MLHAD

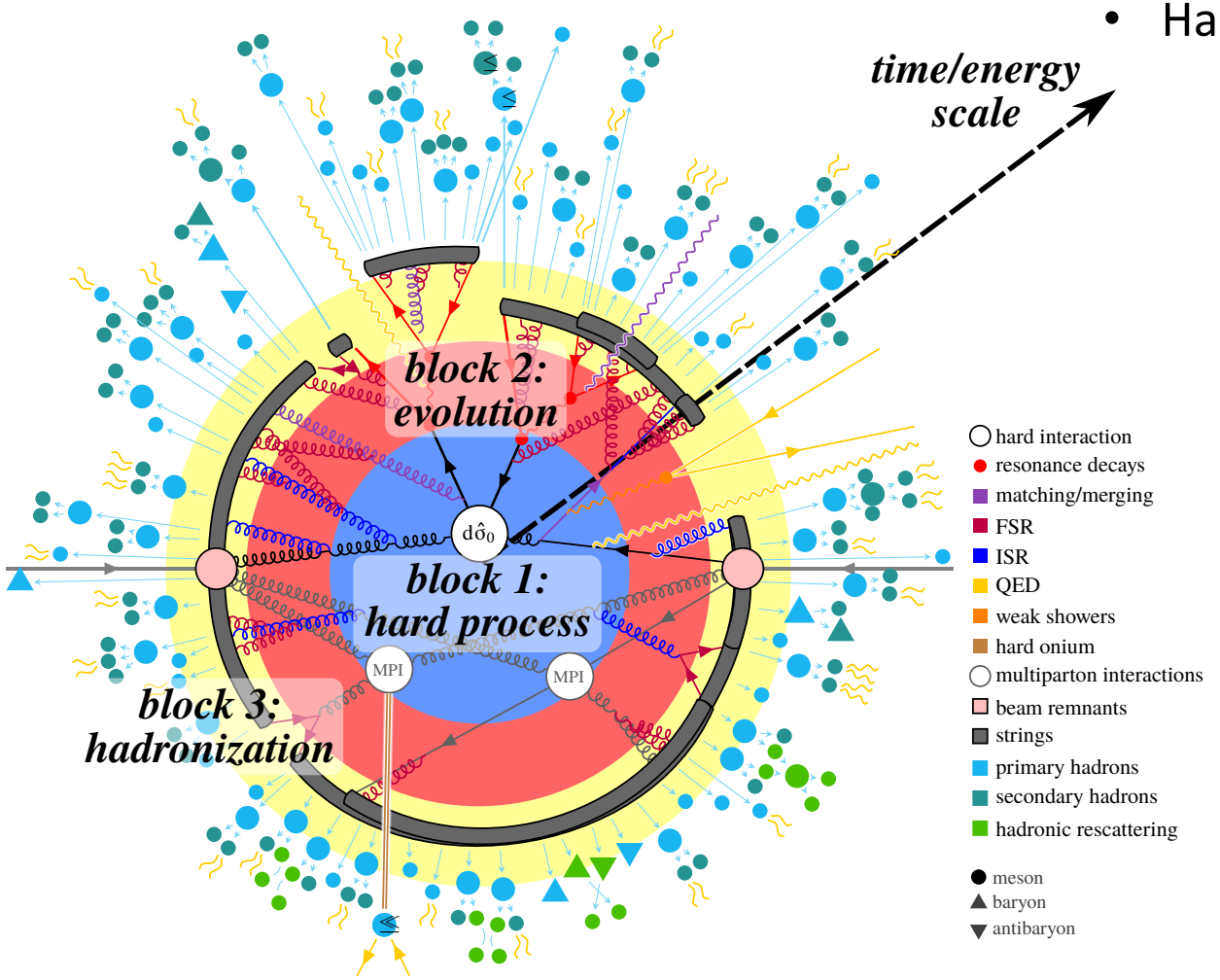
MLhad:
Simulating Hadronization with Machine Learning

Phil Ilten, Tony Menzo, Steve Mrenna,
Manuel Szewc, Michael K. Wilkinson,
Ahmed Youssef, Jure Zupan

8 May 2023

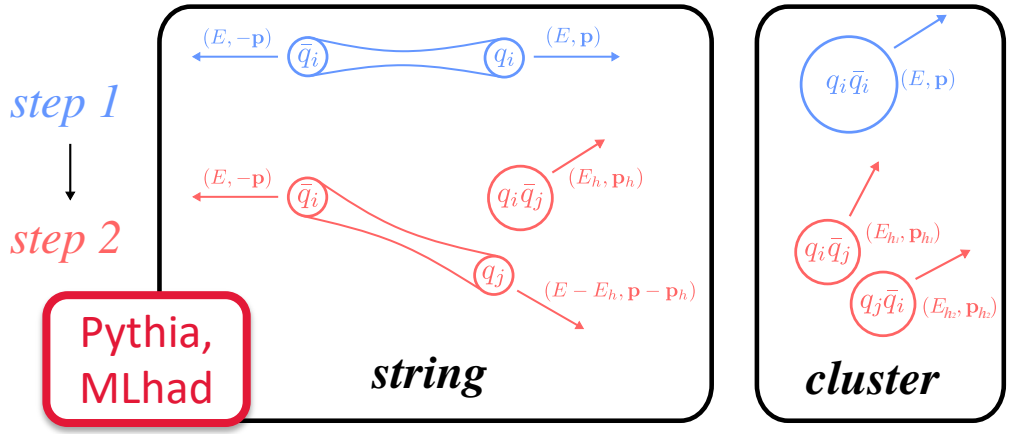
Hadronization

Simulating collisions



- Hard process: initial high-energy interaction (perturbative)
- Evolution: parton shower (perturbative)
- Hadronization: combine quarks and gluons (non-perturbative)

Hadronization models

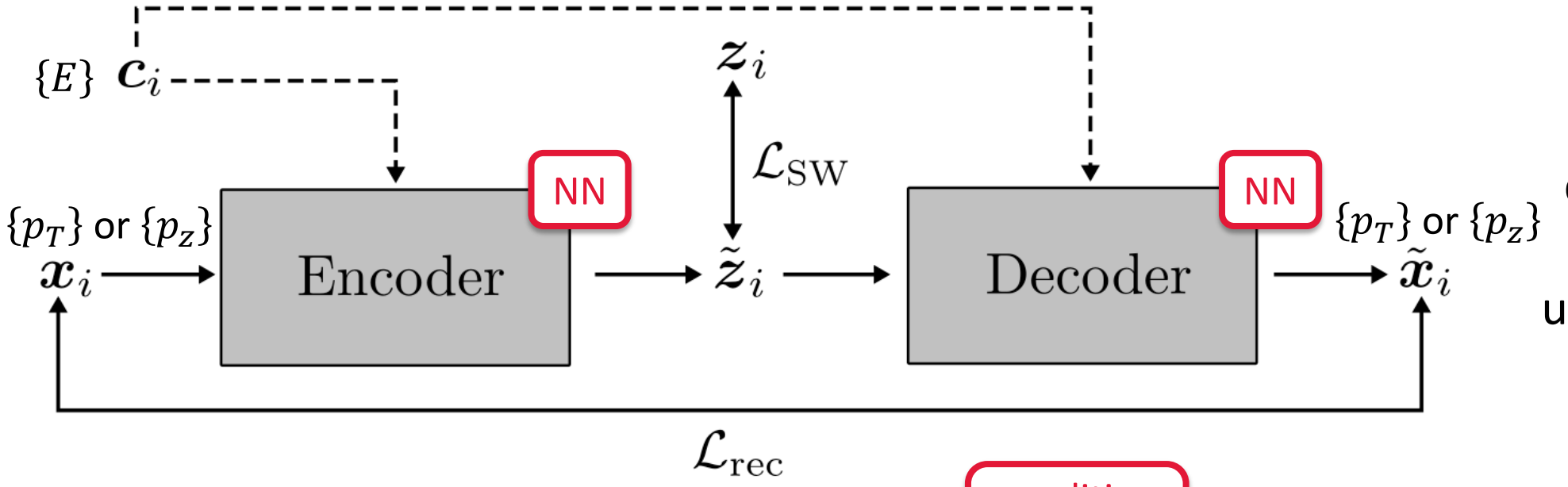


- Two primary models:
  - String: iteratively split parton connected by QCD color strings with linear potential (Lund model)
  - Cluster: pre-confine partons into proto-clusters, then split by two-body decays
- QCD does not determine splitting—opportunity for ML

