THE UNIVERSITY of EDINBURGH

Machine learning for Higgs physics

Liza Mijović, Chancellors Fellow, PPE

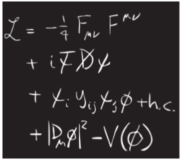
IPNP topical AI/ML meeting, Apr 2025

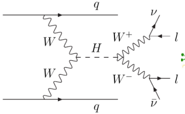
Experimentalist's problem

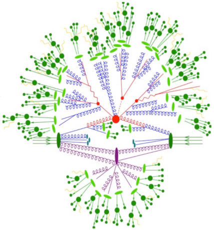
Estimate likelihood:

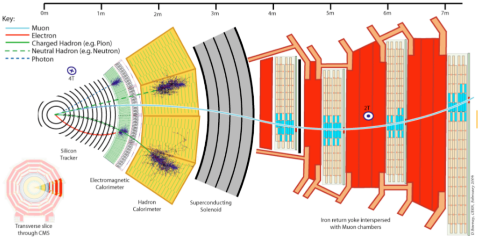
$$\lambda(\mathcal{D};\theta_0,\theta_1) = \prod_{\mathbf{x}\in\mathcal{D}} \frac{p_{\mathbf{X}}(\mathbf{x}|\theta_0)}{p_{\mathbf{X}}(\mathbf{x}|\theta_1)},$$

where θ are theory parameters, $\mathbf{x}$ is high-dimensional, $p_{\mathbf{X}}$ intractable.









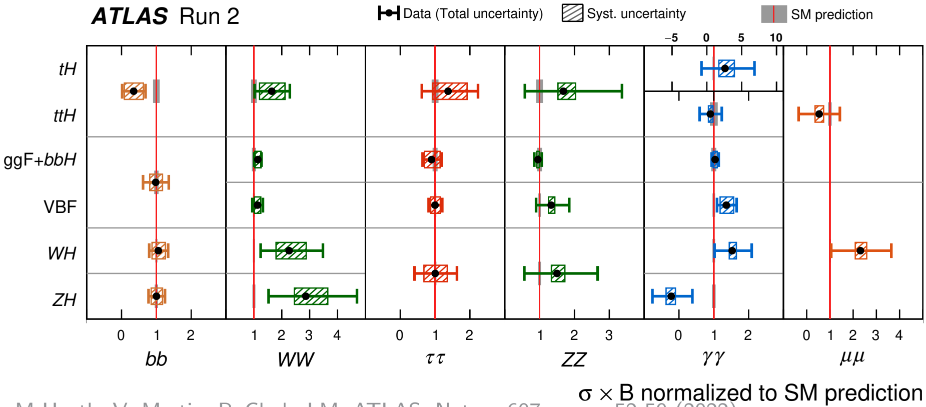
Outline

- **Machine learning in a Higgs analysis we've published.**
  - Focus on: ML reweighting (transforming distribution A to B).
  - Explain ML reweighting with Calibrated Discriminative Classifiers (carl).

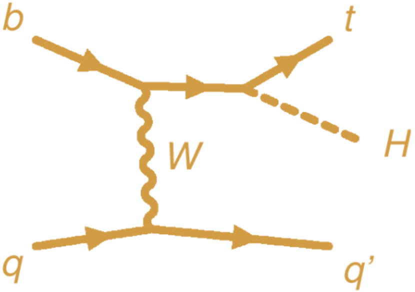
- **How does ML reweighting apply to my current Higgs physics work?**
  - Better theory modelling of Higgs production.
  - Machine learning unfolding.

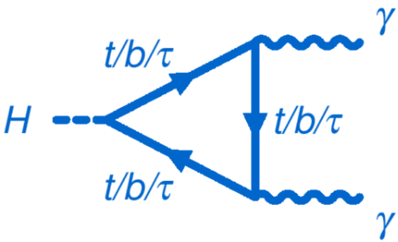
Higgs 10 years after discovery

- Figure: combination of all ATLAS production x-section measurements.
- Focus on: $\mathbf{H \to \gamma\gamma}$ decay channel, and tH production.



Machine learning for tH



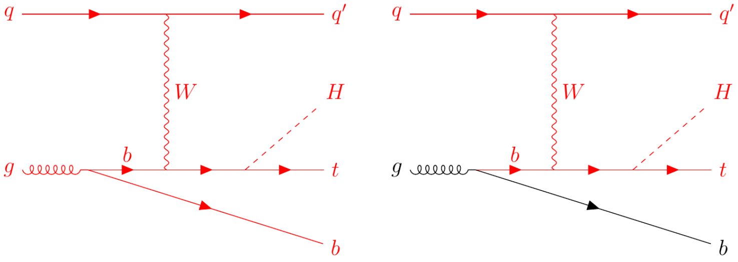


- **S vs B Classification**
  - Adversarial neural network
- **S Classification**
  - Parametric networks for likelihood free inference
  - Fully connected neural nets
  - Feature engineering
- **Modelling uncertainties**
  - Machine Learning (ML) reweighting

ML reweighting: tH

Theory modelling uncertainties:

- Question: impact on your result, if you use model B instead of model A?
- Needs answering after detector simulation and reconstruction.
  Problem: prohibitive computing resource requirements.
- Idea: find weights at parton (/particle) level, such that weighted A $\sim$ B.



