Machine Learning for Hadronization

MLCoffee, LPSC, 26/05/25

Manuel Szewc

University of CINCINNATI

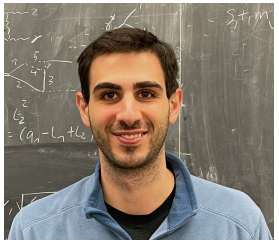
MLHAD

https://uchep.gitlab.io/mlhad-docs/

ICAS

The MLHAD team

Benoit Assi, Christian Bierlich, Phil Ilten, Tony Menzo, Stephen Mrenna, Manuel Szewc, Michael Wilkinson, Ahmed Youssef, and Jure Zupan









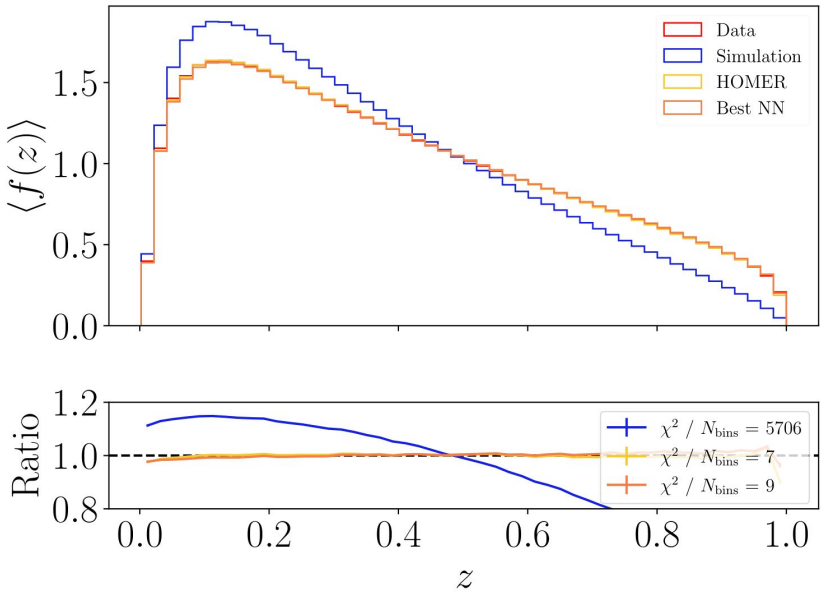
In this talk

Monte Carlo simulations are a fundamental part of **collider physics analyses**.

Machine-Learning can enhance them to meet the increasingly high demands experiments put on simulations.

I'll talk about **Hadronization** and detail current efforts of the MLHad collaboration towards a **data-driven hadronization model.**

In particular, I'll tell you about HOMER, a recently released model that builds explicitly upon **Pythia as a baseline for inductive bias** and its extension to **arbitrary string configurations**.

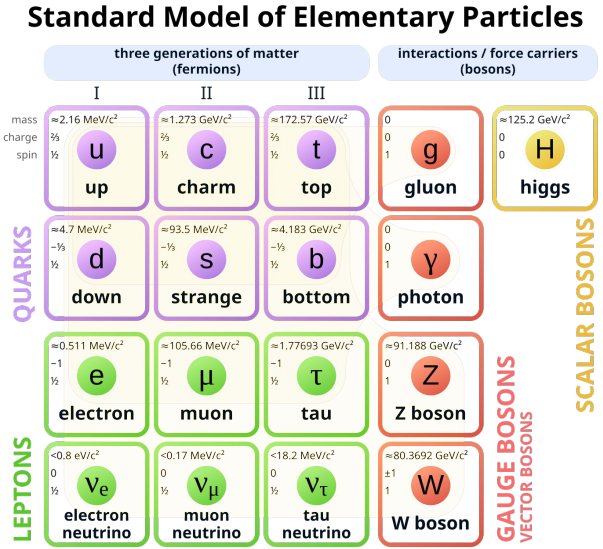


The Standard Model of Particle Physics

The Standard Model of particle physics represents our understanding of (some of) the most fundamental aspects of Nature.

However, **experimental observations** (dark matter, neutrino masses and gravity,...) and **theoretical issues** (θ_{QCD}problem, the Higgs hierarchy problem,...) make clear the **limitations of the SM**.

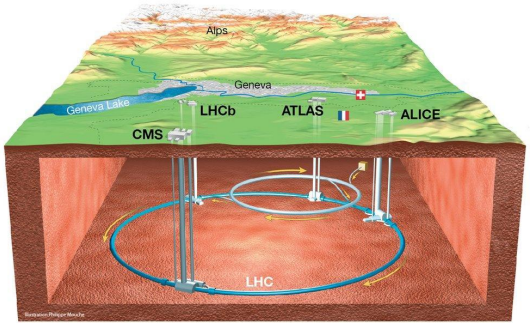
How do we **explore the SM and beyond**?



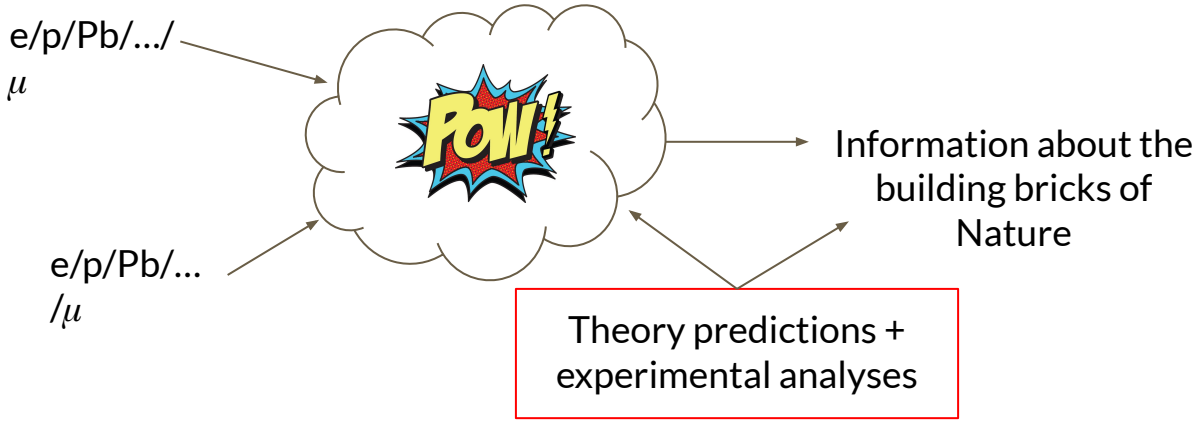
https://en.wikipedia.org/wiki/File:Standard_Model_of_Elementary_Particles.svg

Exploring by colliding

Colliders are among the most powerful tools we have to test and expand our knowledge



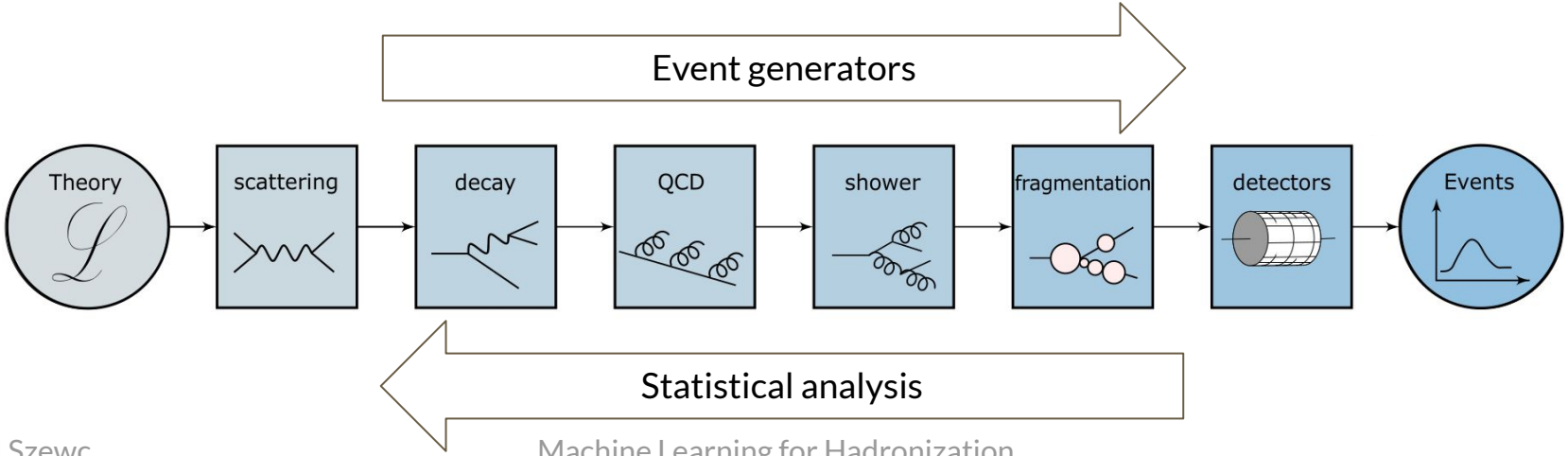
https://www.bbc.com/news/science-environment-36094282



Monte Carlo-based strategies

We perform **dedicated analyses** with specific topologies in mind: e.g. **Vh→1l2b+MET.** The simulation pipeline is fundamental to produce **expected events** and **infer from measured events**.

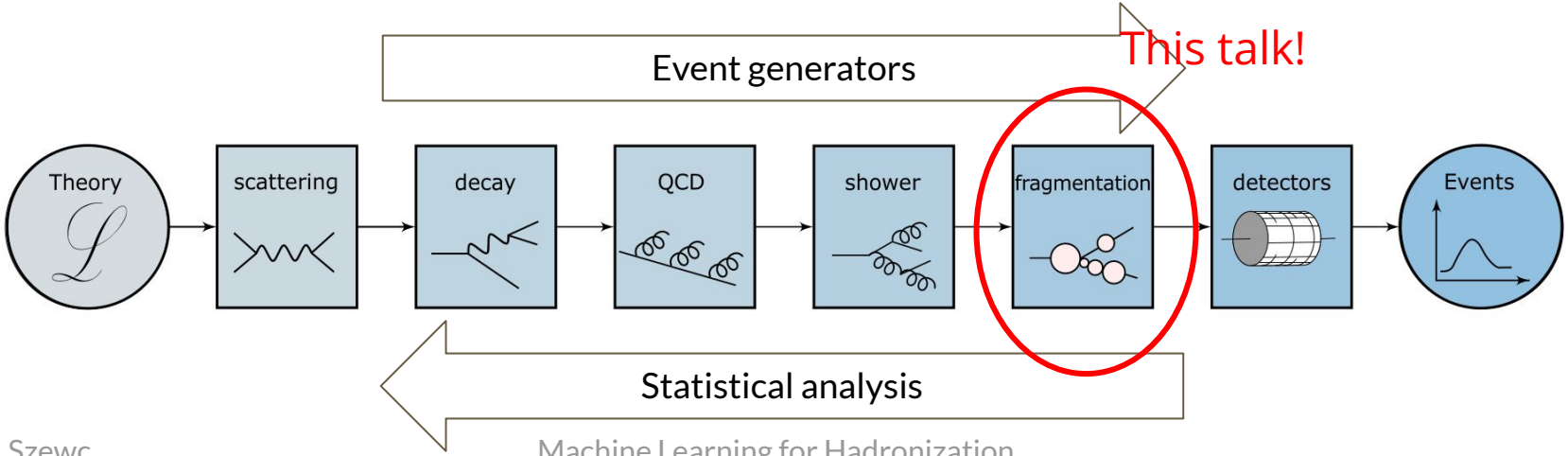
Image from https://arxiv.org/pdf/2203.07460



ML for MC

ML is now very common in HEP. In particular, it has been used to **enhance MC generators** in a multitude of ways. Today I'll talk about (some) applications to **fragmentation**.

