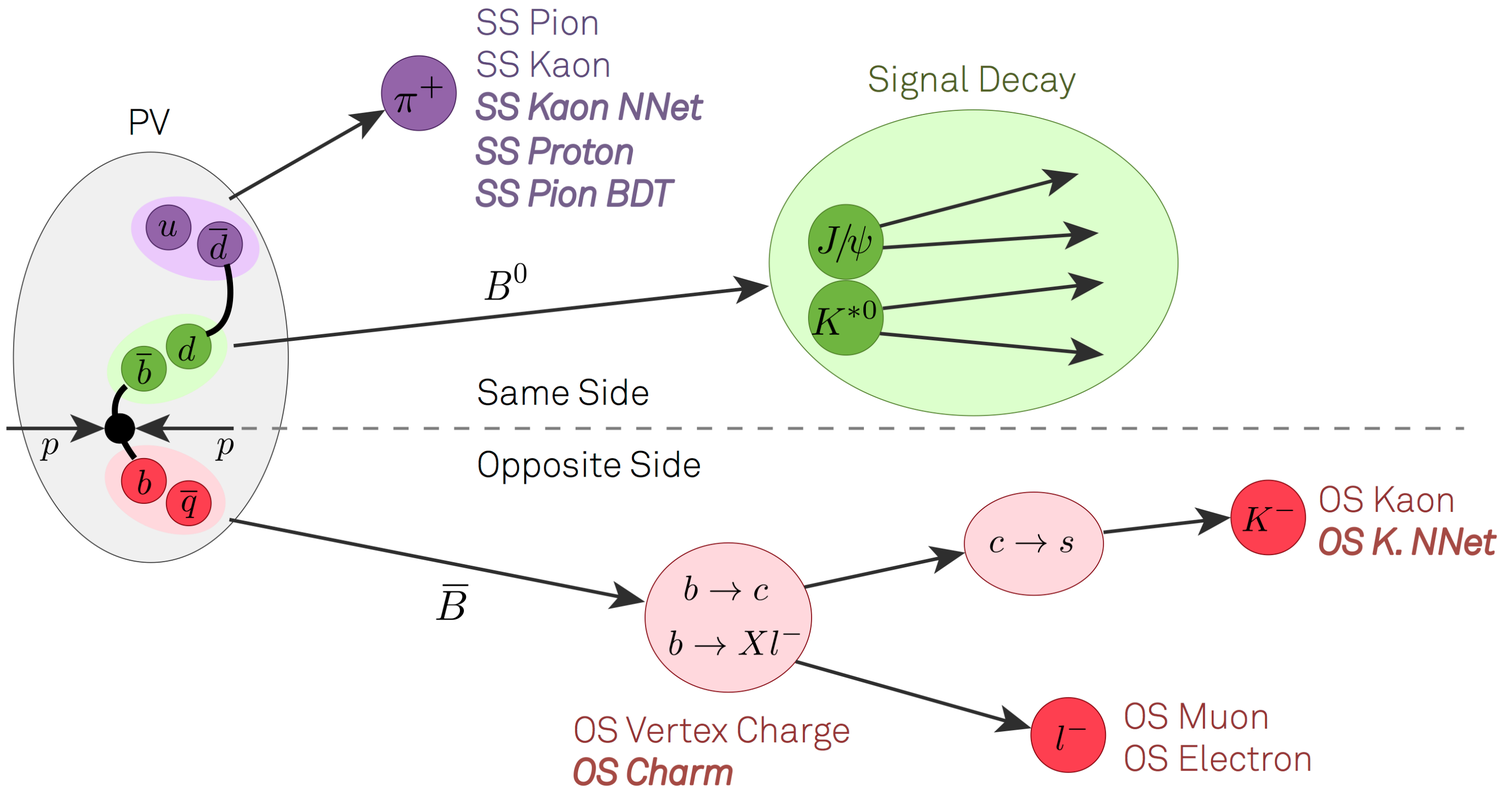
mixing frequency Δm_s





→ How would you do it?

B flavour tagging



Effective tagging power: ~5-7%

$$\overline{\varepsilon}_{\text{eff}} = \varepsilon_{\text{tag}} (1 - 2\overline{\omega})^2$$

Tagging decision

Mistag fraction

Tagger	ε [%]	ω [%]	$\varepsilon\langle D^2\rangle = \varepsilon(1-2\omega)^2$ [%]
OSμ	0.915 ± 0.053	30.713 ± 0.434	1.361 ± 0.062
OSe	4.451 ± 0.038	34.038 ± 0.604	0.454 ± 0.035
OSK	19.600 ± 0.073	37.557 ± 0.315	1.214 ± 0.061
OSVtx	20.834 ± 0.075	36.994 ± 0.308	1.410 ± 0.067
OSc	5.025 ± 0.040	34.062 ± 0.620	0.511 ± 0.040
OScomb	40.154 ± 0.090	35.123 ± 0.211	3.555 ± 0.101
SSK	68.190 ± 0.177	39.667 ± 0.507	2.912 ± 0.286
SSπ	83.486 ± 0.068	42.561 ± 0.145	1.848 ± 0.072
SSp	37.767 ± 0.089	43.645 ± 0.221	0.610 ± 0.042
SScomb	87.590 ± 0.061	41.787 ± 0.142	2.364 ± 0.081

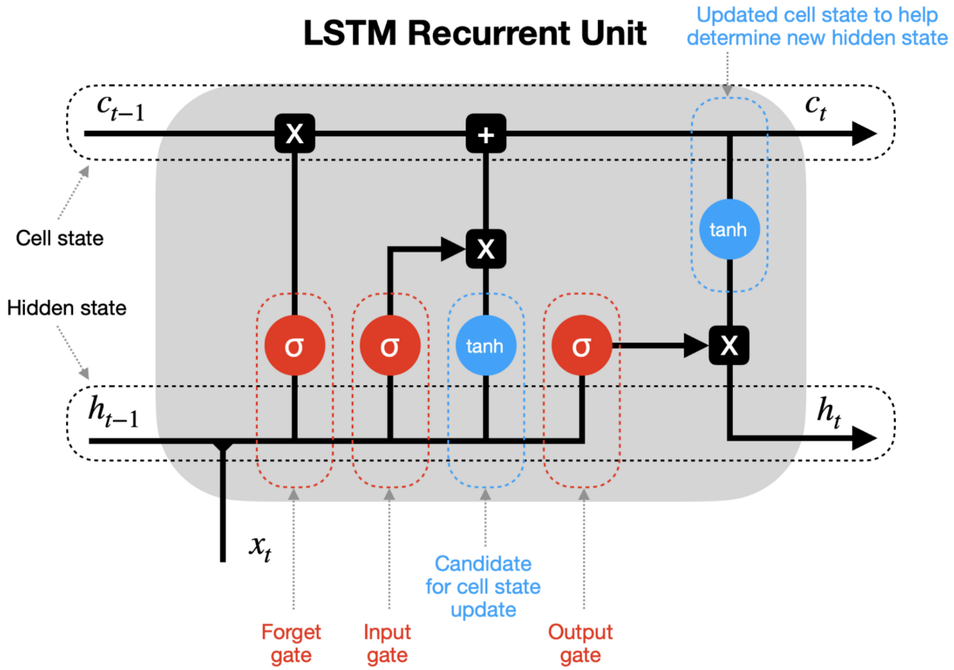
https://cds.cern.ch/record/2668185?ln=en

B flavour tagging

- ‘Inclusive tagger’: add whole event information to make tagging decision

- Order tracks by p_T, use Long-Short Term Memory (LSTM) as a Recurrent NN
  (Idea: rank by ‘importance’ or ‘hadronic closeness’)

→ Also helps to deal with varying #tracks per event

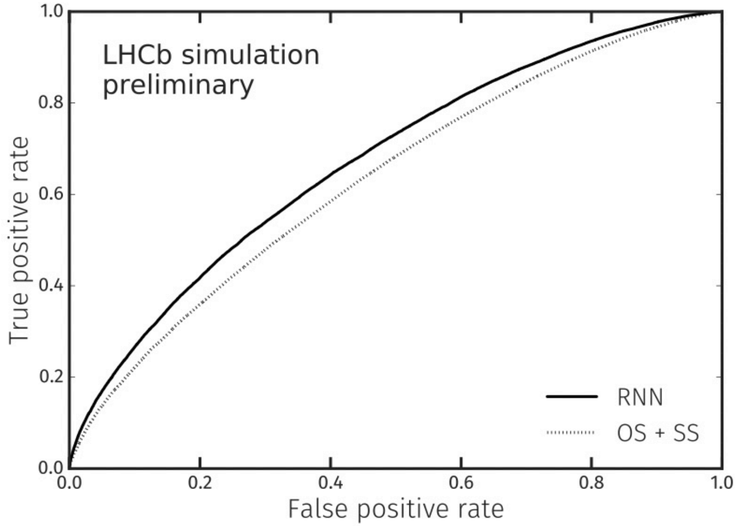


B flavour tagging

- ‘Inclusive tagger’: add whole event information to make tagging decision

- Order tracks by p_T, use Long-Short Term Memory (LSTM) as a Recurrent NN
  (Idea: rank by ‘importance’ or ‘hadronic closeness’)

→ Also helps to deal with varying #tracks per event



(attempt in 2019 calibrated on $B^+ \to J/\psi K^+$)

	Tagger	ϵ	ω	$\epsilon\langle D^2\rangle = \epsilon\,(1-2\omega)^2$
MC	**Classical Combination**	$(81.311 \pm 0.106)\%$	$(37.164 \pm 0.029(\text{stat}) \pm 0.133(\text{cal}))\%$	$(5.359 \pm 0.026(\text{stat}) \pm 0.111(\text{cal}))\%$
	Incl. tagger	$(100.000 \pm 0.000)\%$	$(34.146 \pm 0.031(\text{stat}) \pm 0.112(\text{cal}))\%$	$(10.054 \pm 0.039(\text{stat}) \pm 0.142(\text{cal}))\%$
DATA	**Classical Combination**	$(81.584 \pm 0.049)\%$	$(38.506 \pm 0.011(\text{stat}) \pm 0.058(\text{cal}))\%$	$(4.311 \pm 0.008(\text{stat}) \pm 0.044(\text{cal}))\%$
	Incl. tagger	$(98.223 \pm 0.017)\%$	$(36.091 \pm 0.010(\text{stat}) \pm 0.052(\text{cal}))\%$	$(7.601 \pm 0.011(\text{stat}) \pm 0.057(\text{cal}))\%$

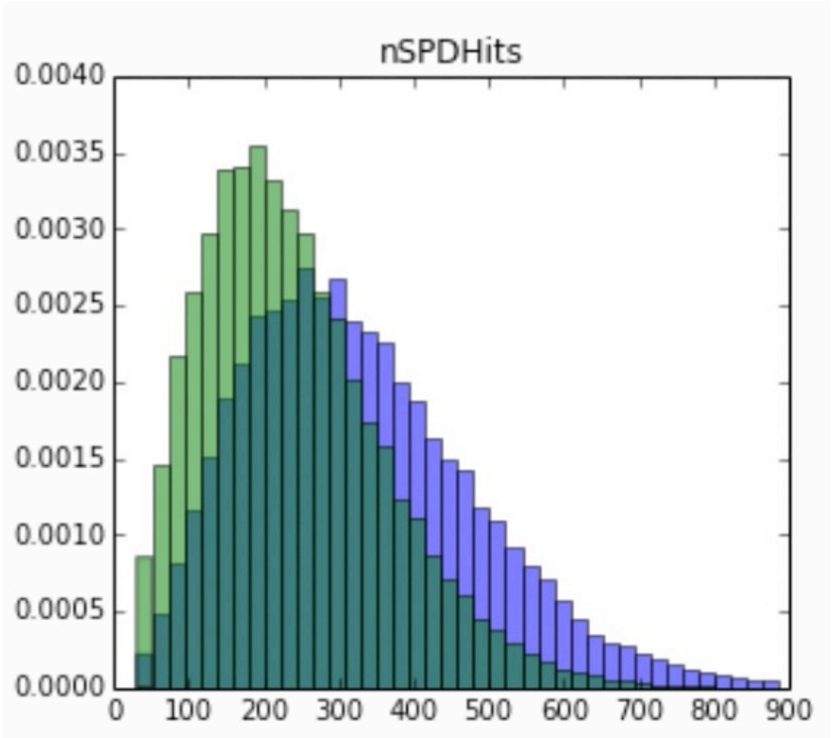
https://cds.cern.ch/record/2668185/files/PoS(LHCP2018)230.pdf

Reweighing distributions

- What if simulation and (background-subtracted) don't agree?

- or: what if your calibration sample has different distributions than your signal? (LHCb: 'PID Calib')

→ traditional 'binned reweighing', thoughts?



https://arogozhnikov.github.io/2015/10/09/gradient-boosted-reweighter.html
https://indico.cern.ch/event/397113/papers/1837841/files/4049-proceedings_reweighting.pdf

Reweighing distributions

- What if simulation and (background-subtracted) don't agree?

- or: what if your calibration sample has different distributions than your signal? (LHCb: 'PID Calib')

→ traditional 'binned reweighing', thoughts?

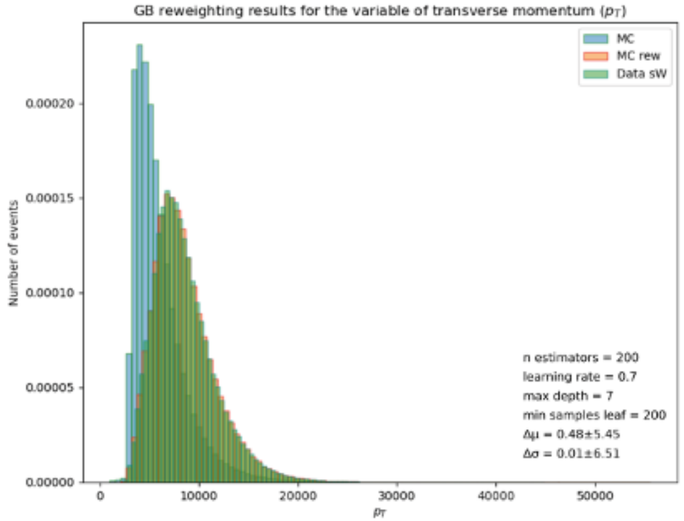
- limited to 'rectangular cuts'
- challenging to optimize binning (empty bins?)

