Generating Particle Cloud Jets with Denoising Diffusion

Hammers and Nails, Switzerland, 2023

Matthew Leigh, Debajyoti Sengupta, Johnny Raine, Guillaume Quetant, Tobias Golling

UNIVERSITY OF GENEVA

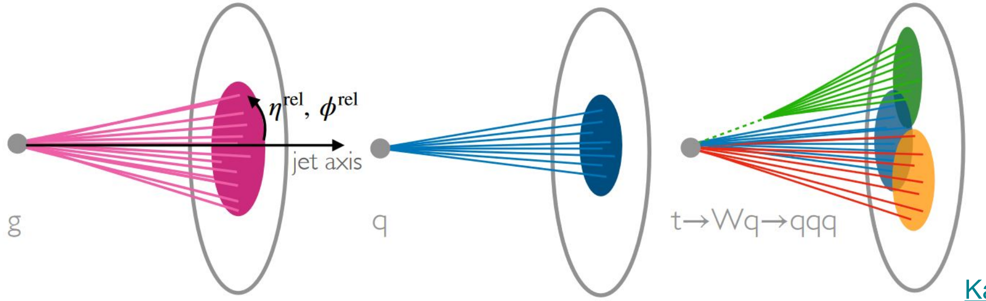
Current work

- **PC-JeDi**: Diffusion for Particle Cloud Generation in High Energy Physics
    - https://arxiv.org/abs/2307.06836
    - March 2023
    - Theory based on Score-Based Generative Modeling through Stochastic Differential Equations

- **PC-Droid**: Faster diffusion and improved quality for particle cloud generation
    - https://arxiv.org/abs/2307.06836
    - July 2023
    - Theory based on Elucidating the Design Space of Diffusion-Based Generative Models and Consistency Models

Problem and Goal

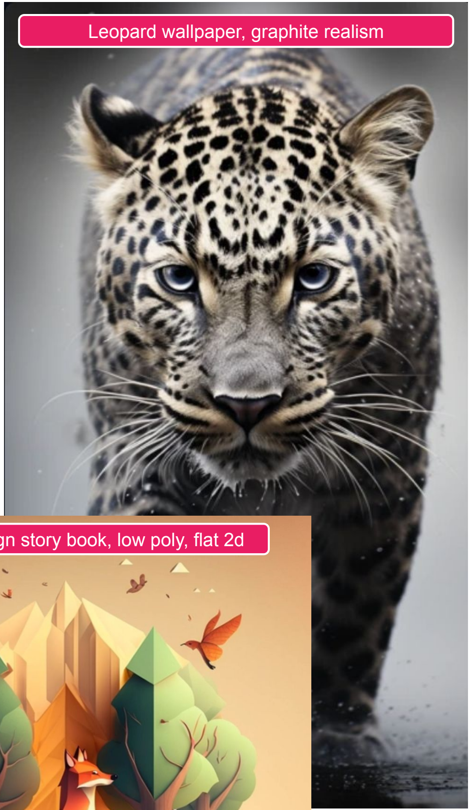
- Deep learning methods for **jet generation** is a growing topic in HEP
  - Fast-Sim:
    - ML methods can **improve generation times** by orders of magnitude
  - Template building:
    - Use for anomaly detection
    - CATHODE



