Machine Learning in Particle Physics

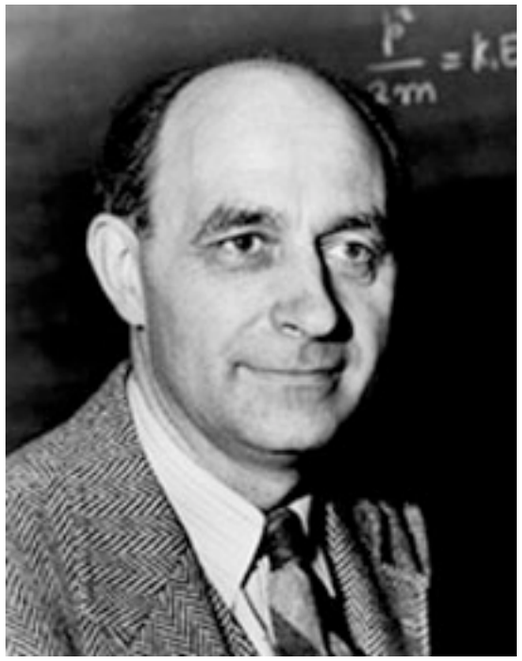
Mike Williams
MIT

June 16, 2017

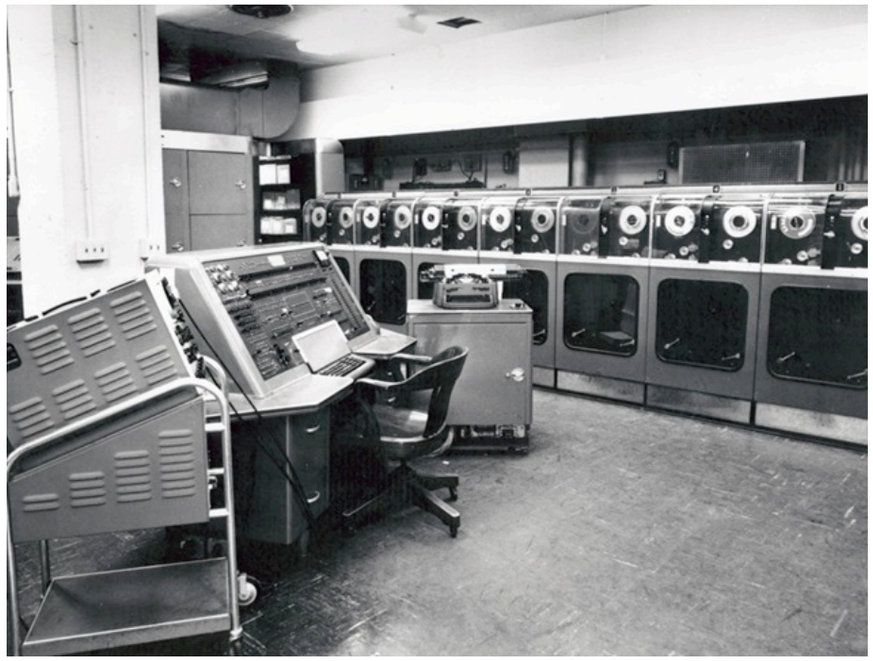
Machine Learning

Supervised machine learning is a broad and evolving field. The most common usage in physics is training algorithms to classify data as signal or background by studying existing labeled (possibly MC) data, though usage is expanding to other areas.

teacher



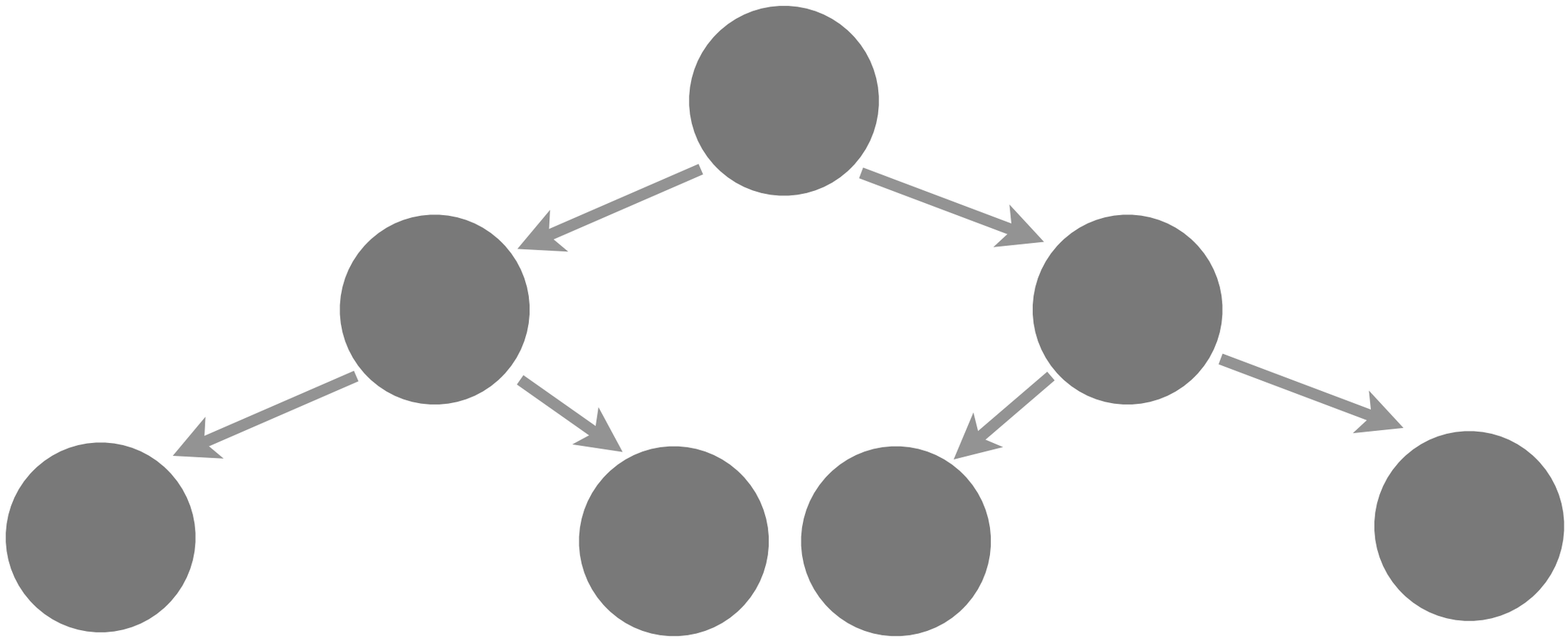
student



There are too many categories of algorithms to even attempt to list them here. In physics, most usage is either a boosted decision tree (BDT) or artificial neural network (ANN) — so I'll briefly describe these. A simple way to think about these is as dimensional-reduction algorithms. The try and learn the optimal way to reduce N-D into 1-D (humans do the same thing with the likelihood ratio).

Decision Trees

Start with a labeled training data sample. These may be obtained from simulation, data sidebands, control modes, etc.

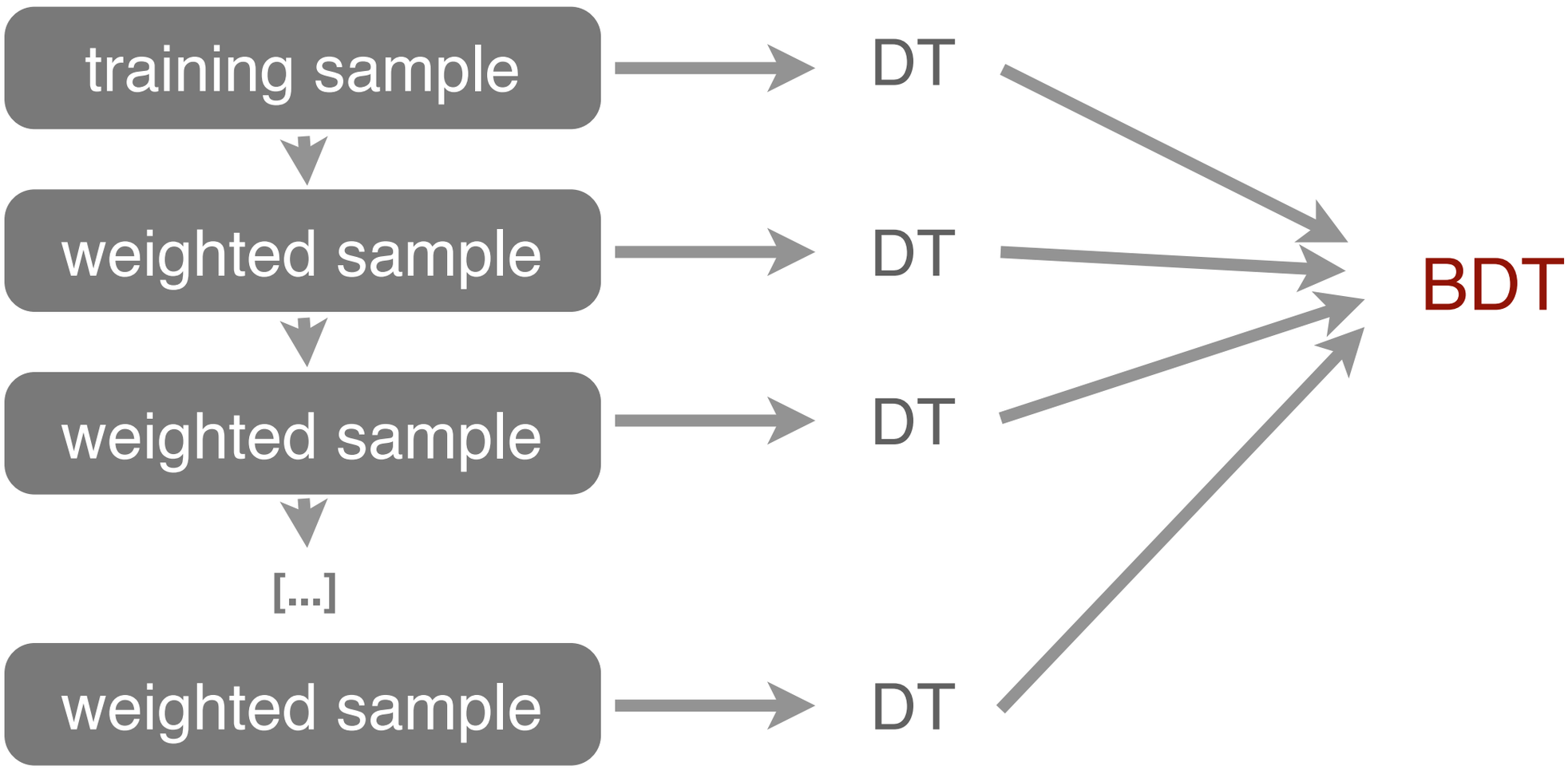


[etc]

Repeatedly split the data to maximize some FOM trying to produce pure B or S “leaves”. Stop when can’t improve FOM, or when reaching some stopping criteria (a subset of algorithm-specific hyper-parameters).

Boosting

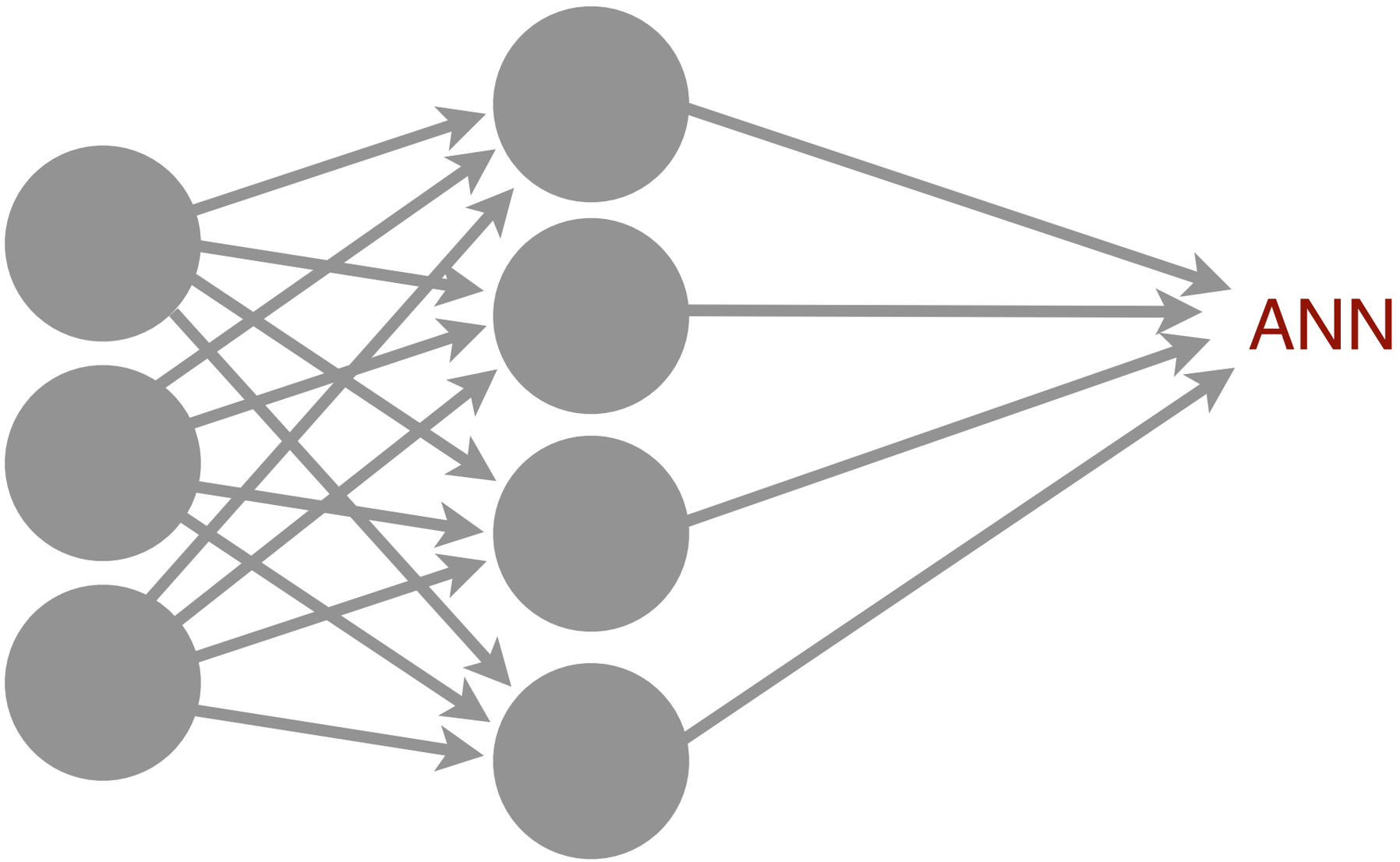
Boosting is a family of methods that produce a series of “weak” classifiers that when combined are extremely powerful.



One common approach: Each DT in the series “boosts” the weight of events based on trying to minimize some loss function. The end result is just a large set of if-type statements leading to a return (many hyper-parameters).

Neural Networks

ANNs send data from input neurons via synapses to a hidden layer (or layers) of neurons, and then to the output neurons via more synapses.



Learning is typically done via back-propagation of the cost-function gradient w.r.t. the NN parameters — the end result is just a function.

Overtraining

Want to avoid learning specifics of the training data set (overtraining). Same idea as overfitting, this results in a less predictive algorithm.

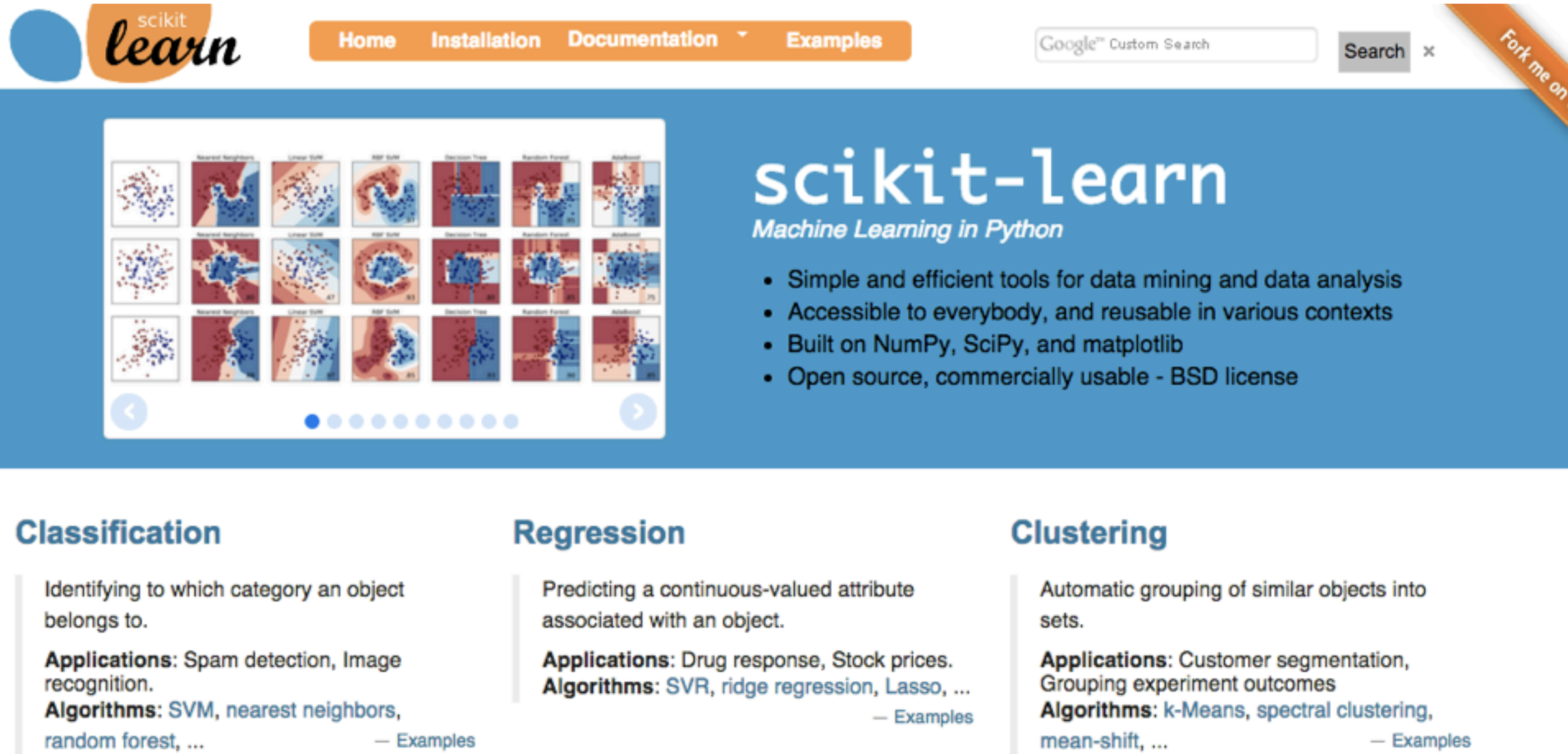
It is easy to spot severe overtraining by comparing the performance on the training data set to an independent (validation/testing) data set (cross validation).

In short, most modern algorithms are pretty good at avoiding this provided “good” hyper-parameters are chosen (these can be problem specific however), and sufficient training data exists.

An enlightened way of optimizing machine learning hyper-parameters is Bayesian Optimization (e.g. the spearmint package is a very good option).

Tools

Physicists used to mostly use TMVA in ROOT; however, the rest of the world is using the python scikit-learn package (sklearn for short), Keras, etc., and our field is also moving this way.



