UNIVERSITÄT SIEGEN

CPPS Center for Particle Physics Siegen

15th February 2024

Landhaus Nordhelle, Meinerzhagen

CPPS group retreat

Data analysis and fits

Diptaparna Biswas, Eleonora Guido, Gilberto Tetlalmatzi-Xolocotzi

Introduction

We have identified **3 main topics of interest** for the whole CPPS group:

1. **Understanding your data and motivations**
2. **Fit techniques**
3. **Machine Learning in Physics**

For each one of the 3 points we will:

- introduce the topic and its basic concepts through some examples from our experience and/or our group work
- suggest useful tools and techniques
- leave some time *for discussion* and for your comments

Outline

1. **Understanding your data (6 mins)**
    1. Introduction (1 min) - Eleonora
    2. Examples: data analyses in astroparticle physics (5 mins) - Eleonora

2. Discussion (4 mins)

3. **Fitting methods (35 mins)**
    1. Non-leptonic amplitudes from fits to data (20 mins) - Gilberto
    2. Signal+Background fits using PDFs and HistFactory (5 mins) - Diptaparna
    3. Profile likelihood unfolding (5 mins) - Buddha
    4. EFTfitter: Interpreting measurements in the context of EFT (5 mins) - Jan

4. Discussion (10 mins)

5. **Machine learning (20 mins)**
    1. CNNs applied to a photon search in Auger (10 mins) - Eleonora
    2. BDT and hyperparameter optimization (5 mins) - Arpan
    3. Jet flavour tagging using Deep Sets and GNN (5 mins) - Diptaparna

6. Discussion (15 mins)

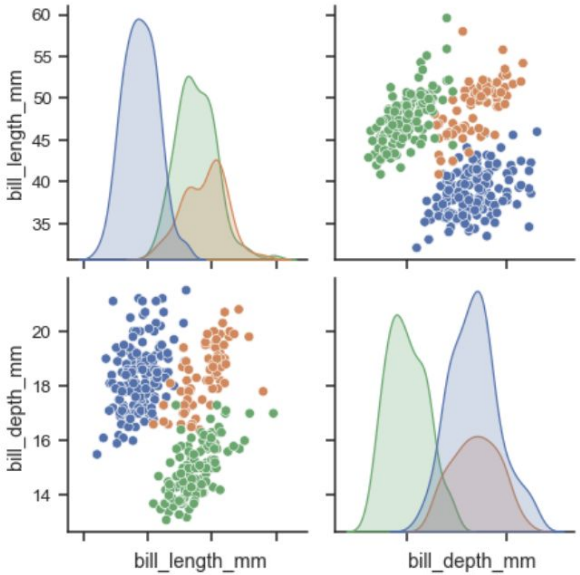
1. Understanding your data

Understanding your data through data visualization

- Data data visualization is the representation of data in graphical/visual format

- **Why is it crucial in physics?**
  - Enhance the *understanding of physical phenomena*
  - Help in *identifying patterns*, *trends* and which variables are the most significant
  - Effectively *communicate the results* to any audience

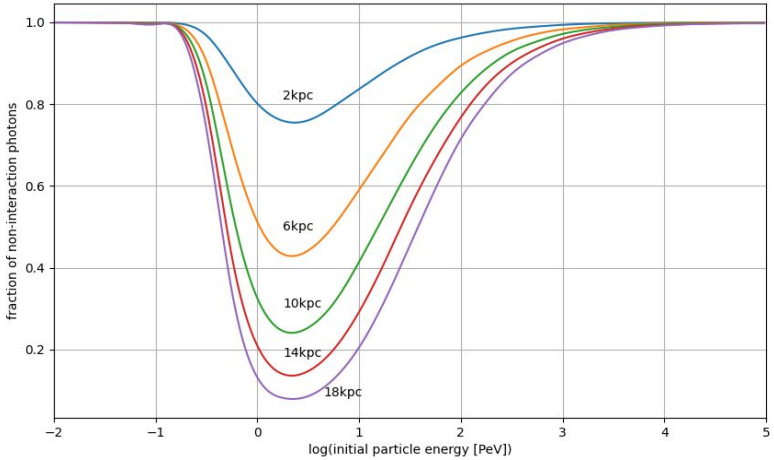
- Common tools for data visualization are for example the **Python libraries matplotlib and seaborn**

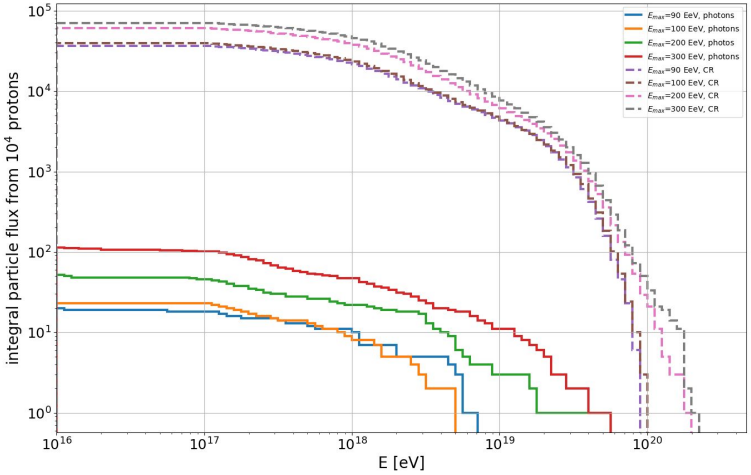


Data analyses in astroparticle physics

Eleonora Guido

Searching for PeV Photons from galactic PeVatrons



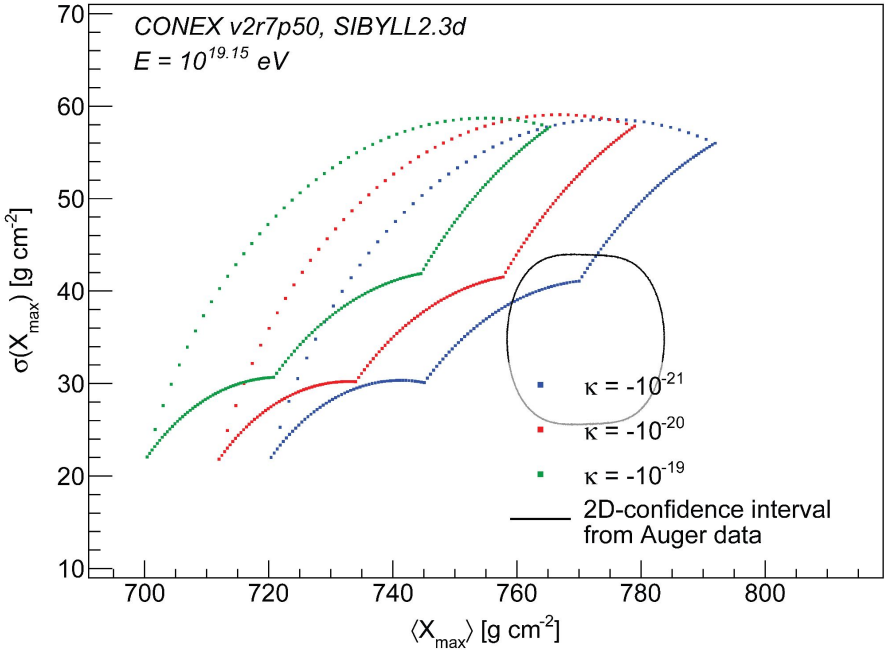


- Evaluating the interactions affecting photons during their propagation (simulated with CRPropa)
  a. Fraction of photons reaching us without interacting, depending on energy and source distance
  b. Integral flux of secondary photons depending on initial proton energy
- Plots generated with Pyplot and numpy.histogram

Chiara Papior

Testing Lorentz violation using air showers

- Testing the impact of LV on the development of simulated air showers
- Comparison between simulated showers and measurements used to set bound on LV parameter κ
- Parameter space related to mass composition: allowed regions for different κ (umbrellas) compared to Auger measurements (black "area")
- Bounds: $-6\times10^{-21} < \kappa < 3\times10^{-20}$
- Plot generated with ROOT libraries

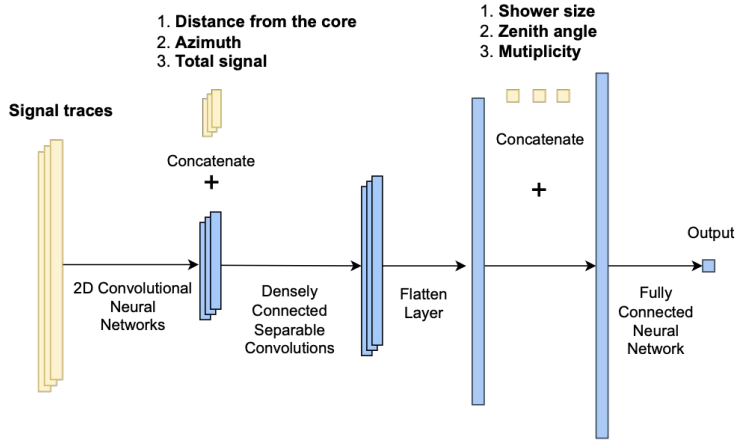


Fabian Dünkel

t-SNE for visualizing high-dimensional data

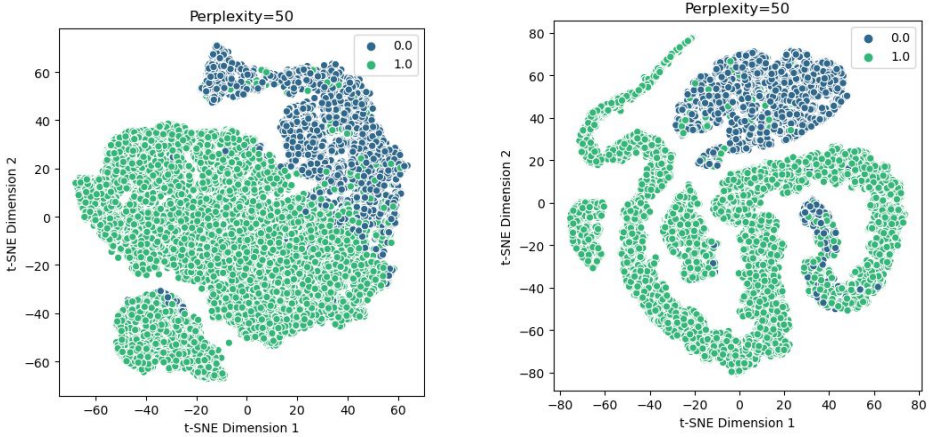
- **t-Distributed Stochastic Neighbor Embedding** is a powerful dimensionality reduction technique that captures nonlinear relationships within complex data
- It allows to **visualize data in a lower dimensional space** (2D/3D), preserving the local structure of data (points that are close together in the original high-dimensional space are close together also in the lower-dimensional space)

➢ t-SNE can be used to visualize the **latent space** of a neural network model
➢ Latent space : the output of a hidden layer (generally the last one)
➢ t-SNE provides a visualization of the inner representation of a neural network, at different layers



t-SNE for visualizing high-dimensional data

- t-SNE used to visualize the latent space of a Convolutional Neural Network (CNN) model
- The CNN model has been trained on a dataset with several hundreds of input variables (a complex and high-dimensional dataset) and aims at distinguishing between two classes of events .



- Visualizing the representations at different layers we can gain insights on the model behaviour
- It allows us to interpret how features are learned, how they evolve across layers
- **Python libraries: scikit-learn and seaborn**

2. Fits

χ^2 fits

Gilberto Tetlalmatzi-Xolocotzi

The role of fits in High energy physics

- Fits play an important role in particle physics.
- Fits can be used to constrain the parameter spaces of theoretical models accounting for the relevant experimental data.
- The comparison of theoretical predictions with experimental data is usually complicated by the presence of uncertainties on the theoretical and the experimental side.

Formulation of the problem

f : likelihood function.

$\vec{x}$: set of observables (branching fractions, asymmetries, etc).

Ω : parameter space (for example NP Wilson coefficient regions).

$\vec{\omega}$: element of Ω.

We want to address two questions:

1. Which choice of parameters describes best the data observed $\vec{x}_0$.

   Best fit point $\hat{\vec{\omega}} = \arg\ \max_{\vec{\omega}\in\Omega} f(\vec{x}_0, \vec{\omega})$

2. Based on the observed data, which parts of the parameter space Ω can be ruled out and at which confidence level.

   Likelihood ratio test for confidence regions

   $$T(\vec{x}) = -2\ln \frac{\max_{\vec{\omega}_0\in\Omega_0} f(\vec{x}, \vec{\omega}_0)}{\max_{\vec{\omega}\in\Omega} f(\vec{x}, \hat{\vec{\omega}})}$$

   $\vec{\omega}_0$ Null hypothesis

Formulation of the problem

f : likelihood function.

$\vec{x}$: set of observables (branching fractions, asymmetries, etc).

Ω : parameter space (for example NP Wilson coefficient regions).

$\vec{\omega}$: element of Ω.