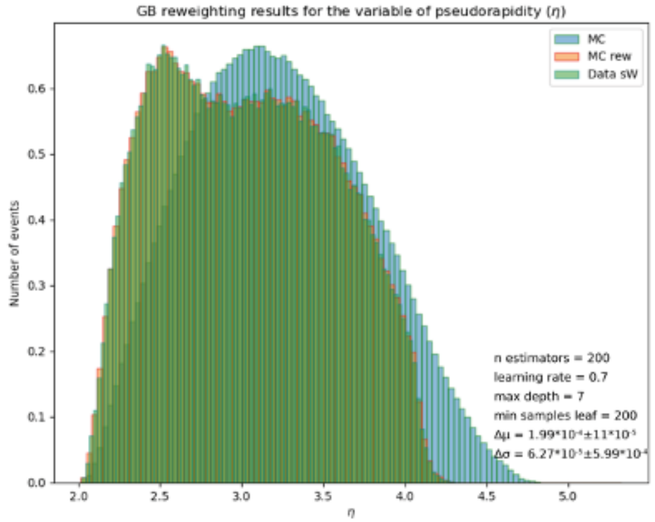
→ use decision trees to split up phase space:

- binning optimised by tree splits
- easy to use:

```
from hep_ml.reweight import GBReweighter
gb = GBReweighter()
gb.fit(mc_data, real_data, target_weight=real_data_sweights)
gb.predict_weights(mc_other_channel)
```

Example from Xic production analysis:
3D gradient boosted reweighing of MC





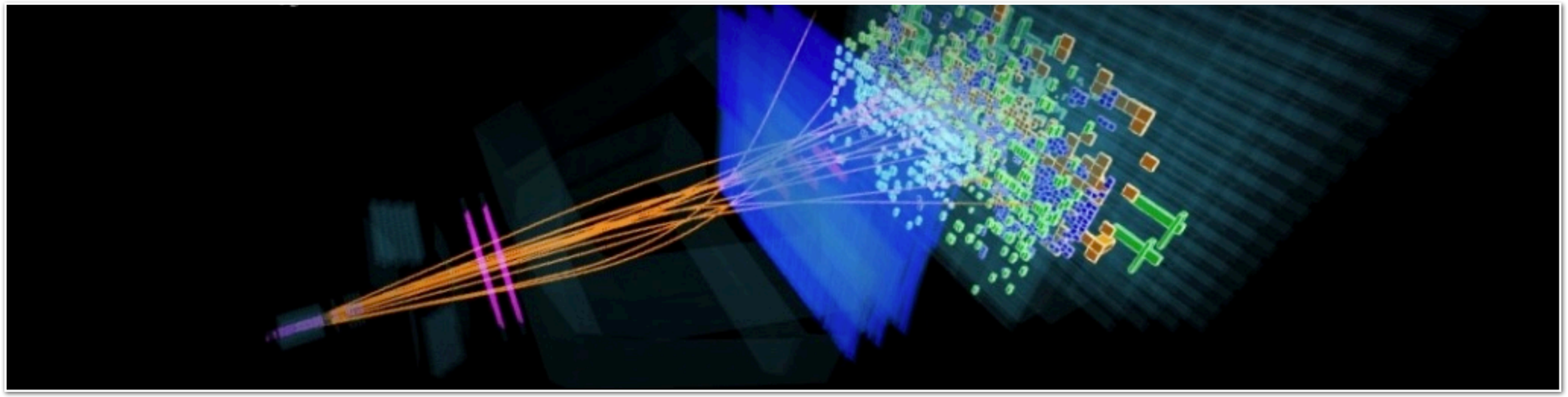
https://arogozhnikov.github.io/2015/10/09/gradient-boosted-reweighter.html
https://indico.cern.ch/event/397113/papers/1837841/files/4049-proceedings_reweighting.pdf

Machine Learning in LHCb

Unsupervised learning

ML in LHCb Simulation

- Simulation generation process slow
  → Geant4 particle-material interaction takes ~1 minute per event
- Need O(10^7) simulated events for efficiency calculations, systematic checks, BDT training, …
- Many channels, many analyses, many backgrounds —> production queue often full



ML in LHCb Simulation

- Simulation generation process slow
  → Geant4 particle-material interaction takes ~1 minute per event

- Need O(10^7) simulated events for efficiency calculations, systematic checks, BDT training, …

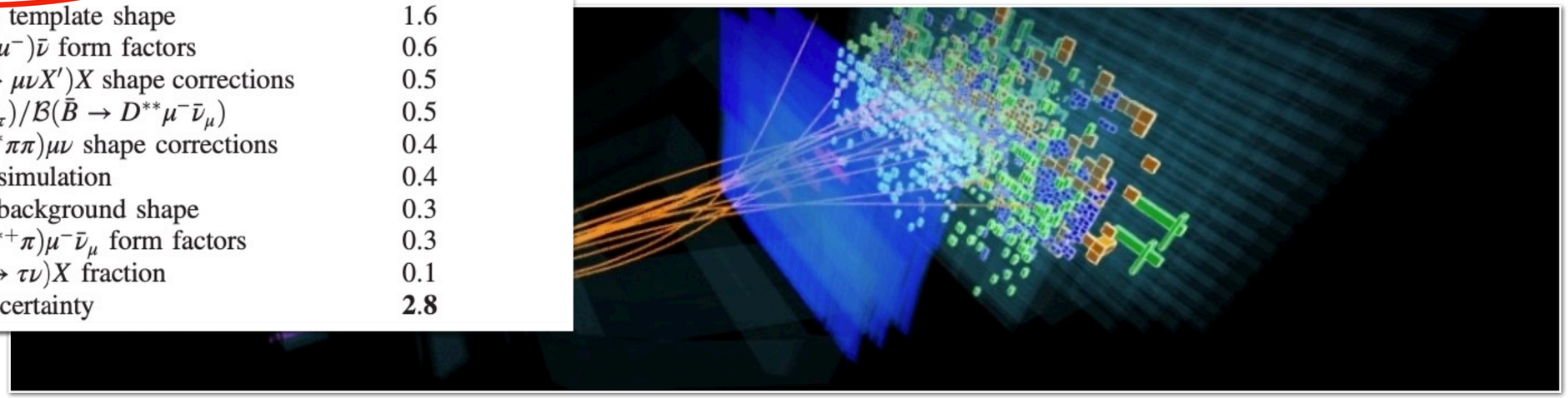
- Many channels, many analyses, many backgrounds —> production queue often full
  → ‘MC stats’ largest systematic error for some important measurements

TABLE I. Systematic uncertainties in the extraction of $\mathcal{R}(D^*)$.

Model uncertainties	Absolute size ($\times 10^{-2}$)
Simulated sample size	2.0
Misidentified μ template shape	1.6
$\bar{B}^0 \to D^{*+}(\tau^-/\mu^-)\bar{\nu}$ form factors	0.6
$\bar{B} \to D^{*+}H_c(\to \mu\nu X')X$ shape corrections	0.5
$\mathcal{B}(\bar{B} \to D^{**}\tau^-\bar{\nu}_\tau)/\mathcal{B}(\bar{B} \to D^{**}\mu^-\bar{\nu}_\mu)$	0.5
$\bar{B} \to D^{**}(\to D^*\pi\pi)\mu\nu$ shape corrections	0.4
Corrections to simulation	0.4
Combinatorial background shape	0.3
$\bar{B} \to D^{**}(\to D^{*+}\pi)\mu^-\bar{\nu}_\mu$ form factors	0.3
$\bar{B} \to D^{*+}(D_s \to \tau\nu)X$ fraction	0.1
Total model uncertainty	**2.8**

$$\mathcal{R}(D^*) = 0.336 \pm 0.027(\text{stat}) \pm 0.030(\text{syst}).$$

https://journals.aps.org/prl/pdf/10.1103/PhysRevLett.115.111803



ML in LHCb Simulation

- Ideas: RICH-less sim, ReDecay, RapidSim, …

- Speed up simulation by *parameterising* detector response

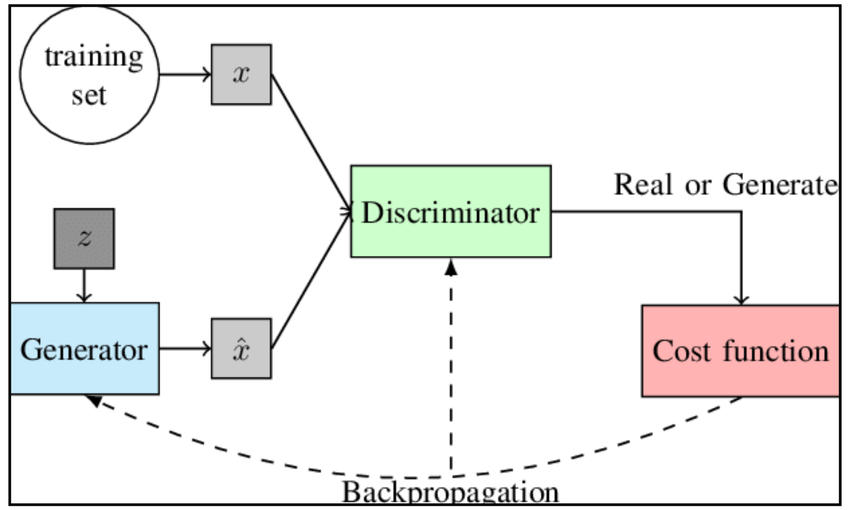
- ‘Delphes’ ultra-fast simulation: collective to gather all fast approaches & implement in Geant

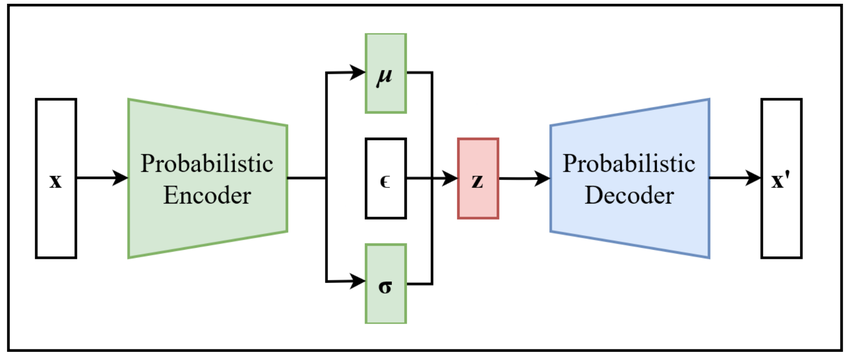
  → from particle inputs (momenta, etc.) to full detector response, or efficiencies

https://twiki.cern.ch/twiki/pub/LHCb/DelphesFastSimulation/LHCb-INT-2019-014_v0r2.pdf

ML in LHCb Simulation

- Ideas: RICH-less sim, ReDecay, RapidSim, …
- Speed up simulation by *parameterising* detector response
- 'Delphes' ultra-fast simulation: collective to gather all fast approaches & implement in Geant
  → from particle inputs (momenta, etc.) to full detector response, or efficiencies
- 'Images' from RICH rings / CALOs: generative models
  → Generative Adversarial Networks
  → Variational Auto-Encoders
- Challenges:
  → Convergence not trivial (& mode collapse)
  → Is efficiency representative in all cases?