Image from https://arxiv.org/pdf/2203.07460



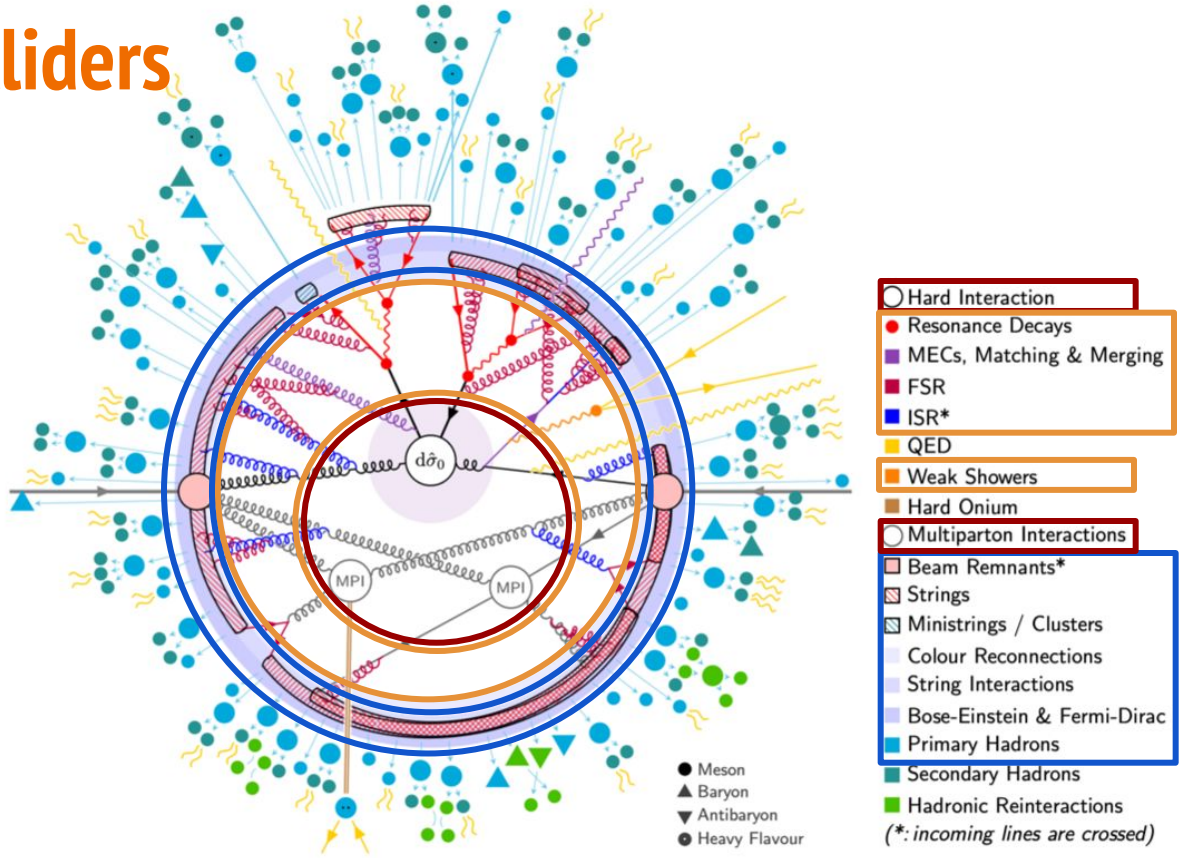
Hadronization at colliders

Image from Pythia 8.3 manual. The radial coordinate is time or 1/energy scale.

Hard process dσ: **perturbatively calculated**, directly related to underlying lagrangian, describes **partons hidden to experiment**.

Shower: **perturbative evolution** of partons from hard to hadronization scale, **hidden to experiment.**

Hadronization: combining partons into **measurable hadrons**, non perturbative → **Empirical models**.

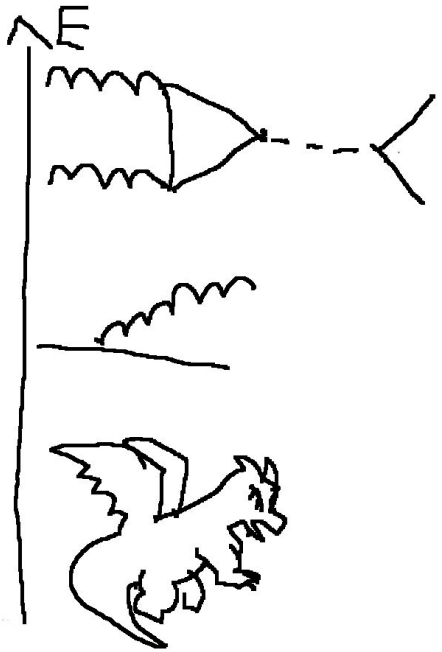


Hadronization empirical models

Inherently non-perturbative process → **Empirical models for predictions**. Two main **parametric models**: the Lund String model (Pythia) and the Cluster model (Herwig).

Very successful, but **at their limit**: leading uncertainties for some precision physics (e.g. m_t and α_s extractions)

Corners of phase space **cannot be reproduced**. **Collective effects** are tricky e.g. heavy baryon production at high event multiplicities as in arxiv:1807.11321.

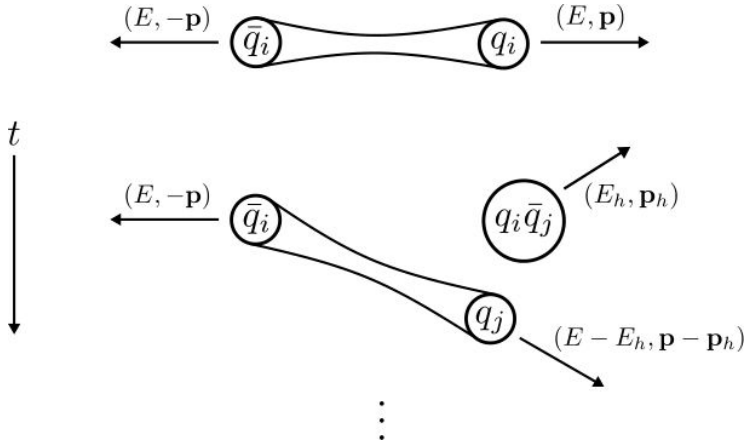


Hadronization empirical models

What makes our empirical models so successful? **Physics and time**. There is a built-in expertise that has been battle-tested against experiments for 40+ years.

Lund String model: explain hadronization kinematics in terms of **the light cone momentum fraction z** and the **transverse momentum of the string break p_T**.

Actual implementation is more complicated: flavor, junctions, acceptance filters... **Lots of modelling tested against experiment**



Lund String Model: Colored singlets + O(20) parameters → Hadrons

Simplified example from arxiv:2203.04983 .

Machine Learning to the rescue?

Complex problem with **no full model flexible enough** and where **training is expensive**? → Machine Learning should be really useful here!

A lot of possible ways to attack this problem. The richness of the involved physics **forbids the use of any plug-and-play algorithms**.

Two groups have recently tackled the subject: **MLHAD** (arxiv:2203.04983, arxiv:2311.09296, arxiv:2410.06342, arxiv:2503.05667) and **HADML** (arxiv:2203.12660, arxiv:2305.17169, arxiv:2312.08453). Different **generators** (Pythia, Herwig) and different **architectures** (cSWAE,BNF,GNNs, GAN) with different **degrees of implementation**.

Normalizing Flows for density learning

A **generative model** that learns how **the data density relates to a simpler base distribution**. We learn the parameters of chosen invertible functions that transform samples from the base distribution to the data distribution.

Less flexible but easier to train + access to the exact likelihood

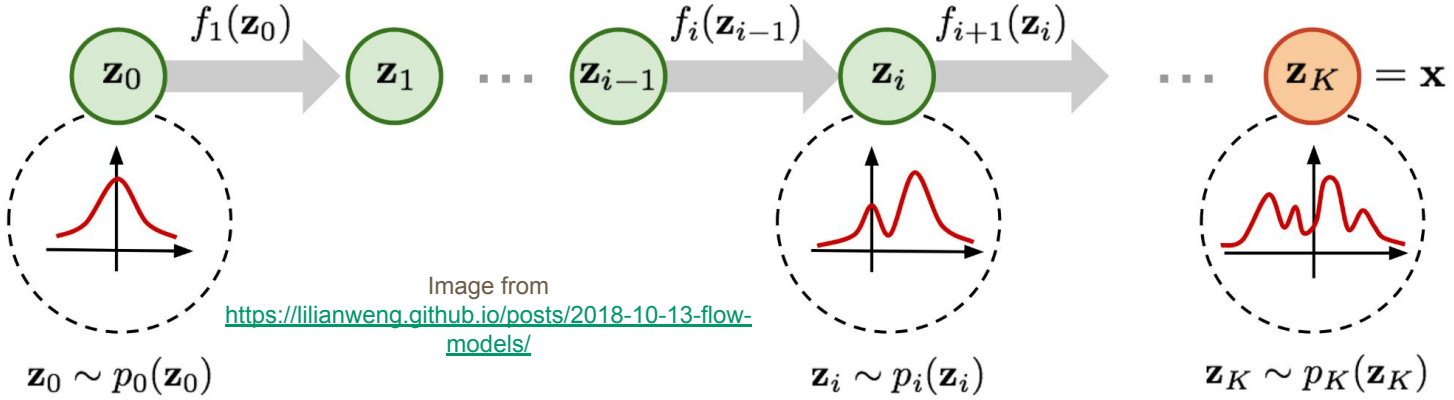


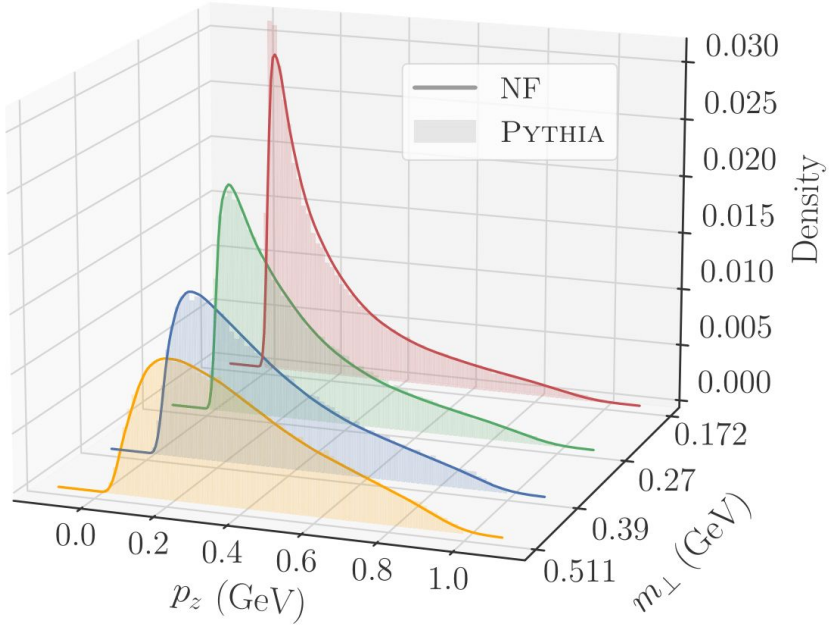
Image from https://lilianweng.github.io/posts/2018-10-13-flow-models/

Normalizing Flows for the Lund fragmentation function

First instinct: **learn the fragmentation pdf from data**.

Explored in arxiv:2311.09296, **first hadronization pseudo-data with only pions** and **x = (p_z,p_T).**

The **NF learns the appropriate Lund distribution** with its complicated relation between m_T and p_z.

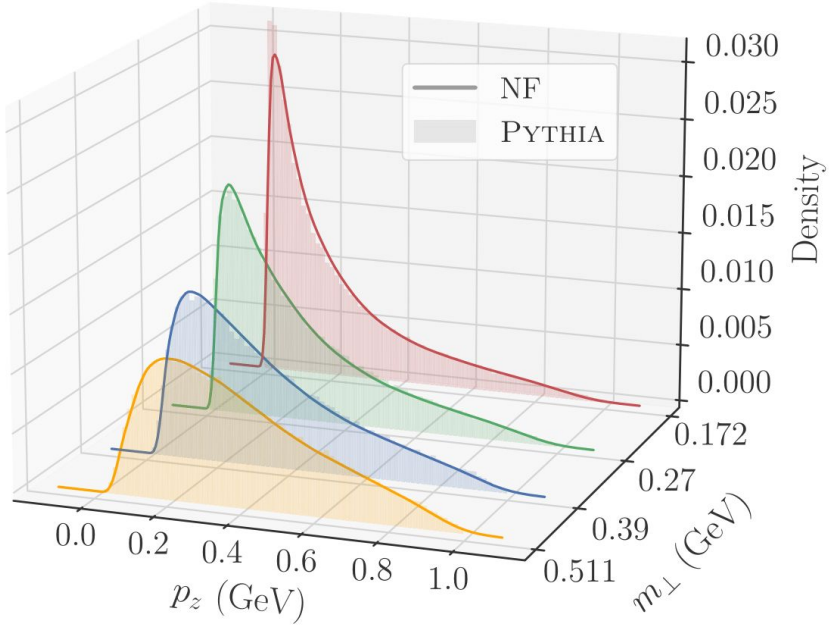


arxiv:2311.09296

Normalizing Flows for the Lund fragmentation function

Works for a simple example with access to individual fragmentations → **Very much not like real data**. We improve upon this with **MAGIC** (ask me about it!)