We want to address two questions:

1. Which choice of parameters describes best the data observed $\vec{x}_0$.

   Best fit point $\hat{\vec{\omega}} = \arg\ \max_{\vec{\omega}\in\Omega} f(\vec{x}_0, \vec{\omega})$

2. Based on the observed data, which parts of the parameter space can be ruled out and at which confidence level. Ω

   Likelihood ratio test for confidence regions

$$T(\vec{x}) = -2\ln \frac{\max_{\vec{\omega}_0\in\Omega_0} f(\vec{x}, \vec{\omega}_0)}{\max_{\vec{\omega}\in\Omega} f(\vec{x}, \hat{\vec{\omega}})}$$

$\vec{\omega}_0$

Null hypothesis

χ^2 Fits

Consider $$f(\vec{x},\vec{\omega}) \propto \exp\left[-\frac{1}{2}\left(\vec{x}-\vec{\mu}(\vec{\omega})\right)^T \Sigma^{-1}\left(\vec{x}-\vec{\mu}(\vec{\omega})\right)\right]$$

$\vec{\mu}(\vec{\omega})$ Theoretical estimation of the observables

$$\chi^2 := -2\ \ln f(\vec{x},\vec{\omega}')$$

Obtaining the best fit point implies minimizing χ^2

The likelihood ratio test becomes $$T(\vec{x}) = \min_{\vec{\omega}_0 \in \Omega_0} \chi^2(\vec{x},\vec{\omega}_0) - \min_{\vec{\omega}' \in \Omega} \chi^2(\vec{x},\vec{\omega}') := \Delta\chi^2(\vec{x})$$

Large values of $\Delta\chi^2$ indicate that the null hypothesis $\vec{\omega}_0$ is unlikely

χ^2 Fits

p-value estimation

$$p(\Omega_0, \vec{x}_0) = \max_{\vec{\omega}_0 \in \Omega_0} \int d^n\vec{y} f(\vec{y}, \vec{\omega}_0) \theta\Big(\Delta\chi^2(\vec{y}) - \Delta\chi^2(\vec{x}_0)\Big)$$

If the null hypothesis is true and we reject it for values of $\Delta\chi^2$ larger than the observed value $\Delta\chi^2(\vec{x}_0)$ the probability of wrongly rejecting the null hypothesis is at most $p(\Omega_0, \vec{x}_0)$

If $f(\vec{x}, \vec{\omega}) \propto \exp\Big[-\frac{1}{2}\Big(\vec{x} - \vec{\mu}(\vec{\omega})\Big)^T \Sigma^{-1} \Big(\vec{x} - \vec{\mu}(\vec{\omega})\Big)\Big]$

$\vec{\mu}(\vec{\omega}) = A\vec{\omega} + \vec{b}.$

$\Longrightarrow \quad p(\Omega_0, \vec{x}_0) = Q\Big(\Delta\chi^2(\vec{x}_0)/2\Big)$

The first condition should hold in the limit of an infinite number of observations, the second condition should hold at least locally

Upper incomplete gamma function

Confidence Level= *1-p*

χ^2 Fits in a nutshell

- Construct the χ^2 function. $\chi^2(\vec{\omega}) = \sum_i \frac{\left(\mathcal{O}_{\text{exp,i}} - \mathcal{O}_{\text{theo,i}}(\vec{\omega})\right)^2}{\sigma^2_{\text{exp,i}} + \sigma^2_{\text{theo,i}}}$

No correlations assumed

- Obtain the best fit point by minimizing over χ^2
- Obtain the confidence region, transform *1-p* value to $\Delta\chi^2$

$\Delta\chi^2 \leq$

$(1-p)$ (%)	$m=1$	$m=2$	$m=3$
68.27	1.00	2.30	3.53
90.	2.71	4.61	6.25
95.	3.84	5.99	7.82
95.45	4.00	6.18	8.03
99.	6.63	9.21	11.34
99.73	9.00	11.83	14.16

Software available

Minuit (Generic)
Flavio (Flavour physics)
EOS (Flavour physics)
MyFitter (Generic)
Make your own…

Example: Non-leptonic amplitudes from fits to data

Gilberto Tetlalmatzi-Xolocotzi

Consider exclusive *B* meson decays into pairs of light pseudoscalars

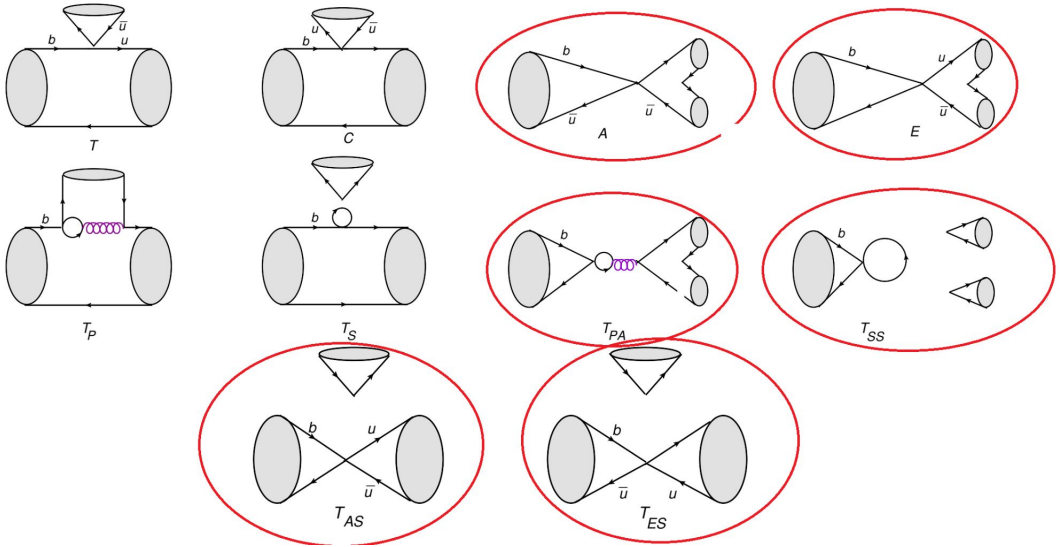
$$B^- \to \pi^0\pi^- \quad B^- \to K^0K^- \quad B^0 \to \pi^+\pi^-$$
$$B^0 \to \pi^0\pi^0 \quad B^0 \to K^+K^- \quad B^0 \to K^0\bar{K}^0$$
$$B_s \to \pi^0K^0 \quad B_s \to \pi^-K^+ \quad B^- \to \pi^0K^-, \text{ etc}$$

The relevant decay amplitudes can be written as $\mathcal{A}^{TDA} = i\frac{G_F}{\sqrt{2}}\left[\mathcal{T}^{TDA} + \mathcal{P}^{TDA}\right]$

$$\begin{aligned}\mathcal{T}^{TDA} = {} & T\ B_i(M)^i_j\bar{H}^{jl}_k(M)^k_l + C\ B_i(M)^i_j\bar{H}^{lj}_k(M)^k_l + A\ B_i\bar{H}^{il}_j(M)^j_k(M)^k_l \\ & + E\ B_i\bar{H}^{li}_j(M)^j_k(M)^k_l + T_{ES}B_i\bar{H}^{ij}_l(M)^l_j(M)^k_k + T_{AS}B_i\bar{H}^{ji}_l(M)^l_j(M)^k_k \\ & + T_SB_i(M)^i_j\bar{H}^{lj}_l(M)^k_k + T_{PA}B_i\bar{H}^{li}_l(M)^j_k(M)^k_j + T_PB_i(M)^i_j(M)^j_k\bar{H}^{lk}_l \\ & + T_{SS}B_i\bar{H}^{li}_l(M)^j_j(M)^k_k,\end{aligned}$$

The expression for $\mathcal{P}^{TDA}$ is analogous.

The subamplitudes have a diagramatic representation



The aim is to use experimental data to extract bounds to the annihilation topologies (diagrams inside the red contours) which lead to divergences in QFT.

Use branching fractions and CP asymmetries as observables

$$\Gamma(\bar{B} \to M_1 M_2) = \frac{S}{16\pi M_B} |\mathcal{A}_{B\to M_1 M_2}|^2 \qquad \mathcal{B}(\bar{B} \to \bar{f}) = \frac{1}{2}\tau_B \Big[\Gamma(\bar{B} \to \bar{f}) + \Gamma(B \to f)\Big]$$

$$\mathcal{A}_{\rm CP}(\bar{B} \to \bar{f}) = \frac{\Gamma(\bar{B} \to \bar{f}) - \Gamma(B \to f)}{\Gamma(\bar{B} \to \bar{f}) + \Gamma(B \to f)};$$

Our χ^2 is just $\chi^2 = \sum_i \left(\frac{\mathcal{O}_i^{\rm Theo} - \mathcal{O}_i^{\rm Exp}}{\sigma_i^{\rm Exp}} \right)^2$

34 total decay channels in total

For every channel there are in principle two observables, however experimentally not all the information is available

36 real quantities to be fitted (modulus and phases for each subamplitude)

- To find the best fit point we used random sampling, i.e. we generate randomly 10^9 points in our 36-dimensional space obeying a flat probability distribution.

- For each point we evaluate the χ^2 and take the lowest 5000 evaluations. .

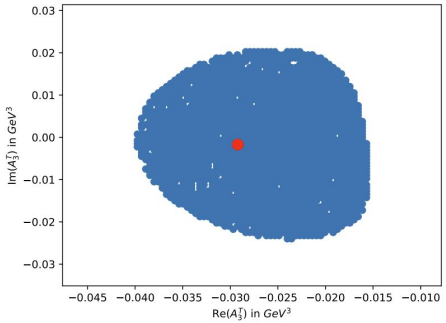
- We do a more sophisticated minimization taking each one of these 5000 new evaluations as starting points. For this we use the algorithm sequential least squares available in python.

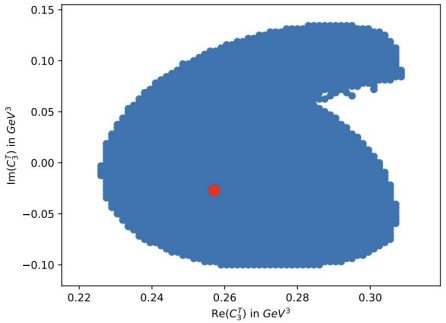
- Finally we determine the confidence regions considering $1 - p = 0.68$ which for the real and imaginary components of the amplitudes translates into

$$\Delta\chi^2 \leq 2.3$$

Our results are reported in the SU(3) irreducible representation basis for the best fit point

$$|A_3^T| = 0.029, \qquad \delta_{A_3^T} = -3.083, \qquad |C_3^T| = 0.258, \qquad \delta_{C_3^T} = -0.105,$$
$$|C_6^T| = 0.235, \qquad \delta_{C_6^T} = -0.079, \qquad |A_{15}^T| = 0.029, \qquad \delta_{A_{15}^T} = -3.083,$$
$$|C_{15}^T| = 0.151, \qquad \delta_{C_{15}^T} = 0.061, \qquad |B_3^T| = 0.034, \qquad \delta_{B_3^T} = 3.087,$$
$$|B_6^T| = 0.033, \qquad \delta_{B_6^T} = -0.286, \qquad |B_{15}^T| = 0.008, \qquad \delta_{B_{15}^T} = -1.892,$$
$$|D_3^T| = 0.055, \qquad \delta_{D_3^T} = 2.942,$$





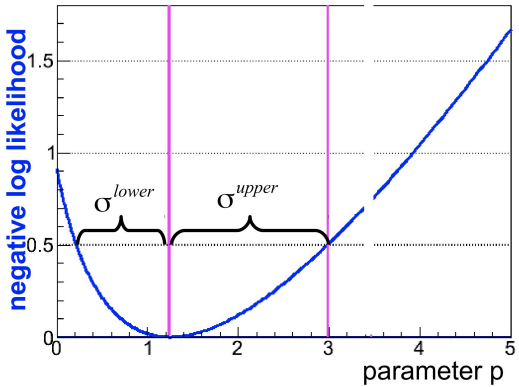
Signal+Background fits using smooth PDFs and HistFactory

Diptaparna Biswas

Maximum likelihood estimation (MLE)

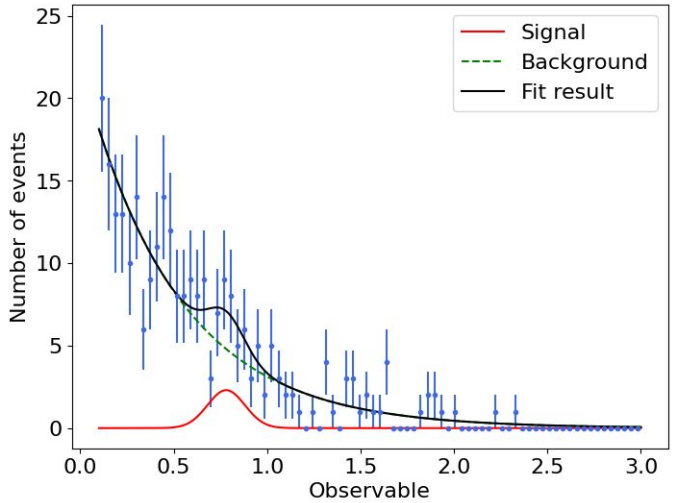
- The χ^2 fit tries to minimize the (relative) difference between the theoretical prediction and observation.
- But what if the prediction is not a point estimate?
  - ➢ Often the model prediction is a probability distribution.
    - Extract parameters using MLE.
- What is MLE?
  - ➢ Evaluate the parameterized PDF at each of the observed data points.
    - The obtained expressions are just functions of the model parameters.
  - ➢ Take product of them (assuming the data points are i.i.d.)
  - ➢ Find the parameter values that maximize the product.
    - In practice, we minimize the negative log-likelihood (nll).

$$L(\vec{p}) = \prod_{n=0}^{N} F(\vec{p}, \vec{x}_n) \cdot \underbrace{Poisson(N_{\text{exp}}, N_{obs})}$$



Fitting signal+background using extended MLE

- Both signal and background distributions are known from simulation.
  - ➢ Fit appropriate PDFs to extract their parameters.
    - Some parameter might be unknown!
  - ➢ Convert them to extended PDFs.
    - Two extra parameters: N_{sig}, N_{bkg}
- Take the sum of those extended PDFs as the combined model.
  - ➢ Fit it to the observed data.
    - Obtain N_{sig}, N_{bkg}, and any other parameters.
- Needs a data-modelling language:
  - ➢ Example: RooFit, zfit, probfit, GooFit

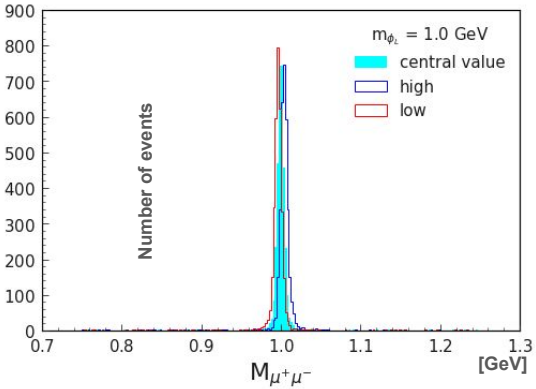


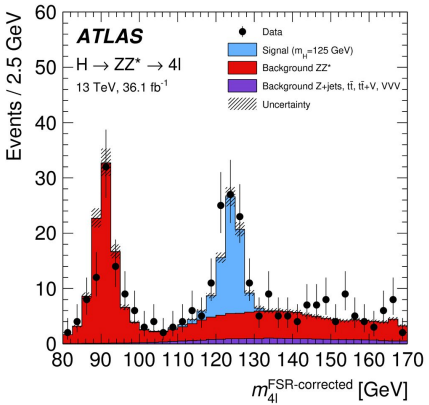

```
obs = zfit.Space('x', limits=(0,3))

mu = zfit.Parameter("mu",0.5, 0, 3)
lambda_ = zfit.Parameter("lambda",-2.0, -4.0, -1.0)
Nsig = zfit.Parameter("Nsig", 20., -20., 100)
Nbkg = zfit.Parameter("Nbkg", 300, 0., 1000)

signal = zfit.pdf.Gauss(obs=obs, mu=mu, sigma=0.1)
                 .create_extended(Nsig)
background = zfit.pdf.Exponential(obs=obs, lambda_=lambda_)
                     .create_extended(Nbkg)
tot_model = zfit.pdf.SumPDF([signal, background])
```

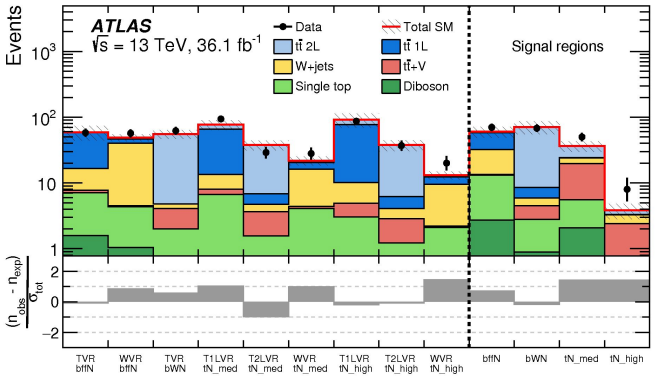
HistFactory

$$\mathcal{P}(n_c, x_e, a_p \mid \phi_p, \alpha_p, \gamma_b) = \prod_{c \in \text{channels}} \left[\text{Pois}(n_c|\nu_c) \prod_{e=1}^{n_c} f_c(x_e|\boldsymbol{\alpha}) \right] \cdot G(L_0|\lambda, \Delta_L) \cdot \prod_{p \in \mathbb{S}+\boldsymbol{\Gamma}} f_p(a_p|\alpha_p)$$

- The signal and background distributions are *defined* by their **histograms**.
  - ➢ Each bin is fitted to a poisson distribution: no need for a *data-modelling language*.
- Easy to include uncertainties.
  - ➢ Statistical uncertainties are included as poisson fluctuations.
  - ➢ Systematic uncertainties are included as “up” and “down” variation histograms.





