End-to-end event simulation with Flow Matching and generator Oversampling

Conference on Computing in High Energy Physics 2024

Filippo Cattafesta, F.Vaselli, P.Asenov, A.Rizzi

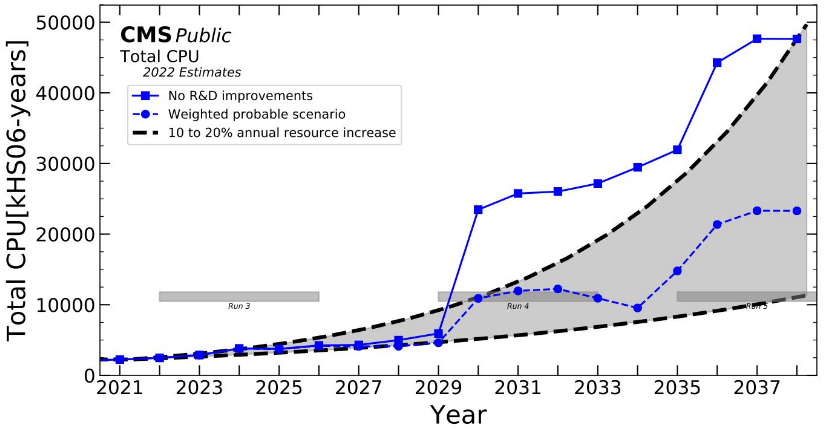
SCUOLA NORMALE SUPERIORE

Istituto Nazionale di Fisica Nucleare

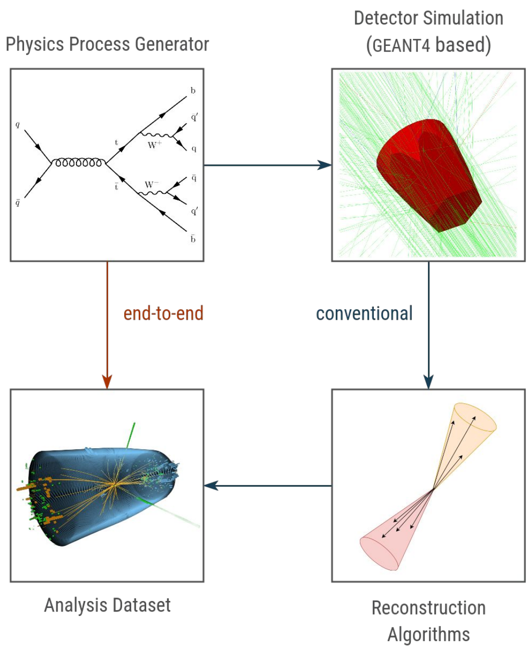
Kraków, 22 Oct. 2024

Challenges of Event Simulation for HEP

- Event simulation employs a large fraction of the CPU budget for LHC experiments
  - Billions of events needed for the analysis
- High Luminosity LHC challenge
  - Larger number of events and more granular detector will make the simulation even more expensive



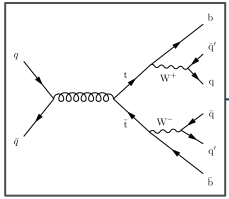
End-to-End Simulation

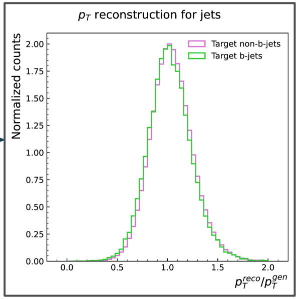


- Faster simulation is needed
  - Maintaining high accuracy (within typical data/sim agreement)
  - Not analysis/process specific
- End-to-end event simulation
  - Generator output as starting point
  - Direct production of high-level analysis objects (jets, muons, etc.)

Generative Models for Faster Simulation

Draw a realistic picture of Kraków during October



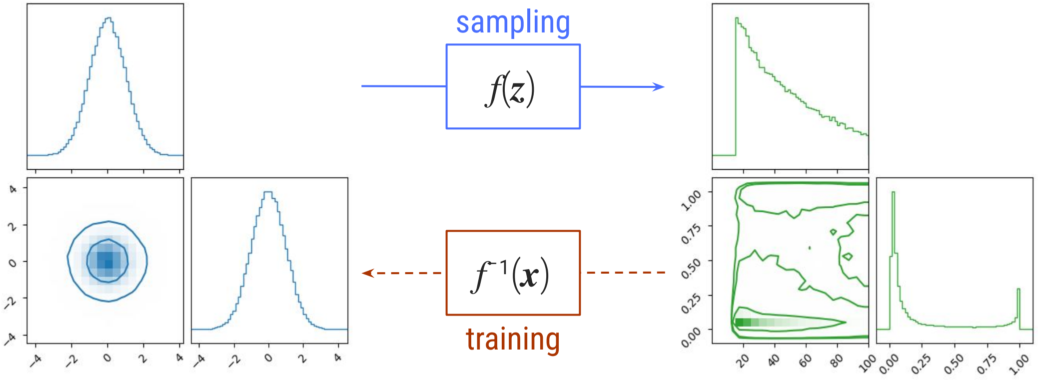


- Generative Models are well-suited for end-to-end simulation
  - Ability to learn target probability distribution conditioned on physical information

$$p(\mathrm{Reco}|\,Gen)$$

  - Fast inference (GPU)
  - Constantly evolving
- Key requirements for HEP
  - Preservation of statistical properties of the distribution

