Modeling Hadronization using Machine Learning

DIS2023

Tony Menzo

PhD candidate, University of Cincinnati

In collaboration with:

Phil Ilten, Stephen Mrenna, Manuel Szewc, Michael Wilkinson, Ahmed Youssef, and Jure Zupan

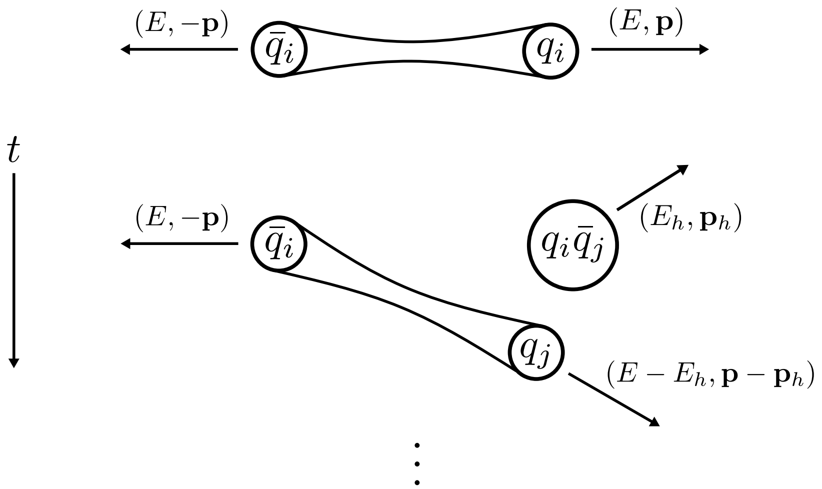
Based upon work in 2203.04983, 23xx.xxxxx

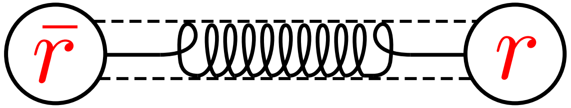
Stringy Hadronization

Early 80s brought many non-perturbative hadronization models: Cluster, percolation, …

Lund String Model

(used in Pythia)

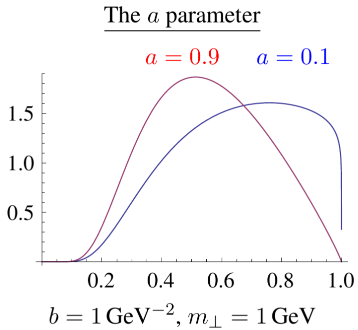


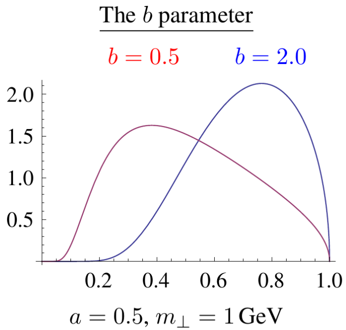


The momentum fraction z of each fragmenting hadron is sampled according to the

Lund fragmentation function

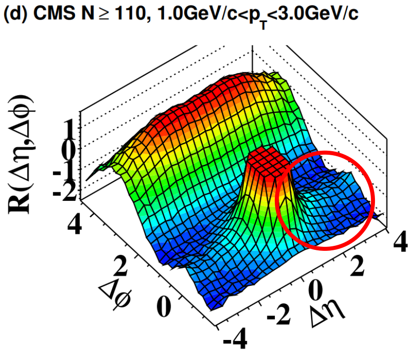
$$f(z) \propto \frac{(1-z)^a}{z}\exp\left(\frac{-bm_\perp^2}{z}\right) \qquad z = \frac{p_z + E_h}{2E}$$



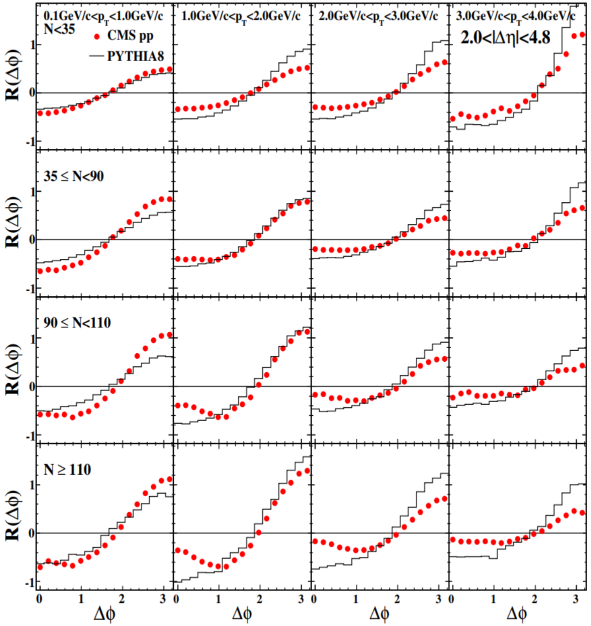


Room for improvement

- Discrepancies in high multiplicty events i.e. enhanced particle production around the azimuthal angle of a trigger jet (CMS) “the ridge”