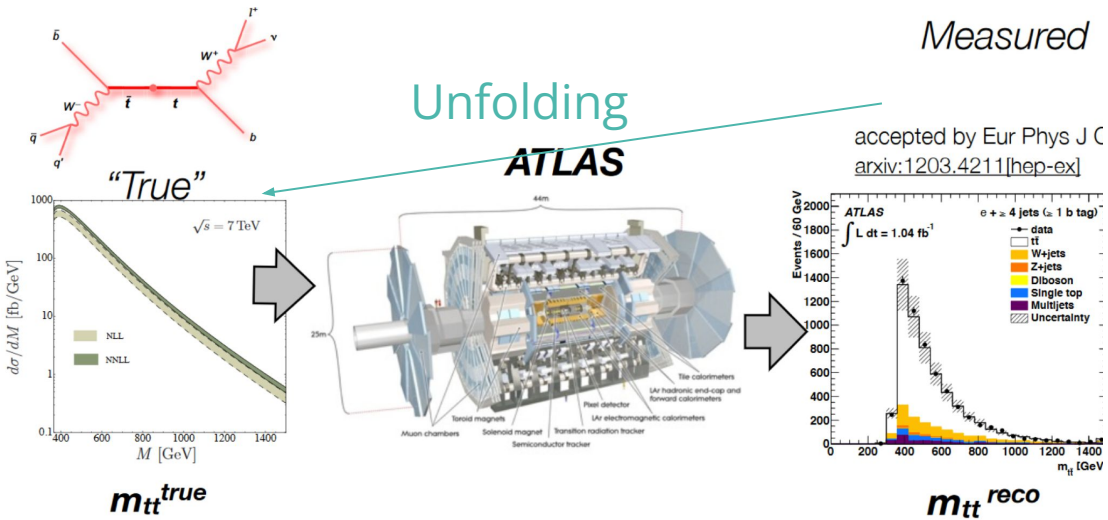


Profile likelihood unfolding

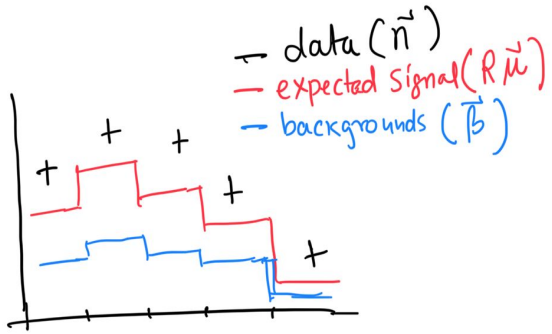
Buddhadeb Mondal

Why unfolding?

- **True spectrum** of a physical quantity is deformed by the **detector effects** -> **observed spectrum**
- **Unfolding** method is used to **correct those detector effects** to get the underlying true spectra (In mathematics known as inverse problem, deconvolution or unsmearing)
- Unfolding is not always necessary
  - If the goal is to compare the measured result with prediction, one can modify the prediction to include the detector distortion
- But in HEP we compare the results with other experiments, without unfolding the its difficult to compare the measurement with results from the other experiments



Formulation of unfolding problem



$$f_{\text{meas}}(x) = \int R(x|y) f_{\text{true}}(y)\, dy$$

R(x|y) is response function

In binned histogram, expected events at reconstruction level

$$R_{ij} = P(\text{observed in bin } i \,|\, \text{true in bin } j)$$

$$\nu_i = \sum_{j=1}^{M} R_{ij}\mu_j + \beta_i$$

μ_j = true histogram bin content

Likelihood construction:

$$P(n_i; \nu_i) = \frac{\nu_i^{n_i}}{n_i!} e^{-\nu_i}$$

$$\ln L(\boldsymbol{\mu}) = \sum_{i=1}^{N} (n_i \ln \nu_i - \nu_i)$$

ML estimator is

$$\hat{\boldsymbol{\nu}} = \mathbf{n}$$

$$\hat{\boldsymbol{\mu}} = R^{-1}(\mathbf{n} - \boldsymbol{\beta})$$

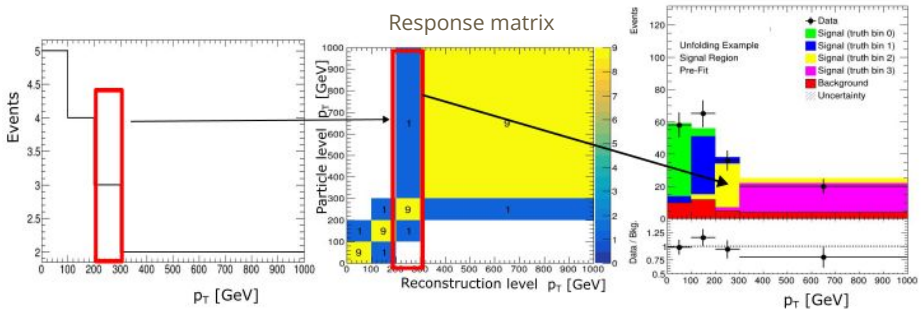
If the matrix isn't easily invertible, this becomes difficult

Profile likelihood unfolding

- Assign free floating normalization factor μ (POI) for each truth bin
- Fold the truth distribution with response matrix to obtain signal at reconstruction level
- Background templates needed at reconstruction level
- Construct the likelihood function (Diptaparna's talk)
  - Systematics are included as nuisance parameter with proper constraints
- Profile likelihood fit to the reconstruction level distribution -> fitted POIs measures the normalization of the truth, thus truth spectrum is obtained

** If the response matrix is not diagonal, use regularization

Toy example



$$L(\vec{n}^{data}|\vec{\mu},\vec{\theta}) = \prod_{c\in \text{Region}} \prod_{b\in \text{Bins}} \text{Pois}(n_{b,c}^{\text{data}}|\nu_{b,c}(\vec{\mu},\vec{\theta})) \cdot \prod_{p} \text{Gauss}(0|\theta_p,1)$$

$$\nu_{b,c} = \left[\sum_i \gamma_{b,c,i} \cdot \mu_i \cdot R_{b,c,i} \cdot T_i\right] + \sum_B \gamma_{b,c}^B \times N_{b,c}^B$$

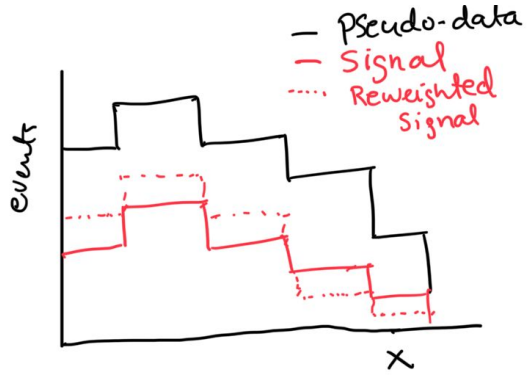
$\vec{\mu}$:= POIs $\vec{\theta}$:= NPs

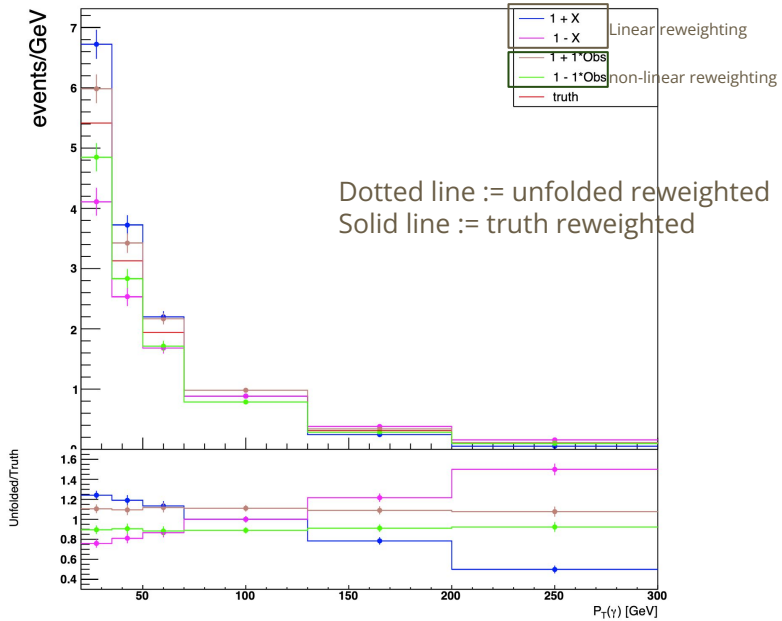
$\vec{T}$:= Truth bins

R := Response matrix

Are we biased by inputs?

- The goal is to correctly measure the signal from the data, unsmear the detector effects
- Unfolding using inputs obtained from MC (truth spectrum, response matrix), are we biased by the input shape?
- Stress test can be performed before actual fit is performed to data
  - Construct pseudo-data (Sum of all the MC templates)
  - Reweight the signal template in pseudo-data with some function
  - unfold using the nominal inputs, compare the unfolded result and truth





Example reweighting function

$$\text{weight} = 1 + \text{Y} \times \frac{100 - i}{300} = 1 + \text{Y} \times \text{X},$$

Y=+1, -1
i = bin

EFTfitter: Interpreting measurements in the context of effective field theories

Jan-Joachim Hahn

EFTfitter

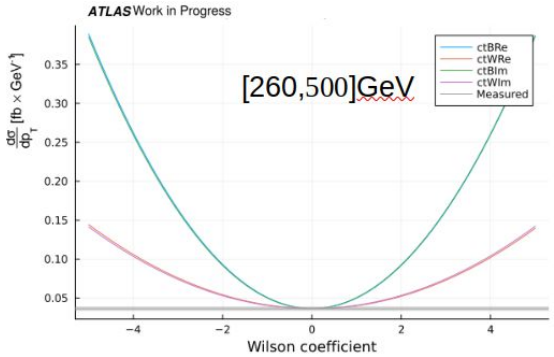
- Check compatibility of measurements with effective field theory (EFT) models
- Framework to perform fits to different models
- Written in Julia → Language optimised for calculation
- Needs measurements and correlation between them
- Declare observables→ Contain dependence of measured property on parameters to be fitted
- Input in multiple txt files, declaring properties in Julia-internal format
  - Vectors, matrix etc.
  - Can also interface with Python

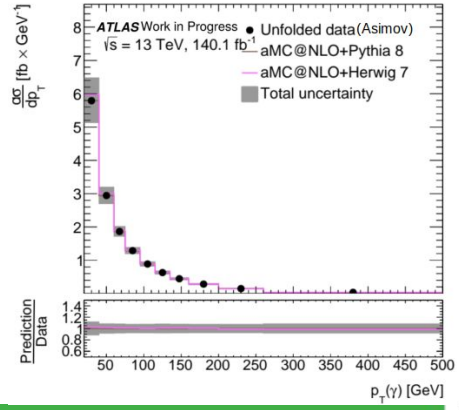
Website: https://tudo-physik-e4.github.io/EFTfitter.jl/stable/
arXiv: 1605.05585

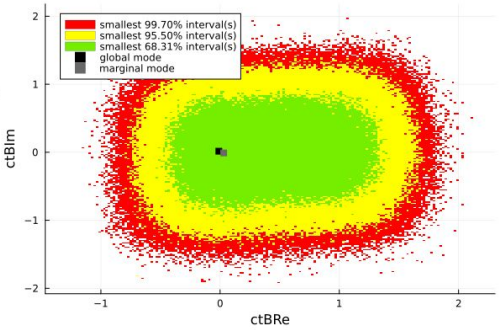
How does it work

- Use Bayesian inference
- Build likelihood from covariance M, measured value x and dependence on model parameter $p(\vec{x}|\vec{y}) = \sum_{i=1}^{n}\sum_{j=1}^{n}[\vec{x} - U\vec{y}]_i \mathcal{M}_{ij}^{-1}[\vec{x} - U\vec{y}]_j$
- Sample parameter space according to chosen prior using Markov-Chain Monte Carlo
  - Propose step, calculate probability to accept, draw number to choose if accepted, repeat
  - User just defines model, algorithm, number of samples, as well as range and prior
  - Outputs histograms and sampled values, as well as summary (mean, confidence level etc.)