Data is much **more complicated** than our example→ Maybe our NF needs **more information**. We should **learn from Pythia's 40+ years of model building.**

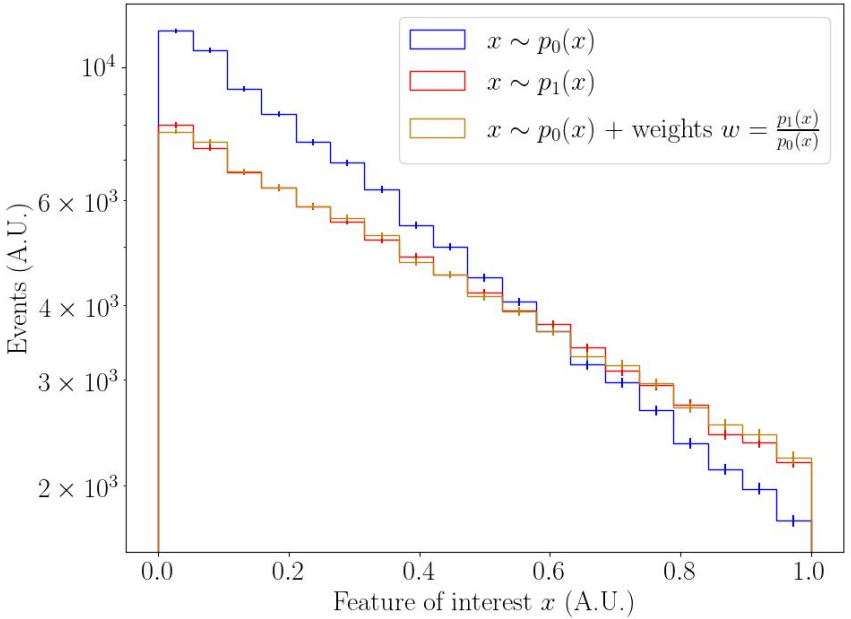


arxiv:2311.09296

Learning a data-driven ~~function~~ reweighting

Use Pythia as a **starting point** and **fine-tune** our model → **take advantage of subtleties of the Lund String fragmentation model** built into Pythia by a dedicated effort of 40+ years!

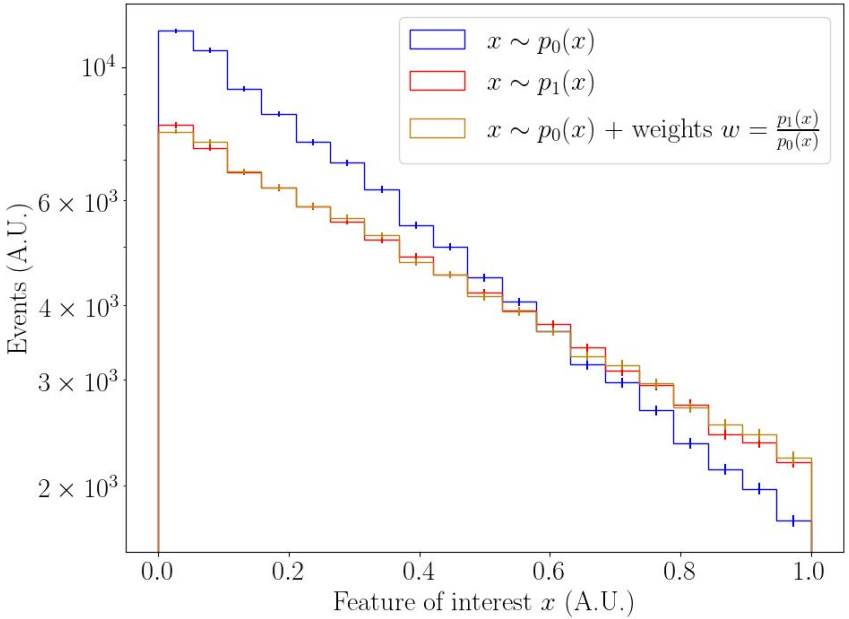
Frame in terms of **reweighting** starting from the Lund string model**.** Weights should be such that simulations and data indistinguishable at the **measurable level**.



Learning a data-driven ~~function~~ reweighting

We learn **how the measurable quantities should look like**, and obtain a fragmentation function reweighter that **morphs simulated events** accordingly.

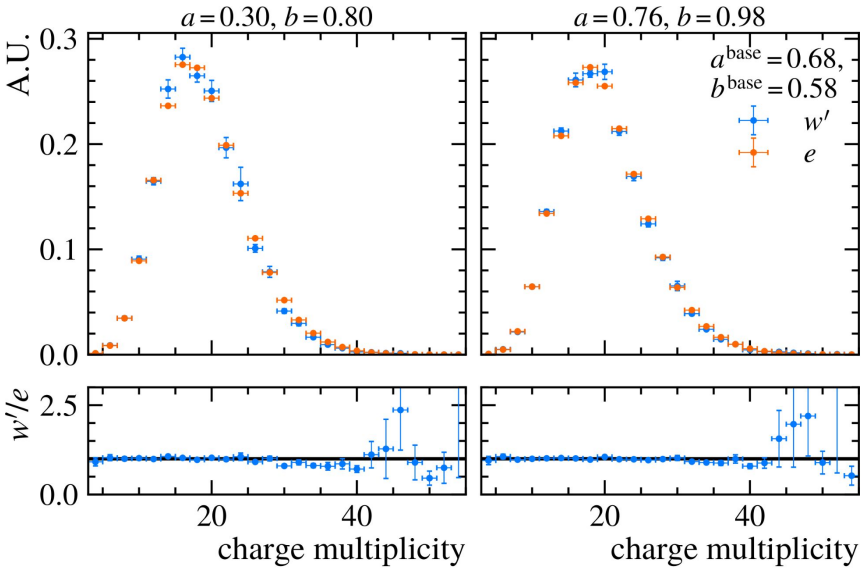
Data-driven weights, data-driven fragmentation function.



Slight detour: Pythia Kinematic Reweighter

In arxiv:2308.13459, we introduced a way to account for **parametric variations on the fragmentation model in Pythia**.

For pre-specified baseline hadronization parameters and a set of alternative choices, we produce **a set of events with associated weights**. These weights can be used to **reweight** the events from "**baseline**" to any of the desired "**alternatives**".



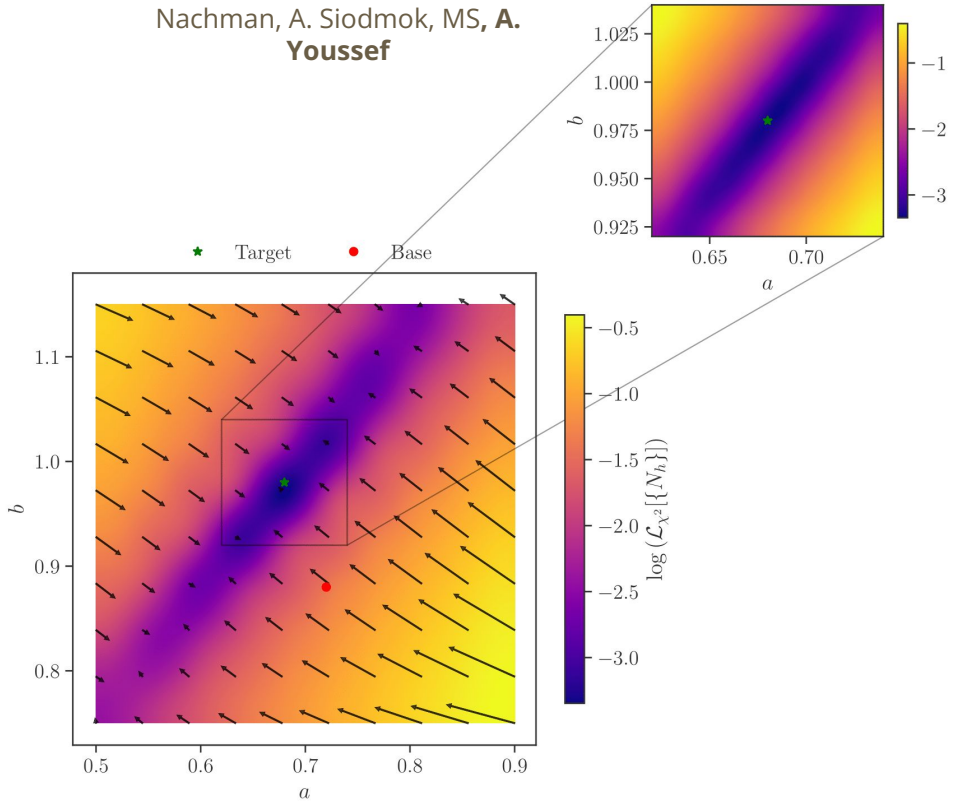
More efficient tunes: RSA

Rejection **S**ampling with **A**utodifferentiation. Applied to the Lund fragmentation function as **learnable Pytorch module** but can be extended to other problems with similar forward models.

Useful for **reweighting, parameter estimation, model exploration and unbinned fitting**.

Example: efficiently exploring **the loss landscape in parameter space.**

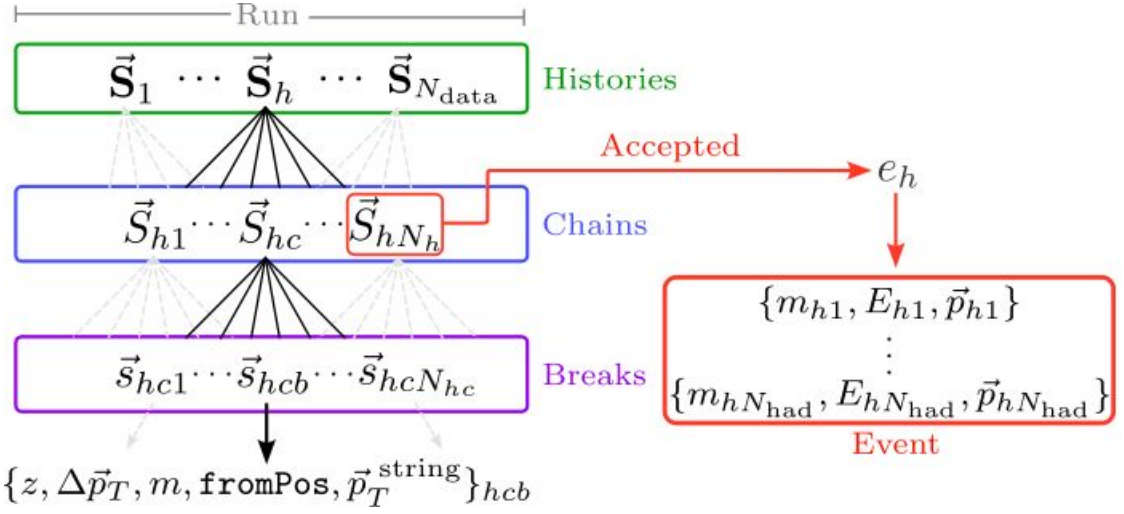
arxiv:2411.02194 by **N. Heller**, P. Ilten, **T. Menzo**, S. Mrenna, B. Nachman, A. Siodmok, MS**, A. Youssef**



Learning from data: HOMER

Histories and **O**bservables for **M**onte-Carlo **E**vent **R**eweighting: We take **Pythia as baseline** and reweight that.

We restrict ourselves to the **qq string emitting only pions**. For simulation, we record everything. **For pseudo-data, only observable quantities**.