Basics: Adaboost DT or Multilayer Perceptron NN (MLP); State-of-the-Art: XGBoost DT or Deep NN (e.g. Tensorflow).

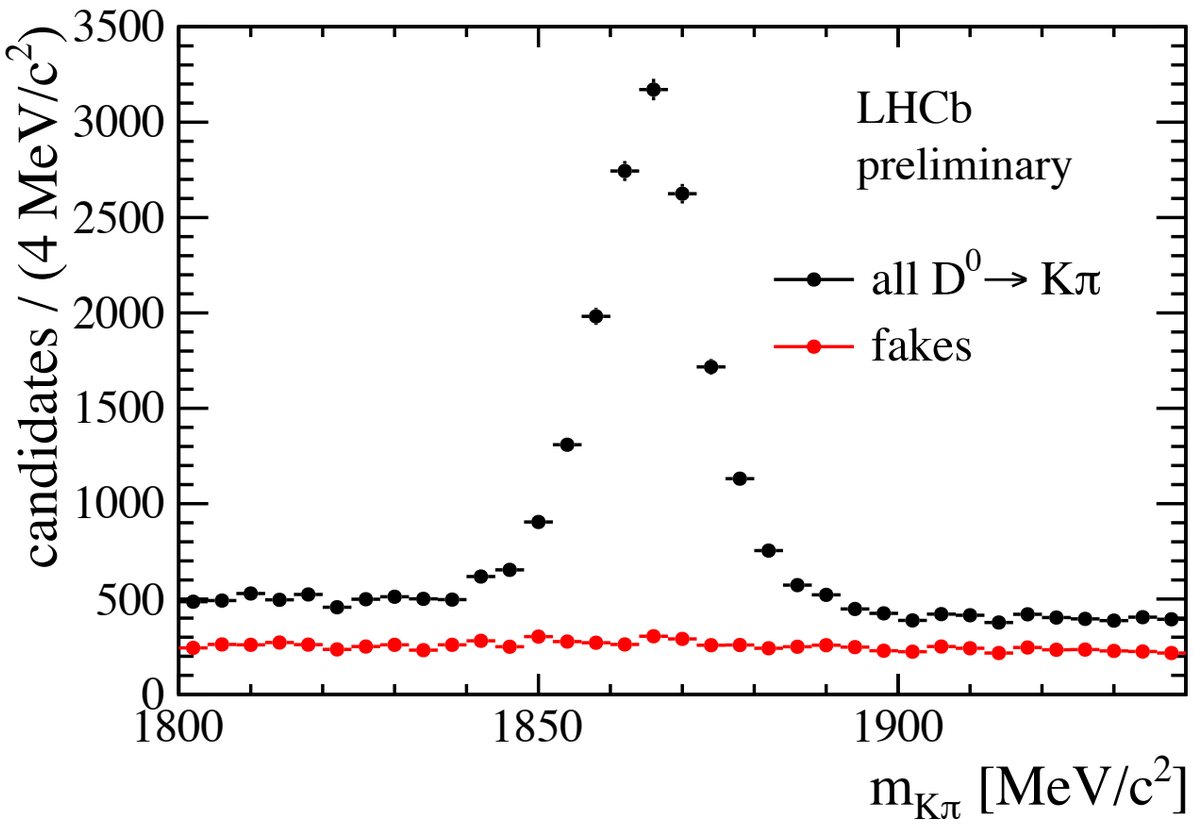
In theory there's no difference between theory and practice. In practice there is.

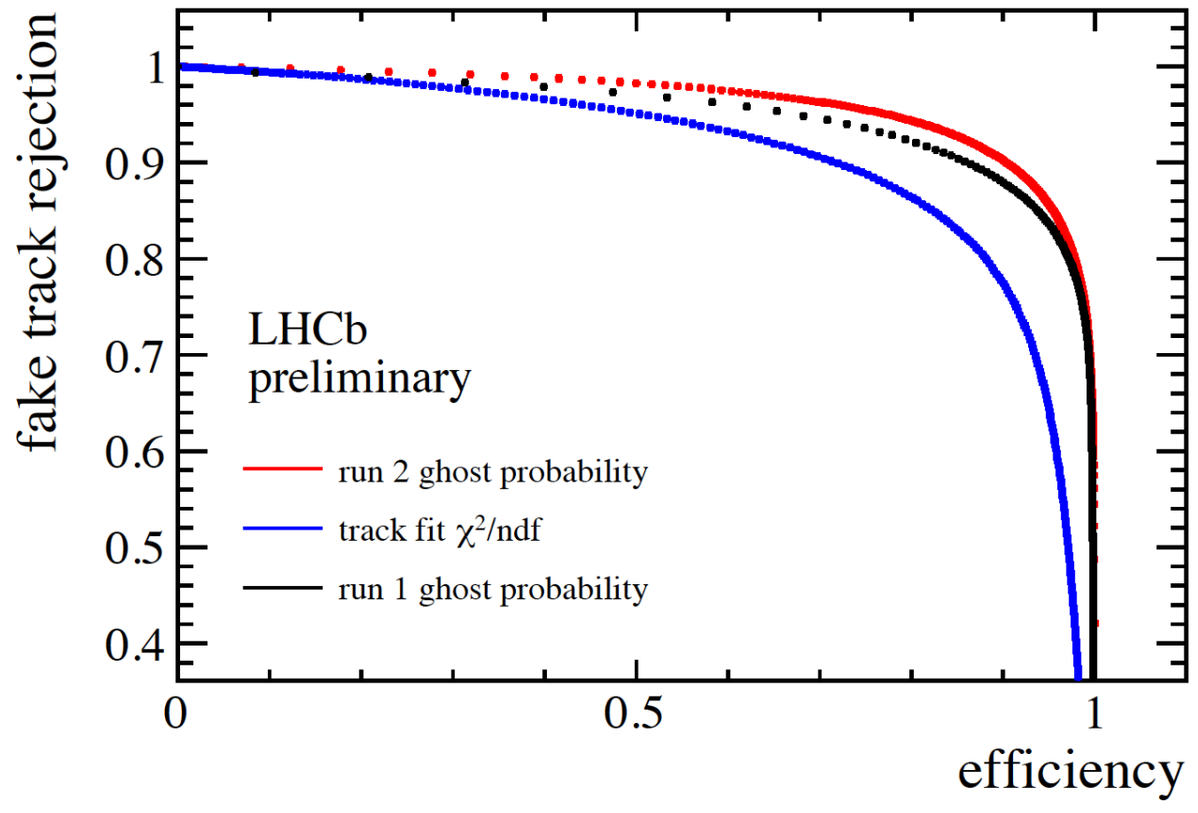
Yogi Berra

Fake-Track Killer

Fake-track-killing neural network, most important features are hit multiplicities and track-segment chi2 values from tracking subsystems.

LHCb-PUB-2017-011

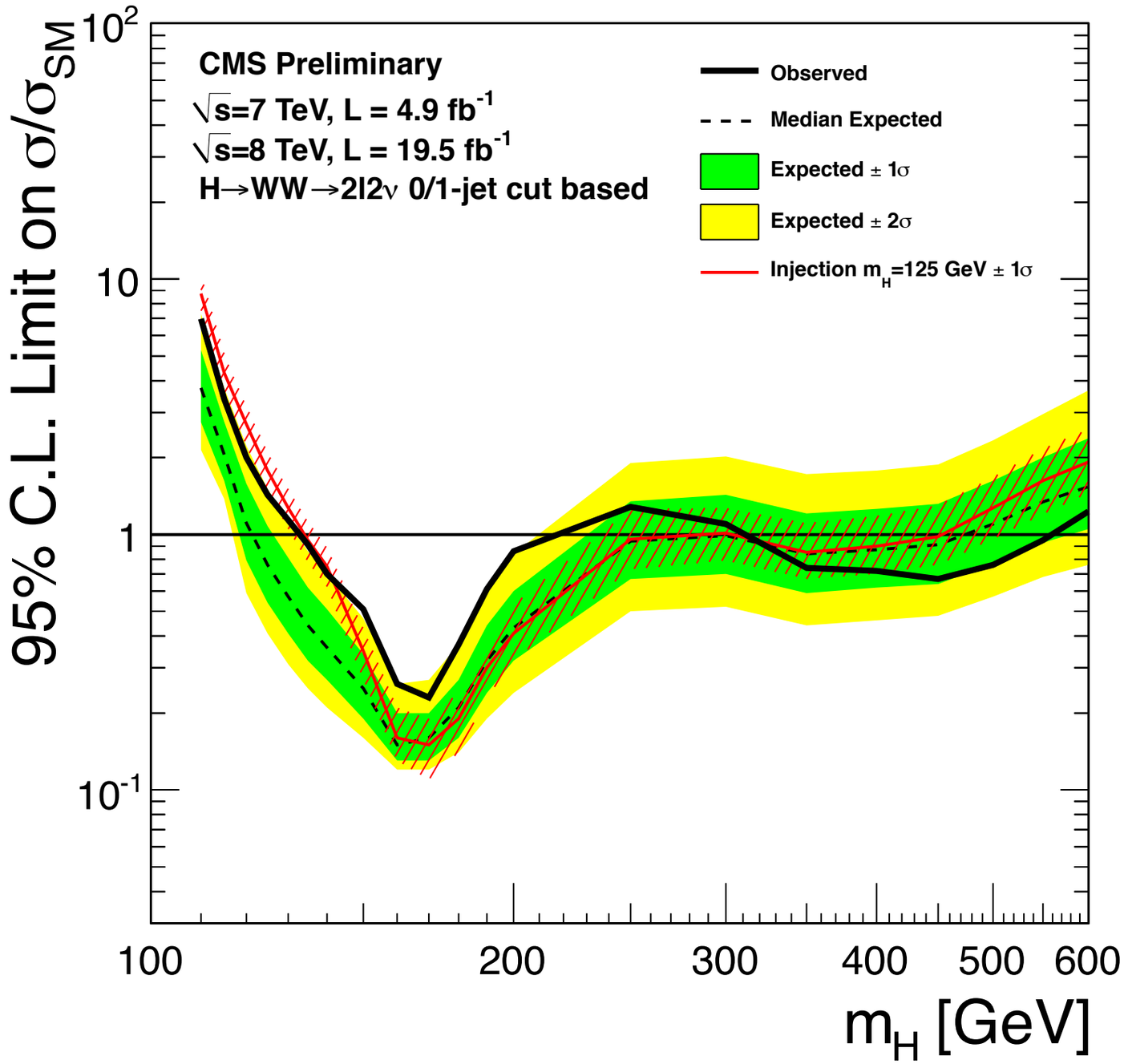




Run in the trigger on all tracks, so must be super fast. Use of custom activation function and highly-optimized C++ implementation (ROOT's TMVA package provides stand-alone C++ code to run the trained algorithm).

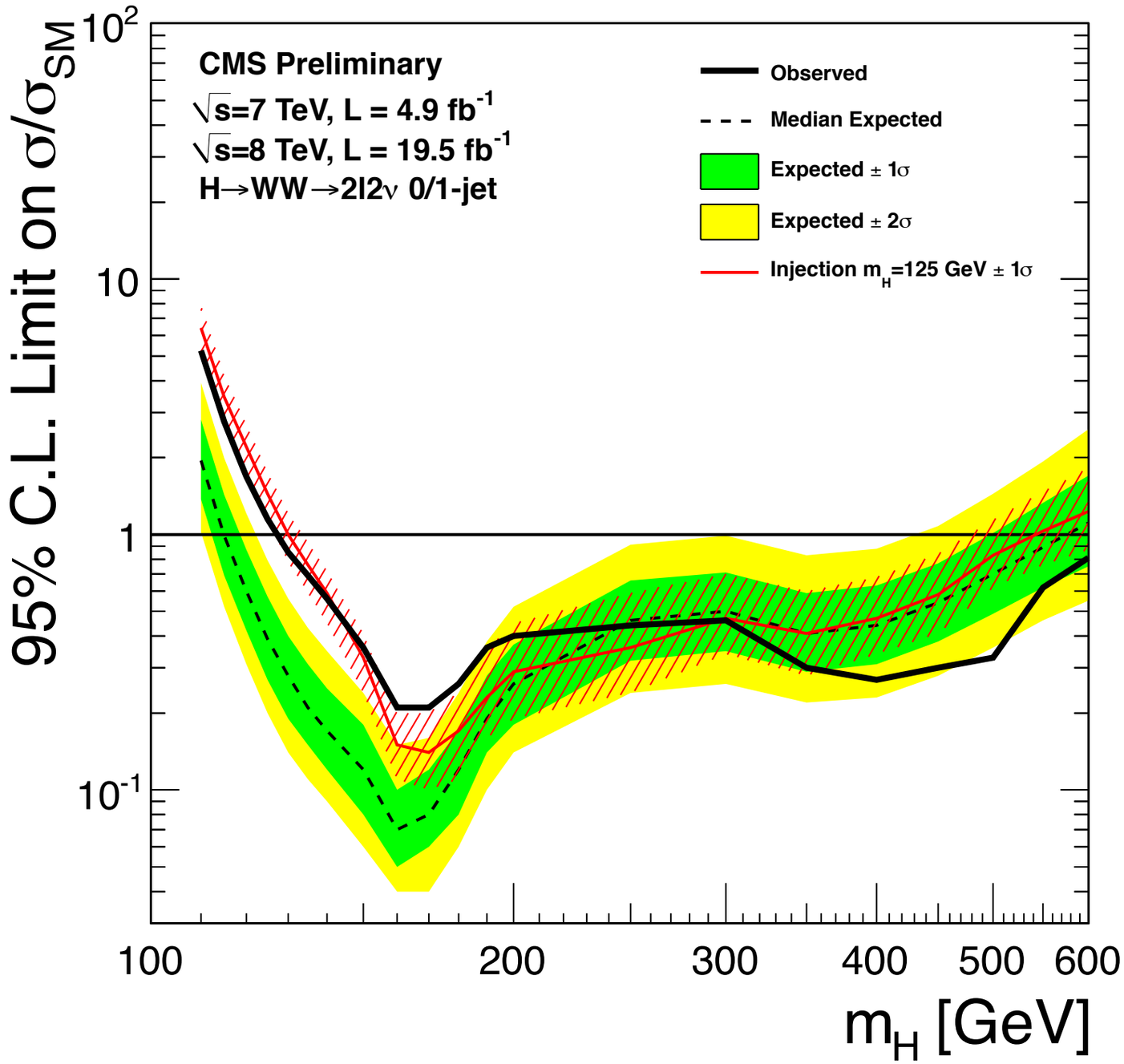
Higgs to WW @ CMS

CMS actually published both cut-based and BDT-based Higgs exclusions for the WW decay mode — so we can see directly what is gained:



Higgs to WW @ CMS

Large regions of possibly m(H) values are excluded by the BDT-based analysis that are NOT excluded by the cuts — using the same data!



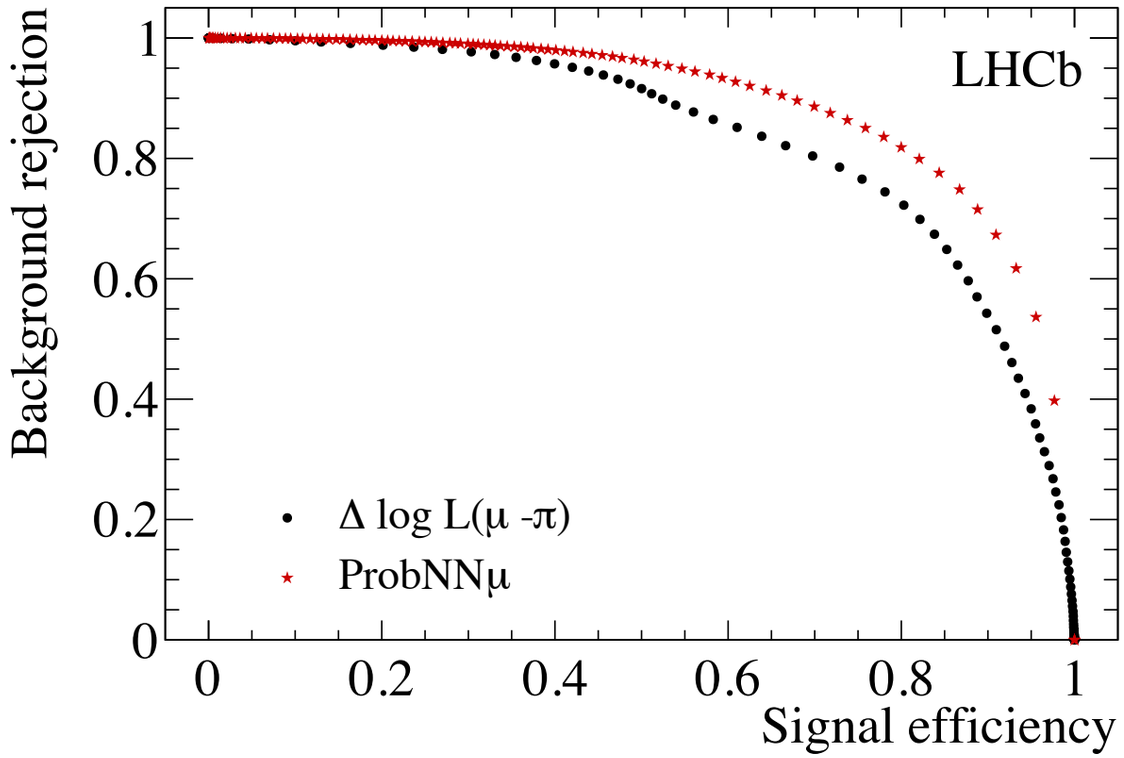
Charged PID

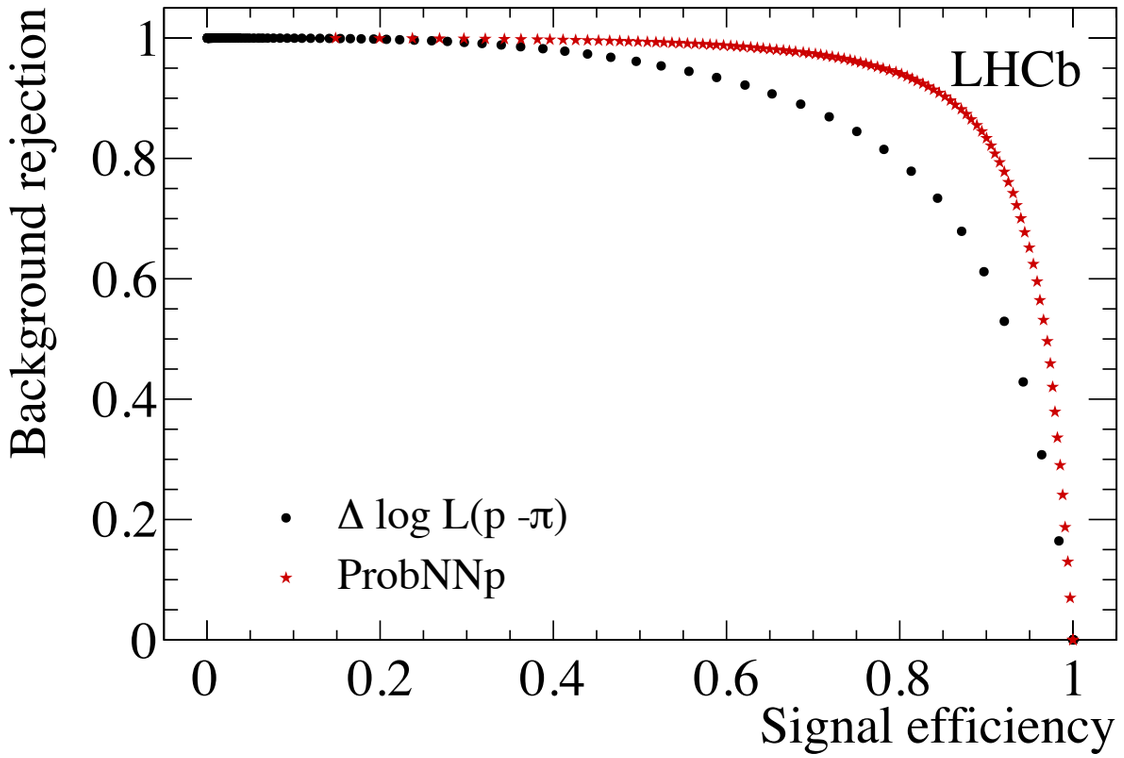
Charged PID: determining whether a track originates from an e, μ, π, K, p, or fake.