Procedure to extract the limits of EFT parameters







Measured distribution

Uncertainties

Correlations

Monte Carlo EFT samples

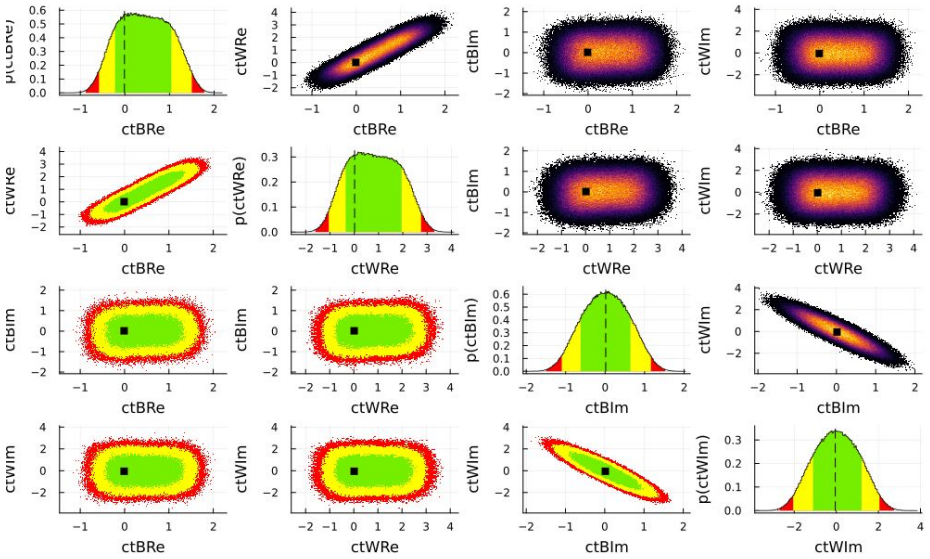
EFTfitter

Limits

Extract EFT dependence

Example

- This is a set of outputs as produced by the framework
- Shown are examples where the inputs are defined




```
measurements = (
        pTMeasDiff=MeasurementDistribution(diff_xsec,[
3.159712e-2,1.836804e-2,1.141040e-2,5.395894e-3,2.037598e-3,1.060480e-3
],
                                    uncertainties = (stat = [
5.8339180e-3,2.5110032e-3,1.3324190e-3,4.8867668e-4,1.5804446e-4,7.8118801e-5
            ],),                                    active=[true,true,true,true,true,true]),
)
```

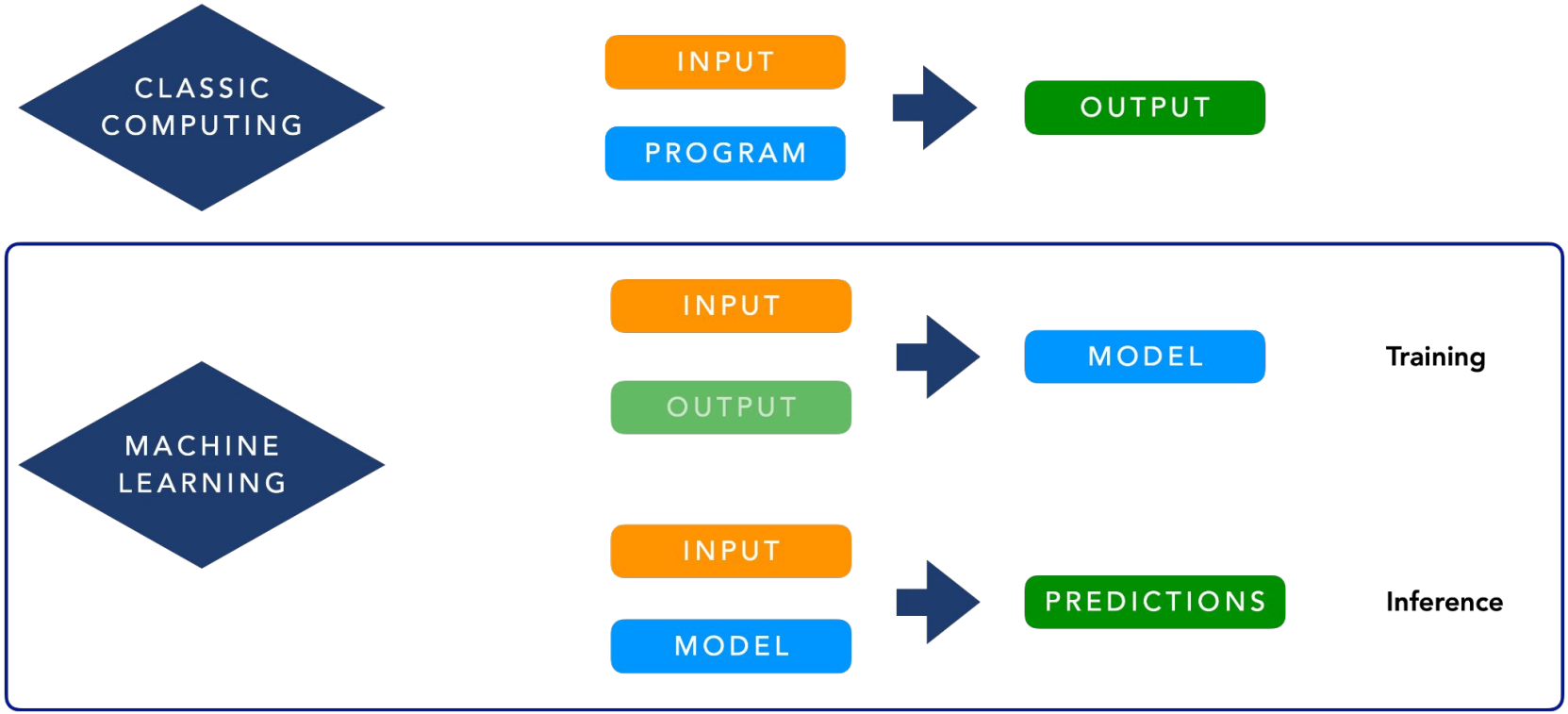
```
function ctWRe_Bin1(params)
    c=[4.572820e-2,4.546186e-4,3.720397e-5]
    return c[1]+c[2]*params.ctWRe+c[3]*params.ctWRe*params.ctWRe
end
```

```
correlations = (
    stat = Correlation([1.0000 0.7976 0.7506 0.6885 0.5093 0.4328 ;
                        0.7976 1.0000 0.7413 0.7107 0.4787 0.3680 ;
                        0.7506 0.7413 1.0000 0.7725 0.5642 0.4428 ;
                        0.6885 0.7107 0.7725 1.0000 0.6042 0.4848 ;
                        0.5093 0.4787 0.5642 0.6042 1.0000 0.4689 ;
                        0.4328 0.3680 0.4428 0.4848 0.4689 1.0000 ;], active=true),
)
```

3. Machine learning for physics

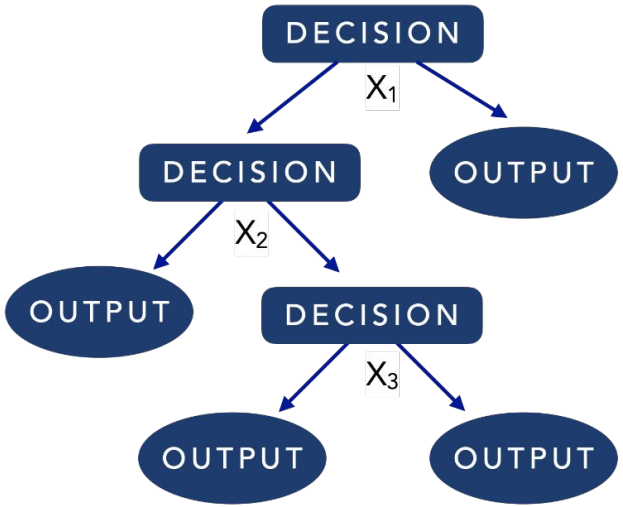
Eleonora Guido

From classic computing to machine learning

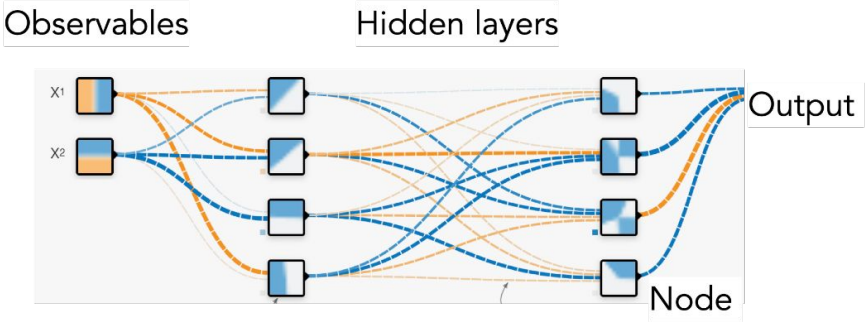


What about deep learning?

- A decision tree is an example of **machine learning algorithm** that classifies data basing on some selected observables x_i



- In a **deep neural network** the observables are combined in several layers to predict the output
- Numerous combination of the observables are formed



- The inputs variables are mapped to the output
- It is useful if we have many features (observables) that have to be mapped
- The algorithm identifies features and then combine them to produce the output

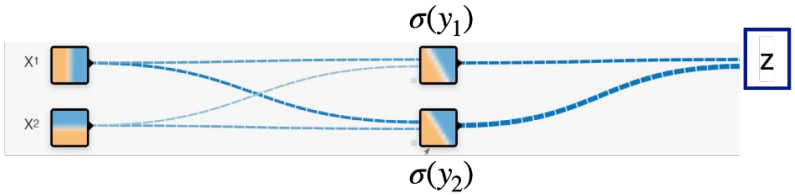
Building neural networks

- In each node we have a linear mapping with a matrix of weights **W** and a displacement parameter **b** (bias)

$$\vec{y} = \mathbf{W}\,\vec{x} + \vec{b} \quad \text{(e.g. } y_1 = W_{11}x_1 + W_{12}x_2 + b_1\text{)}$$

- Such a model is series of linear mapping and it is suitable only for linear problems
- It can be generalised to nonlinear problems by adding the **activation function** σ to each node after the mapping

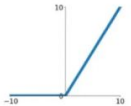
$$\sigma(\vec{y})$$



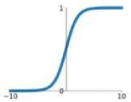
Simple example:
1 hidden layer
with 2 nodes

- Different possible activation functions: **ReLU** is one of the most common, but also sigmoid, hyperbolic tangent, …

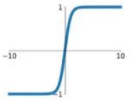
ReLU
$\max(0, x)$



Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$

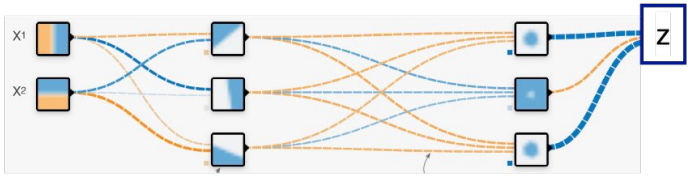


tanh
$\tanh(x)$



Building neural networks

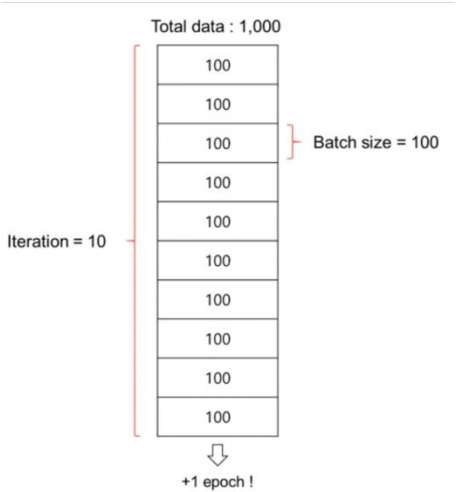
- If there are more layers the output of the previous layer is the input of the next one
- In each layer **W and b** are the adjustable parameters of the network



- The final output is called **prediction**
- **The tuning of parameters to get the most correct prediction is called training**

- In a **regression task** the network is like a function and returns an continuous output z in response to an input value x (z=f(x))
- In a **classification task** the output is the assignment of the input into a category (called class)

Training, validation and test



- The network training is an **iterative procedure** in which the parameters are optimised in many small steps
- **Epoch :** a complete use of the training data

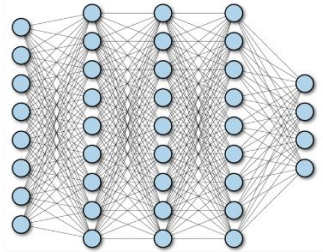
➜ **the number of epochs is the number of reuses of all the training data**

- **Minibatch (or batch) :** a subset of training data which is used in a single training step to make the network less time-consuming and more efficient

- Only a subset of the data set should be used for training ➜ the rest to evaluate the performances of our model
- There are three subsets (data can be split to 1/3 per each):
  1) The **training data** set is used to train network
  2) The **validation data** set is used at the end of each epoch to check the quality of the predictions
  3) The **test data** set is used when the parameters are already optimised and is used to estimate the final quality of the network predictions
- If the number of events is limited, the test and validation sets are typically reduced (e.g. 60% for training, 15% for validation, 25% for test)

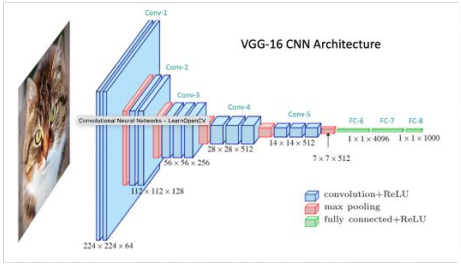
Different kinds of neural networks

- **Fully-connected neural networks (FCN):**
  - Basic network architecture, built as described in the previous slides
  - Network with several layers in which each node is connected to all the nodes of the following layer
  - In each node a linear mapping is performed, then the activation function is applied

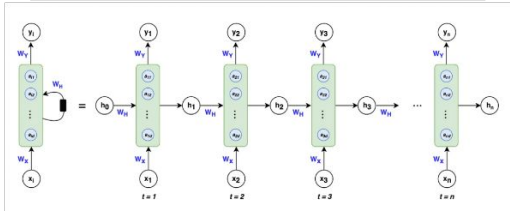


There are also many other neural network architectures, such as…

- **Convolutional Neural Networks (CNN):**
  - Standard approach for image-like data
  - Use the symmetries within the images to simplify the numerical problem (more efficient)



- **Recurrent Neural Networks (RNN):**
  - Suitable for sequential data (e.g. time series)
  - Internal memory that allows to process sequences of different lengths



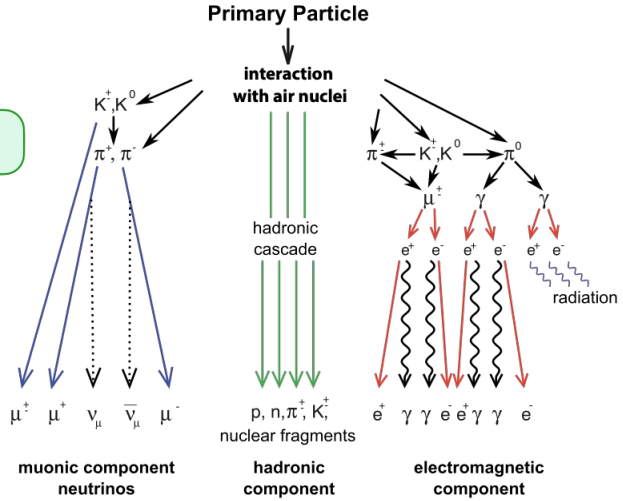
CNNs applied to a photon search in Auger

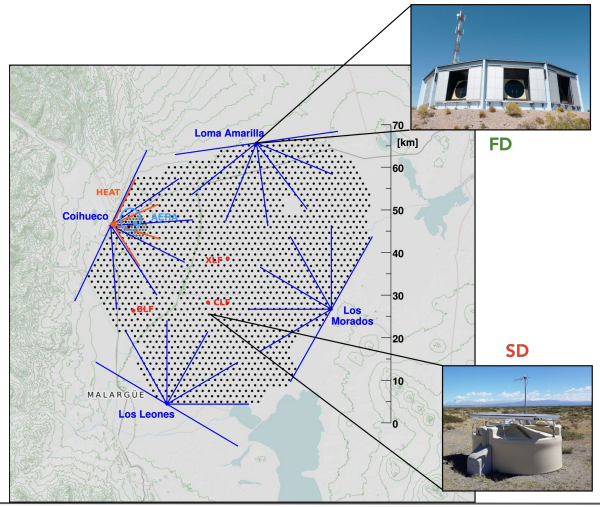
Eleonora Guido

Extensive air showers

Very low rate of particles at **ultra-high energies** → detection through **extensive air showers** (EAS)

- UHE particles start interacting with atmospheric nuclei (N, O, Ar)
  → cascades of **ionised particles** + **electromagnetic radiation**
- Cascades observed by ground-based detectors, like the Pierre Auger Observatory
  → **the type of primary particle can be inferred by the air shower characteristics**

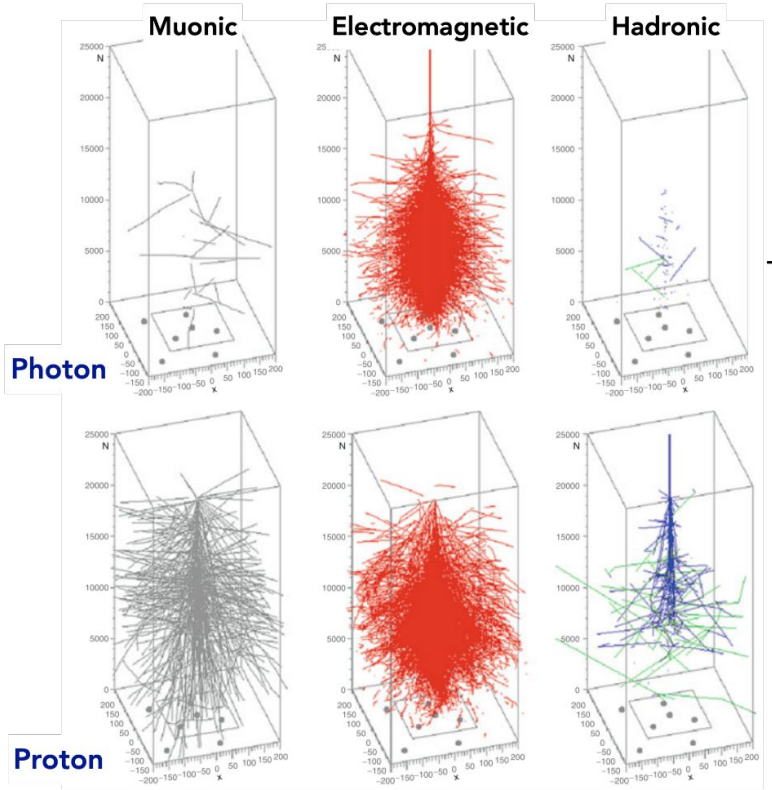




The Pierre Auger Observatory

Hybrid detector (Surface Detector (**SD**) + Fluorescence Detector (**FD**)) located in Argentina, close to Malargüe (~1400 m a.s.l.)