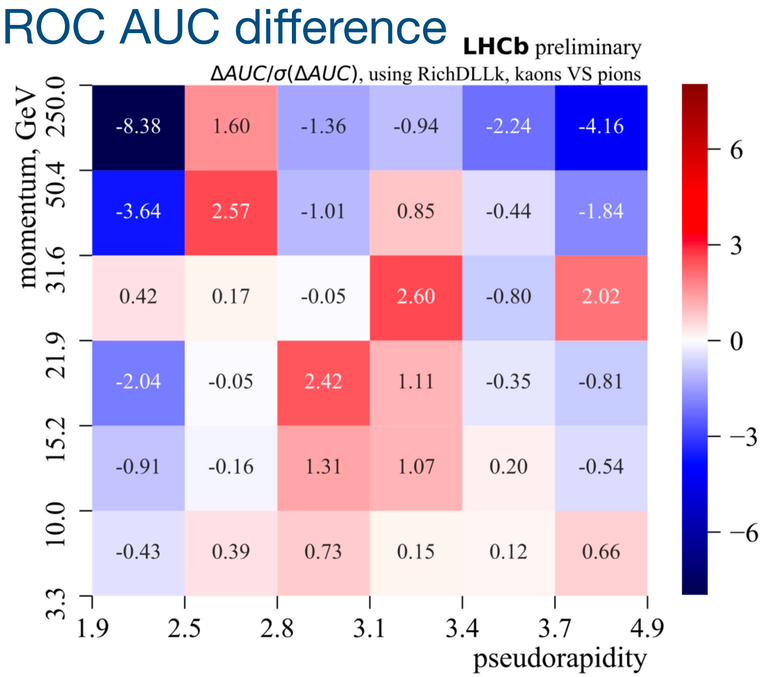
https://twiki.cern.ch/twiki/pub/LHCb/DelphesFastSimulation/LHCb-INT-2019-014_v0r2.pdf

GAN in action:

https://colab.research.google.com/drive/1eFA1VqIwhnZBdDJCJ27PN6V2-wid-eSU

RICH GAN Simulation

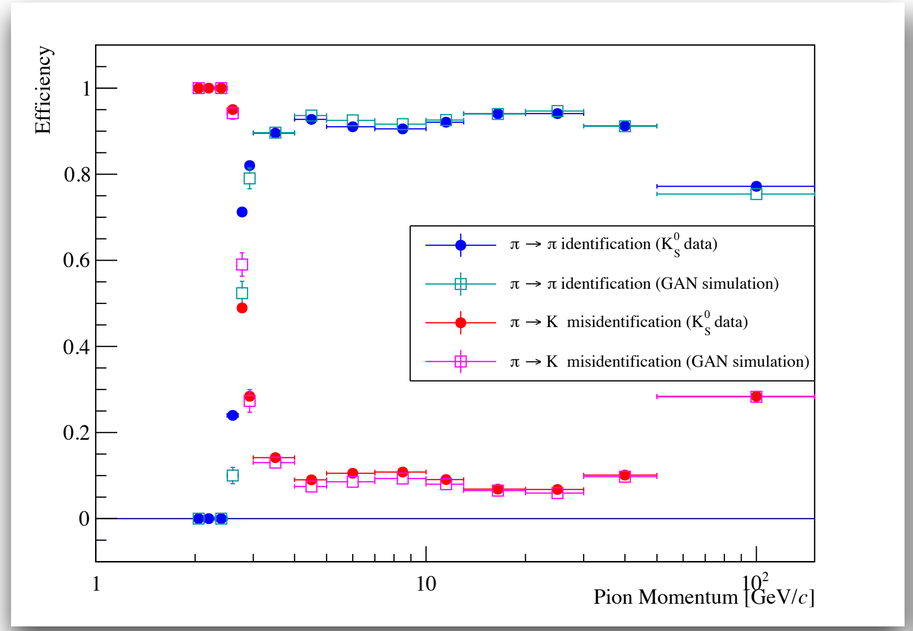
- 128 neurons x 10 hidden layers, ReLU activation.
- latent space: 64 inputs, gaussian noise
- Trained on GPU Tesla K80, takes ~1 day
- Simulate likelihoods of 5 hypotheses

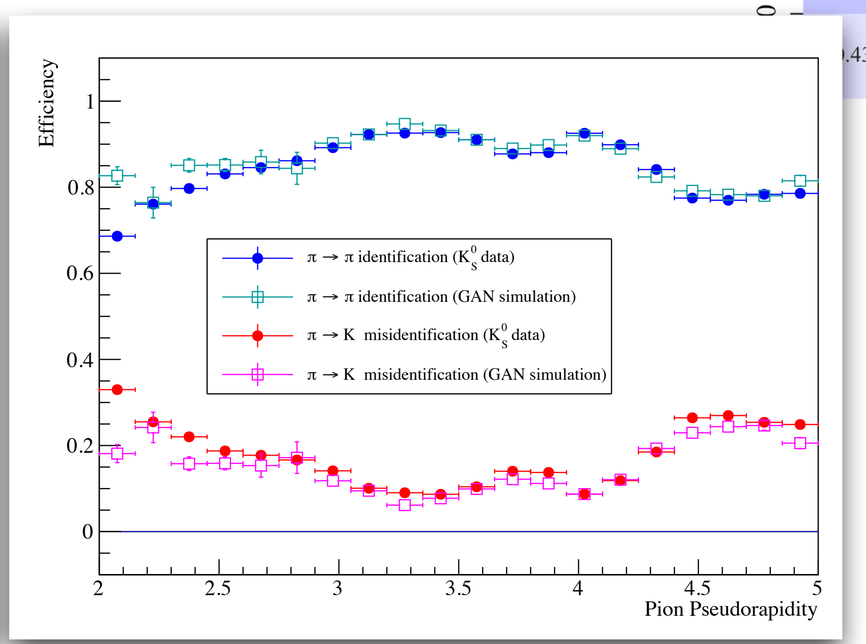


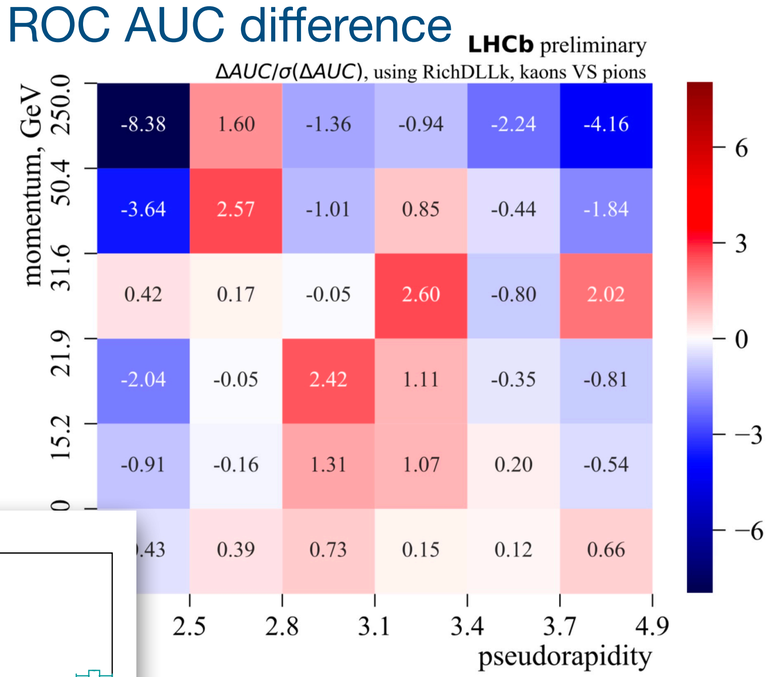
https://twiki.cern.ch/twiki/pub/LHCb/DelphesFastSimulation/LHCb-INT-2019-014_v0r2.pdf

RICH GAN Simulation

- 128 neurons x 10 hidden layers, ReLU activation.
- latent space: 64 inputs, gaussian noise
- Trained on GPU Tesla K80, takes ~1 day
- Simulate likelihoods of 5 hypotheses





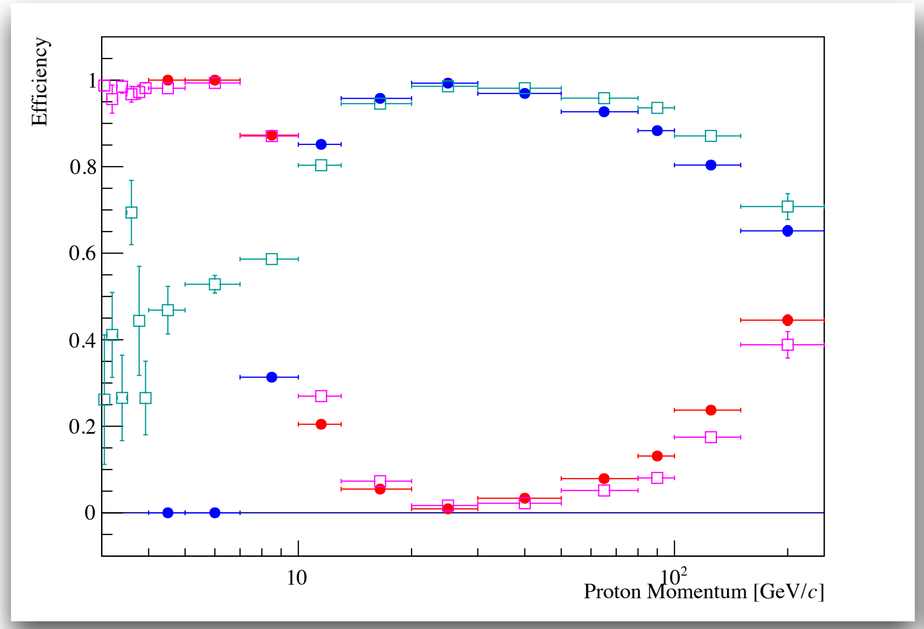


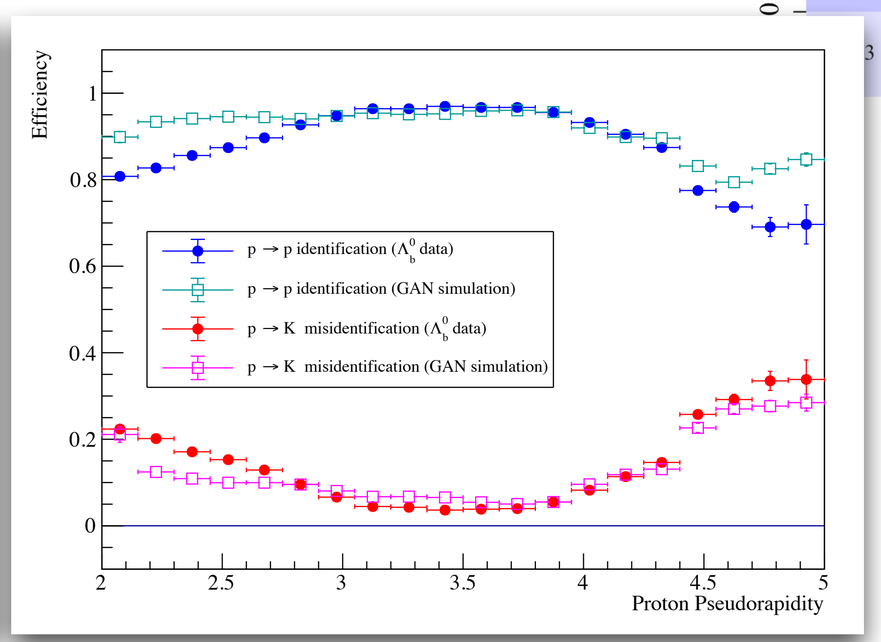
→ not bad!

https://twiki.cern.ch/twiki/pub/LHCb/DelphesFastSimulation/LHCb-INT-2019-014_v0r2.pdf

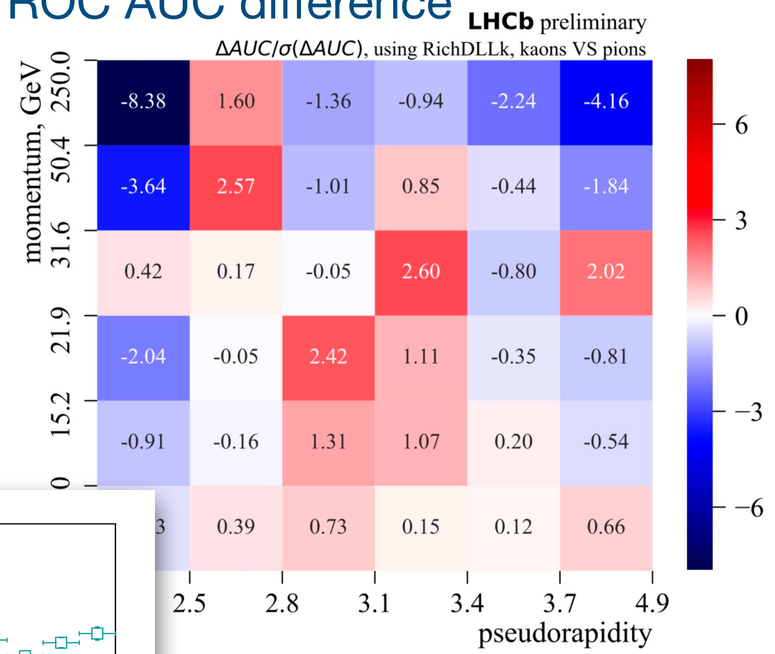
RICH GAN Simulation

- 128 neurons x 10 hidden layers, ReLU activation.
- latent space: 64 inputs, gaussian noise
- Trained on GPU Tesla K80, takes ~1 day
- Simulate likelihoods of 5 hypotheses





ROC AUC difference



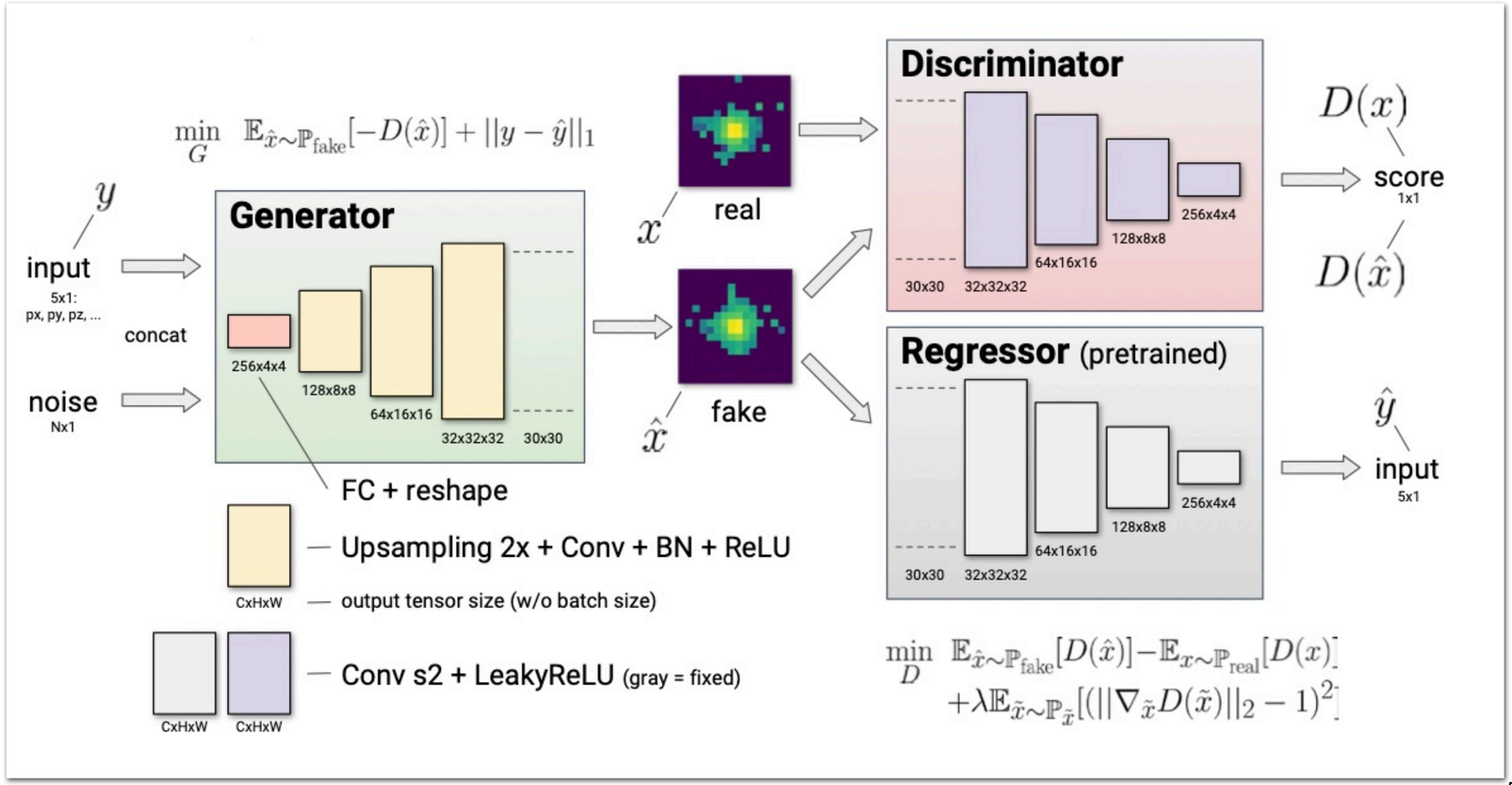
→ but not perfect…

https://twiki.cern.ch/twiki/pub/LHCb/DelphesFastSimulation/LHCb-INT-2019-014_v0r2.pdf