Info from the tracking, calorimeter, RICH, and muon systems all play an important role here—and are correlated.

PID NNs

Single-hidden-layer NN trained on 32 features from all subsystems. Each is trained to identify a specific type of particle (or fake track).





Typically get a factor of 3x less pion contamination in a muon sample than using the CombDLL approach — 10x less in a dimuon sample!

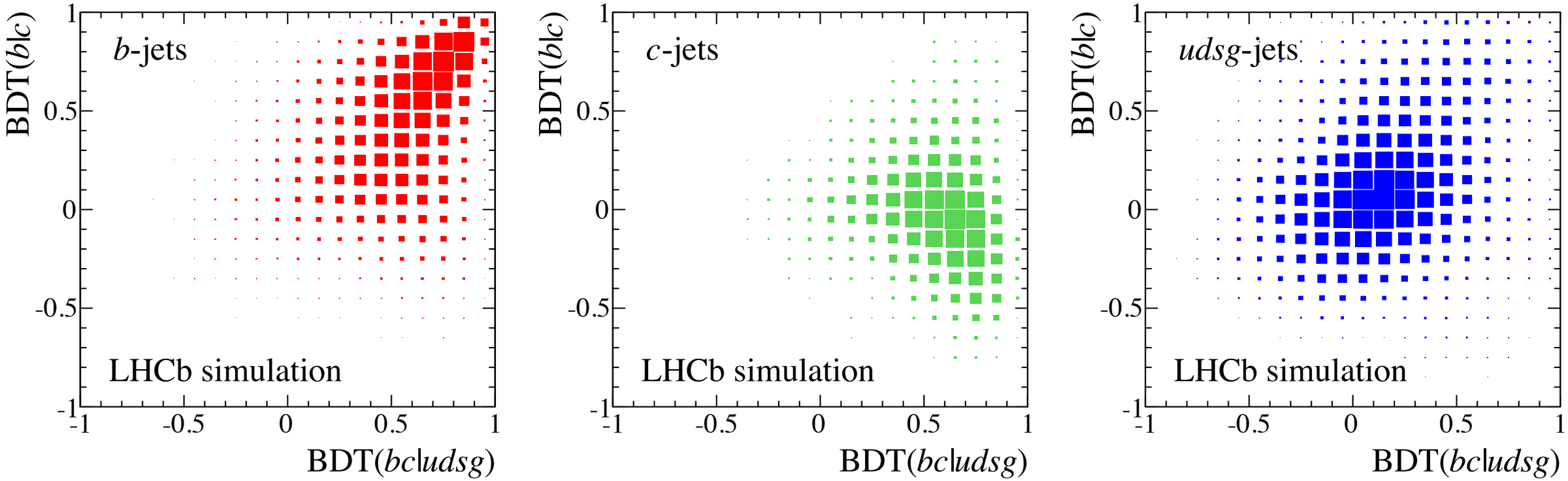
Currently exploring state-of-the-art: XGBoost ~ Deep NN ~ 50% less BKGD than basic BDT or ANN, which again give 2-3x less BKGD than DLLs.

ML Jet Tagging

JINST 10 (2015) P06013
LHCb-PAPER-2015-016

Put 10 features into two BDTs: one for b,c vs light, and another for b vs c. No feature can fully separate types, but their correlations (largely) can.

LHCb simulation: each distribution normalized to one; 70%, 25%, 1% of b, c, light jets are tagged.



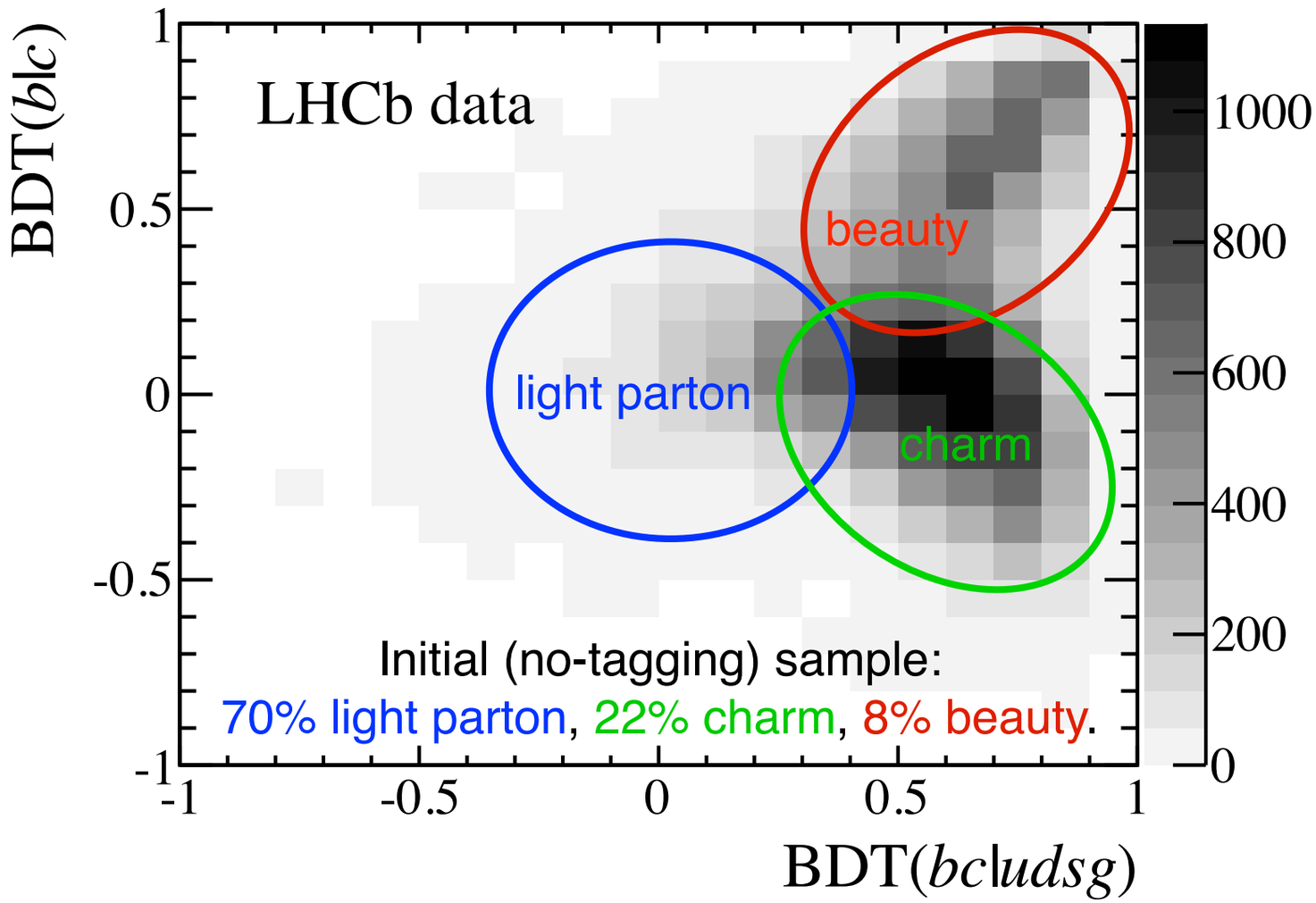
Could cut on BDT responses to obtain high-purity b-jet or c-jet samples. Alternatively, fit 2-D BDT distribution to extract the b-jet and c-jet yields.

Looked at doing a single 3-class algorithm but that doesn't seem to help here (shown to work better in other applications).

ML Jet Tagging

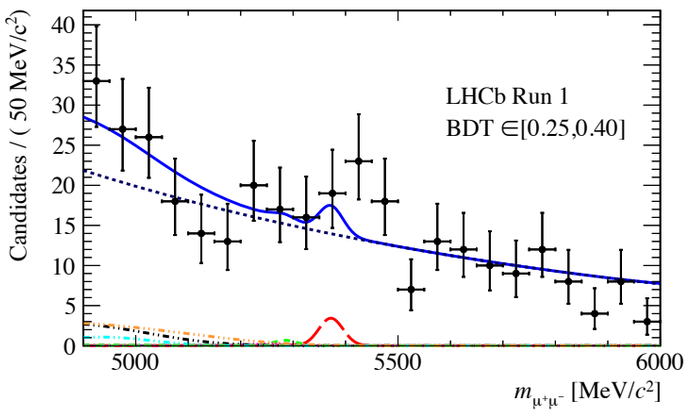
JINST 10 (2015) P06013
LHCb-PAPER-2015-016

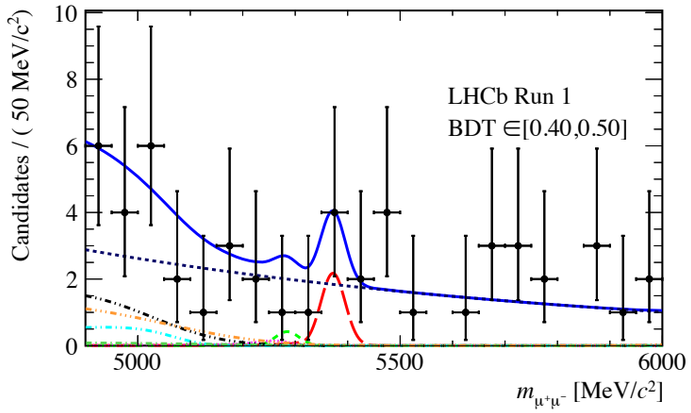
2-D BDT plane (nearly) optimally utilizes 10-D info to ID b, c, and light jets.

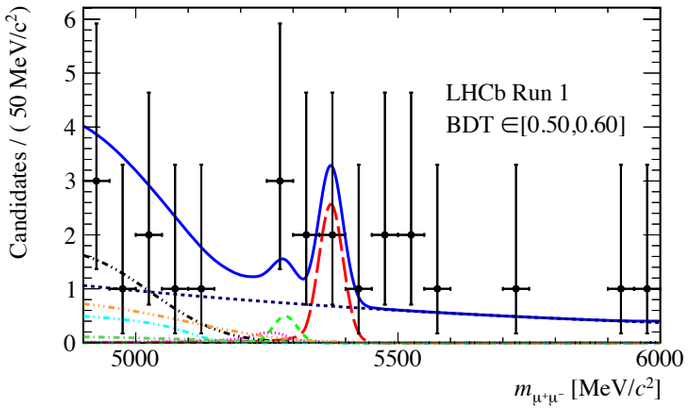


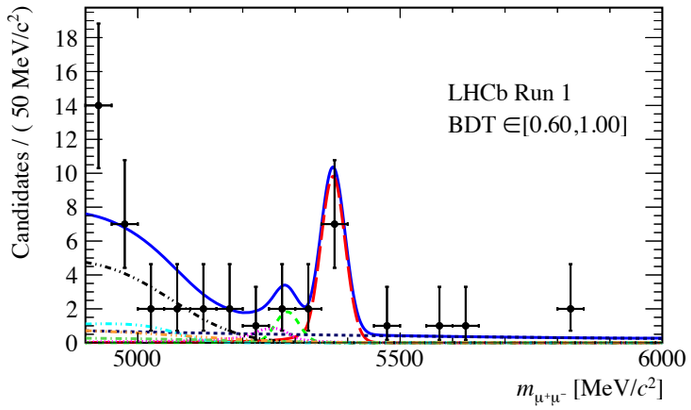
Performance validated & calibrated using large heavy-flavor-enriched jet data samples (2-D data validation much easier than 10-D!). Some analyses cut on these BDT responses, others fit the 2-D distributions to extract b,c,l yields.

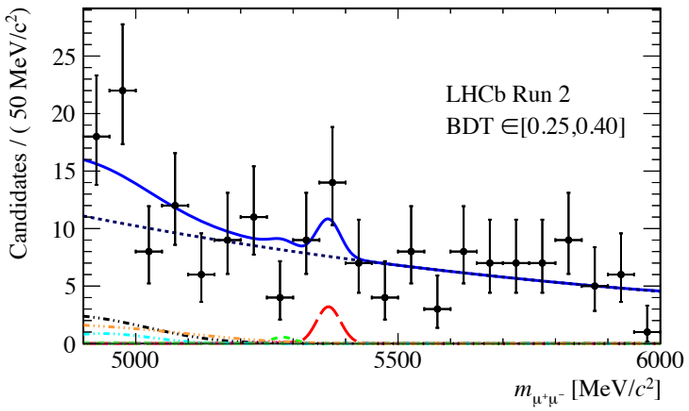
ML in Analysis

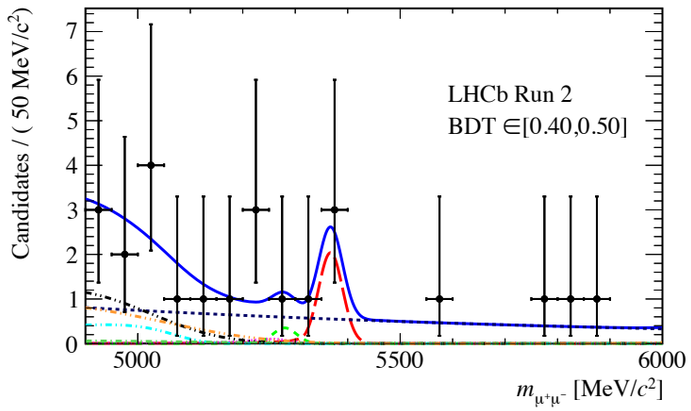


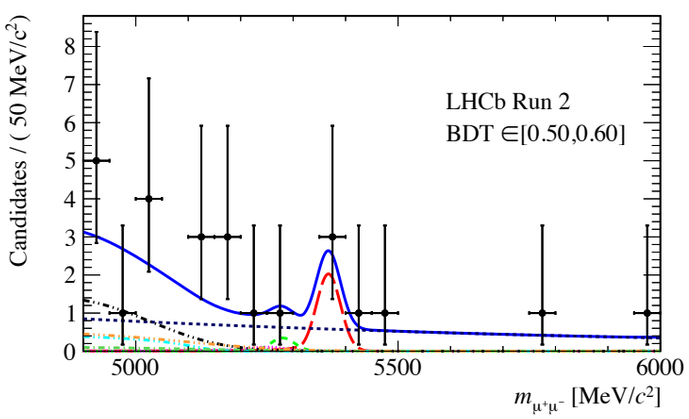


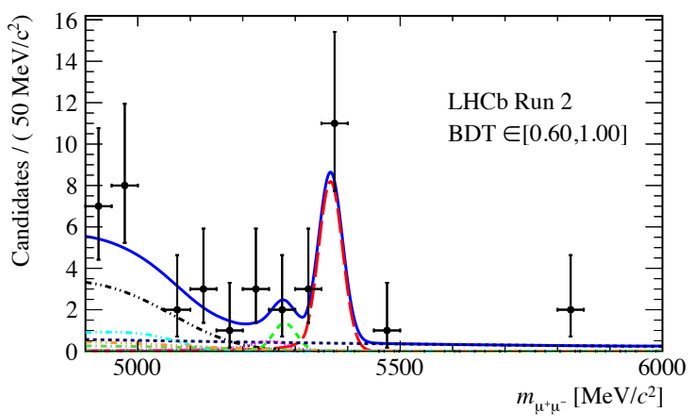












LHCb-PAPER-2017-001

Continuing to move beyond just cutting on response … Calibrate BDT to have uniform response on $B_s \to \mu\mu$ signal, bin data in BDT response and analyze all dimuon mass distributions simultaneously.

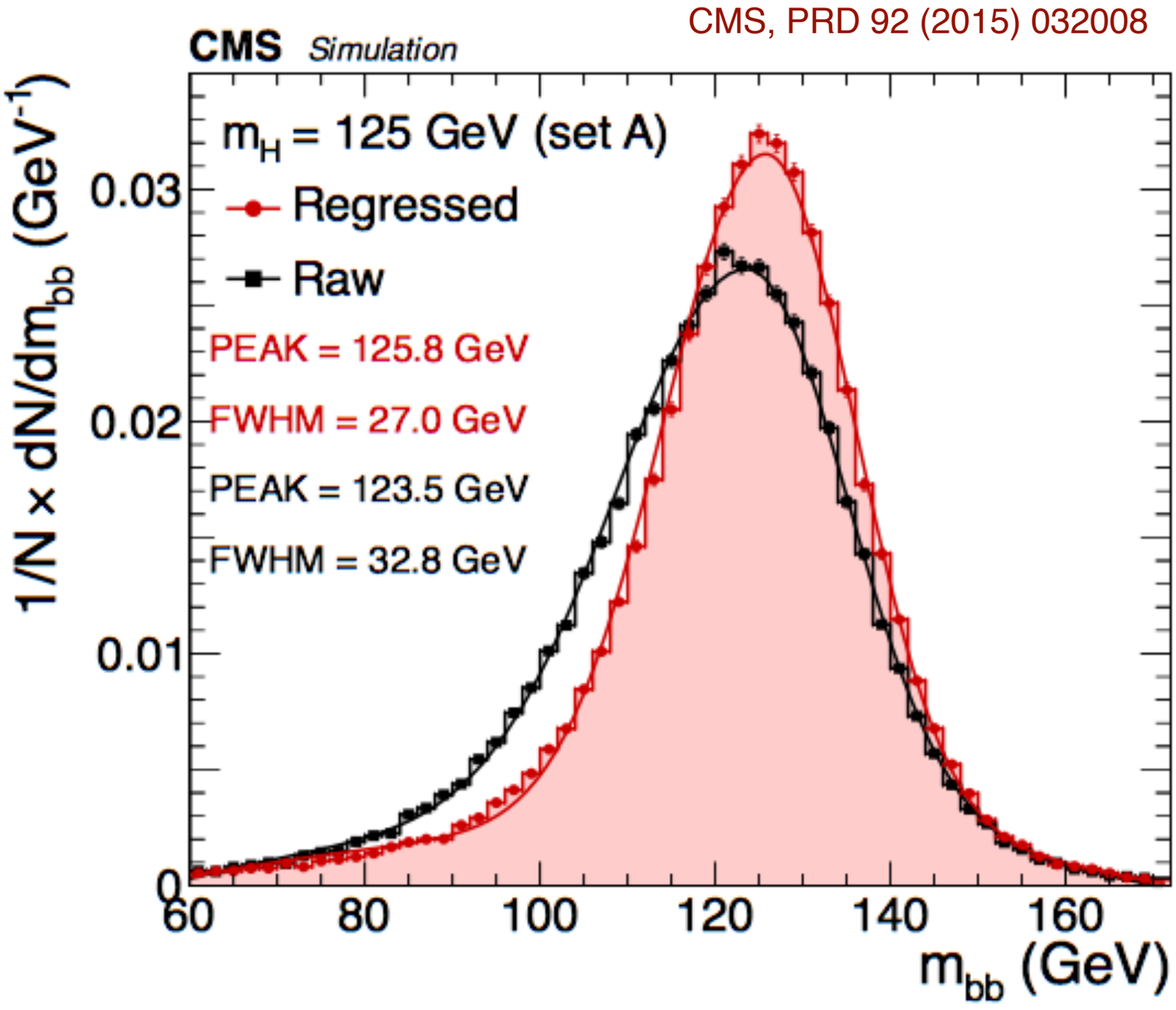
Constraints added to the likelihood for relationships between yields and shapes of the various components from bin to bin.

