Data-driven hadronization models

ML4Jets, LPNHE, Paris, France
05/11/24

Manuel Szewc

University of CINCINNATI

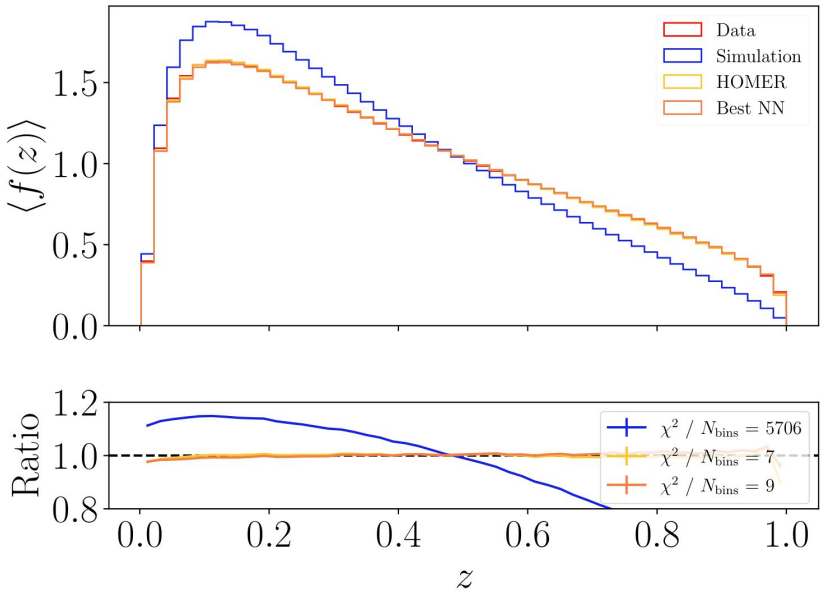
https://uchep.gitlab.io/mlhad-docs/

ICAS

In this talk

I hope to convey how **Hadronization** is a complicated and worthwhile target and mention two recent developments:

- RSA to leverage **automatic differentiation** to **enhance hadronization tunes**
- HOMER to learn a **data-driven hadronization model**