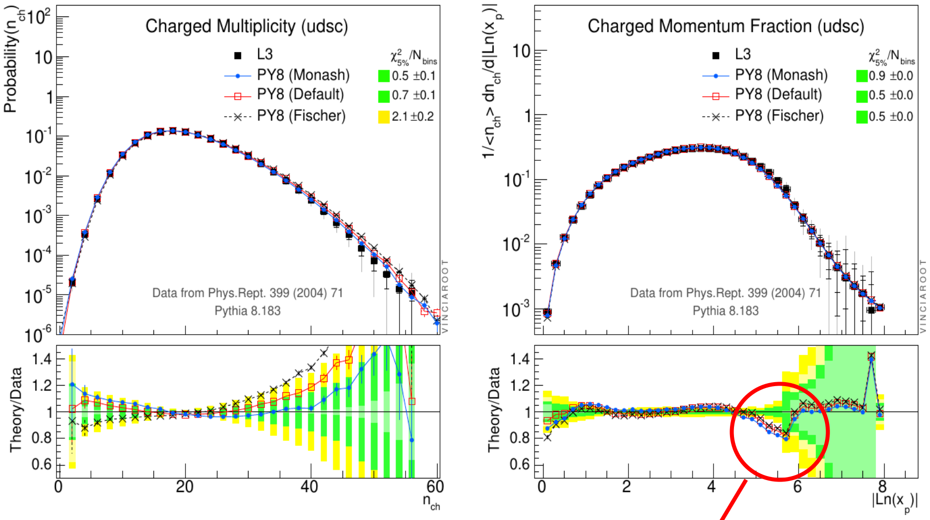


1009.4122



- Unavoidable discrepancies in parameter tuning

$$f(z) \propto \frac{(1-z)^a}{z} \exp\left(\frac{-bm_\perp^2}{z}\right)$$



Monash Tune (a,b,σ_{perp}) 1404.5630

Cannot be improved by retuning

Motivation

The main motivation is to create a better simulation of collider events.

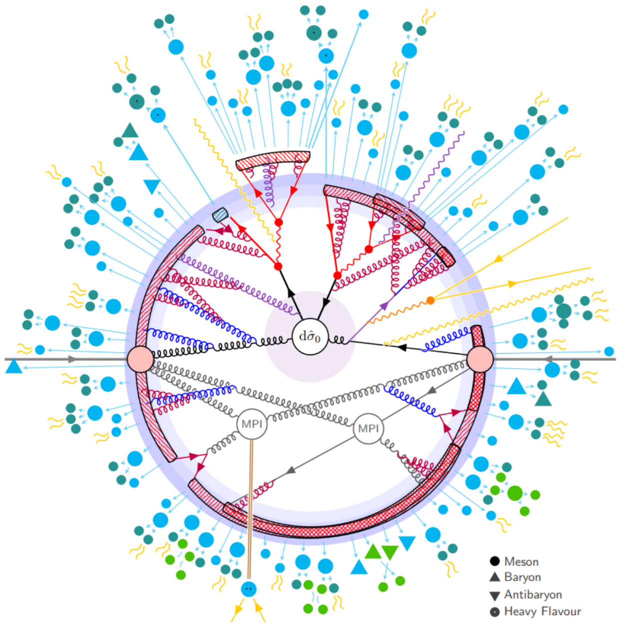
But also to promote a paradigm shift in the modeling of non-perturbative physics.

Goal of event generators:

Predict experimentally measured distributions **from microscopic dynamics (SM + nonperturbative models)**.

NLO, NNLO, N^{3}LO, ...

???



2203.11601

How to improve the generator: two* approaches

- Improve model
  - MPIs, rope hadronization, transverse mass supression, flavor asymmetries, hadronic rescattering, multiscale models (string → hydrodynamical), flavor selector, etc.
  - Utilize techniques from gauge-gravity duality

  Hard to come up with mathematically precise model without established calculational techniques

- Data-driven generator
  - Sample directly from global distributions

  Non-universal and extremely difficult to convert into representative particle flow data

*** or a combination of both (machine learning methods)**

Where can/will machine learning be useful in event generators?

