Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

ML@ATLAS & CMS: setting the stage

Outline

Give an idea of how we use and tweak ML algorithms in HEP through examples

Typical use is a mixture of ML technology and physics insight

- The role of Monte Carlo (MC) simulations
- Data / MC disagreements and systematic uncertainties
- Detailed examples:
  - Energy regression
  - Event classification
- Flash examples:
  - Pixel clustering
  - Track quality classification
  - Data placement
- Final remarks

Some ATLS/CMS ML applications

A non-exhaustive list of machine learning applications in ATLAS/CMS:

- Pattern recognition: clustering hits
- Tracks classification: duplicate removal, quality selection,…
- Energy / momentum regressions: photons, electrons, (b-)jets,…
- Objects identification: select electron, b/c-jet,… form (typically jets) background
- Entire event classification: separate signal from background(s) events
    - Fisher discriminant, Likelihoods, Neural Networks, BDT, 1D/2D fit MVA outputs
    - Mixed use: look at the classifier output to design simpler selections
- Data quality monitoring - outliers rejection
- Data placement: predict which samples will become hot

The vast majority of these application moved from "cut-based" solutions to supervised learning techniques (unsupervised learning at present not used)

Monte Carlo & Systematics uncertainties

Monte Carlo Simulations

Labeled samples are usually (~always) obtained from Monte Carlo Simulations:

- sometimes background candidates are taken from data in *signal-free* control regions

Monte Carlo datasets are CPU-expensive which means we typically have limited statistics:

- signals O(10^5 - 10^6) events
- backgrounds O(10^6 - 10^7) events

Full / Fast simulations = higher / lower description accuracy = More/Less CPU Expensive
Fast simulations are obtained using simplified descriptions of the detectors

The vast majority of our datasets are in Full Simulation.

Fast simulations mostly used for “parameter scans”:
A theory/model predicts the existence of a signal and is governed by a few parameters.
Scan the parameters space and for each point produce a signal sample

Cannot use fast simulations for large samples in one point:
hit the limits of the simplified description

Ref. http://arxiv.org/abs/1005.4568

Data/MC as Systematic uncertainties

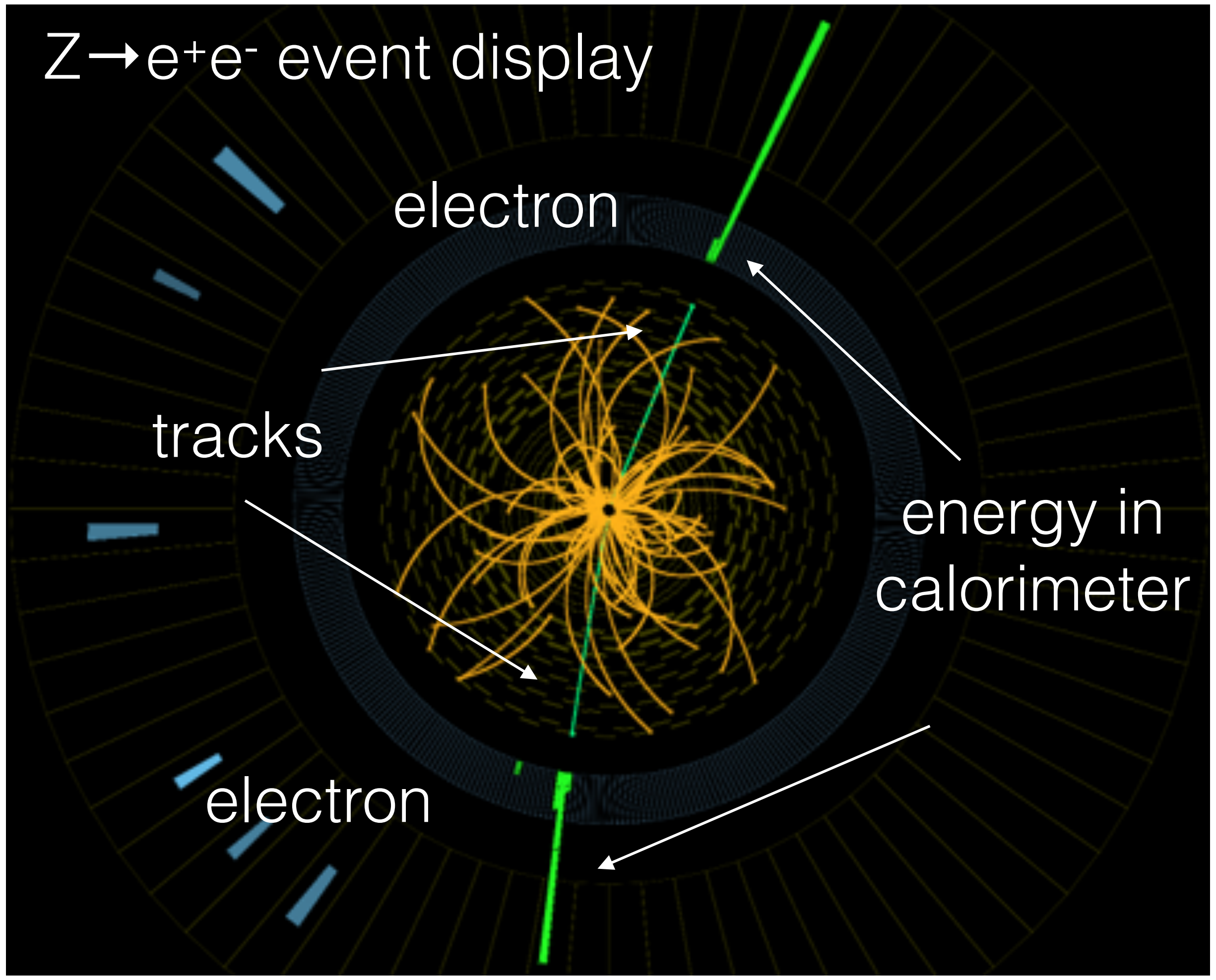
At each step in the MC chain (see V. Innocente) you can have mis-modelings leading to systematic deviations/uncertainties between simulations and data.

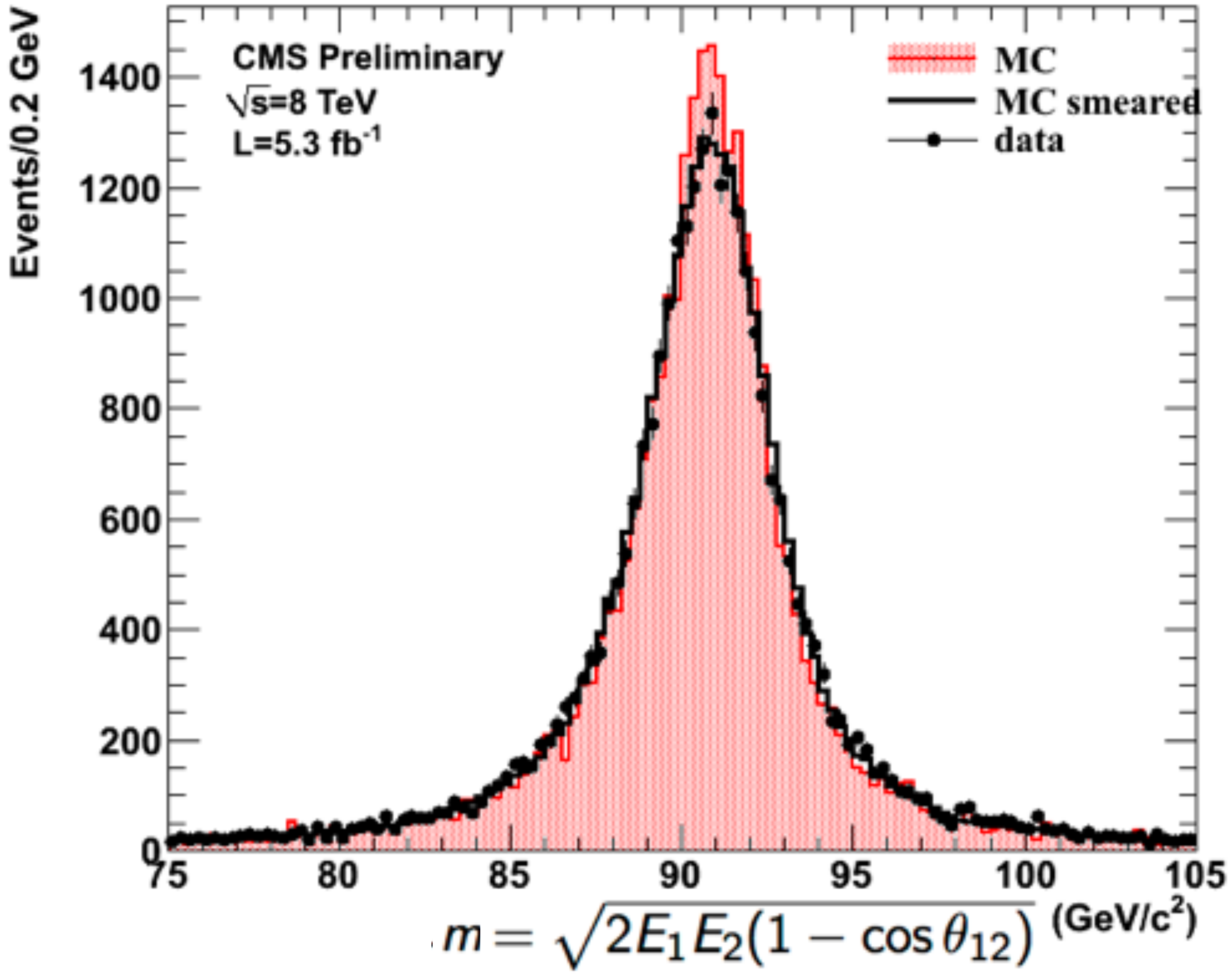
Data/MC disagreement:
- generators: physics process (see T. Golling)
- simulations: detector response

Simulations are compared to data using control regions or “standard candle” signals: data/ MC disagreements are propagated to the final measurement’s systematic uncertainty

HEP physicist's best friend

The Z-boson has been studies with high precision in previous experiments (LEP).
Now Z-boson is used as standard candles to calibrate the detector.





$$m = \sqrt{2E_1E_2(1-\cos\theta_{12})}$$

Set the energy scale of the detector: scale the energy/momentum of the electrons to match the Z mass position in data to the one in MC. Smear the MC to match the Data resolution.

(Same idea applies to Z→μμ, Z→bb, etc… to set μ, b-jets,etc… scale/resolution)

Residual differences are *typically very small* and are quoted as systematic uncertainties.