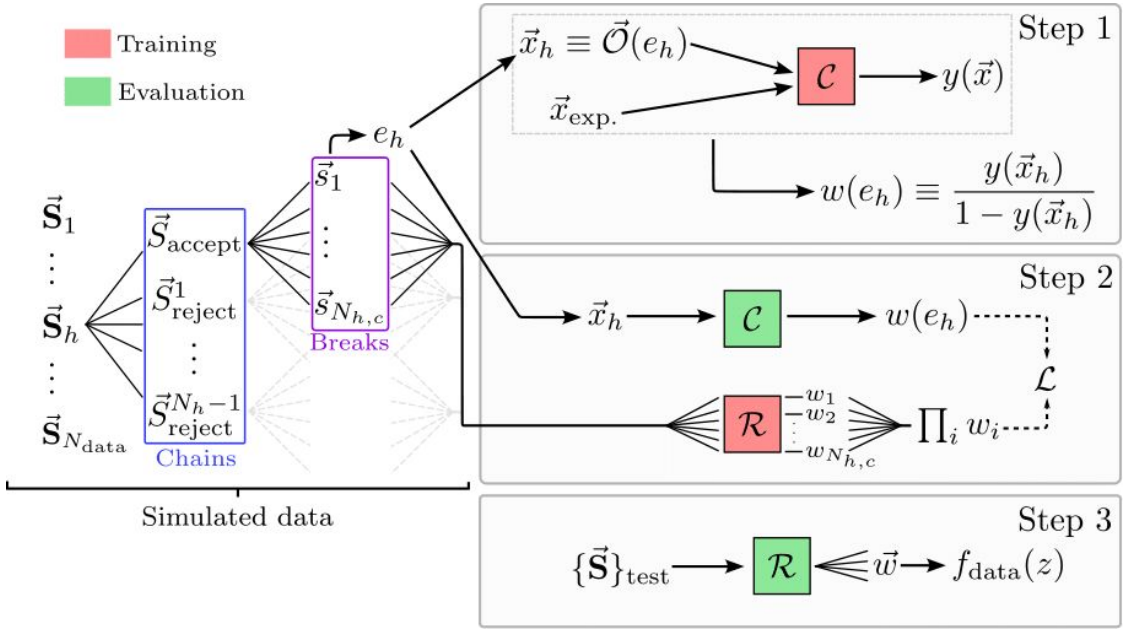


Learning from data: HOMER

Histories and **O**bservables for **M**onte-Carlo **E**vent **R**eweighting: We take **Pythia as baseline** and reweight that.

Two step procedure: We generate **once** and learn the appropriate **reweighting function**. New model is **Pythia + weight.**

(No rejection sampling needed here, but we do need to account for filtering!)



Comparing different possible measurements

Our simulation and data consist of **full events**. We use different Pythia parameters to generate simulation and pseudo-data.

We consider three possibilities for available measurements:

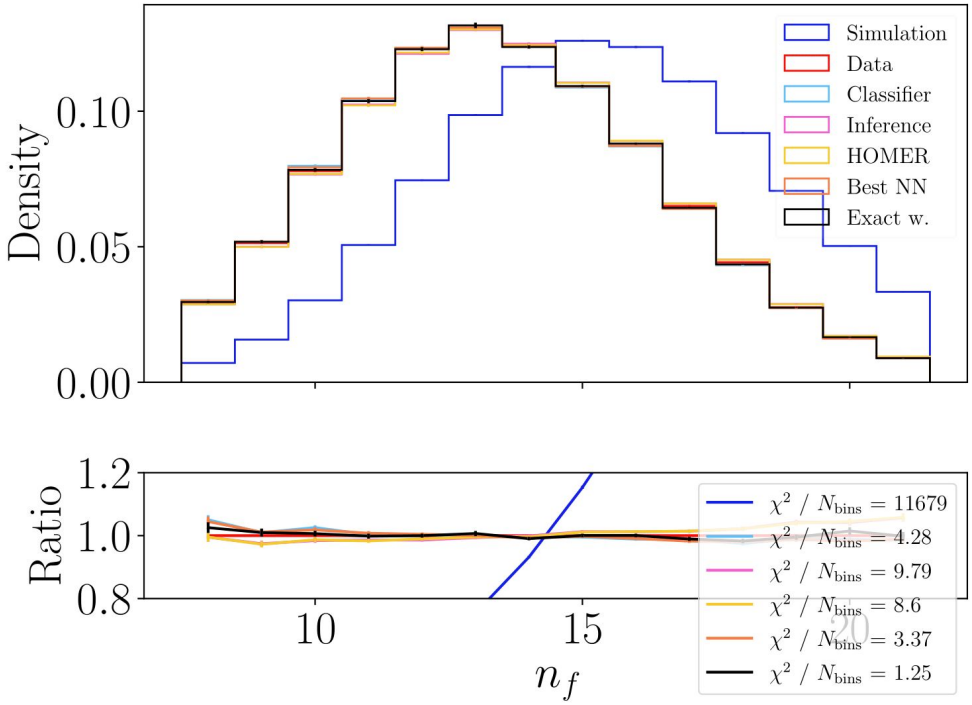
- **High-level observables** (thrust, multiplicity,...) available only through **one-dimensional histograms** (corresponds to available data).
- **High-level observables** available on an **event-by-event** basis.
- The complete set of **particles with associated four-momenta** (the **point cloud** representation).

For all three cases, we perform HOMER. Performance is evaluated both at the measurement level and at the fragmentation level

Matching sim to data (unbinned high-level obs.)

Our reweighted simulation **matches** the data at the **observable level**.

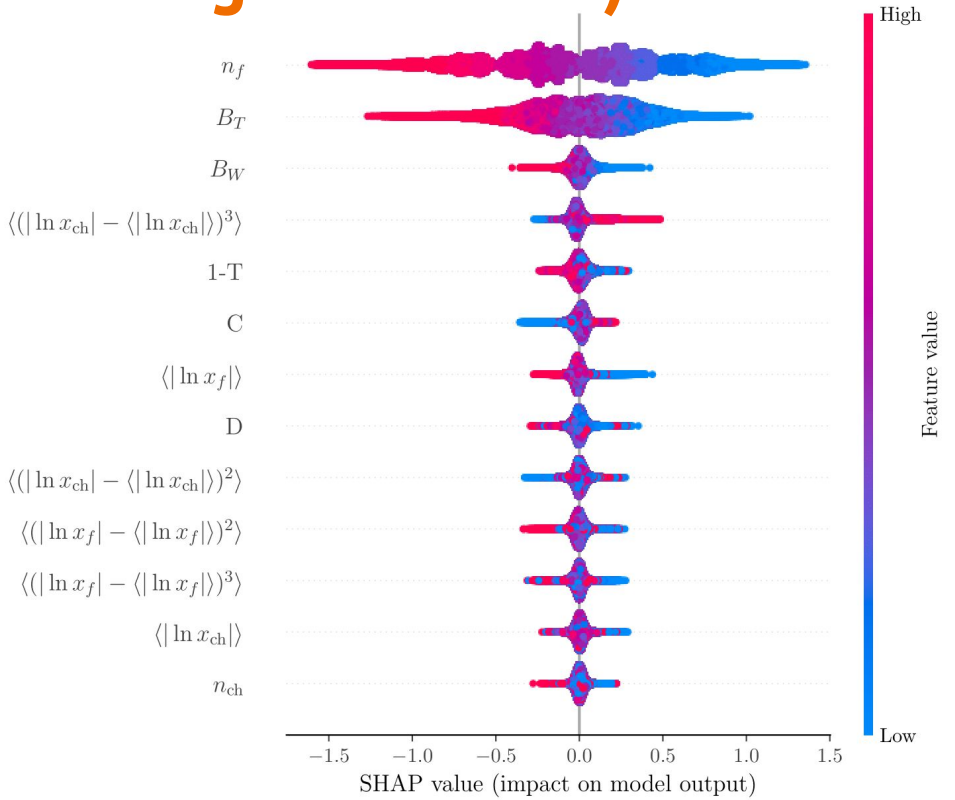
The agreement is better for step 1 than for step 2 / full weights due to the combination of **additional bias + imperfect training**.



Matching sim to data (unbinned high-level obs.)

For our first step using high-level variables, we use BDTs → **Powerful, easy to tune** and also gives us **feature importances** through SHAP values.

Multiplicity is by far the most important feature when matching simulation to data in our example.

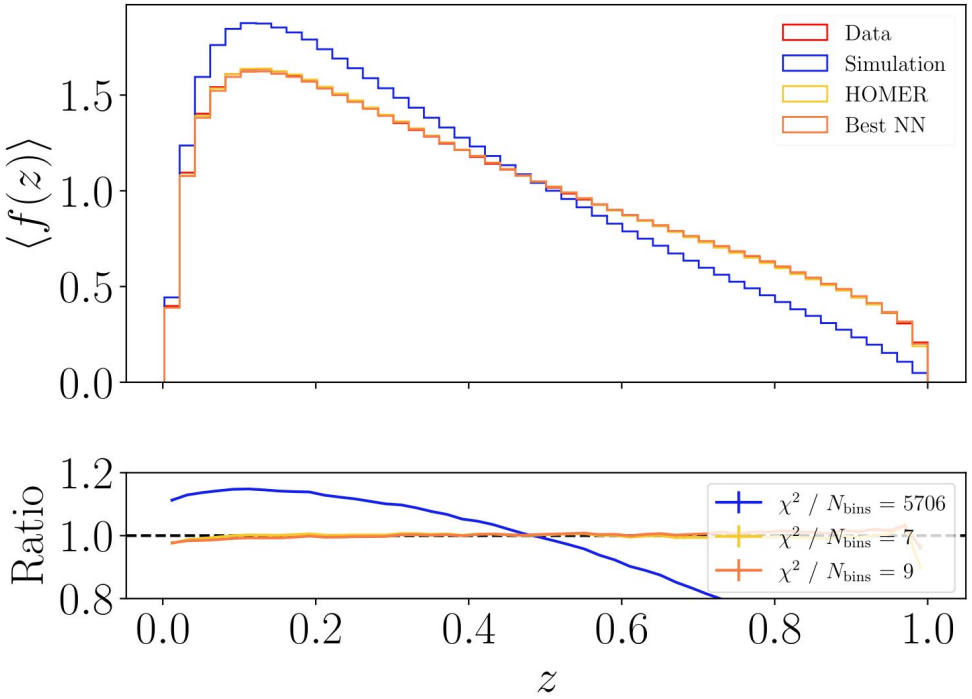


New underlying model (unbinned high-level obs.)

The weights are translated to a **data-driven fragmentation function** via a Graph Neural Network-based architecture.

We see how the model almost **saturates** what we can learn using the chosen NN architecture and is a **great fit** to **data**.

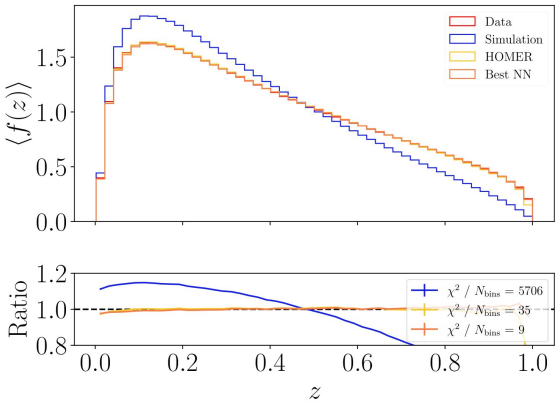
This is also observed for different m_T bins.