cSWAE architecture

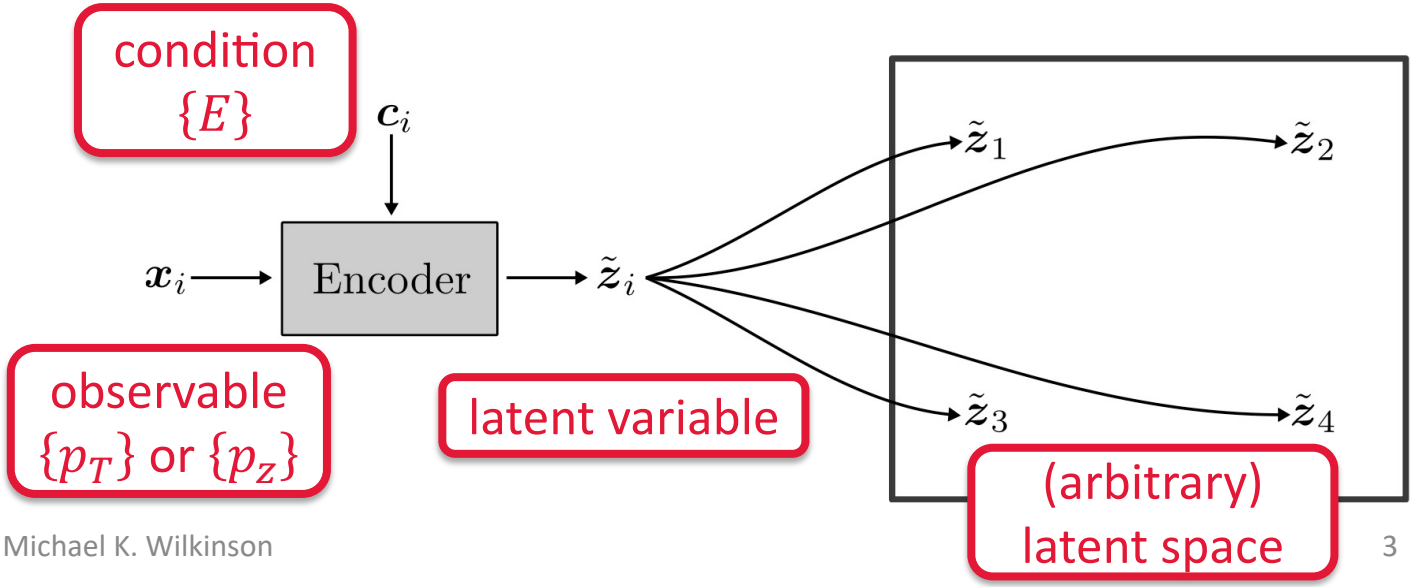
conditional Sliced-Wasserstein AutoEncoder

SciPost Phys. 14, 027 (2023)
arXiv:2303.04983



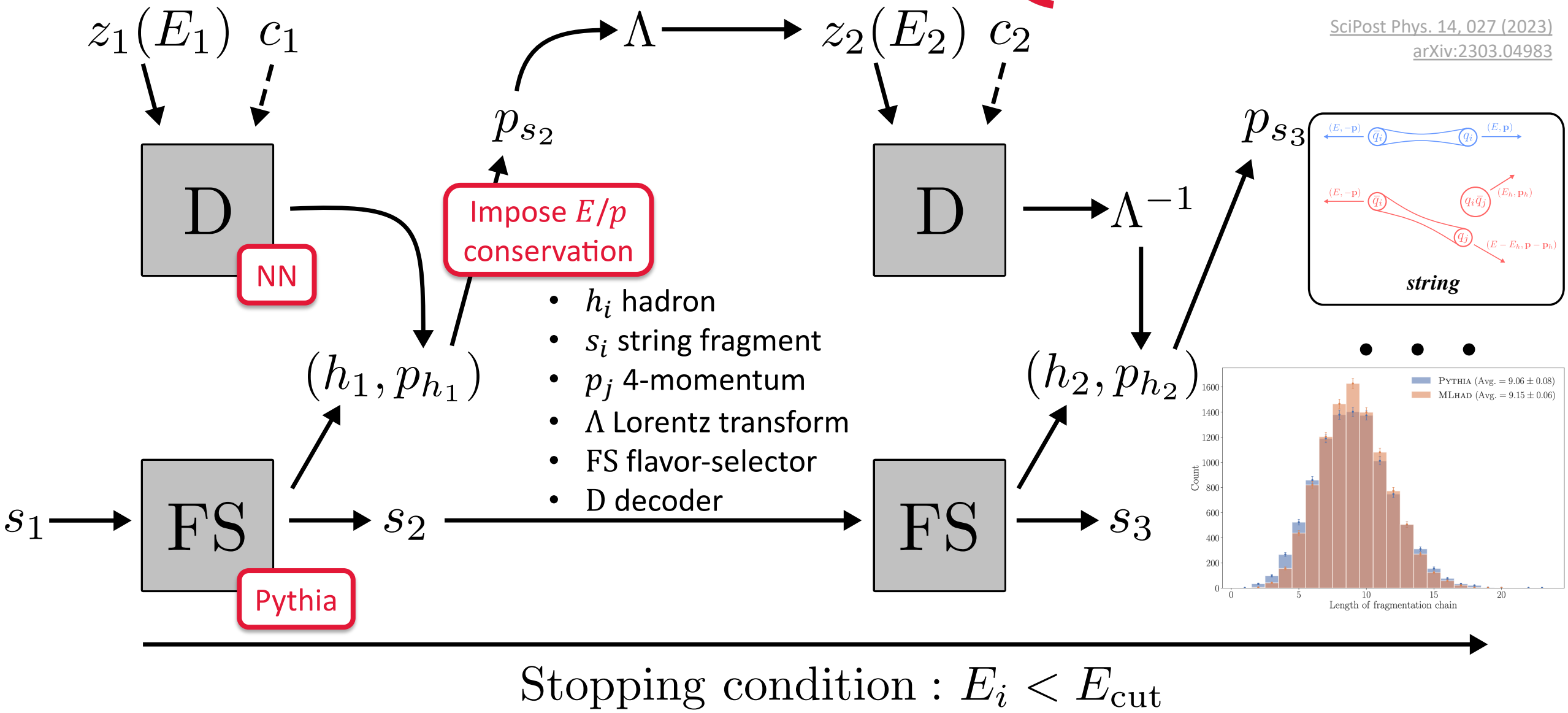
Train encoder and decoder to minimize $\mathcal{L}_{\mathrm{SW}} + \mathcal{L}_{\mathrm{rec}}$ using Pythia-simulated pion emissions

- $\boldsymbol{c}_i$ is a condition (quark energy)
- $\boldsymbol{x}_i$ is an observable (hadron p_T or p_z)
- $\boldsymbol{z}_i$ is a latent variable following a user-defined distribution
- $\mathcal{L}_{\mathrm{SW}}$ is the sliced-Wasserstein distance
- $\mathcal{L}_{\mathrm{rec}}$ is the reconstruction loss
- En/decoder is a neural network



cSWAE for hadronization

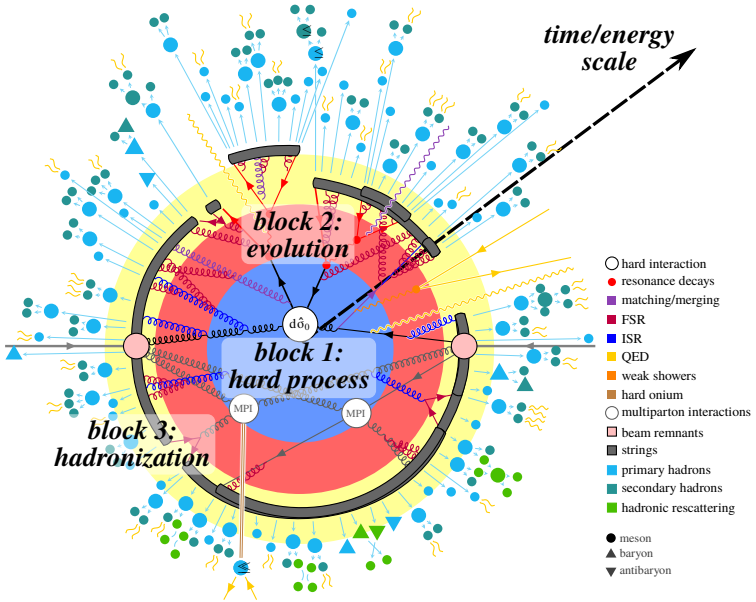
SciPost Phys. 14, 027 (2023)
arXiv:2303.04983