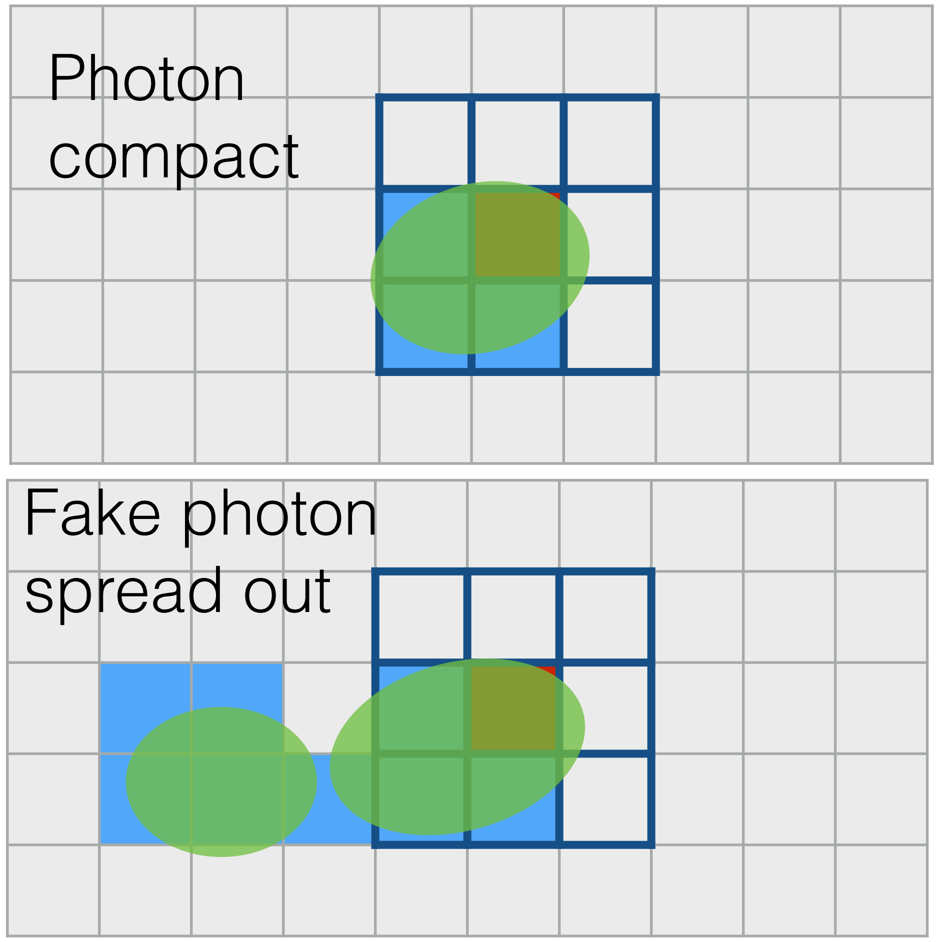
Ref. http://arxiv.org/abs/1502.02702

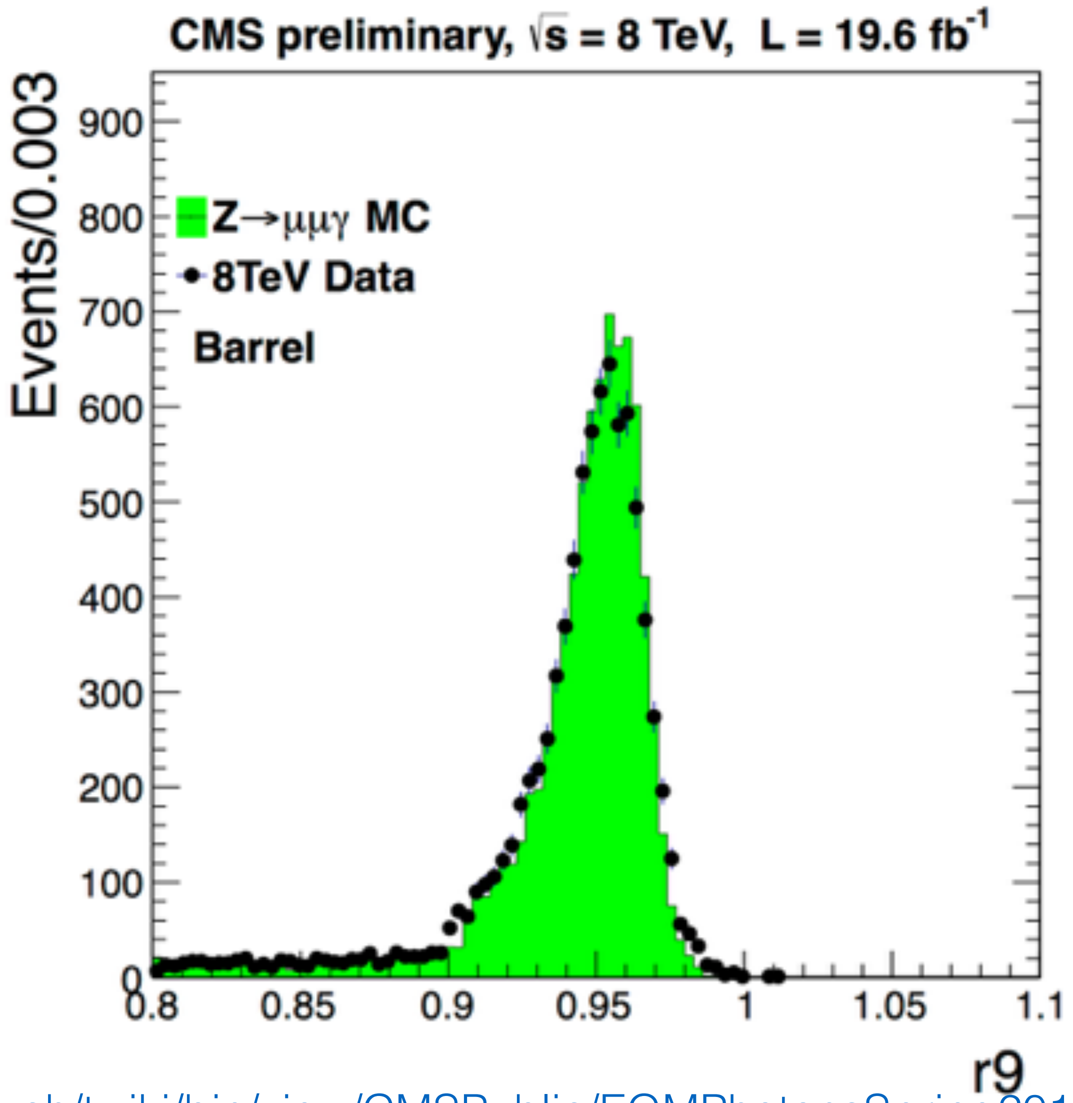
Systematic uncertainties on inputs

Systematics uncertainties typically lead to non-optimal classification/regression.
We know how to set a systematic on the input variables but don't have a standard recipe to assign systematics to BDT outputs.

Example: photon identification
BDT classifier to separate photons from fake photons i.e. jets ($\pi^0 \rightarrow \gamma\gamma$)
o(12) input variables, some of which are correlated, mostly describing the shape of the calorimeter cluster

Use physics driven features not full information

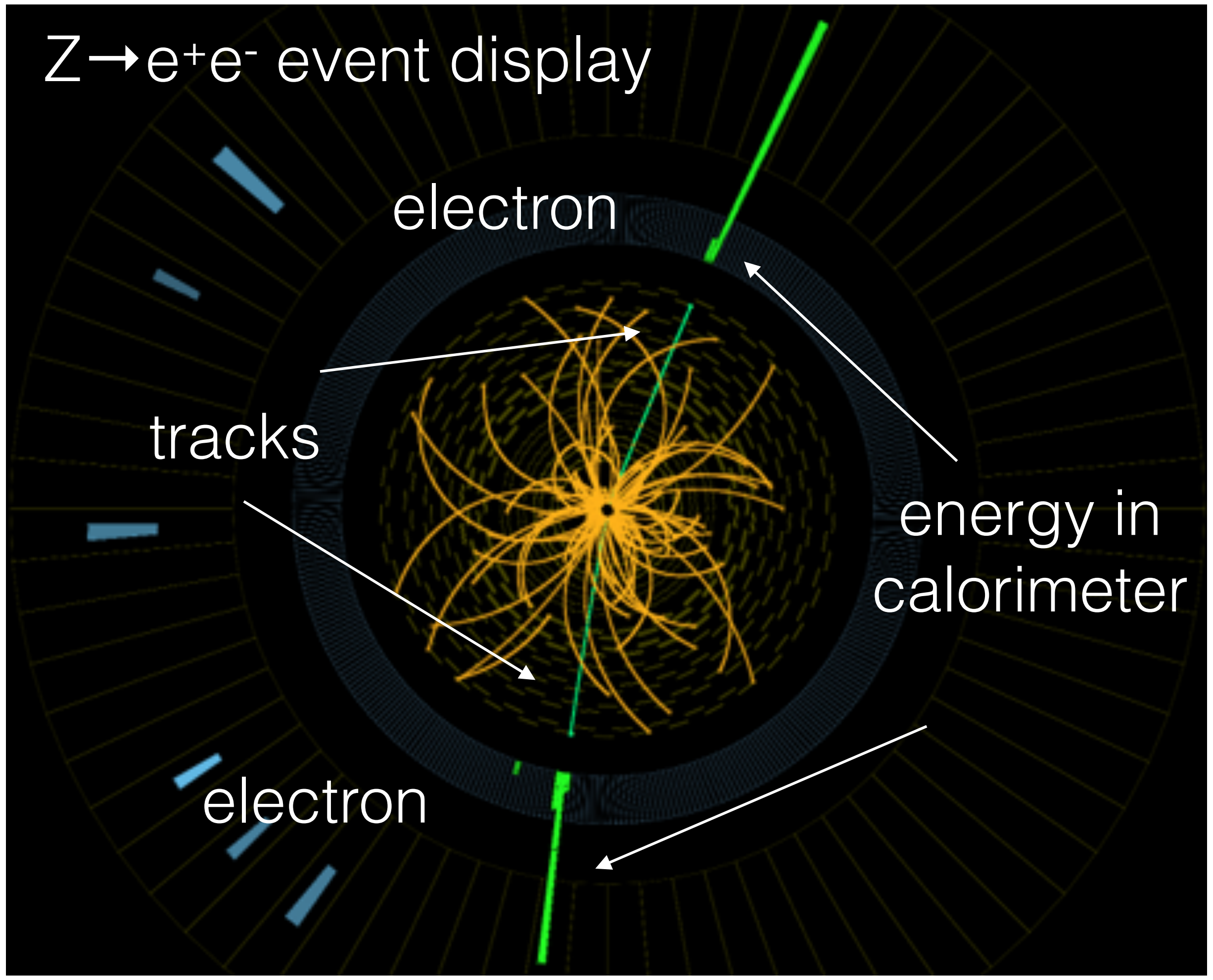


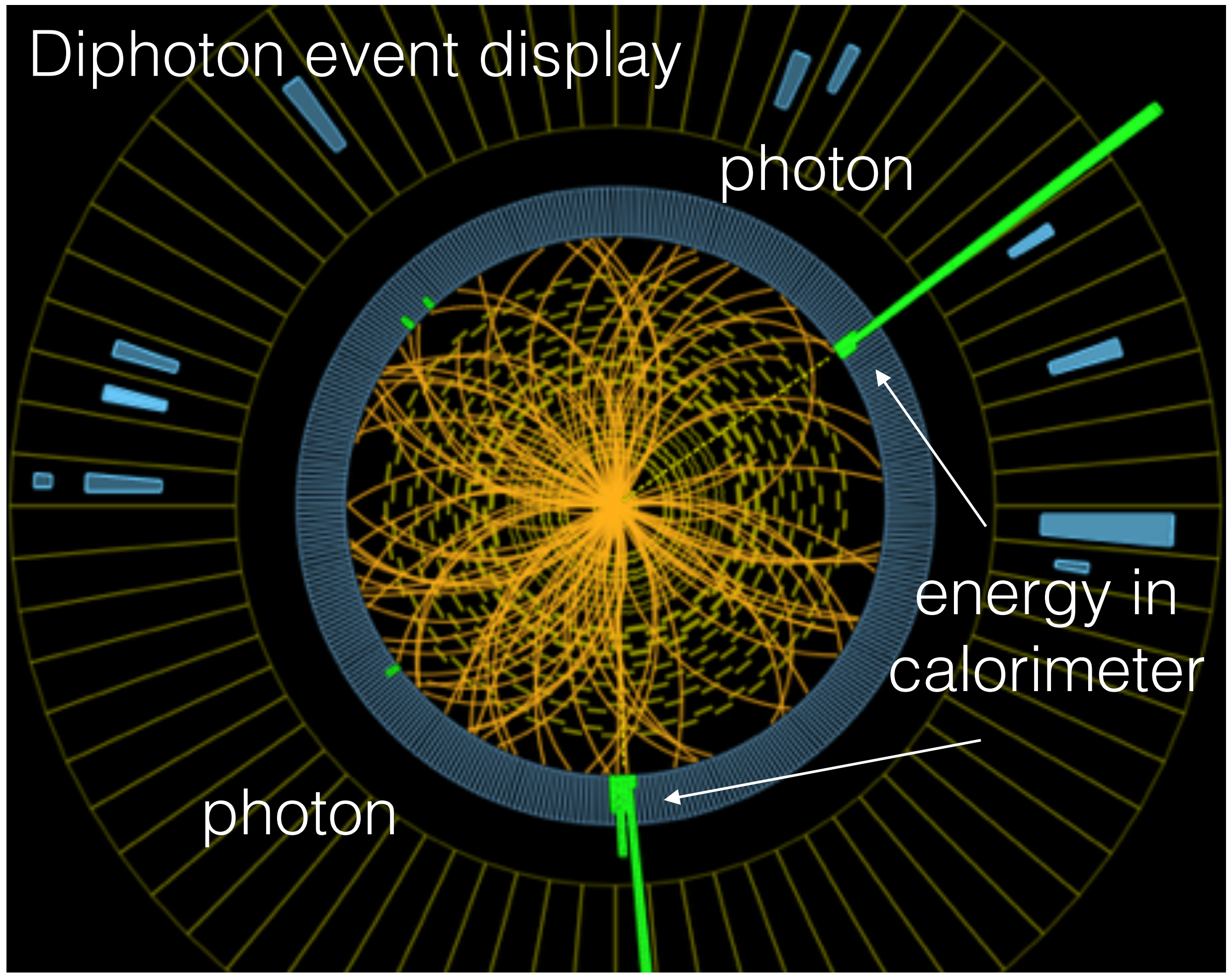


Intermezzo: no photon source

Nature decided not to give us a standard candle for photons (there is no $Z \rightarrow \gamma\gamma$)
Validation can be done on electrons reconstructed as photons using again the $Z \rightarrow e^+e^-$

