New generative models for LHC event generation

Nathan Huetsch

Institute for Theoretical Physics
Heidelberg University

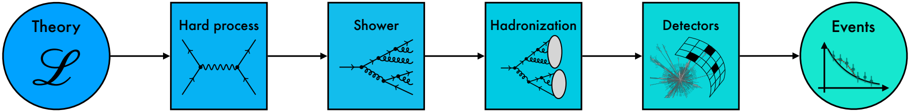
A. Butter, N. Huetsch, S. Palacios Schweitzer, T. Plehn, P. Sorrenson, J. Spinner
arXiv:2305.10475

UC Louvain May 2023

UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

Table of Contents

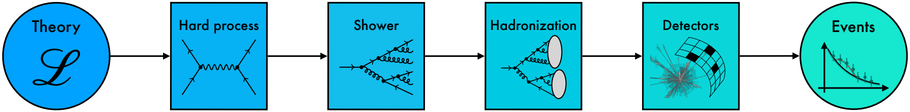
From theory to experiment in LHC physics



LHC physics is at its core simulation-based inference

Figure from A. Butter et al.: arXiv:2203.07460, R. Winterhalder

Enhancing the simulation chain with ML



- Loop integrals: arXiv:2112.09145
- Importance Sampling: arXiv:2212.06172
- Event unweighting: arXiv:2012.07873
- Hadronization: arXiv:2305.17169
- Detector simulation: arXiv:2110.11377
- **End-to-end event generation: arXiv:2305.10475**

Figure from A. Butter et al.: arXiv:2203.07460, R. Winterhalder

Generative Machine Learning

- Given a set of samples X_{train} from a distribution, the task is to learn the underlying density $p_{\text{data}}(x)$
- Most generative models are based on learning a transformation between a simple latent space and the complex target phase space

$$x \sim p_{\text{model}}(x|\theta) \approx p_{\text{data}}(x) \quad \longleftrightarrow \quad r \sim p_{\text{latent}}(r) = \mathcal{N}(0,1)$$

  where the dependence on θ represents the network training
- So far only Normalizing Flows have been shown to achieve percent-level precision for LHC event generation (arXiv:2110.13632)

Normalizing Flows

- Define the mapping between the latent and the target space as a bijective function

$$x \sim p_{\text{model}}(x|\theta) \quad \overset{G_\theta^{-1}(x)\rightarrow}{\underset{\leftarrow G_\theta(r)}{\longleftrightarrow}} \quad r \sim p_{\text{latent}}(r)$$

- Make use of the change of variables formula to write the model density as

$$p_{\text{model}}(x|\theta) = p_{\text{latent}}\left(G_\theta^{-1}(x)\right) \left|\det \frac{\partial G_\theta^{-1}(x)}{\partial x}\right|$$

  and train via Maximum Likelihood Estimation
- Construct the bijective map G as a composition of simple invertible nonlinear maps such that it is versatile enough to model complex densities yet still allows for efficient Jacobian calculation

New generative models

Diffusion

State-of-the-art image generation
Midjourney, StableDiffusion, ...

Transformer

State-of-the-art language generation
ChatGPT, Bard, ...

ChatGPT

⇒ **State-of-the-art for LHC physics applications as well?**

Diffusion Models

Generative models

Learn mapping between simple latent space and target phase space

$$x \sim p_{\text{model}}(x|\theta) \quad \longleftrightarrow \quad z \sim p_{\text{latent}}(z) = \mathcal{N}(0,1)$$

Diffusion models

Define mapping as time-dependent diffusion process

$$x_0 \sim p_{\text{model}}(x_0|\theta) \quad \xleftrightarrow{\;t\;} \quad x_T \sim p_{\text{latent}}(x_T) = \mathcal{N}(0,1)$$

$\Rightarrow$: Gradually add noise to data samples to transform them to gaussians

$\Leftarrow$: Gradually remove noise from gaussians to obtain data samples

Conditional Flow Matching [arXiv:2210.02747]