UHE photons at the Pierre Auger Observatory

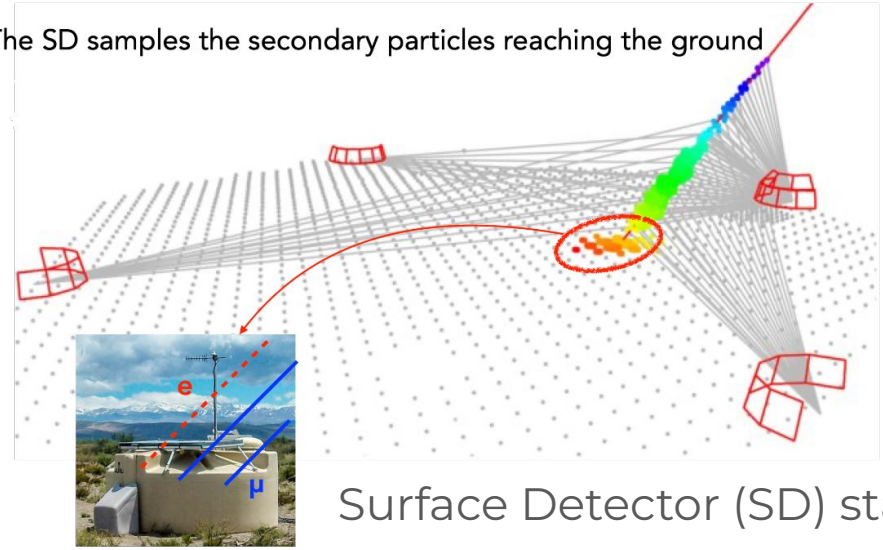


Also photons produce air showers in atmosphere

Photon showers differ in particle content and development

- **Very low muon content, electromagnetic component dominant**
- The maximum development is reached deeper in atmosphere

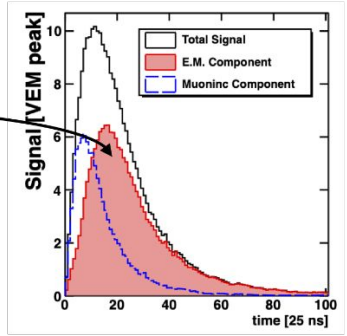
The SD samples the secondary particles reaching the ground



Surface Detector (SD) station

Signal traces of SD stations

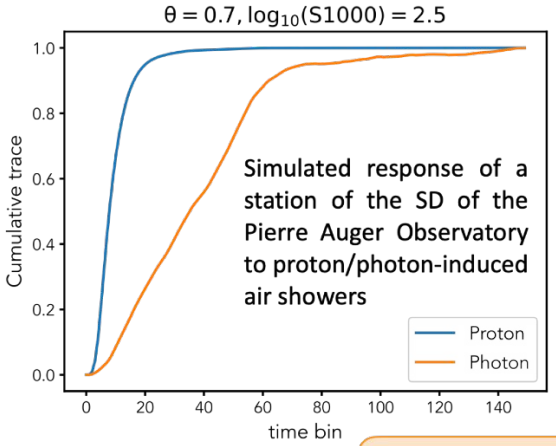
- Main idea: using the **SD signal shape of the FADC traces** to distinguish photon-induced EAS
- The **electromagnetic component** arrives later and it is more spread in time



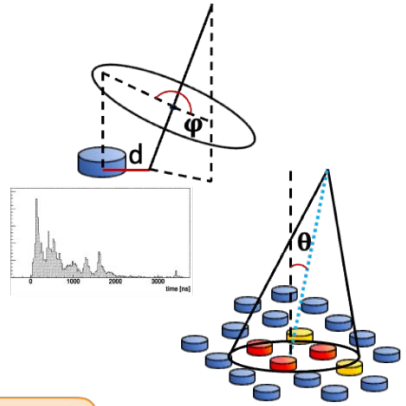
- Distinctive features in the **whole time range of the traces** **(both rising and falling parts)**

→ **cumulative traces** of SD stations can be considered

→ **first 150 time bins are used as input variables**

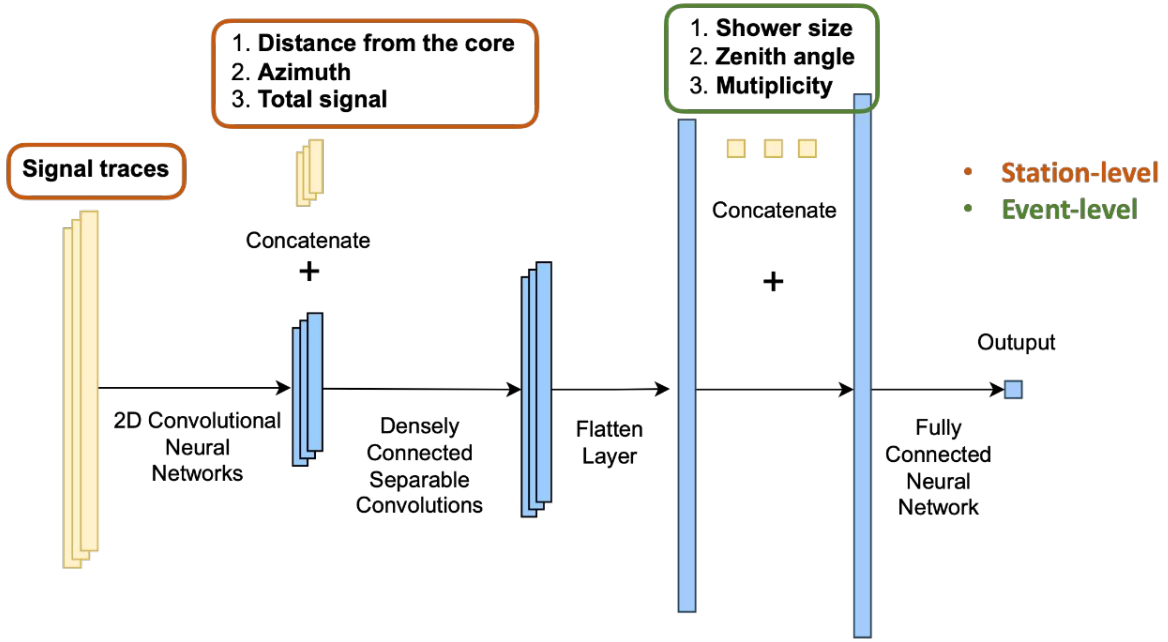


- The signal traces of the **three stations recording the highest signals** are selected
- Other dependency of the signal:
  - **Station-level**: shower geometry (distance from the shower axis d, azimuth in shower plane coordinates **φ**), total recorded signal S_{tot}
  - **Event-level**: zenith angle **θ**, shower size (energy estimator), number of triggered stations N_{stat}



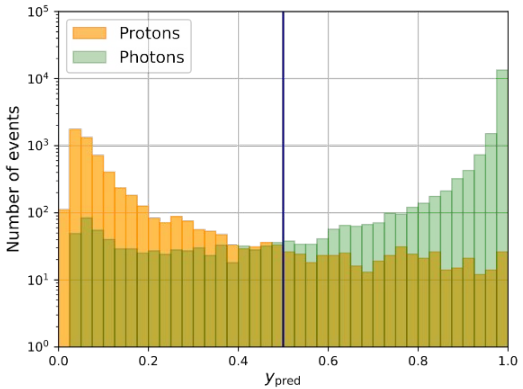
→ **deep learning techniques are used to deal with hundreds of variables**

Convolutional Neural Network



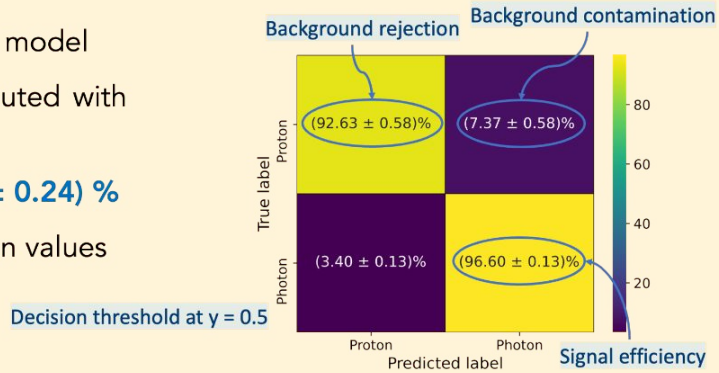
- In the first part of the network the signal traces and the other station-level variables are analyzed with convolutional layers
- The information at the level of the event is then concatenated and analyzed with a fully connected neural network
- **Output: value between 0 and 1 to classify events as photons (closer to 1) or protons (closer to 0).**

Results



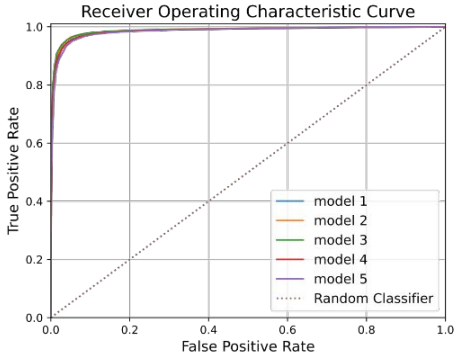
- Metrics are evaluated for each model
- The average results are computed with their uncertainty:

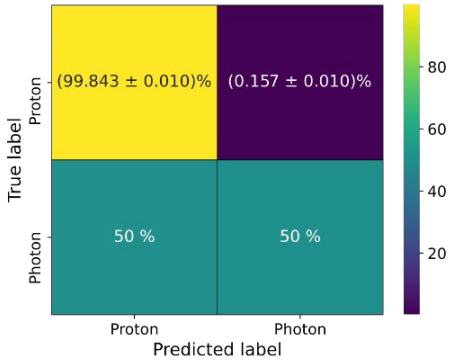
  ➜ **Balanced accuracy: (94.62 ± 0.24) %**
- Confusion matrix with the mean values



- In order to compare our results with other Auger analysis we are interested in **the background rejection at 50% signal efficiency**
- The decision threshold y = 0.99 provides a 50% signal efficiency

  ➜ **Background rejection is (99.843 ± 0.010)%**
- Similar to Auger analyses with hybrid events (99.1-99.9 %)



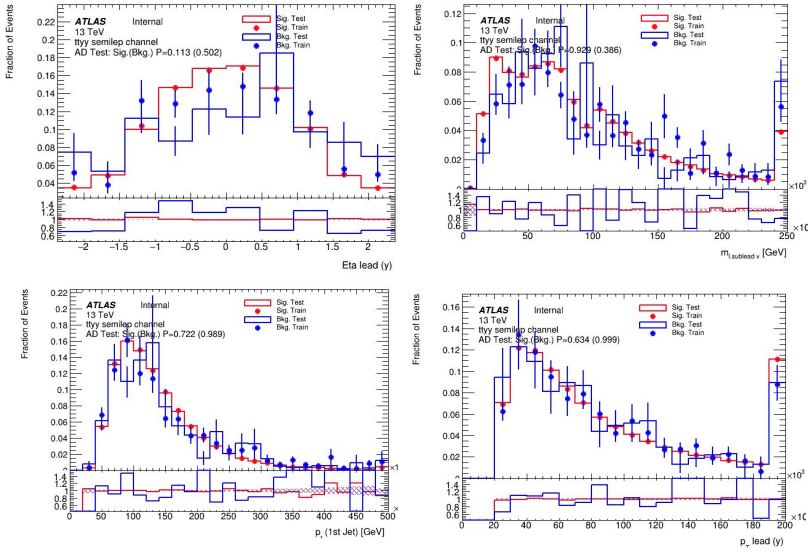


Signal/Background separation using BDT and hyperparameter optimization

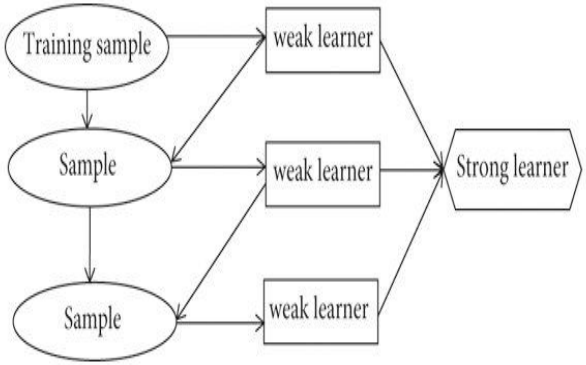
Arpan Ghosal

Introduction

- In HEP, datasets with diverse variables challenge signal-background discrimination - most events not characteristic of either signal or background.
- **Boosted Decision Trees** build on cut based selection, extending the search for signal-background discrimination to multivariate regime.
- **Idea -** combine many weak classifiers (trees) to form a powerful discriminant. "*Nuance over exclusion"* - Idea is to consider all events, even those failing individual criteria. BDTs evaluate multiple criteria, avoiding outright event rejection based on single failures, continuously refine classification and minimize misclassifications over successive iterations.
- **Transparency and Decision Clarity -** Simpler formulation, clearer boundaries and easier optimization compared to other "black-box" NN techniques.
- **Impressive Success Stories in HEP so far -**
  - MiniBooNE at Fermilab for neutrino oscillation anomaly studies
  - D0 experiment at Tevatron for single top search
  - Higgs boson discovery at LHC
  - τ-lepton identification and flavour tagging in ATLAS in LHC Run2
  - particle identifications in T2K, NOvA



Algorithm and Specifications



- BDTs combine multiple weak decision trees to form a strong classifier.
- Each tree is built on the errors of the previous, improving accuracy iteratively.