Kansal et. al. 2022

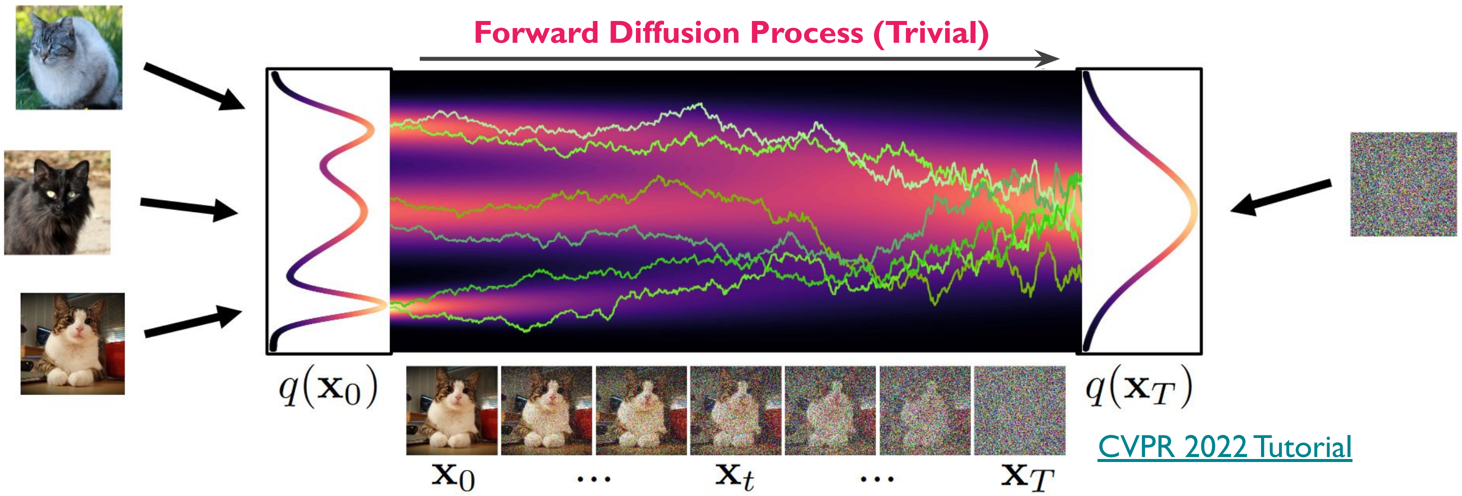
Proposal

- Use **Diffusion**
  - Generation = iterative **denoising steps**
- **Point clouds**
  - Replace the typical UNet with a message passing network
- Can use the **conditional generation**
  - Generate jets with desired high-level features
    - Momentum, mass, signal type
  - Required for Fast-Sim and template building



Score Matching Theory

Diffusion as an **SDE**

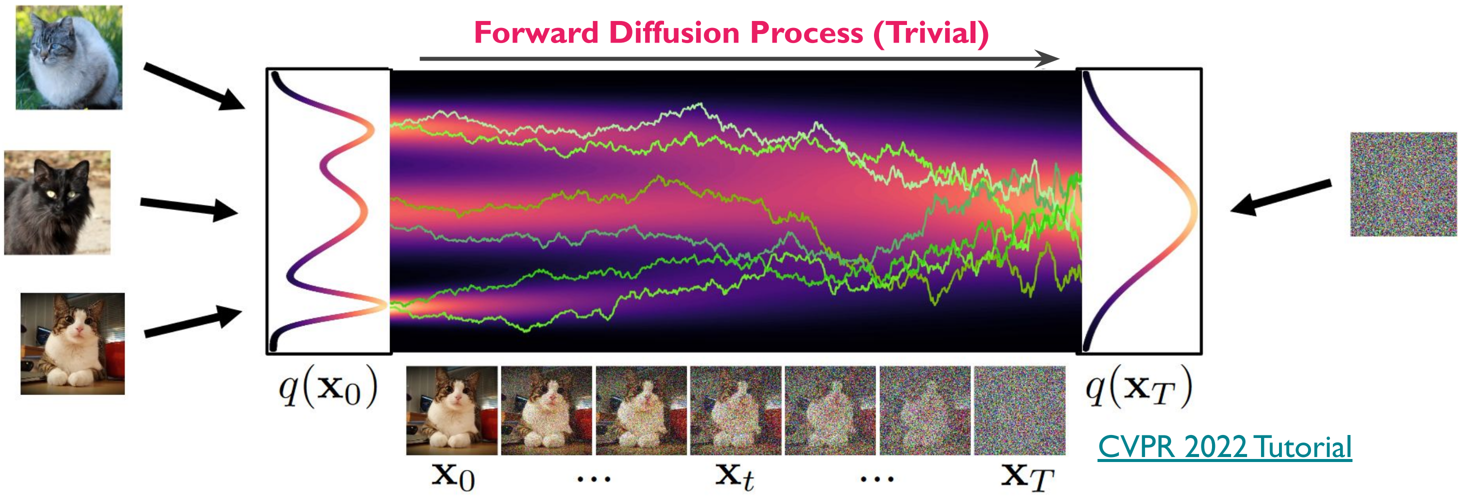


CVPR 2022 Tutorial

$$\mathrm{d}\boldsymbol{x}_t = f(\boldsymbol{x}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w}$$