- The time evolution of individual samples follows an ODE

$$\frac{dx(t)}{dt} = v(x,t)$$

- The time evolution of the density follows a continuity equation

$$\frac{\partial p(x,t)}{\partial t} + \nabla_x \left[p(x,t)v(x,t)\right] = 0\ .$$

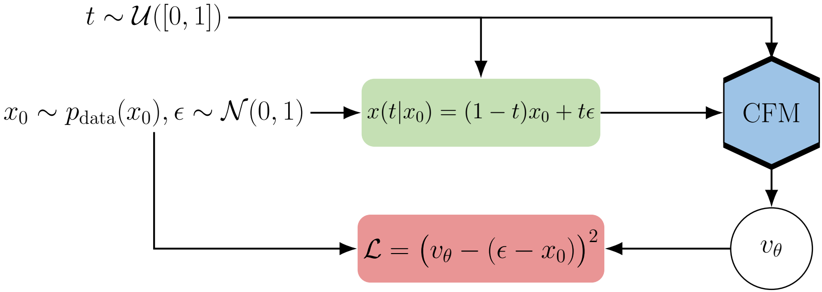
- Define the time-dependent density via

$$\begin{aligned} p(x,t) &= \int dx_0\ p(x,t|x_0)\ p_{\text{data}}(x_0) \\ &= \int dx_0\ \mathcal{N}(x;(1-t)x_0,t)\ p_{\text{data}}(x_0) \\ &\to \begin{cases} p_{\text{data}}(x) & t \to 0 \\ p_{\text{latent}}(x) = \mathcal{N}(x;0,1) & t \to 1 \end{cases} \end{aligned}$$

$\Rightarrow$ **Learn the associated velocity field v_θ from data**

CFM Training

$$\mathcal{L}_{\text{CFM}} = \left\langle \left[v_\theta(x,t) - v(x,t|x_0) \right]^2 \right\rangle_{t\sim\mathcal{U}([0,1]), x_0\sim p_{\text{data}}, \epsilon\sim\mathcal{N}(0,1)}$$



CFM models as Continuous Normalizing Flows

- Once the model is trained, the ODE defines a bijective mapping

$$\frac{d}{dt}x(t) = v_\theta(x(t), t) \qquad \text{with} \quad x_1 = x(t=1) \sim \mathcal{N}(0,1)$$

$$\Rightarrow \qquad x_0 = x_1 - \int_0^1 v_\theta(x,t)dt \equiv G_\theta(x_1)$$

- CFM models have access to phase space likelihoods like NFs

$$p_{\text{model}}(x_0|\theta) = p_{\text{latent}}(G_\theta^{-1}(x_0)) \left|\det \frac{\partial G_\theta^{-1}(x_0)}{\partial x_0}\right| \quad \text{with}$$

$$\left|\det \frac{\partial G_\theta^{-1}(x_0)}{\partial x_0}\right| = \exp\left(\int_0^1 dt \nabla_x v_\theta(x(t), t)\right)$$

Denoising Diffusion Probabilistic Model [arXiv:2006.11239]

- The forward time evolution follows a discrete Markov process

$$p(x_1, ..., x_T|x_0) = \prod_{t=1}^{T} p(x_t|x_{t-1})$$

$$\text{with} \qquad p(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t) \ .$$

- The reverse time evolution is approximated to follow the same form

$$q_\theta(x_0, ..., x_{T-1}|x_T) = \prod_{t=1}^{T} q_\theta(x_{t-1}|x_t)$$

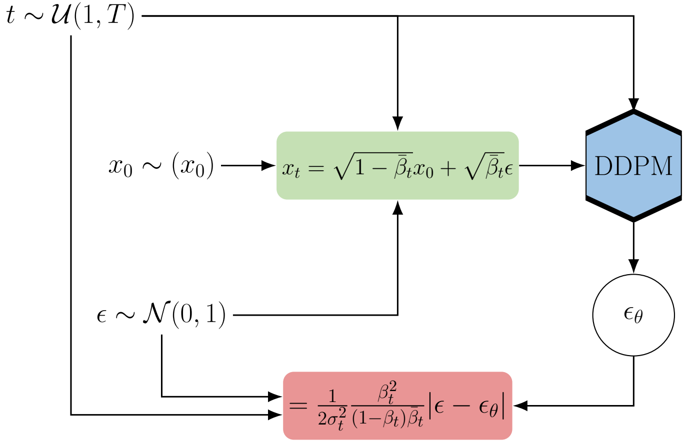
$$\text{with} \qquad q_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_\theta^2(x_t, t)) \ .$$

