Normalizing flows and uncertainty quantification in hadronization

Phenomenology 2023 Symposium

Based on SciPost Phys. 14, 027 (2023) and 2306.XXXXX

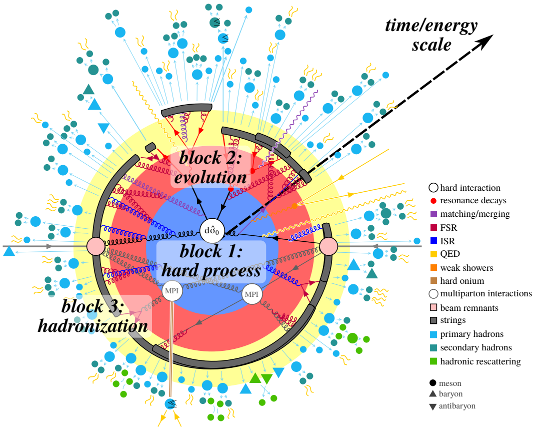
Ahmed Youssef

P. Ilten, T. Menzo, S. Mrenna, M. Szewc, M.K. Wilkinson, and J. Zupan

May 8, 2023

University of Cincinnati Department of Physics

Motivation



Hard process: initial high-energy interaction
Evolution: parton shower
} *perturbative*

Hadronization: combine quarks and gluons } non-perturbative

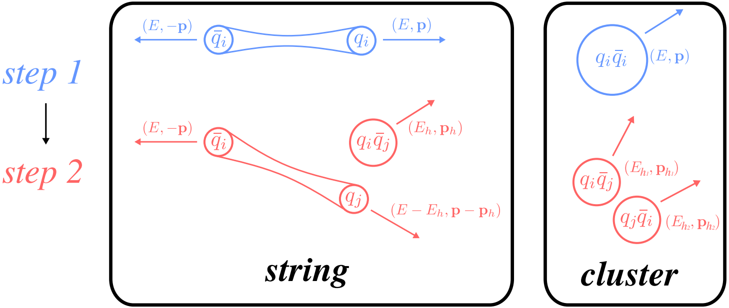
First step: Create a Machine Learning (ML) Architecture that is able to reproduce the simplified Lund String Model

Goal: **Train on experimental data** and replace or complement the Hadronization model in PYTHIA

MLHAD

Hadronization Models

Two primary hadronization models are used



String model:

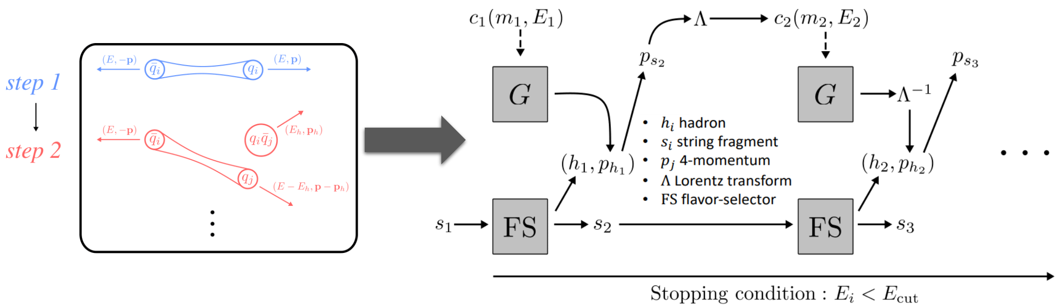
Iteratively split parton connected by QCD color strings with linear potential

Cluster model:

pre-confine partons into proto-clusters, then split by two-body decays

$\Longrightarrow$ **Lund-String model is used in PYTHIA**

MLHAD Pipeline

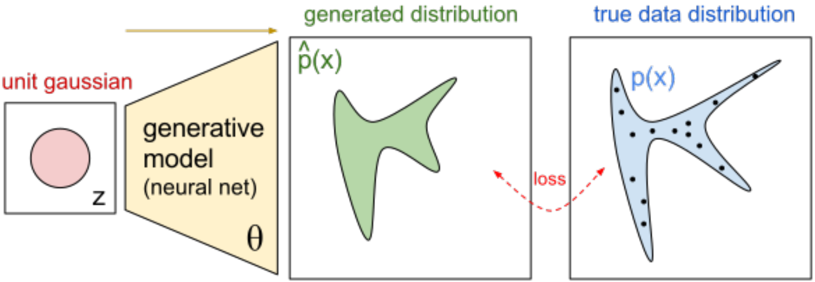


We need a generative model:

$\Rightarrow$ Sample hadron kinematics: train on $\{\mathbf{p_z}, \mathbf{p_T}\}$

$\Rightarrow$ Emission of different Mesons: Condition on mass (**m**) and energy (**E**)

Generative Models



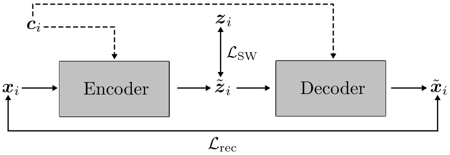
Source: generative models

$\Rightarrow$ Task: Learn the probability distribution $p(x)$ of the data

Which generative model should we choose?

Is it able to learn complex distributions?	Do we have access to the exact probability distribution?

Conditional Sliced Wasserstein (SW) Autoencoder



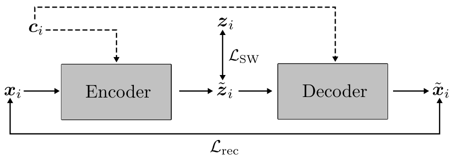
(Architecture used in SciPost Phys. 14, 027 (2023))

- SW distance enables learning any sampleable latent distribution
  $\Rightarrow$ **Can learn complex distributions**
- Decoder ”just” generates samples
  $\Rightarrow$ **No access to the probability distribution**

For simplicity, the previous MLHAD architecture emits pions only

Generative Models

Conditional Sliced Wasserstein (SW) Autoencoder

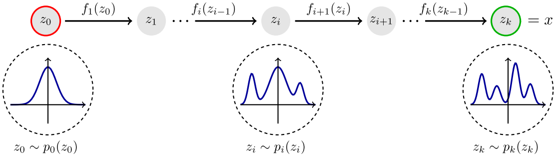


(Architecture used in SciPost Phys. 14, 027 (2023))

- SW distance enables learning any sampleable latent distribution
  $\Rightarrow$ **Can learn complex distributions**
- Decoder "just" generates samples
  $\Rightarrow$ **No access to the probability distribution**

For simplicity, the previous MLHAD architecture emits pions only

Normalizing Flow (NF)



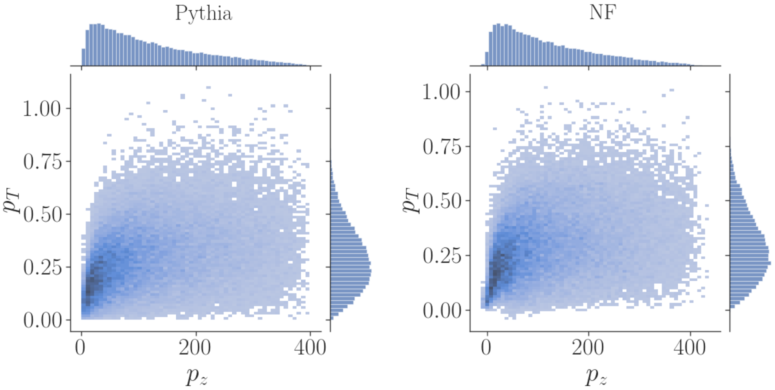
(Figure taken from github/janos/awesome-normalizing-flows)

- Chain of invertible transformation *f*
  $\Rightarrow$ **Can learn complex distributions**
- Distribution is obtained by change of variables
  $\Rightarrow$ **Access to the exact probability distribution**

Updated MLHAD architecture can emit different mesons

Training Results cNF

*Preliminary



NFs, conditioned on different masses and energies, learn the correlation between p_z and p_T

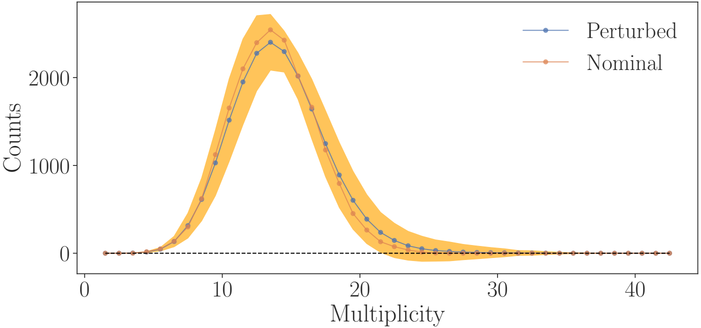
Uncertainty Quantification

- Correlated uncertainties
- Statistical and training uncertainties
- Model uncertainties (not in this talk)

youssead@ucmail.uc.edu

Correlated Uncertainties

NFs can be used to capture correlated uncertainties



Generate multiple datasets with varied Pythia parameters to mimic correlated uncertainties