1. Event generation
    a. Input experimental/simulated data → Output replica data
        i. Generative machine learning algorithms

2. Parameter tuning
    a. Input model parameters → Output optimal parameters
        i. Hadronization has O(200 parameters), requires new tuning paradigm: Simulation based inference

3. Model exploration
    a. Input experimental/simulated data → Output potential models
        i. Hadronization models already do well! Symbolic regression + graph neural networks may allow for determination of perturbations

Where can/will machine learning be useful in event generators?

1. Event generation
   a. Input experimental/simulated data → Output replica data
      i. Generative machine learning algorithms

2. Parameter tuning
   a. Input model parameters → Output optimal parameters
      i. Hadronization has O(200 parameters), requires new tuning paradigm: Simulation based inference

3. Model exploration
   a. Input experimental/simulated data → Output potential models
      i. Hadronization models already do well! Symbolic regression + graph neural networks may allow for determination of perturbations

Where can/will machine learning be useful in event generators?

1. Event generation 2203.04983
   a. Input experimental/simulated data → Output replica data
      i. Generative machine learning algorithms

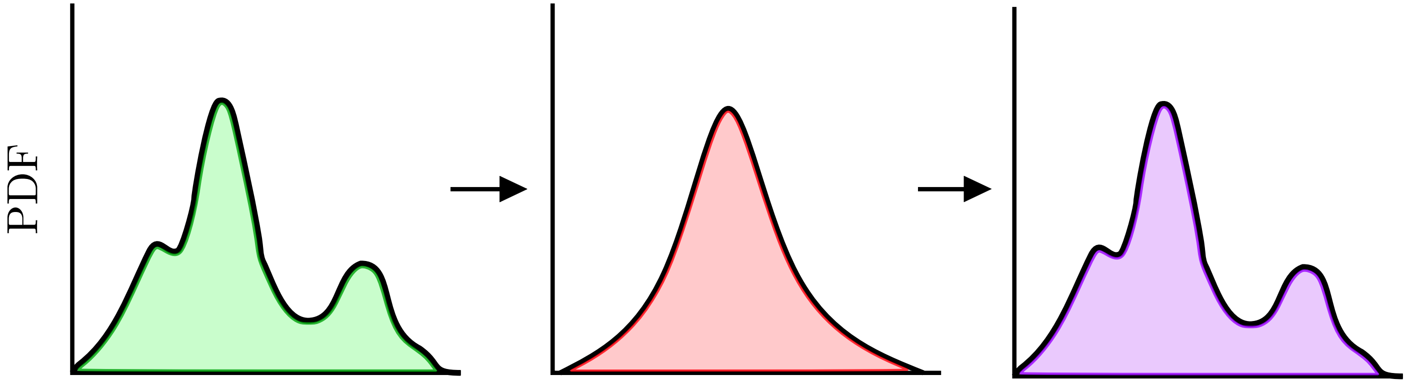
2. Parameter tuning
   a. Input model parameters → Output optimal parameters
      i. Hadronization has O(200 parameters), requires new tuning paradigm: Simulation based inference

3. Model exploration
   a. Input experimental/simulated data → Output potential models
      i. Hadronization models already do well! Symbolic regression + graph neural networks may allow for determination of perturbations

Generative machine learning

To make any headway we need a tool which will allow us to efficiently sample probability distributions whose analytic form is unknown.

Generative machine learning algorithms are the perfect tool!



Proof of concept (2203.04983)