***f*, *g* are hyperparameters**

Diffusion as an **SDE**



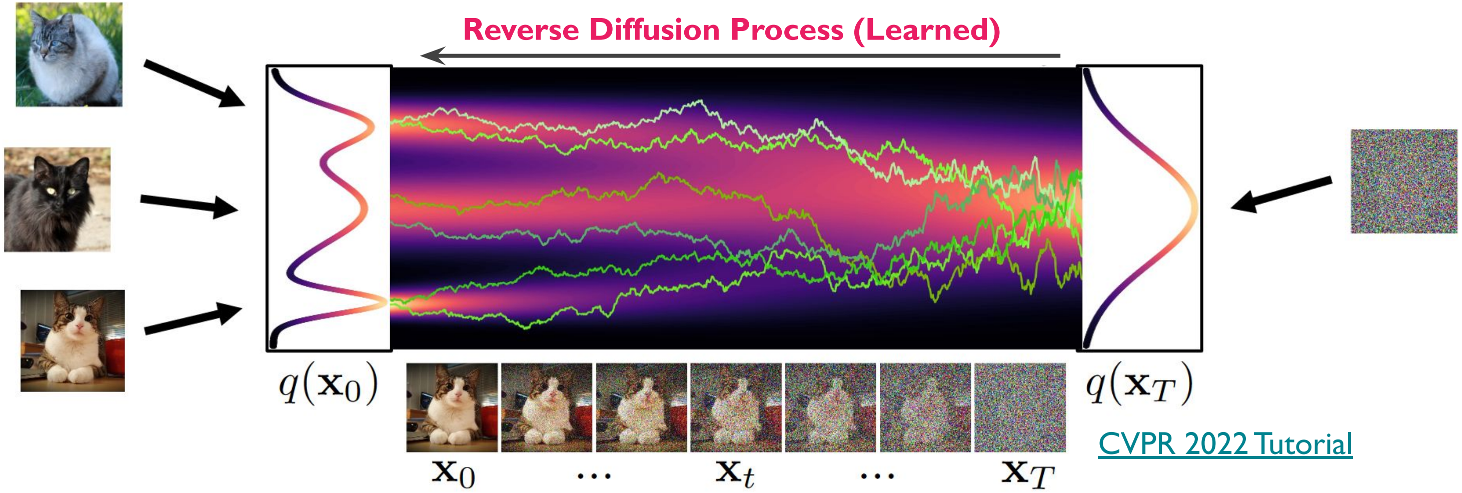
CVPR 2022 Tutorial

$$\mathrm{d}\boldsymbol{x}_t = f(\boldsymbol{x}_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}\mathbf{w}$$

If *f* is affine

$$p(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \gamma(t)\boldsymbol{x}_0, \sigma(t)^2\,\boldsymbol{I})$$

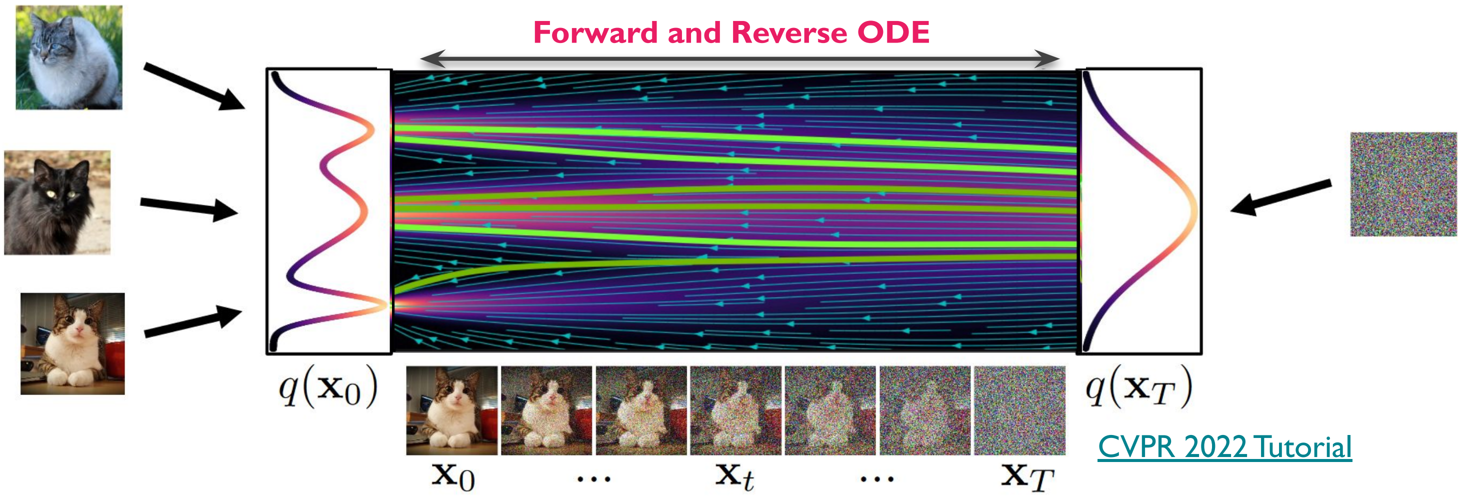
Diffusion as an **SDE**



CVPR 2022 Tutorial

$$\mathrm{d}\boldsymbol{x}_t = \left[f(\boldsymbol{x}_t, t) - g(t)^2 \underbrace{\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)}_{\text{score function}} \right] \mathrm{d}t + g(t)\, \mathrm{d}\bar{\mathbf{w}}$$