Error bands correspond to varying bLund parameters

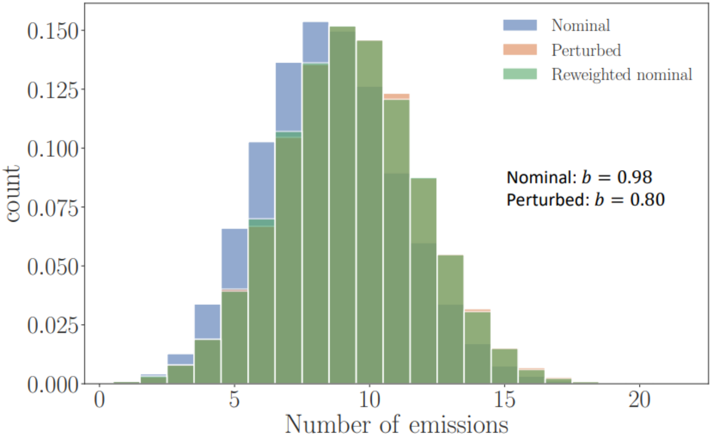
$\Rightarrow$ **We can reweight between error bands with the weight:**

$$w = \prod \frac{p^{(i)}_{Nom}(z)}{p^{(i)}_{pert}(z)} ,$$

Reweighting with NFs

*Preliminary

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



Train nominal NF, get likelihood

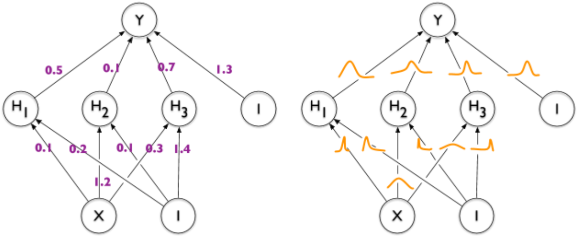
Train perturbed NF, get likelihood

Reweight nominal output using ratio of likelihoods

We can obtain multiple datasets without resampling using the correlated uncertainties

$\Rightarrow$ **Much less time expensive than fully simulating with new parameters**

Bayesian Neural Networks



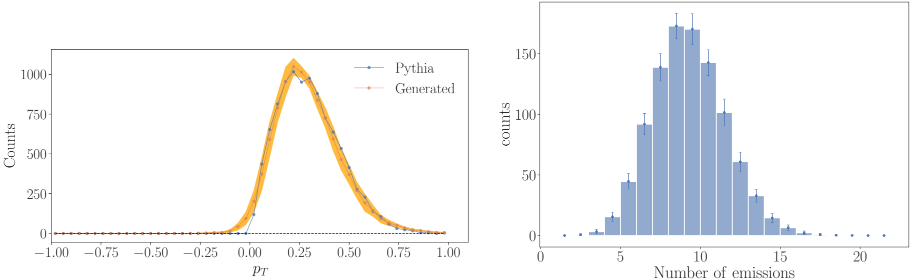
(Image source: The very Basics of Bayesian Neural Networks)

- Quantifies statistical and training uncertainty
- Modify network such:
  - $\longrightarrow$ Weights are sampled from a distribution
  - $\longrightarrow$ Additional loss function for weight distribution

youssead@ucmail.uc.edu

Bayesian NF Results

*Preliminary



Now we get errors on the kinematic distributions

$\Rightarrow$ **Can be used to estimate the statistical and training errors on observables**

Conclusion and Outlook

- First MLHAD pipeline based on cSWAE was published in SciPost Phys. 14, 027 (2023)
- NFs overcome the limitations of cSWAE - can emit in principle any meson and have access to pdf
- NFs allow us to reweight events and capture uncertainties

Work in progress

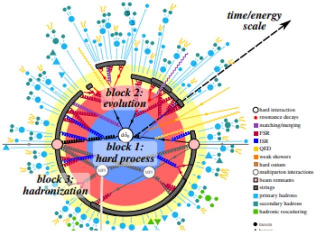
- Finalize normalizing flows architecture (include model uncertainty)
- PYTHIA reweighting (Release as part of Pythia)
- Flavor Selector
- Performing training on physically accessible observables to train MLHAD on **experimental data**