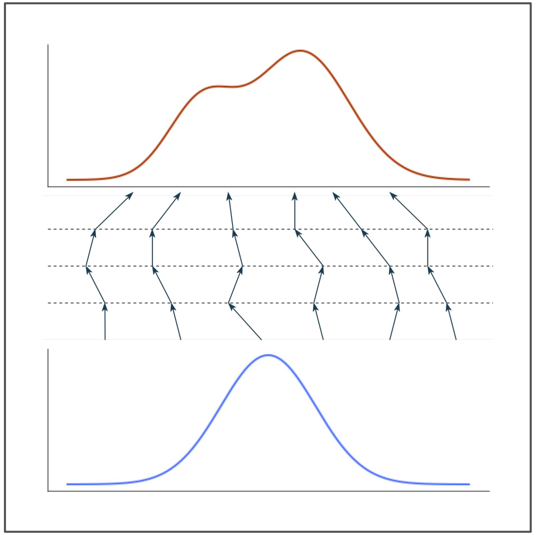
Normalizing Flows: Key Concepts



- The task is to sample from an unknown pdf
    - Invertible transformation (*flow*) is applied to Gaussian noise
    - The inverse transformation is learned in the training process
- The flow can be a Neural Network
- The flow is defined by the push-forward equation

$$\begin{cases} x = f(z) \\ p_x(x) = p_z(z) \det \left| \frac{dz}{dx} \right| \end{cases}$$

Discrete Flows



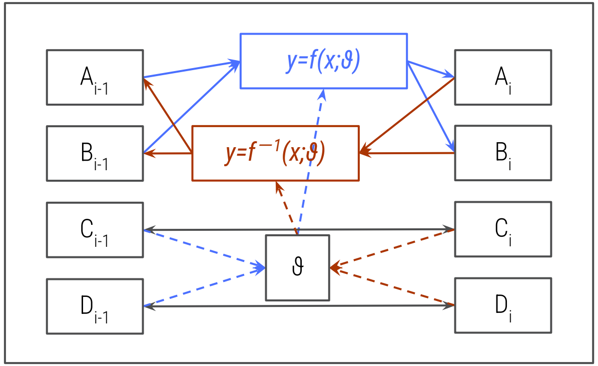
Adapted from https://ehoogeboom.github.io/post/en_flows/

$$f = f_K \circ f_{K-1} \circ \ldots \circ f_1$$

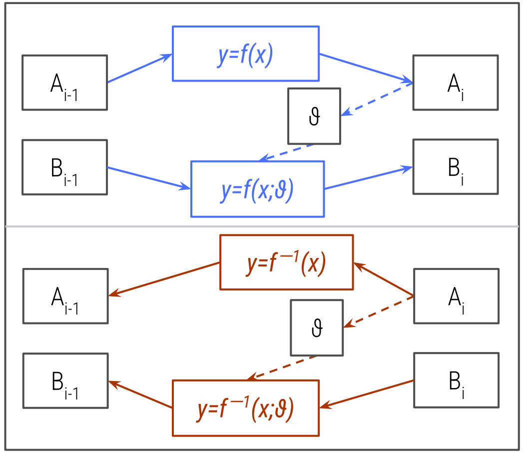
https://arxiv.org/abs/1912.02762

- The flow must be easily invertible
  - Analytic inverse
  - Tractable Jacobian
- Composition of a finite number of simple transformations
  - Affine transforms or splines
  - Variables transformed to have a (block) triangular Jacobian (*autoregressive* or *coupling* architectures)

Coupling

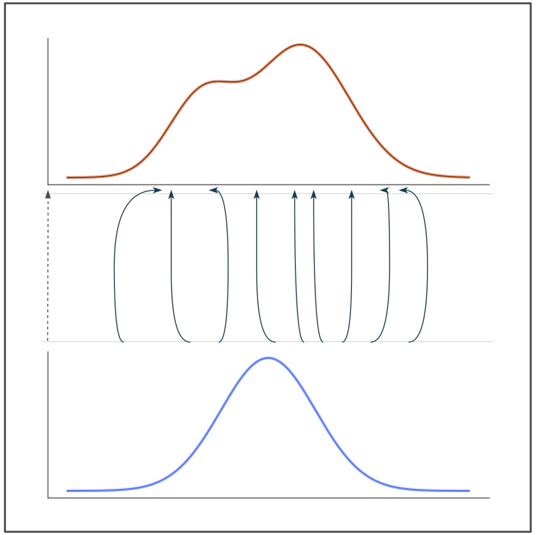


Autoregressive



Forward/Backward

Continuous Flows



Adapted from https://ehoogeboom.github.io/post/en_flows/

- **The flow is defined by a continuous parameter**
  - Time-dependent flow satisfying the following ordinary differential equation (ODE)

$$\begin{cases} \frac{d}{dt} f_t(z) = v_t(f_t(z)) \\ f_0(z) = z \end{cases}$$

  - The vector field v_t is modeled with a neural network
  - Integration on path during inference

$$x = z + \int_0^1 v_t(z)\, dt$$

- **Exact Density Estimation**
  - Invertibility maintained at each point along the path
- **Training challenges**
  - ODE numerical solutions
  - v_t modelling

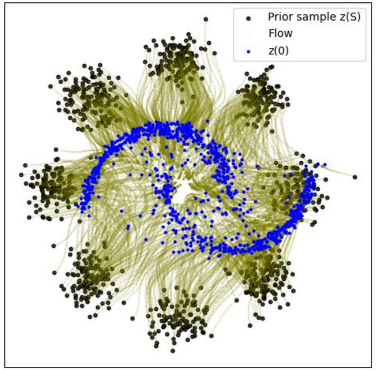
Conditional Flow Matching

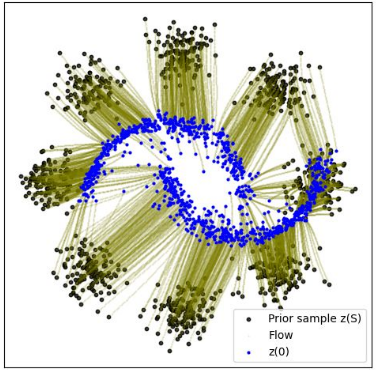
- Probability Density Path (arbitrary)

$$p_t(z) \begin{cases} p_{t=0}(z) = \mathrm{Gaus}(z) \\ p_{t=1}(z) = \mathrm{Target}\ pdf \end{cases}$$