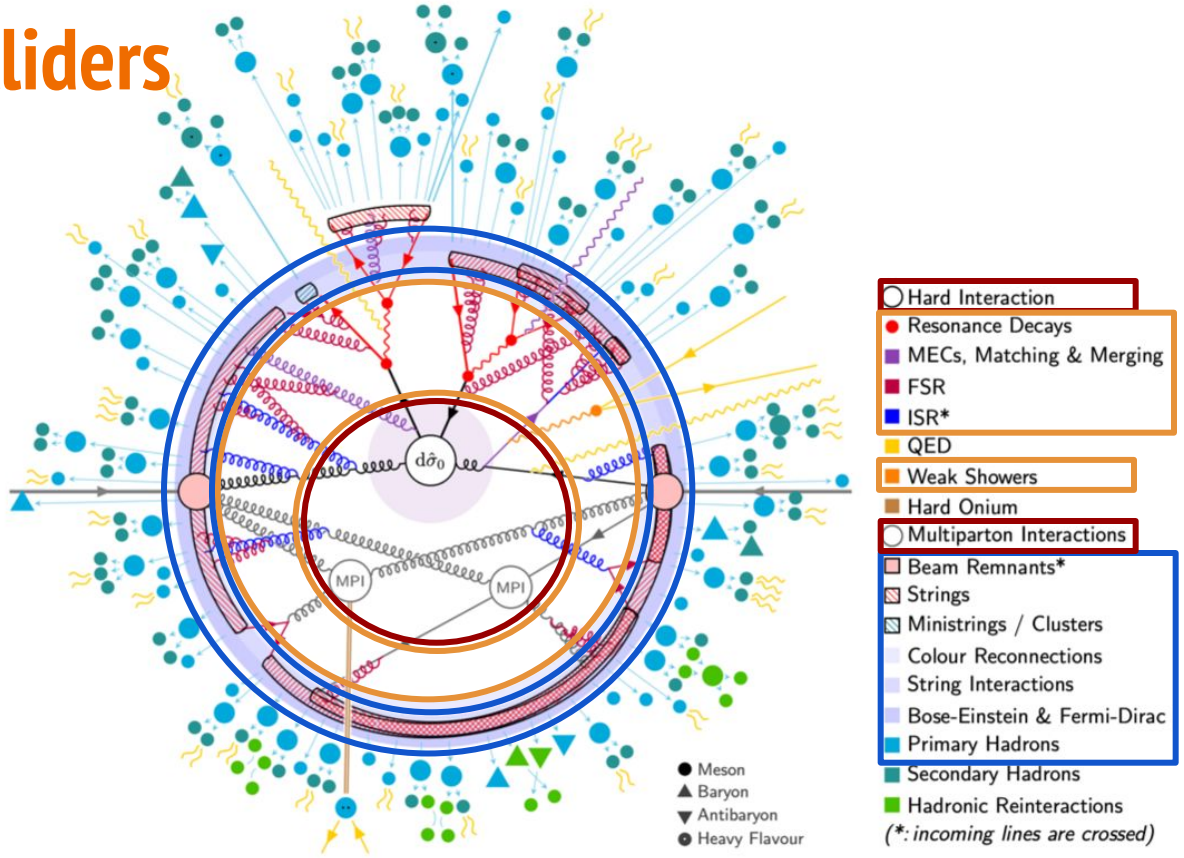
Hadronization at colliders

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.**

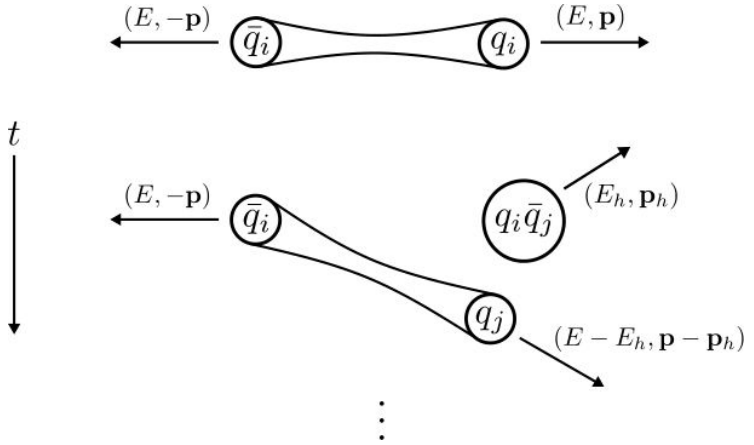
Hadronization: combining partons into **measurable hadrons**, non perturbative → **Empirical models**.



Hadronization empirical models

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

Tuned simulators are **very** successful. **However**, we are pushing the models to their limits. Collective effects in general are tricky to recover e.g. heavy baryon production at high event multiplicities as in arxiv:1807.11321.



Lund String Model: Colored singlets + ~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!

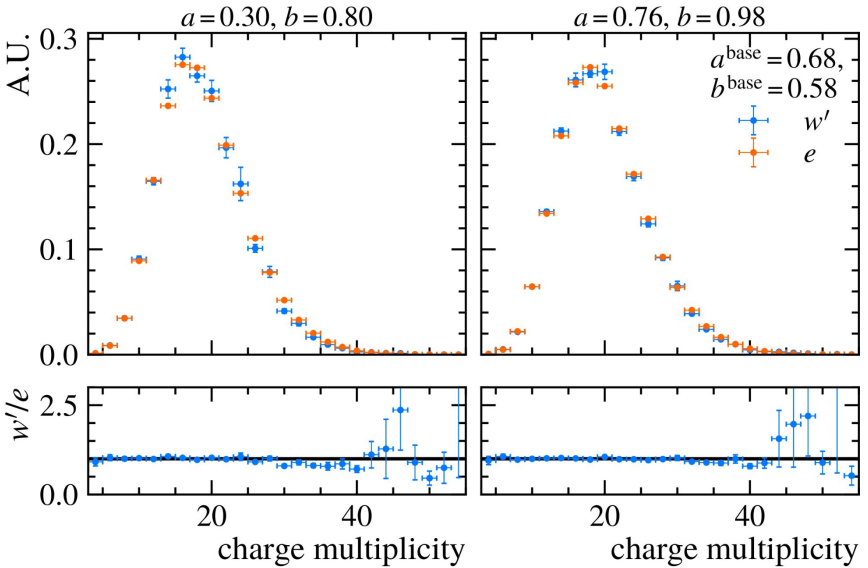
A lot of possible ways to attack this problem. The richness of the involved physics **forbids the use of any plug-and-play algorithms**.