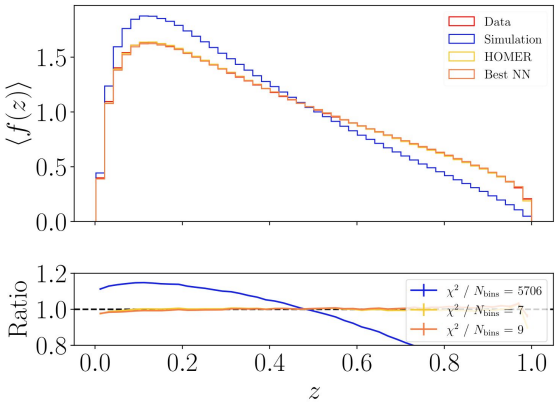
Comparing all three measurement choices

The three fragmentation functions are:

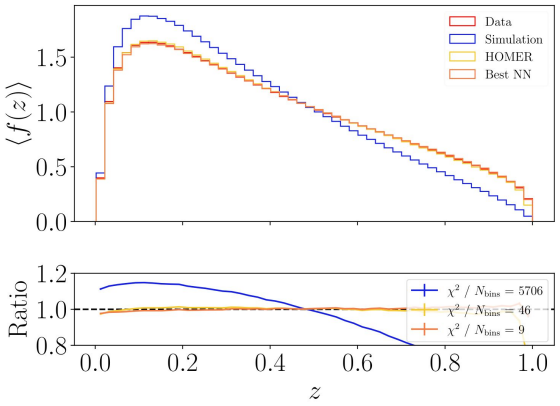
Binned high-level



Unbinned high-level



Point cloud



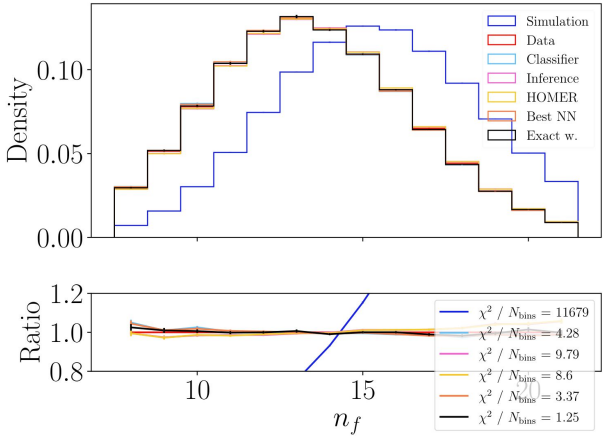
All three percent-level agreement! (and unbinned saturates the Best NN architectural bound)

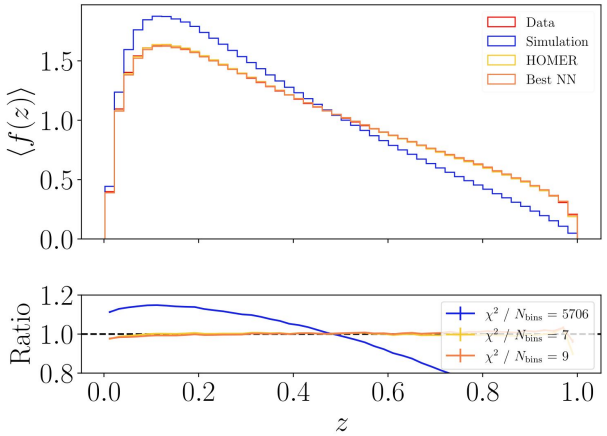
Take home message

We recover the **underlying fragmentation function** given only **observable quantities**.

The Lund String model is still used as a **baseline**, properly accounting for the event structure is **essential**.

So far **flavour has been ignored**, with only pions considered. Additionally, **gluons make things trickier**.





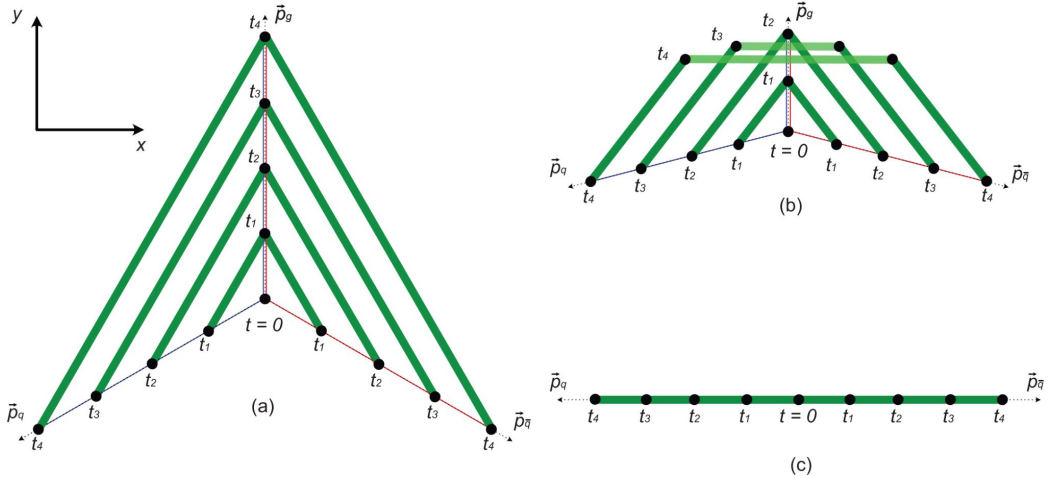
HOMERv2: Arbitrary multiparton strings

More realistic. More complex evolution but **unchanged sampled variables (z,p_T,flavor)**.

Two fundamental challenges:

- **Degeneracies:** observable events less constraining of the hadronization chain
- **Multiple initial states:** accounting for finalTwo more challenging

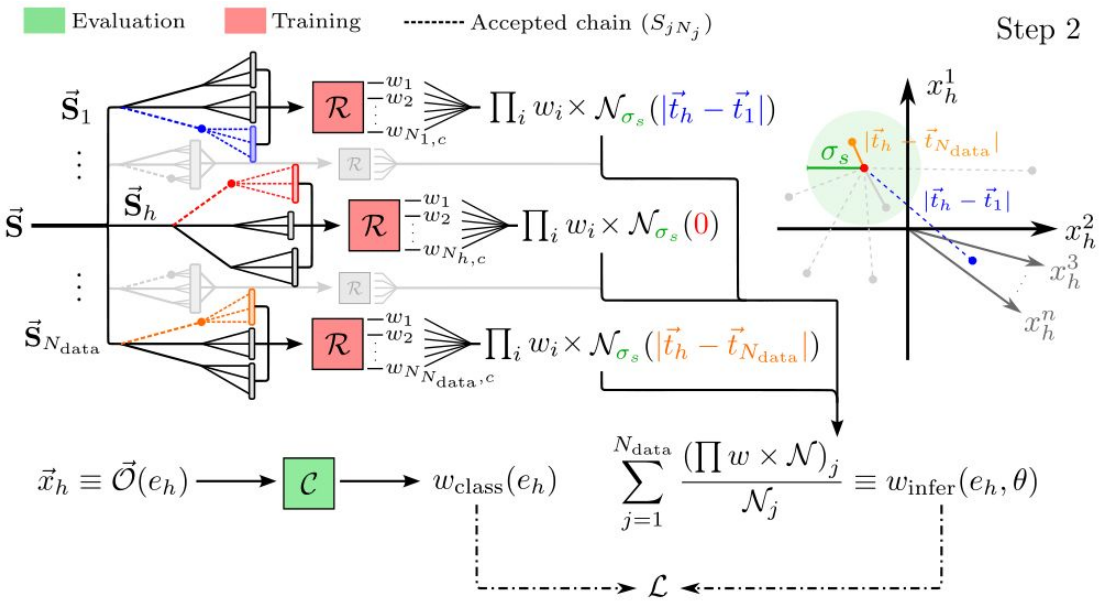
Additional challenge: parton shower dominates the kinematics → **Harder to train first step classifier**.



HOMERv2: Arbitrary multiparton strings

$$w_{\text{infer}}(e_h, \theta) = \frac{\sum_j w_{\text{HOMER}}(\vec{\mathbf{S}}_j)\mathcal{N}_{\sigma_s}(|\vec{t}_{e_h} - \vec{t}_{e(\vec{S}_{jN_j})}|)}{\sum_j \mathcal{N}_{\sigma_s}(|\vec{t}_{e_h} - \vec{t}_{e(\vec{S}_{jN_j})}|)}$$

Similarity kernel with a **tunable parameter σ_s** to **average over compatible histories** (ie **marginalization**)

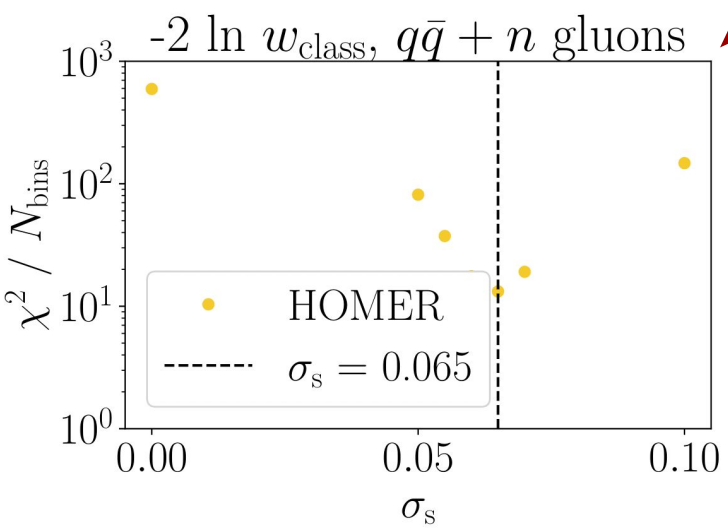


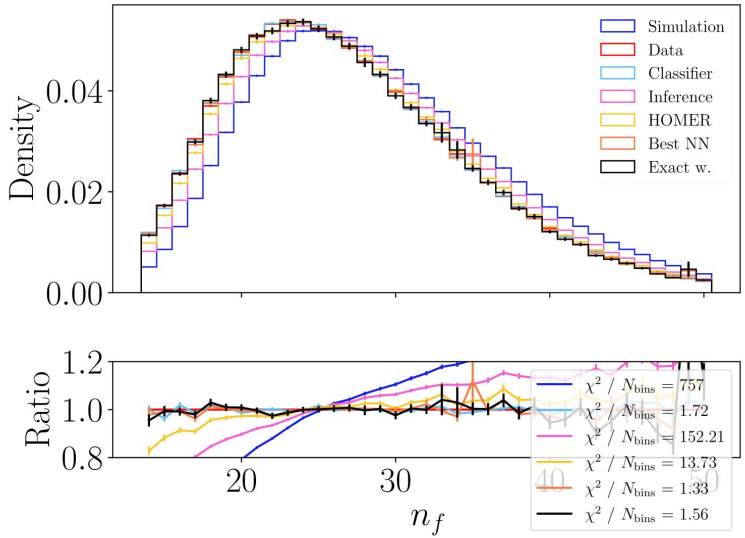
Automatically **takes care of multiple initial states**!

HOMERv2: Arbitrary multiparton strings

$$w_{\text{infer}}(e_h, \theta) = \frac{\sum_j w_{\text{HOMER}}(\vec{\mathbf{S}}_j)\mathcal{N}_{\sigma_s}(|\vec{t}_{e_h} - \vec{t}_{e(\vec{S}_{jN_j})}|)}{\sum_j \mathcal{N}_{\sigma_s}(|\vec{t}_{e_h} - \vec{t}_{e(\vec{S}_{jN_j})}|)}$$

Data-driven!

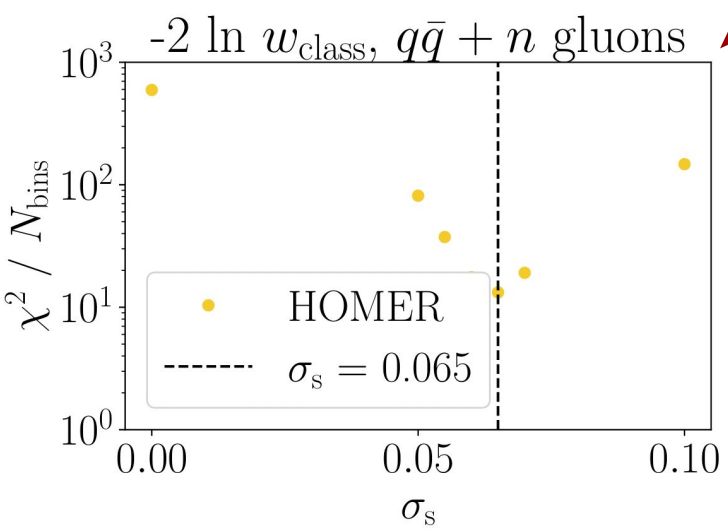


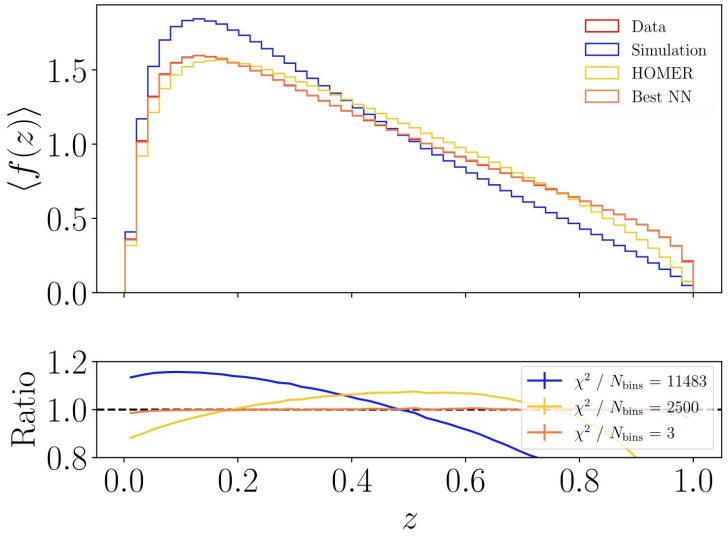


HOMERv2: Arbitrary multiparton strings

$$w_{\text{infer}}(e_h, \theta) = \frac{\sum_j w_{\text{HOMER}}(\vec{\mathbf{S}}_j)\mathcal{N}_{\sigma_s}(|\vec{t}_{e_h} - \vec{t}_{e(\vec{S}_{jN_j})}|)}{\sum_j \mathcal{N}_{\sigma_s}(|\vec{t}_{e_h} - \vec{t}_{e(\vec{S}_{jN_j})}|)}$$

Data-driven!