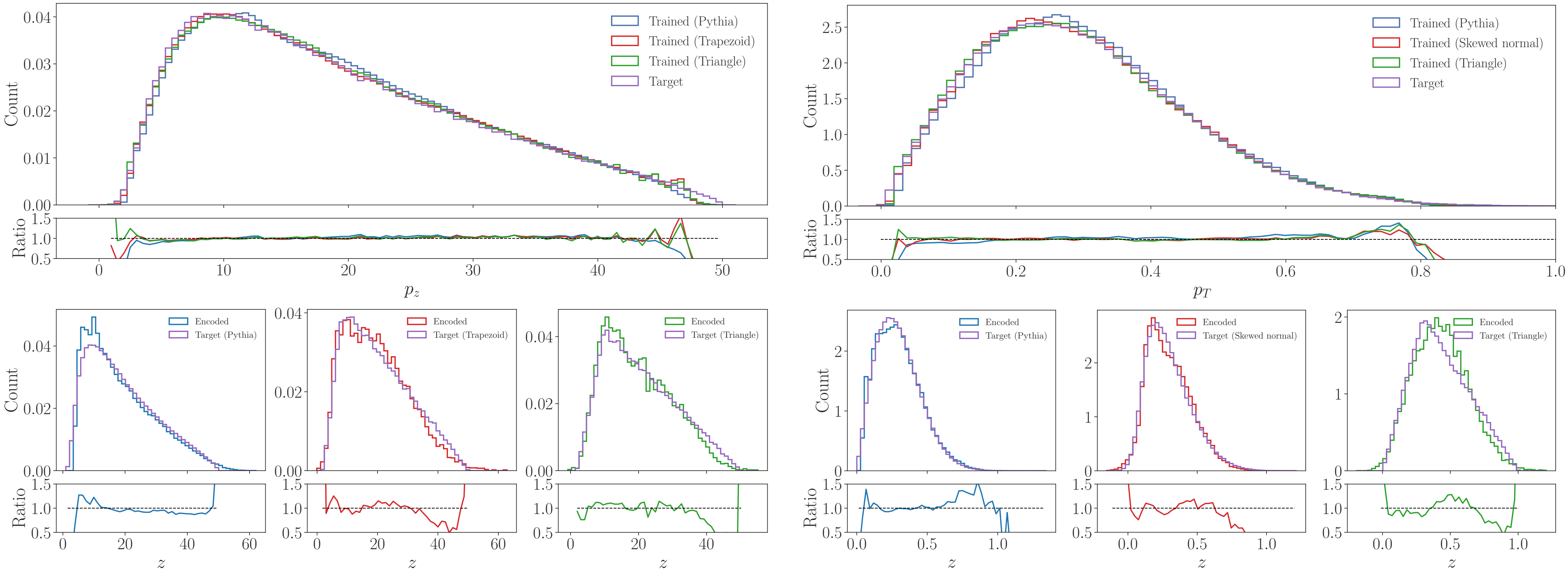
Consider Pythia output as 'experimental data' and try to reproduce hadronization observables by training on single emission kinematics (~learn the fragmentation function f(z)).

Start from simplest hadronizing system:

1. $q\bar{q} \rightarrow \pi$'s
2. Assume no correlations between emissions
3. E_{cut}~5 GeV (To avoid termination effects)

Train on p_z and p_T distributions of 1st emitted π

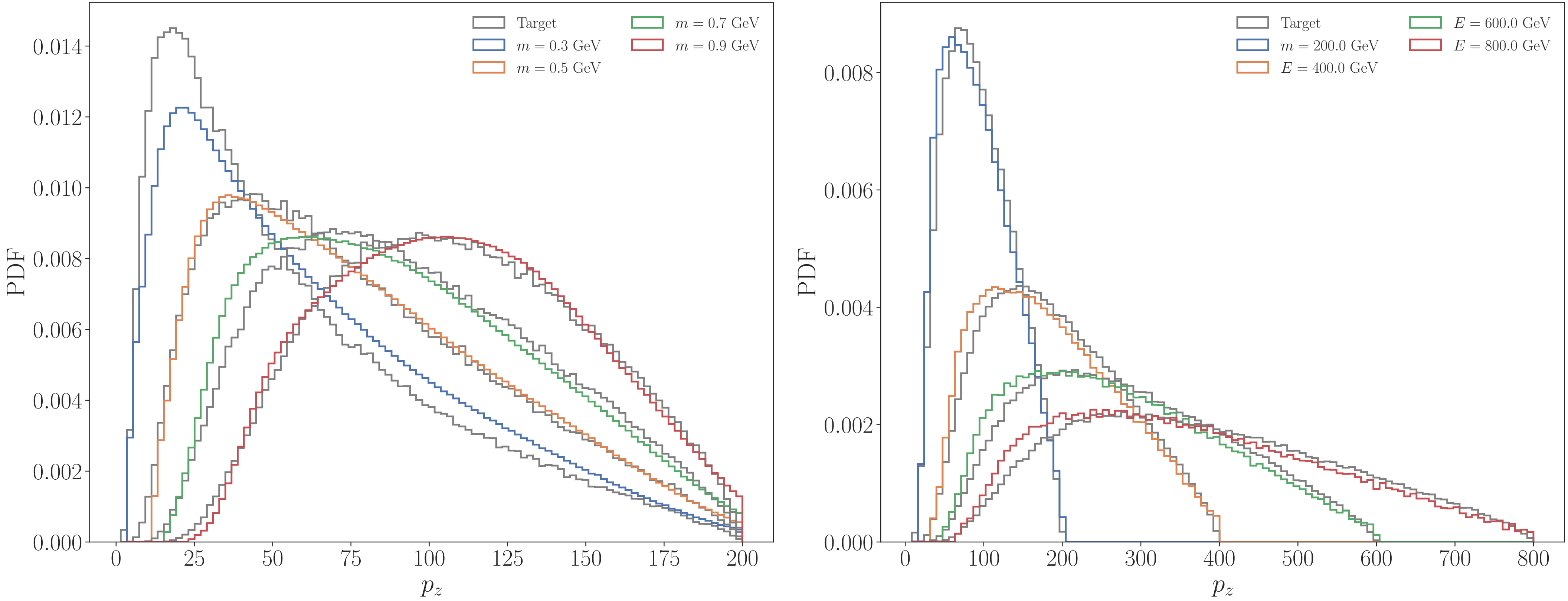
Training Results (cSWAE)