Two groups have recently tackled the subject: **MLHAD** (arxiv:2203.04983, arxiv:2311.09296, arxiv:2410.06342) 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**.

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**".



10^6 events per parameter choice

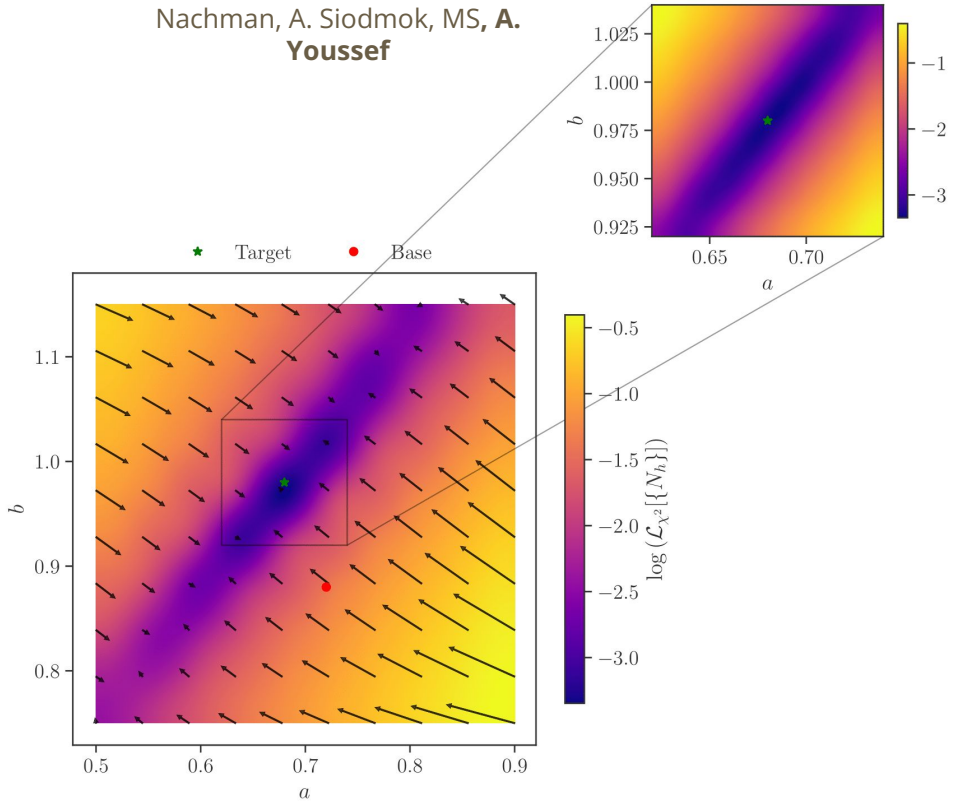
More efficient tunes: RSA

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**



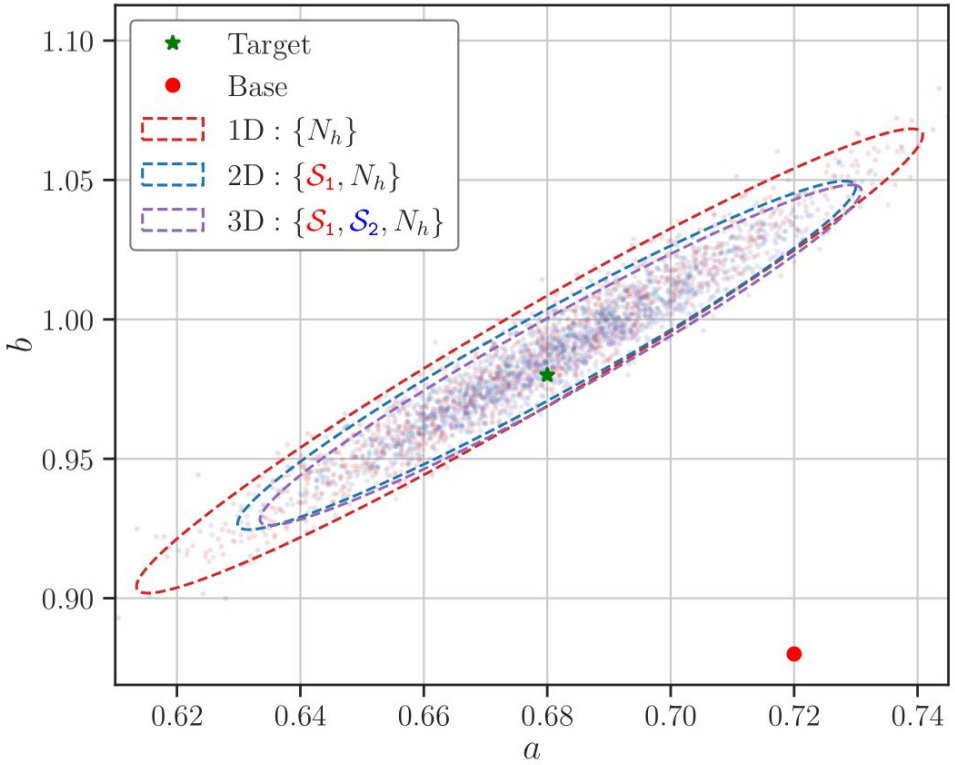
Take home message

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 a data-driven reweighting

A more ambitious goal is a **data-driven fragmentation function.**