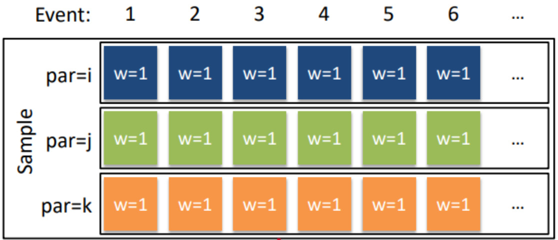
Back up

youssead@ucmail.uc.edu

Reweighting

- 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

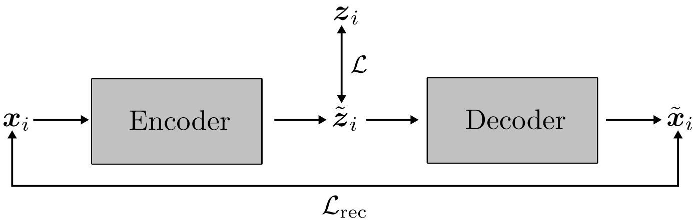




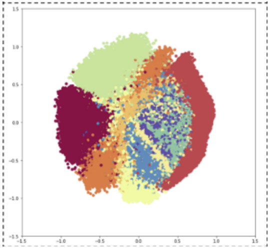
Instead of generating three samples with weight=1, generate one sample with weight={1, w_j, w_k}

VAE

- VAE is a commonly used generative model:
  - $\longrightarrow$ Not flexible with the latent representation
  - $\longrightarrow$ kl-divergence limits latent distribution to a simple analytical form (e.g. Gaussian)

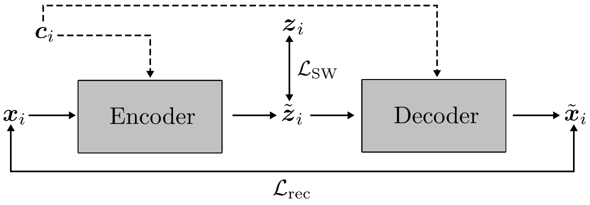


(a) Vanilla VAE
(arXiv: 1312.6114)

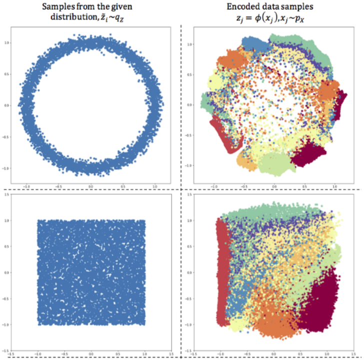


(b) VAE latent space

cSWAE



(a) cSWAE architecture



(b) SWAE latent space
(arXiv: 1804.01947)

Total loss: $\mathcal{L} = \mathcal{L}_{rec} + \mathcal{L}_{SWD}$

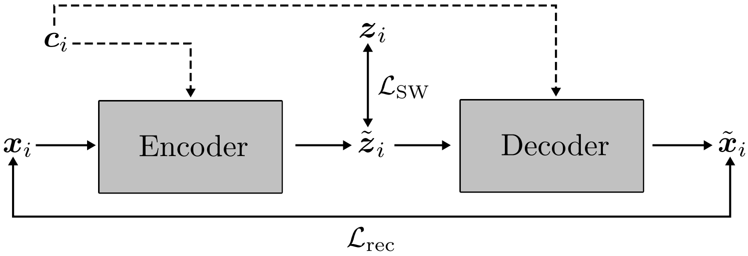
cSWAE Training-process

- conditioned on initial string energy $E_i \rightarrow c_i = (\bar{c}_i, 1 - \bar{c}_i)$:

$$E_i = E_{min}\bar{c}_i + E_{max}(1 - \bar{c}_i) \quad \Rightarrow \quad \bar{c}_i = \frac{E_{max} - E_i}{E_{max} - E_{min}}$$

- Encoder ϕ:
  - Input data $\mathbf{x_i}$ is a $N_e = 100$ dimensional vector, where $\mathbf{x_i} \in [p_{z,k}^{(i)}, p_{T,k}^{(i)}]$
  - p_z and p_T are uncorrelated and treated seperately
  - Takes as input x_i and c_i; returns the latent space vector $\bar{z}_i = \phi(x_i, c_i)$
- Decoder ψ:
  - Takes as input $\bar{z}_i$ and returns $\bar{x}_i = \psi(\phi(x_i, c_i))$
- Limit the training on light quark flavors and only pions as final state hadrons