ECAL simulation

- Calo simulation can take ~50% of CPU time
- Often need for full detector ‘picture’ due to complex physics use-case: Bremsstrahlung, triggers, …
- Generate 30x30 image from particle $\overrightarrow{p}$, x, y + ‘noise’
- Regressor: estimate inputs from generated image, add to Discriminator gradient

https://doi.org/10.1051/epjconf/201921402034



ECAL simulation

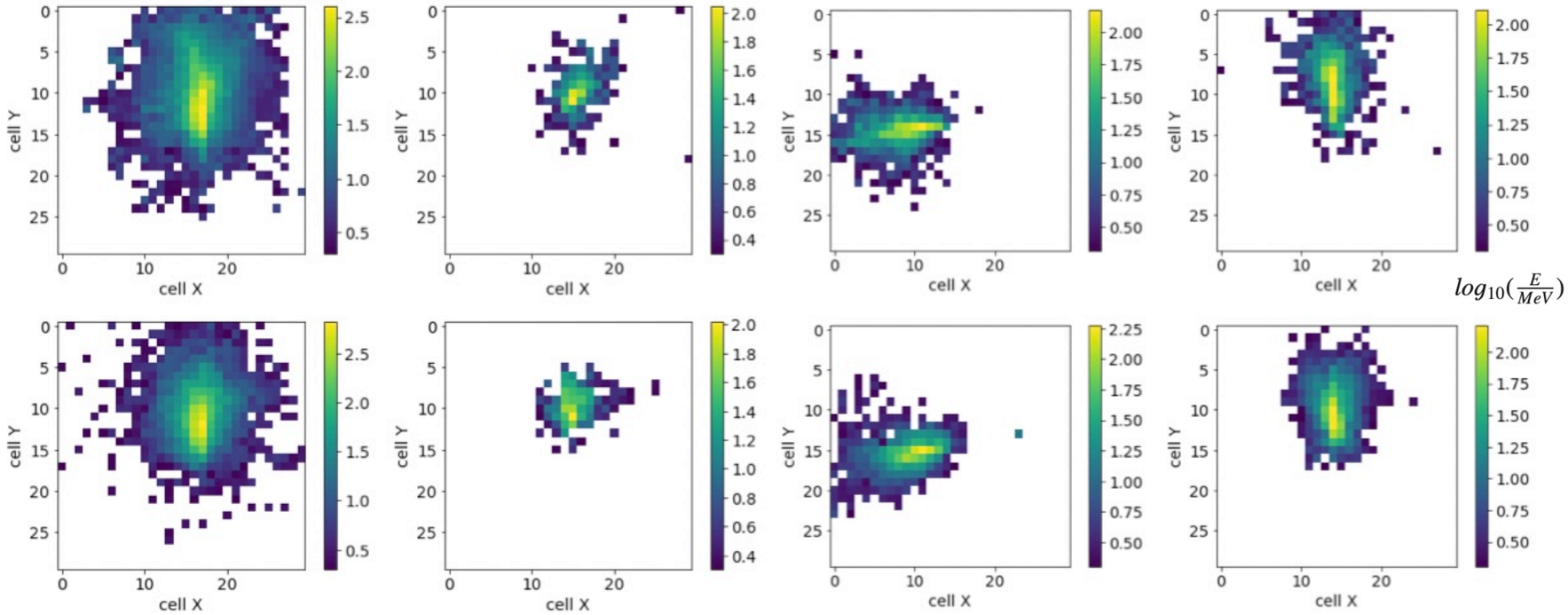
- Training time ~70h on NVidia Tesla K80

  (sampling time on GPU: 0.07 ms/sample, vs 4.9 ms/sample on CPU)

https://doi.org/10.1051/epjconf/201921402034

Geant →

GAN →



(a) $E_0 = 63.7$ GeV

(b) $E_0 = 6.5$ GeV

(c) $E_0 = 15.6$ GeV

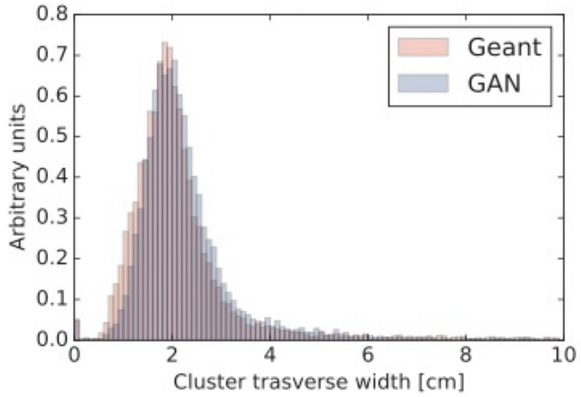
(d) $E_0 = 15.9$ GeV

ECAL simulation

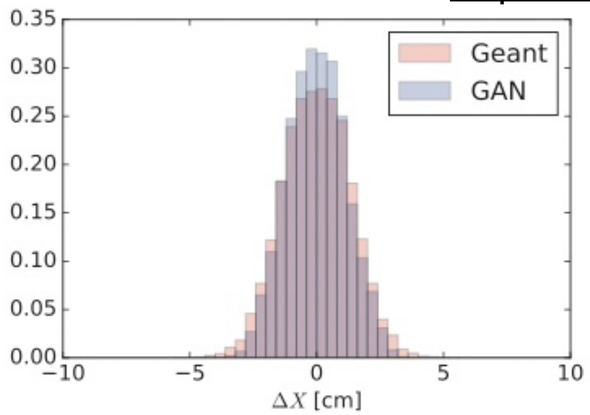
- Training time ~70h on NVidia Tesla K80

  (sampling time on GPU: 0.07 ms/sample, vs 4.9 ms/sample on CPU)

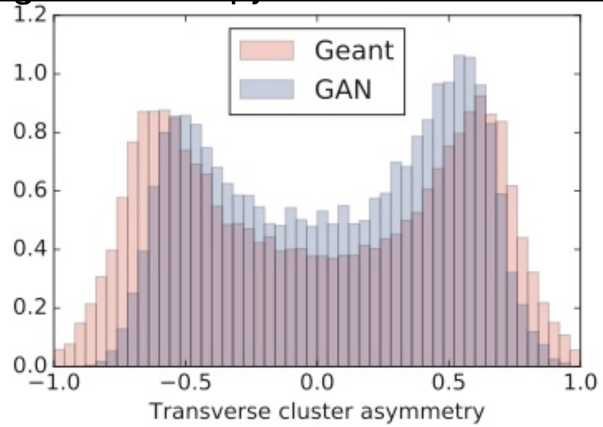
https://doi.org/10.1051/epjconf/201921402034



(a) The transverse width of real and generated clusters



(c) ΔX between cluster center of mass and the true particle coordinate



(e) The transverse asymmetry of real and generated clusters

→ Promising! Can it scale to full CALO sizes? Backgrounds?

→ Interested? give it a go yourself at the CALO Challenge:

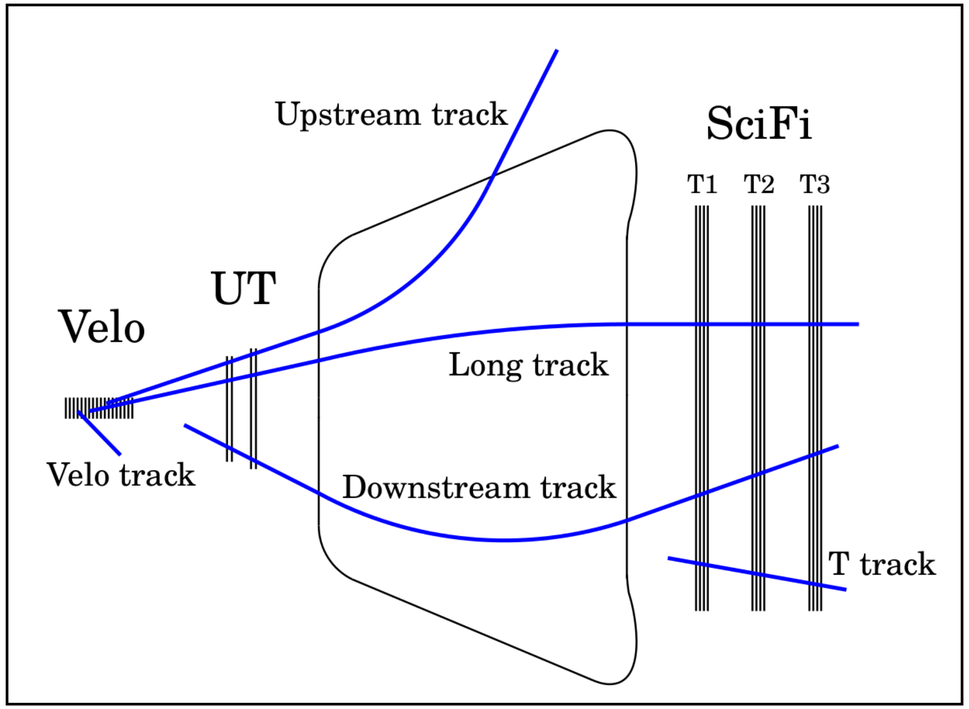
https://calochallenge.github.io/homepage/

Track reconstruction

- Two phases:

  1) pattern recognition (‘which hits belong together?’)

  2) track fit (‘what are the best track parameters given those hits?’)



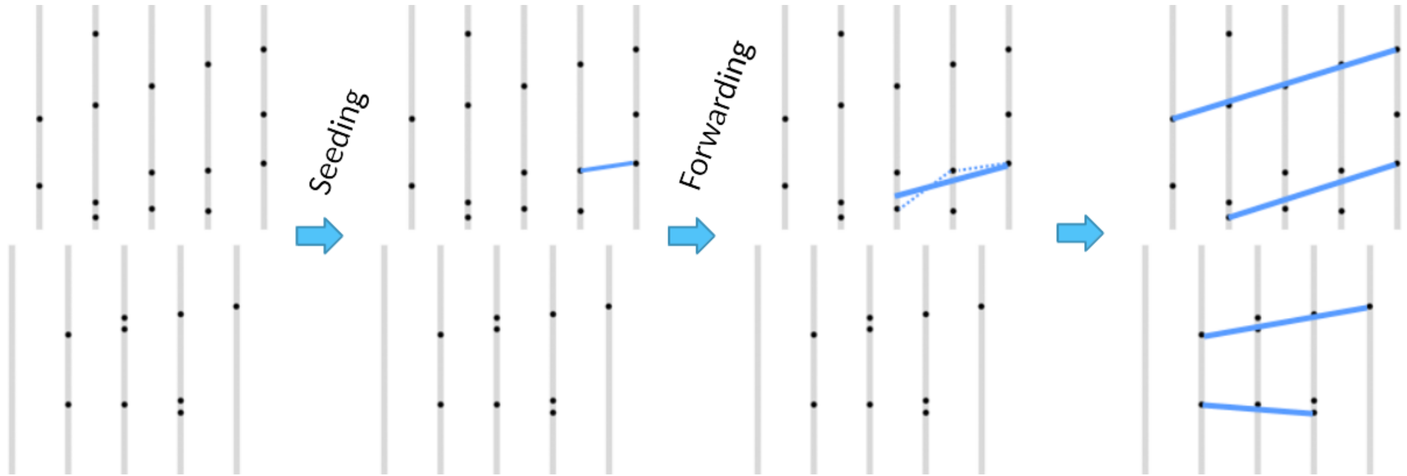
Track reconstruction

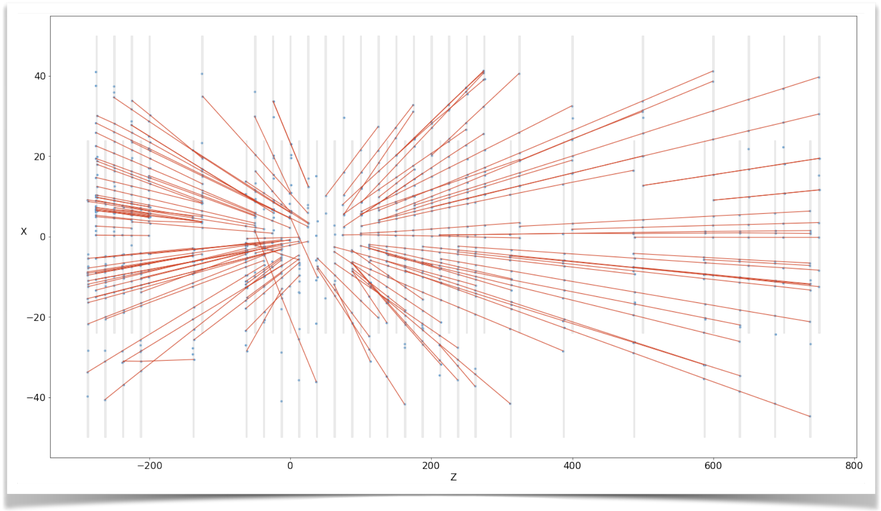
- Two phases:

  1) pattern recognition (‘which hits belong together?’)