Reweighting can help! Assume the Lund string model is **appropriate as a reference model to reweight from.** Weights should be such that simulations and data indistinguishable at the **measurable level**. **Data-driven weights, data-driven fragmentation function.**

Two step procedure: We learn **how the measurable quantities should look like**, and then train a fragmentation function reweighter that **morphs simulated events** accordingly.

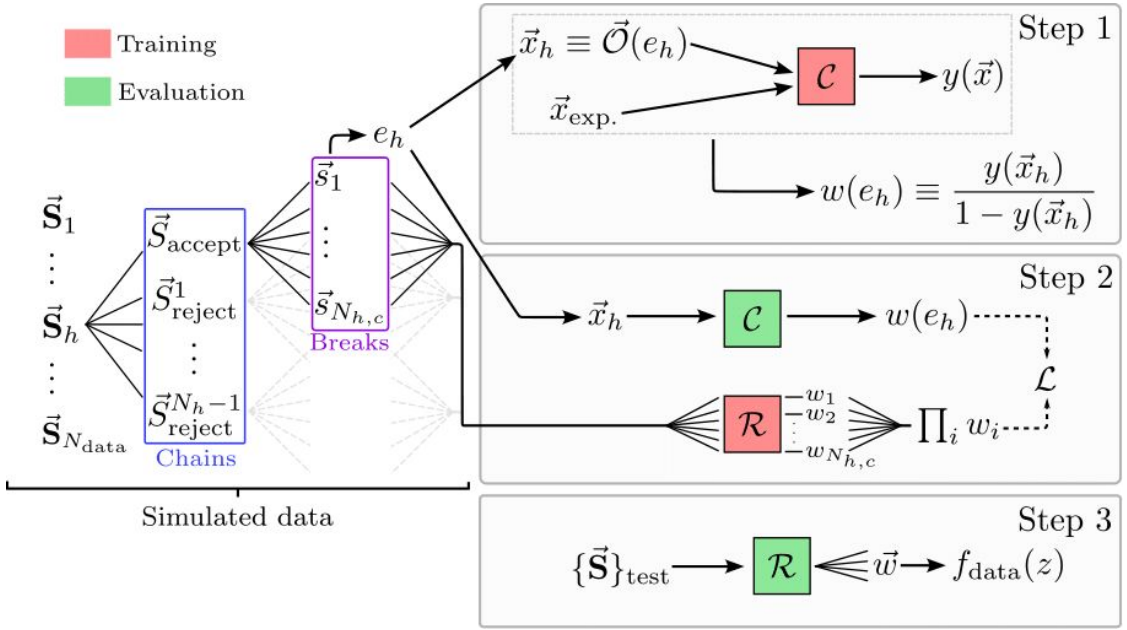
A nice **bonus**: we can **more easily incorporate subtleties of the Lund String fragmentation model** built into Pythia by a dedicated effort of 40+ years!