Training Results

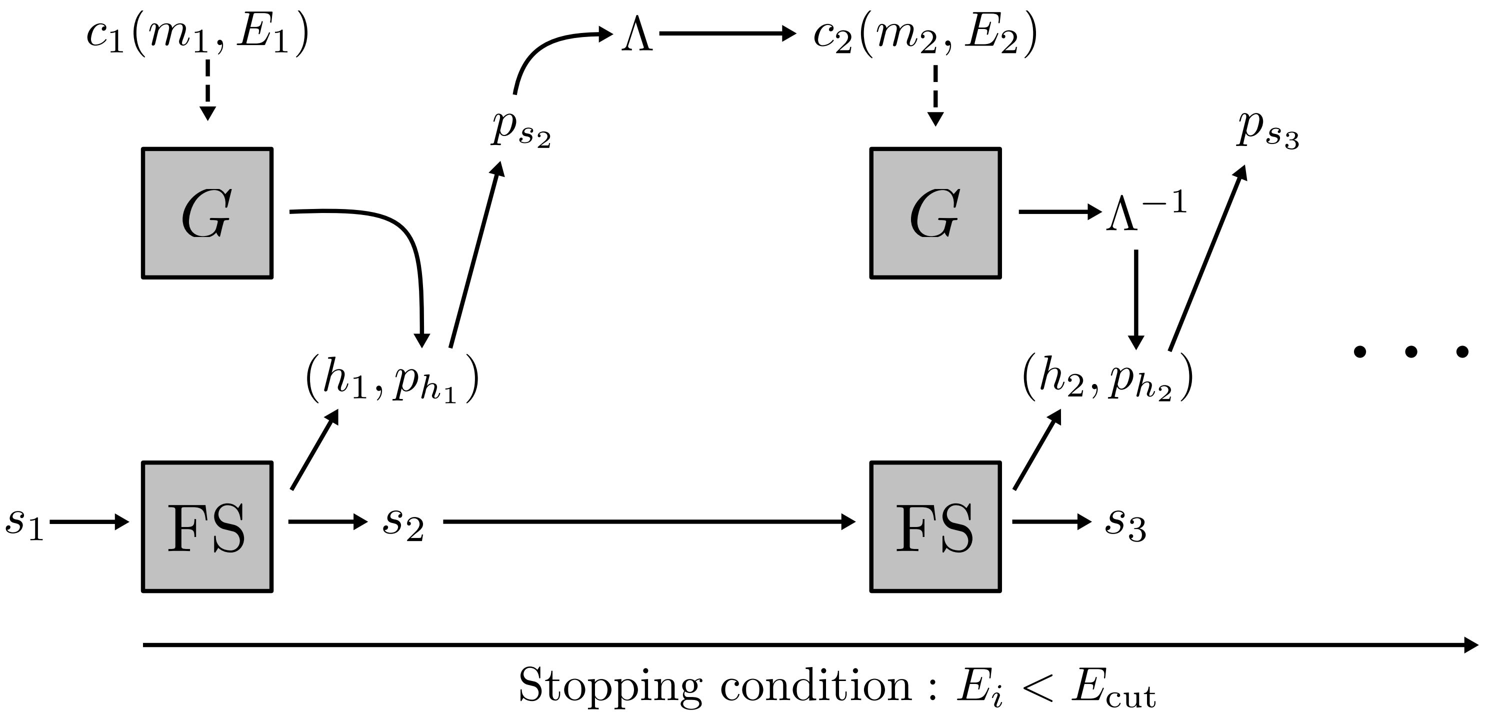
(cSWAE with labels and boundaries)