  - u_t is the associated vector field
- Conditional Flow Matching
  - Probability path per-example x
  - Regression of u_t with the neural network v_t

$$\boxed{\mathcal{L}_{\mathrm{CFM}} = \mathbb{E}_{t,q(x),p_t(z|x)} \| v_t(z) - u_t(z|x) \|^2}$$





Taken from
https://github.com/atong01/conditional-flow-matching

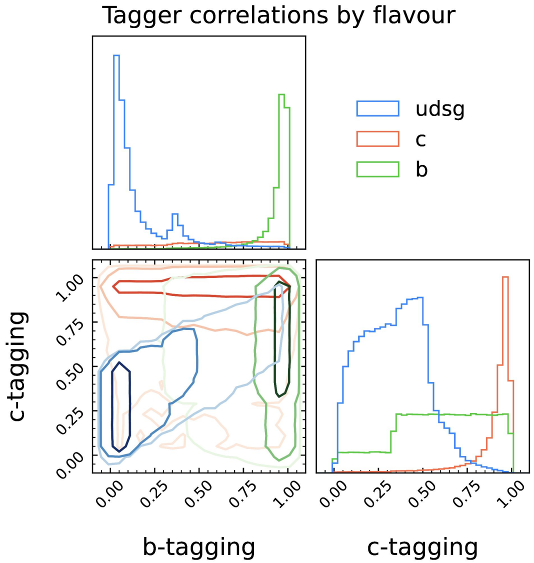
- Gaussian Probability Path
  - More regular trajectories and much easier to train

$$p_t(z|x) = \mathrm{Gaus}(z; tx, 1-(1-\sigma_{\min})t)$$

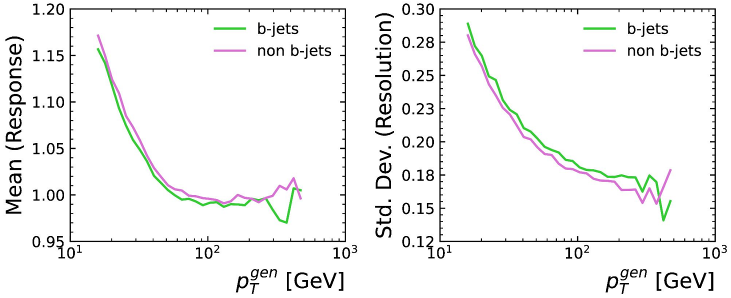
$$u_t(z|x) = \frac{x-(1-\sigma_{\min})z}{1-(1-\sigma_{\min})t}$$

https://arxiv.org/abs/2210.02747
https://arxiv.org/abs/2302.00482

Dataset Description



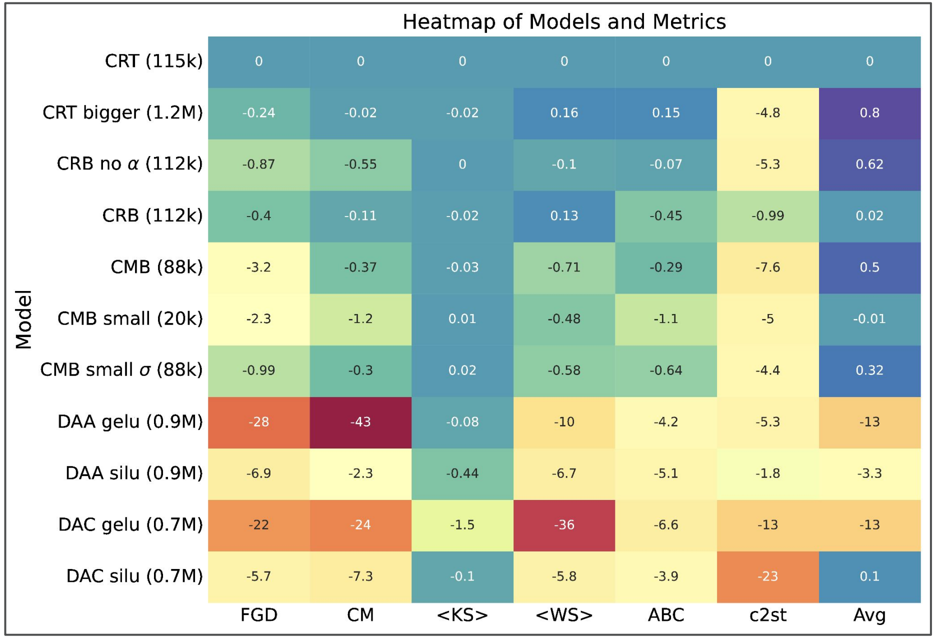
- Jet dataset
  - PYTHIA8 generator (tt*bar*, Z+jets, WW, QCD multijet)
  - Jet clustering using Fastjet
  - Simple (but realistic) detector response (correlations)



- Generator-level input (6 variables)
  - Kinematics, jet flavour, number of muons
- Target (16 variables)
  - Kinematics, b/c-tagging, energy fractions, number of secondary vertices