electron = track + calorimeter cluster
photon = calorimeter cluster





Electrons

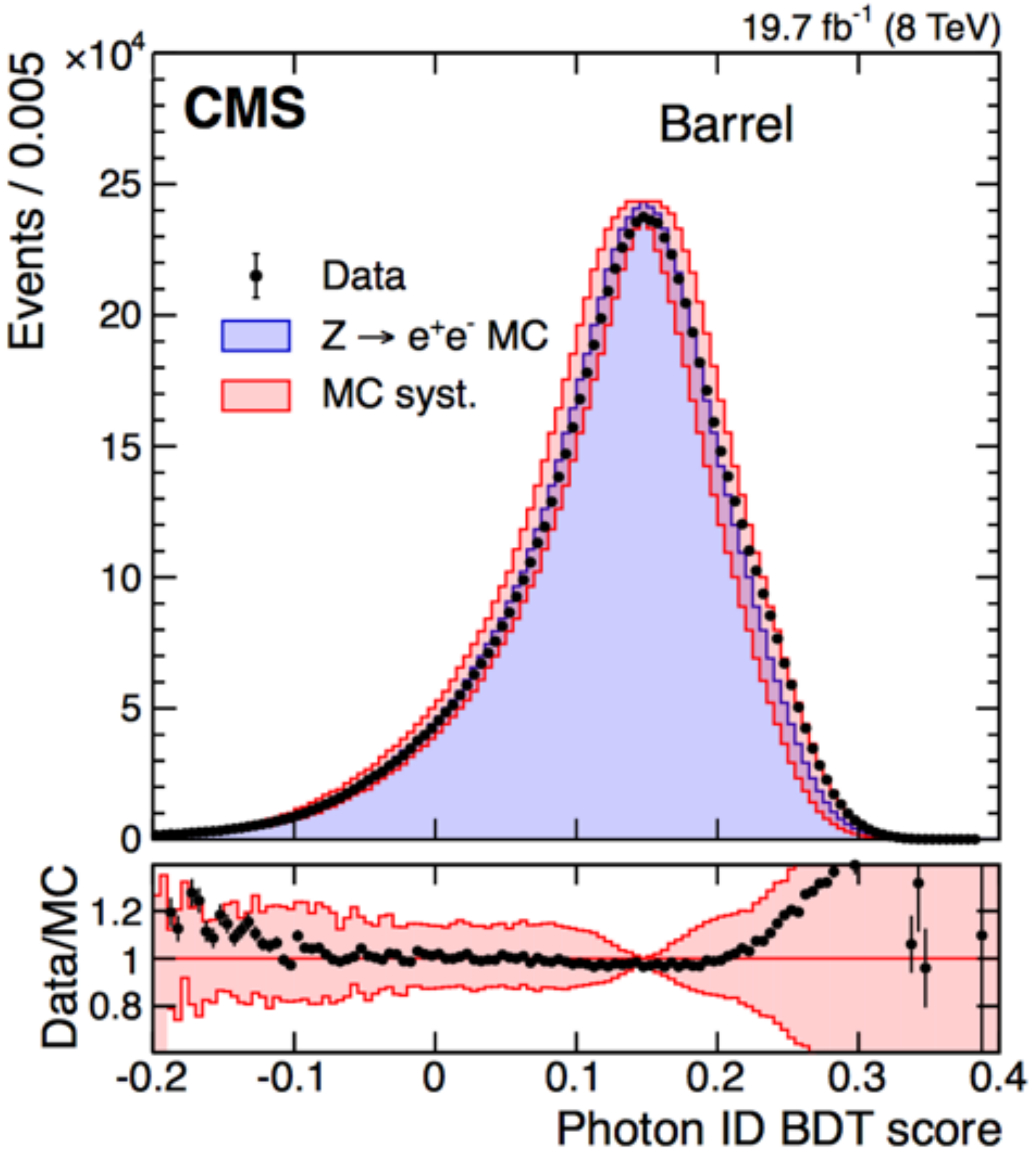
Photons

The same inputs variables can be used for electrons and photons

Systematic uncertainties on BDT

The systematic uncertainty on the Photon ID BDT is small and it is set as the envelop that covers the data/MC discrepancies:

BDT output shift ±0.01



Ref. http://arxiv.org/abs/1502.02702

Example: Photon Energy regression

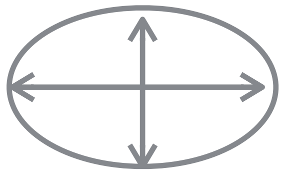
Photon Energy corrections

We measure E_{rec} we want the particle true energy E_{true} :

Correction = E_{rec}/E_{true}

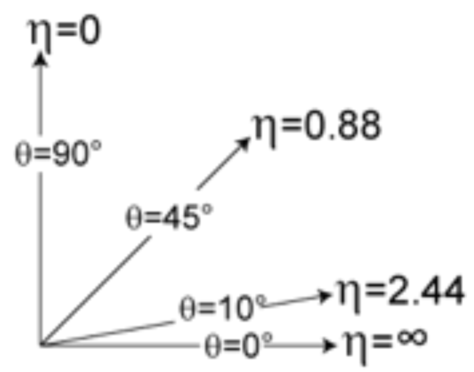
Initial strategy was to parametrise the correction based on 3 physics driven variables:

"photon shape" in the calorimeter coded in one variable ⇒ "shape" = σ_ϕ/σ_η



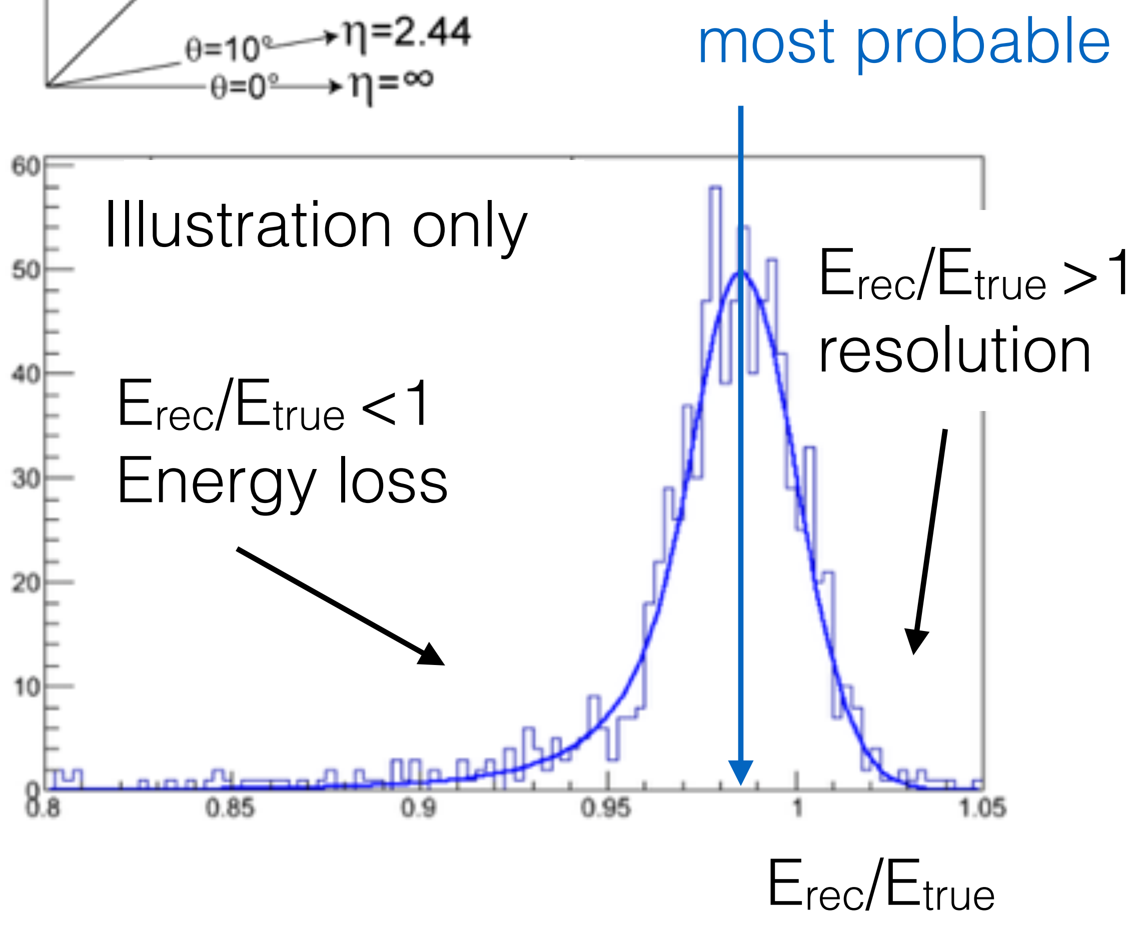
$F_1(\sigma_\phi/\sigma_\eta,\eta)F_2(E)$

The shape depends on the material the photon crosses ⇒ need η



The higher the energy the narrower the photon ⇒ need E_{rec}

MC: single photon gun (uniform energy) [3-300] GeV uniformly in the detector volume (η,ϕ)



Method:

Bin the space {shape, η, E_{rec}}

For all bins fit the most probable value of E_{rec}/E_{true}

Build a 3D lookup table that for each bin in {shape, η, E_{rec}} returns as correction the fitted most probably probable value.

We get one value per bin

Ref. http://arxiv.org/abs/1502.02702

Photon Energy regression

How to improve the corrections ? Add more variables in the description :

- difficult to model correctly the correlations
- curse of dimensionality

Move to a multivariate approach: BDT (Gradient Boosting)

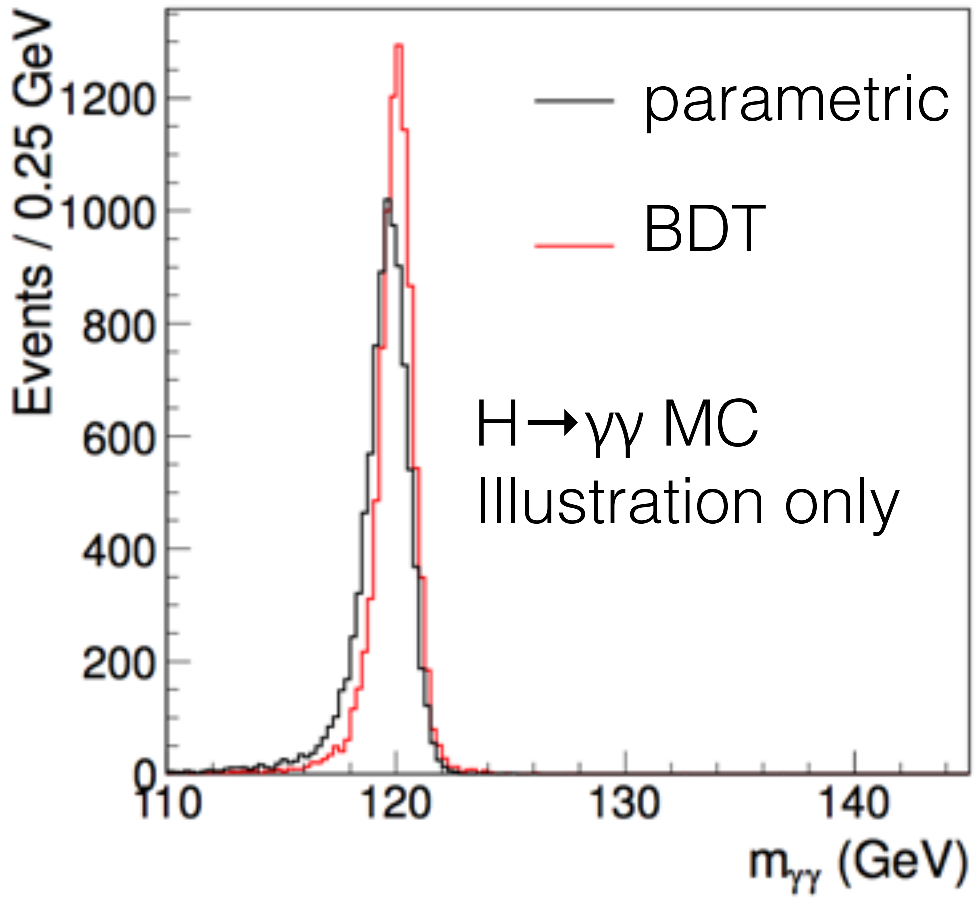
Use many more variables (first try O(80) then down to O(20))
correct treatment of the correlations by the BDT.

Basically add whatever variable makes sense to describe the photon

"photon shape" variables
photon coordinates (eta, phi)
median energy density ρ in the event

Training sample: again single particle gun MC (uniform energy spectrum [3-300] GeV and uniform in the detector volume (η,ϕ))

Target Variable: E_{rec}/E_{true}
10-30% improvement on resolution depending on the energies and region of the detector



Still we get one value per bin of the input space

Photon Energy regression final implementation

Intuition: for each of the leaves of the trees you get a constant number (i.e. f(x) is a piece-wise constant function over the input-variables space)

But from physics we know what is the shape of the E_{true}/E_{reco} in each leaf: Erec/Etrue can be modelled with a double Crystal Ball function.

$$f(z;n,\alpha,\mu,\sigma) = N \cdot \begin{cases} \left(\frac{n_l}{|\alpha_l|}\right)^{n_l} \cdot \exp\left(-\frac{|\alpha_l|^2}{2}\right) \cdot \left(\frac{n_l}{|\alpha_l|} - |\alpha_l| - \frac{z-\mu}{\sigma}\right)^{-n_l}, & \text{if } \frac{x-\mu}{\sigma} \leq -\alpha_l, \\ \left(\frac{n_r}{|\alpha_r|}\right)^{n_r} \cdot \exp\left(-\frac{|\alpha_r|^2}{2}\right) \cdot \left(\frac{n_r}{|\alpha_r|} - |\alpha_r| - \frac{z-\mu}{\sigma}\right)^{-n_r}, & \text{if } \frac{x-\mu}{\sigma} \geq \alpha_r, \\ \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right), & \text{otherwise} \end{cases}$$