***Preliminary**

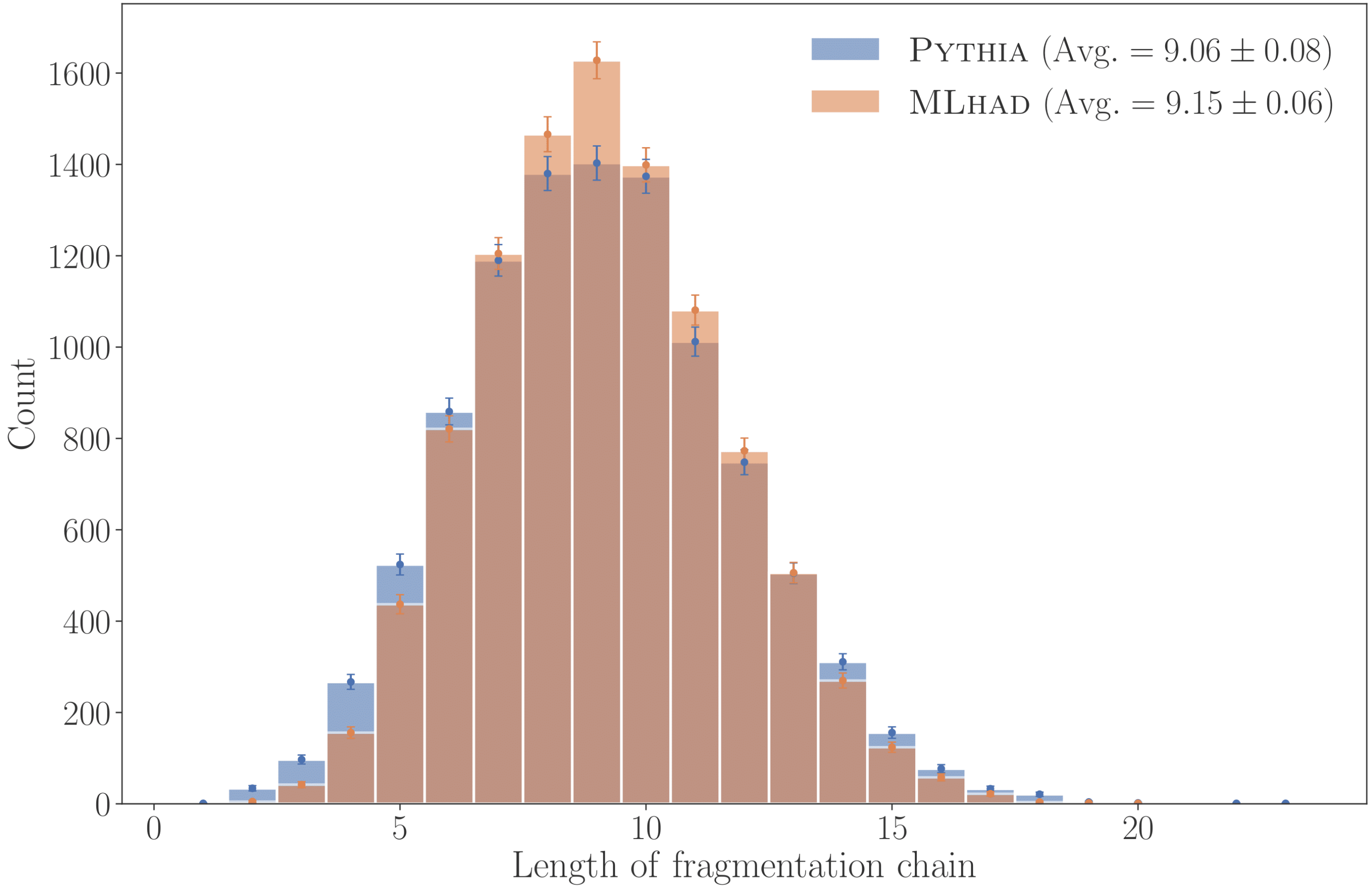


Hadronization (kinematics + flavor selector)