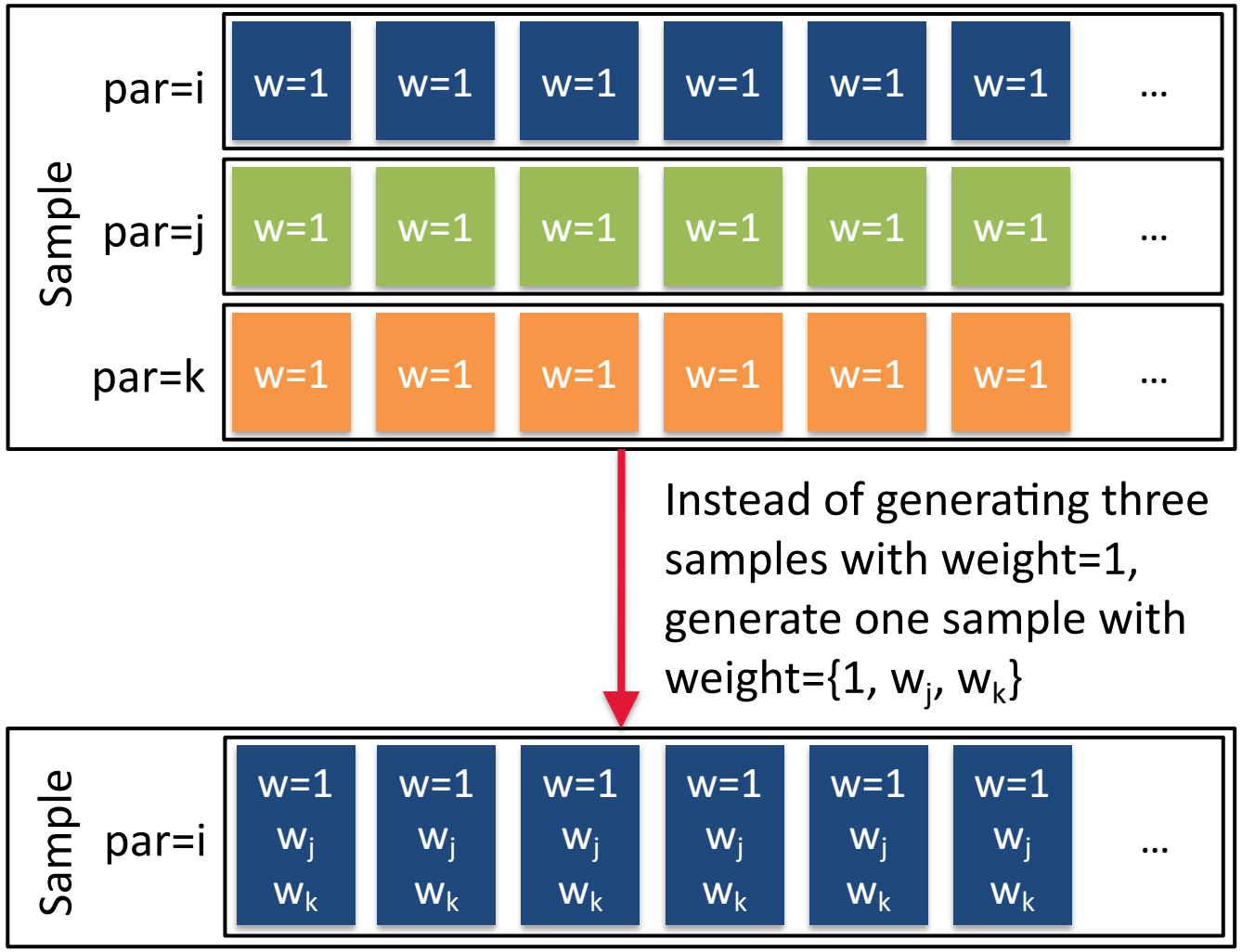


- h_i hadron
- s_i string fragment
- p_j 4-momentum
- Λ Lorentz transform
- FS flavor-selector
- D decoder

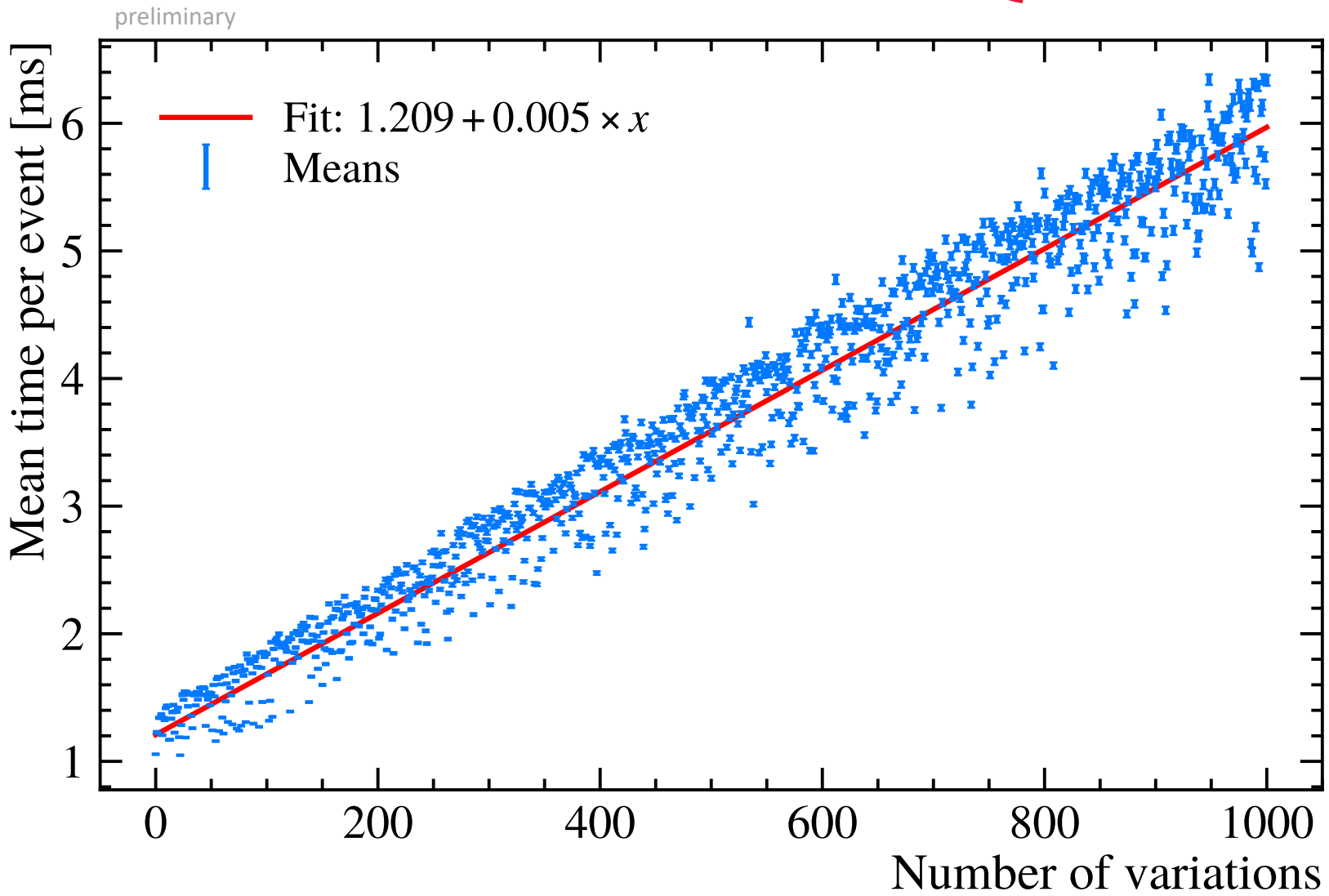
Reweighting Pythia events

- Event generation is time-consuming, so we want to **reweight** events **without regenerating**
- We calculate event weights for different **hadronization options** in a single Pythia event generation





Reweighting saves time

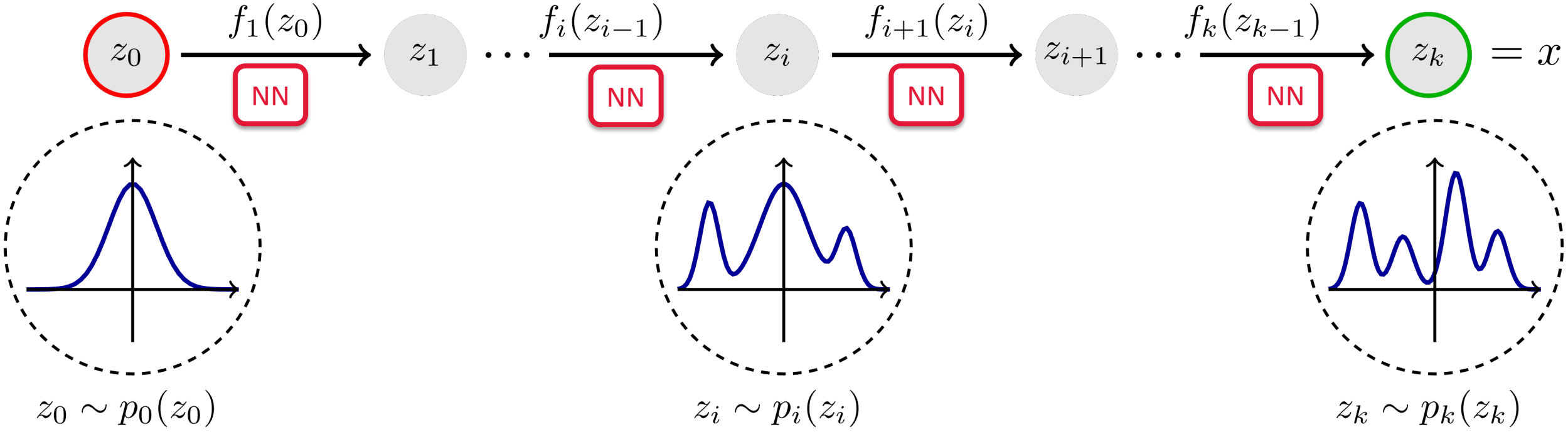


Marginal cost of ≈ **5 µs per variation** vs. ≈ 1,000 µs per unvaried event vs. ≈ 1,000,000 µs with detector sim.