(a) *4-flavour event generation.* (b) *5-flavour event generation.*

Figure: M Heath's thesis, https://cds.cern.ch/record/2801233

Reweigh: analytical probability, 1D

Reference:

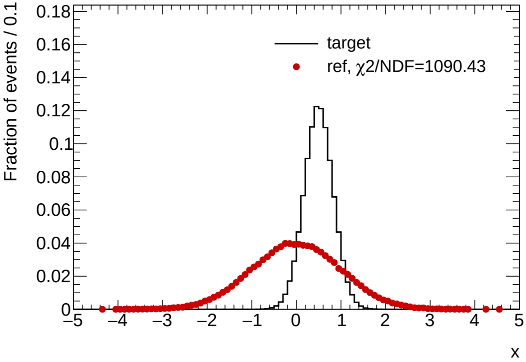
$$\mu = 0, \sigma = 1$$

$$p_0(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

Target:

$$\mu = 0.5, \sigma = 0.3$$

$$p_1(x) = \frac{1}{0.3\sqrt{2\pi}} e^{-\frac{(x-0.5)^2}{2\cdot 0.3^2}}$$



Idea1: reweight reference to target:

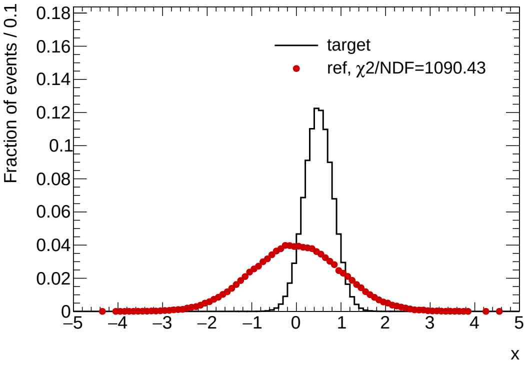
$$p_1'(x) = p_0(x) \cdot w(x) = p_0(x) \cdot \frac{p_1(x)}{p_0(x)}$$

Reweigh: no analytical probability

$$w(x) = \frac{p_1(x)}{p_0(x)}$$

- Idea2: estimate w_i for each bin i.
- Idea3: discriminant classifier trained for p_0 vs p_1:

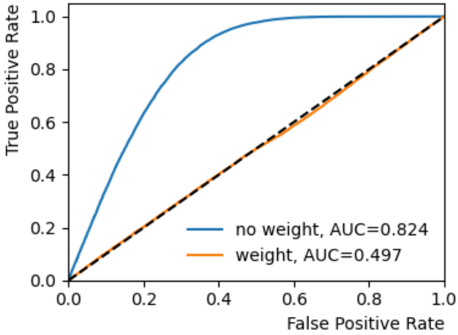
$$w(x) = \frac{1 - D(x)}{D(x)}$$

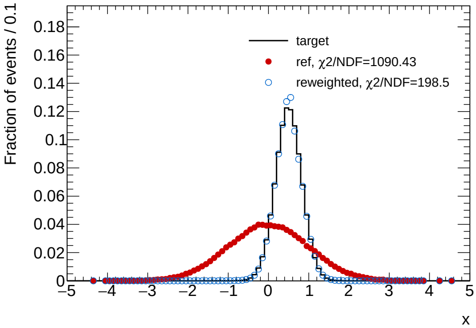


Idea3 from Cranmer et al, arXiv:1506.02169, https://github.com/lmijovic/carl-torch, forked from https://github.com/leonoravesterbacka/carl-torch

Idea3 in practice

- Train DNN classifier to discriminate p0 and p1
- Construct & deploy weight from the discriminant (right fig).
- Left fig: 2nd classifier trying to discriminate between p1 vs (weighted) p0?