Diffusion as an **ODE**



CVPR 2022 Tutorial

$$\mathrm{d}\boldsymbol{x}_t = \left[f(\boldsymbol{x}_t, t) - \frac{1}{2} g(t)^2 \underbrace{\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)}_{\text{score function}} \right] \mathrm{d}t$$

Results in same marginal diffused densities as the SDE

Denoising Learning Objective

*full derivation in backup!

$$\min_{\theta} \mathbb{E}_{t\sim U(0,1)} \mathbb{E}_{\boldsymbol{x}_0\sim p(\boldsymbol{x}_0)} \mathbb{E}_{\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\boldsymbol{I})} \frac{1}{\sigma(t)^2} ||\hat{\boldsymbol{\epsilon}}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{\epsilon}||^2$$

Sample time:	`t~U[0,1]`
Sample data:	`x0~{Training set}`
Sample noise:	`ε~N(0,1)`$_d$
Corrupt data:	`xt = γ(t)*x0 + σ(t)*ε`
Get loss:	`L = c(t)*[NN(xt,t) - eps]^2`

Sample Generation

$$\mathrm{d}\boldsymbol{x}_t = \left[f(\boldsymbol{x}_t, t) - g(t)^2 \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)\right] \mathrm{d}t + g(t)\,\mathrm{d}\bar{\mathbf{w}}$$

$$\mathrm{d}\boldsymbol{x}_t = \left[f(\boldsymbol{x}_t, t) - \frac{1}{2} g(t)^2 \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)\right] \mathrm{d}t$$

- Numerically **integrate** reverse process
  - Which process (SDE or ODE) and which integration method is flexible
- Each step **requires a forward pass** of the network
  - Generation needs **more computation** than GANs and Flows
  - Main **detriment** to using **diffusion** models

Sample Generation

$$\mathrm{d}\boldsymbol{x}_t = \left[f(\boldsymbol{x}_t, t) - g(t)^2 \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)\right] \mathrm{d}t + g(t)\, \mathrm{d}\bar{\mathbf{w}}$$

$$\mathrm{d}\boldsymbol{x}_t = \left[f(\boldsymbol{x}_t, t) - \frac{1}{2} g(t)^2 \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)\right] \mathrm{d}t$$

- Numerically **integrate** reverse process
  - Which process (SDE or ODE) and which integration method is flexible
- Each step **requires a forward pass** of the network
  - Generation needs **more computation** than GANs and Flows
  - Main **detriment** to using **diffusion** models

Always a trade-off between time and fidelity

Sample Generation

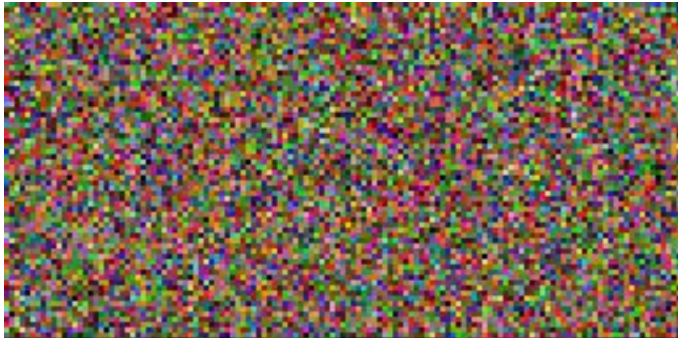
$$\mathrm{d}\boldsymbol{x}_t = \left[f(\boldsymbol{x}_t, t) - g(t)^2 \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)\right] \mathrm{d}t + g(t)\,\mathrm{d}\bar{\mathbf{w}}$$

$$\mathrm{d}\boldsymbol{x}_t = \left[f(\boldsymbol{x}_t, t) - \frac{1}{2} g(t)^2 \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)\right] \mathrm{d}t$$