Generated samples of 100,000 events with more and more r_b variations in Pythia

Normalizing flows

https://github.com/janosh/awesome-normalizing-flows



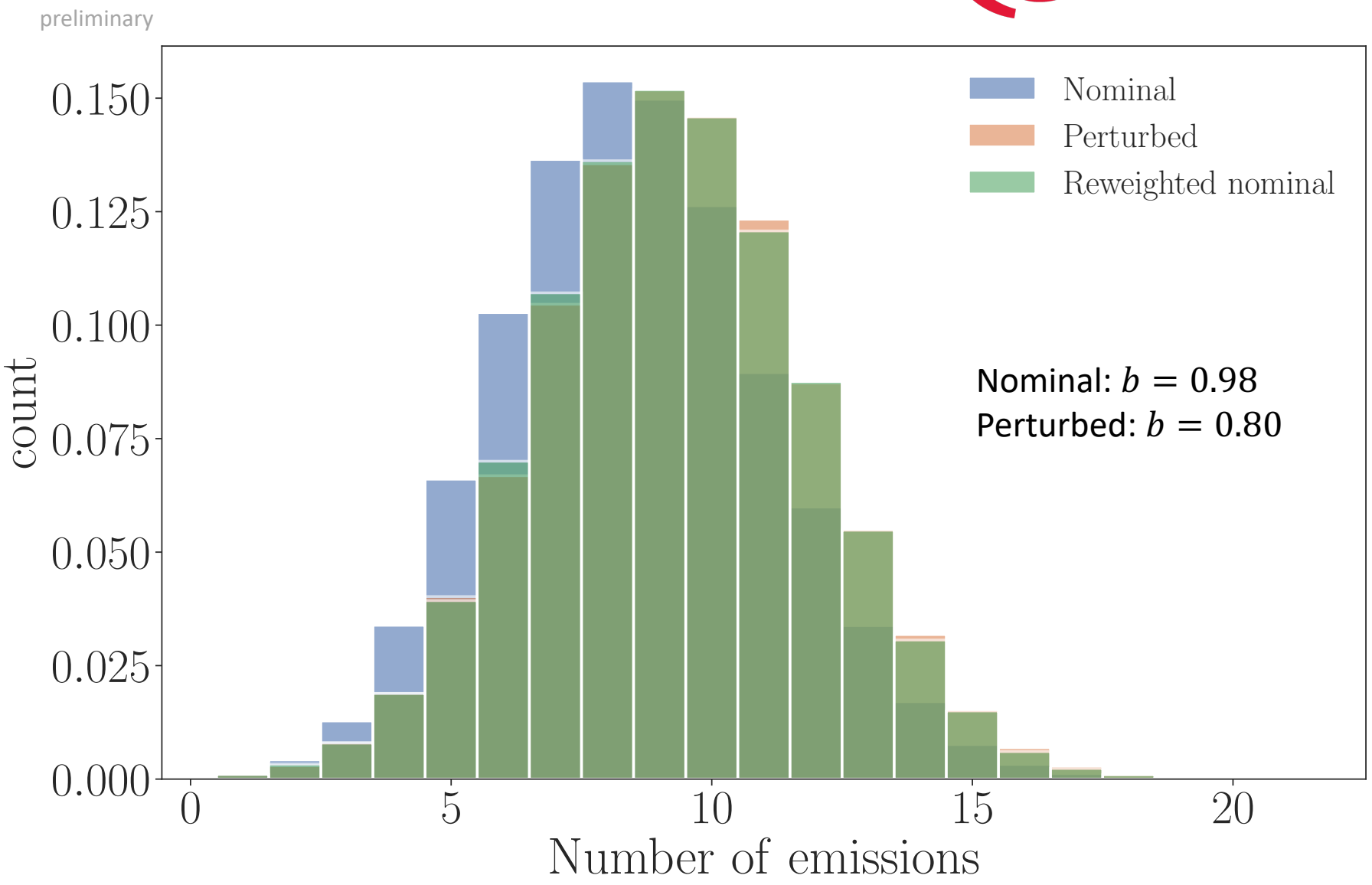
Start with Gaussian, want to transform into observable distribution

Each NN in the NF learns a **differentiable, invertible** function f_i, which together convert from $p_0(z_0)$ to $p_k(z_k)$

NFs learn **likelihood directly** without training separate Encoder and Decoder

NFs can be reweighted

b is a free parameter in the Lund function used in Pythia: [StringZ:bLund]



Train nominal NF, get likelihood

Train perturbed NF, get likelihood

Reweight nominal output using ratio of likelihoods

Next steps

- Finalize Pythia reweighting
  - Implement/validate more fragmentation parameters
  - Compare reweighting for more observables
  - Release as part of Pythia
- Finalize normalizing flows architecture
  - Train with/interpolate between more meson masses
  - Validate reweighting
- Ultimately, adapt architecture to train on experimental data

cSWAE publication:

SciPost Phys. 14, 027 (2023)