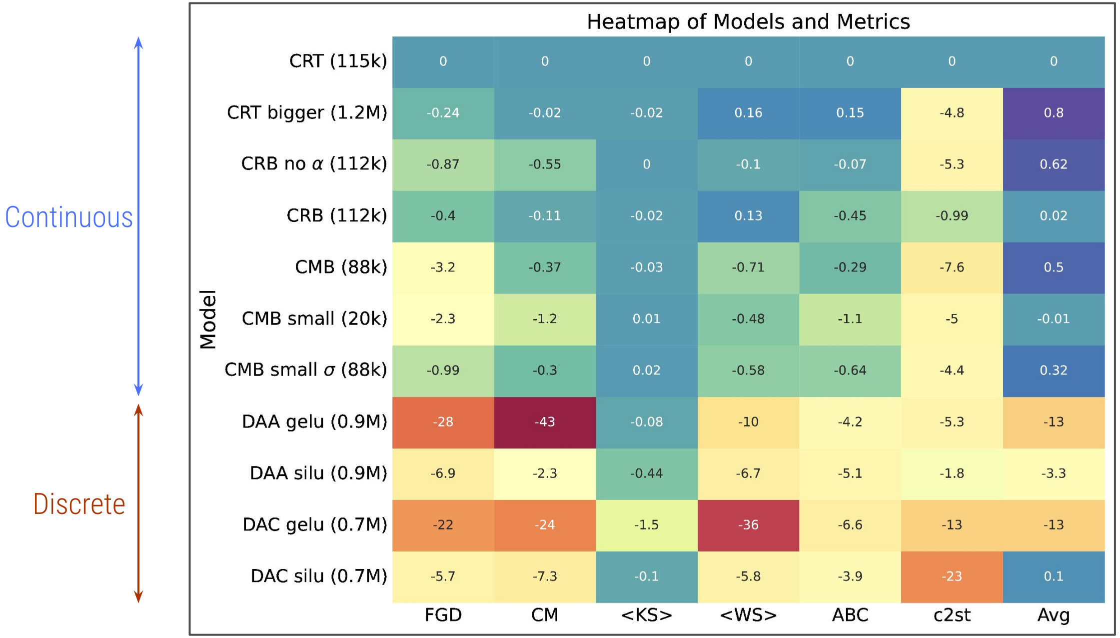
Model Comparison and Metrics

- **Comparison of different architectures**
  - Training on 500k jets
  - Validation on 650k
- **Metrics**
  - Distances on 1-dimensional distributions (Kolmogorov-Smirnov, Wasserstein)
  - “Multi-dimensional” distances (Covariance Matching, Frechet Gaussian distance)
  - Classifier Two Sample test (Gradient Boosting)
  - Area between b-tagger ROC curves (ABC)



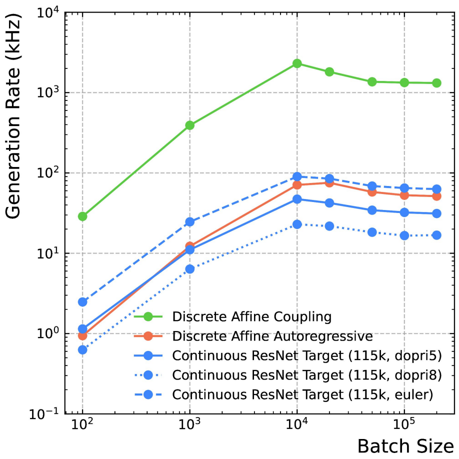
Model Comparison and Metrics

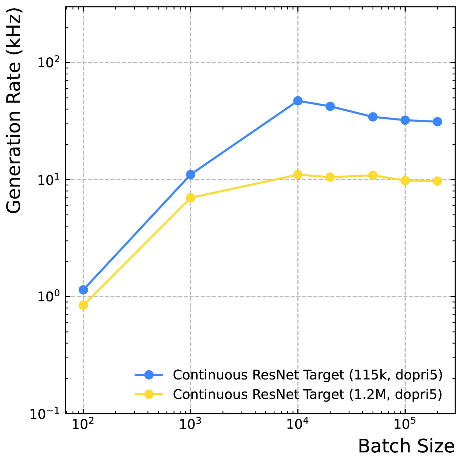
- Discrete Flows
  - Affine + Autoregressive/Coupling
  - Different layer activation functions
- Continuous Flows
  - Different Flow Matching strategies
  - ResNet/MLP
- CFM architectures perform best on every metric



Simulation Speed

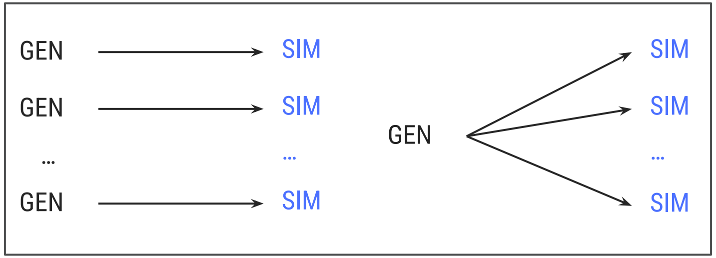
- **Simulation rate depends on the architectures**
  - Discrete Affine-Coupling is the fastest
- **Continuous ResNet Target achieves up to 100 kHz**
  - Depending on the ODE solver
  - Increasing the number of parameters (×10) slows down the rate up to a factor 4





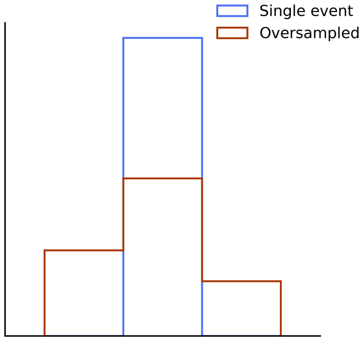
Applications

- **Increasing the size of a simulated dataset**
  - Producing more GEN events and using the flow-based response (*one–to–one*)
  - Using the same GEN event to produce more SIM events (*one–to–many*)