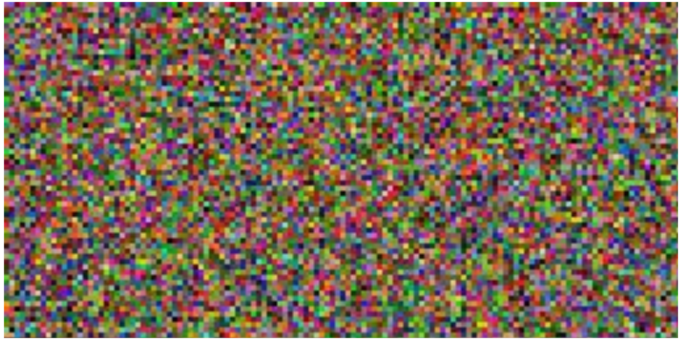
- Numerically **integrate** reverse process
  - Which process (SDE or ODE) and which integration method is flexible
- Each step **requires a forward pass** of the network
  - Generation needs **more computation** than GANs and Flows
  - Main **detriment** to using **diffusion** models

Always a trade-off between time and fidelity

SDE with Euler-Maruyama



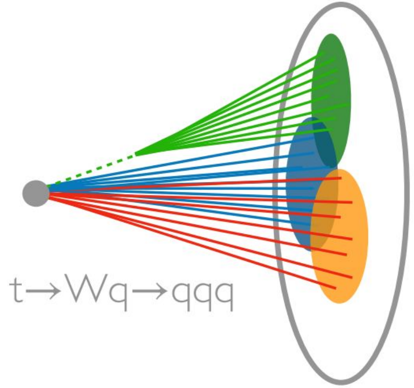
ODE with DDIM

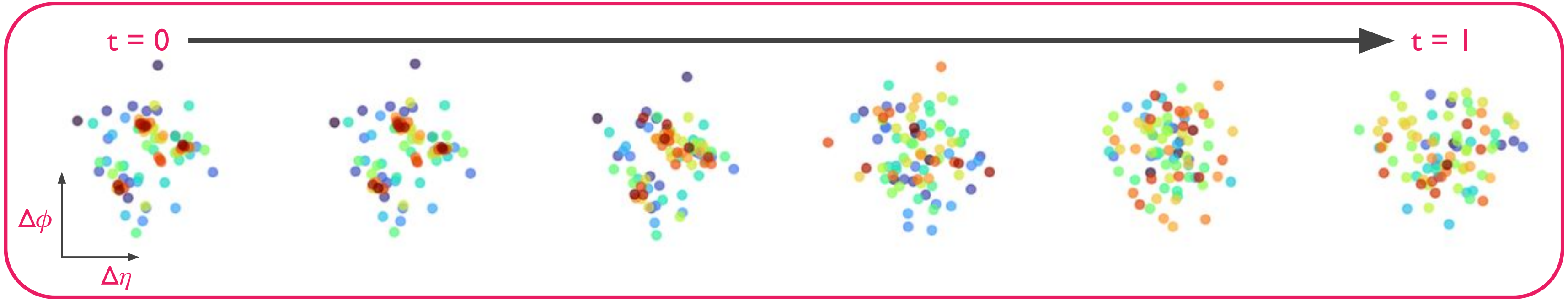


PC Droid

Dataset

- Using the JetNet dataset and metrics
- Large radius **point clouds** jets
  - Gluon, Quark, Top, W, Z
  - Up to 150 constituents
  - ($\Delta\eta$, $\Delta\phi$, p_T)