Test data from: https://github.com/lmijovic/carl-torch, forked from https://github.com/leonoravesterbacka

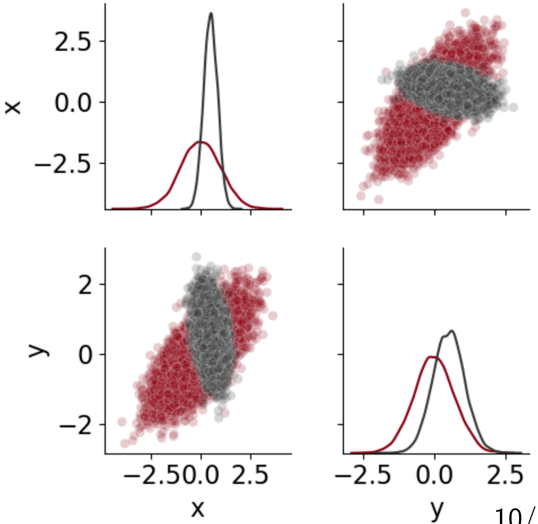
No analytical probability, multi-D

- Idea2:
  estimate w_i for each bin i:
  multi-D histograms

- Idea3:

$$w(x) = \frac{1 - D(x)}{D(x)}$$

  multi-D reweighting $\rightarrow$ 1D



Did we loose any information?

Want to estimate likelihood:

$$\lambda(\mathcal{D};\theta_0,\theta_1) = \prod_{\mathbf{x}\in\mathcal{D}} \frac{p_{\mathbf{X}}(\mathbf{x}|\theta_0)}{p_{\mathbf{X}}(\mathbf{x}|\theta_1)}, \tag{1}$$

Performed: $D : \mathbf{x} \to \mathbf{U}$

Cranmer et al [arXiv1506.02169]: (2) is strictly equivalent to (1):

$$\lambda'(\mathcal{D};\theta_0,\theta_1) = \prod_{\mathbf{x}\in\mathcal{D}} \frac{p_{\mathbf{U}}(u = s(\mathbf{x})|\theta_0)}{p_{\mathbf{U}}(u = s(\mathbf{x})|\theta_1)} \tag{2}$$

if s is strictly monotonic with $\frac{p_{\mathbf{X}}(\mathbf{x}|\theta_0)}{p_{\mathbf{X}}(\mathbf{x}|\theta_1)}$.

Notes on DNN reweighting

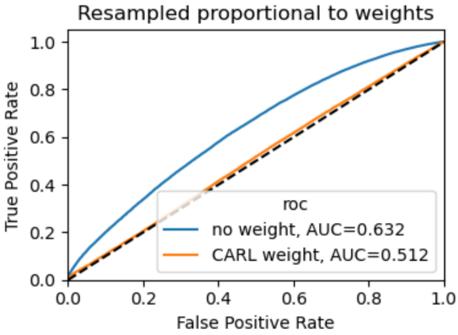
- Further, Cranmer et al [arXiv1506.02169] have shown that $D : \mathbf{x} \to \mathbf{U}$ through supervised learning satisfies the strictly monotonic criterion.

Useful to keep in mind:

- could use other mapping, rather than DNN.
- Idea2 of estimating $p_1(x)/p_0(x)$ could be realised with generative ML, used in modern reweighting techniques.
- Could calibrate the discriminant, and the likelihood estimate would remain equivalent to (1). Ubiquitous in ML reweighting.

DNN reweighting for tH

- Train DNN classifier to discriminate 4fl and 5fl tH
- Construct & deploy weight from the discriminant (right fig).
- Left fig: can discriminate distribution resampled acc to *w* (orange)?



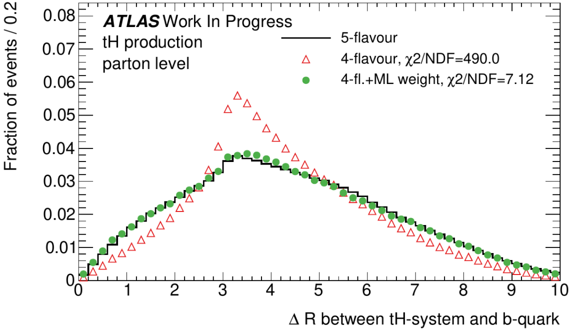
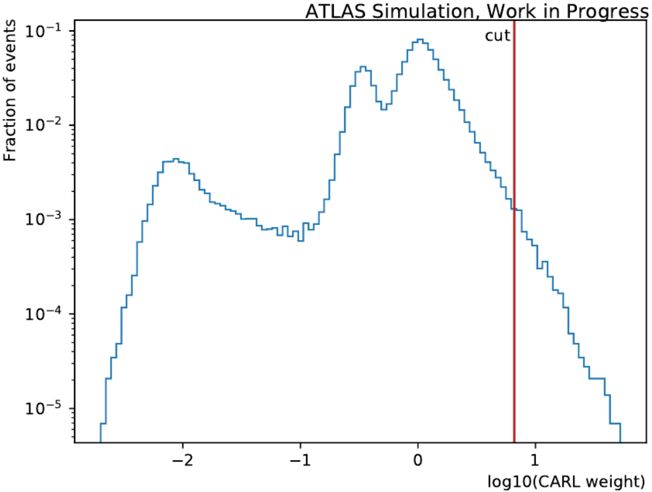


Figure: M Heath's thesis, https://cds.cern.ch/record/2801233

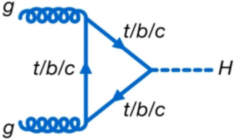
DNN reweighting limitation

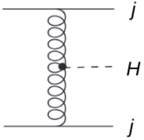
- Large spread of weights.
- Degrades statistical power.