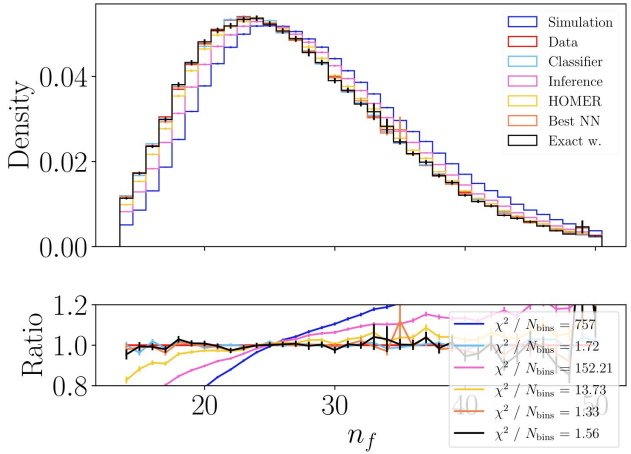


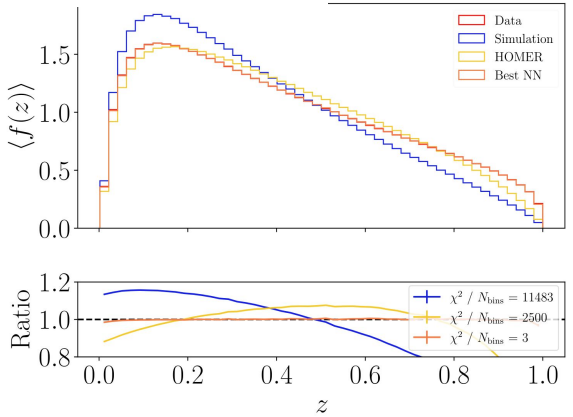


Take home message

By introducing **a notion of distance,** we can effectively **average out compatible histories**. The additional hyperparameter is **tuned in data**.

This allows HOMER to deal with (in principle) **arbitrary string topologies** and **degeneracies in the measurements**.





Conclusions

We have developed a method for learning a **data-driven fragmentation function** from **available measurements.** The model is tied to the string fragmentation picture to take advantage of the Pythia event generator.

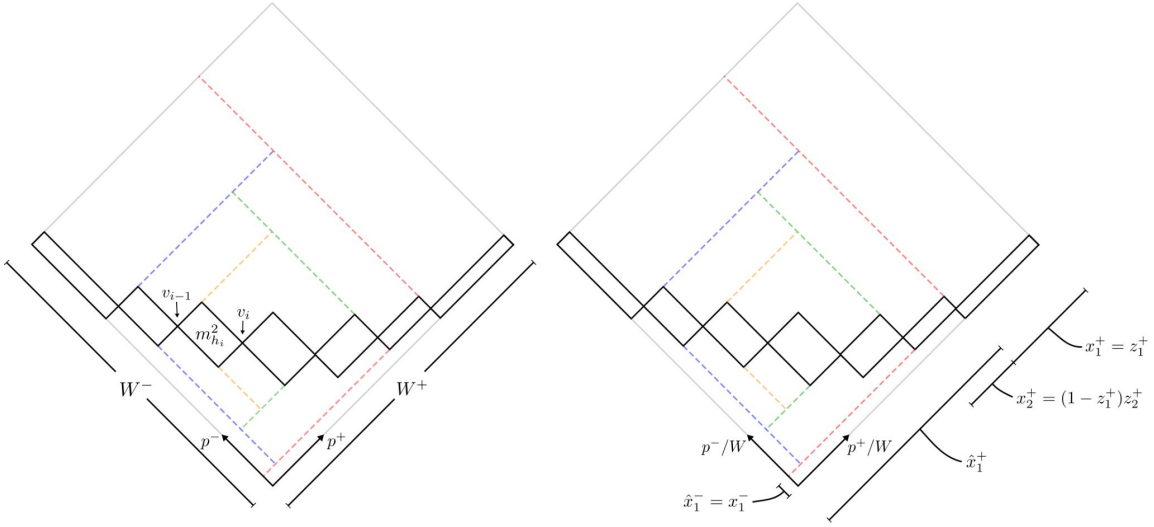
The method has been tested for the single string case with arbitrary gluons and producing only pions, with **general flavor to be incorporated**.

Any developments impact e**xisting tuning efforts and uncertainty estimations** → RSA is an example of **differentiable simulators through reweighting**

This is very much an open problem with many questions still unposed!

Backup slides

Momentum space for finding next hadronization



Limitations of the Lund String model

O(20%) to O(50%) discrepancies between proton-proton and ion-ion collisions

Heavy particle composition as a function of event multiplicity is mismodelled at high event multiplicities

Mismodelling of the mass dependence of the average transverse momentum

Minimum bias description can be incompatible because of low transverse momentum mismodelling

Ridge in pp collisions missing in Pythia (and in general long range correlations are hard to model)

Charged particle multiplicity spectrum is very sensitive to color reconnections and MPI modelling

Learning with uncertainties

Any hadronization model should be able to deal in different uncertainties:

- **Training** uncertainties due to **finite sample size of the training dataset** and the **bias of the model**
- **Data** uncertainties inherent to the training data, ie **systematic effects**

Even for the well established Lund String model, obtaining the uncertainties on the parameters and translating them to observables is **complicated**

Can ML be a more natural way of working with uncertainties?

Uncertainty estimation in NNs

Usual ML models work as deterministic functions (even if the task is probabilistic such as in NFs!). A lot of effort has gone into assigning uncertainties to NNs. The two main methods are **Ensembles** and **Bayesian Neural Networks**.

BNNs learn a **distribution over parameters**, usually through Variational Inference.

$$y = f\left(x; \theta(X, Y)\right) \rightarrow y = \int d\theta \, f(x;\theta) p(\theta|X, Y)$$

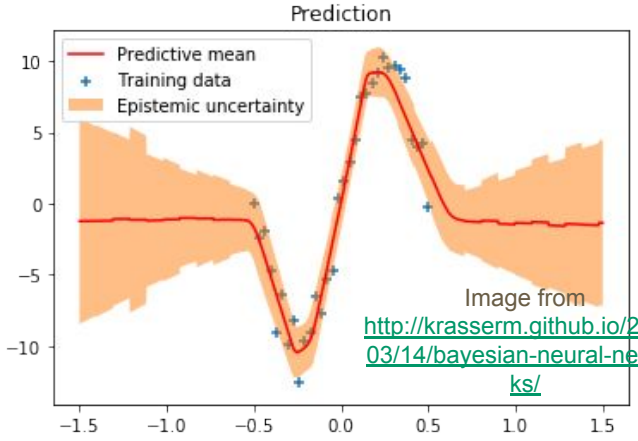


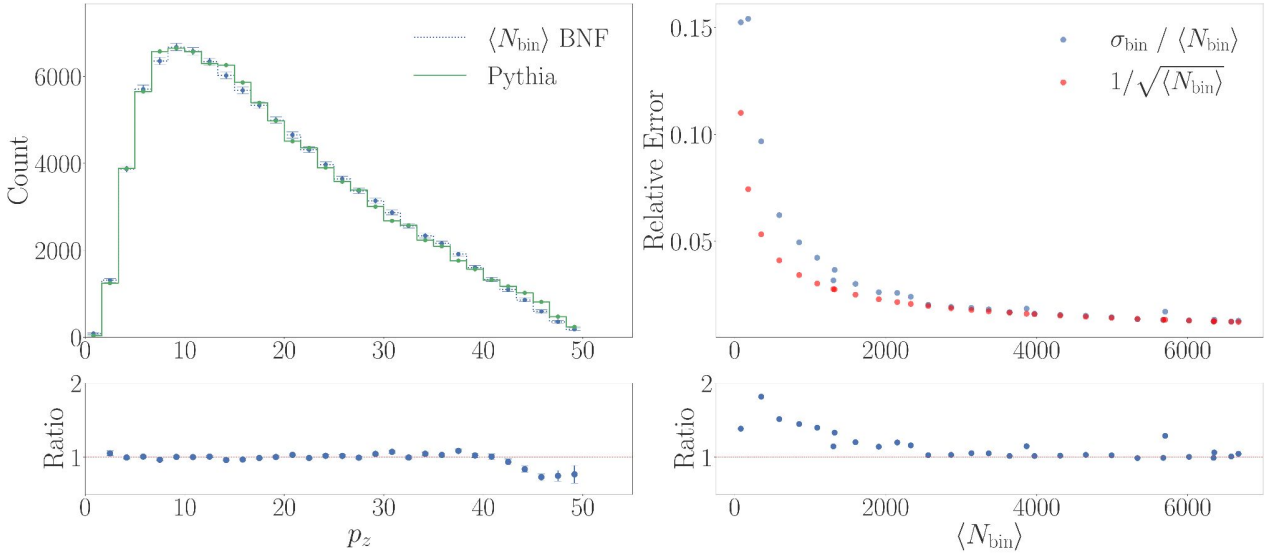
Image from http://krasserm.github.io/2019/03/14/bayesian-neural-networks/

Bayesian Normalizing Flows

We deploy a BNF to capture the training error. We still train on first hadronization but now we capture **a family of distributions**

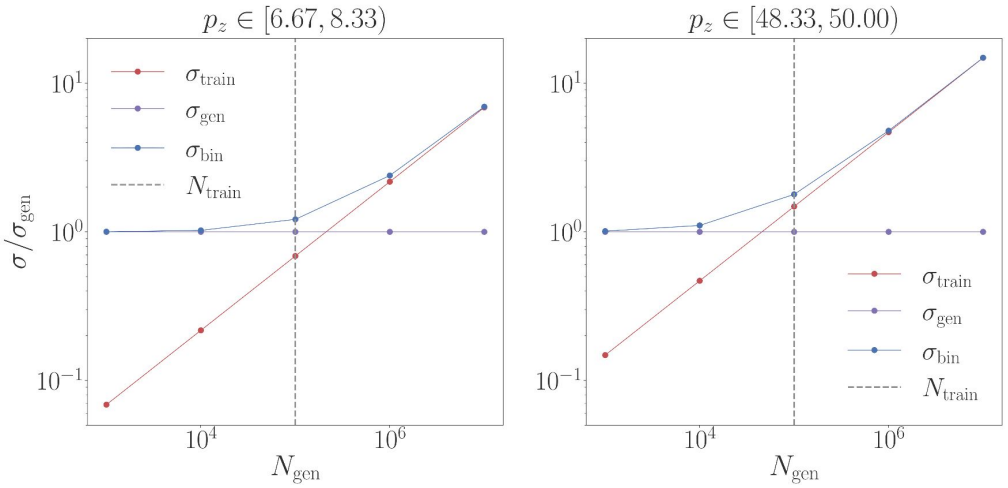
The BNF **captures the underlying distribution**.

The **total** uncertainty is always **higher than the aleatoric Poisson uncertainty** due to the additional **epistemic training error**.



Pythia: 10^5 first hadronizations used to train the model. BNF: 10^5 first hadronizations for 5×10^4 models sampled from the BNF.