Can enforce no mass dependence if desired: see J. Stevens, MW [1305.7248]; Rogozhnikova, Bukva, Gligorov, Ustyuzhanin, MW [1410.4140].

Regression

ML is also now being used for regression. For example, CMS has moved to ML-based jet-energy corrections (e.g., for Higgs to bb).

CMS, PRD 92 (2015) 032008



Details

•We typically train our ML algorithms on MC, then characterize their performance using data control samples (same way we characterize our hardware). In principle, data samples could also be used in the training, but then one would need to deal with BKGD in those samples (this is not hard to do using event weights). N.b., make sure to not confuse non-optimal with wrong or the inverse.

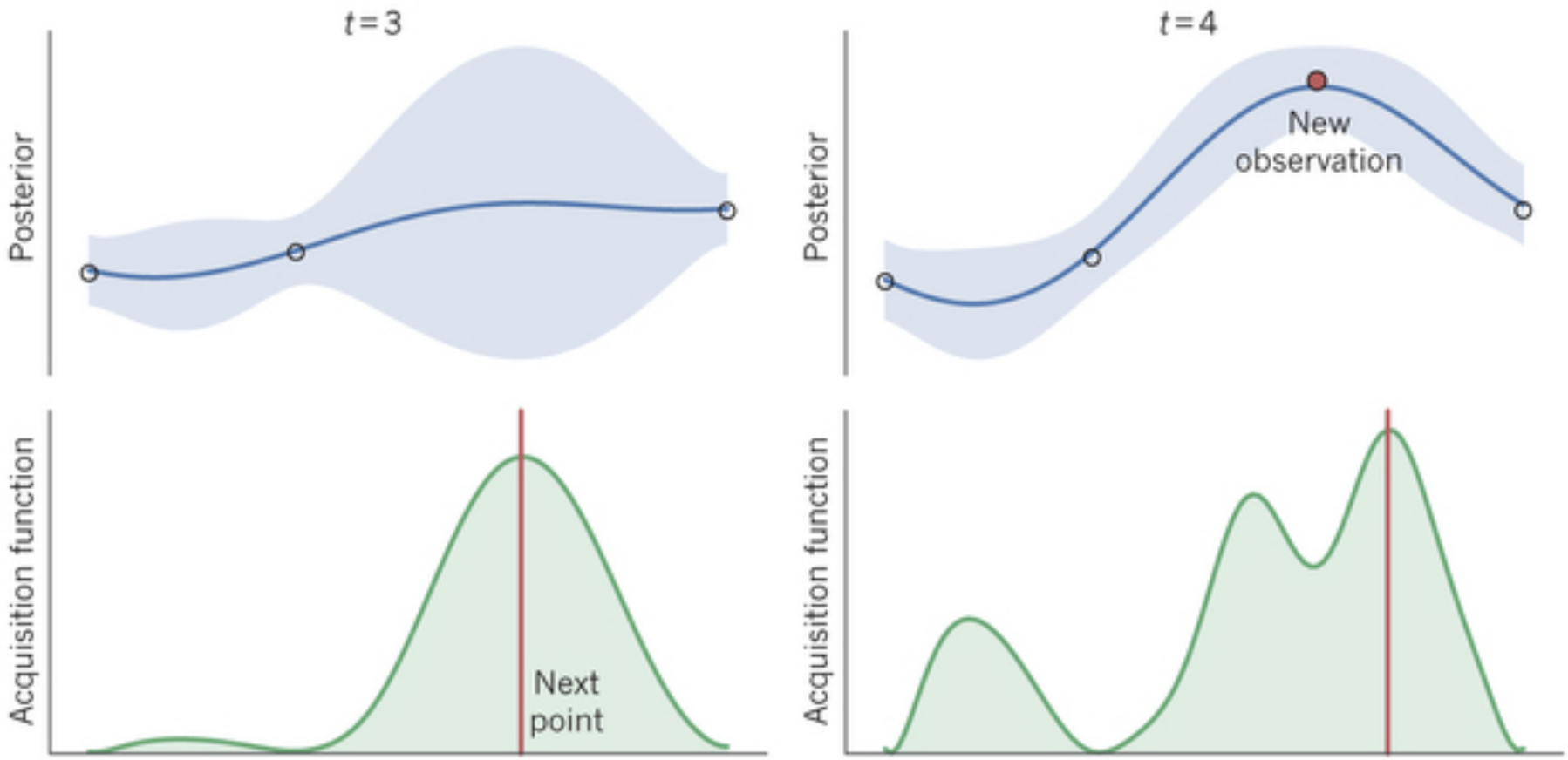
•Dimensional reduction achieved by ML makes it possible to maximize performance without complicating data-driven validation. There are many standard candles at the LHC to use for data-driven validation — and also at JLab (we're doing this now for GlueX).

•As an aside, systematics tend to scale with inefficiency, so a highly-performant black box often incurs a smaller systematic than a simple, less performant algorithm — and also is easier to deal with than hardware (of course there are exceptions).

•Bottom line: We use ML because it enables great science. It greatly improves performance in many areas, even converting some measurements from infeasible to simple & precise. The LHCb trigger is even mostly ML-based, and has been since the start of 2011 data taking (see V.Gligorov, MW, JINST 8 (2012) P02013).

Bayesian Optimization

Bayesian optimization refers to a family of methods that do global optimization of black-box functions (no derivatives required).



Start from prior for objective function, treat evaluations as data and produce a posterior used to determine the next point to sample.

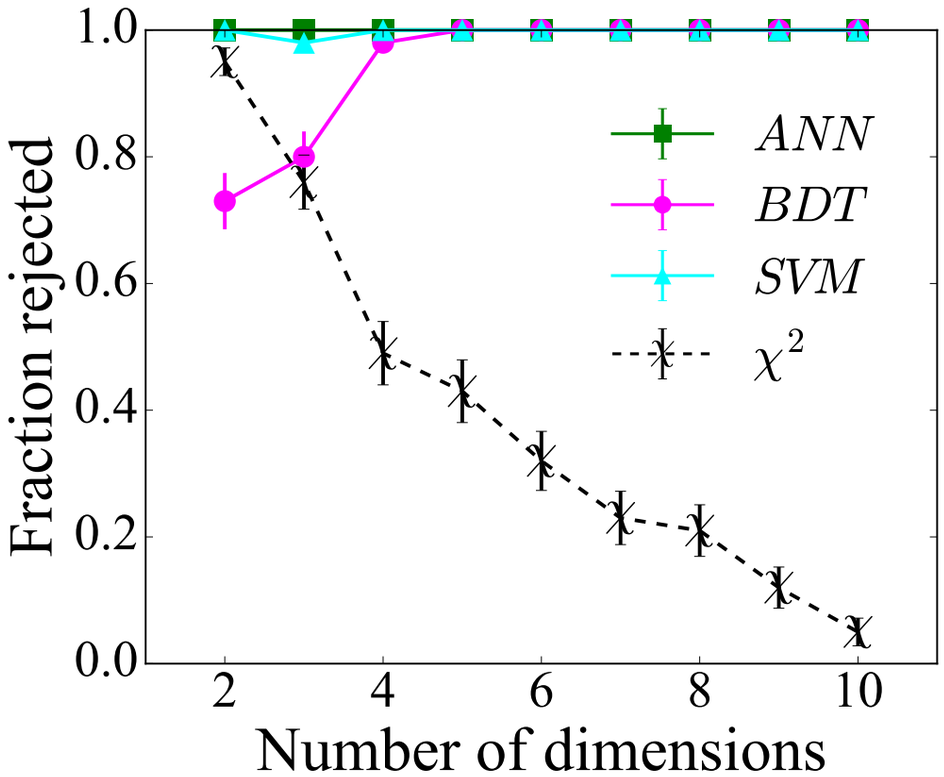
See https://github.com/HIPS/Spearmint for an excellent package in python. Example app: Ilten, MW, Yang, Monte Carlo tuning using Bayesian Optimization, [arxiv:1610.08328], with fully working code on GitHub here: https://github.com/yunjie-yang/TuneMC (could also be used for data calibration, or any black-box problem).

ML & GoF

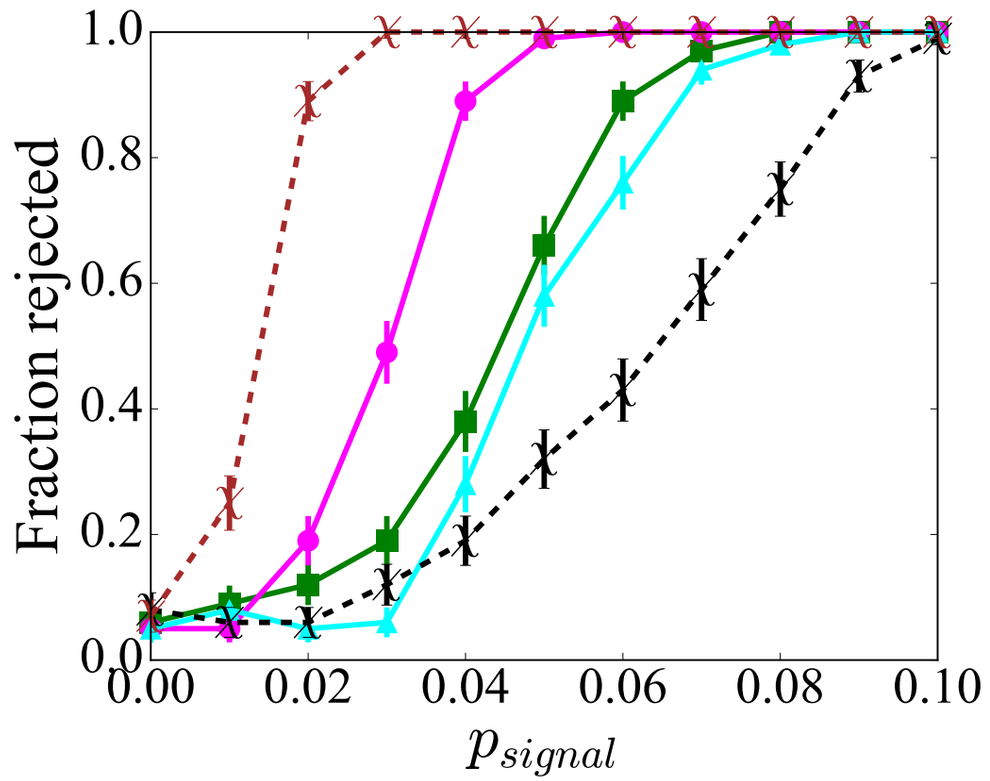
Since ML algorithms learn dimensional reduction, they can also be used to do goodness of fit in high dimensions. This is simple: train a ML algorithm using the data and an MC sample generated from the fit PDF, produce an unbiased 1-D ML response distribution for each data type, then do a 1-D GoF test (e.g. chisquare) on these 1-D distributions (simple).

Weisser, MW [1612.07186]

N-D Gaussian (5% width change)



2-body decay using 4-vectors (not mass)



The ML learns an approximation of the mass here. In this toy example, the mass (which is a weird 8-D manifold) is optimal. Knowing that—and using it—we can beat the machine (Whiteson et al showed Deep Learning can really learn things like the mass and other human-designed features).

Tools, etc.

•ML algorithms in HEP used to be mostly ROOT's TMVA , but are now migrating more and more to scikit-learn, Keras, etc.; i.e., we are moving away from physics-specific software and towards the tools used by the wider ML community. Hyper-parameter tuning using spearmint, hyperopt, etc. (see also Ilten, MW, Yang [1610.08328]).

•Custom loss functions, e.g., response is de-correlated from some set of features (Stevens, MW [1305.7248]; Rogozhnikova, Bukva, Gligorov, Ustyuzhanin, MW [1410.4140]). Already used in several papers (e.g. LHCb, PRL 115 (2015) 161802), and currently being used in many papers to appear soon.

•Many useful tools provided in the HEP-ML package pypi.python.org/pypi/hep_ml/0.2.0, which is basically a wrapper around sklearn, and in REP https://github.com/yandex/rep (both produced by our colleagues at Yandex).

•N.b., beware of non-general optimizations in some algorithms (e.g. CNNs), i.e. make sure to use the right tool for your job.

•Attend the IML meetings (virtually) https://iml.web.cern.ch to learn about what other experiments are doing. Send your students/post-docs to the HEP-ML school, etc.

Third Machine Learning in High Energy Physics Summer School 2017

17-23 July 2017
Reading
Europe/London timezone

Support

The Third Machine Learning summer school organized by Yandex School of Data Analysis, Laboratory of Methods for Big Data Analysis of National Research University Higher School of Economics and Imperial College London will be held in Reading, UK from 17 to 23 July 2017.

The school is intended to cover the relatively young area of data analysis and computational research that has started to emerge in High Energy Physics (HEP). It is known by several names including “Multivariate Analysis”, “Neural Networks”, “Classification/Clusterization techniques”. In more generic terms, these techniques belong to the field of “Machine Learning”, which is an area that is based on research performed in Statistics and has received a lot of attention from the Data Science community.

There are plenty of essential problems in High energy Physics that can be solved using Machine Learning methods. These vary from online data filtering and reconstruction to offline data analysis.

Students of the school will receive a theoretical and practical introduction to this new field and will be able to apply acquired knowledge to solve their own problems. Topics ranging from decision trees to deep learning and hyperparameter optimization will be covered with concrete examples and hands-on tutorials. A special data-science competition will be organized within the school to allow participants to get better feeling of real-life ML applications scenarios.

Expected number of students for the school is 50-60 people

Summary

•Machine learning algorithms are now commonplace in HEP analyses. Open-source tools are now very good and getting better daily.

•ML algorithms exploit high-dimensional correlations to improve on cut-based selections (can also view them as dimensional-reduction methods). Even basic algorithms tend to give big improvements, and state-of-the-art algorithms are now easy for novices to use.

•It's vital that such algorithms can be validated/calibrated in a data-driven approach—and not just because your more senior colleagues don't like them!

•Trust me, this is all easier to use than you think. I will (hopefully) prove this to you after lunch. The real work is in obtaining/creating good training samples, and in validating the performance in a data-driven way. Everything else is really trivial now, which means physicists can spend ~100% of their time on systematics instead of designing selections (as it should be).

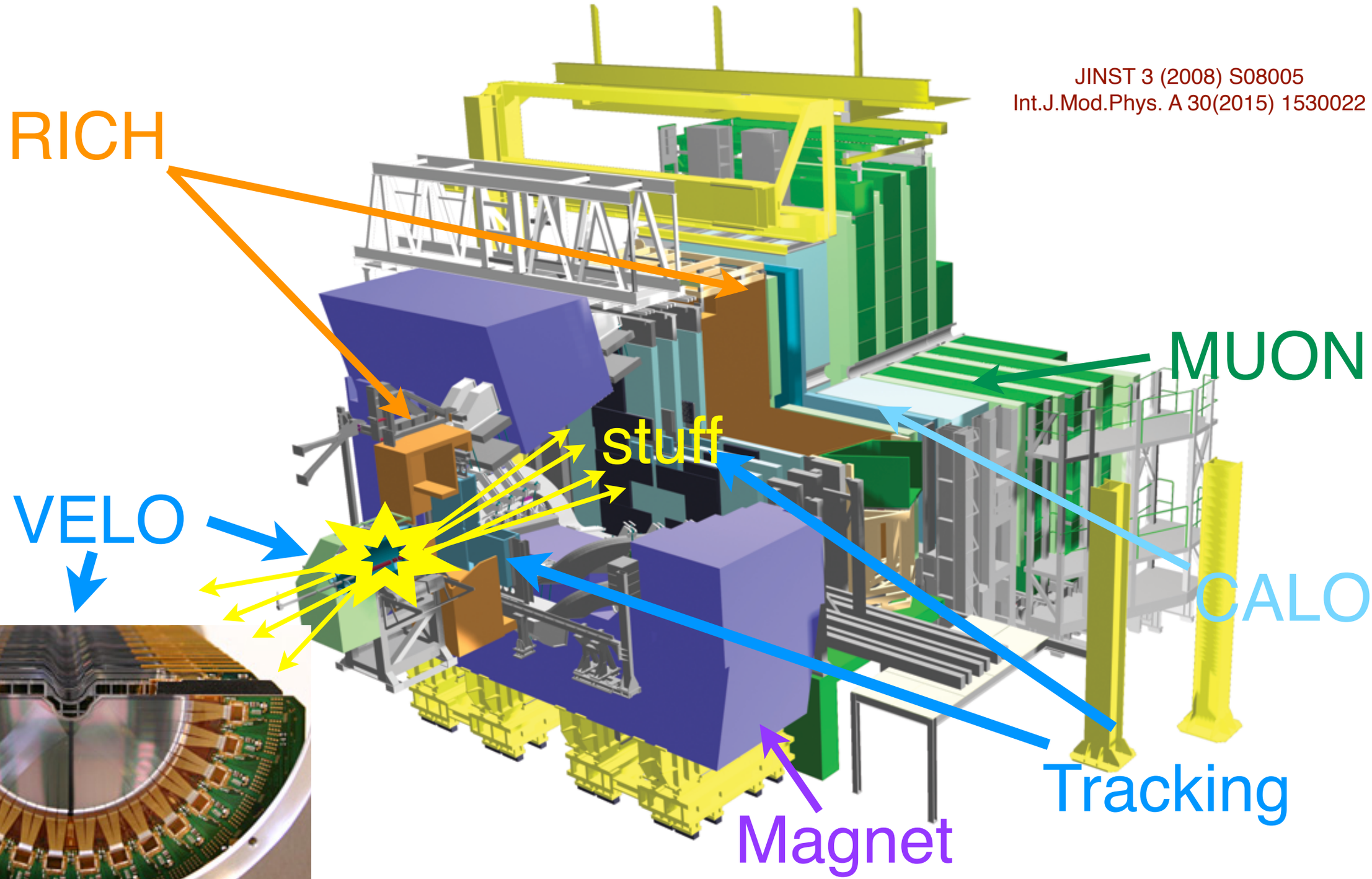
•Near future: Deep learning, ML-based reconstruction, ML-based compression, …

LHCb Detector

LHCb is a forward Spectrometer ($2 < \eta < 5$)

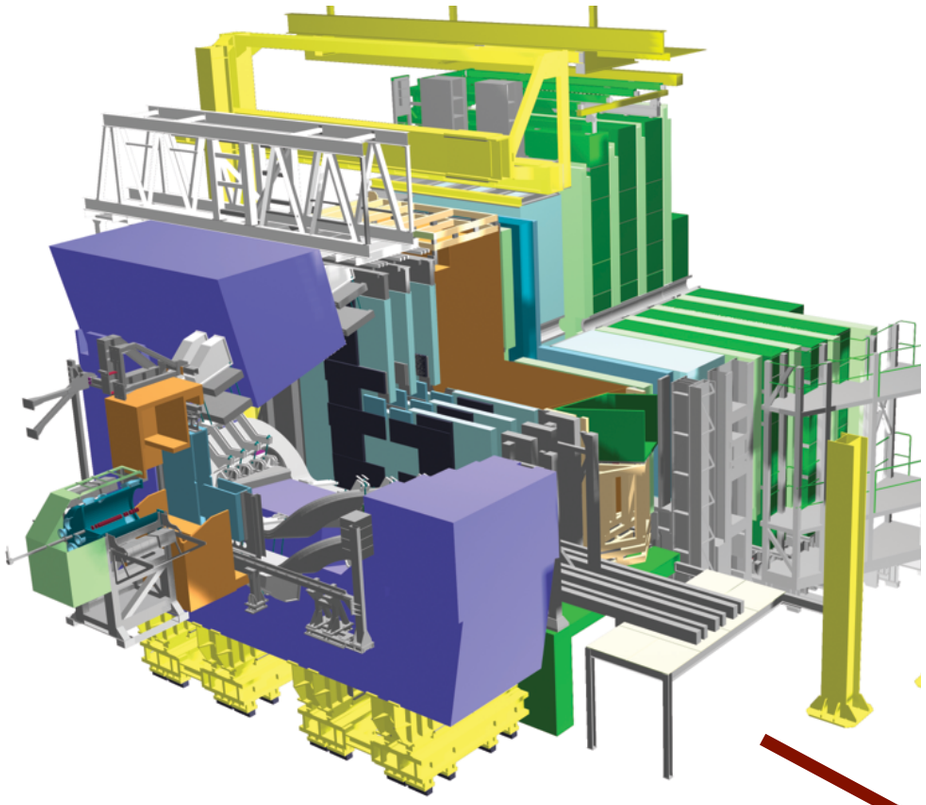
(roughly 1-15^o)

JINST 3 (2008) S08005
Int.J.Mod.Phys. A 30(2015) 1530022



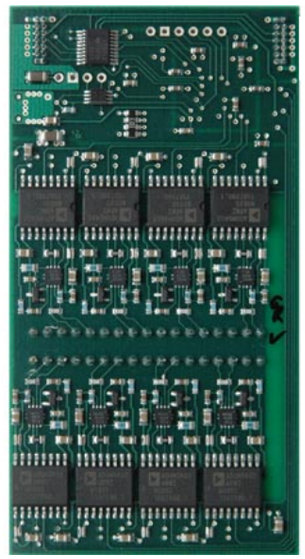
JINST 8 (2013) P04022

Real-Time Processing



Simple feature-building in custom electronics (e.g. FPGAs) required to reduce the data volume to a transferable rate.