General Algorithm:

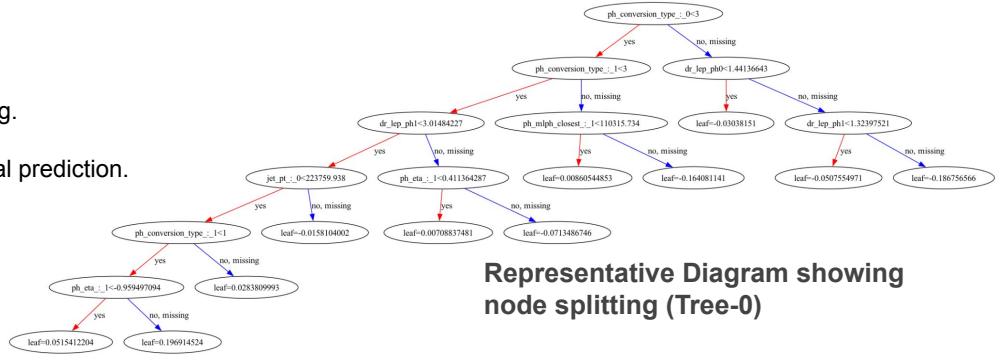
- **Initialization:** Start with a single decision tree.
- **Iteration:** For each subsequent tree, focus on misclassified events by the previous trees.
- **Boosting:** Increase the weight of misclassified events, making them a priority for the next tree.
- **Final Prediction from combination:** Aggregate decision of all trees (e.g. Gradient Boosting). In some algorithms, it's weighted performance (e.g. AdaBoost).

Key (Hyper)Parameters:

- **Learning Rate:** Controls step size.
- **Tree Depth:** Maximum number of splits in each tree.
- **Number of Trees:** Total trees to combine.
- **Splitting Criterion:** Metric to choose best variable and its threshold for each split (e.g., Gini impurity, entropy etc.).

Additional Specifications:

- **Regularization:** Techniques like leaf size constraints to prevent overfitting.
- **Loss Function:** Defines penalty for misclassification.
- **Feature Importance:** Measures contribution of each input variable to final prediction.



Representative Diagram showing node splitting (Tree-0)

Hyperparameter Optimisation

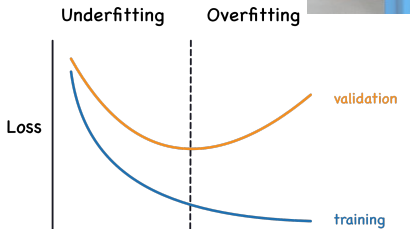
- **Why is it important? OVERFITTING !!**
  Maximize the performance, minimize overfitting.

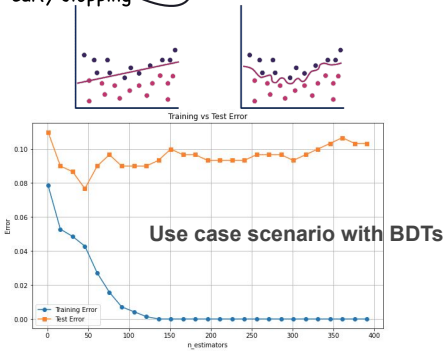
  When model spends much time on the data (model is complex), it starts to learn the noise in the data.

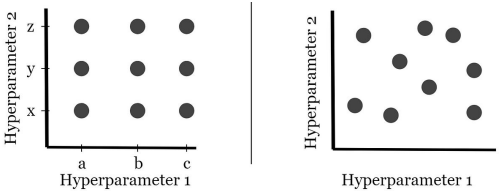
  What follows? Poor generalization to new data. High variance - model is overly sensitive.

  Cross-validation: key strategy to evaluate model performance on unseen data. Split data into sets and use them iteratively for training and validation.

- Manual Optimization: tedious, prone to human bias, inefficient exploration of search space, difficult to scale with complex models. **Try at your own risk!!**
- **Various techniques of Automatised Tuning:**
  - Grid search: can be computationally expensive and inefficient in high-dims.
  - Random search: more efficient in high-dim spaces.
  - Bayesian optimization: uses prior evaluations to inform next set of hyperparameters. “Reason” before “Run”.
  - Others like Gradient-based optimization, Evolutionary Optimization (based on mutations), Early-Stopping based Search etc.
- **Several tools available** - Hyperopt (Bayesian optimization), Ray Tune, Scikit-Optimize (skopt, efficient optimization for ML), SigOpt (commercial, sophisticated interface), Katib (Kubernetes, distributed environments), Google Vizier (internal Google use, black-box optimization), Optuna.







Grid search **Random Search**

Hyperparameter Optimisation with Optuna

Open-source Python library, supports various optimization algorithms (TPE by default), efficiently explores the search space of hyperparameters by intelligently selecting promising values to evaluate, automatically stops unpromising trials to save resources.

1. **Define the Search Space**: Specify the hyperparameters to optimize and their ranges.

```
# Define hyperparameters to optimize
learning_rate = trial.suggest_float("learning_rate", 0.001, 0.1, log=True)
max_depth = trial.suggest_int("max_depth", 3, 10)
max_features = trial.suggest_categorical("max_features", ["sqrt", "log2"])
n_estimators = trial.suggest_int("n_estimators", 100, 1000)
```

2. **Objective Function**: Create a function that returns the model's performance metric to be optimized.

```
# Calculate the accuracy of the model
accuracy = accuracy_score(y_valid, y_pred)
```

3. **Optimization**: Create study object to organize the results. Optuna iteratively suggests hyperparameter values, evaluates them using the objective function, and refines its suggestions based on past results.

```
# Create a study object for optimization
study = optuna.create_study(direction="maximize")  # maximize accuracy

# Optimize the objective function
study.optimize(objective, n_trials=1000)  # You can adjust the number of trials
```

4. **Best Parameters**: After completing the trials, Optuna identifies the hyperparameter set that resulted in the best performance.

```
# Get the best hyperparameters
best_params = study.best_params
best_accuracy = study.best_value
```

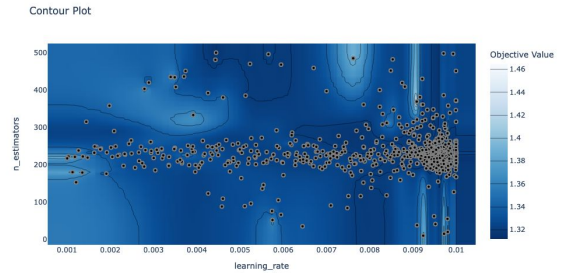
Example code on Github

Visualization with Optuna

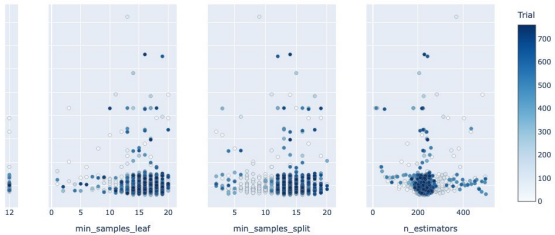
- Study results can be stored as structured database.
- Plotly - helps with visual plots of Optuna's search space.

```
study = optuna.create_study(study_name='trial1',load_if_exists=True,storage="sqlite:///./results.db",
direction="minimize")  # minimizing loss here
```

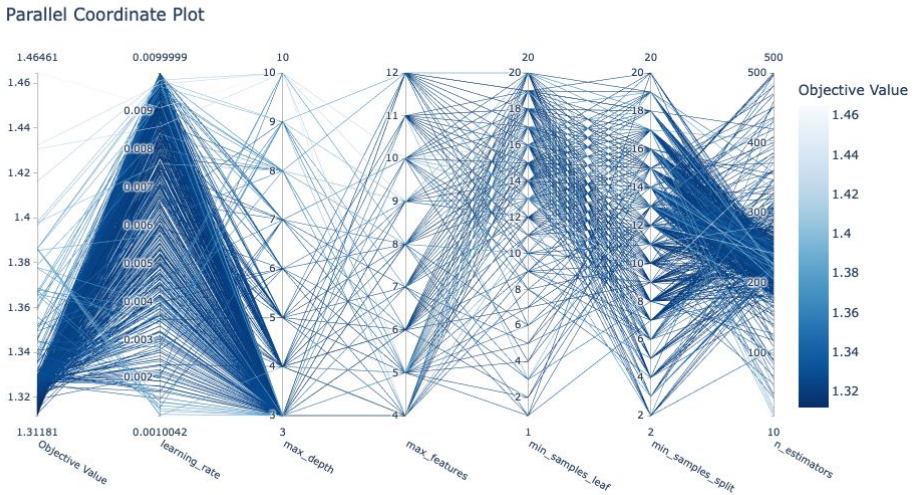
```
from optuna.visualization import plot_contour

plot_contour(study, params=['learning_rate', 'n_estimators'])
```




```
from optuna.visualization import plot_slice

plot_slice(study)
```




```
from optuna.visualization import plot_parallel_coordinate

plot_parallel_coordinate(study)
```

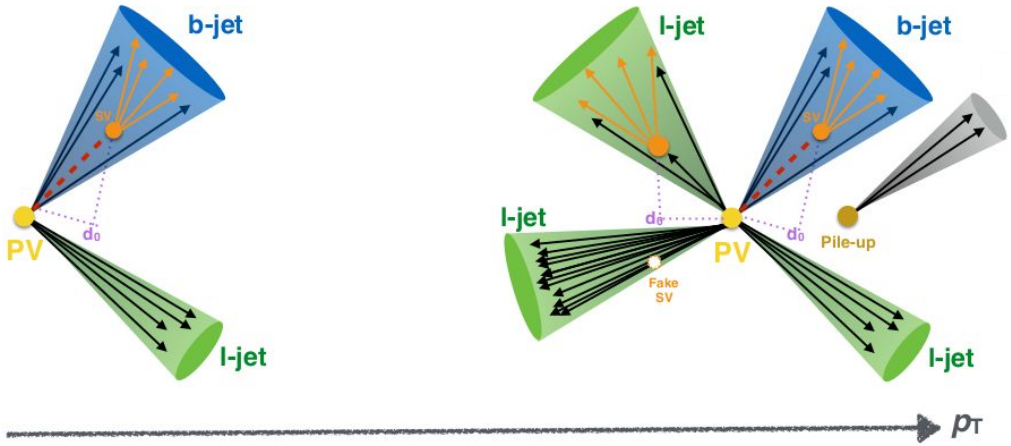


Example code on Github

Jet flavour tagging using Deep Sets and GNN

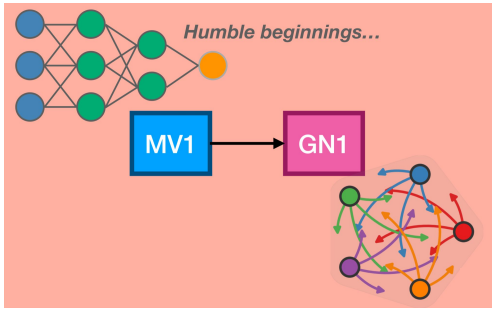
Diptaparna Biswas

History of jet flavour tagging at ATLAS



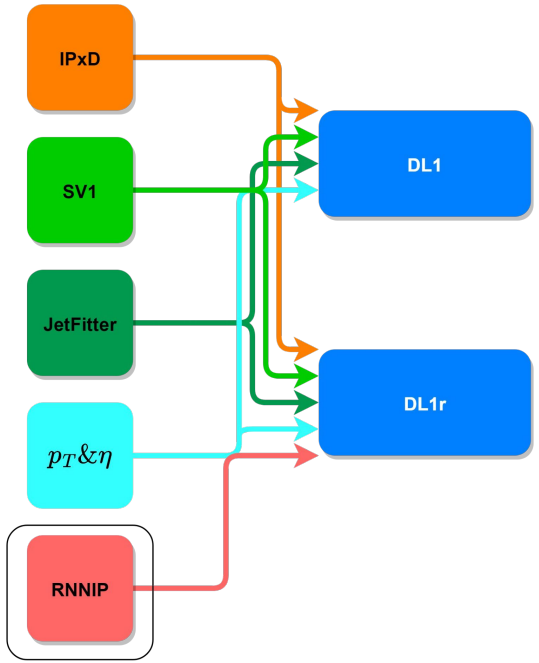
Things get complicated as we go to higher p_T

- B-hadrons have long lifetime (~1.5 ps)
  - ➢ B-taggers exploits this feature.
    - Search for secondary vertex.
- But a (fake) secondary vertex can get reconstructed!



Manually optimized low-level taggers

- Jet and track inputs are fed to low level taggers: IPxD, SV1 and JetFitter
- Output of these taggers are fed into a high-level tagger: BDT (MV2) or DNN (DL1)
  - ➢ We get classification scores: p_b, p_c and p_l
- **Can we feed the low-level jet and track features directly into a DNN?**
  - ➢ A jet can contain variable number of tracks.
  - ➢ RNN can handle variable sequence length, right?
    - This gives us DL1r (along with low-level taggers)
    - Yes, but it's not *permutation invariant!*



Track-based Neural Network

Mathematical interlude: Deep sets

- $\square = \{x_1, x_2, x_3, \ldots\}$ (unordered collection of objects)
- Any such function $f : \mathbb{R} \to \mathbb{R}$ can be decomposed into:
  - $\phi : \mathbb{R} \to \mathbb{R}^n$
  - $\rho : \mathbb{R}^n \to \mathbb{R}$
- Examples:
  - $f \equiv$ count
    - $\phi(x) = 1$, $\rho(v) = v$
  - $f \equiv$ sum
    - $\phi(x) = x$, $\rho(v) = v$
  - $f \equiv$ average
    - $\phi(x) = [x, 1]$, $\rho(v) = v[0]/v[1]$

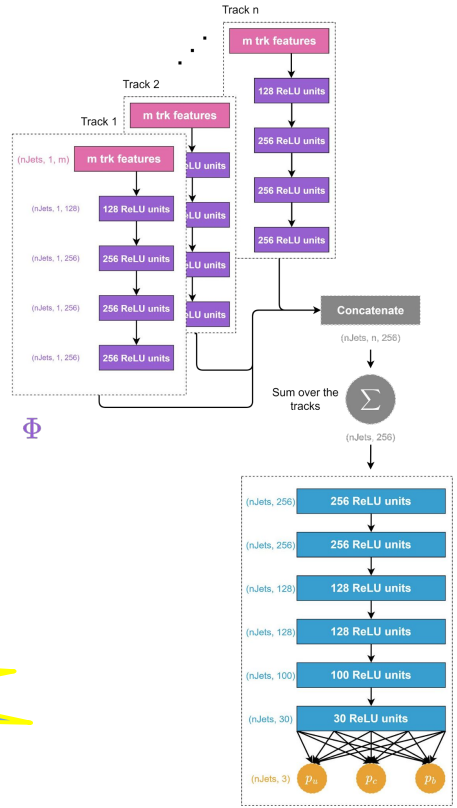
$$f(\chi) = \rho\left(\sum_{x \in \chi} \phi(x)\right)$$

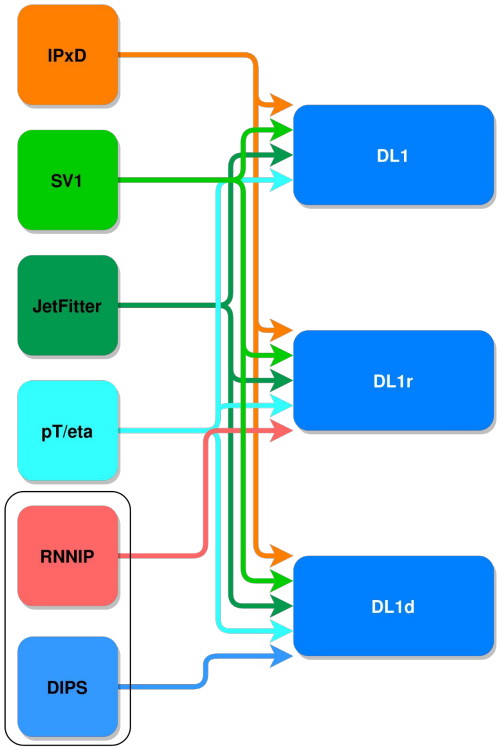
arXiv:1703.06114 arXiv:1810.05165

Deep impact parameter sets (DIPS)

- How to get ϕ and ρ ?
  - ➢ Approximate them using DNN.
  - ➢ This gives us **DIPS**.
  - ➢ DIPS, along with low-level taggers, gives us **DL1d**.
- Can we get rid of the low-level taggers?