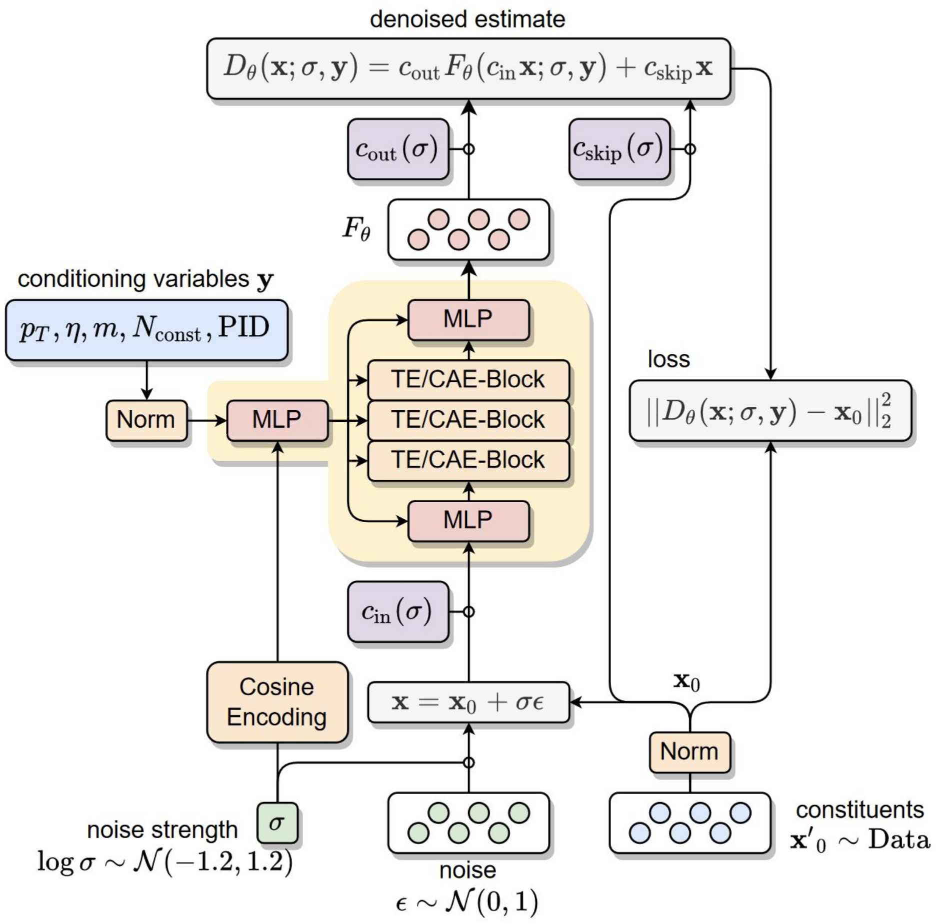




PC-Droid

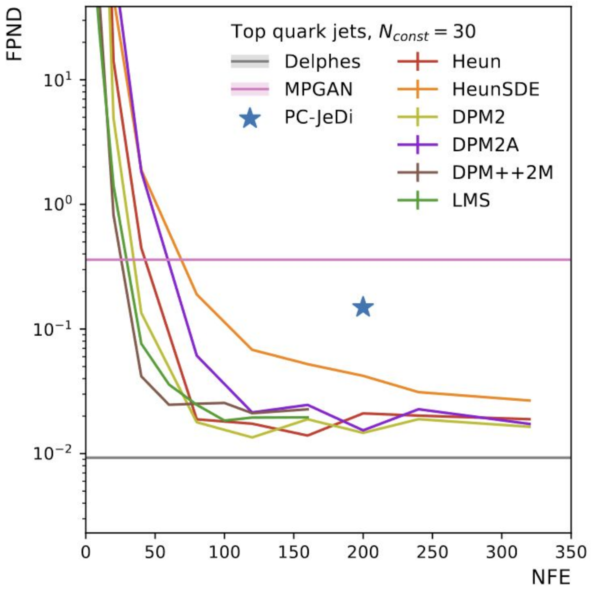
One conditional model for all jet types

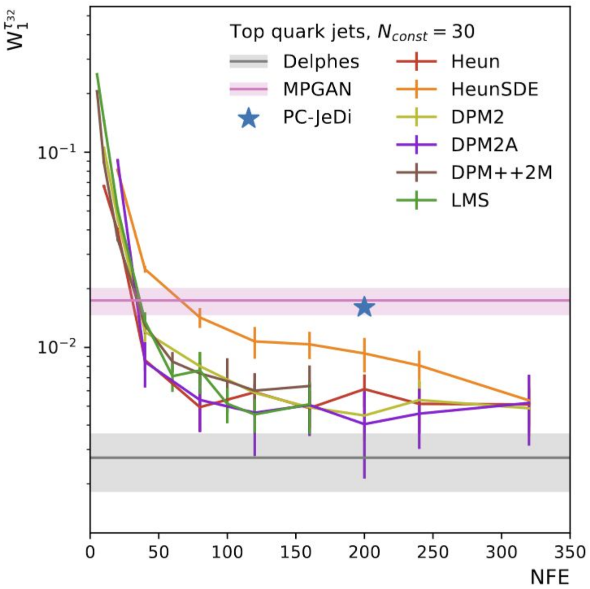
- Transformer variant $O(N^2)$
- CAE variant $O(N)$



PC-Droid Results

- Massive improvements over our older diffusion model and MPGAN on 30 constituent dataset
- Significantly overtaking SOTA models