Bayesian Normalizing Flows



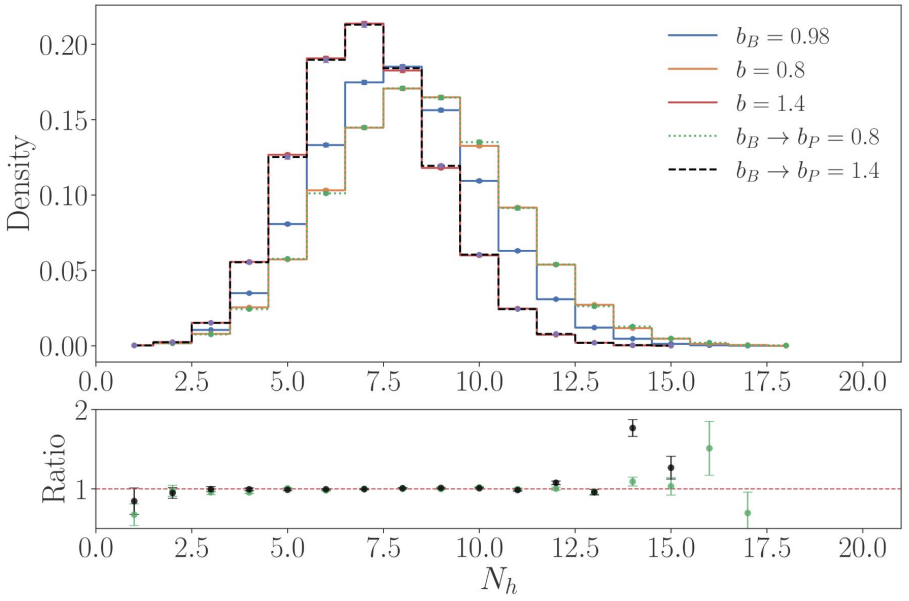
The **training uncertainty becomes the limiting factor** after a certain generated size (which depends on the quality of training).

Systematic Uncertainties

We capture systematic uncertainties by **conditioning during training**.

We **propagate** these uncertainties through **reweighting** using the **exact learned likelihood**.

b is an hadronization parameter that has the same effect as changing physical nuisance parameters.



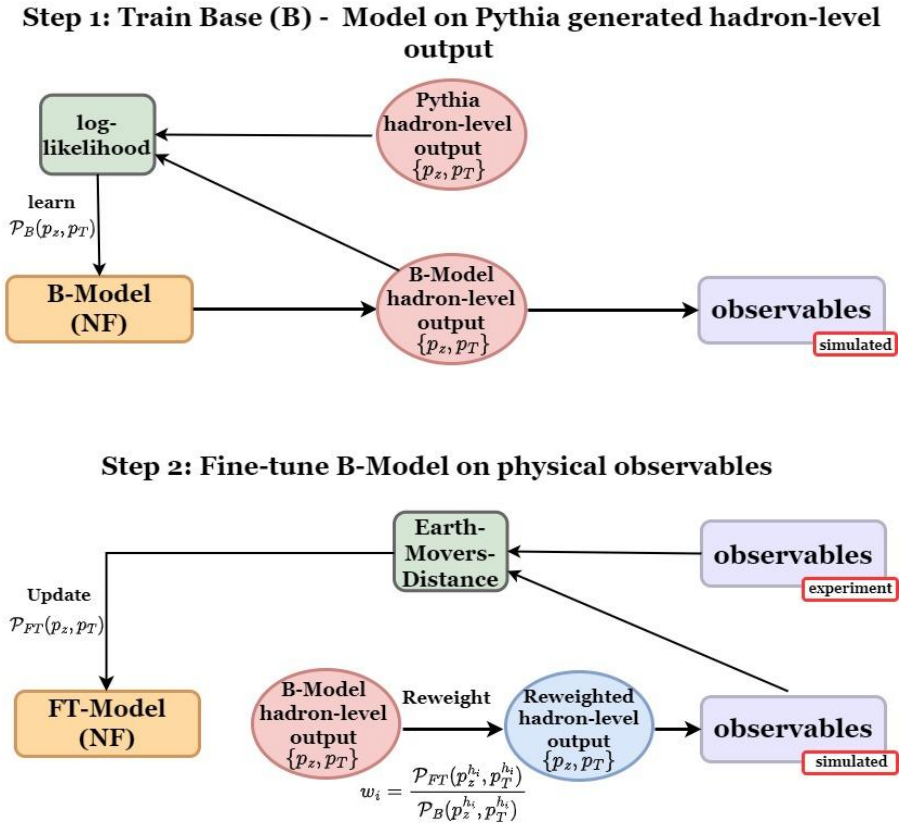
5x10^4 strings

Learning from data: MAGIC

We propose a new strategy, called **M**icroscopic **A**lterations **G**enerated from **I**nfrared **C**ollections. It is a two-step procedure.

We generate **once** and learn the appropriate **reweighting function**.

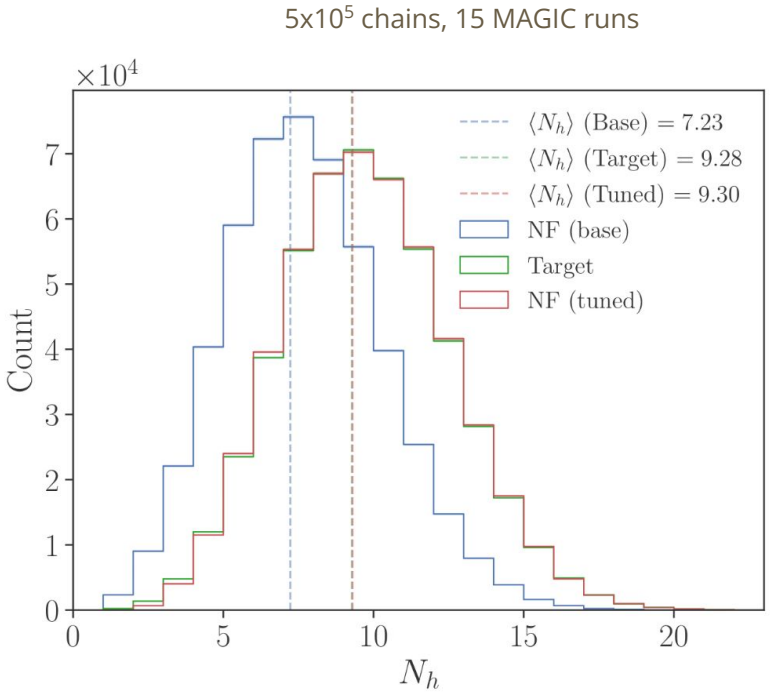
NF allows us to access exact likelihood!



MAGIC

We compare the original learned model (**base**) with the **target** (here pseudo-data from Pythia with different parameters) and obtain a new model (**tuned**)

We learn $p(p_z)$ from $p(N_h)$ (1 observable for a 1-dimensional fragmentation function).

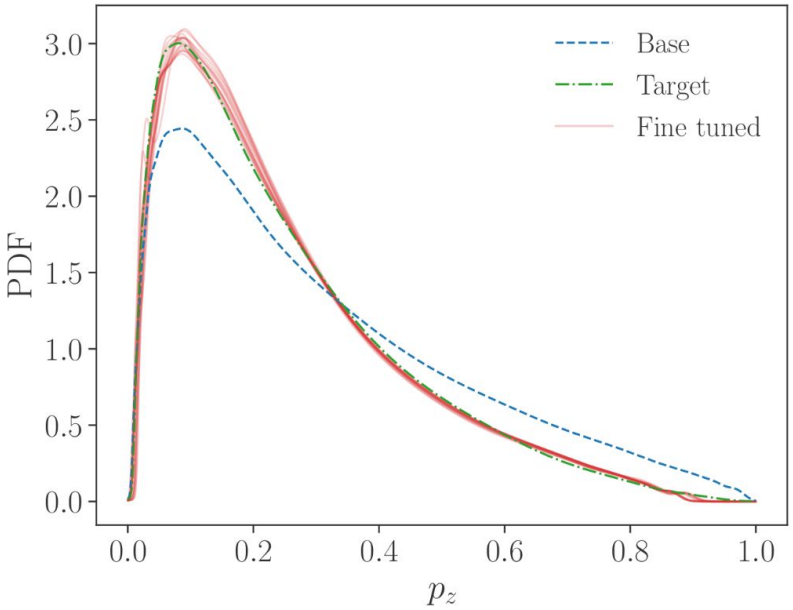


MAGIC

Because we are reweighting in terms of the underlying hadronizations, we obtain a **new hadronization model**!

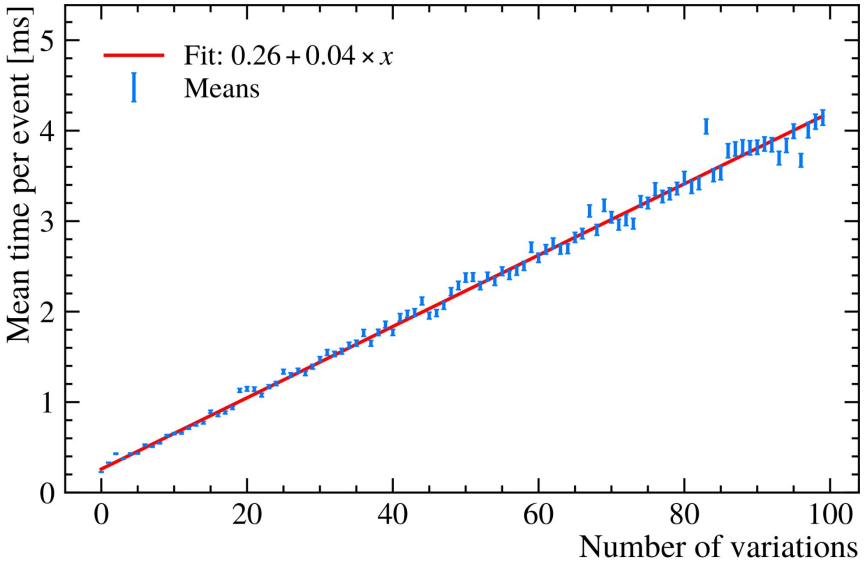
Here we worked only with p_z. For (p_z,p_T), we need more observables to break degeneracies.

15 MAGIC runs



Pythia Kinematic Reweighter

For reasonable choices where the coverage between baseline and alternative is good, this represents a very powerful method for reducing simulation costs and perhaps a better way to compute uncertainties on a given Monte Carlo prediction.



Pythia Kinematic Reweighter

Coverage diagnostics can be the mean of event weights and the effective degrees of freedom.

As we move away from the baseline, performance worsens.

variation	$1-\mu$	n_{eff}/N	figure
$r_b^{\text{base}} = 0.855$	0	1	figs. 3 and 6
$r_b = 0.657$	$(0.1 \pm 7.9) \times 10^{-4}$	6.2×10^{-1}	
$r_b = 0.459$	$(1.2 \pm 2.4) \times 10^{-3}$	1.9×10^{-1}	
$r_b = 1.792$	$(1.1 \pm 0.4) \times 10^{-1}$	7.3×10^{-4}	
$a^{\text{base}} = 0.68$	0	1	fig. 1
$a = 0.30$	$-(0.5 \pm 4.0) \times 10^{-3}$	5.8×10^{-2}	
$a = 0.55$	$-(2.4 \pm 4.7) \times 10^{-4}$	8.2×10^{-1}	
$a = 0.76$	$-(1.7 \pm 2.6) \times 10^{-4}$	9.4×10^{-1}	
$b^{\text{base}} = 0.98$	0	1	fig. 2
$b = 0.58$	$(4.0 \pm 2.0) \times 10^{-2}$	2.3×10^{-3}	
$b = 0.80$	$(1.4 \pm 1.4) \times 10^{-3}$	3.4×10^{-1}	
$b = 1.07$	$-(3.4 \pm 3.7) \times 10^{-4}$	8.8×10^{-1}	
$\sigma_{pT}^{\text{base}} = 0.350$	0	1	fig. 4
$\sigma_{pT} = 0.283$	$(1.2 \pm 0.8) \times 10^{-2}$	1.4×10^{-2}	
$\sigma_{pT} = 0.360$	$-(4.9 \pm 3.1) \times 10^{-4}$	9.1×10^{-1}	
$a^{\text{base}} = 0.68$, $b^{\text{base}} = 0.58$	0	1	fig. 5
$a = 0.30$, $b = 0.80$	$(4.6 \pm 1.3) \times 10^{-2}$	5.7×10^{-3}	
$a = 0.76$, $b = 0.98$	$(1.2 \pm 0.7) \times 10^{-2}$	2.1×10^{-2}	