- A neural network is trained to fit the reverse process to the inversion of the forward process

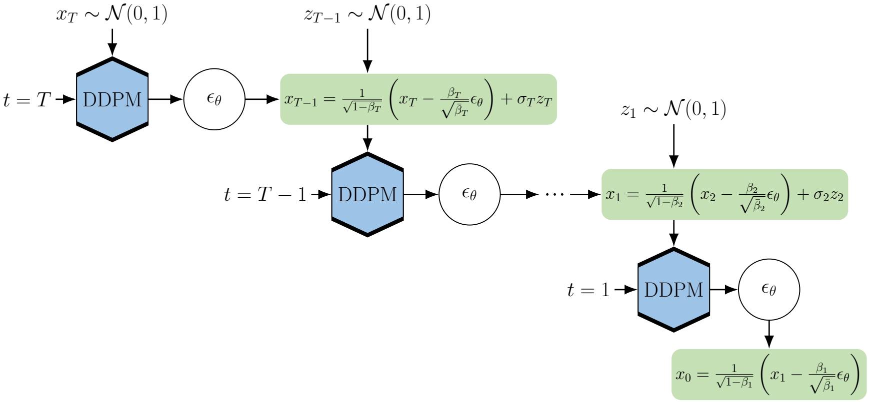
$$q_\theta(x_{t-1}|x_t) \approx p(x_{t-1}|x_t, x_0)$$

DDPM Training

$$\mathcal{L}_{\text{DDPM}} = \left\langle C_t[\epsilon(x, t|x_0) - \epsilon_\theta(x, t)]^2 \right\rangle_{t\sim\mathcal{U}(0,T), x_0\sim p_{\text{data}}, \epsilon\sim\mathcal{N}(0,1)} .$$



DDPM Sampling



Autoregressive Transformer: JetGPT

- Estimate the density autoregressively

$$p_{\text{model}}(x|\theta) = \prod_{i=1}^{n} p(x_i|x_1, ..., x_{i-1}) \approx p_{\text{data}}(x) \ ,$$

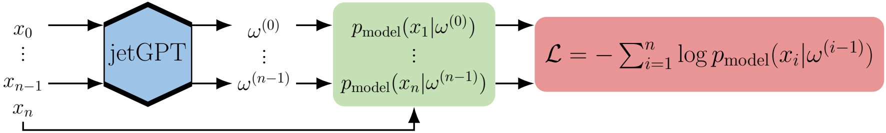
- Fit each of the conditional probabilities as a Gaussian mixture

$$p(x_i|\omega^{(i-1)}) = \sum_{\text{Gaussian } j} w_j^{(i-1)} \mathcal{N}(x_i; \mu_j^{(i-1)}, \sigma_j^{(i-1)}) \ .$$

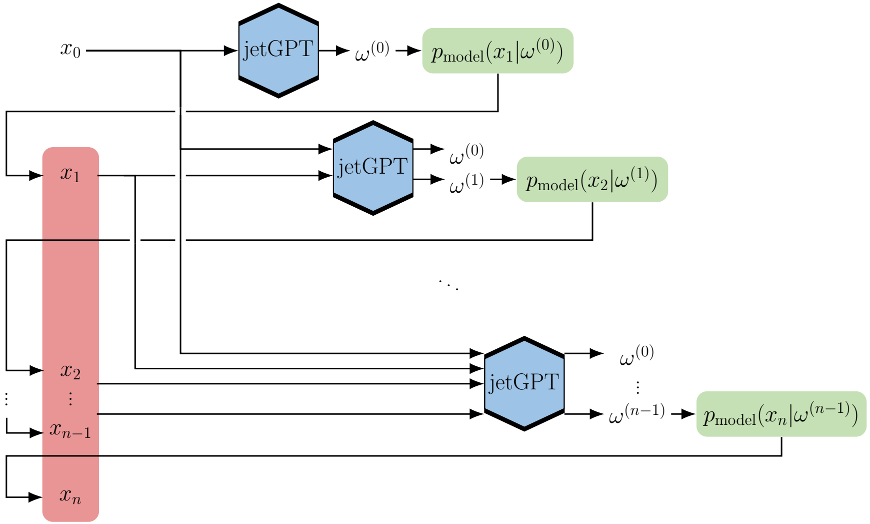
- Train a neural network to predict the parameters $\omega^{(i-1)}$ successively, always conditioned on the previous components