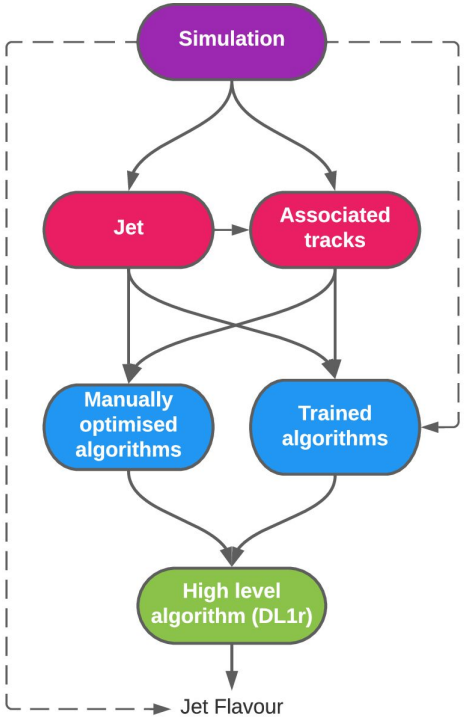
Introducing GN1

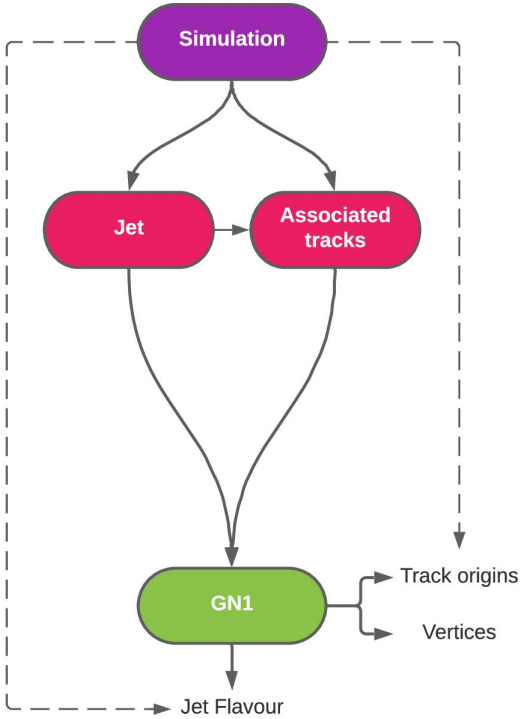




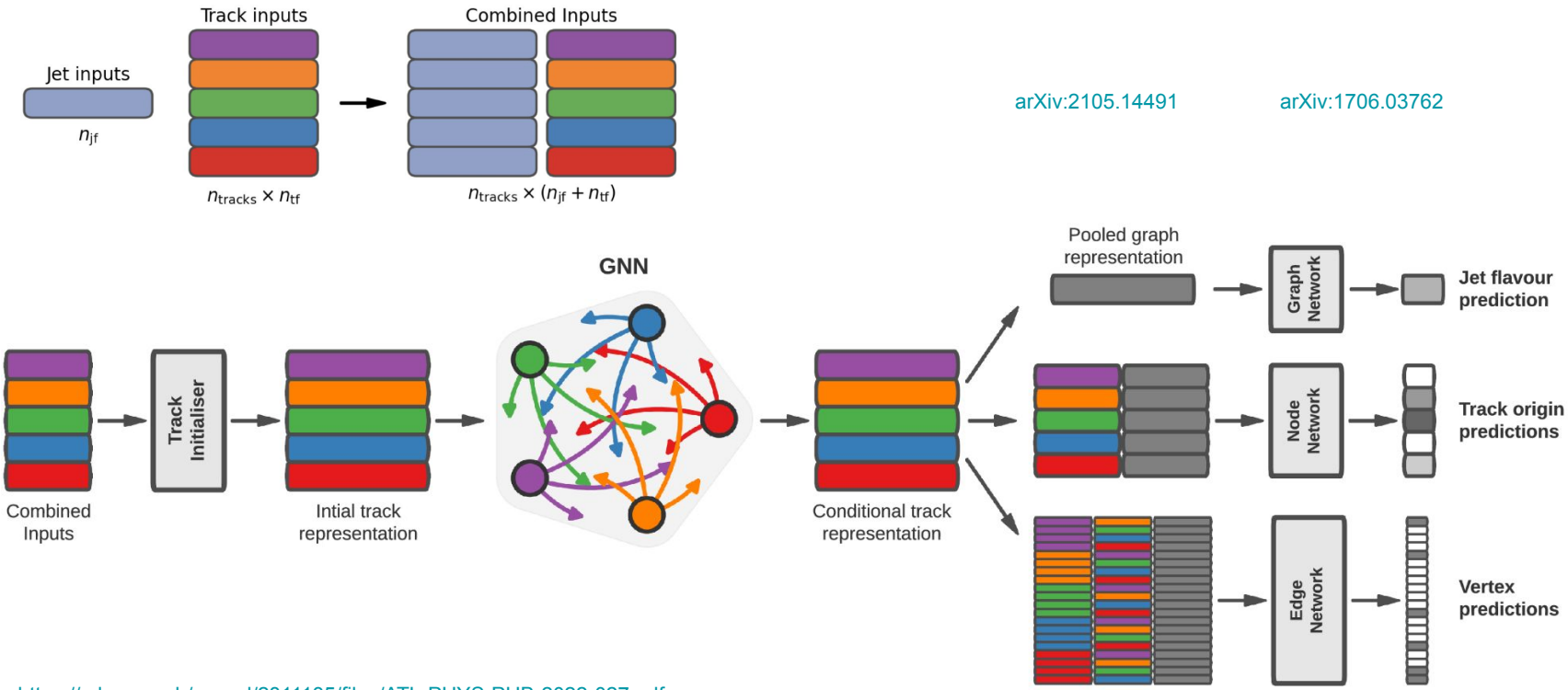
Track-NN based

Graph neural network based “all in one” tagger





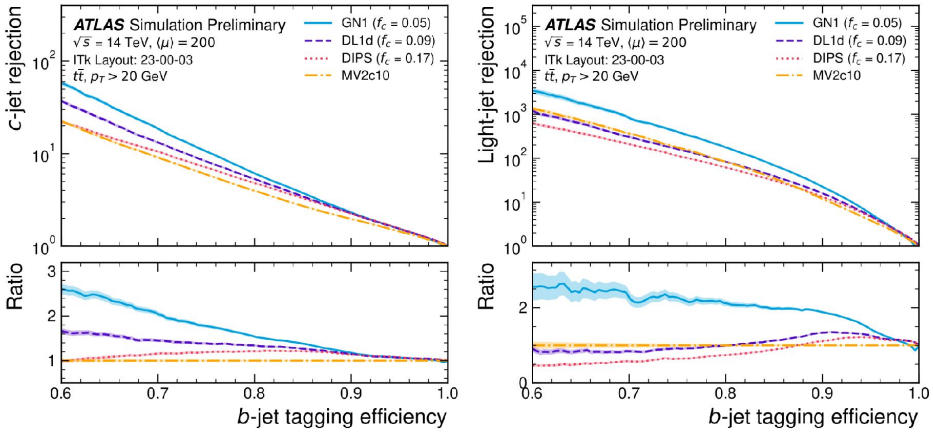
Architecture of GN1/GN2 models

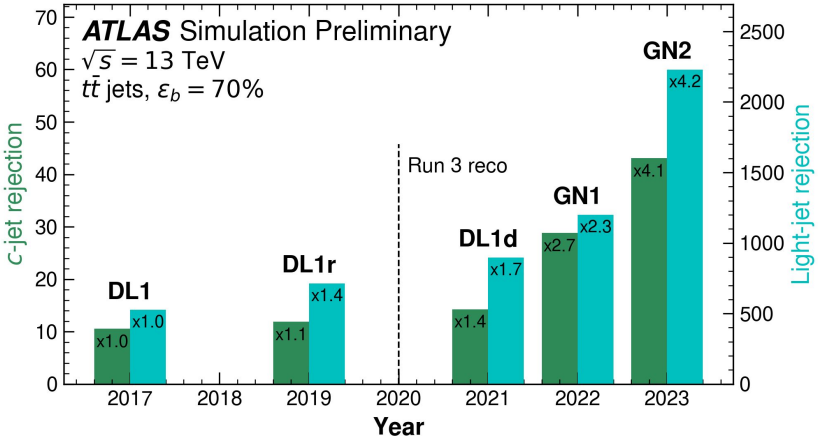


arXiv:2105.14491 arXiv:1706.03762

https://cds.cern.ch/record/2811135/files/ATL-PHYS-PUB-2022-027.pdf

Performance of GN1 and GN2





Thank You!

Backup slides

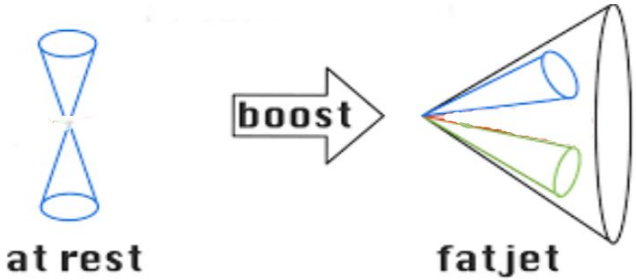
Double-Charm Tagging using Event Shapes

Gilberto Tetlalmatzi-Xolocotzi

- The main goal is improving our sensitivity towards the measurement of the charm Yukawa.
- In general, bottom- or charm-taggers are designed to find jets initiated by individual b or c quarks.
- In searches for light or boosted resonances that decay into a charm or bottom pair, such algorithms might not be ideal, as they neglect correlations between the decay products.
- To increase the sensitivity in searches for new physics or Higgs boson measurements it can be beneficial to design dedicated 2-prong reconstruction algorithms that allow to utilise more information about the decaying resonances.
- Observables that are particularly sensitive to the radiation profile of the event are so-called event shape observables.
- We developed a procedure to identify jets initiated by $c\bar{c}$ pairs from Higgs boson decays based on the application of different event shapes and the transverse momenta of leptons (e and μ).

Signal channel $pp \rightarrow H(\rightarrow c\bar{c})Z$

Background channel $pp \rightarrow Z + \text{jets}. \quad Z \rightarrow l^+l^-, \text{ for } l = e, \mu$



A boosted Higgs decaying into a pair $c\bar{c}$ produces a jet with a large active area $R_{c\bar{c}}$

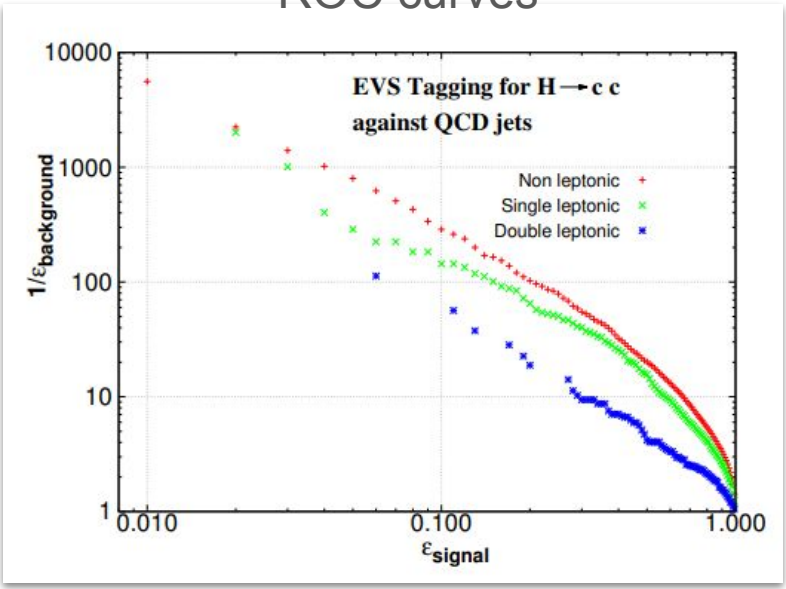
Reconstruct the Z boson in the highly boosted regime using dileptonic final states.