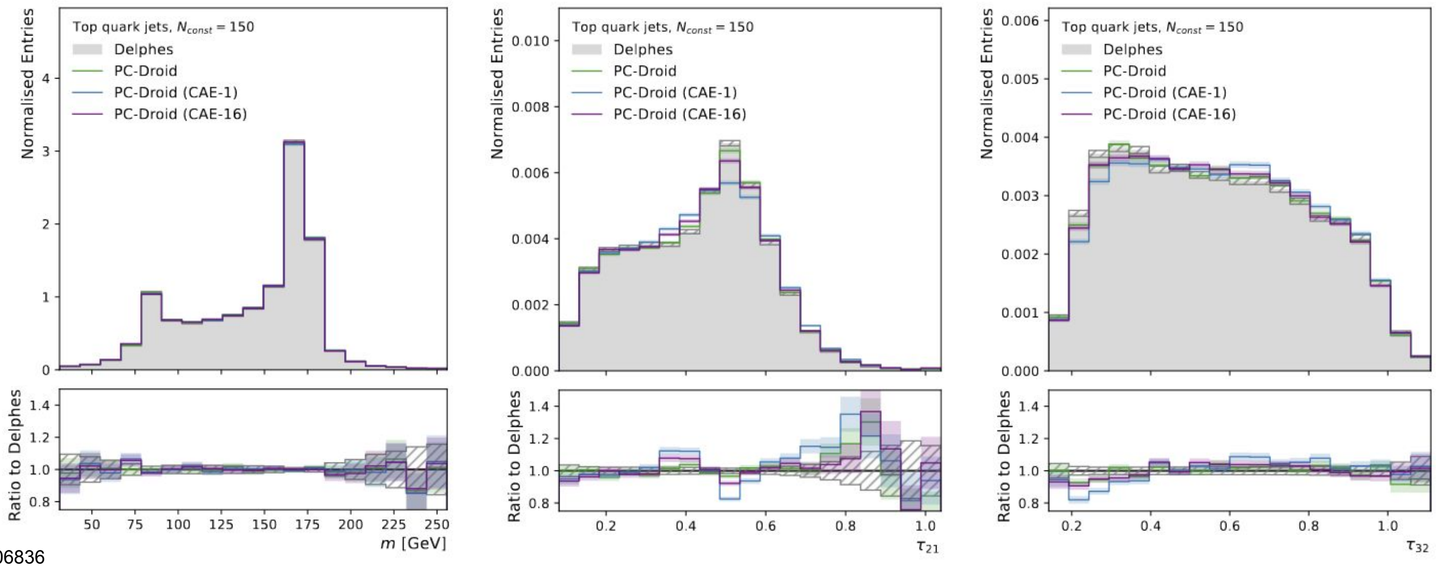




https://arxiv.org/abs/2307.06836

PC-Droid Results

- Great performance on 150 dataset
- New CAE network performs similarly with a big increase in generation speed

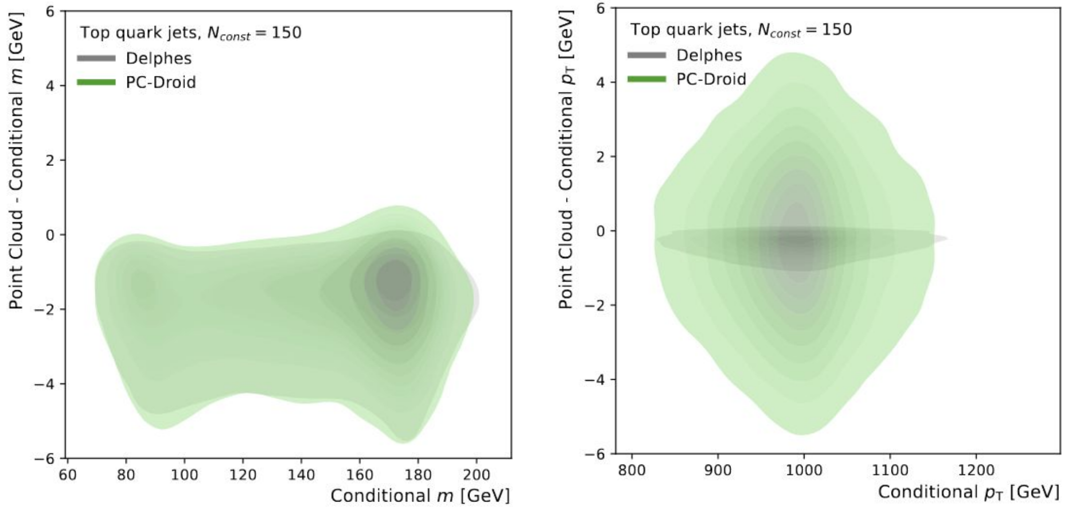


https://arxiv.org/abs/2307.06836

Conditional Adherence

Is our conditional model actually obeying its conditions?