TB/s
(post zero suppression)

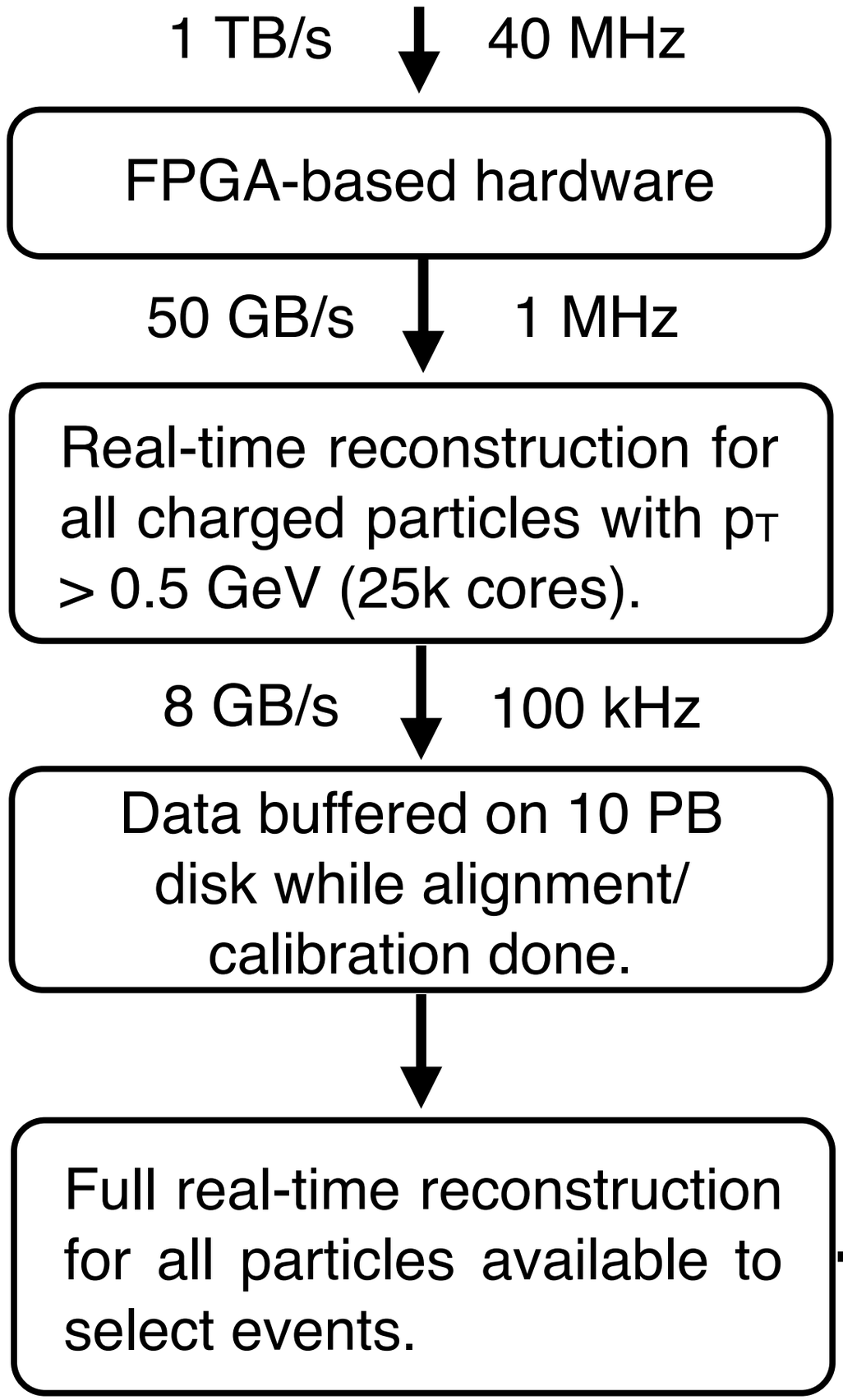


50 GB/s

Online computing farm processes 250 PB / year, can only persist 1% of this.

LHCb will move to a triggerless-readout system for LHC Run 3 (2021-2023), and process 5 TB/s in real time on the CPU farm.

Real-Time Processing (Run 2)



Precision measurements benefit greatly from using the final (best) reconstruction in the online event selection—need real-time calibration!

Final event selection done with access to best-quality data (mostly done during down time between fills), removing the need (but perhaps not the desire) to retain the ability to re-reconstruct the data offline.

Real-Time Processing (Run 2)

1 TB/s ↓ 40 MHz

FPGA-based hardware

50 GB/s ↓ 1 MHz

Real-time reconstruction for all charged particles with p_T > 0.5 GeV (25k cores).

Heavy use of machine learning algorithms throughout the Run 1 and Run 2 trigger.

V.Gligorov, MW, JINST 8 (2012) P02013.

70% of output events here classified using ML algorithms.

8 GB/s ↓ 100 kHz

Data buffered on 10 PB disk while alignment/ calibration done.

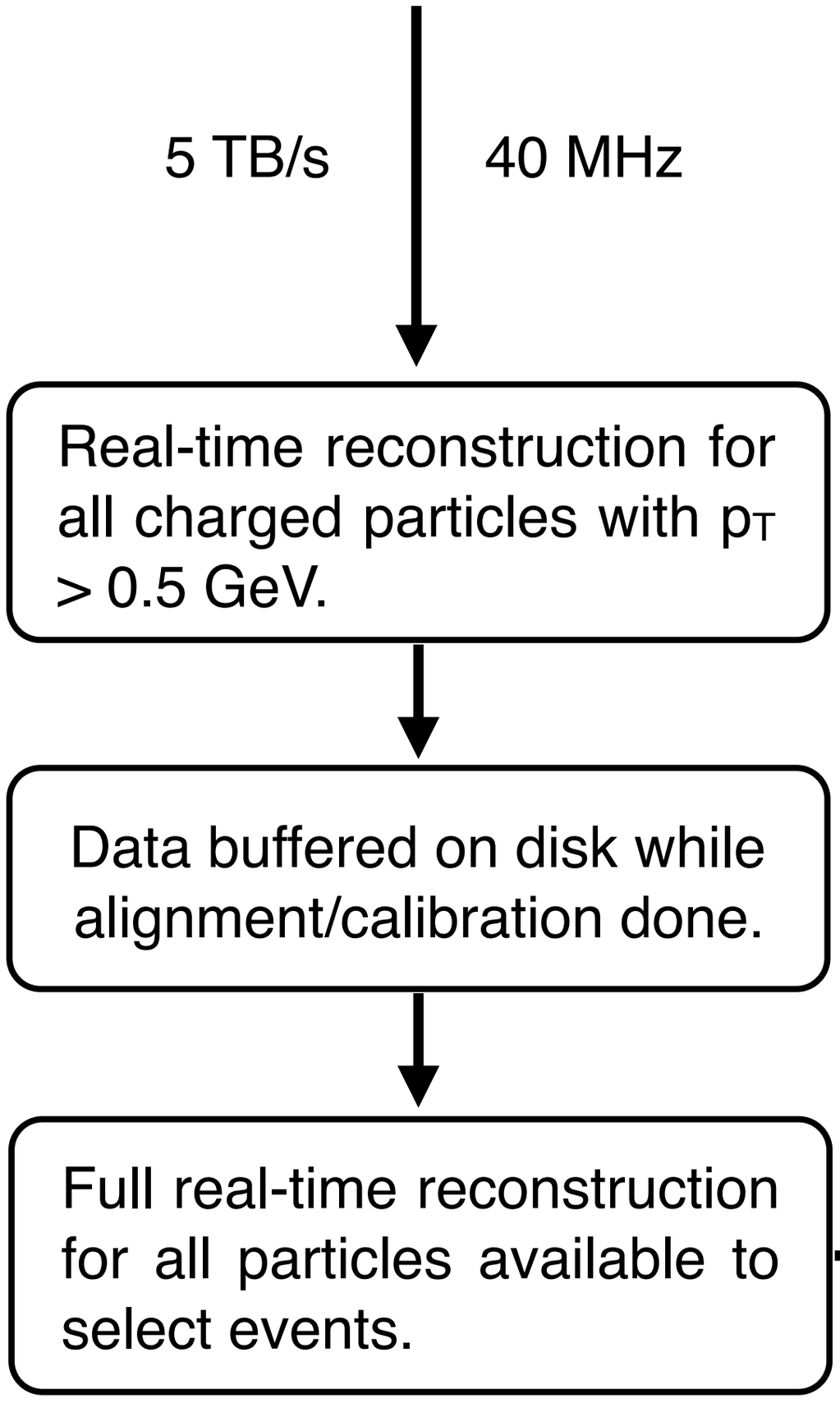
40% of output events here classified using ML algorithms.

ML also used online in tracking, particle ID, etc. (more on this later).

Full real-time reconstruction for all particles available to select events.

→ 5 PB/year (mix of full events & ones where only high-level info kept)

Real-Time Processing (Run 3)



Performing the charged-particle reconstruction on 5 TB/s of data in real time will be a challenge. Investigating ALL options here — use ML to speed it up? (Indeed, we already do some of this.)

Keeping the vast wealth of physics data will also be a challenge. Plan to migrate most of remaining classification to ML-based algorithms. Autoencoder-based data compression?

We are also working on ML-based anomaly detection.

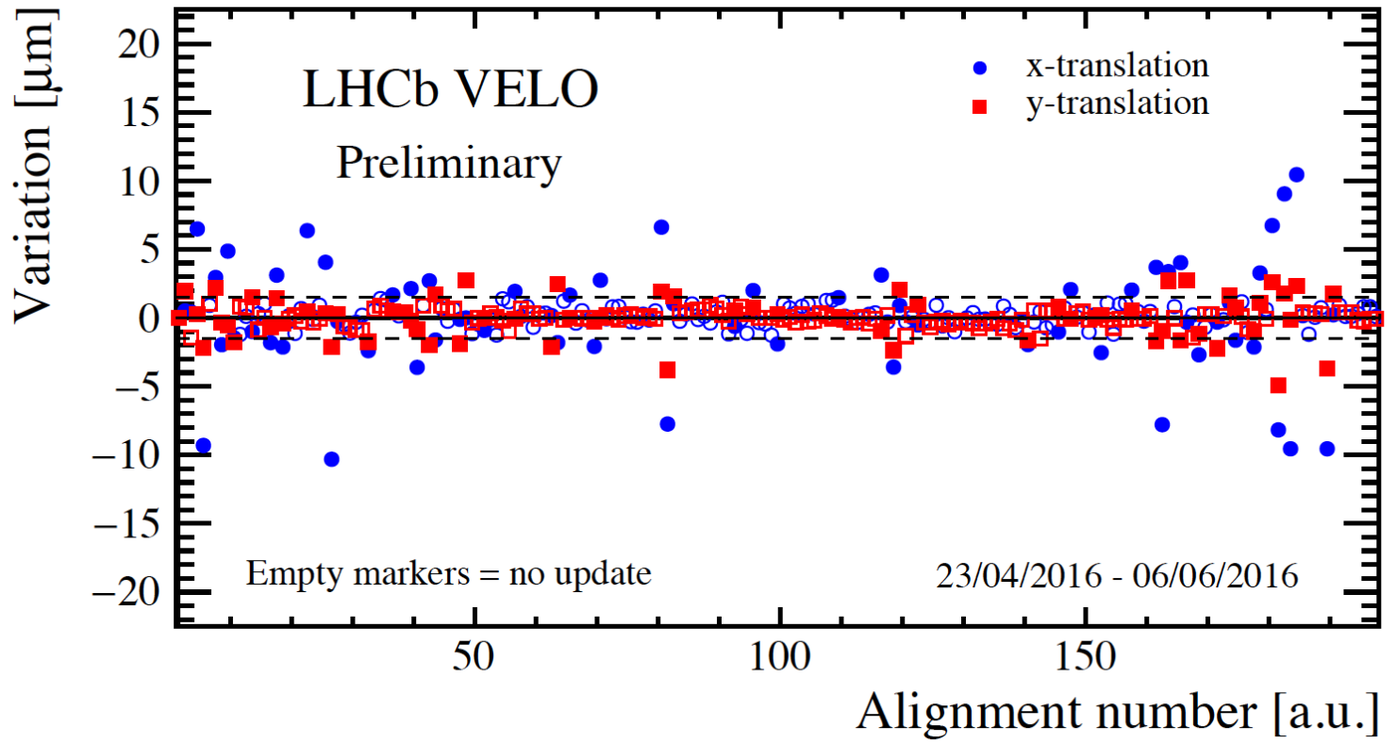
N.b., real-time alignment and calibration is NOT required to use ML in an online system.

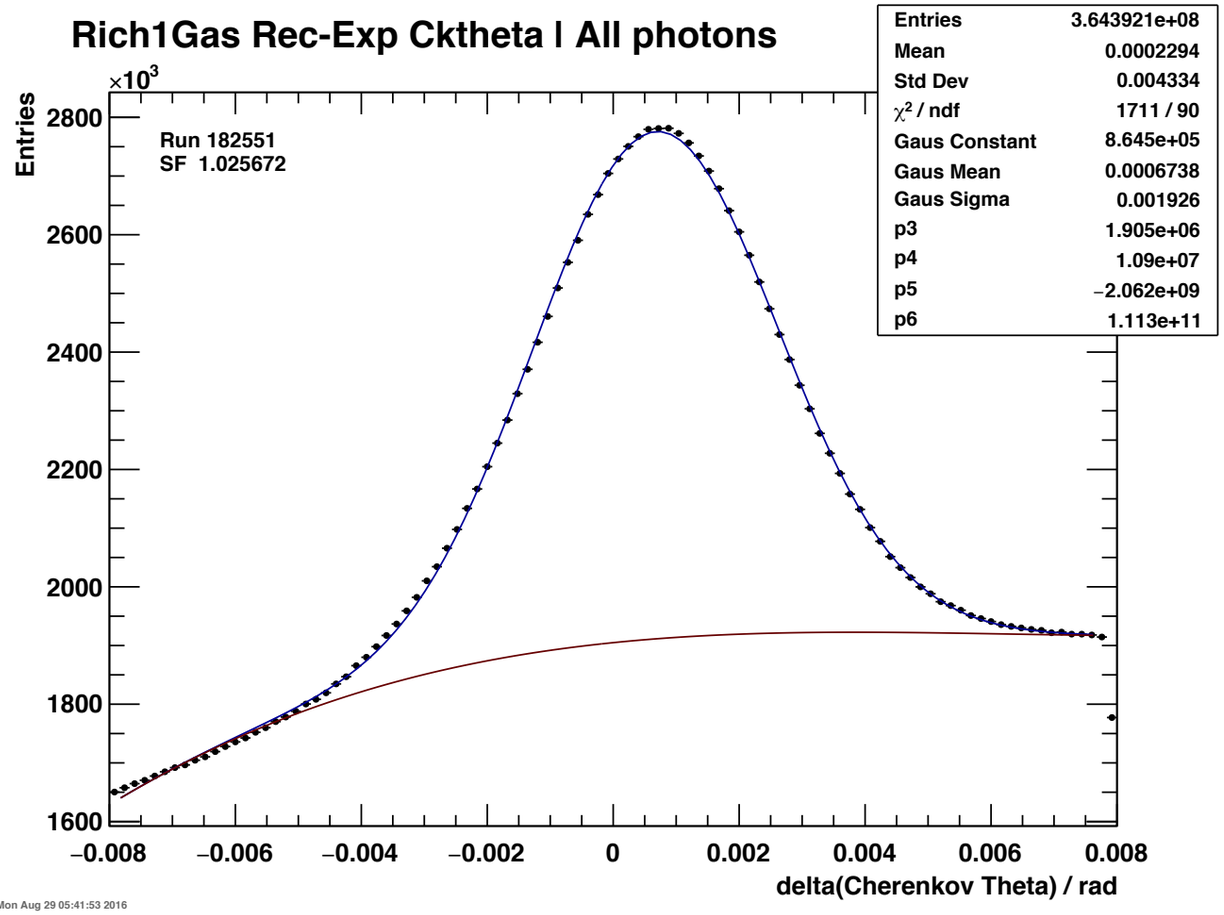
We first introduced ML into our primary event-classification algorithm at the start of 2011 data taking, but real-time calibrations were not implemented until 2015.

Our Run 1 ML-based trigger algorithm collected the data used in about 200 papers to date — and it was run on imperfect data (but designed to be robust against run-time instabilities).

Real-Time Calibration

VELO opens/closes every fill, expect updates every few fills. Rest of tracking stations only need updated every few weeks.