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.**

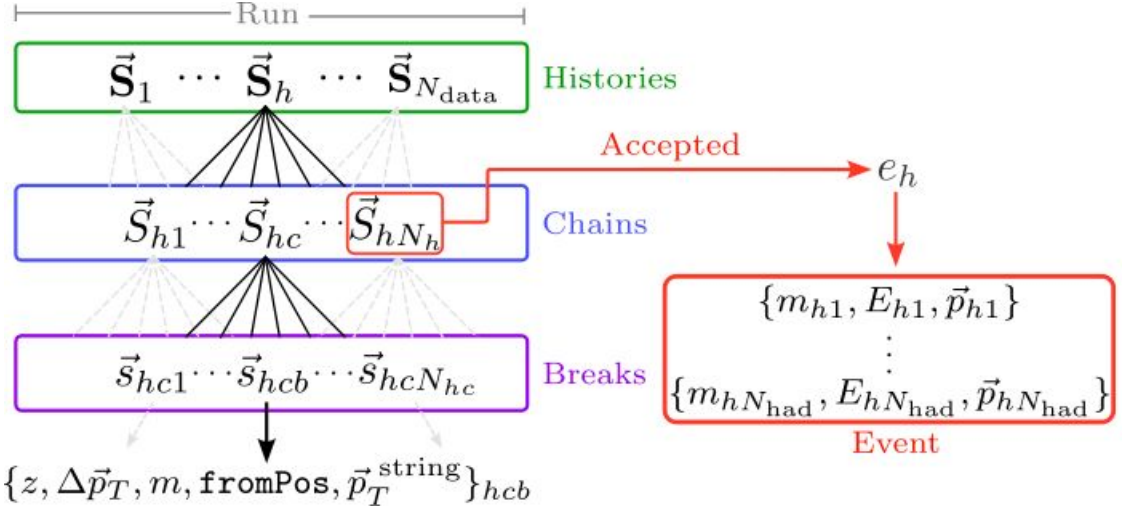
(No rejection sampling needed here!)



Learning from data: HOMER

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

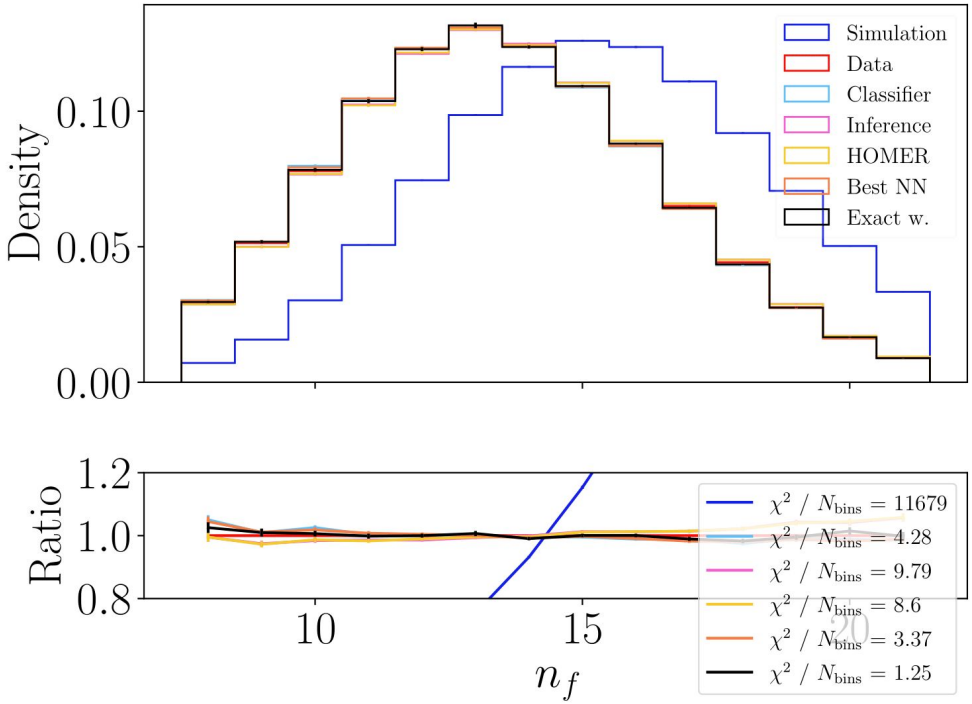
We restrict ourselves to the **qq string emitting only pions**. For simulation, we record everything. **For pseudo-data, only observable quantities**.