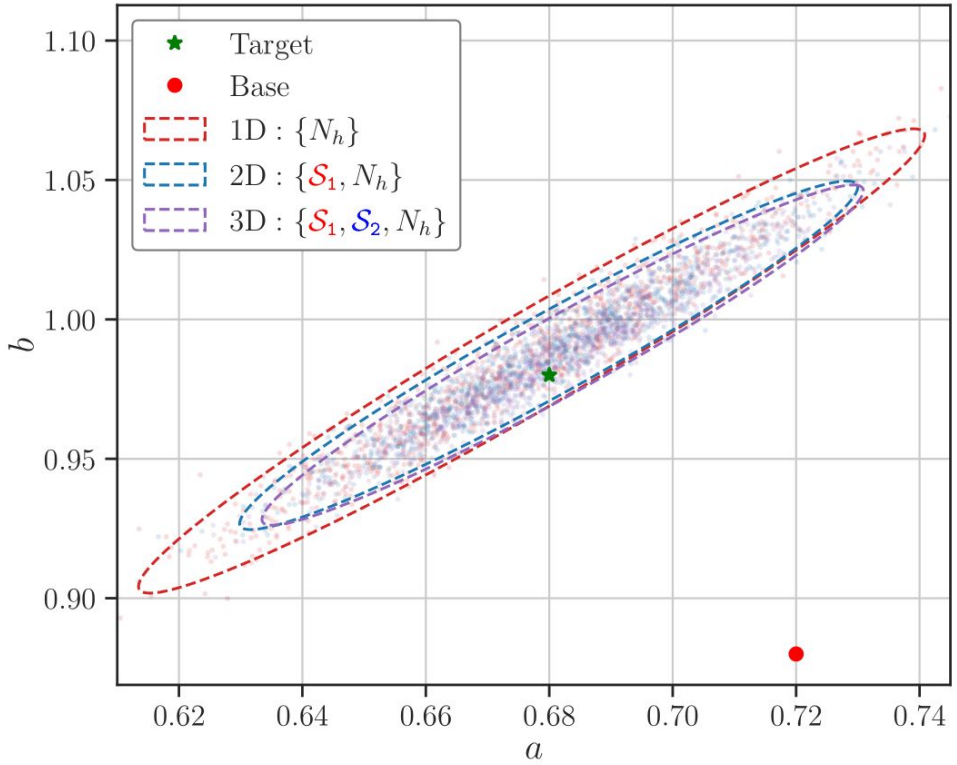
RSA

arxiv:2411.02194 by **N. Heller**, P. Ilten, **T. Menzo**, S. Mrenna, B. Nachman, A. Siodmok, MS**, A. Youssef**

Versatile and flexible approach to parameter estimation. SBI via reweighting → ML observables for tuning.

Example: **Two parameter fit** using different observables, either high-level or ML based. Here, contours estimated via bootstrapping but autodiff gives us access to **gradients for uncertainty quantification.**

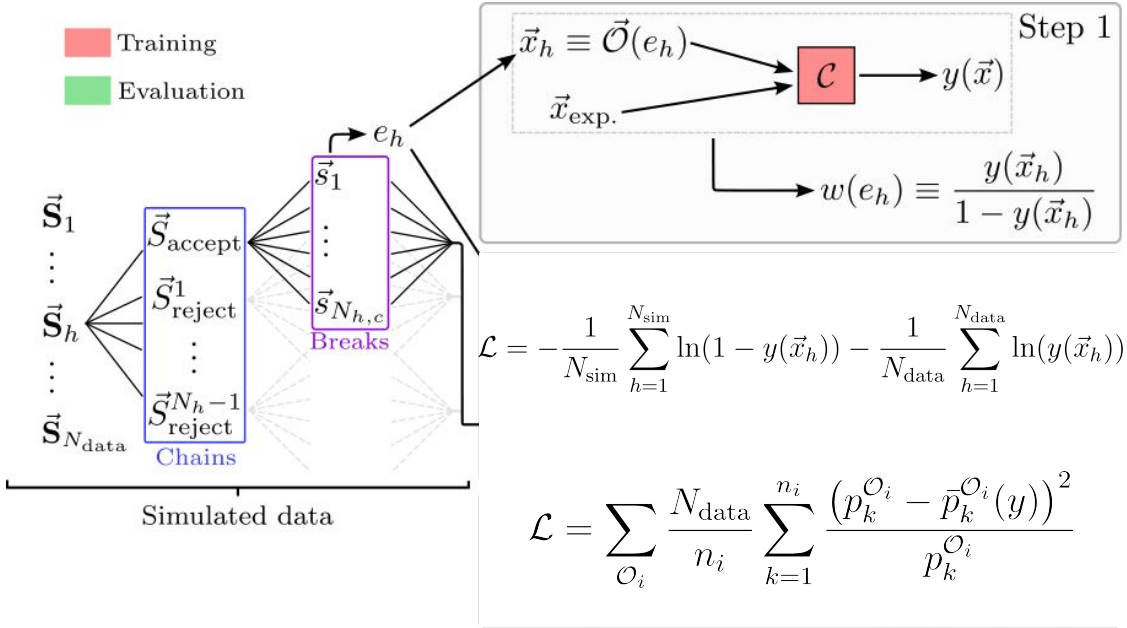
Memory footprint and **multiple base parameterizations** should be addressed through dedicated efforts, especially when scaling to more parameters, which RSA is **built to do** .



Learning from data: HOMER

Histories and **O**bservables for **M**onte-Carlo **E**vent **R**eweighting: We take **Pythia as baseline** and reweight that.

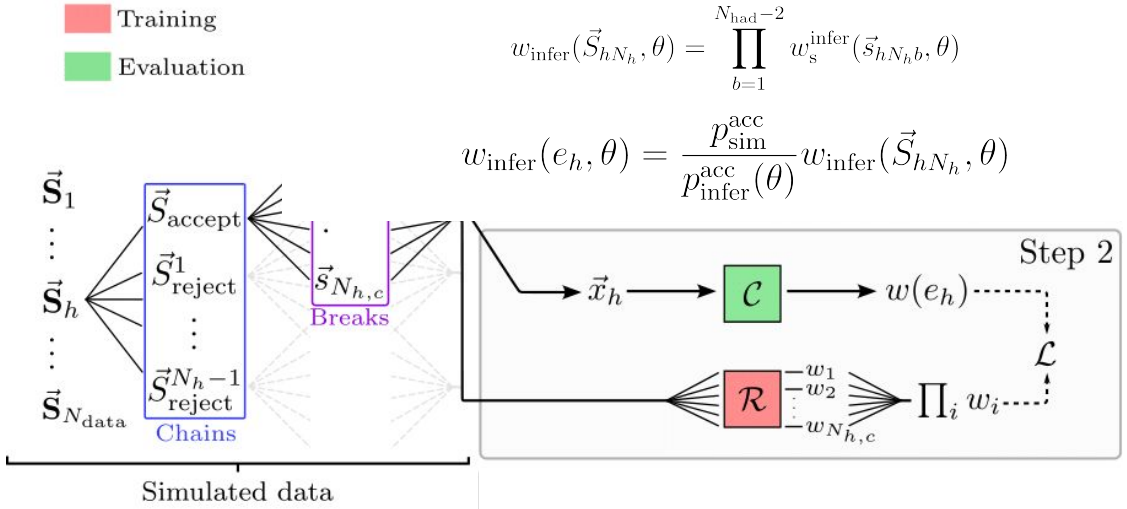
The first step can deal with **event-by-event observables (unbinned)** or with **collections of 1d histograms (binned)**.



Learning from data: HOMER

Histories and **O**bservables for **M**onte-Carlo **E**vent **R**eweighting: We take **Pythia as baseline** and reweight that.

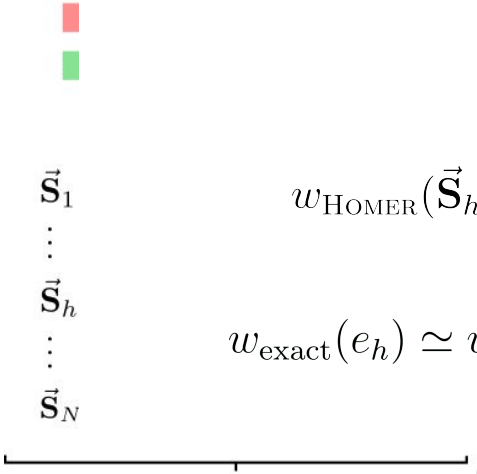
For the second step, we relate the **event weights** to the **fragmentation weights** and train using a task specific loss function



Learning from data: HOMER

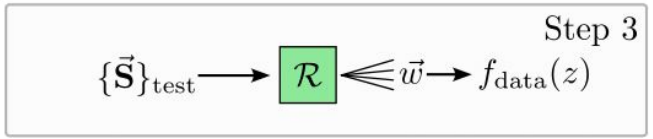
Histories and **O**bservables for **M**onte-Carlo **E**vent **R**eweighting: We take **Pythia as baseline** and reweight that.

After training, **we reweight each history** so that the reweighted run has the correct high-level distributions→ **Implicit definition of $f_{data}(z)$.**



$$w_{\text{HOMER}}(\vec{\mathbf{S}}_h) = \prod_{c=1}^{N_h} \prod_{b=1}^{N_{hc}} w_{\text{s}}^{\text{infer}}(\vec{s}_{hcb}, \theta)$$

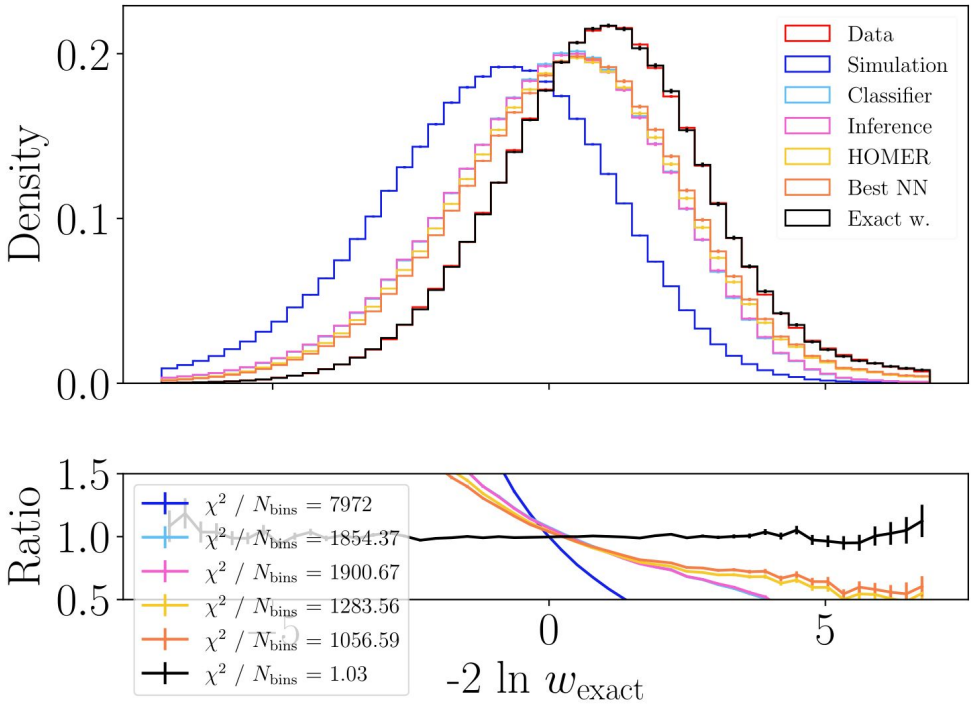
$$w_{\text{exact}}(e_h) \simeq w_{\text{HOMER}}(e_h) \equiv \langle w_{\text{HOMER}}(\vec{\mathbf{S}}_h) \rangle_{e(\vec{\mathbf{S}}_h)=e_h}$$



New underlying model

We are using pseudo-data generated with a different set of Pythia parameters → We know **the optimal observable** to distinguish between simulation and data.

This observable shows that our model may be very good but still not perfect → **Degeneracies in high-level observables + imperfect NN modelling**.



Comparing all three measurement choices

The best performance is for **unbinned high-level observables**. It is a good compromise between correlation and inductive bias.

However, the drop-off is not too large for **binned high-level observables**. The correlations are **implicitly approximated via the simulator**.

The point cloud performance is **limited by finite statistics**: we are “wasting” events learning a good representation that is sensitive to hadronization.

Comparisons between choices are dependent on our **simplified benchmark** and the fact we’re performing a **closure test**.