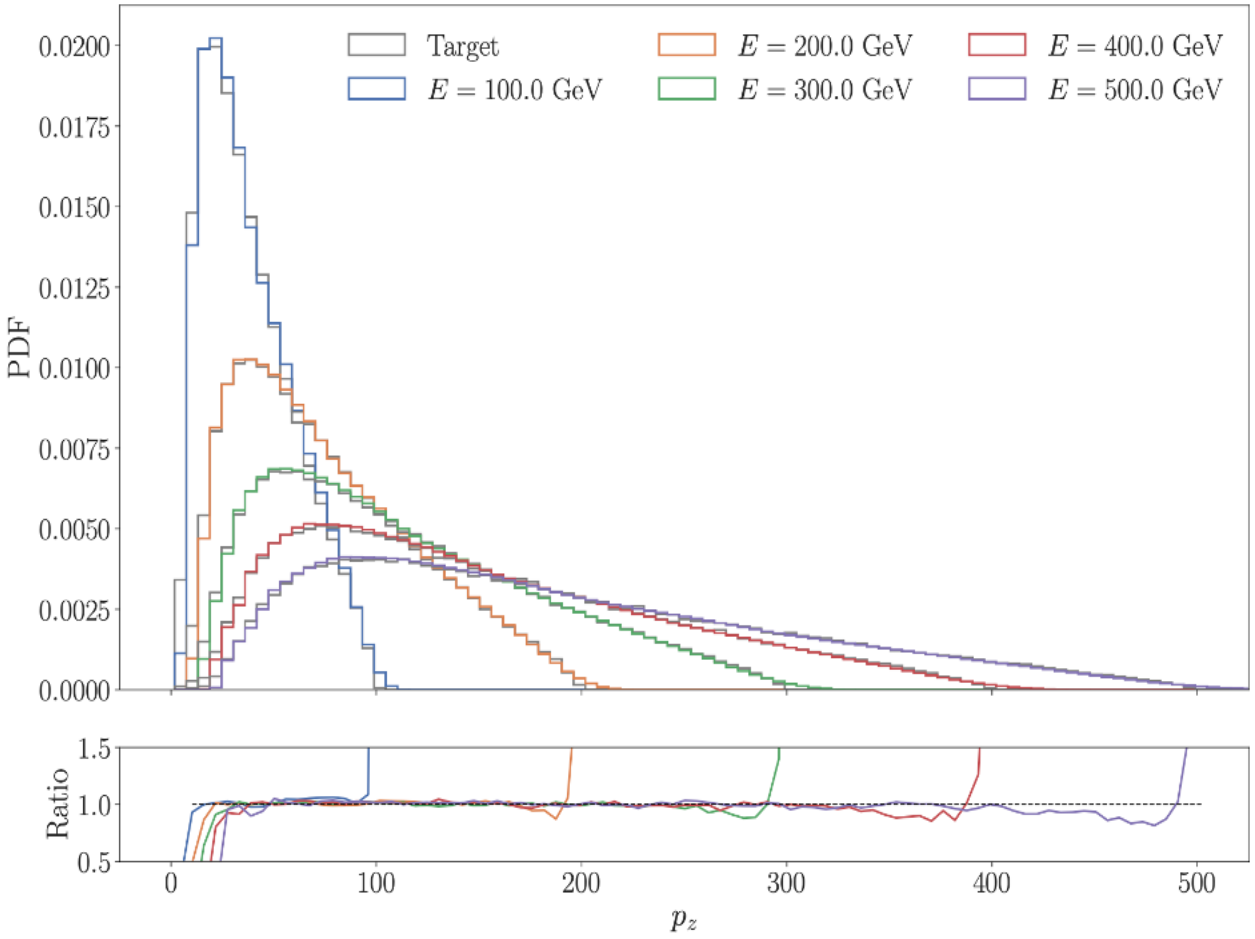
arXiv:2303.04983

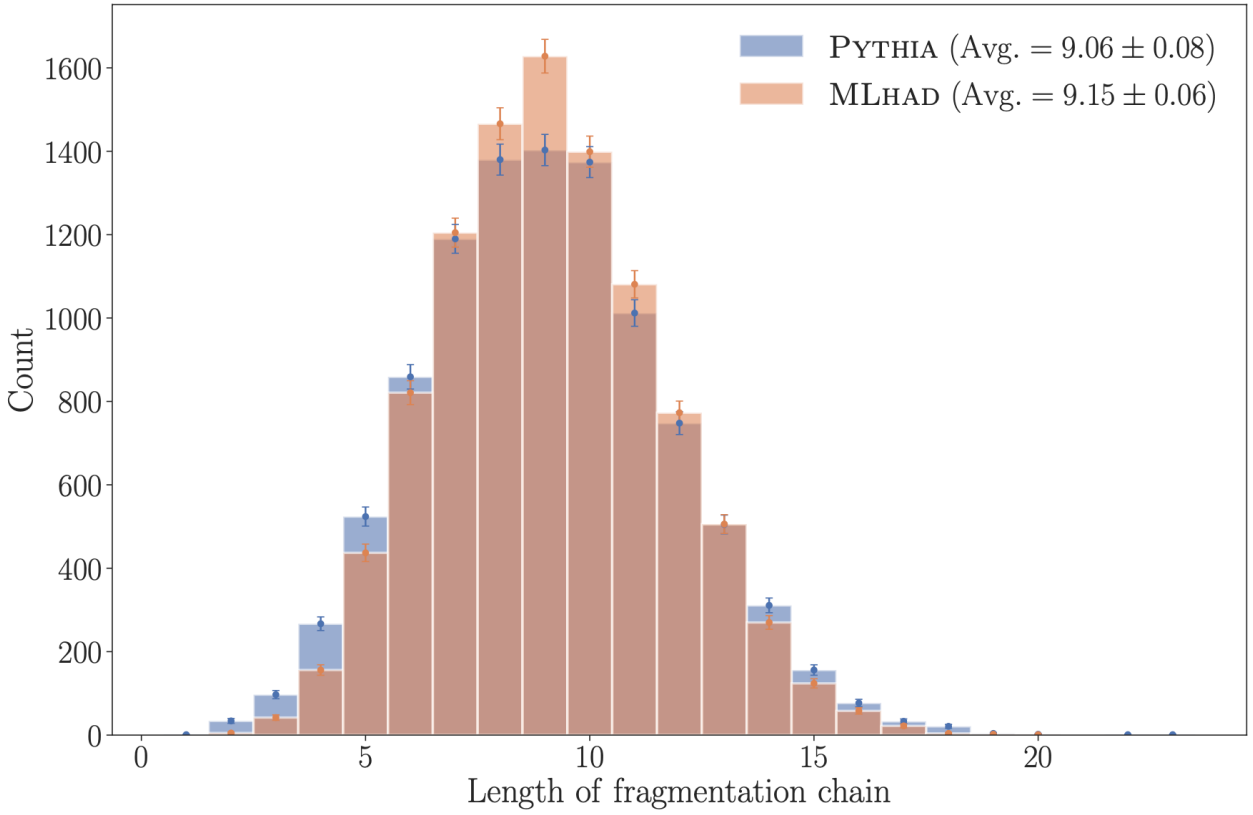
FIN

BACKUP

cSWAE replicates Pythia pions

SciPost Phys. 14, 027 (2023)
arXiv:2303.04983



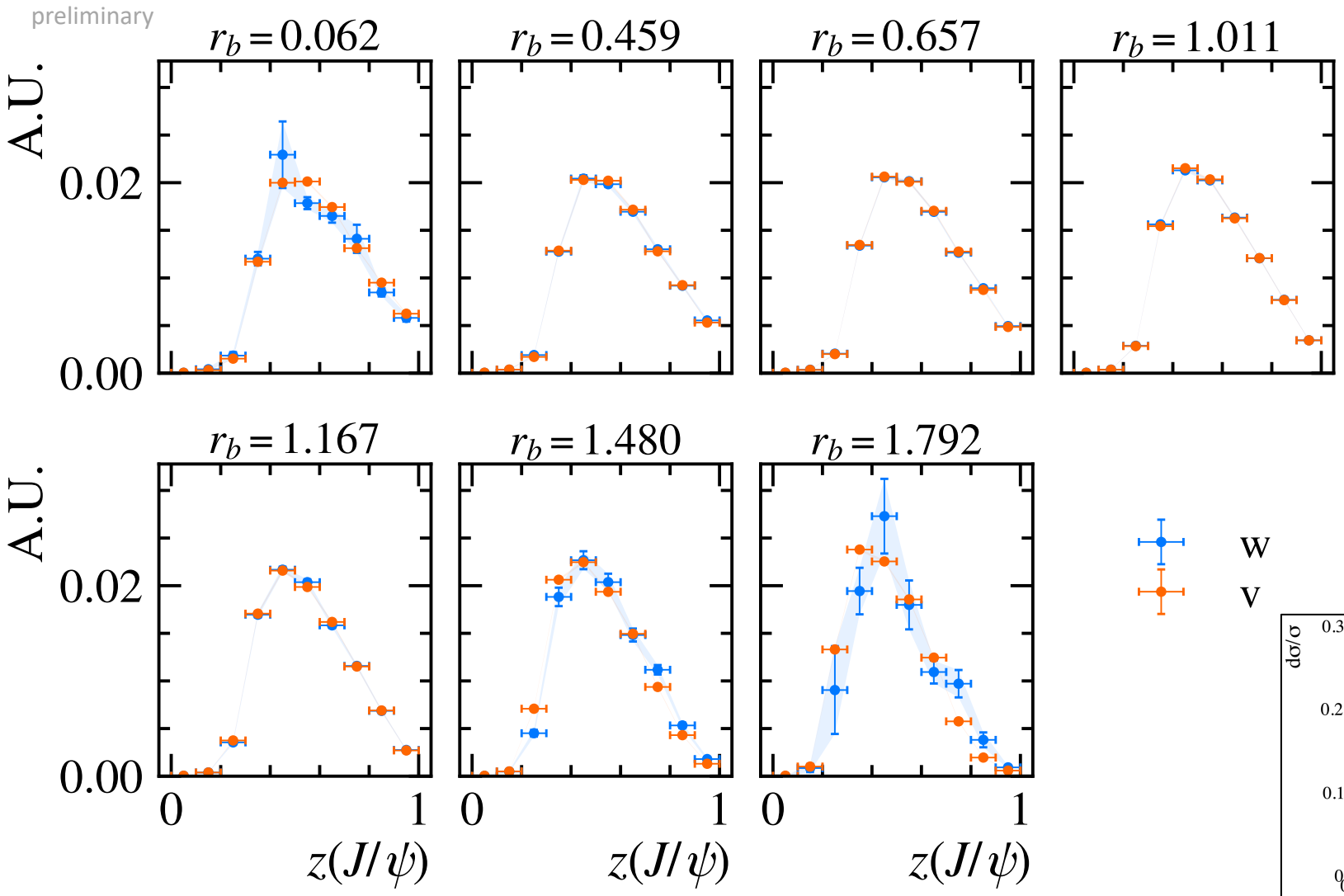


Reweighting vs. setting values

Can reweight r_b **without redoing generation** in Pythia

r_b is the Bowler modification to the Lund function: [StringZ:rFactB]

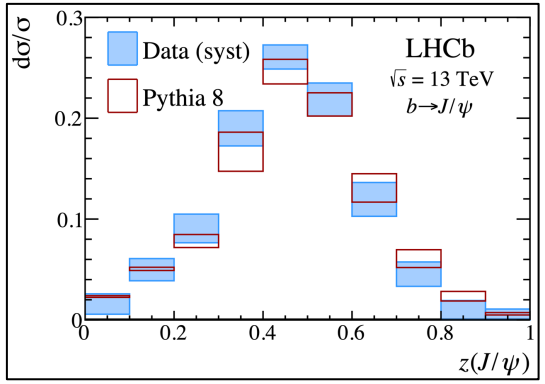
Blue sample and each orange sample correspond to 1,000,000 Pythia-simulated events



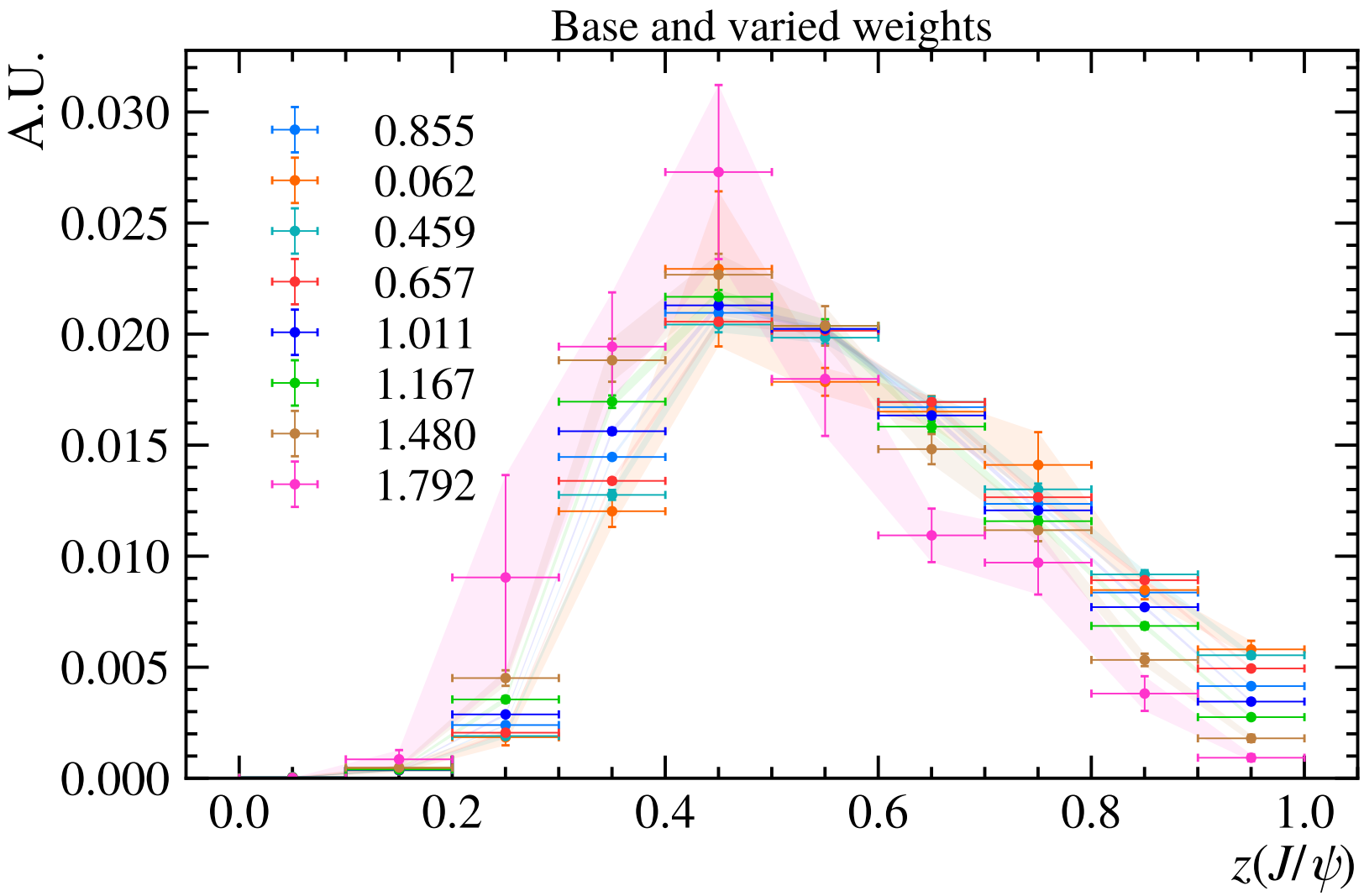
w is reweighted from $r_b = 0.855$
v has r_b set explicitly