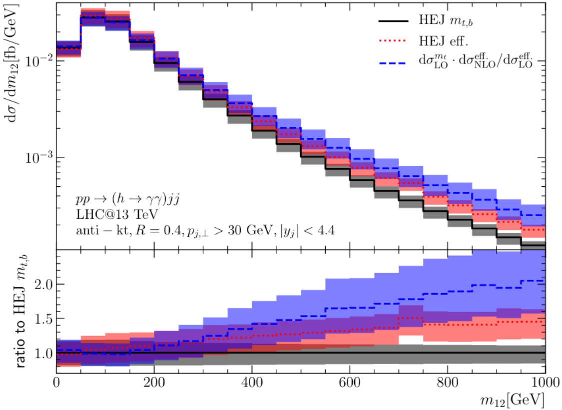


ML reweighting: ongoing work*

- Main constraint on Higgs physics reach at HL-LHC:
  theory modelling uncertainties.
- Improve? ML reweighting to blend models superior in their target phase-space.
- Example: HEJ for ggF.



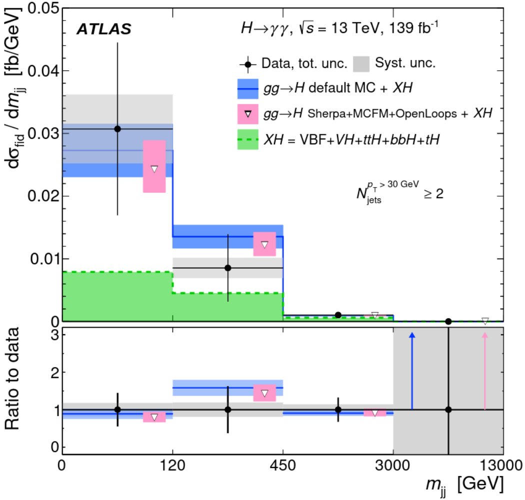




ML reweighting: ongoing work*

- Outcome of LHC data measurements: *unfolded distributions*, corrected for detector effects.
- $$p_{\text{det}}(\mathbf{x}) = \mathcal{R}\, p_{\text{particle}}(\mathbf{x}),$$

  where:
  $\mathbf{x}$ = observable, eg. m_{jj},
  $\mathcal{R}$ = detector response matrix.
- Problem: limits information.



*With Victoria Martin, Alex Sopio, Quaratulain Zahoor. Figure arXiv:2202.00487

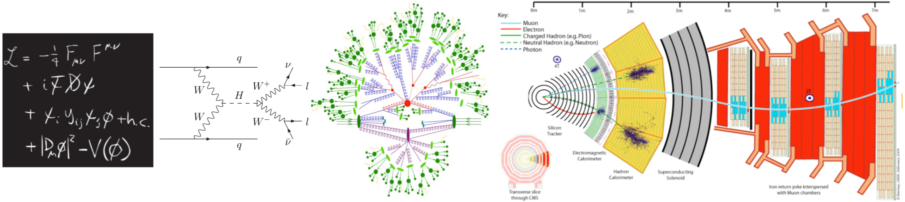
ML unfolding

- Current:

$$p_{\mathrm{det}}(\mathbf{x}) = \mathcal{R}\,p_{\mathrm{particle}}(\mathbf{x}),$$

  where $\mathbf{x}$ = observable (low-dim)
- Q: How would we achieve: $\mathbf{x}$ = full event kinematics (high-dim)?
- A1: weighting with calibrated discriminant classifier ($D : \mathbf{x} \to U$).
- A2: generative algorithm mapping $p_{\mathrm{det}} \to p_{\mathrm{particle}}$.

$p_{\mathrm{particle}}(\mathbf{x})$ $p_{\mathrm{det}}(\mathbf{x})$



Summary

When you need to estimate a **ratio of probability distributions**, you could:

- **Idea1: calculate it analytically.**
  Not applicable to high energy physics experimental data analysis, as $p_{\mathbf{X}}$ intractable.
- **Idea2: estimate $w_i = p_1(\mathbf{X}_i)/p_0(\mathbf{X}_i)$ for each bin i.**
  Not applicable to HEP-ex as $\mathbf{x}$ is high-dimensional.
  NB: you could use a generative model to map $p_1(\mathbf{X}_i)$ to $p_0(\mathbf{X}_i)$.
- **Idea3: use classifier discriminant $D : \mathbf{X} \to 1D$;** $p1/p0 = (1 - D(x))/D(x)$.
  Old idea from [Cranmer et al] (CARL), still getting plenty use & development.