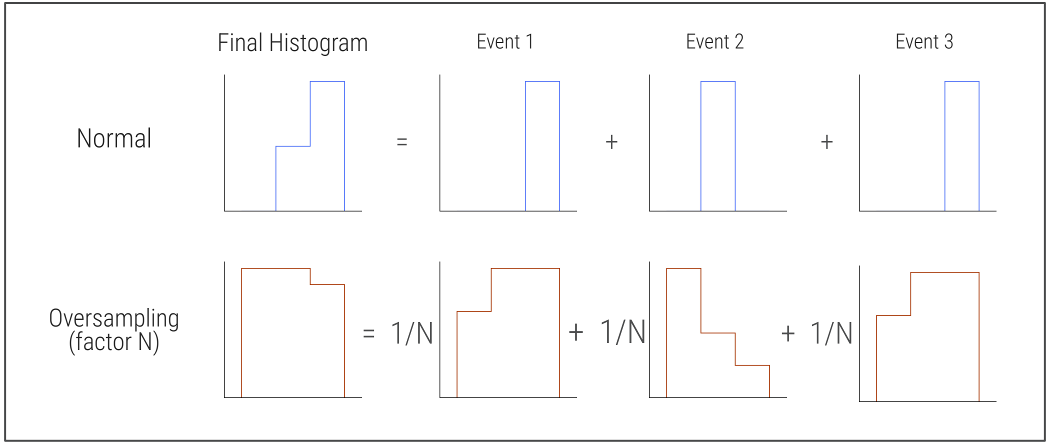


- **Oversampling**
  - Possible because of the stochasticity of the flow-based simulation
  - Useful if the GEN time becomes a bottleneck
  - Events sharing the same GEN are correlated

Statistical Treatment for Oversampling

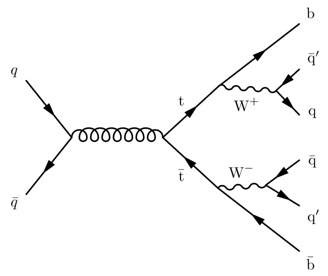


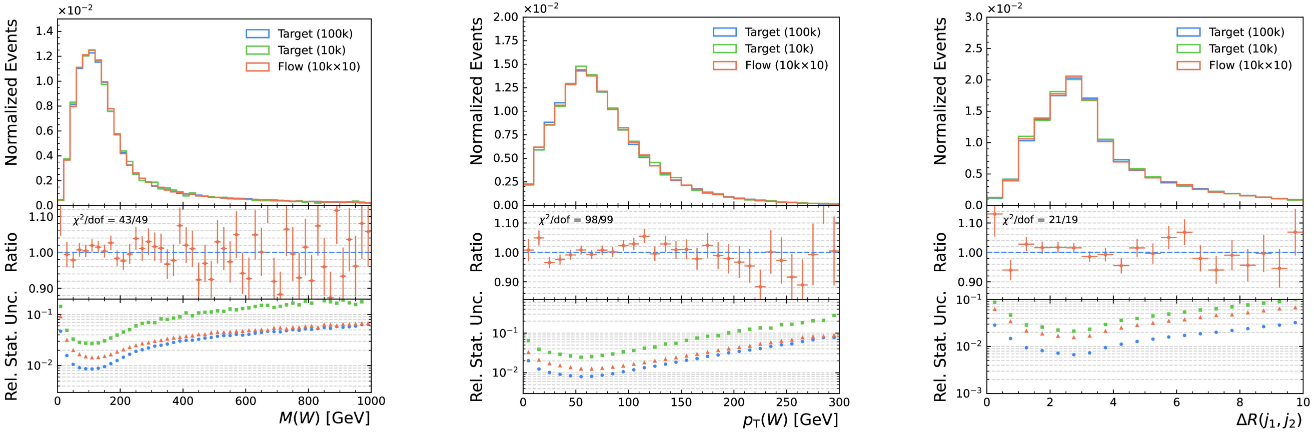
- Oversampling
  - 1 GEN event is associated with a *distribution* of SIM events
  - Final histogram is the weighted sum of sub-histograms
- **Final uncertainty is larger than just filling the histogram**



Results on Pseudo-Analysis

- Test on pseudo-analysis
  - Reconstruction of W boson in tt*bar* production
  - Statistical Uncertainty reduction for low-resolution variables (e.g. W mass)
  - No significant biases with equal number of events



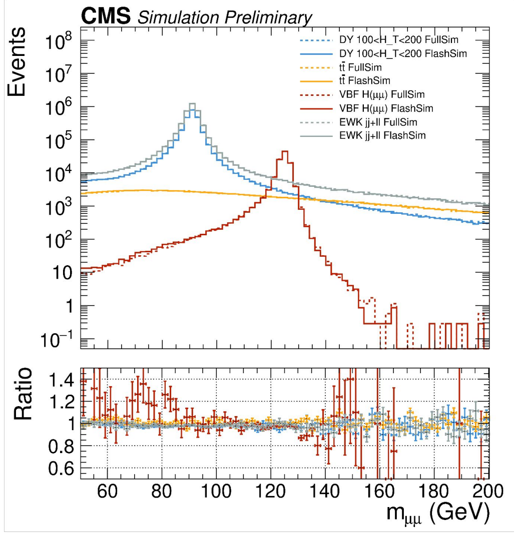


Conclusions

- **Tested multiple models on benchmark jet dataset**
  - Conditional Flow Matching has the best performances
  - Estimated simulation rate 10–1000 larger than the conventional simulation

			Millions of events per day on a HPC Node						*Ratio to Conventional sim*				
Generator	**Gen time s/event**	**Fold size**	**Conventional (20s/event)**	**Object sampling speed [kHz] 1**	**5**	**10**	**50**	**100**	**Object sampling speed [kHz] 1**	**5**	**10**	**50**	**100**
Existing	0	1	0.138	17.3	86.4	172.8	864.0	1728.0	125	625	1250	6250	12500
Simple	0.02	1	0.138	15.4	53.2	76.8	119.2	128.0	111	385	556	863	927
		10	0.138	17.1	81.3	153.6	531.7	768.0	123	588	1111	3847	5556
Average	1	1	0.132	2.4	2.7	2.7	2.8	2.8	18	20	21	21	21
		10	0.138	10.6	20.9	23.8	26.8	27.2	77	152	173	195	198
Accurate and slow	20	1	0.069	0.14	0.14	0.14	0.14	0.14	2	2	2	2	2
		10	0.126	1.28	1.4	1.4	1.4	1.4	10	11	11	11	11