  2) track fit (‘what are the best track parameters given those hits?’)

VELO: ‘search-by-triplet’





→ Play with it yourself at https://cds.cern.ch/record/2809707
https://github.com/dcampora/velopix_tracking

Track reconstruction

- Two phases:

  1) pattern recognition (‘which hits belong together?’)

  2) track fit (‘what are the best track parameters given those hits?’)

Can you think of alternatives?

Track reconstruction

- Two phases:

  1) pattern recognition (‘which hits belong together?’)

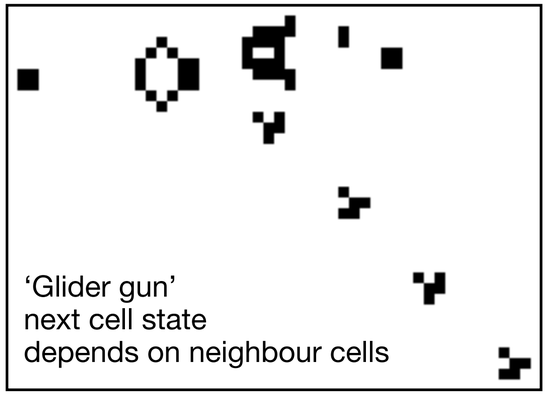
  2) track fit (‘what are the best track parameters given those hits?’)

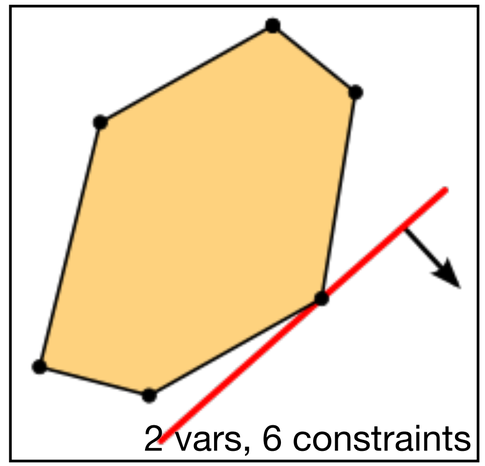
---

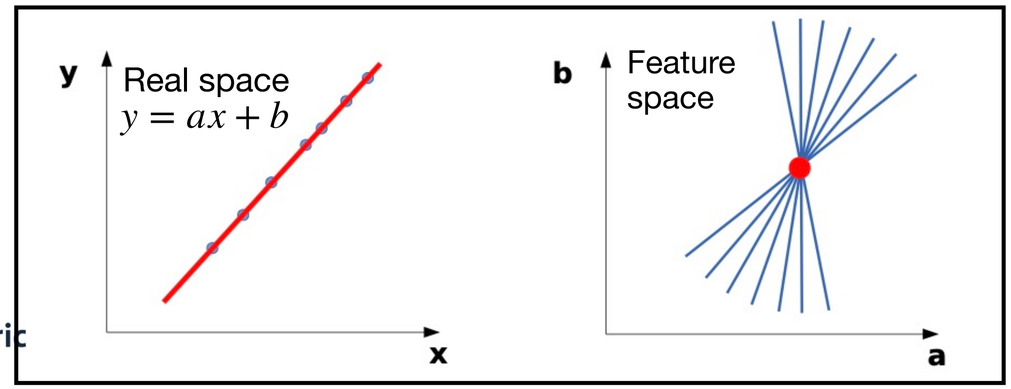
https://scripties.uba.uva.nl/search?id=652350

Can you think of alternatives?

- Cellular automata
- Global optimisation with LP
- Hough transform







Track reconstruction

- Two phases:

  1) pattern recognition (‘which hits belong together?’)

  2) track fit (‘what are the best track parameters given those hits?’)

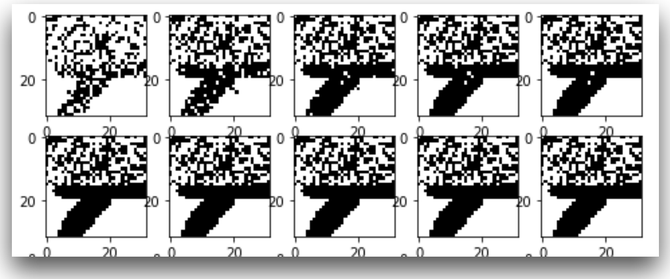
Can you think of alternatives?

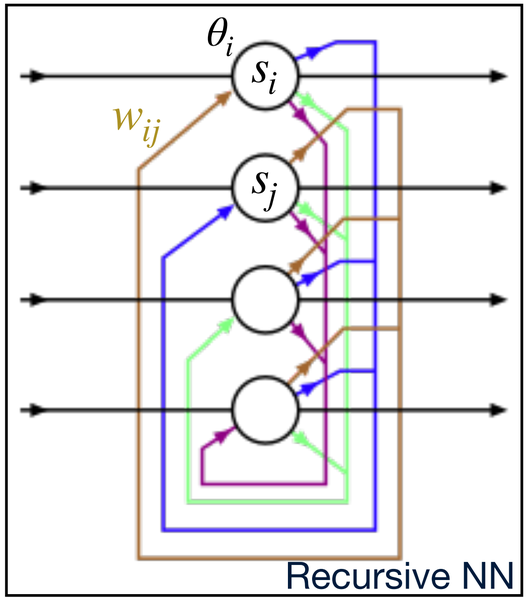
- Cellular automata
- Global optimisation with LP
- Hough transform
- Clustering
- Hopfield networks

$$s_i \leftarrow \begin{cases} +1 & \text{if } \sum_j w_{ij} s_j \geq \theta_i, \\ -1 & \text{otherwise.} \end{cases}$$

$$E = -\frac{1}{2} \sum_{i,j} w_{ij} s_i s_j \qquad \Delta E \leq 0$$

Learns about connections (w_{ij}) between track segments (s_i)





and can correct noisy or incomplete data!

Search-by-triplet

Particle category	Reco eff.	Clone fraction	Hit purity	Hit eff.
All	98.52	2.14	99.30	96.45
Strange	98.13	1.58	99.48	97.35
From B	99.30	1.16	99.74	98.11
Electrons	97.38	2.74	98.18	97.02
From B electrons	97.00	3.68	98.42	96.68
Overall fake fraction		0.86		

Preliminary hopfield network

Particle category	Reco eff.	Clone fraction	Hit purity	Hit eff.
All	75.0	10.53	99.56	82.93
Long	92.5	17.58	99.51	79.77
Long > 5 GeV	95.0	18.41	99.45	79.97
Long Strange	85.8	10.06	99.48	88.52
Long Strange > 5 GeV	90.4	5.88	99.28	90.43
Overall fake fraction		1.9		

(credits to DKE students @ UM & Daniel Campora)

Track reconstruction

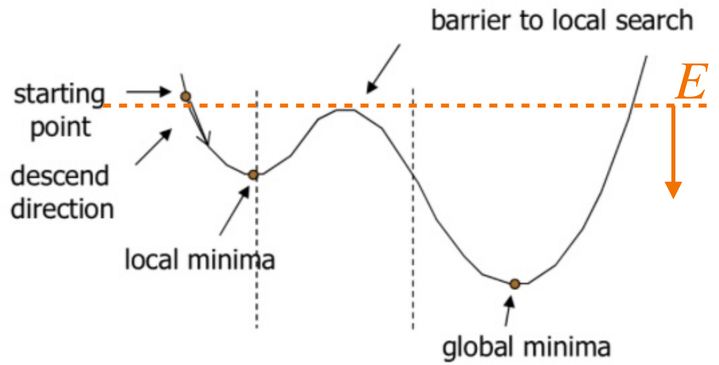
- Two phases:

  1) pattern recognition (‘which hits belong together?’)

  2) track fit (‘what are the best track parameters given those hits?’)

Can you think of alternatives?

- Cellular automata
- Global optimisation with LP
- Hough transform
- Clustering
- Hopfield networks
- (Simulated) Annealing
- Graph neural networks? …?



$$H(\sigma) = -\sum_{\langle i\ j \rangle} J_{ij}\sigma_i\sigma_j$$

- Random step, calculate E
- Accept if ΔE < 0, or
  if ΔE > 0 with $P = e^{-\Delta E/t}$
- Lower t

Track reconstruction

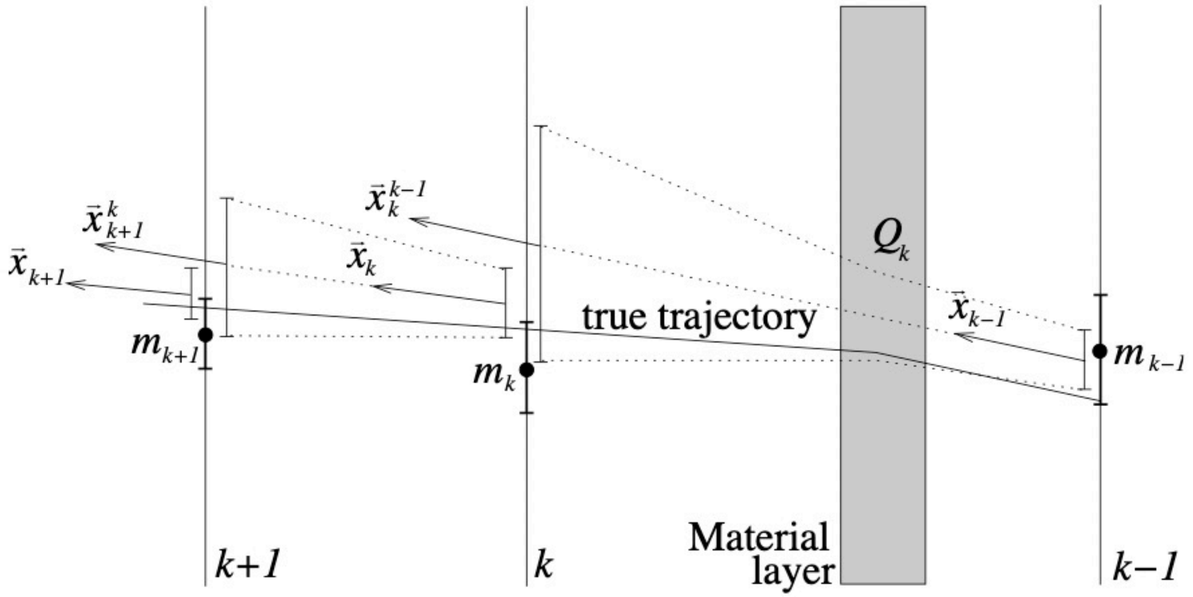
- Two phases:

  1) pattern recognition (‘which hits belong together?’)

  2) track fit (‘what are the best track parameters given those hits?’)

LHCb: Kalman filter

- Measurements m_k, e.g. (x, y) at z_k,
  Track states x_k $(x, y, t_x, t_y, q/p)$ at z_k



https://research.vu.nl/ws/portalfiles/portal/75836522/complete+dissertation.pdf

Track reconstruction

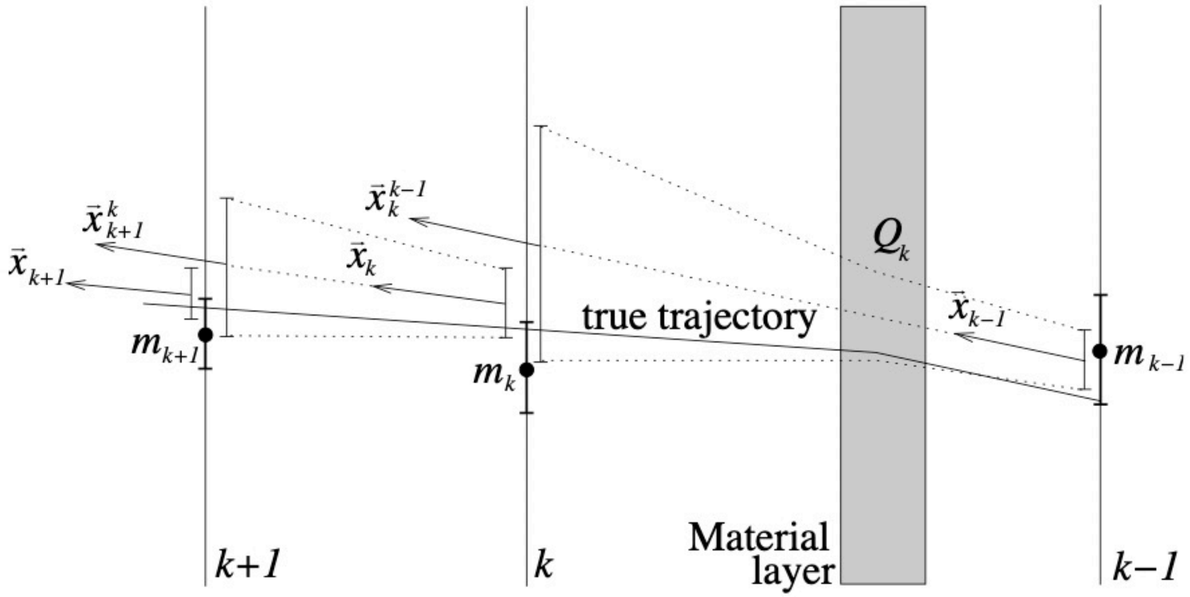
- Two phases:

  1) pattern recognition (‘which hits belong together?’)