Gaussian core with left/right exponential tails

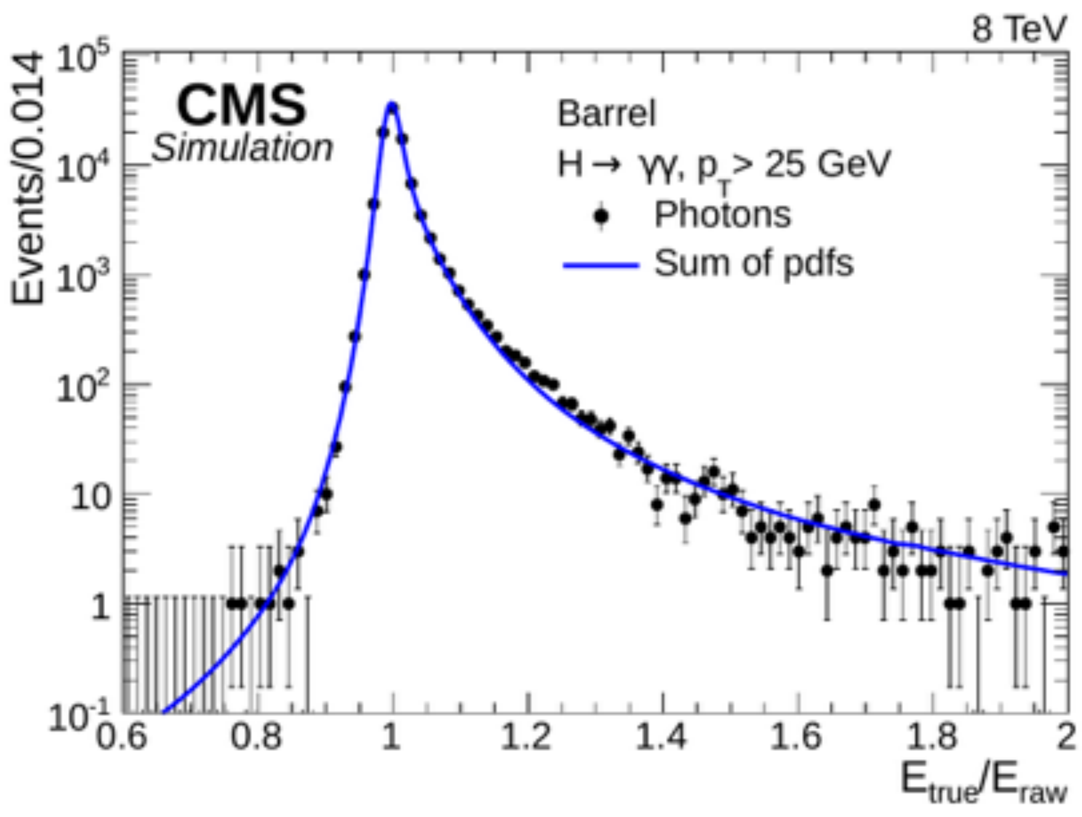
$$p(y|x) = f(y|\mu(x), \sigma(x), \alpha_l(x), n_l(x), \alpha_r(x), n_r(x))$$

The likelihood for the training is :

$$L = -\sum_{MC\ photons} \ln p(y|x)$$

BDT (Gradient Boosting)

This way we can obtain the parameters of the crystal ball as regression outputs:
e.g. we get the E-correction (μ) and the resolution (σ)