- **Approach used in CMS FlashSim**
  - It scales to higher dataset dimension, multiple objects and real detector response
  - Good results in simplified analysis (Andrea Rizzi's Monday Plenary Talk)



Contacts

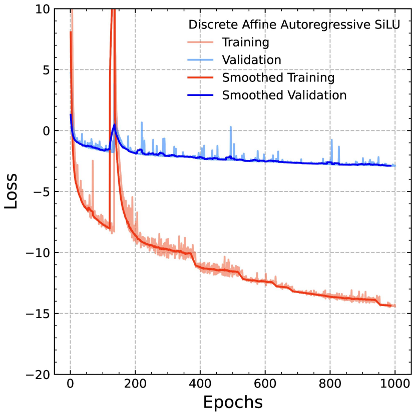
For more information, feel free to reach out filippo.cattafesta@cern.ch

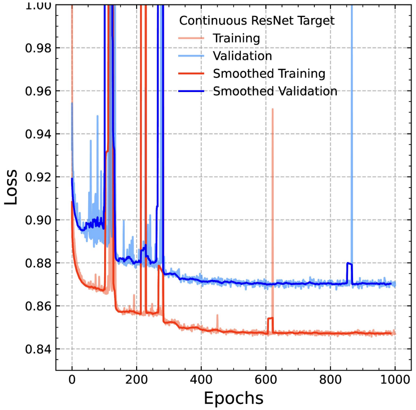
Further details
https://iopscience.iop.org/article/10.1088/2632-2153/ad563c

Project repository:
https://github.com/francesco-vaselli/FlowSim

Backup

Losses





$$p_x(\mathbf{x}) = p_z(f^{-1}(\mathbf{x}))\det\left|\frac{d\mathbf{z}}{d\mathbf{x}}\right|$$

$$\log(p_x(x)) = \log(p_z(f^{-1}(\mathbf{x}))) + \log\left(\det \mathbb{J}_{f^{-1}}(\mathbf{x})\right)$$

$$\downarrow$$

$$\mathcal{L}(\phi) = -\mathbb{E}_{p_x^*(\mathbf{x})}[\log(p_z(f^{-1}(\mathbf{x};\,\phi))) + \log\left(\det \mathbb{J}_{f^{-1}}(\mathbf{x};\,\phi)\right)]$$

Oversampling: Statistical Treatment

- Non-oversampled case
  - w statistical weight associated with the MC event
  - For the i-th bin of an histogram, the probability of being in this bin and the associated uncertainty are

$$p_i = \frac{\sum_{j \in \text{bin}} w_j}{\sum_{k \in \text{sample}} w_k} \qquad \sigma_i = \frac{\sqrt{\sum_{j \in \text{bin}} w_j^2}}{\sum_{k \in \text{sample}} w_k}$$

- Oversampled case
  - A fold is the set of RECO events sharing the same GEN

$$p_i = \frac{\sum_{j \in \text{bin}} \sum_{l \in \text{fold} \in \text{bin}} w_{jl}}{N \sum_{k \in \text{sample}} w_k} = \frac{\sum_{j \in \text{bin}} \sum_{l \in \text{fold} \in \text{bin}} w_{jl}/N}{\sum_{k \in \text{sample}} w_k} \equiv \frac{\sum_{j \in \text{bin}} w_j p_j^{\text{fold}}}{\sum_{k \in \text{sample}} w_k}$$

$$\sigma_i = \frac{\sqrt{\sum_{j \in \text{bin}} (w_j p_j^{\text{fold}})^2}}{\sum_{k \in \text{sample}} w_k}$$

Dataset Details

Generator level variables	Description
p_T, η, ϕ, mass	Kinematic properties of the generated jet
jet flavour	Distinguishing b, c jets from light quarks or gluon jets
number of μ in jet	Counting the number of muons within the jet radius
Basic reconstructed variables	**Description**
p_T, η, ϕ, mass	Kinematic properties of the reconstructed jet
b-tagging discriminator	Score in [0,1] mimicking a tagging algorithm
number of constituents	Counting the number of reconstructed jet constituents
Extended dataset variables	**Description (in addition to basic variables)**
Neutral Hadron Fraction (nhf)	fraction of jet energy carried by Neutral Hadrons
Charged Hadron Fraction (chf)	fraction of jet energy carried by Charged Hadrons
Neutral Electromagnetic Fraction (nef)	fraction of jet energy carried by photons and π^0 mesons
Charged Electromagnetic Fraction (cef)	fraction of jet energy carried by electrons
Quark-Gluon discriminator (qgd)	Discriminator score mimicking a quark/gluon tagging algorithm
Jet Identification (jetId)	Discriminator score mimicking a jet Identification algorithm
Number of Charged Particles (ncharged)	Number of reconstructed charged particles
Number of Neutral Particles (nneutral)	Number of reconstructed neutral particles
c-tagging discriminator	Score of c-tagging algorithm, correlated with b-tagging
Number of Secondary Vertices (nSV)	Poisson distributed number of Secondary Vertices in jets

- The 1-d *Wasserstein* score (WS) [36] and the two-sample *Kolmogorov-Smirnov* distance (KS) for comparing 1-d distributions between the target and the samples produced by the model. A WS is assigned to each variable.
- The *Fréchet* distance as a global measure. It is the distance between Multivariate Gaussian distributions fitted to the features of interest, which [36] calls the Fréchet Gaussian Distance (FGD). It is generally called the Fréchet Inception Distance (FID) in image generation tasks:

$$d^2(x, y) = \|\mu_x - \mu_y\|^2 + \mathrm{Tr}(\Sigma_x + \Sigma_y - 2(\Sigma_x \Sigma_y)^{1/2}). \tag{8}$$