$$z(J/\psi) = \frac{p_T(J/\psi)}{p_T(\text{jet})}$$

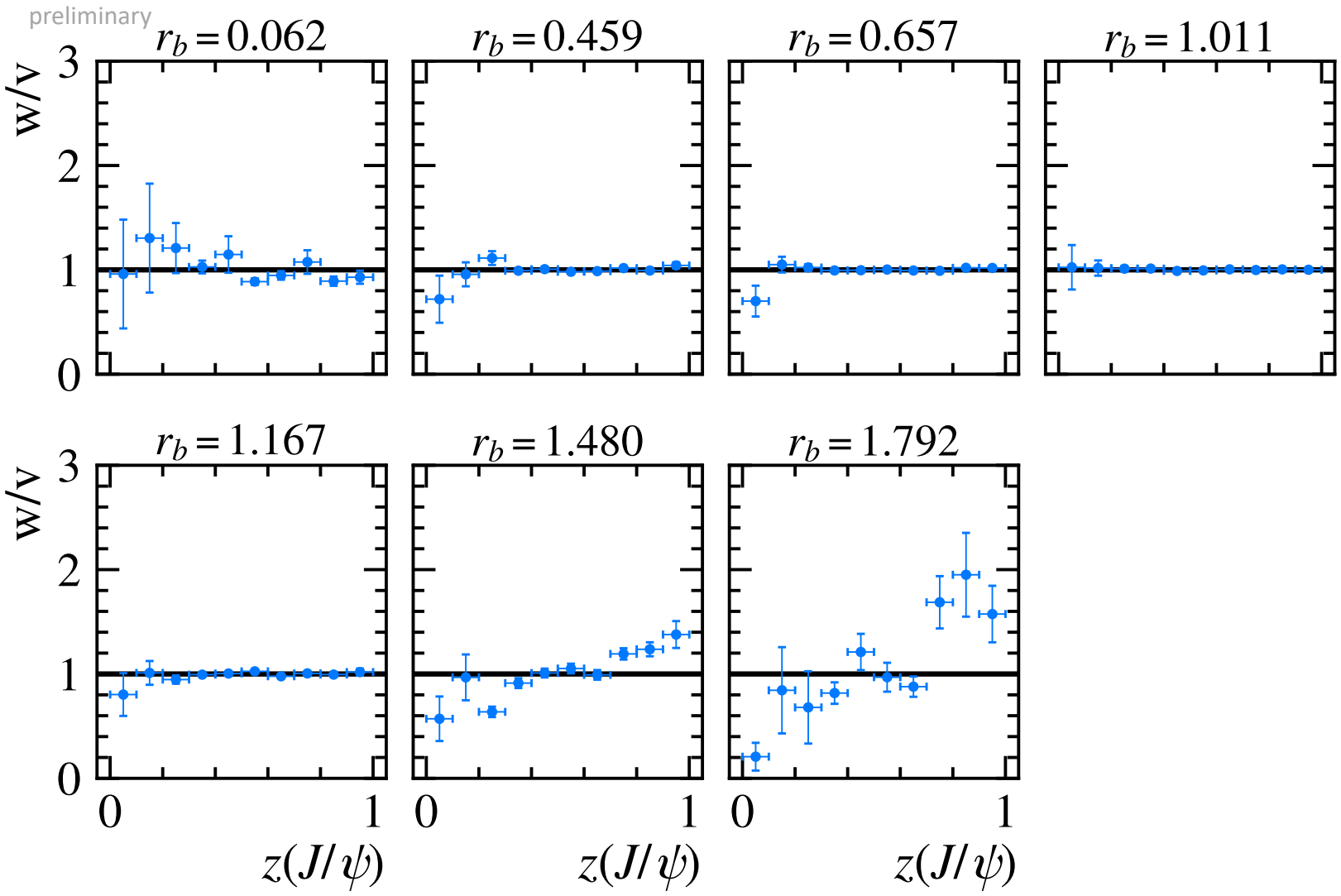
arXiv:1701.05116



Distributions



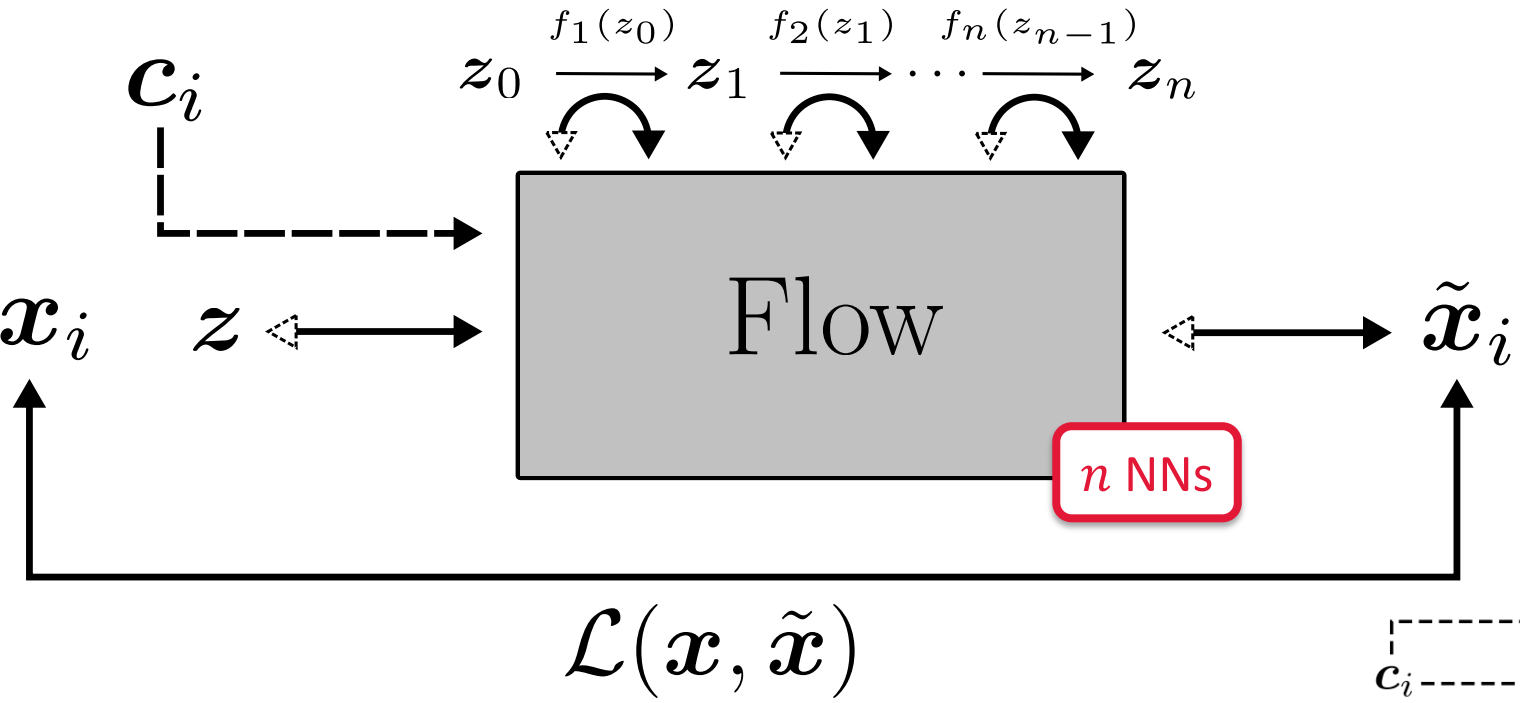
Ratios


preliminary
$r_b = 0.062$
$r_b = 0.459$
$r_b = 0.657$
$r_b = 1.011$
$r_b = 1.167$
$r_b = 1.480$
$r_b = 1.792$
w/v
0
1
2
3
$z(J/\psi)$