Ref. http://arxiv.org/abs/1502.02702

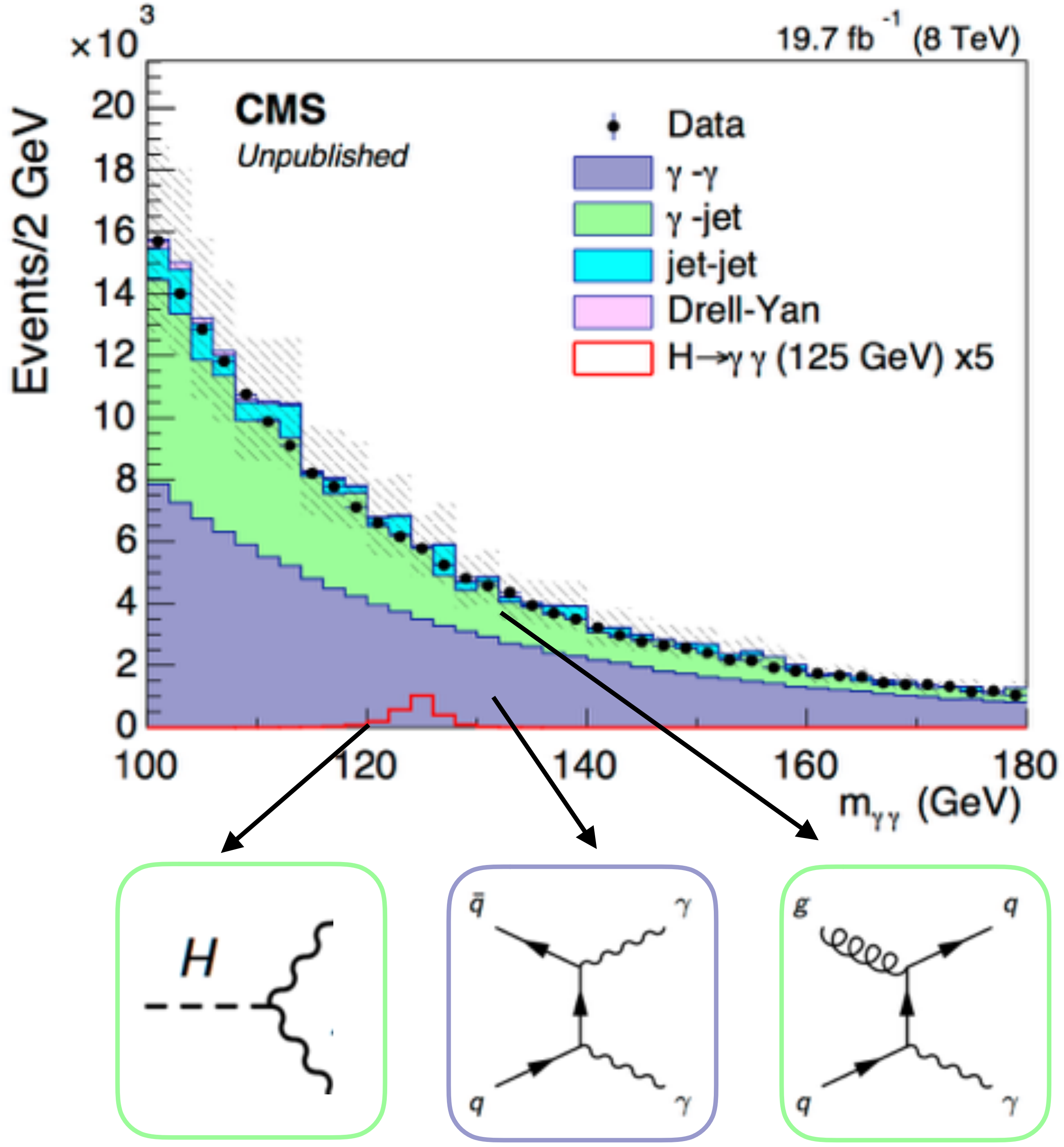
Example: Higgs→γγ classifier

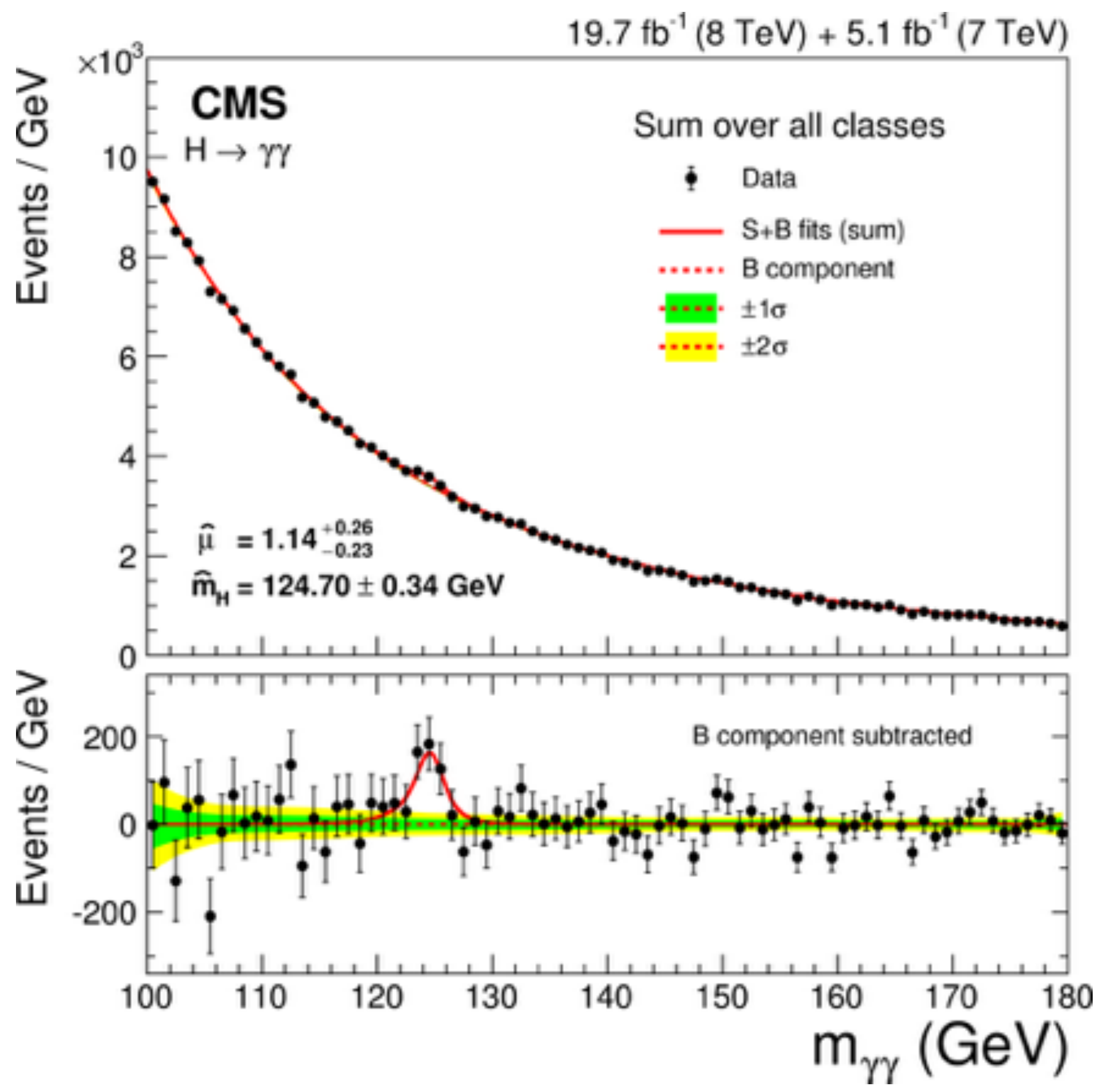
Signal and backgrounds

Higgs→γγ search

Signal: pair of photons from Higgs→γγ

Background: pair of photons or fake photons produced by other mechanisms



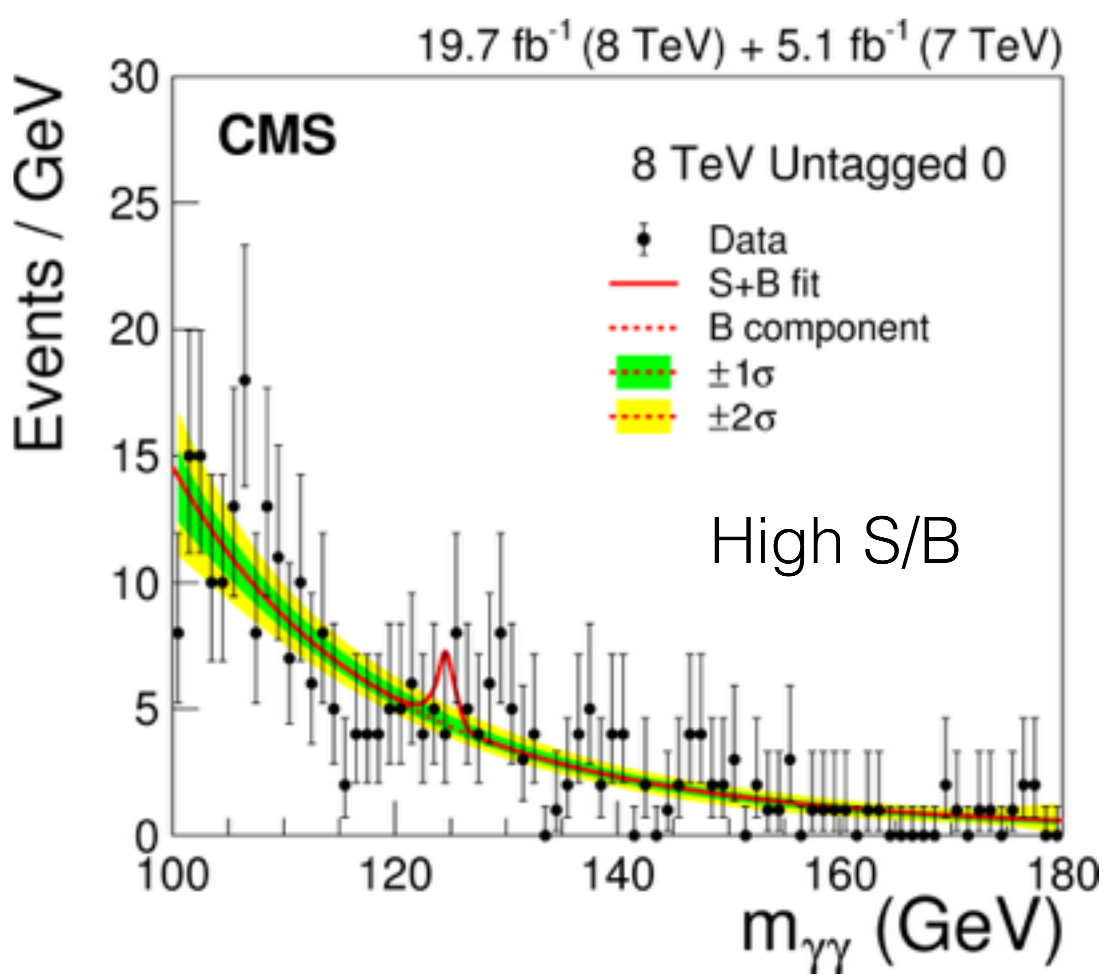


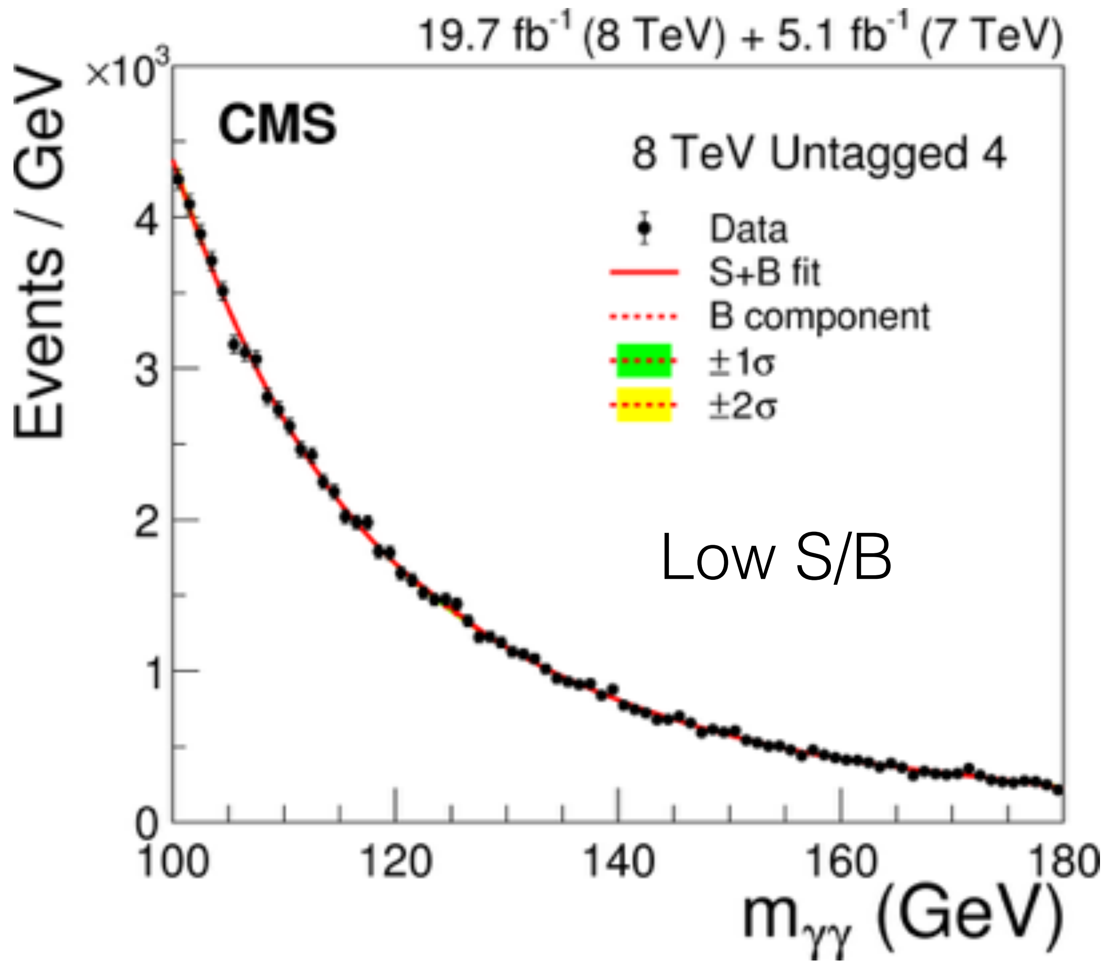
Fit a mass bump is the best way to convince anyone of the presence of a new particle

Ref. http://arxiv.org/abs/1407.0558

Event classes

To increase the analysis sensitivity we don't fit the total $m_{\gamma\gamma}$ distribution, but we split the events into classes based on signal resolution and Signal/Background





Strategy: 1) train one BDT classifier to separate Higgs signal from background,
2) bin events in several classed based on the BDT output,
3) fit the invariant mass in each category

Why don't you use the BDT until the end and extract the signal from the BDT distribution?

…fit a mass bump is the best way to convince anyone of the presence of a new particle. It has a direct physics meaning

Diphoton classifier

Train a BDT on MC:

- background: mix of all background mechanisms
- signal: before discovery we didn't know the mass of the Higgs boson !

  but for each mass we could build a very precise model:

  train at one given mass (e.g. 123 GeV) and make the BDT blind to the mass

We don't want to discover a bump at the signal mass used for training !

Dividing all variables with dimensions by the invariant mass of the candidate ($m_{\gamma\gamma}$) we hide the mass to the BDT (and it cannot learn it indirectly because of the incomplete information on the kinematics of the event)

Energy γ_1 / $m_{\gamma\gamma}$
Energy γ_2 / $m_{\gamma\gamma}$
η (γ_1)
η (γ_2)
cos ($\Delta\phi_{\gamma\gamma}$)
Photon ID (γ_1)
Photon ID (γ_2)
σ_m / $m_{\gamma\gamma}$

…

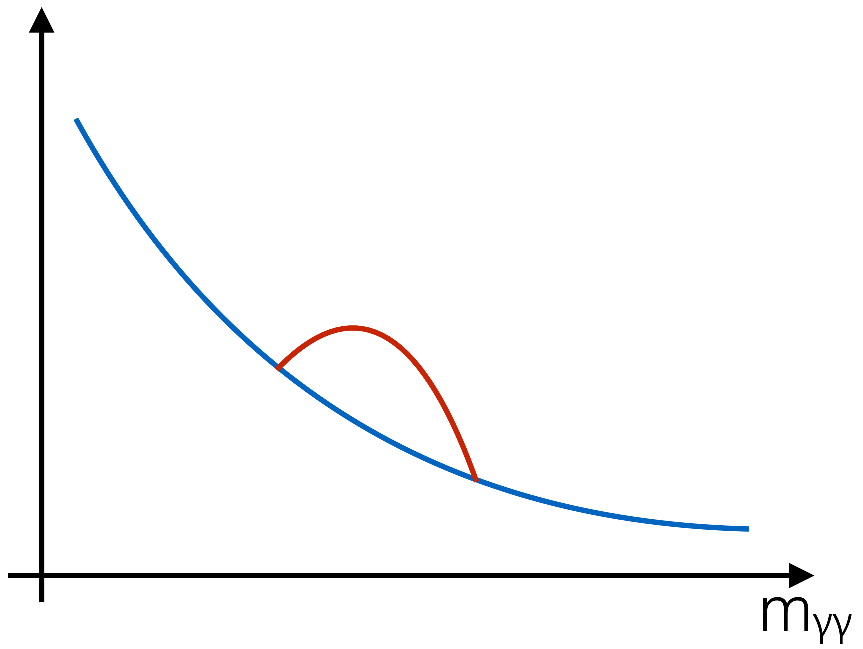
Signal like candidates in general have:

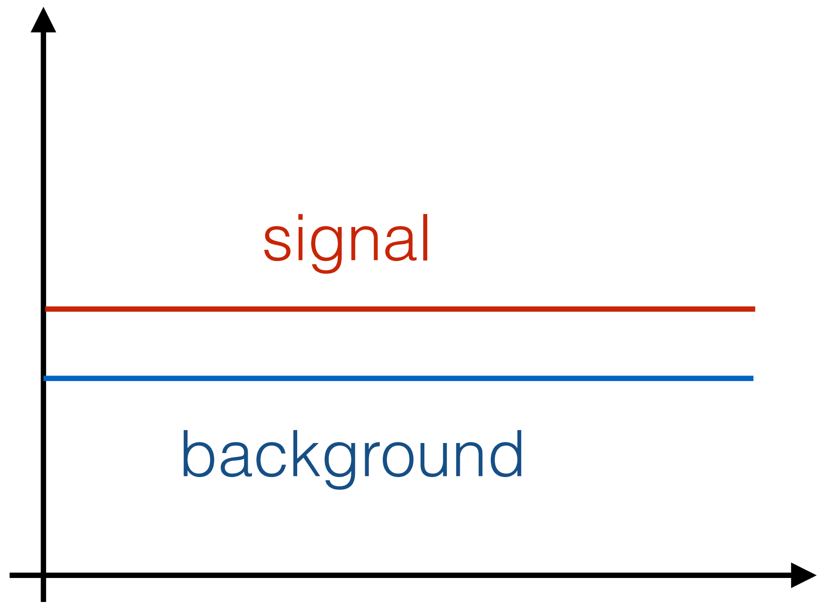
- higher photon energy
- more centrally produced
- better identified photons
- better mass resolution

Use BDTs in cascade: exe input energy regression, photon classifier, etc…

Diphoton classifier

Making the BDT mass blind…





…also means losing the important handle on the resolution information.
To bring the resolution back we apply a signal weight:

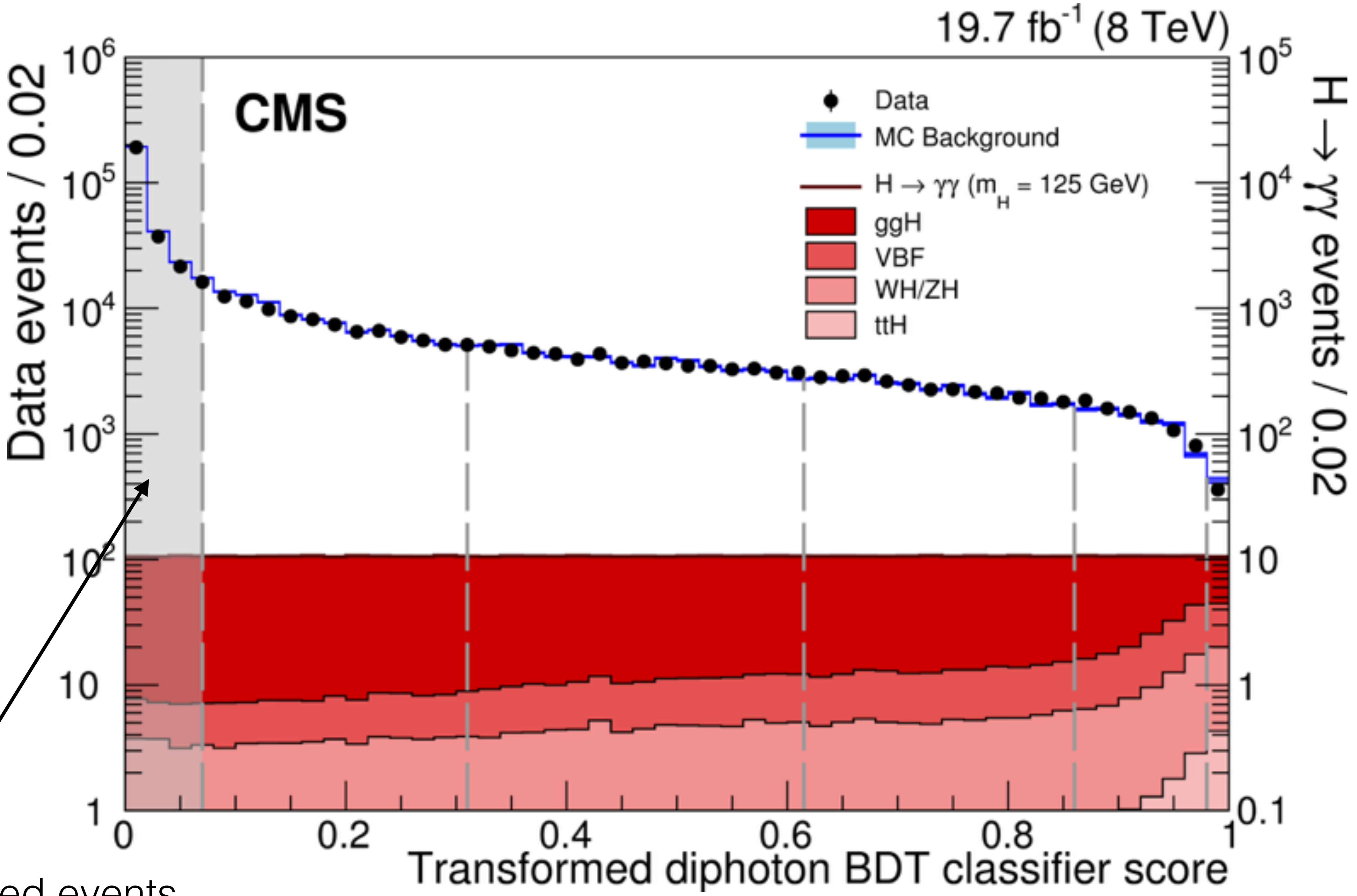
Weight = $1 / (\sigma_m / m_{\gamma\gamma})$ the better the resolution the larger the weight

In reality we don't know precisely the vertex (we get it from another BDT) and so we build a more complex weight that takes this into account:

Weight = $p^{vtx} / (\sigma^{\text{right vertex}}_m / m_{\gamma\gamma}) + (1 - p^{vtx}) / (\sigma^{\text{wrong vertex}}_m / m_{\gamma\gamma})$

BDT output

Number of classes (5) and boundaries chosen to optimize the S/B.
(discard events in the lowest score bin)