  2) track fit (‘what are the best track parameters given those hits?’)

LHCb: Kalman filter

- Measurements m_k, e.g. (x, y) at z_k,
  Track states x_k $(x, y, t_x, t_y, q/p)$ at z_k
- Start with guess from pattern reco.
- Predict state from x_{k-1} to x_k
  → propagate state and covariance matrix
  → Magnetic field, detector material map
- Update prediction with measurement
  → Projection matrix 2x5 dim
  → Filter linear algebra
- After loop: update (‘smooth’) all nodes with full track info



http://cds.cern.ch/record/885750/files/

Track reconstruction

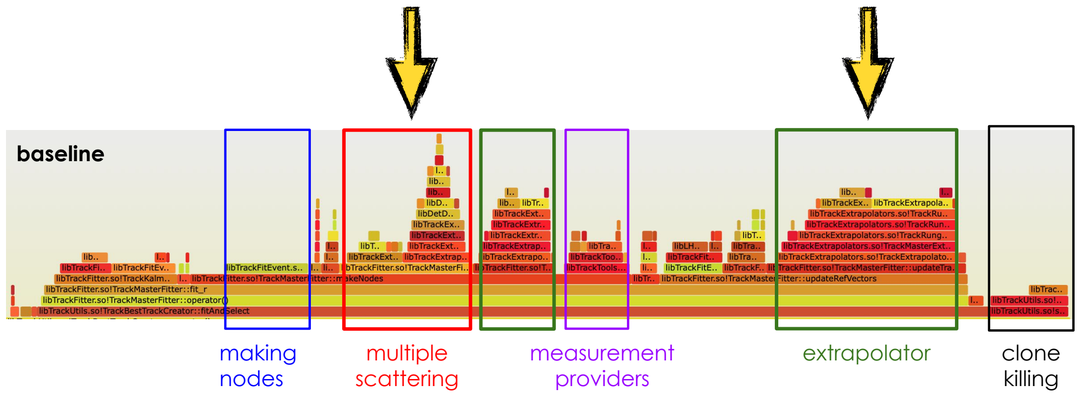
- Two phases:

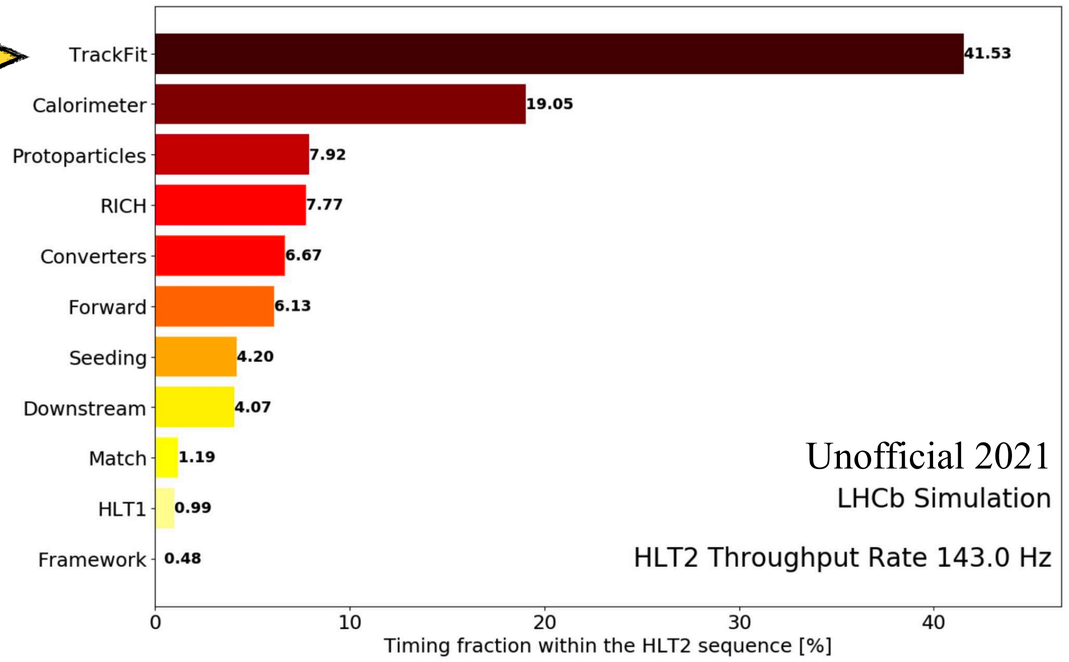
  1) pattern recognition (‘which hits belong together?’)

  2) track fit (‘what are the best track parameters given those hits?’)

LHCb: Kalman filter

- Takes the longest time in the Hlt2 reconstruction sequence
- Can we improve?





(credit to Miriam Martinez & Andrii Usachov)

Track reconstruction

- Two phases:

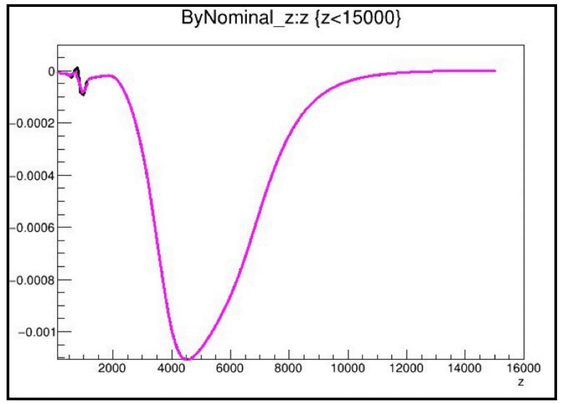
  1) pattern recognition (‘which hits belong together?’)

  2) track fit (‘what are the best track parameters given those hits?’)

- Extrapolation through magnetic field:

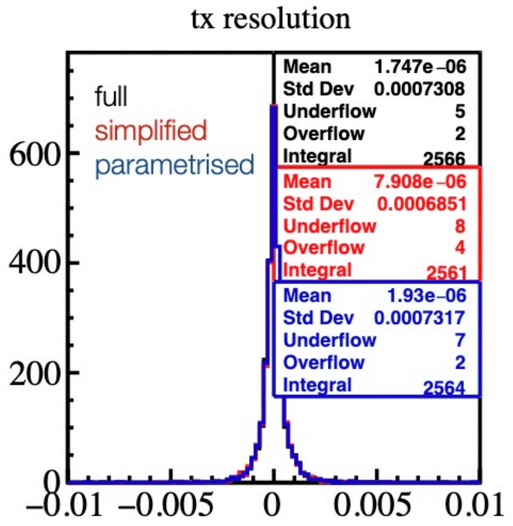
  - Parameterise B-field in grid points as set of coefficients

$$B_x = \alpha_1 + 2\alpha_4 x + \alpha_6 z + \alpha_8 y$$
$$B_y = \alpha_2 + 2\alpha_5 y + \alpha_7 z + \alpha_8 x$$
$$B_z = \alpha_3 - 2\alpha_4 z - 2\alpha_5 z + \alpha_6 x + \alpha_7 y$$

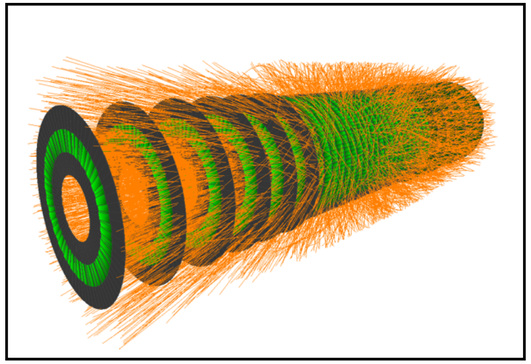


- Parameterised scattering in material:

  - material noise as function of node type lookup tables (4 DoF) * momentum



- TrackML: ML not always the best solution!



https://arxiv.org/pdf/1904.06778.pdf https://www.kaggle.com/c/trackml-particle-identification

Track reconstruction

- Two phases:

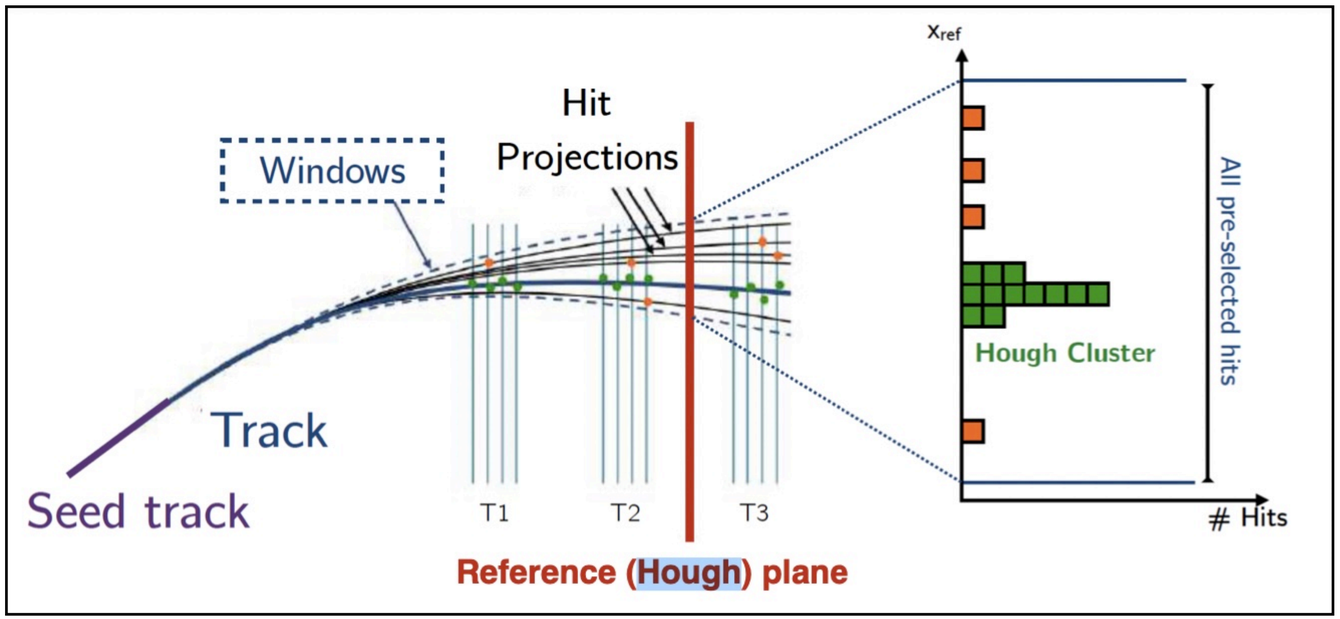
  1) pattern recognition (‘which hits belong together?’)

  2) track fit (‘what are the best track parameters given those hits?’)

- Finding hits after the magnet…
  (aka ‘forward tracking’)

- Vertex finding with kernel methods…
  https://arxiv.org/pdf/2103.04962.pdf

→ for another time.



https://indico.cern.ch/event/658267/contributions/2881131/attachments/1620334/2577923/rquaglia_CDT2018_final.pdf

Machine Learning in LHCb

Exotic menu

LHCb Trigger in run 3

- Data rate: too much to store! (40 MHz x 55 kB = 2.2 TB/s)

  → real-time data filter ('trigger'): L0 + Hlt1 + Hlt2

- Run 1&2 bottleneck: L0 hardware (FPGA) trigger (40 MHz —> 1 MHz)

- Very low-level information used to make tough decisions:

  → energy in ECAL cells: $E_T > 3\ (3.7)$ GeV [450 (150) kHz]

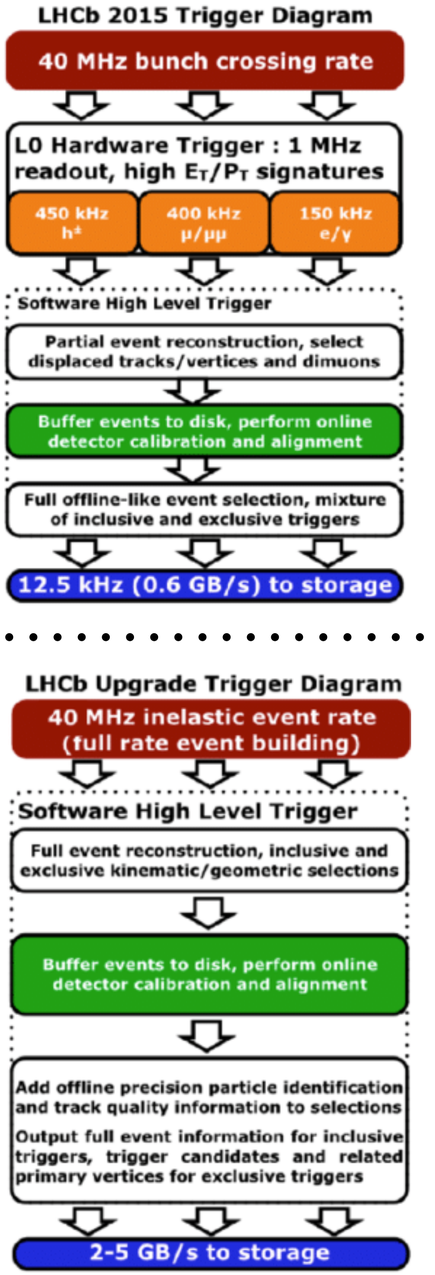
  → muon hits: $p_T > 1.76\ (1.6)$ GeV/c (~20% resolution) [400 kHz]