Normalizing flows architecture

preliminary

NFs



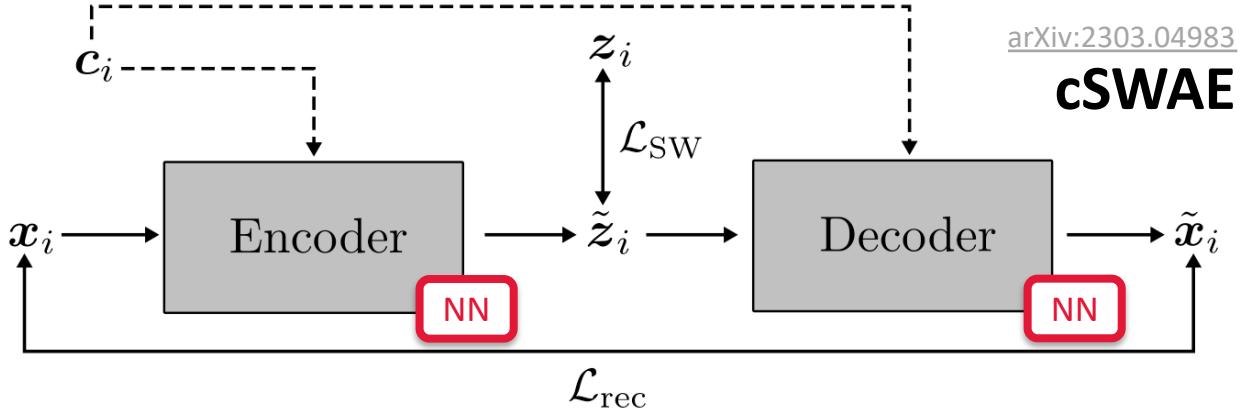
$$p_n(z_n) = p_0(z_0) \prod_{j=1}^{n} \left| \det \left(\frac{\partial f_j(z_{j-1}, c_i)}{\partial z_{j-1}} \right) \right|^{-1}$$

- $z_n = \tilde{x}_i$
- $z_j \sim p_j(z_j)$
- $p_0 = \mathcal{N}$ (Gaussian distribution)
- f_j a differentiable, invertible function (learned by a neural network)
- n neural networks in the normalizing flow