AT Training

$$\mathcal{L}_{\mathsf{AT}} = \sum_{i=1}^{n} \Big\langle - \log p(x_i|\omega^{(i-1)}) \Big\rangle_{x \sim p_{\mathsf{data}}}$$



AT Sampling



x_0
jetGPT
$\omega^{(0)}$
$p_{\text{model}}(x_1|\omega^{(0)})$
x_1
jetGPT
$\omega^{(0)}$
$\omega^{(1)}$
$p_{\text{model}}(x_2|\omega^{(1)})$
x_2
x_{n-1}
x_n
jetGPT
$\omega^{(0)}$
$\omega^{(n-1)}$
$p_{\text{model}}(x_n|\omega^{(n-1)})$

What about uncertainties?

The learned phase space density comes with uncertainty due to

- $\rightarrow$ Lack of training data
- $\rightarrow$ Insufficient model flexibility
- $\rightarrow$ Stochastic optimization of model parameters

Bayesian Neural Networks

1. Promote the deterministic network weights θ to distributions
2. Place a (meaningless) prior $p(\theta)$ over the weights
3. Train the network via variational approximation of the posterior

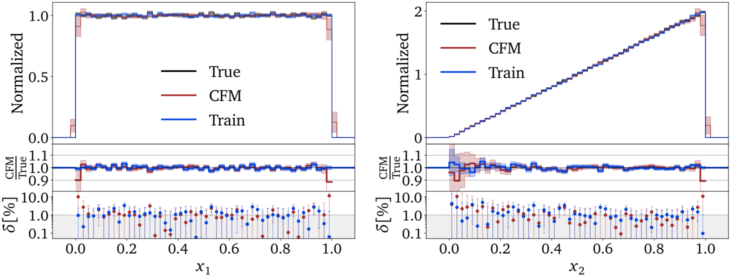
$$q(\theta) \approx p(\theta|X_{\text{train}}) = \frac{p(X_{\text{train}}|\theta)p(\theta)}{p(X_{\text{train}})}$$

4. Evaluate the network by calculating the posterior expectation

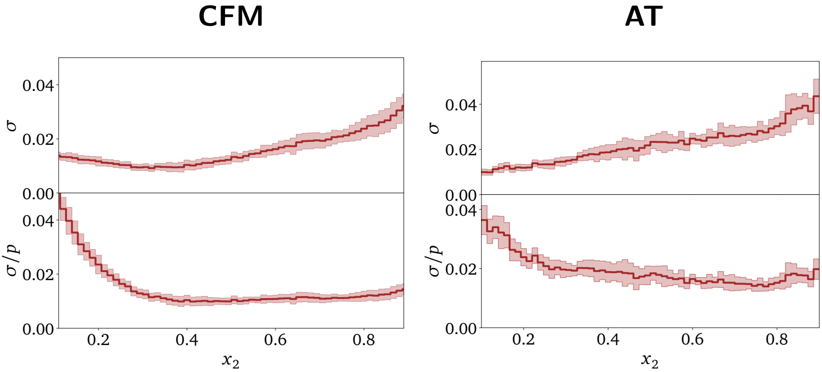
$$\langle\, p \,\rangle(x) = \int d\theta \; p(x|\theta)p(\theta|X_{\text{train}})$$

Toy example 1: Linear Ramp

$$p_{\mathrm{ramp}}(x_1, x_2) = 2x_2$$



Toy example 1: Linear Ramp



Can we understand the difference?