- Run 2: *directly* take 40 MHz to software: **GPUs** ('Allen' project)

  → Reconstruct tracks, vertices, in parallel scheme

  → Make more informed decisions, greatly increase efficiency

  → Nikhef in a lead role (Daniel Campora, Roel Aaij, Gerhard Raven)



http://dx.doi.org/10.1088/1742-6596/762/1/012046
https://doi.org/10.1007/s41781-020-00039-7

LHCb Trigger in run 3

- Purchased 200 NVidia Ampere RTX A5000 GPU cards
- Hlt1 limited by throughput (bandwidth), not by latency

→ Room for new algorithms!
- Full decoding of ECAL
- GPUs ideal for Machine Learning
- Tensor Cores & Ray-Tracing Cores unused

→ eScience collaboration: develop ML for Allen,
using TensorRT, use-case electron reconstruction/ID
('Accelerating Scientific Discovery')

→ *Ray tracing for RICH reconstruction?*

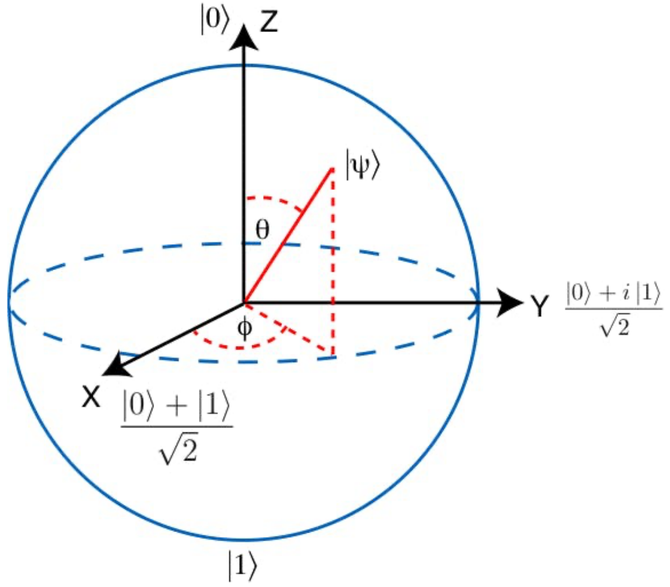
→ *..?*

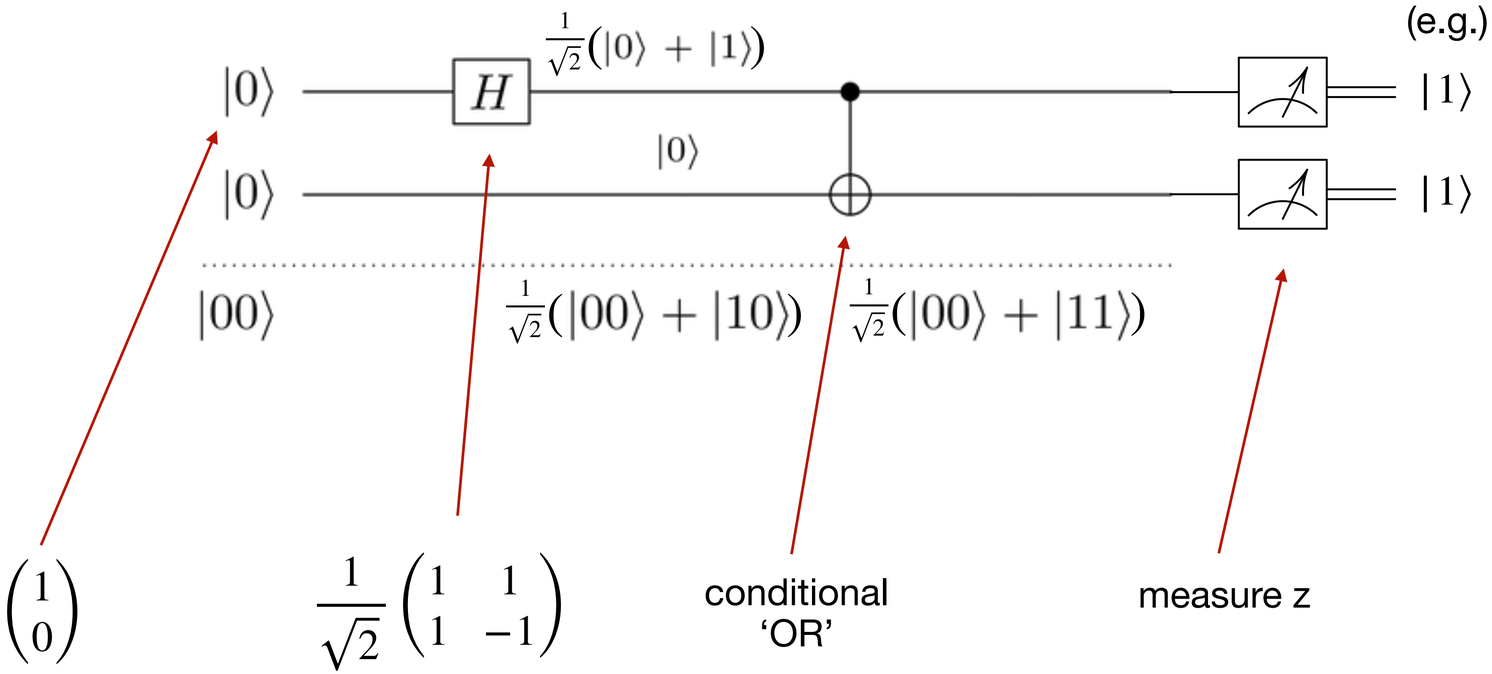
SPECIFICATIONS

GPU memory	**24 GB GDDR6**
Memory interface	**384-bit**
Memory bandwidth	**768 GB/s**
Error-correcting code (ECC)	**Yes**
NVIDIA Ampere architecture-based CUDA Cores	**8,192**
NVIDIA third-generation Tensor Cores	**256**
NVIDIA second-generation RT Cores	**64**

Quantum ML

- bit ⟶ qubit, not discrete [0,1] but spin-1/2 state with continuous values & phases
  ⟶ encode more information per ‘bit’
  ⟶ superposition of 2^N states
- New computational operations: unitary transformations (‘gates’)
  ⟶ superposition of states, entanglement





Want to play around with some real hardware?
https://quantum-computing.ibm.com

Quantum ML

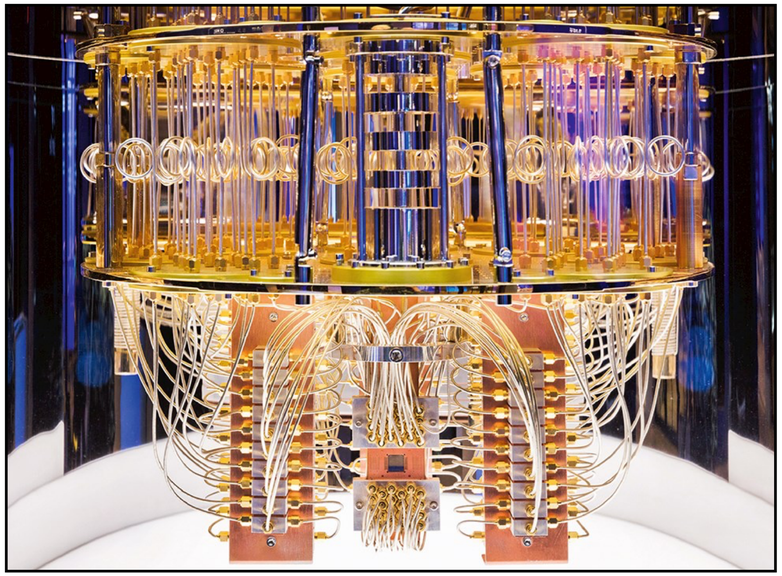
- Idea: could provide (exponential) speedup in some scenarios.
- Status: ~50 qubits, NISQ era. Simulate up to ~30 qubits.

Some algorithms with (future) potential:

- Fourier transform / Shor's algorithm (prime factorisation)
- Grover's algorithm (unstructured search)
- HHL (linear systems of equations)
- ~~Simulated~~ quantum annealing of (Ising-like) Hamiltonian (D-wave)

Dream of qML: speedup of training time

→ For now: Investigate what we can do.
Design algorithms that scale favourably.



Quantum ML

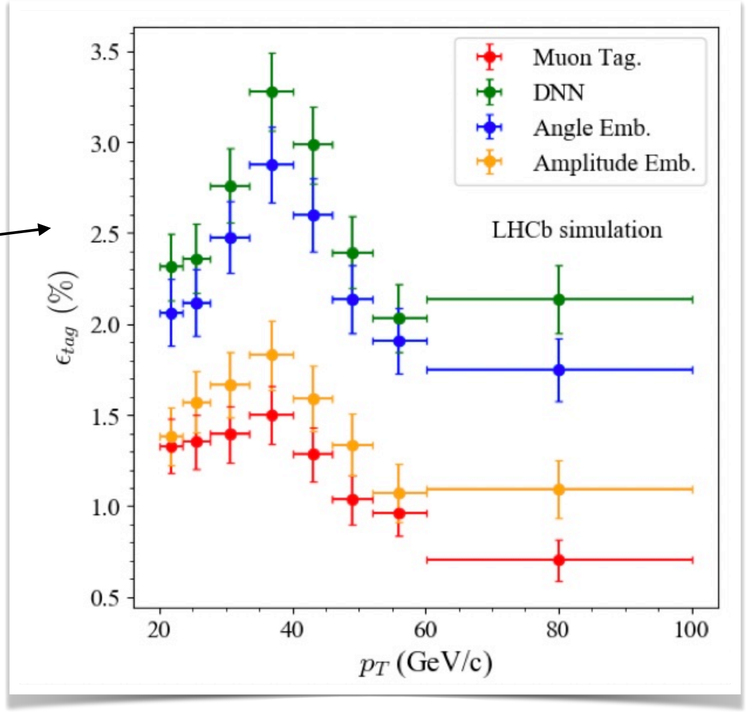
- Gaining popularity at CERN: OpenLab, QTI
- Locally: collaboration between Nikhef, Maastricht, SURF & IBM

- Topics: track reconstruction, b-flavour tagging

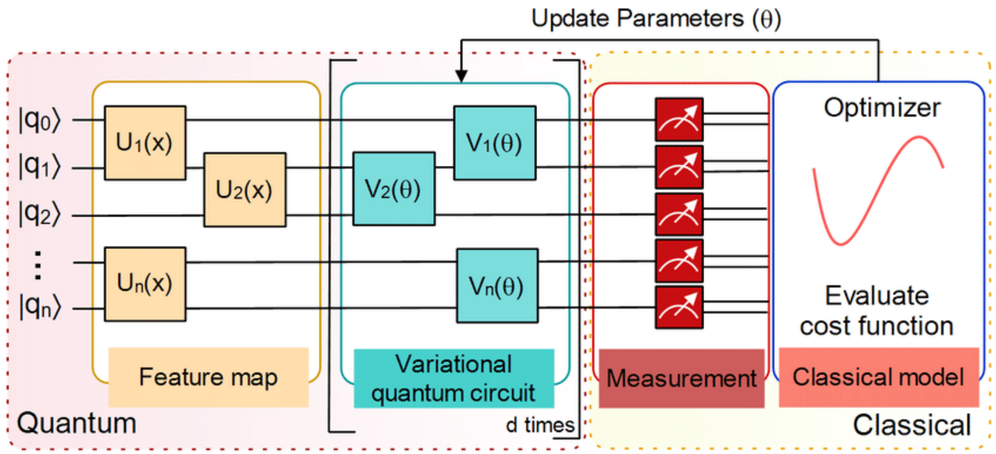
  ⟶ Learn classical parameters θ (phases of qubit rotations)

  ⟶ Performance in restricted scenario's comparable to classical

  ⟶ Challenges: data embedding, iterations, noise simulation, …



https://arxiv.org/abs/1804.11326
https://arxiv.org/abs/2202.13943



Tutorial this afternoon I

⟶ Investigate the possibility for a New Physics anomaly detection trigger line for Hlt1 / Allen

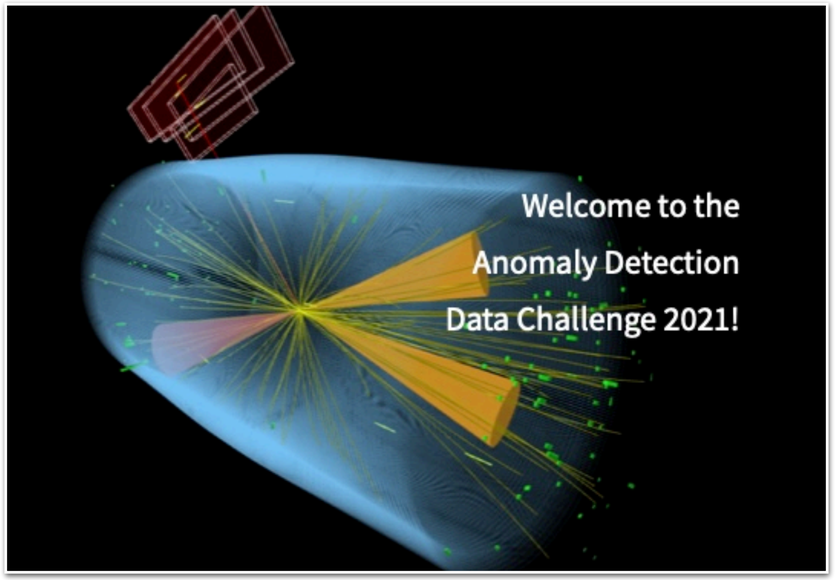
Should have:
- high rejection rate for SM
- high efficiency for NP
- fast inference evaluation timing

⟶ Use datasets of Standard Model, and four ‘anomalous’ sets:
A->4 leptons, leptoquarks -> b tau, h0 -> tau tau, and h+ -> tau nu