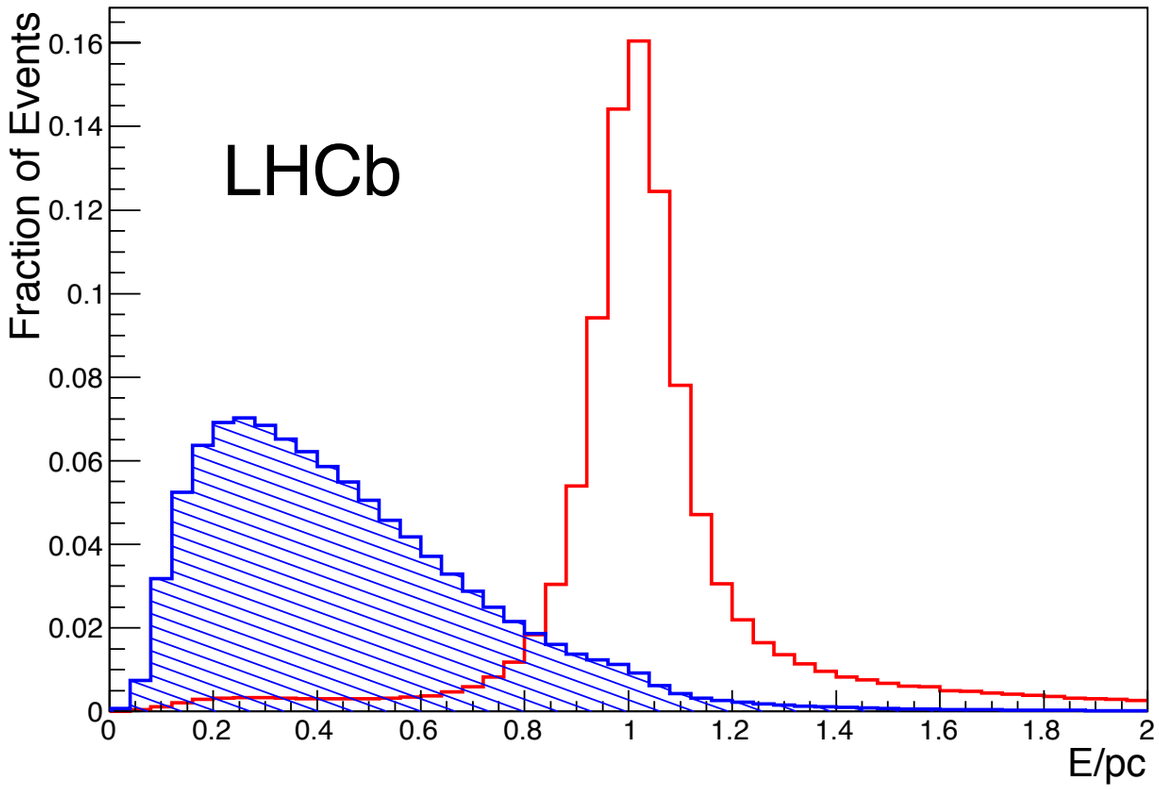
RICH gases indices of refraction must be calibrated in real time; requires ~1 min to run, and new calibrations are required for each run.

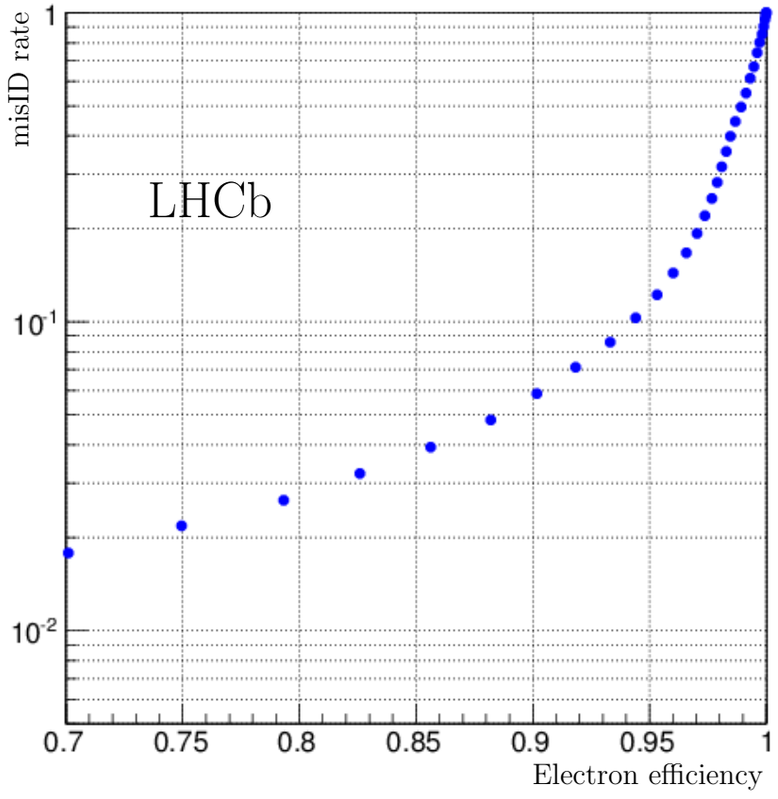
Calibration data is sent to a separate “stream” from the physics data after the first software-trigger stage. This permits running the calibrations on the online farm simultaneously with running the trigger.

Calorimeters

The primary use of the calorimeters for charged PID is in identifying electrons.

$$\Delta\log\mathcal{L}^{\mathrm{CALO}}(e-h) = \Delta\log\mathcal{L}^{\mathrm{ECAL}}(e-h) + \Delta\log\mathcal{L}^{\mathrm{HCAL}}(e-h) + \Delta\log\mathcal{L}^{\mathrm{PS}}(e-h)$$

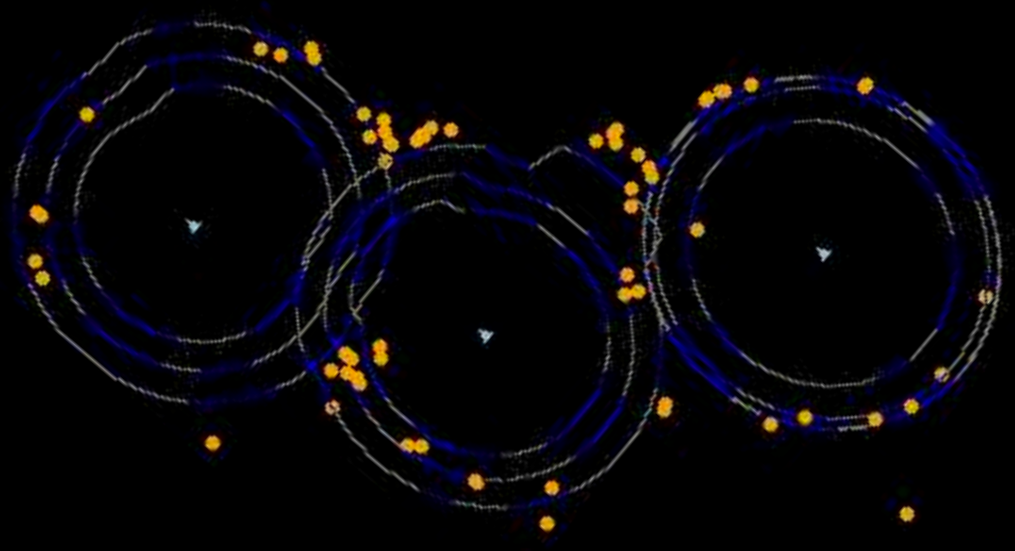


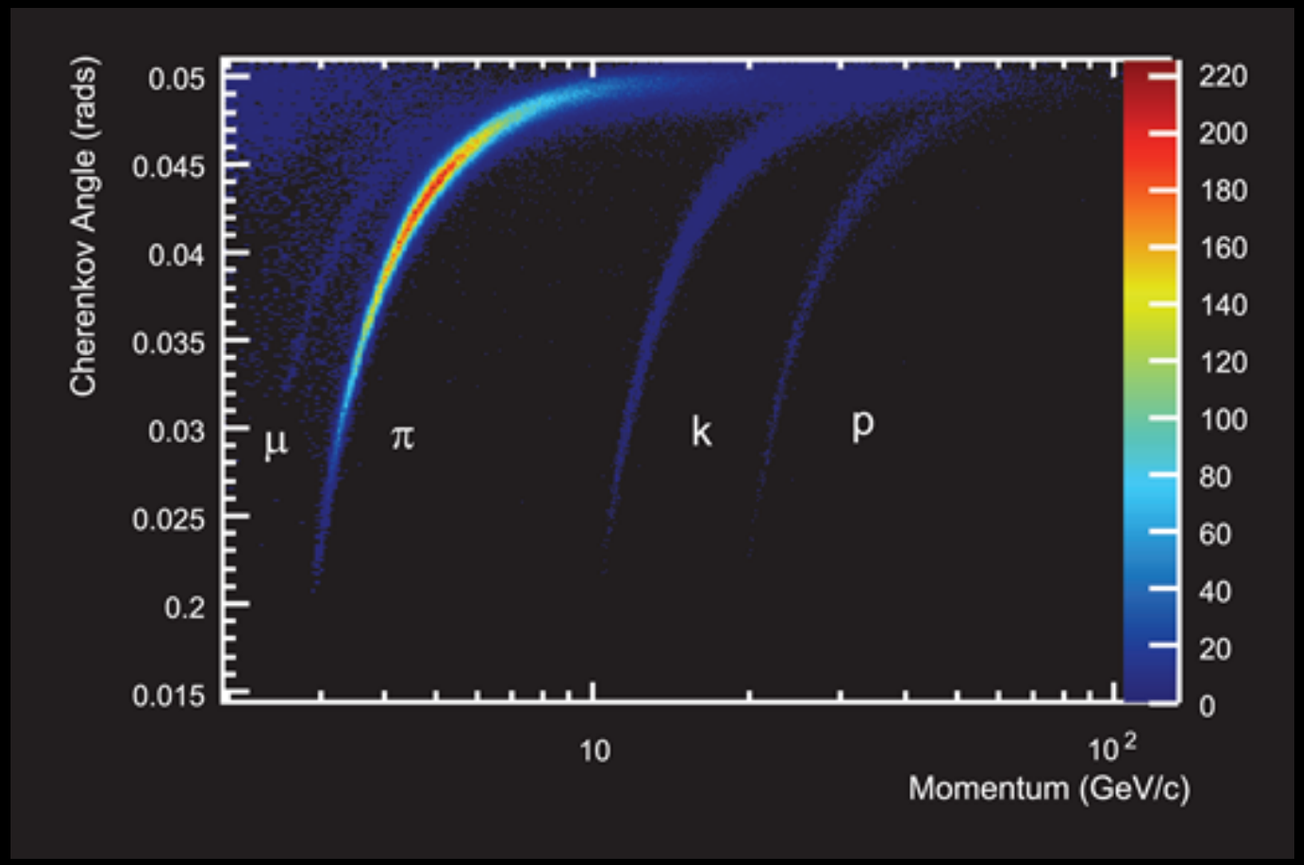


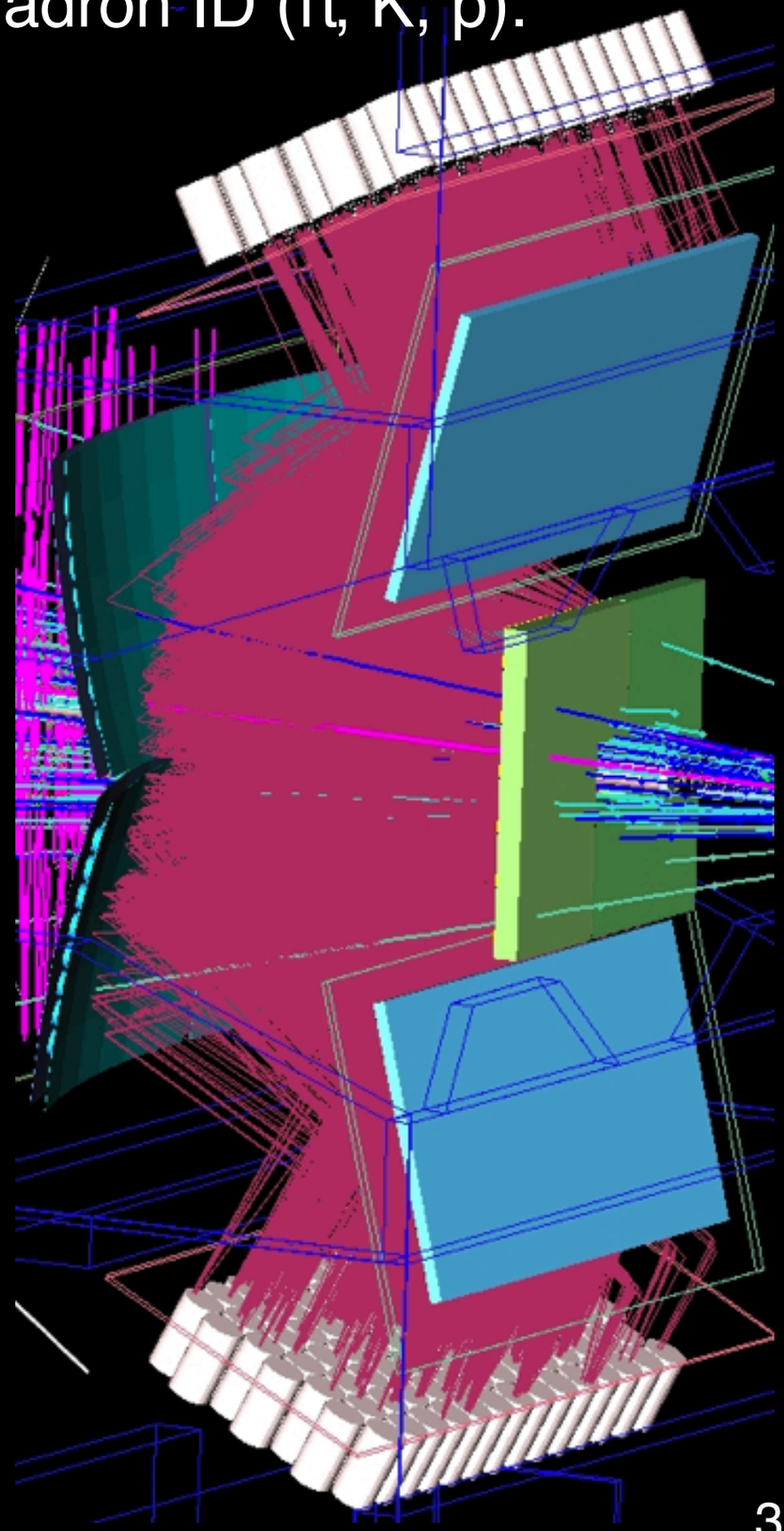
Using electrons from photon conversions and hadrons from D^0 decays, e and h PDFs are constructed from data vs track 3 momentum.

RICHs

The primary role of the RICHs is charged-hadron ID (π, K, p).

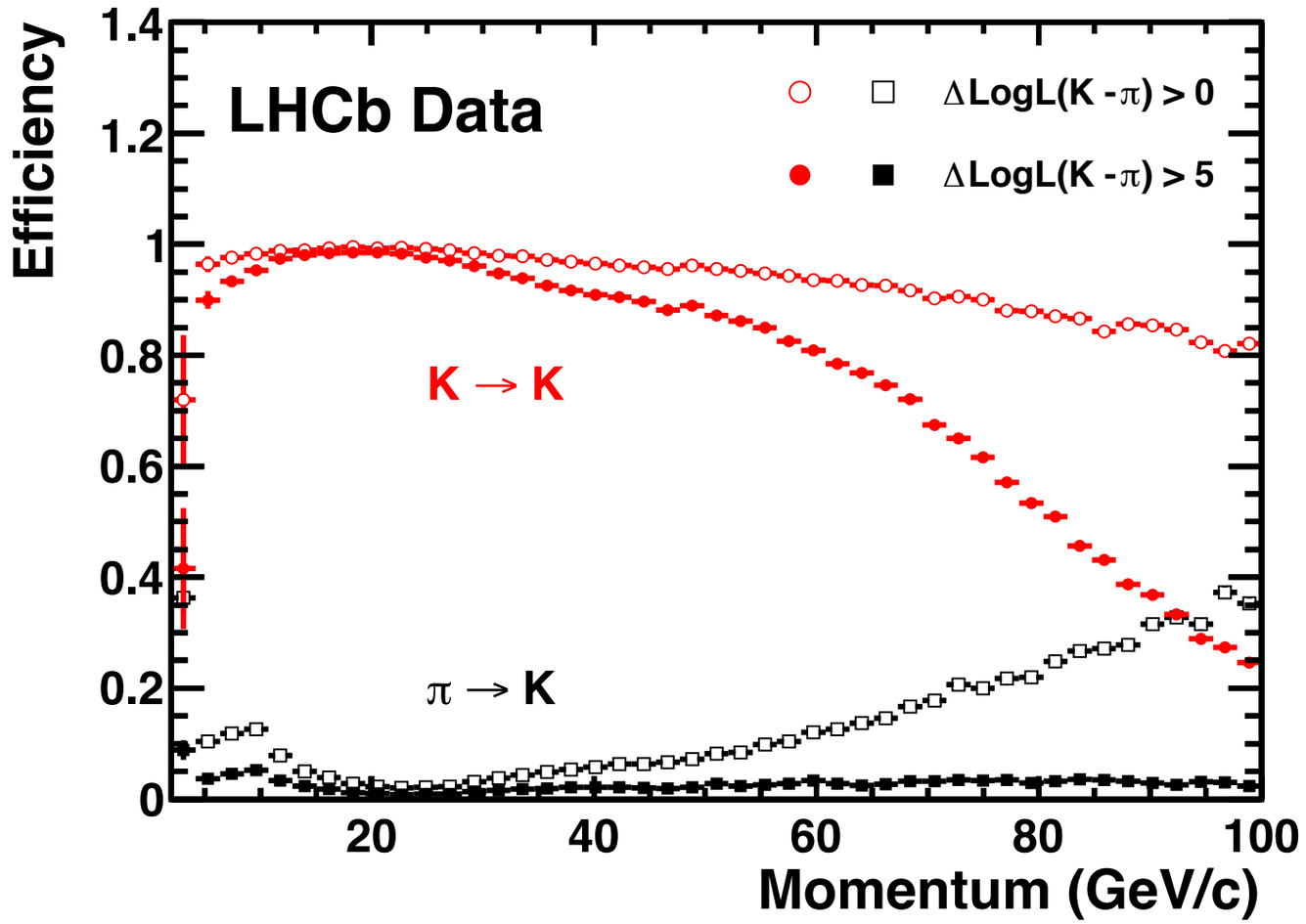






RICHs

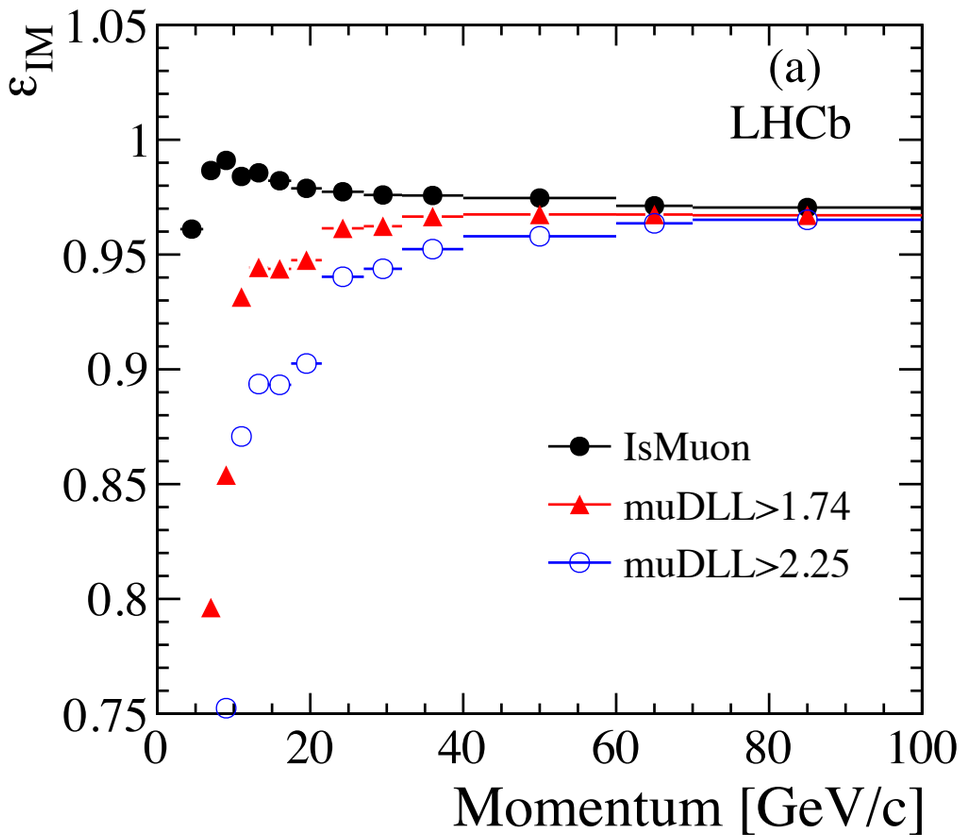
The primary role of the RICHs is charged-hadron ID (π, K, p).