NFs learn **likelihood directly** without training separate Encoder and Decoder

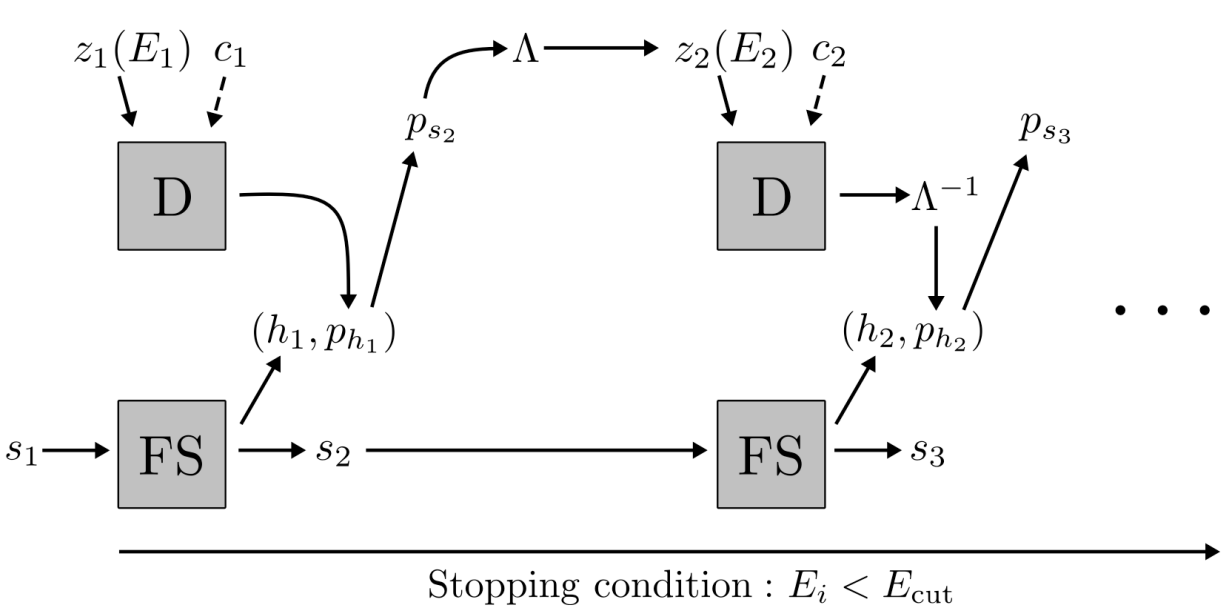
arXiv:2303.04983

cSWAE

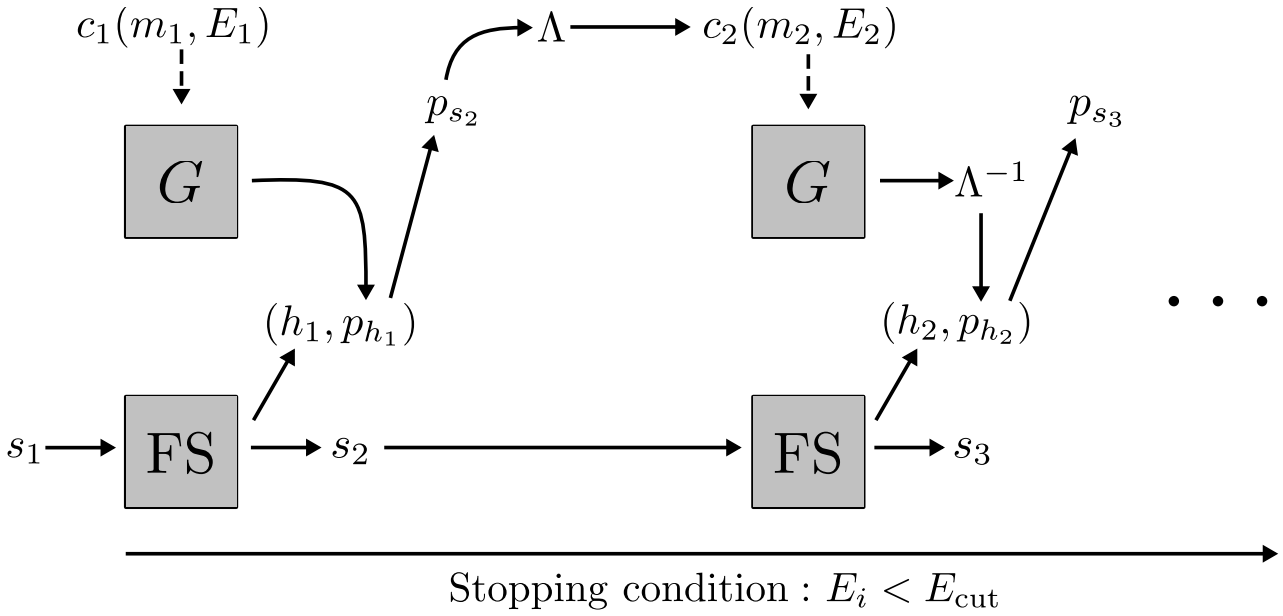


NFs for hadronization

cSWAE arXiv:2303.04983

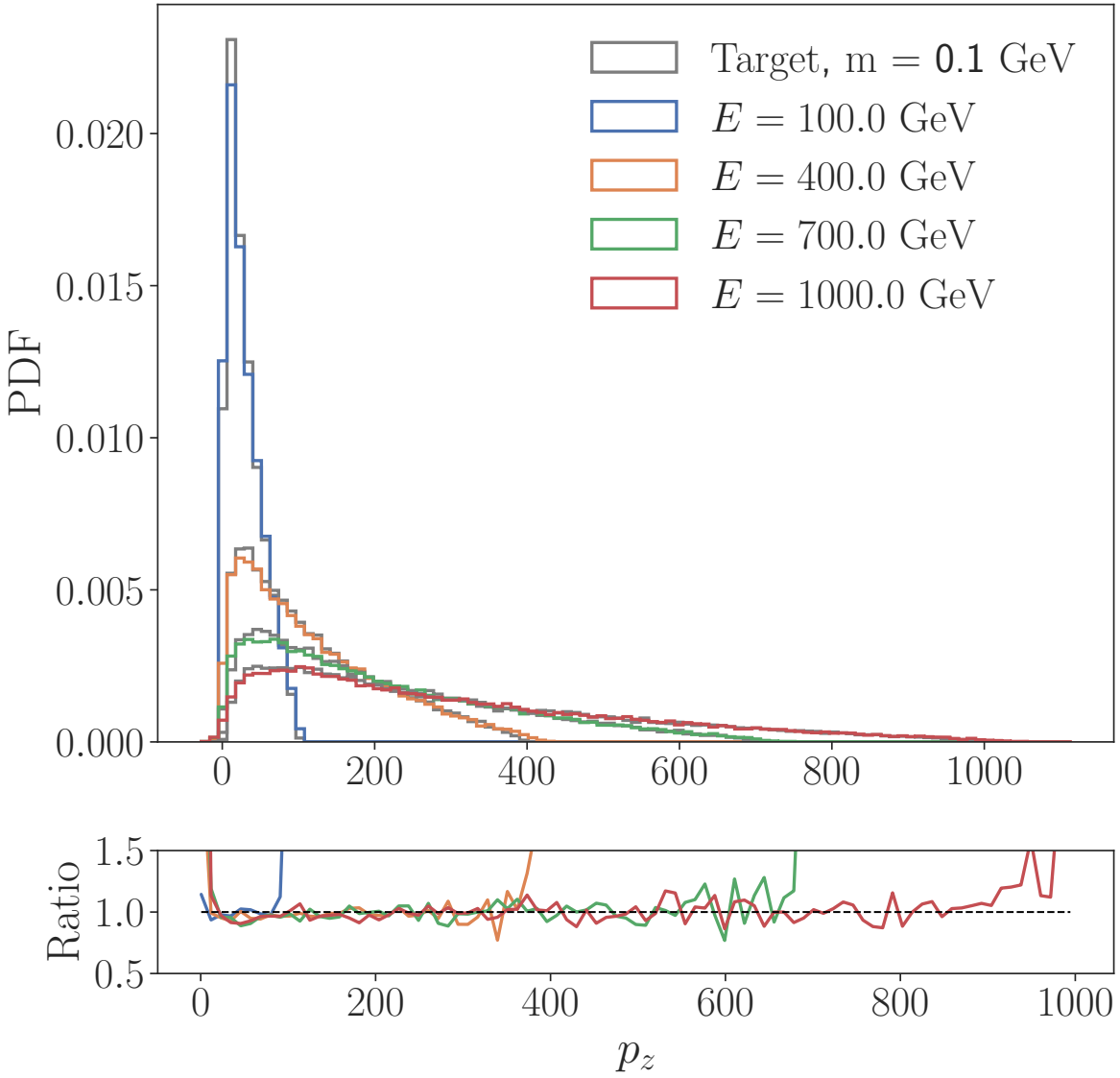


NFs preliminary

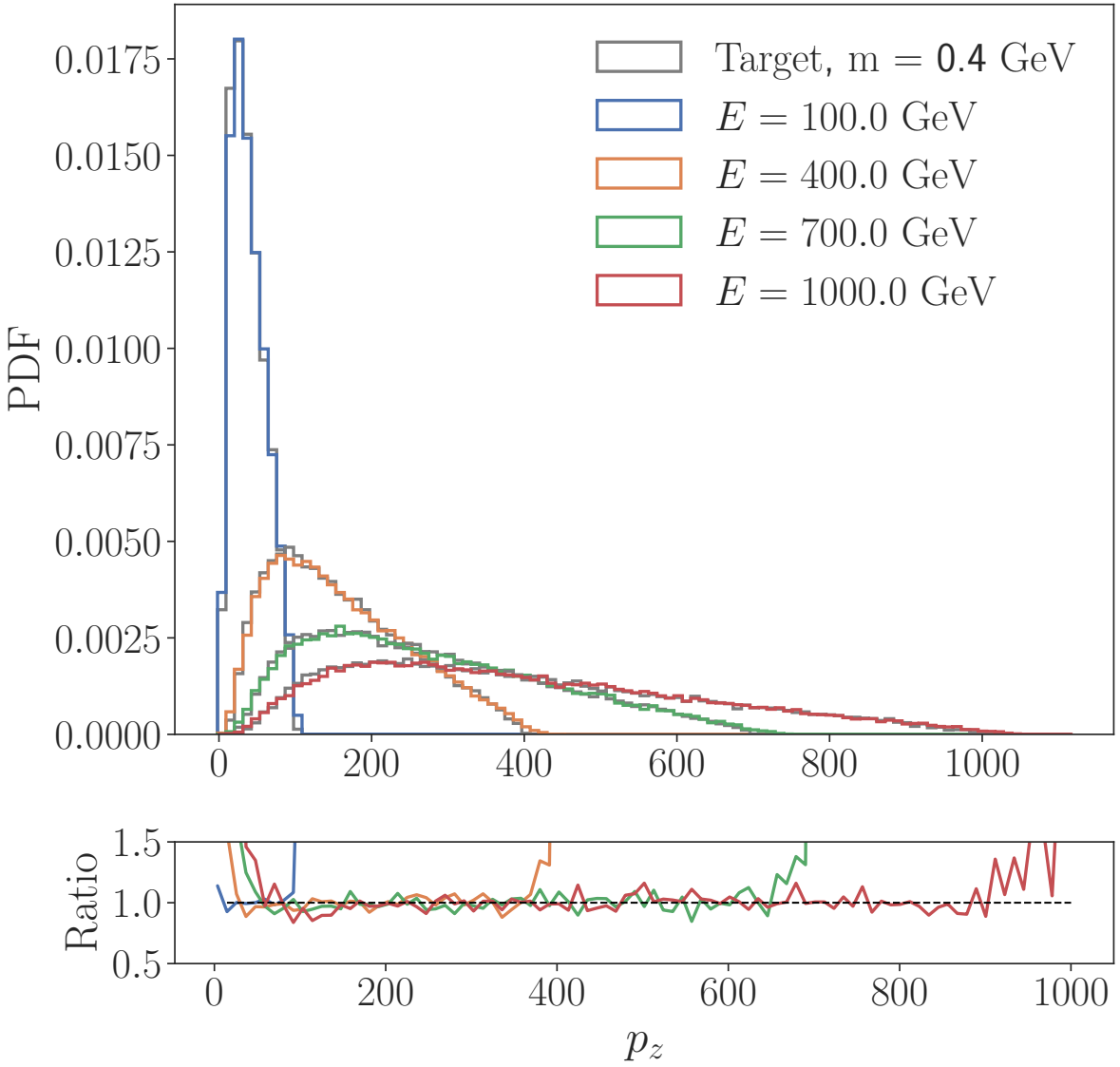


Normalizing flows replicate Pythia

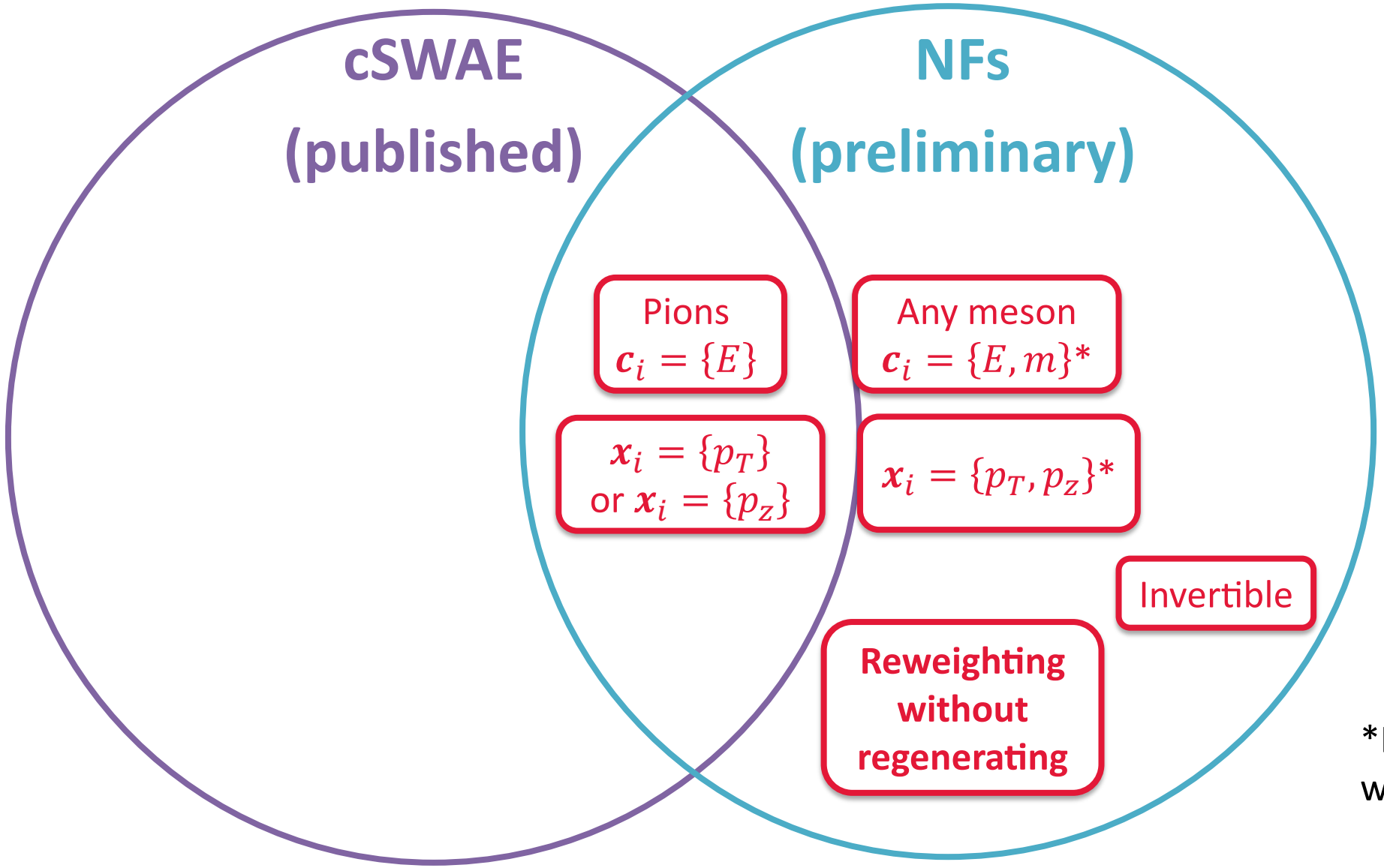
preliminary



preliminary



cSWAE vs. normalizing flows



*Have also done with cSWAE