⟶ Also a ‘black box’ dataset: submit for competition!

https://mpp-hep.github.io/ADC2021/

https://www.nature.com/articles/s41597-022-01187-8

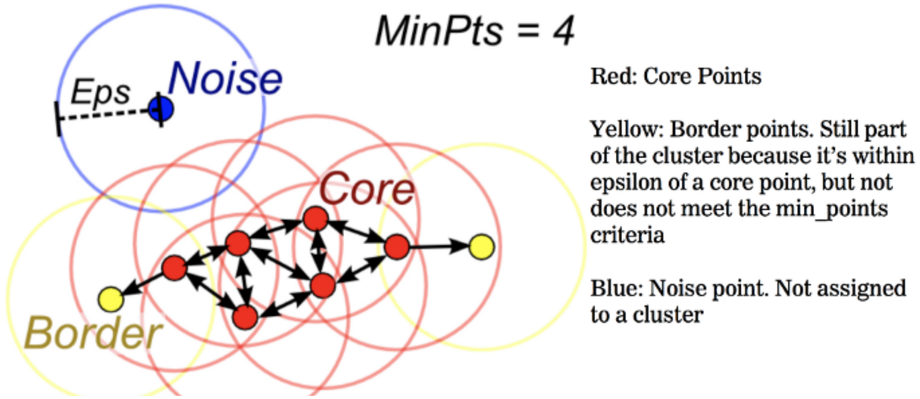


Anomaly detection

Clustering algorithms

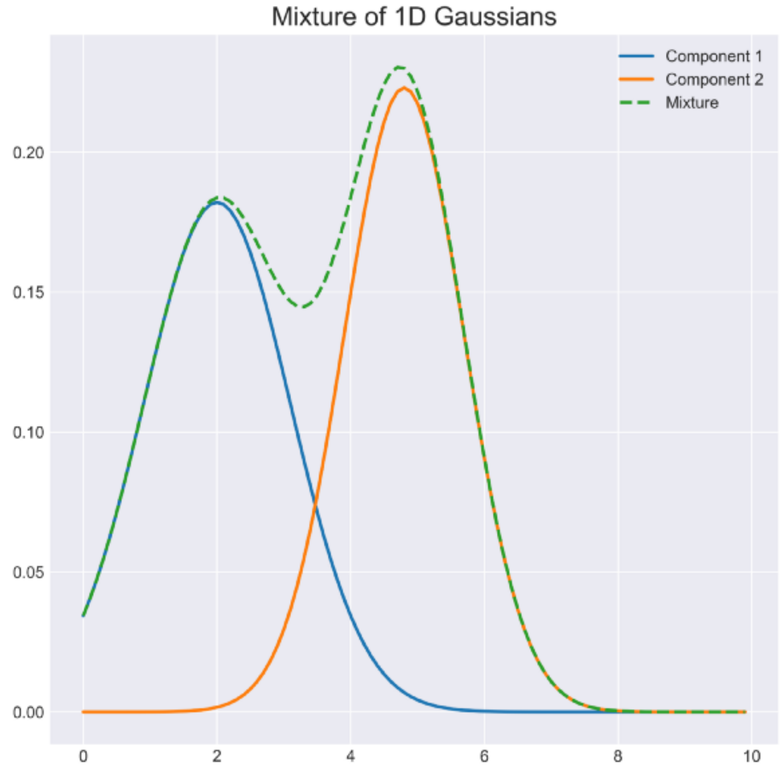
→ DBSCAN

- Learn cluster centers from SM data
- New events: estimate distance to clusters



→ Variational Bayesian Gaussian Mixture

- Learn underlying distribution of SM data as Gaussians
- New events: returns probability to belong to a certain Gaussian



[sklearn.cluster.DBSCAN]
[sklearn.mixture.BayesianGaussianMixture]

Anomaly detection

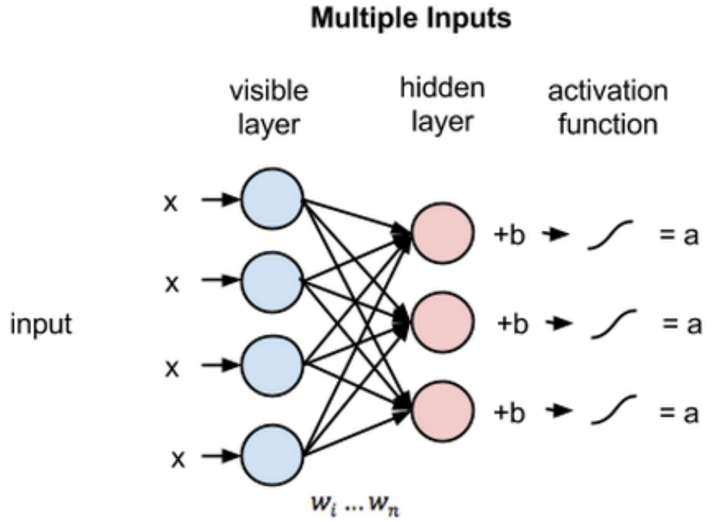
Restricted Boltzmann machine

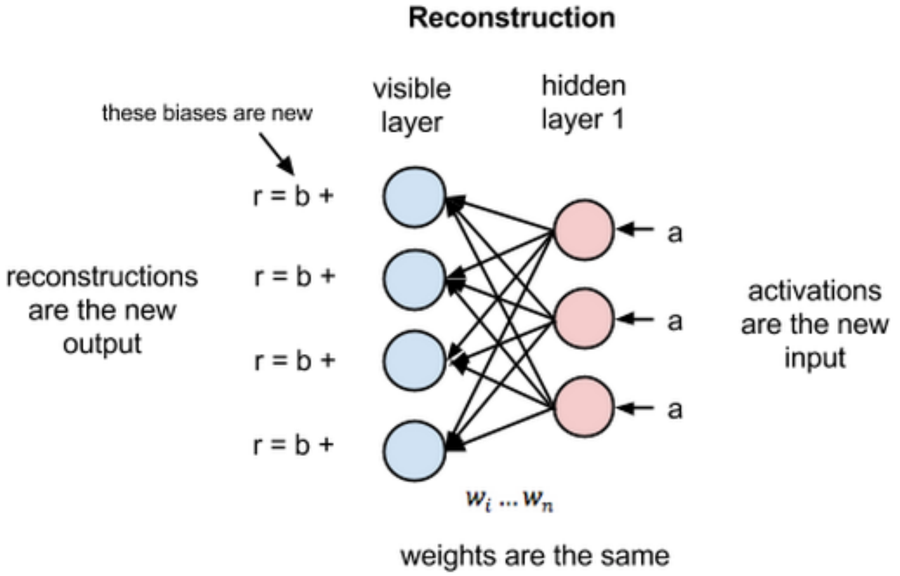
- Back-and-forth learning representation of data in hidden layer
- Quality of encoding: ‘Energy’ as cost function (a la Ising model / hopfield network)

$$E(v,h) = -\sum_{i \in visible} a_i v_i - \sum_{j \in hidden} b_j h_j - \sum_{i,j} v_i h_j W_{ij} \quad ; \qquad P(x) = \sum_{x'} \frac{e^{-E(x,x')}}{Z}$$

- High Energy —> Less probability of creating pattern x
- Training: find W for which P(x) is max, given {x}, e.g. with ‘Contrastive Divergence’
- Stacked RBMs (Deep belief networks), continuous RBMs, …

sklearn.neural_network.BernoulliRBM



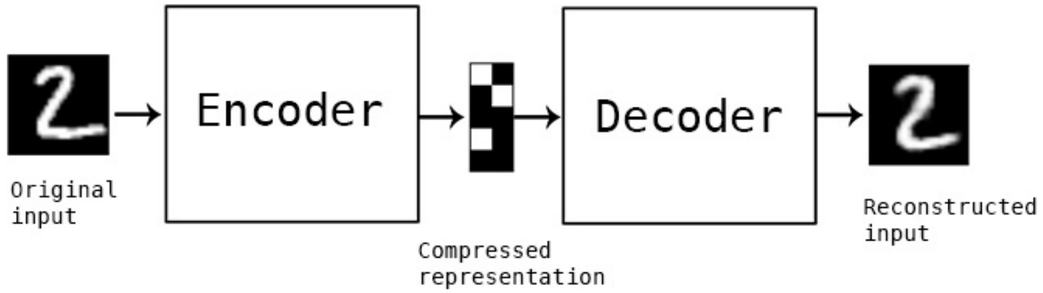


Anomaly detection

Autoencoders

- Learn to encode and decode SM data
- New events: cannot properly encode,
  large loss (= difference between reco and original)

- Consider structure of encoder/decoder:
  for instance a convolutional layer to find
  local correlations between particles

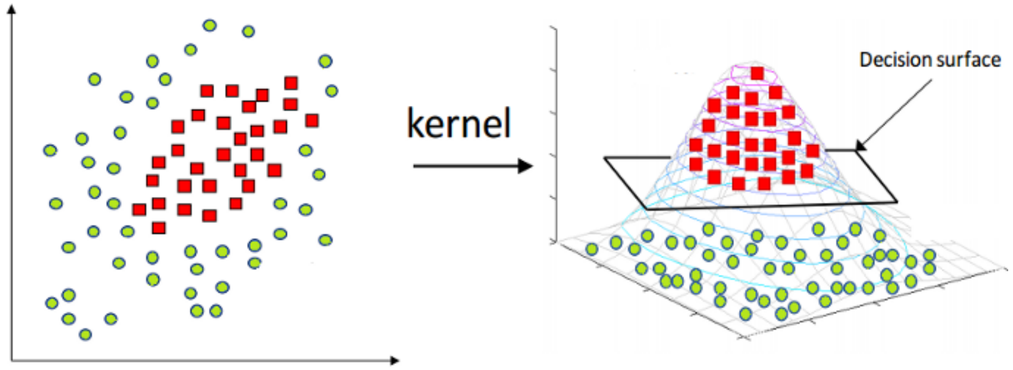


https://blog.keras.io/building-autoencoders-in-keras.html

Anomaly detection

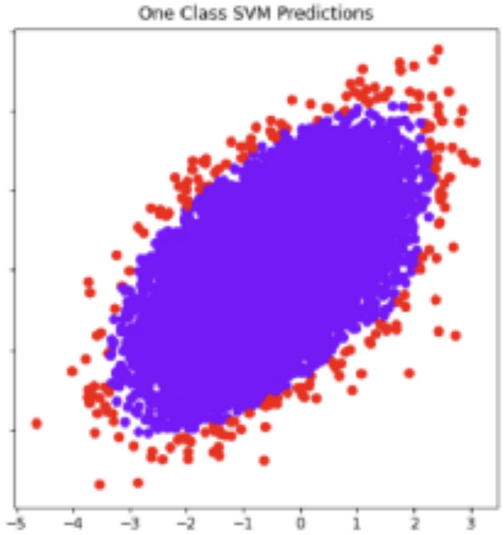
SVM

- Use kernel to transform to hyperspace
- Find optimal separation plane in hyperspace



→ one-class SVM

- Learn boundary of normal data
- New events: Label ‘anomaly’ if outside of boundary



- Choice of Kernel is vital

[sklearn.svm.OneClassSVM]

Tutorial this afternoon II

⟶ Pick your favourite anomaly detection algorithm

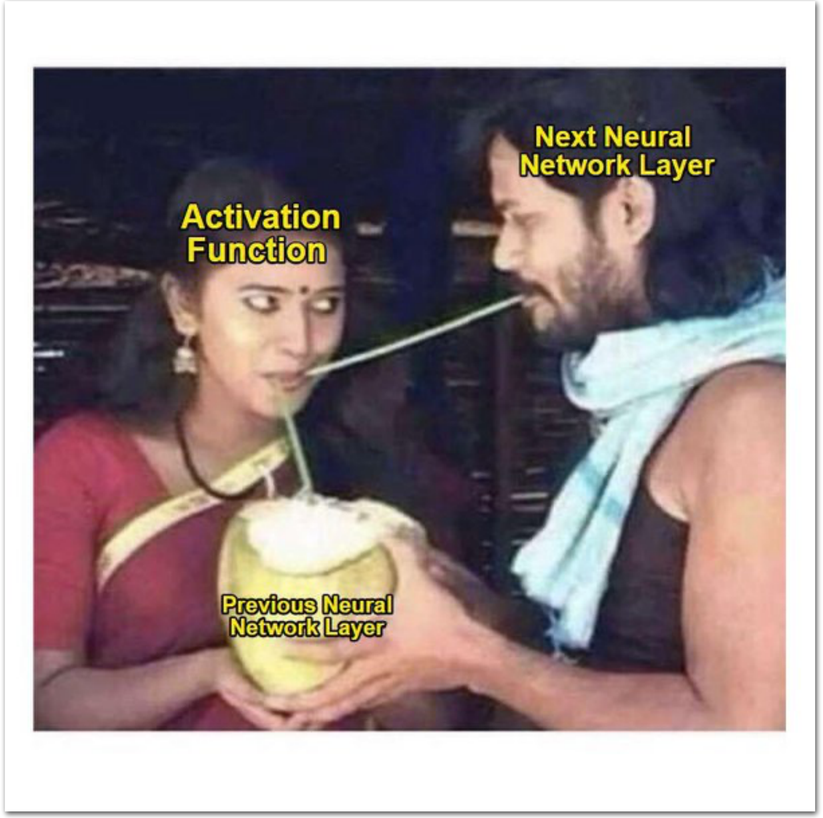
⟶ Pick your favourite New Physics candidate

⟶ Find your anomalies!

⟶ At the end, we will compare performance
- ROC AUC
- inference timing

(- training time?)

⟶ Submit black box dataset for competition?



https://colab.research.google.com/drive/1BNAqCEOdGx5erR7Pr56VDjLfkRAq8yPe?usp=sharing