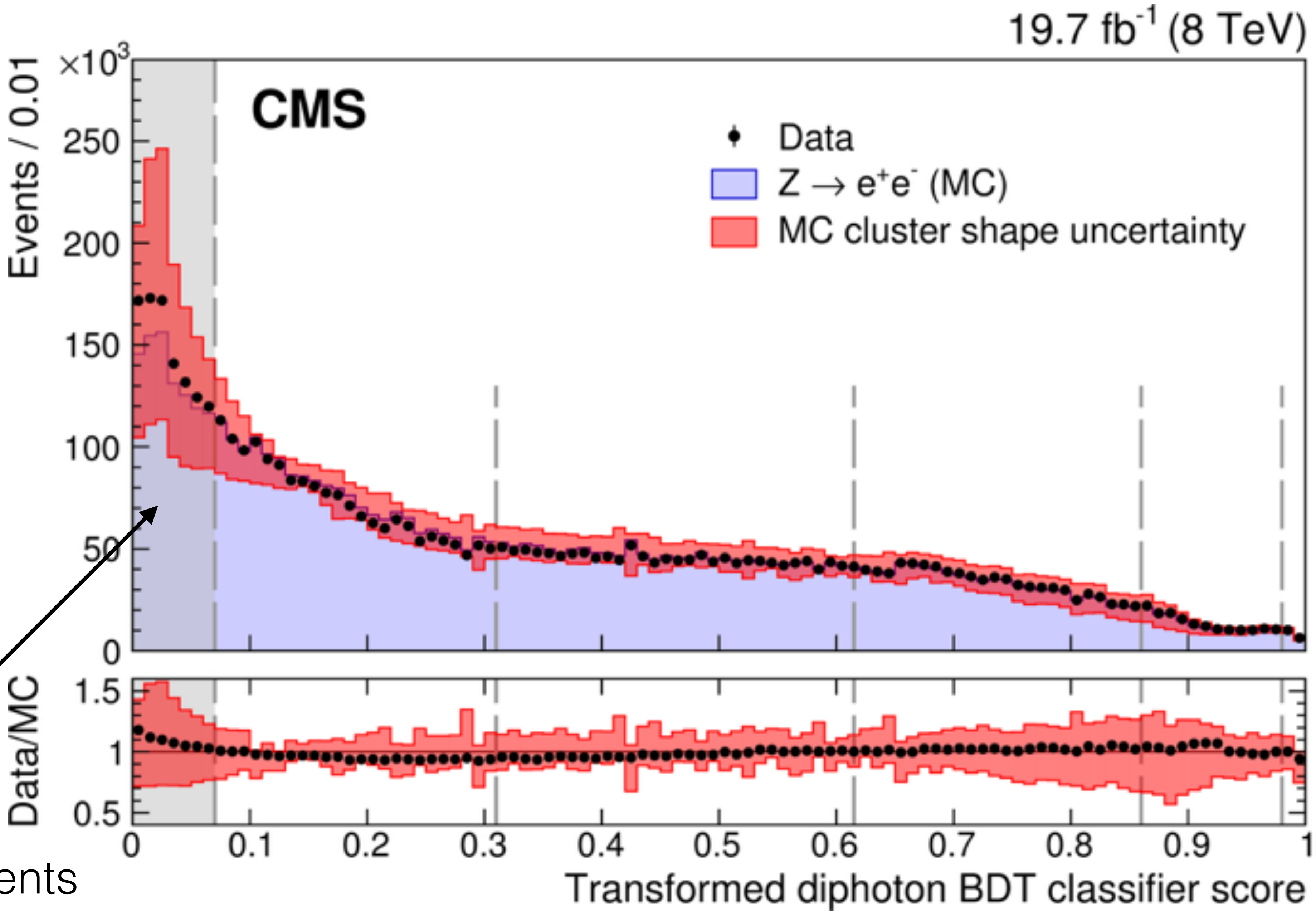


Discarded events

Transformed such that the sum of the signal components is flat

Validation

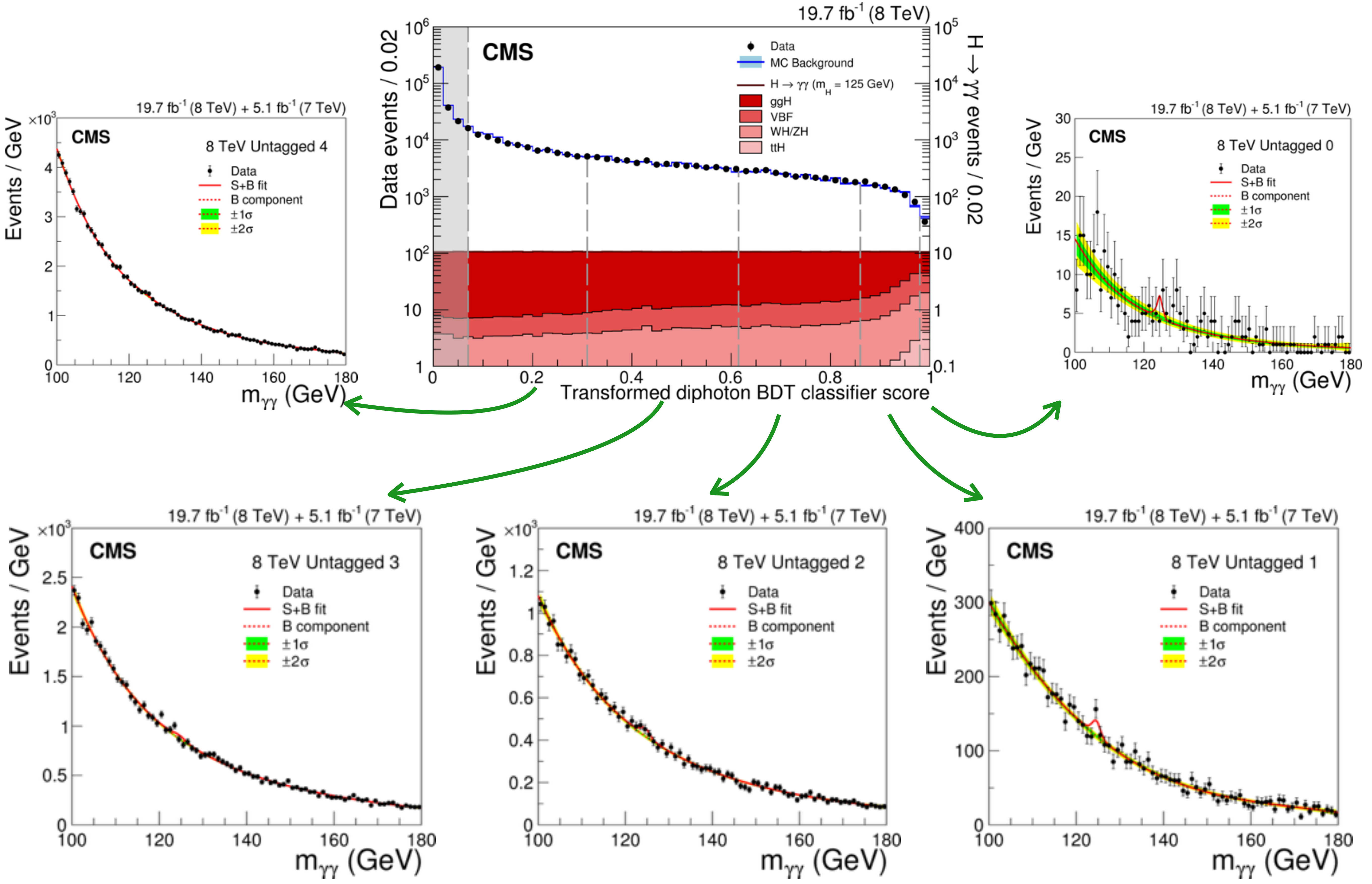
Validation on standard candle Z→ee events reconstructed as photons:
check that input variables and their correlations in the MC is sufficiently accurate



Discarded events

Systematic uncertainty band obtained propagating the photon identification uncertainty and the energy resolution uncertainty