Toy example 1: Linear Ramp

We follow the discussion of arXiv:2104.04543:

- Consider a constrained fit to the density:

$$p(x_2) = ax_2 + b = a\left(x_2 - \frac{1}{2}\right) + 1 \qquad \text{with} \qquad x_2 \in [0, 1]$$

- Estimating a then leads to an uncertainty in the density of

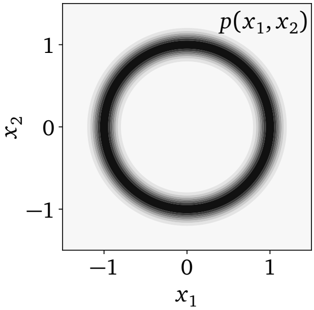
$$\sigma \equiv \Delta p \approx \left| x_2 - \frac{1}{2} \right| \, \Delta a \, ,$$

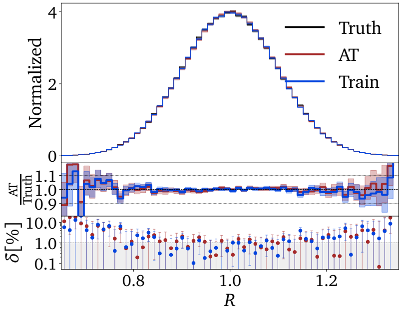
  featuring a local minimum in $x_2 = 0.5$

- Making this setup one step more realistic and also estimating the interval boundaries leads to a constant offset in the uncertainty, consistent with what is observed for Diffusion models and NFs

Toy example 2: Gaussian Ring

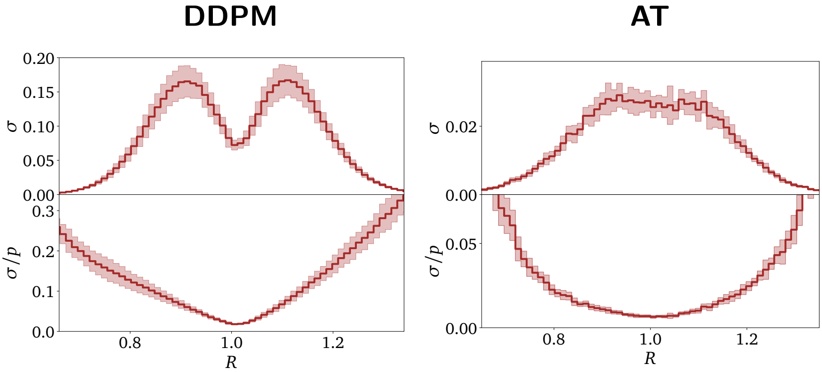
$$p_{\text{ring}}(x_1, x_2) = \mathcal{N}(\sqrt{x_1^2 + x_2^2}; 1, 0.1)$$



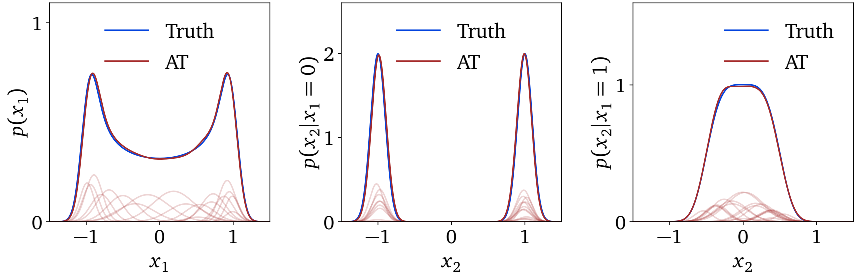


Following a similar discussion to the ramp, we find that a parametric fit would feature a minimum in the uncertainty at $R \approx 1$

Toy example 2: Gaussian Ring



Toy example 2: Gaussian Ring



LHC use case: Z+jets

- We follow the example process proposed in arXiv:2110.13632: Leptonically decaying Z boson with a variable number of QCD jets