- *Covariance matching*: another global metric used to measure how well an algorithm is modelling the correlations between the various target features. Given the covariance matrices of the two samples, target and model, we compute the *Frobenius Norm* of the difference between the two:

$$||\mathrm{Cov}(X_{\mathrm{target}}) - \mathrm{Cov}(X_{\mathrm{model}})||_F = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |c_{ij}^{\mathrm{t}} - c_{ij}^{\mathrm{m}}|^2}. \tag{9}$$

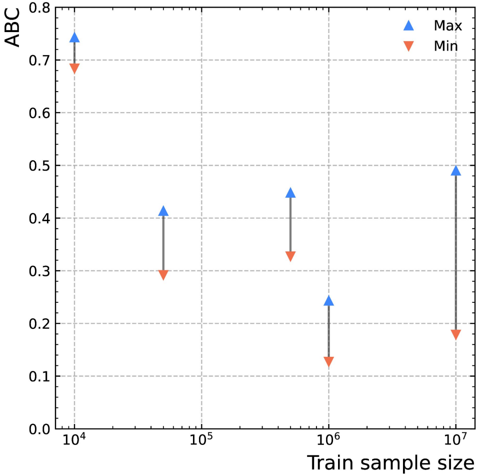
  Correlations in the model samples are also visually evaluated through the use of dedicated plots.

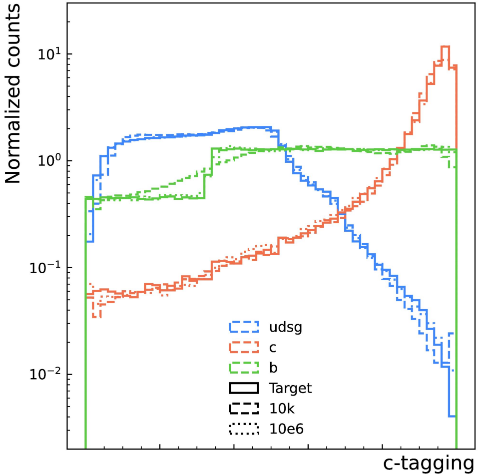
Metrics 2

- As b and c-tagging are such important tasks in the study of jets, we compute the *receiver operating characteristic* (ROC) curves for both scores. To quantify the performance of a model, we compute the difference in log-scale between the ROC coming from the model and that from the target distribution. Log-scale is used because the true positive rate (TPR) and false positive rate (FPR) span different orders of magnitude. We call this evaluation metric the *Area Between the Curves* (ABC).
- Finally, we implement a *classifier two-sample test* (c2st): we train a classifier to distinguish between training samples and samples coming from our models, giving as additional input the gen information. The output is the percentage P_{c2st} of samples which were *incorrectly* classified. For the optimal model, it has a maximum value of 0.5. We thus report our results as $0.5 - P_{\mathrm{c2st}}$: in this way the best model has the lowest c2st value. We use a scikit-learn [37] *HistGradientBoostingClassifier* with default parameters as our classifier.

Training Dataset Dependence

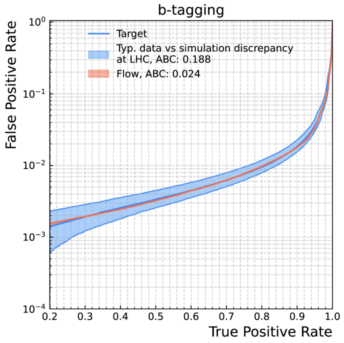
- Training dataset size variations
  - Validation on 1M of generated jets
  - Accuracy plateau reached at ~100k