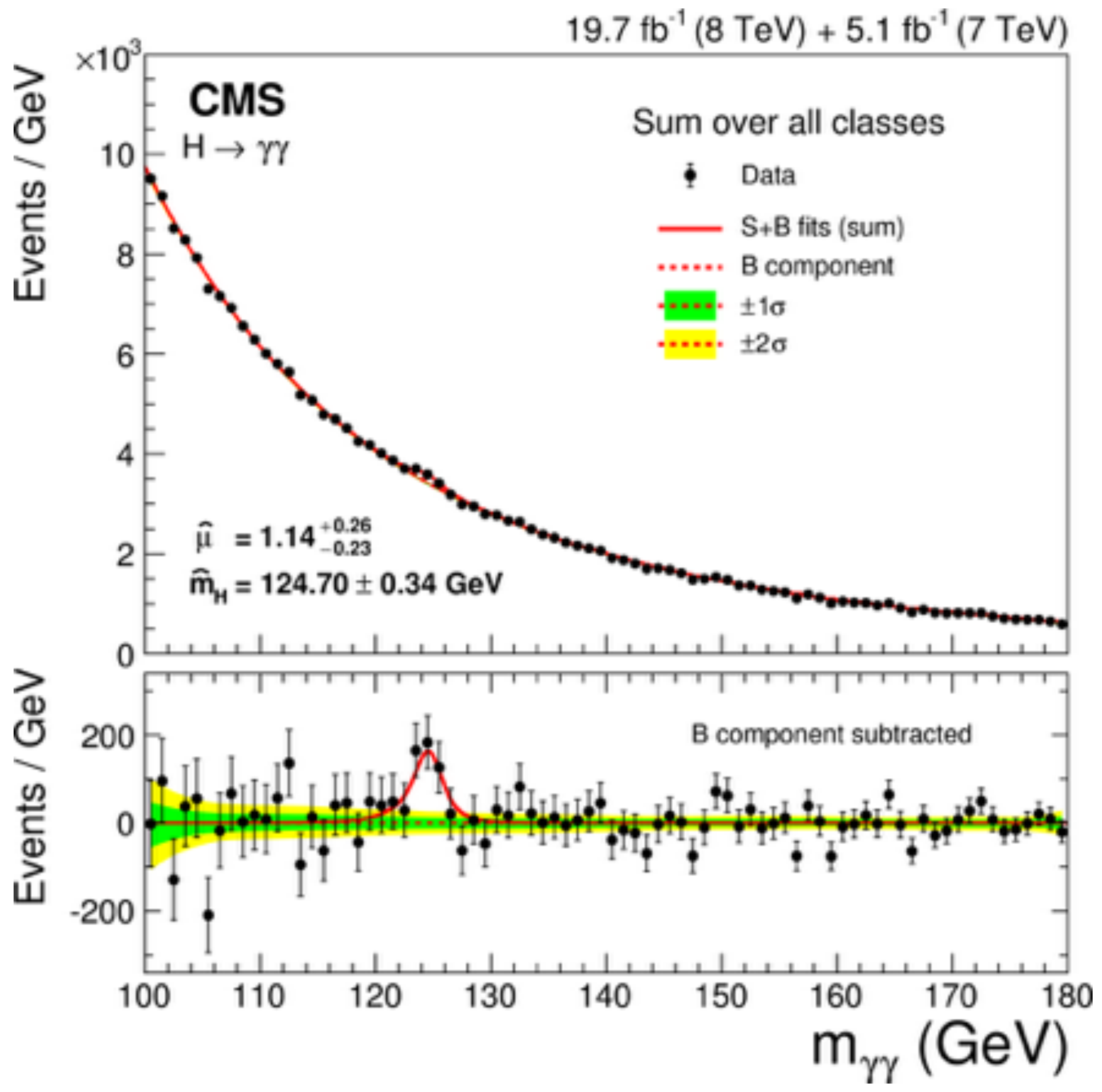
Mass fit

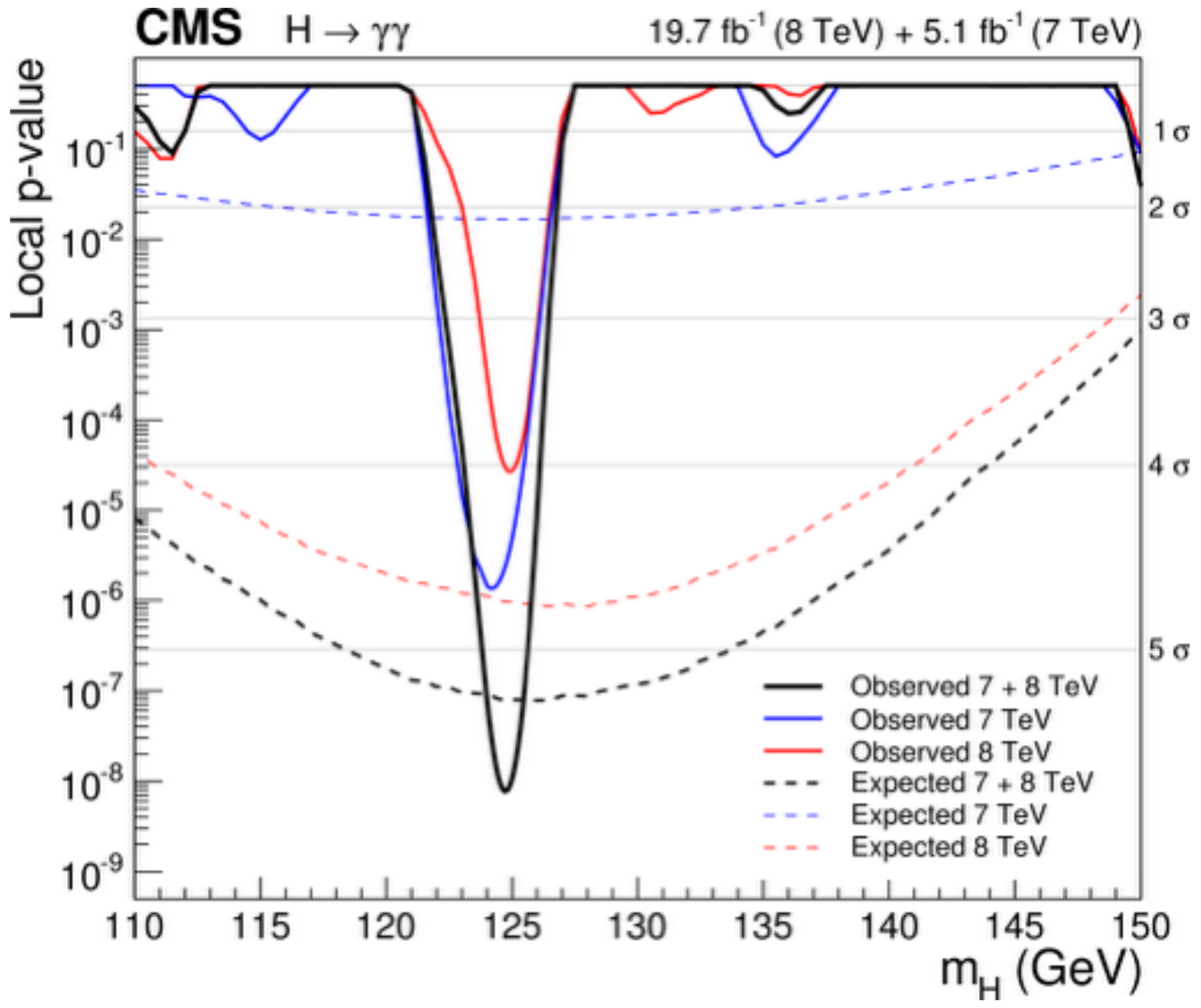


19.7 fb^{-1} (8 TeV)
CMS
Data
MC Background
H → γγ (m_H = 125 GeV)
ggH
VBF
WH/ZH
ttH
Data events / 0.02
H → γγ events / 0.02
Transformed diphoton BDT classifier score
19.7 fb^{-1} (8 TeV) + 5.1 fb^{-1} (7 TeV)
8 TeV Untagged 4
8 TeV Untagged 3
8 TeV Untagged 2
8 TeV Untagged 1
8 TeV Untagged 0
S+B fit
B component
±1σ
±2σ
Events / GeV
$m_{\gamma\gamma}$ (GeV)

Higgs discovery

All previous classes (together with tagged events and 7 TeV classes) combined in the final result



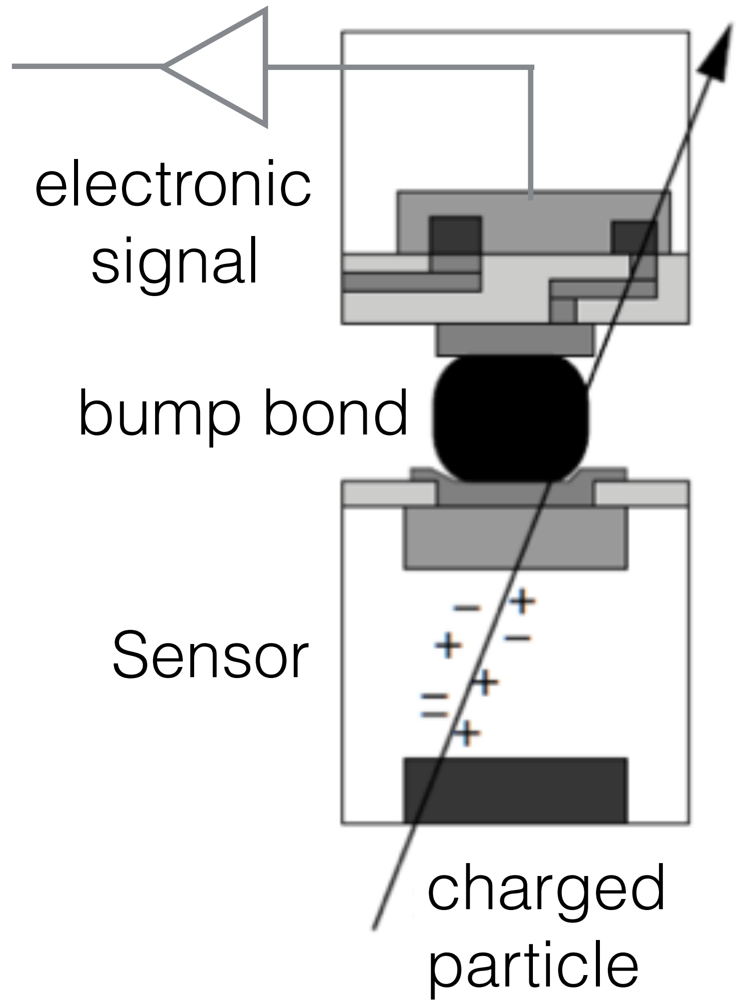


Ref. http://arxiv.org/abs/1407.0558

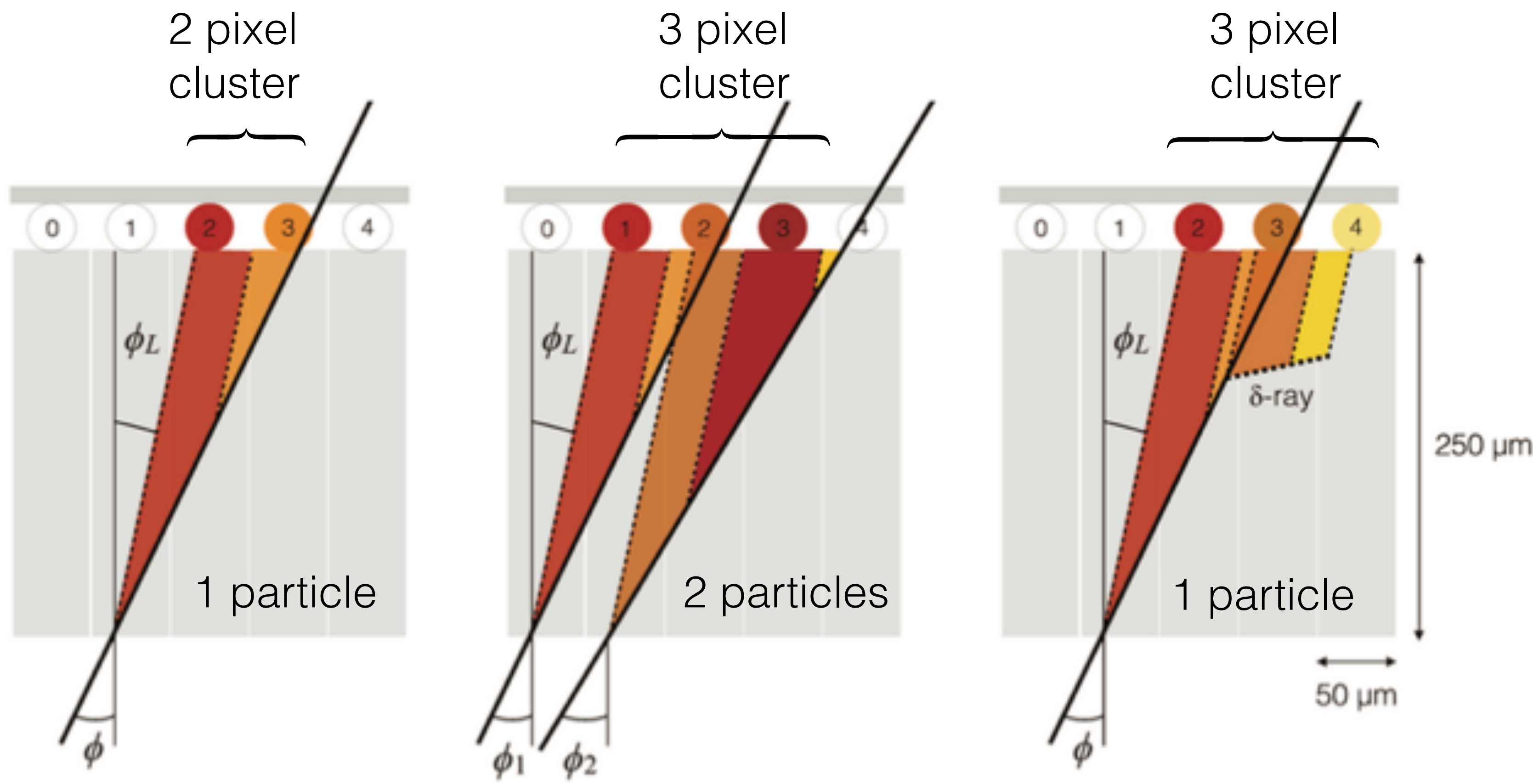
Flash examples

Pixel clustering

Charged particles passing through a pixel unit creates an electronic signal.



Pixels units are arranged in a 2D matrix in a module (here is a cross section).



Pixels are merged into clusters using a CCA (8 cell connected component analysis). Barycentre is used to improve resolution

The innermost layer of the pixel detector is the one that suffer the most by merged hits (higher particle density per cm^2)

How to assign / split clusters to different tracks ?

Already identifying merged clusters improves track quality through ambuigity-resolving

Ref. http://arxiv.org/abs/1406.7690

Neural Network for cluster splitting

To split pixel clusters we use 10 Neural networks:

- 1 NN to estimate the probability that a cluster was created by (1, 2, 3) tracks
- 3 NNs (x,y)x(1,2,3) outputs used to estimate the impact point of the particle (regression)
- NNx x(1,2,3) + NNy x(1,2,3) used to estimate the uncertainty on the impact point of the particle in the transverse / longitudinal directions

No feature extraction the full information is provided to the NN: 60 input nodes
7x7 matrix of the charged collected in the cluster, size of the pixels in the cluster
layer number and if barrel/endcap, angles of incidence of the charged particle

All networks include two hidden layers

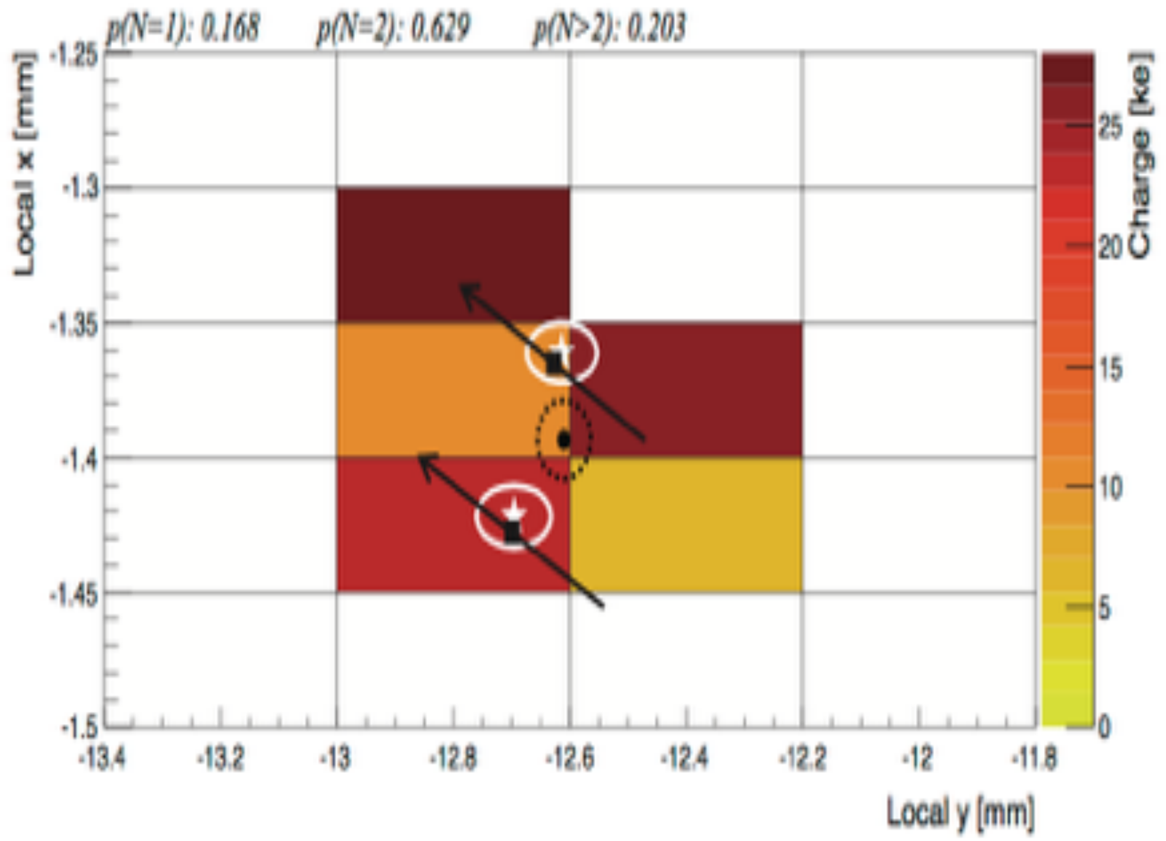
Training samples are chosen to represent busy particle environments (jets core) and the training is performed with JetNet