Matching sim to data

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

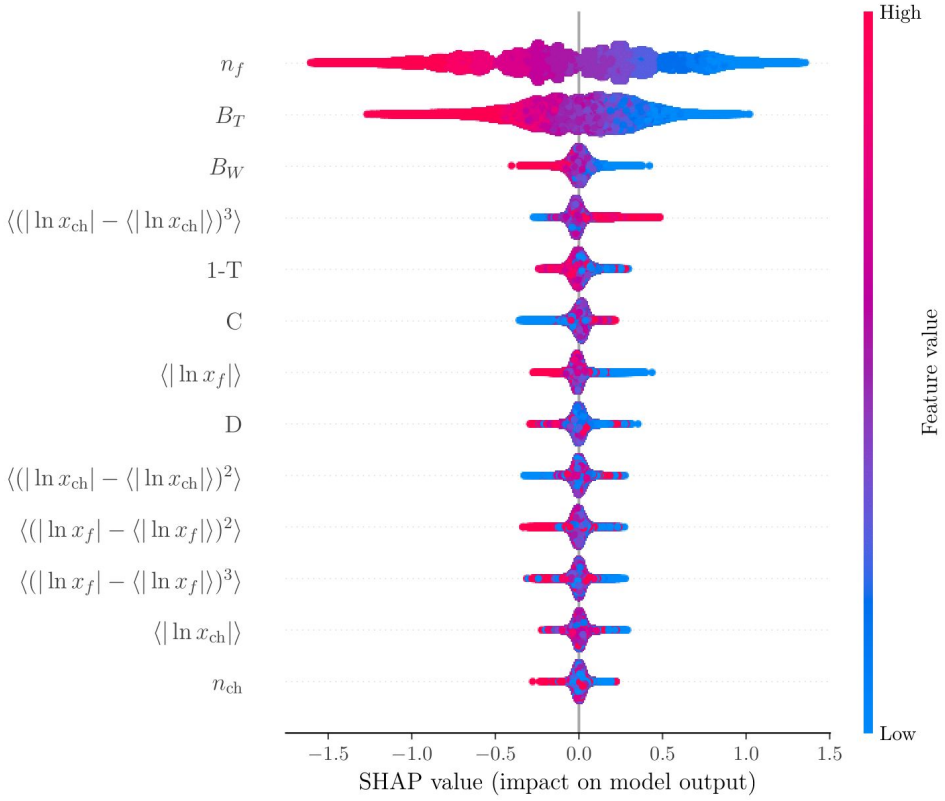
The agreement is better for step 1 than for step 2 / full weights due to the combination of **additional bias + imperfect training**.



Matching sim to data

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.