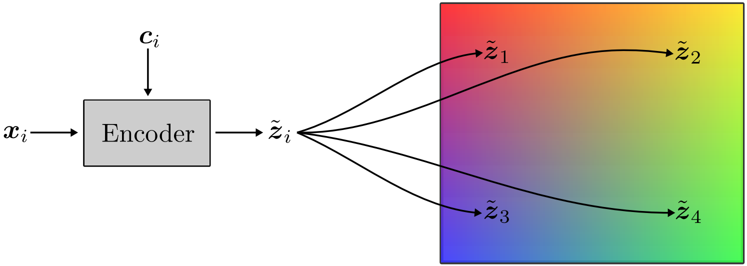
cSWAE architecture



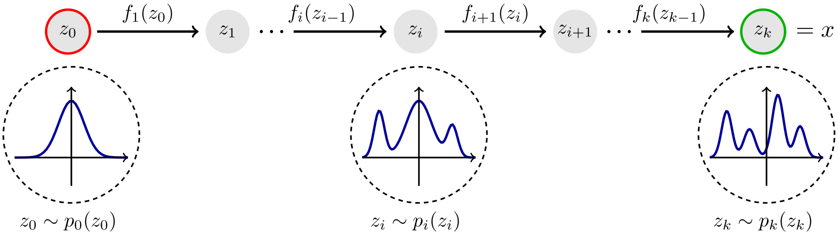
$$\mathcal{L}_{\mathrm{rec}} = \frac{1}{N_{\mathrm{tr}}} \sum_{i=1}^{N_{\mathrm{tr}}} \left[\frac{1}{Q} d_2^2(\boldsymbol{x}_i, \psi(\phi(\boldsymbol{x}_i, \boldsymbol{c}_i))) + d_1(\boldsymbol{x}_i, \psi(\phi(\boldsymbol{x}_i, \boldsymbol{c}_i))) \right],$$

$$\mathcal{L}_{\mathrm{SW}} = \frac{\lambda}{L N_{\mathrm{tr}}} \sum_{\ell=1}^{L} \sum_{i=1}^{N_{\mathrm{tr}}} d_{\mathrm{SW}}(\boldsymbol{\theta}_\ell \cdot \boldsymbol{z}_{[i]_\ell}, \boldsymbol{\theta}_\ell \cdot \phi(\boldsymbol{x}_{[i]_\ell}, \boldsymbol{c}_i)),$$

cSWAE architecture



NFs



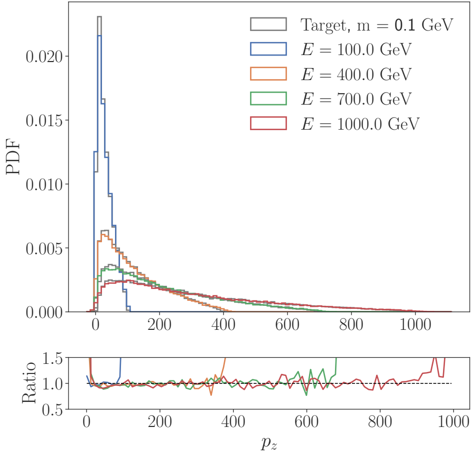
- z_0 is a random vector sampled from a simple distribution (usually a Gaussian) $z_0 \sim p_0(z_0)$
- f is an invertable NN
- Calculate x by change of variables:

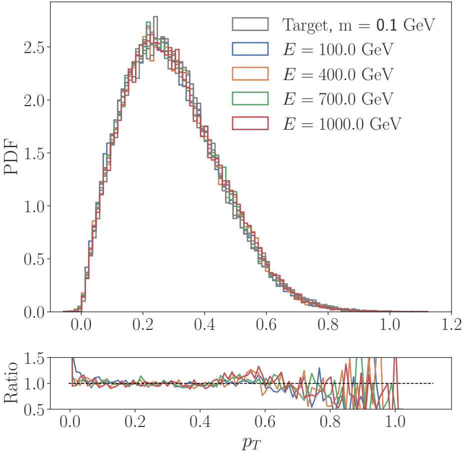
$$p_k(x) = p_0(z_0) \prod_{i=1}^{K} \left| \det \left(\frac{\partial f_i(z_{i-1})}{\partial z_{i-1}} \right) \right|^{-1}$$

youssead@ucmail.uc.edu

Training Results cNF

*Preliminary





youssead@ucmail.uc.edu

Hadron Multiplicity vs String Energy (cNF)

* Preliminary