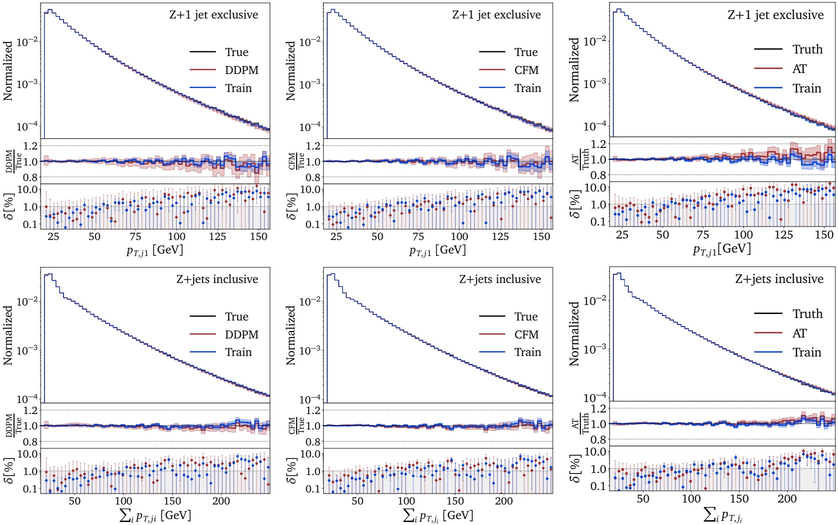
$$pp \to Z_{\mu\mu} + \{1, 2, 3\} \text{ jets} \; .$$

- Events are generated with Sherpa at 13 TeV, including ISR and parton shower with CKKW merging, hadronizaton, but no pile-up. The jets are defined using the anti-k_T algorithm and appliying the basic cuts

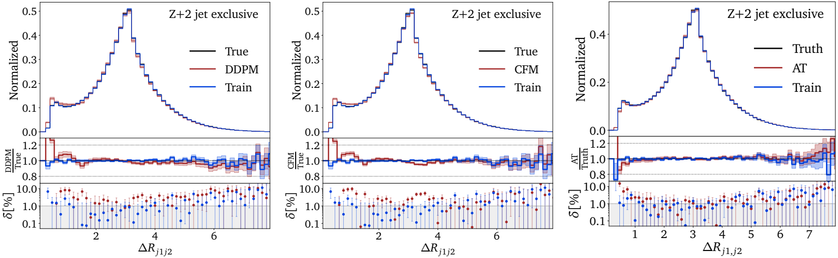
$$p_{T,j} > 20 \text{ GeV} \qquad \text{and} \qquad \Delta R_{jj} > 0.4$$

- Events are represented as $\{p_T, \eta, \phi, m\}$ and ordered by transverse momentum. The phase space dimensionality reduces to 9, 13, 17 by dropping the muon masses and one azimuthal angle

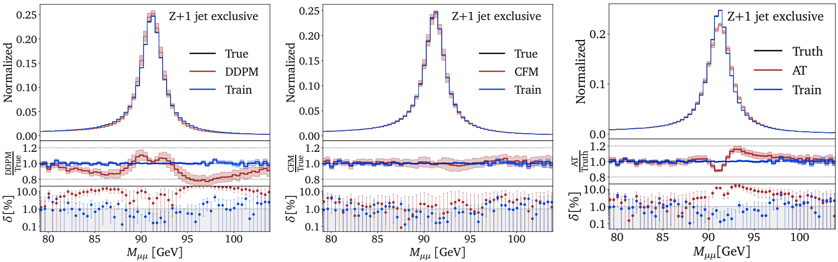
Z+jets: Transverse momenta



Z+jets: Jet seperation



Z+jets: Lepton mass peak



Summary

- We adapted two diffusion models and an autoregressive transformer model to LHC event generation and developed Bayesian versions that allow us to quantify their uncertainties
- Experiments on toy examples indicate that diffusion models, similar to Normalizing Flows, show patterns of a constrained fit while the transformer learns the density patch-wise
- These new models match or even surpass the percent-level precision of Normalizing Flows in end-to-end LHC event generation