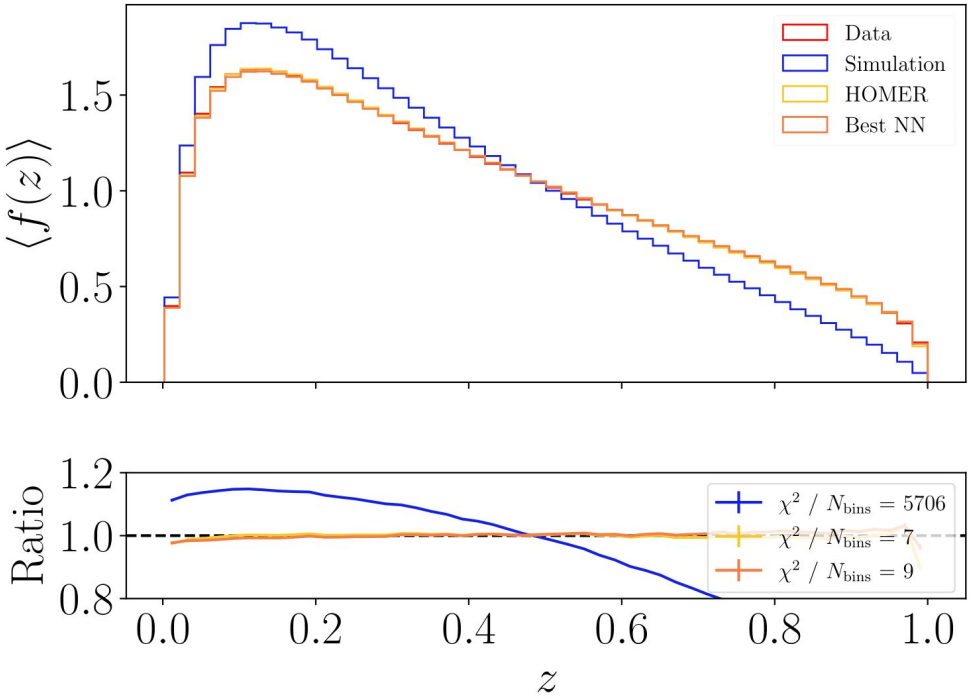


New underlying model

The weights are translated to a **data-driven fragmentation function**.

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.

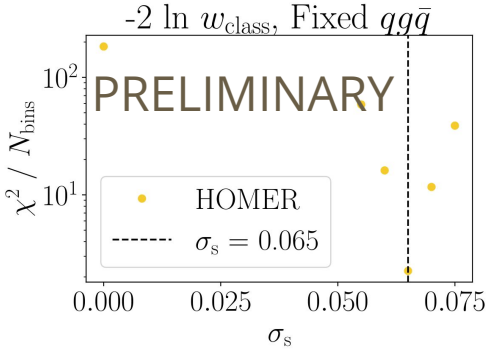


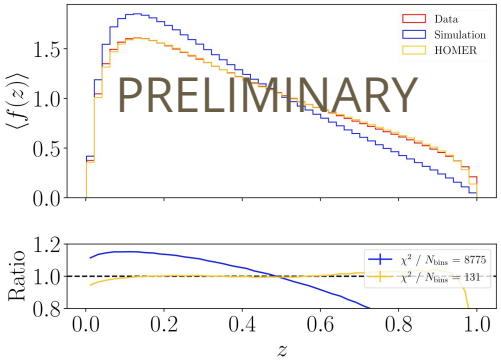
Take home message

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**, with slight modifications to be shown in forthcoming publication (arxiv:241X:XXXX?).





Conclusions

Hadronization is the type of problem you dream of if you want to work in ML for HEP: **physically meaningful** and **complicated enough** that it is not simply a case of plug-and-play with any ML algorithm.

Data representation, simulation-based inference and latent generative models all come into play here.

Any developments impact existing tuning efforts and uncertainty estimations.

This is very much an open problem, so **feedback is necessary**!

The MLHAD team

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





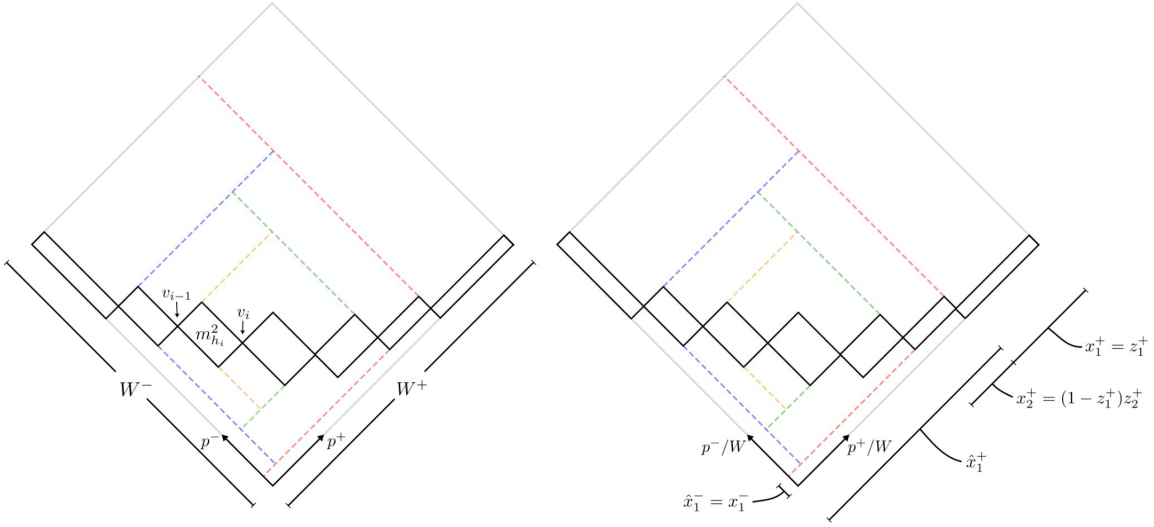




Thanks!