NN cluster splitting runs 6 times slower than CCA still fast enough:
only 5% of the total event reconstruction time

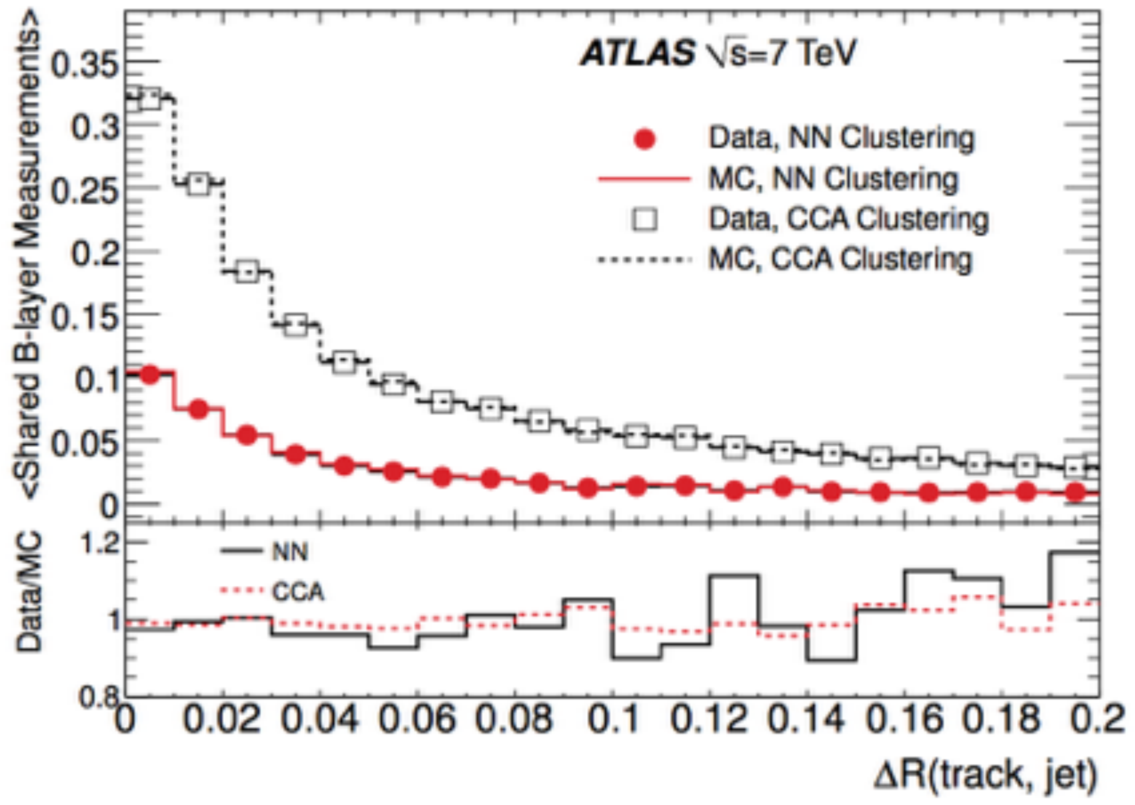
Cluster splitting performance

The NNs can mistakenly split a cluster and increase the risk of fake/duplicate tracks.
Working point tuned on simulations: 71% efficiency in splitting 2 particles with
7.5% single particle clusters wrongly split in two

first NN output

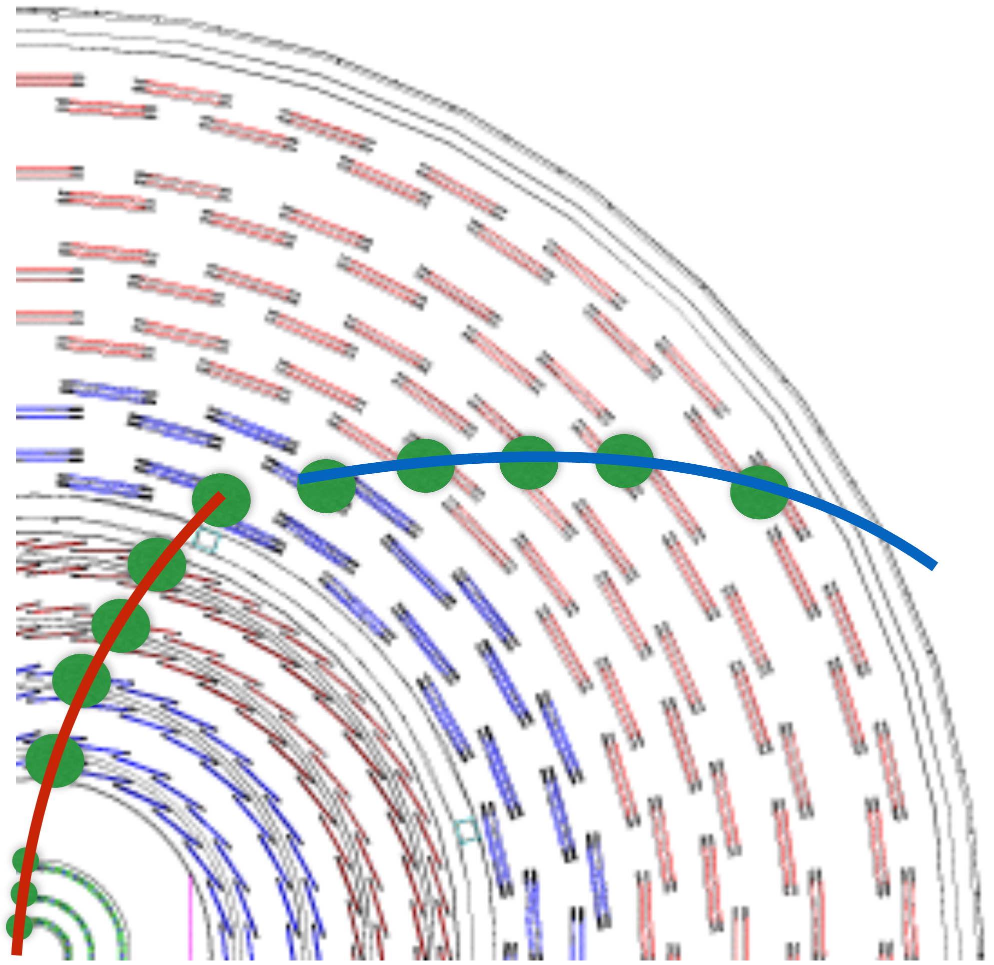


Example of cluster split



Average number of shared measurements in the innermost pixel layer, before (CCA) and after splitting (NN)

Track selection

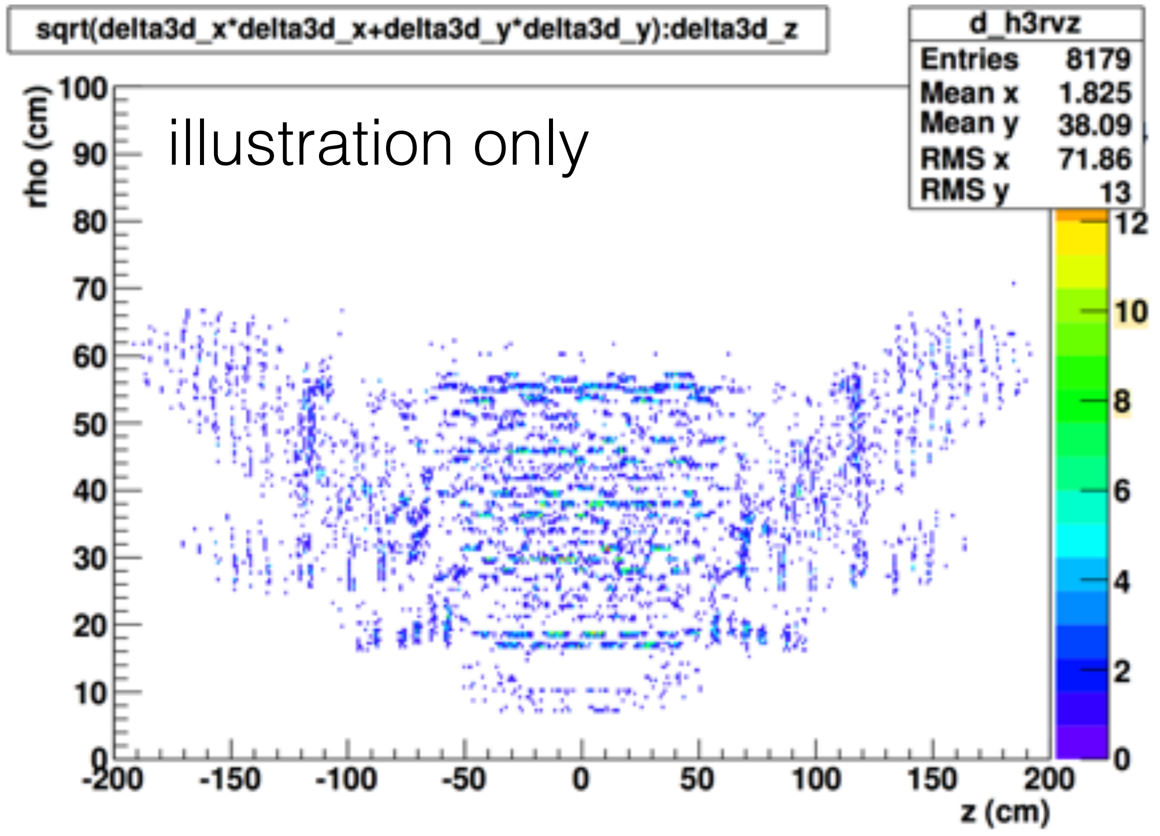


In MC, we observed that a number of true tracks are associated to 2 or more reconstructed tracks.

We sought to identify pairs of tracks that are duplicates and to merge them into a single track.

Train a BDT using as input variables e.g.:
- nMissingOuterHits from inner track
- nMissingInnerHits from outer track
- Average Hit R
- Average Hit Z

Reduced on average the number of duplicate tracks by a factor of ~2



Data popularity

CMS has a Dynamic Data Placement group which uses historical information to place popular dataset to sites: optimize resources usage.

Predict which datasets will become popular once they appear and also predict decline in popularity of certain datasets, reduce redundant activity, improve resource allocation, etc.

Main steps:

- Dataframe generator toolkit: collect/transform data from CMS data-services (DBS/PhEDEx/SiteDB / PopularityDB/Dashboard)
- Select input variables (nfiles, size, nsites, nreleases, creator, naccess, nuser, totcpu,…
- Feed Machine Learning (ML) algorithms (python/R code) for data analysis

Achieved the first proof of concept

Ref. https://indico.cern.ch/event/365073/contribution/0/attachments/726102/996447/DCAF4CERN_IT_Analytics.pdf

- V. Kuznetsov, T. Wildish, L. Giommi, D. Bonacorsi, "Exploring Patterns and Correlations in CMS Computing Operations Data with Big Data Analytics Techniques", presented by D. Bonacorsi at ISGC'15 (March 15-20 2015), accepted for publication in PoS SISSA, http://pos.sissa.it/archive/conferences/239/008/ISGC2015_008.pdf

- V. Kuznetsov, T. Wildish, D. Giordano, N. Magini, T. Boccali, M. Neri, M. Girone, D.Bonacorsi, "Exploiting CMS data popularity to model the evolution of data management for Run-2 and beyond", presented by D. Bonacorsi at CHEP'15 (April 13-17, 2015), to be published in IOP-CS, https://indico.cern.ch/event/304944/session/5/contribution/335

- "Studi di data popularity nell'analisi distribuita su Grid dell'esperimento CMS a LHC", B.Sc. thesis of Matteo Neri (can be provided upon request - available only in Italian though)
- "Predicting CMS datasets popularity with Machine Learning", B.Sc. thesis of Luca Giommi (can be provided upon request)
- "Evaluation of Apache Spark as framework for CERN's Big Data Analytics", CERN openlab 2015 report of Siddha Ganju (can be provided upon request)

Final remarks

ROOT
Data Analysis Framework

TMVA

The role of ROOT / TMVA

All our analyses are developed within the ROOT framework.
The ML package most widely used is ROOT/TMVA

Because TMVA is ROOT one can move from a cut based analysis to a ML algorithm in no time
Many algorithms are supported:

- Rectangular cut optimisation
- Multi-dimensional likelihood estimation
- Linear and nonlinear discriminant analysis
- Artificial neural networks
- Support vector machine
- Boosted/bagged decision trees
- etc…

TMVA opened the door of ML to the HEP community !

Some considerations:

- most TMVA users use ML algorithms as black boxes: it works why should I care ?
- little/ no attempt at optimising the default settings: it works why should I care ?
- on the other hand… it works why should I care ?

With a zero-cost and performing tool, it is difficult to move forward to explore something new.
Adding to inertia: ROOT / ML tools use different interfaces (HEP works with ROOT-NT)

This workshop is a golden occasion to answer the important questions:
How optimal is our use of ML algorithm ? How much can we gain ?

Summary

Machine Learning algorithms in ATLAS/CMS are ubiquitous

Analysis design and design of Machine Learning models/tools are deeply connected. We see through our detectors and tools

Begin the discussion with people from different backgrounds to create new ideas to answer fundamental physics questions

Thanks for the fruitful discussions to :
J, Bendavid, T. Golling, V. Innocente, V. Kuznetsov, P. Musella, G. Petrucciani, M. Pierini, J.R. Vlimant, M. Walker