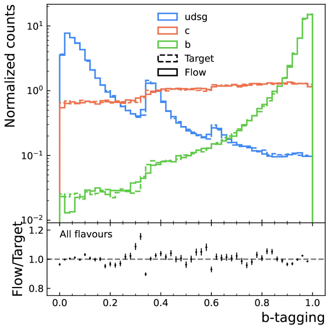


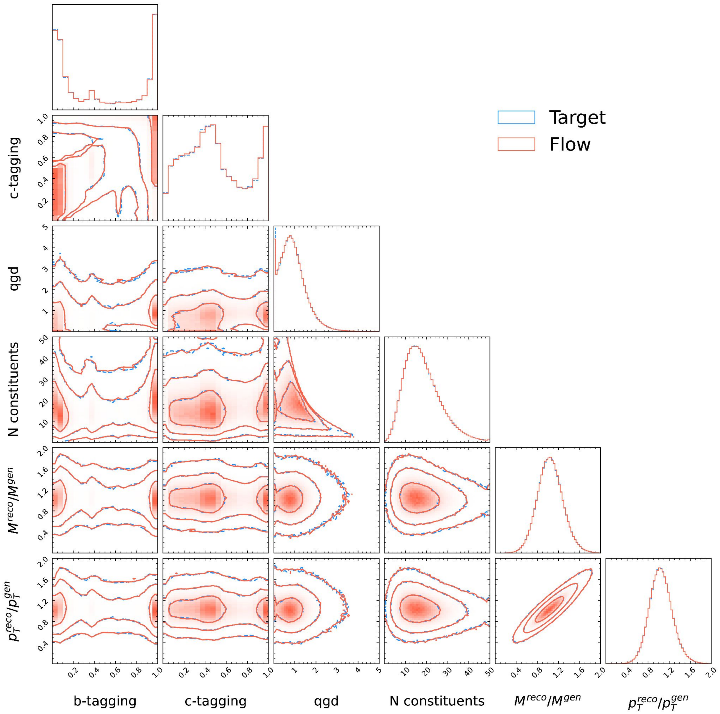


Results

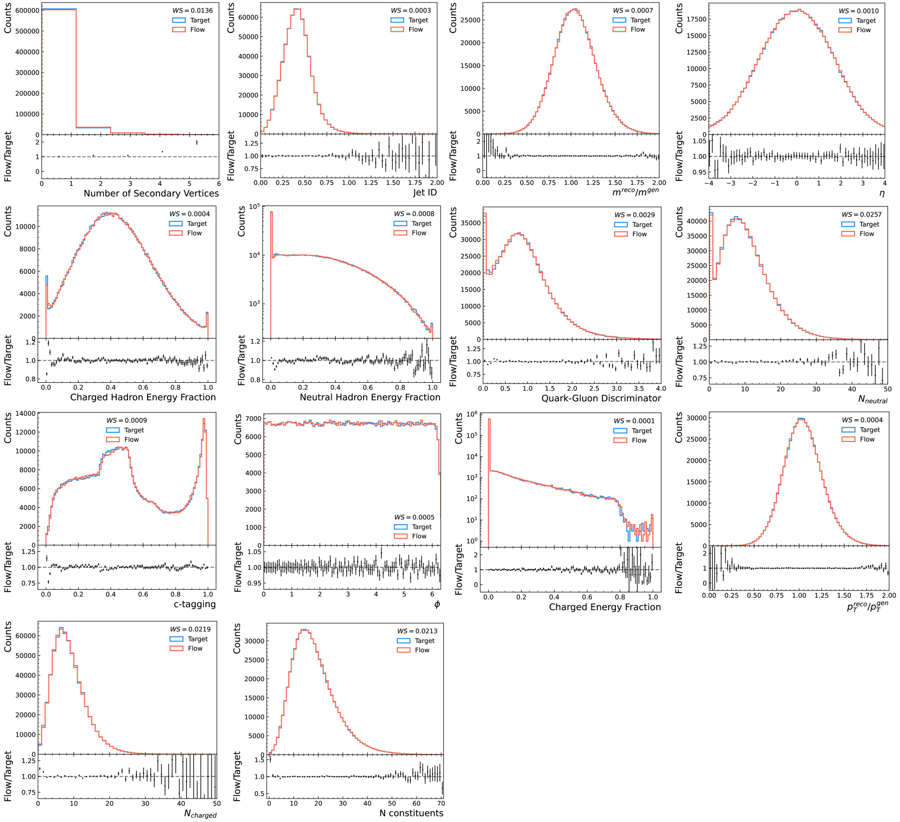
- Excellent results
  - No significant biases in 1D distributions
  - Good correlations (2D distributions)
  - Output correctly influenced by the conditioning

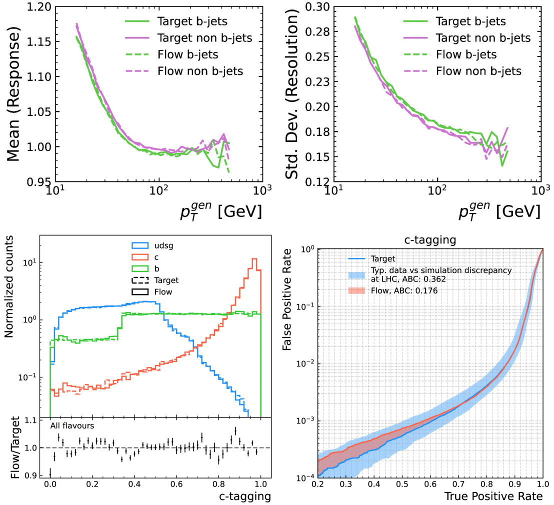




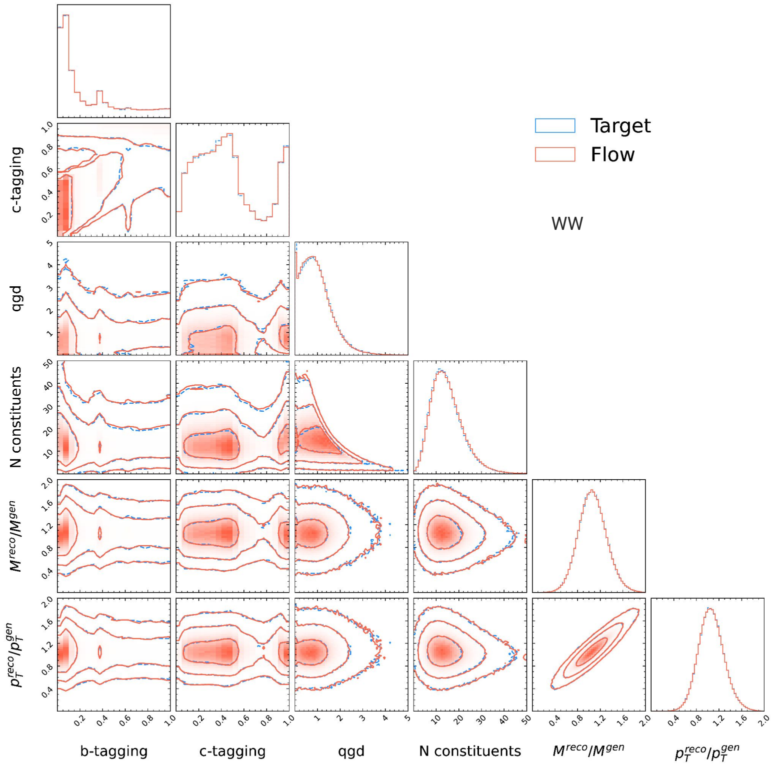


Results (tt*bar*)





Results (Other Processes)



- Validation on different processes
  - No retraining
  - Excellent performances on 1-d and correlations
- The flow learned the generalized response
  - Process independent simulation