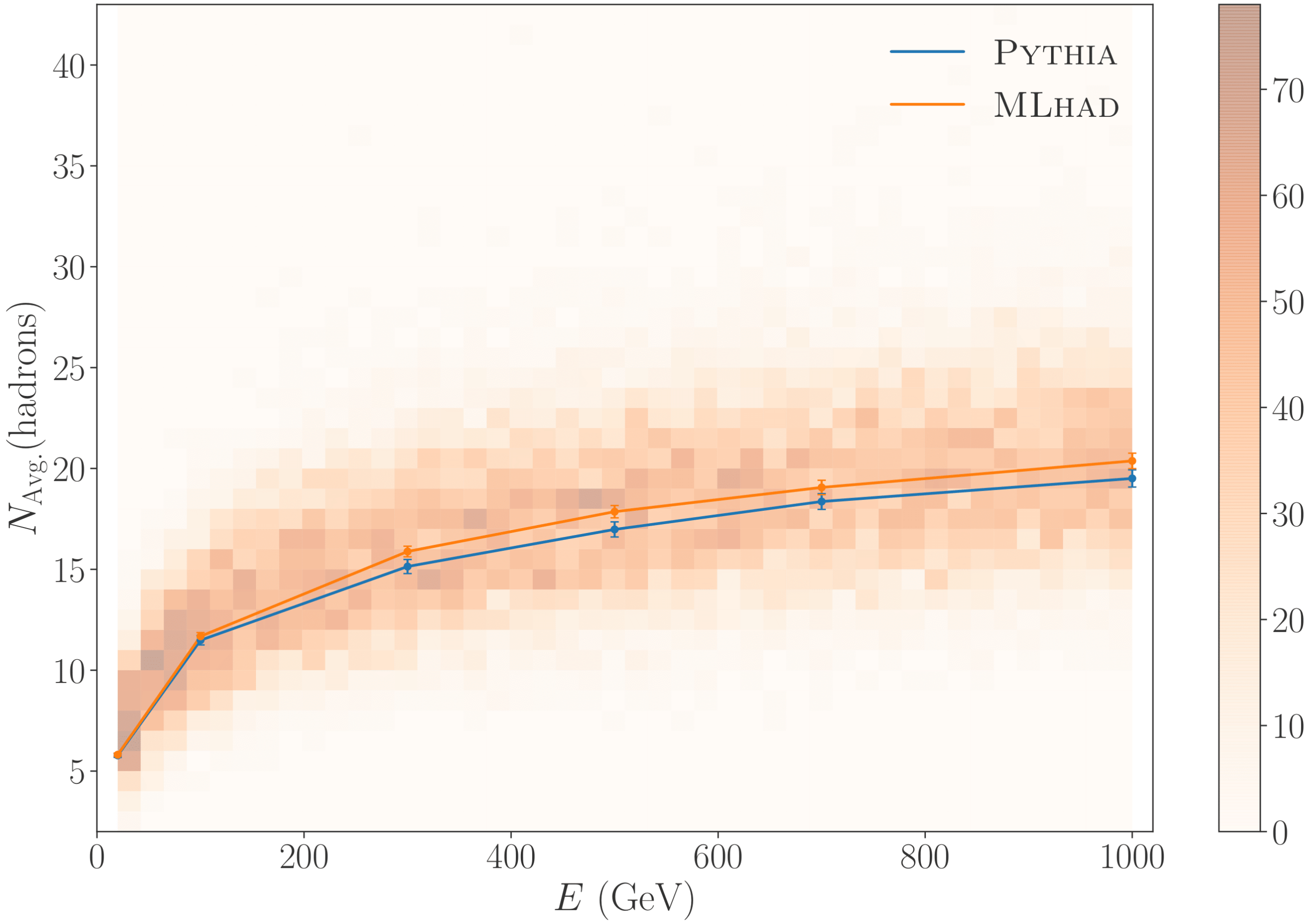
The trained model distributions now need to be integrated into a chain of fragmentations



Global observable (Hadron multiplicity cSWAE)