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 the Lund Fragmentation model

First task: learn the **Lund String hadronization pdfs** for $e^+e^- \to q\bar{q}$

Checks feasibility of the problem.

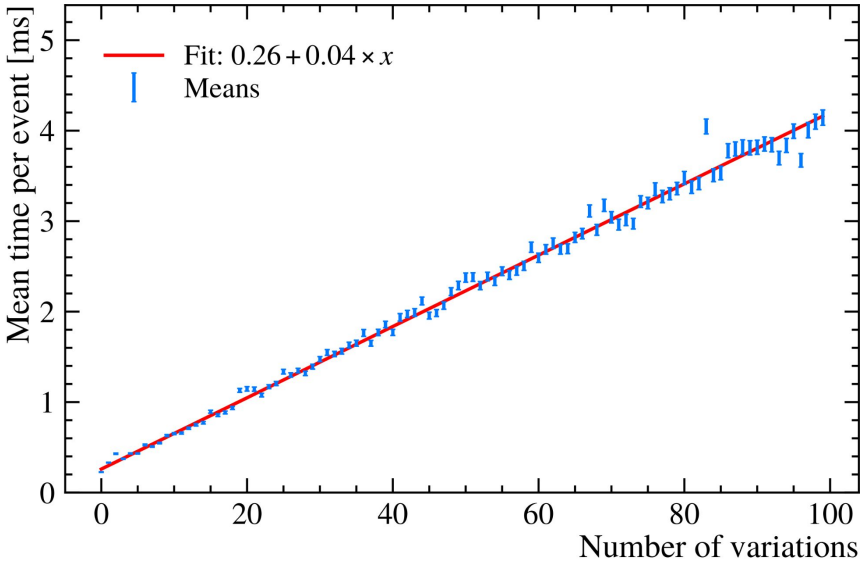
Introduce **inductive bias**. Improve over the existing empirical model by first mapping it to a learnable model.

The first hadronization pdf can be iteratively applied to get a full chain.

$$p_{\mathrm{Lund}}(x) \to p_{\mathrm{ML}}(x)$$

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}	

However...

These efforts aimed to learn a first hadronization function and then fine tune it using data. Our baseline generates (p_z,p_T) directly based on **our simplified Lund string model**.

However, this ignores a lot of subtleties of the Lund String fragmentation model and how data is produced from this. These subtleties are built into Pythia by a dedicated effort of 40+ years!

We need to take advantage of that.

Graph Neural Networks (GNNs)

A more general representation of data. The Neural networks are applied over **graphs** defined by **nodes, features and edges** with a process called **Message passing.** The Neural Network updates each node by looking at its relation to its neighbors and learns an **embedding** of the graph to be used **downstream**.

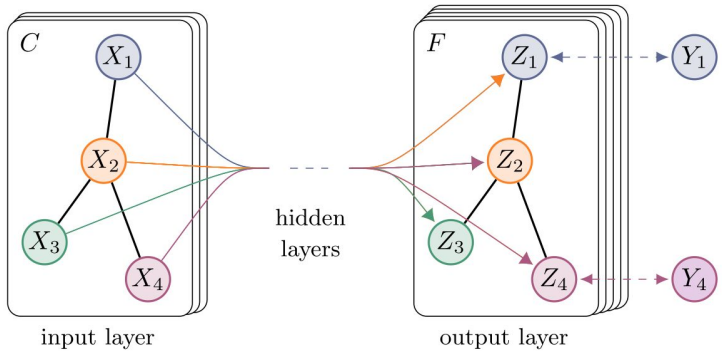


Image from https://arxiv.org/abs/1609.02907

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**.