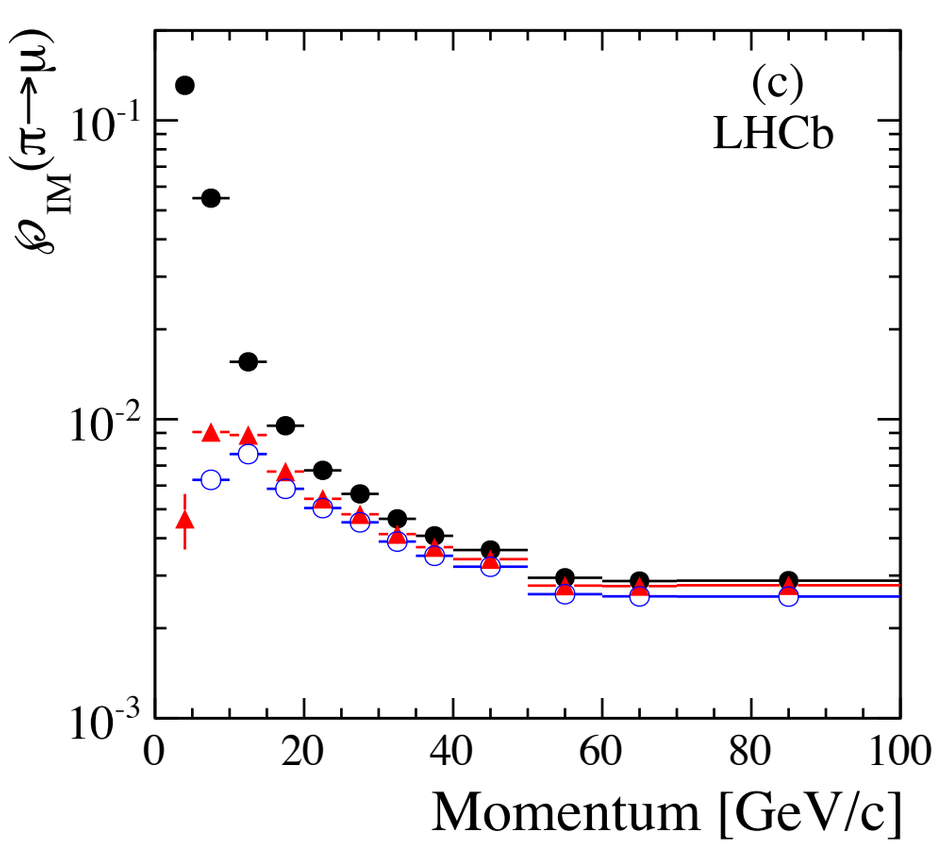


Calculate the likelihood of each RICH ring pattern observed under various PID hypotheses, then use "DLL" to arbitrate (calibrate/validate using $K_S \rightarrow \pi\pi$, $\Lambda \rightarrow p\pi$, and $D^0 \rightarrow K\pi$ data samples).

Muon System

Muons are identified by looking for hits in the muon system, which is shielded by both the ECAL, HCAL, and whose stations are interleaved with iron absorbers.

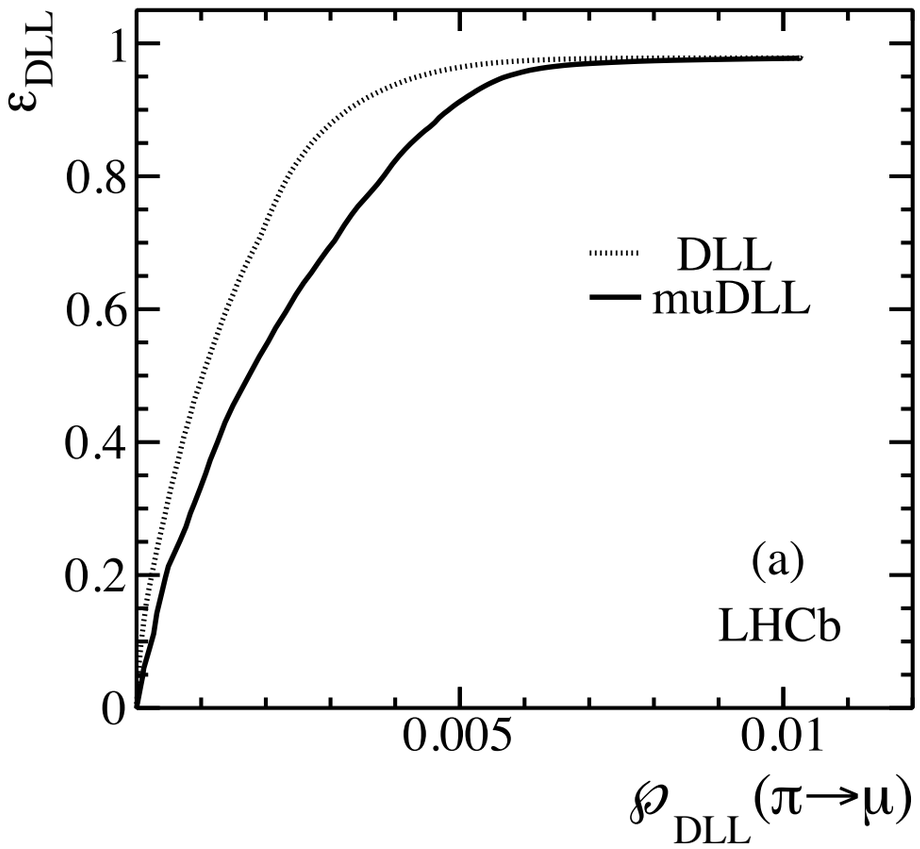


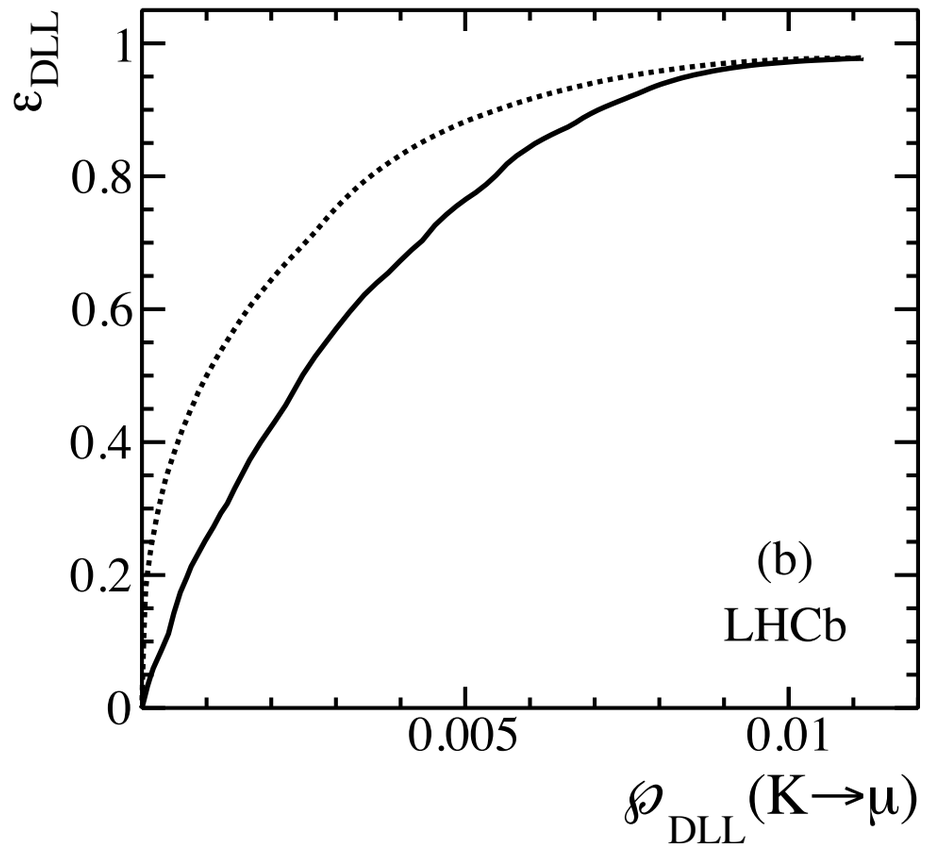


MisID from π,K→μ in flight, shared hits with a real muon, punch through, etc.

Combined DLLs

By combining the likelihoods from the RICHs, calorimeter system, and the muon system, LHCb obtains even better PID performance.





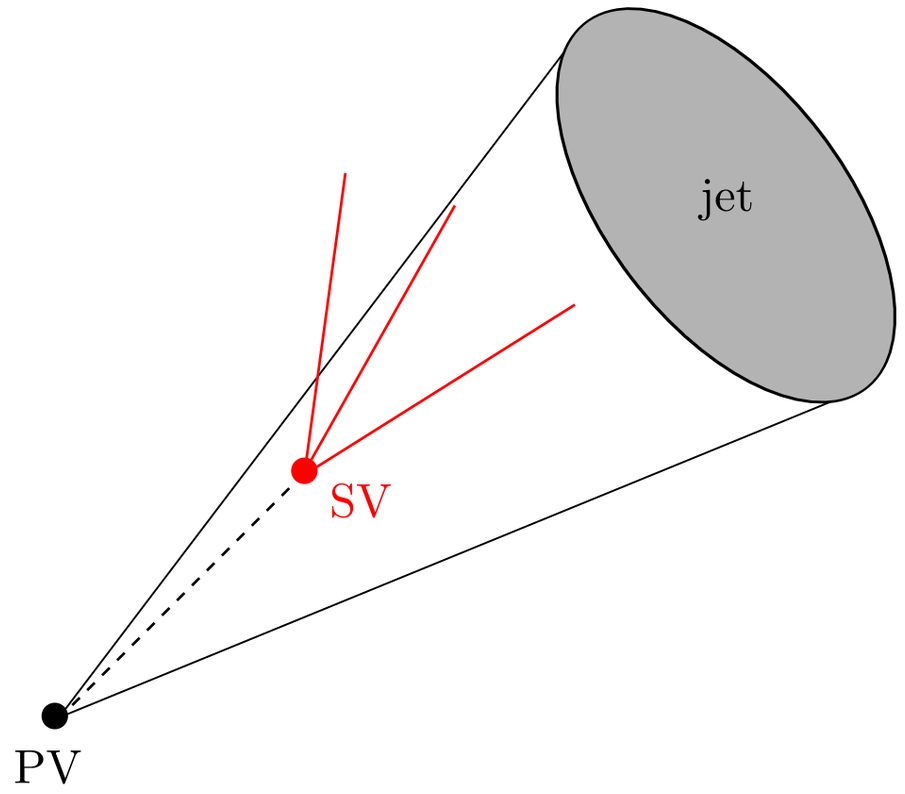
Consider the common case of K→μ decay in flight. If it was still a kaon when it passed through the RICH, then the RICH likelihood will show this.

E.g., CombDLL reduces the B→hh misID rate by a factor of 6 for a loss of only 3% of B_s→μμ signal.

SV Tagging

JINST 10 (2015) P06013
LHCb-PAPER-2015-016

Look for an SV “in” the jet (direction of flight in the jet cone). This occurs about 70%, 25%, 1% of the time for b, c, light jets. Next, use SV features to discriminate:



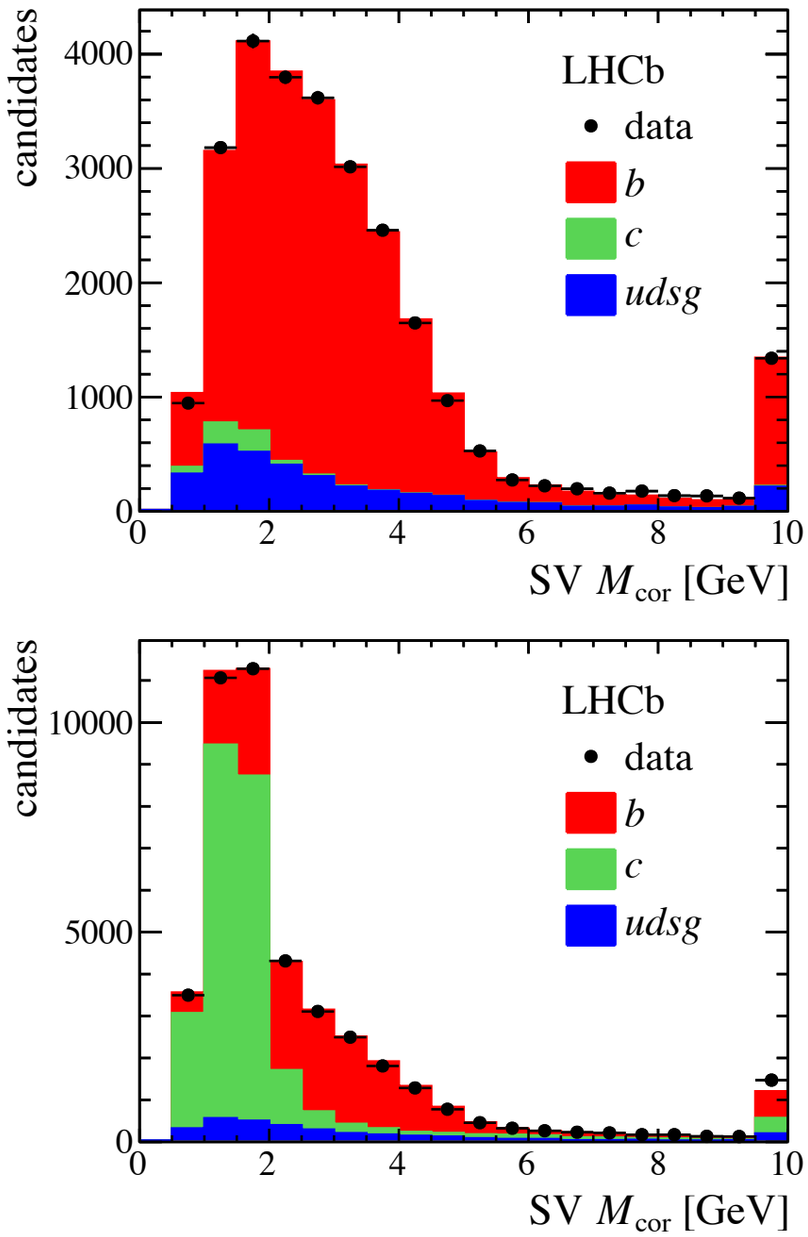
- mass and “corrected” mass;
- transverse distance from and flight distance x^2 of PV to SV;
- $p_T(SV)/p_T(jet)$ and $\Delta R(SV,jet)$;
- number of tracks in SV, number not in the jet, and sum of IP x^2 of all SV tracks;
- net charge of the SV.

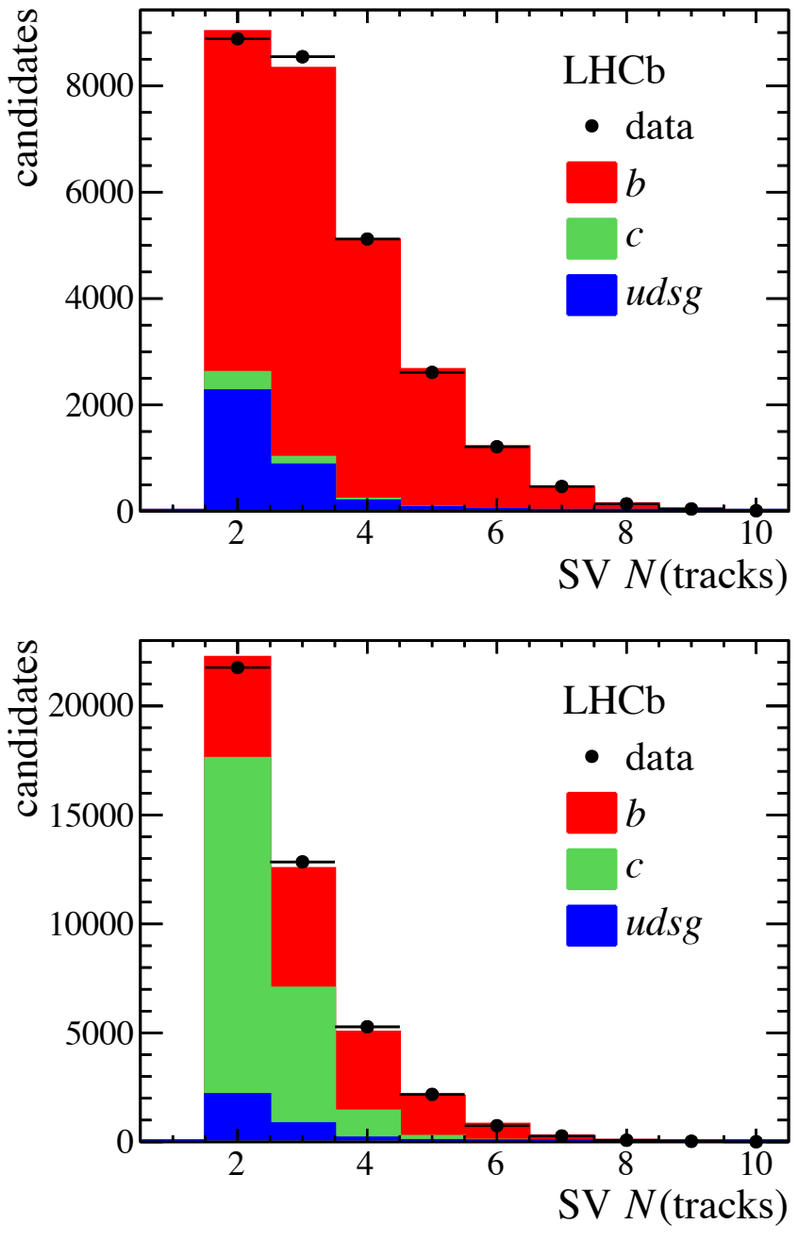
Each feature provides some discrimination power, but typically only between b,c vs light or b vs c -- none are powerful enough to fully separate types.