Outlook

- The next step is to incorporate these models into different parts of the LHC simulation and analysis chain
- This includes, but is not limited to Importance Sampling, Matrix Element Methods, Unfolding, ...
- We expect that LHC physics will benefit from different model classes

Conditional Flow Matching

- Define a conditional diffusion process that evolves a sample x_0

$$
\begin{aligned}
x(t|x_0) &= (1-t)x_0 + t\epsilon \\
v(x,t|x_0) &= \frac{dx(t|x_0)}{dt} = -x_0 + \epsilon \\
p(x,t|x_0) &= \mathcal{N}(x; (1-t)x_0, t)\ .
\end{aligned}
$$

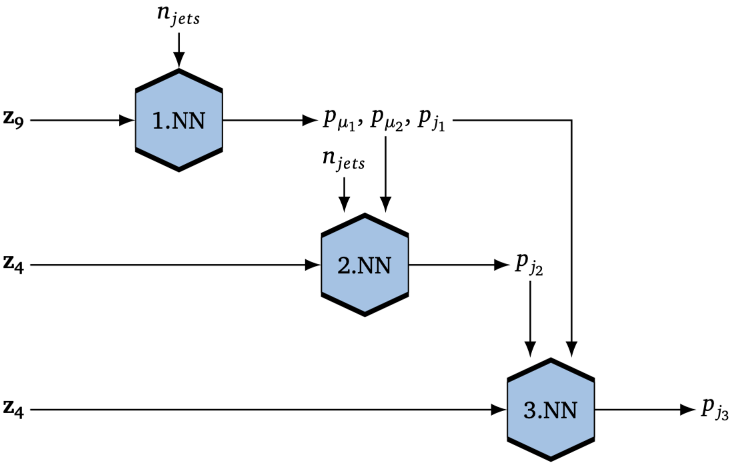
- Define the whole process as

$$
p(x,t) = \int dx_0\ p(x,t|x_0)\ p_{\text{data}}(x_0)\ .
$$

- It turns out that we can write the according velocity field as

$$
v(x,t) = \int dx_0\ \frac{v(x,t|x_0)p(x,t|x_0)p_{\text{data}}(x_0)}{p(x,t)}\ .
$$

Backup: Conditional Z+jets setup



DDPM Hyperparameters

	toy models	LHC events
Timesteps	1000	1000
Time Embedding Dimension	-	64
# Blocks	1	2
Layers per Block	8	5
Intermediate Dimensions	40	64
# Model Parameters	20k	75k
LR Scheduling	one-cycle	one-cycle
Starter LR	10^{-4}	10^{-4}
Maximum LR	10^{-3}	10^{-3}
Epochs	1000	1000, 3000, 10000
Batch Size	8192	8192, 8192, 4096
# Training Events	600k	3.2M, 850k, 190k
# Generated Events	1M	1M, 1M, 1M

CFM Hyperparameters

	toy models	LHC events
Embedding Dimension	-	32
# Blocks	1	2
Layers per Block	8	5
Intermediate Dimensions	40	128, 64, 64
# Model Parameters	20k	265k, 85k, 85k
LR Scheduling	cosine annealing	cosine annealing
Starter LR	10^{-2}	10^{-3}
Epochs	1000	1000, 5000, 10000
Batch Size	8192	16384
# Training Events	600k	3.2M, 850k, 190k
# Generated Events	1M	1M, 1M, 1M

AT Hyperparameters

	toy models	LHC events
# Gaussians m	21	43
# Bins m	64	-
# TransformerDecoder N	4	4
# Self-attention Heads	4	4
Latent Space Size d	64	128
# Model Parameters	220k	900k
LR Scheduling	one-cycle	one-cycle
Starter LR	3×10^{-4}	10^{-4}
Maximum LR	3×10^{-3}	10^{-3}
Epochs	200	2000
Batch Size	1024	1024
RAdam ϵ	10^{-8}	10^{-4}
# Training Events	600k	2.4M, 670k, 190k
# Generated Events	600k	1M, 1M, 1M