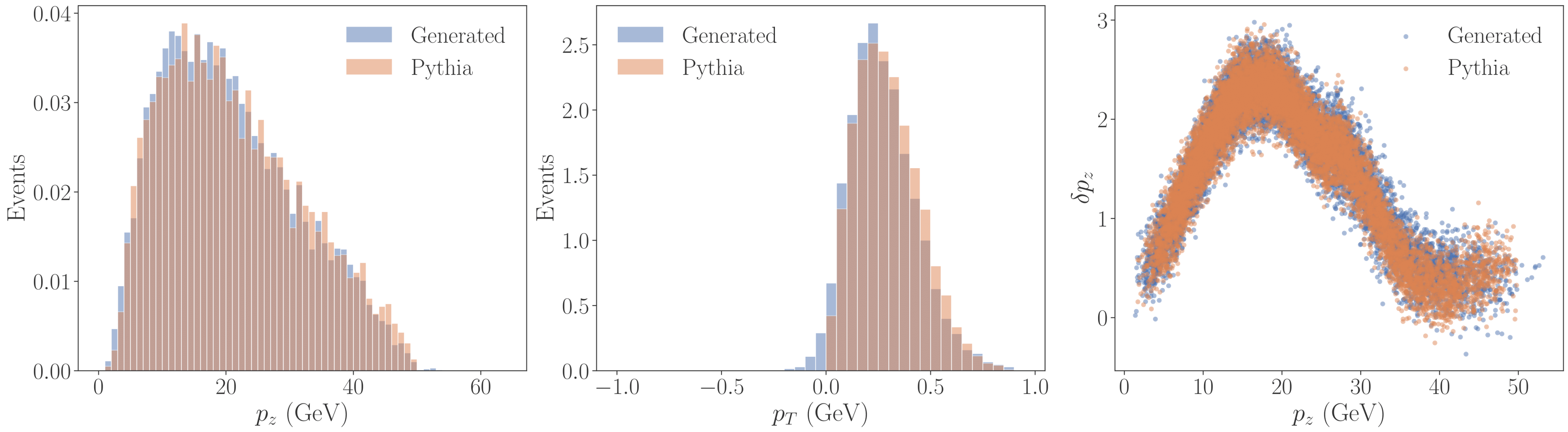
Global scaling (Hadron multiplicty vs string energy cSWAE)



Error estimation (BINN)

Incorportating (theoretical/experimental) errors from training dataset errors into the hadronization simulation

***Preliminary**



Conclusion

Model + machine learning methods **CAN** be used to implement hadronization within event generators and provide an explicit path for improvement.

What's next:

- **ML-improved (data-improved) model of hadronization**
- **ML flavor selector**
- **Hadronization tuning**
- **Error estimation**

Check out our repo!

MLHAD

https://gitlab.com/uchep/mlhad

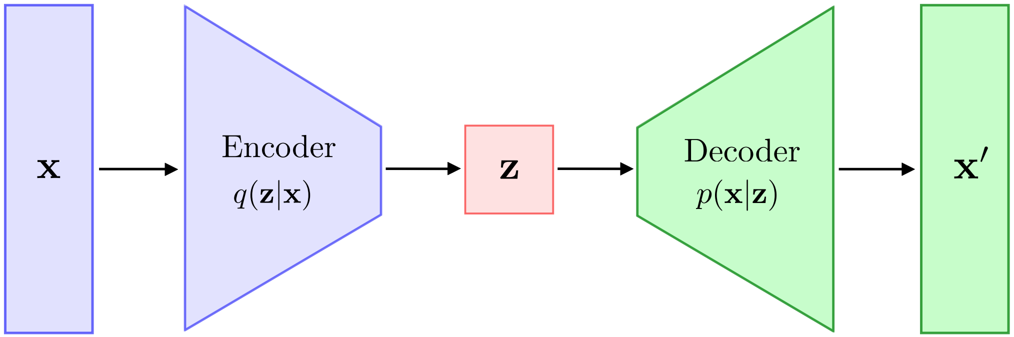
Check out our paper:

arXiV: 2203.04983

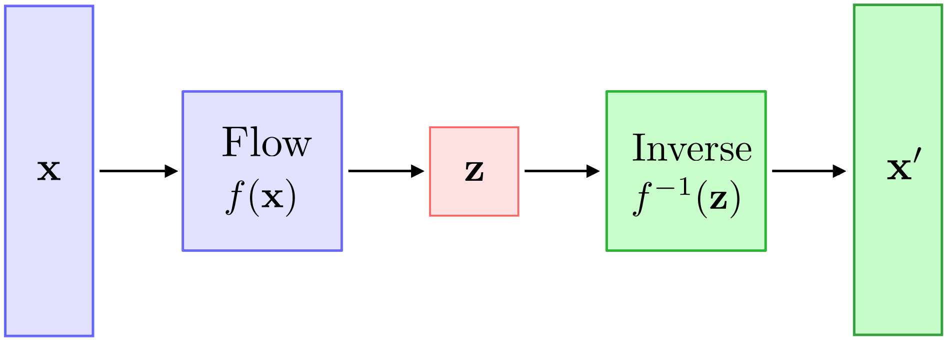
Back-up

Architectures

Conditional sliced-Wasserstein Autoencoder (cSWAE)



Conditional normalizing flow (cNF)



Training Results

(cNF with labels)

***Preliminary**