SV Features

JINST 10 (2015) P06013
LHCb-PAPER-2015-016

Comparison of 2 of the best features in data and simulation:





B+jet

D+jet

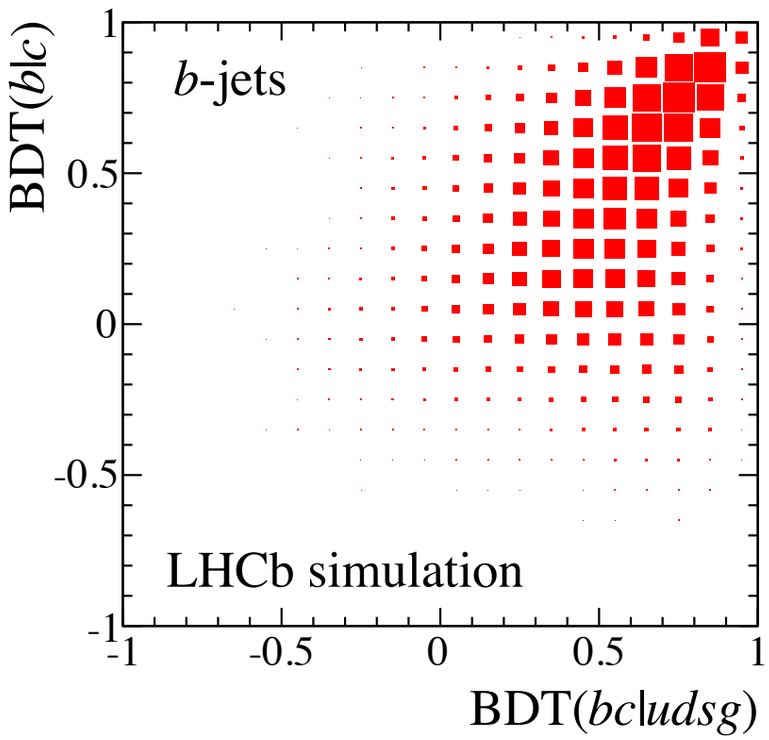
Any cut we would make here would either be inefficient or lack purity.

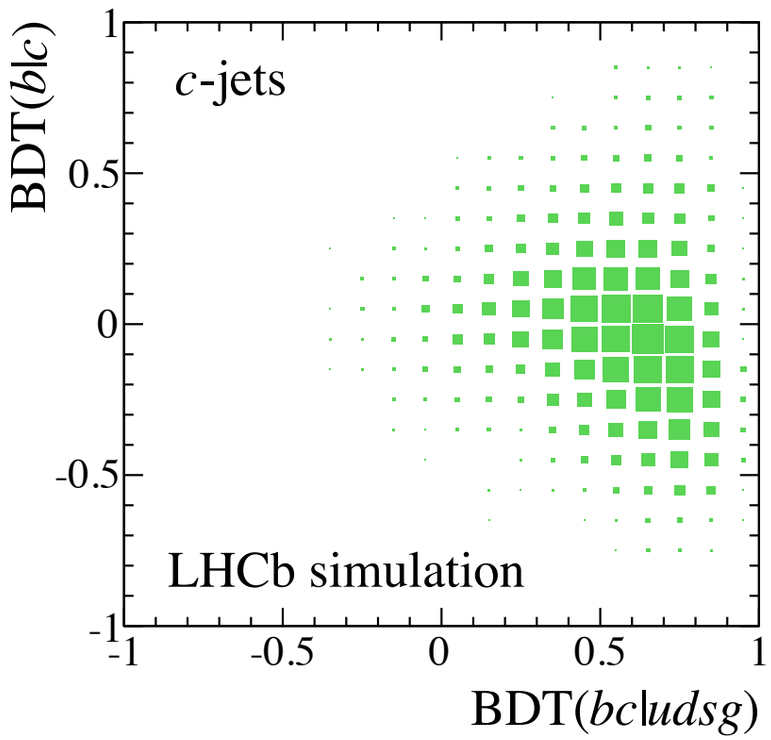
ML SV Tagging

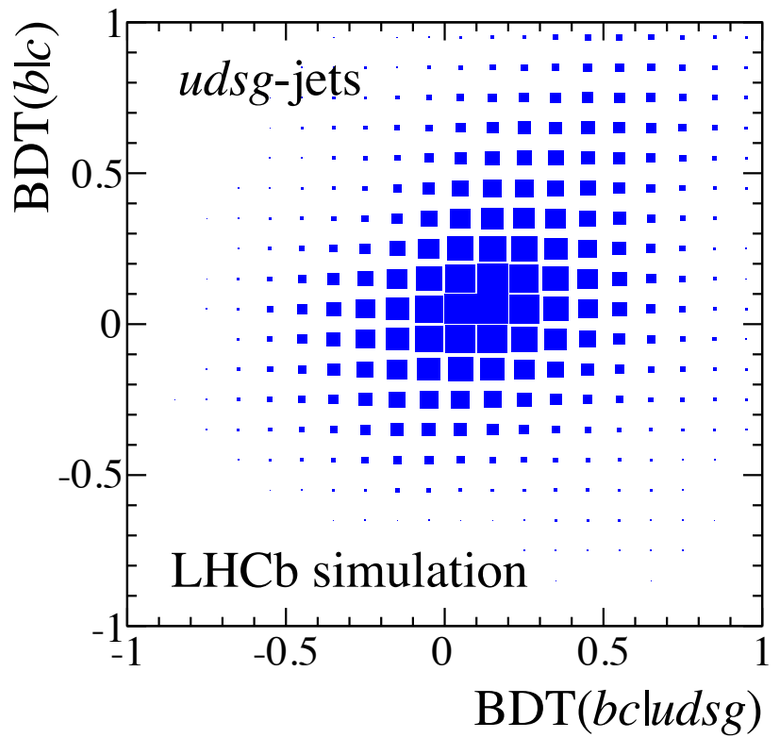
JINST 10 (2015) P06013
LHCb-PAPER-2015-016

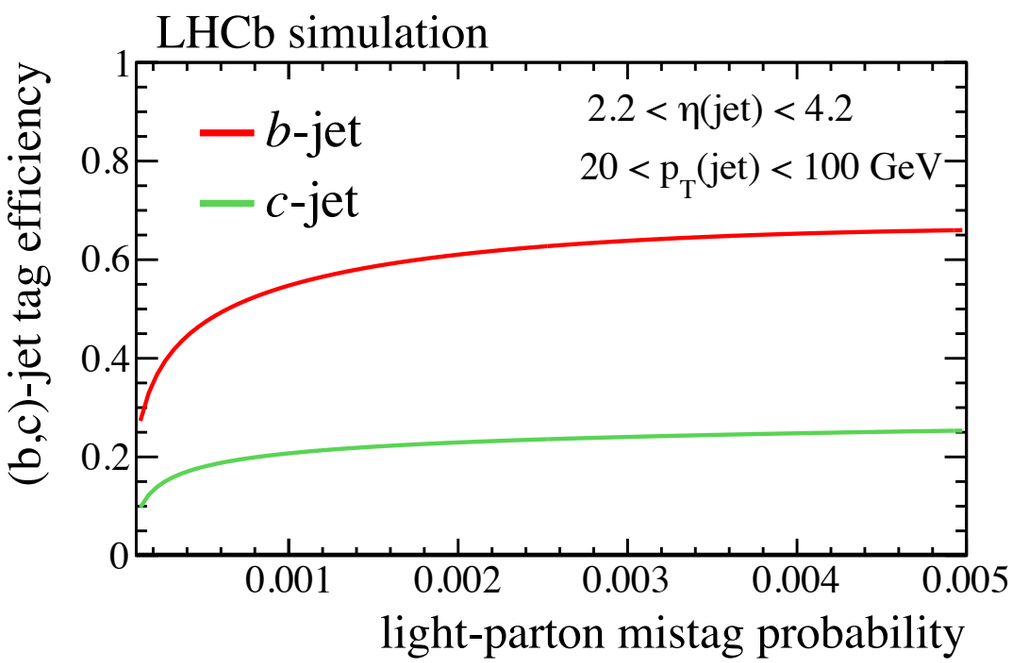
Put 10 features into two BDTs: one for b,c vs light, and another for b vs c.

LHCb simulation: each distribution normalized to one; recall that 70%, 25%, 1% of b, c, light jets have an SV.







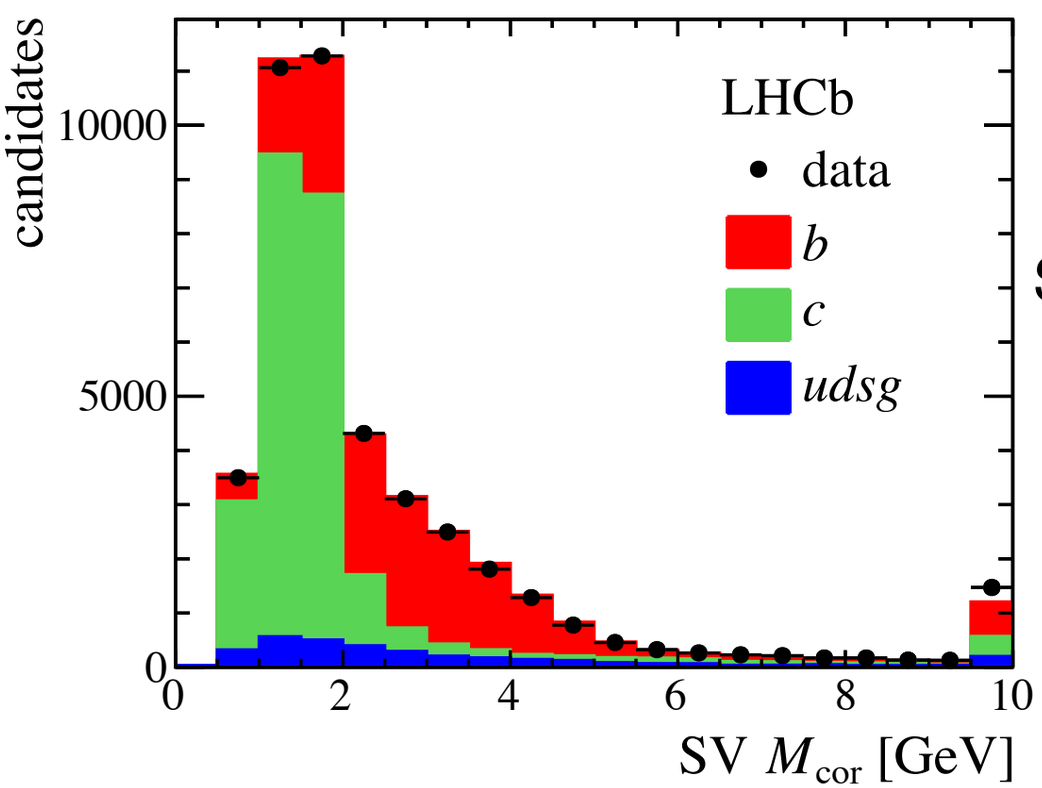


Could cut on both BDT responses and obtain high-purity b-jet or c-jet samples. Alternatively, can fit the 2-D BDT distribution in data to extract the b-jet and c-jet yields.

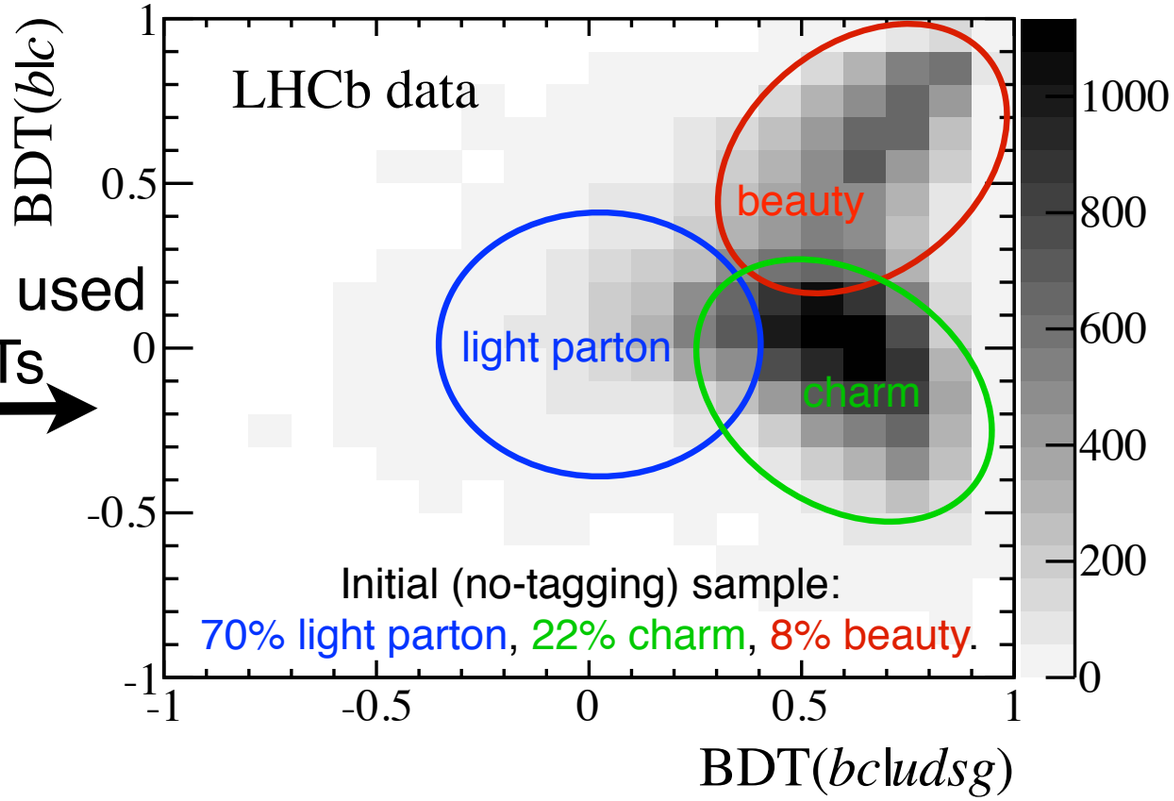
2-D BDT Fits

JINST 10 (2015) P06013
LHCb-PAPER-2015-016

The 2-D BDT plane optimally utilizes all info that can separate (x-axis) blc vs light and (y-axis) b vs c.



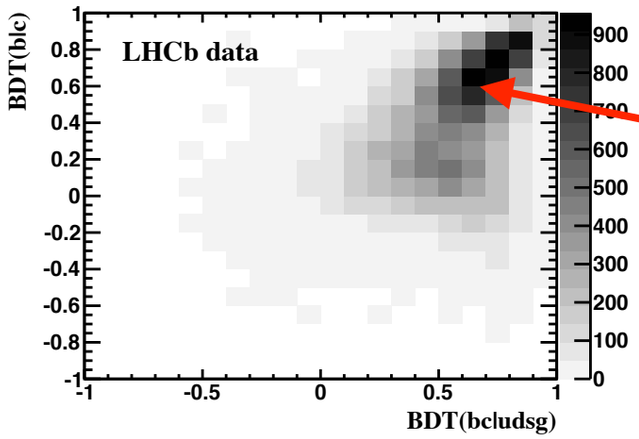
SV features used in 2 BDTs

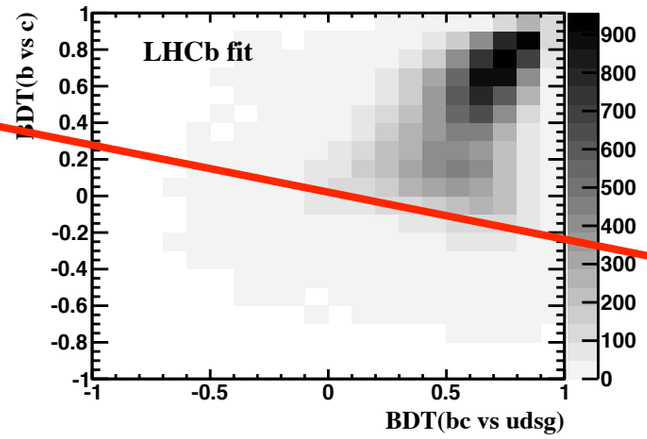


Performance validated & calibrated using large heavy-flavor-enriched jet data samples. Two-D BDT distributions fitted to extract SV-tagged jet flavor content; c-jet and b-jet yields each precisely determined simultaneously.

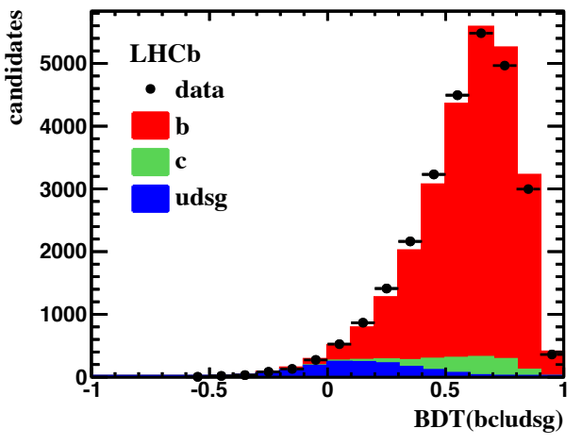
Calibration Data

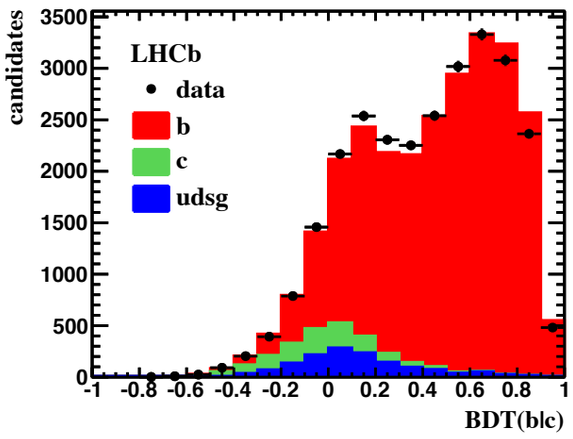
JINST 10 (2015) P06013
LHCb-PAPER-2015-016

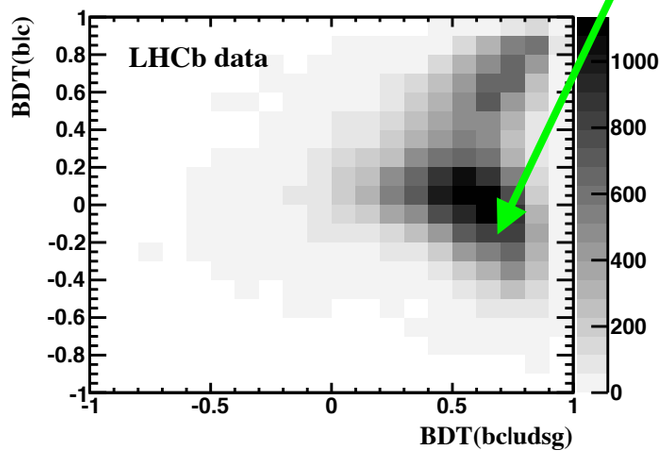


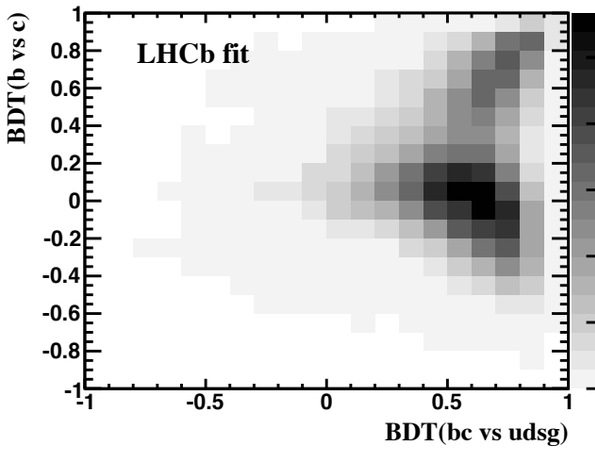


Can clearly see b and c jet bands in the 2-D BDT plots (light is blob near origin).









MC templates are pretty good “out of the box”, but this data was used to calibrate them even better.

Both b-jet and c-jet yields determined precisely from these fits.