We used TMVA and a Boosted Descition Tree as our multivariable classifier

Main discriminating observables

Z + jets		
Non leptonic	**Single leptonic**	**Double leptonic**
C parameter	Cone total jet mass	Global thrust minor defined with jets outside the dijet region dijet region
Cone heavy jet mass with exponentially suppressed forward term	C parameter	C parameter
Thrust major	Thrust major	$P_{T,e}$
3-jet resolution y_3 Durham (P-scheme)	3-jet resolution y_3 Durham (P-scheme)	3-jet resolution y_3 Durham (P-scheme)
Fractional energy correlation $x = 1.5$	Transverse sphericity	Transverse sphericity
Global thrust minor defined with jets outside the dijet region	$P_{T,\mu}$	$P_{T,\mu}$
Transverse sphericity	$P_{T,e}$	

ROC curves



Results: Optimal tagging point

$$\varepsilon_{c\bar{c}} = 0.40 \quad \varepsilon_{QCD,jets} = 0.03$$

$$\sigma_{pp\to H(\to c\bar{c})Z} = 0.08 \text{ fb} \quad \sigma_{pp\to Z+\text{jets}} = 23.56 \text{ fb} \quad \int \mathcal{L}dt = 3000 \text{ fb}^{-1}$$

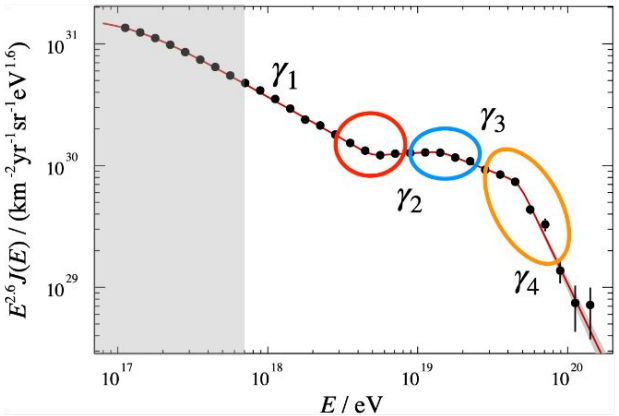
$$\mathcal{S}/\sqrt{\mathcal{B}} = 2.0$$

Combined interpretation of data measured at the Pierre Auger Observatory

Eleonora Guido

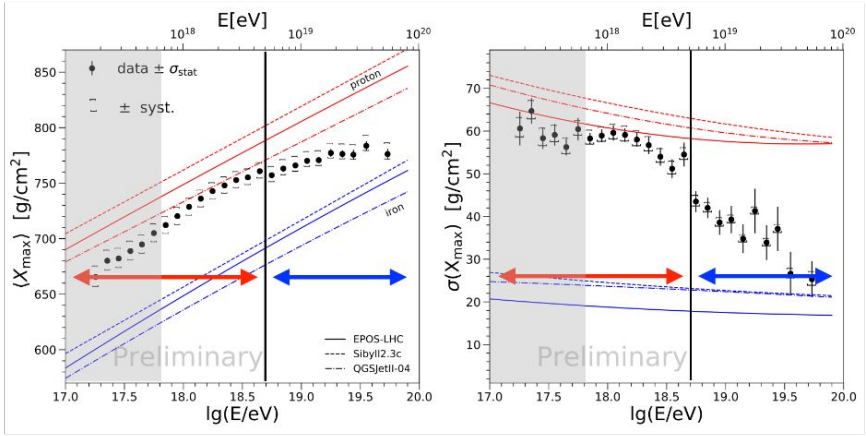
Measurements of UHECRs at the Auger experiment

Energy spectrum for the events measured with the SD array



The X_{max} distribution in each energy bin is sensitive to the mass composition

→ first two moments shown for figurative purposes



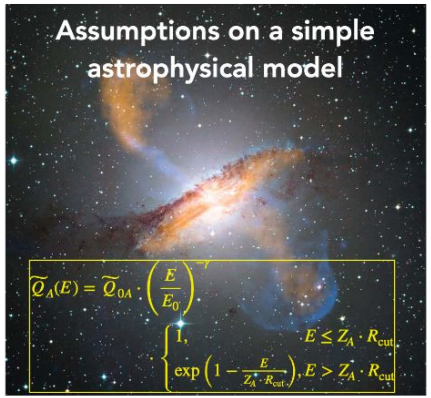
Combining the information from the two data sets is crucial to interpret the features

- We aim at including the ankle region
- We want to focus on the energy region where the Galactic CRs are not dominant anymore

} **Data above E ~ 6 x 10^{17} eV are considered**

Combined fit of Auger data

CR$_S$ EJECTED BY EG ACCELERATORS

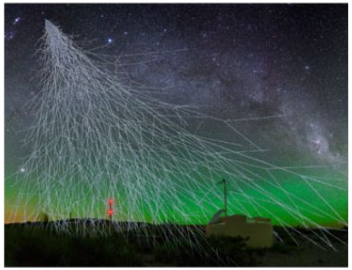


Choice of propagation models for uncertain quantities

PROPAGATION THROUGH THE INTERGALACTIC MEDIUM

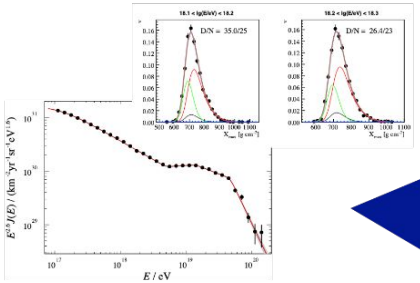
Choice of hadronic interaction model

PRODUCTION OF SHOWERS IN THE ATMOSPHERE



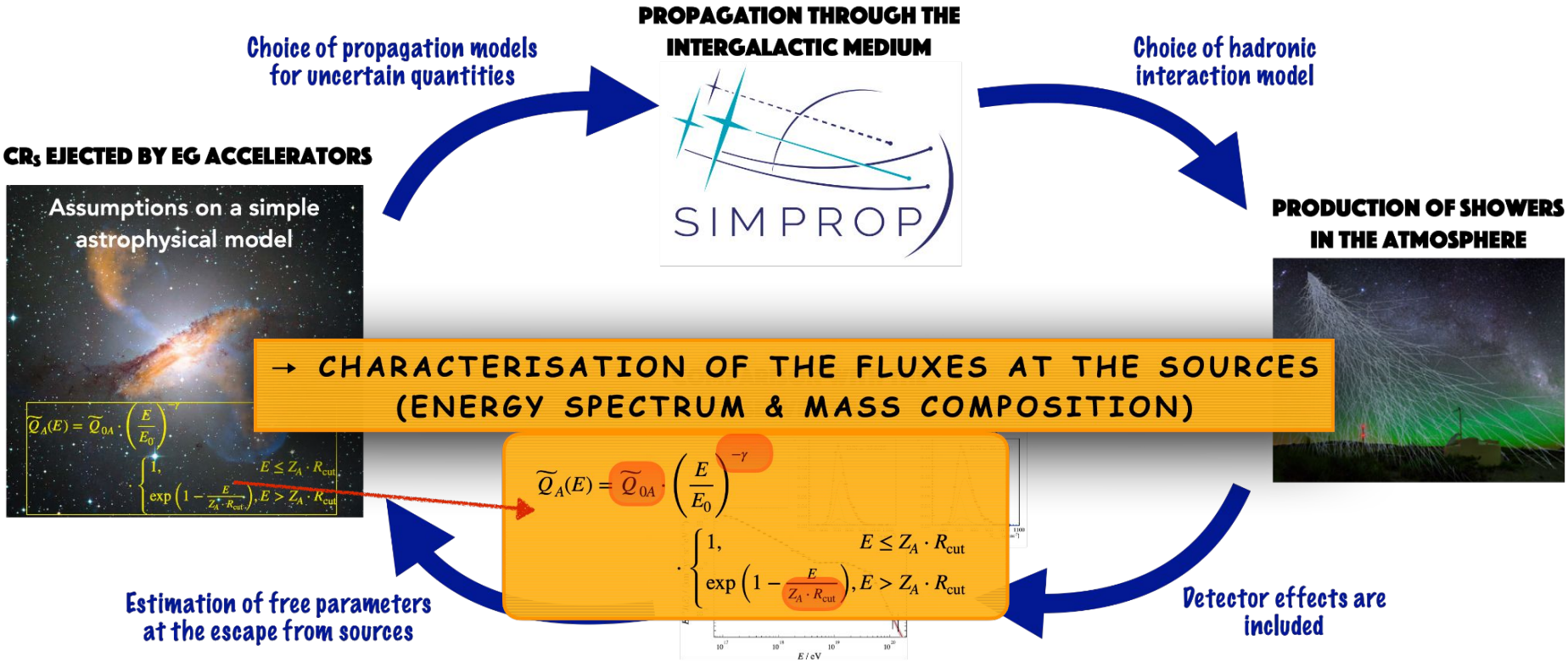
Detector effects are included

COMPARISON WITH THE OBSERVED DATA



Estimation of free parameters at the escape from sources

Combined fit of Auger data



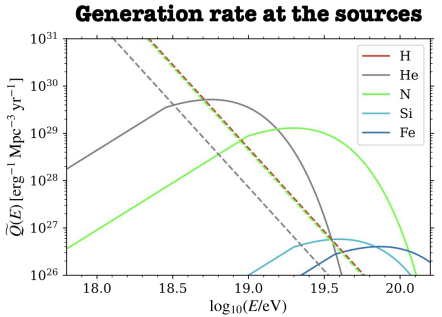
Fit procedure

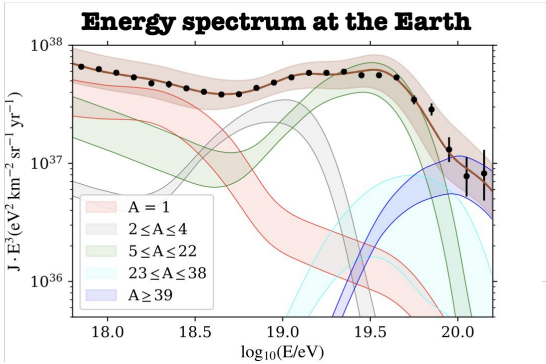
Combined fit of the energy spectrum and X_{max} distributions above $\sim 6 \times 10^{17}$eV

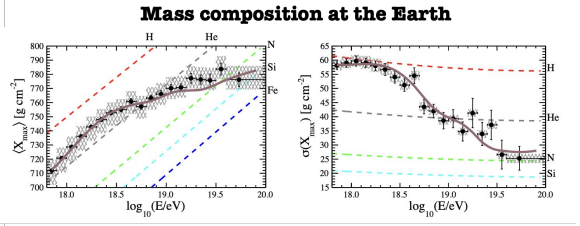
→ compare simulated and measured fluxes at the Earth with the **maximum likelihood method**

$$D = D(J) + D(X_{max}) = -2\ \ln\left(\frac{\mathscr{L}}{\mathscr{L}^{sat}}\right) = -2\ \ln\left(\frac{\mathscr{L}_J}{\mathscr{L}_J^{sat}}\right) - 2\ \ln\left(\frac{\mathscr{L}_{X_{max}}}{\mathscr{L}_{X_{max}}^{sat}}\right)$$

- This analysis is performed with a **program written in C++** and making use of the **ROOT libraries**
- The **Minuit package within ROOT** is used for the minimization and for estimating most of the parameter uncertainties





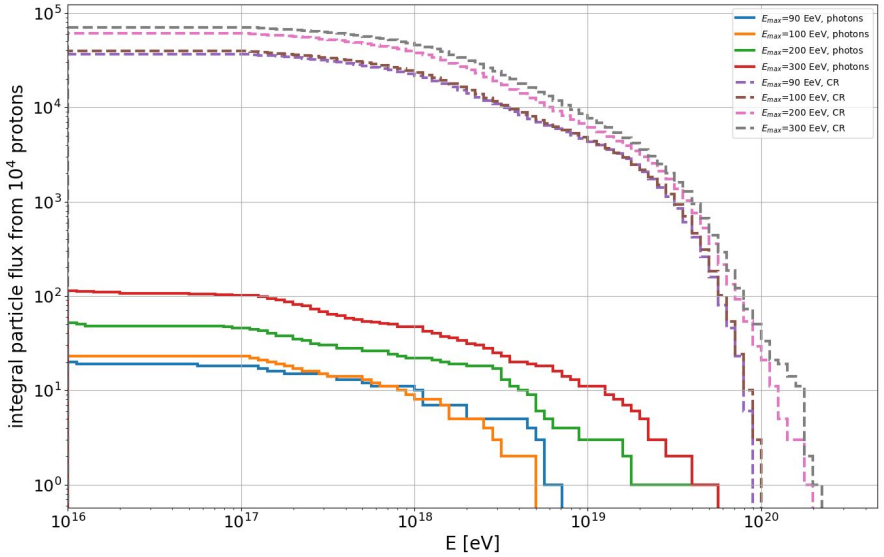


Data visualization
Additional plots

Chaira Papior and Fabian Dünkel

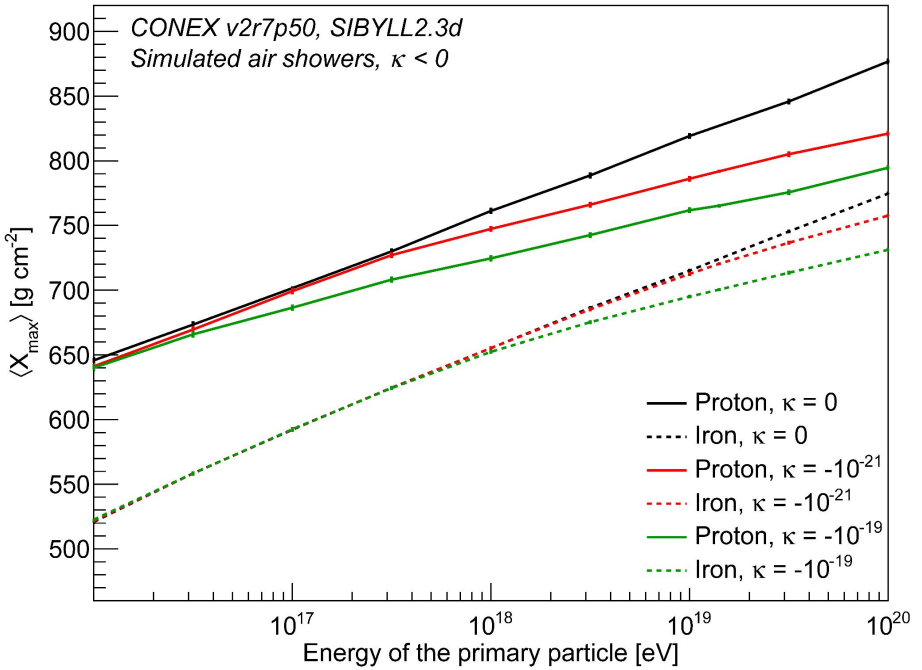
Searching for secondary photons

- Besides primary photons, we can also search for secondary photons produced during the propagation of protons compared to the total flux of cosmic-ray particles
- A histogram is used to show the integral flux of particles depending on initial proton energy
- Plot is generated using pyplot and numpy.histogram



Testing Lorentz violation using air showers

- Testing the impact of LV on the development of simulated air showers
- With LV parameter $\kappa < 0$, photon decay above a threshold possible
- LV accelerates shower development, showers become shorter

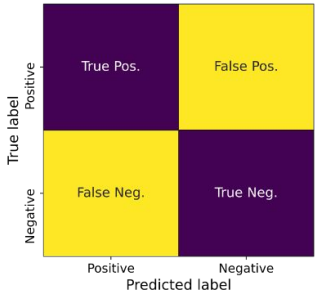


CNN back-up

Eleonora Guido

Measures of evaluation in a classification task

- In a binary classification case each object can either belong or not belong to a target class
- A **confusion matrix** is a table summarising the predictions against the actual class labels in the data set
- **Accuracy**: rate of correct classifications $Accuracy = (TP + TN)/(TP + TN + FP + FN)$
- **Sensitivity or efficiency**: true positive rate (number of correct assignments to the target class divided by the total number of samples of that class
- **Precision**: the ratio between the number of true positive samples to the total number of samples classified as positive (either correctly or incorrectly)



$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{FP + TN}$$

$$Precision = TP/(TP + FP)$$

- The **Receiver Operating Characteristic (ROC)** curve plots the TPR (true positive rate) vs the FPR (false positive rate)
- The **Area Under the Curve (AUC)** is its integral, the overall probability that the network assigns a positive element to the positive class rather than a negative one