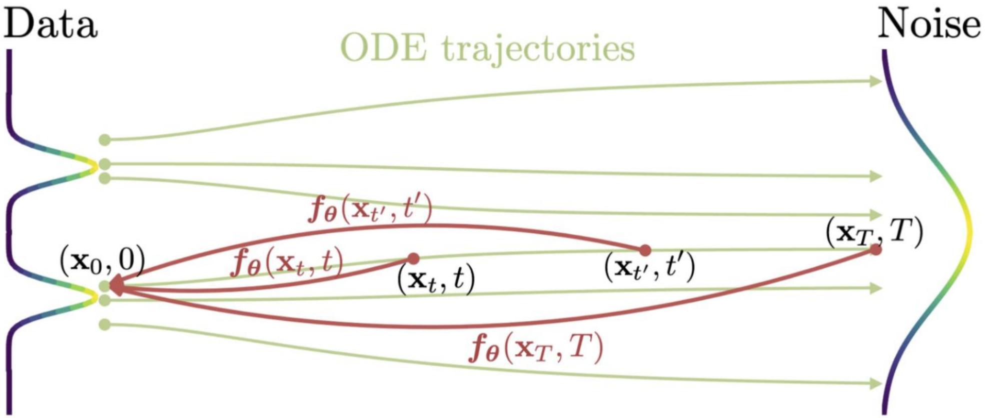
- Natural difference between conditional and point cloud variables in the data
- Slightly larger spread in $\mathbf{p_T}$
    - Majority within 0.3%



https://arxiv.org/abs/2307.06836

Consistency Distillation

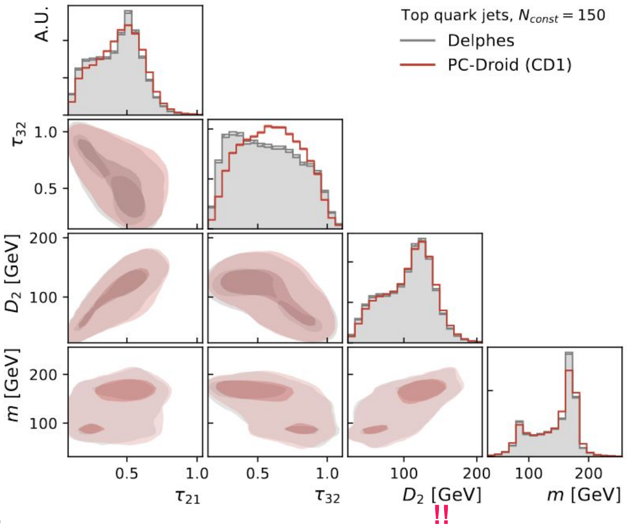
- One of many diffusion **distillation methods**
- Use a **pretrained model** to train a **student model** to perform diffusion in less steps
- In some cases even allowing generation in **1 step**

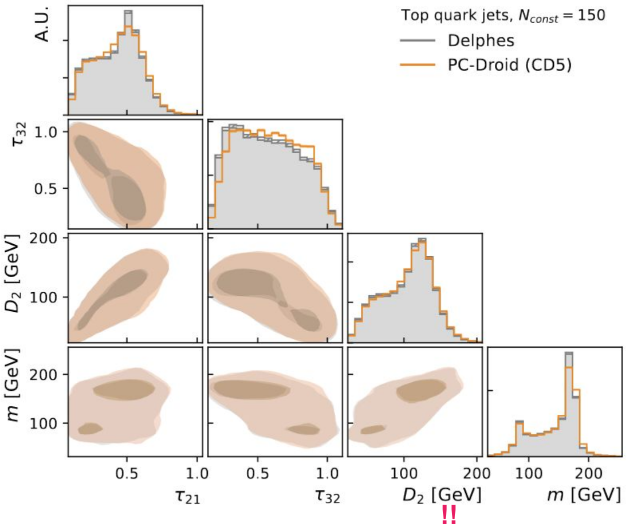


Consistency Models

CD Models Results

- Tested CD model with **1 and 5** step generation
- **Significantly faster** than base model (100x) but lower quality

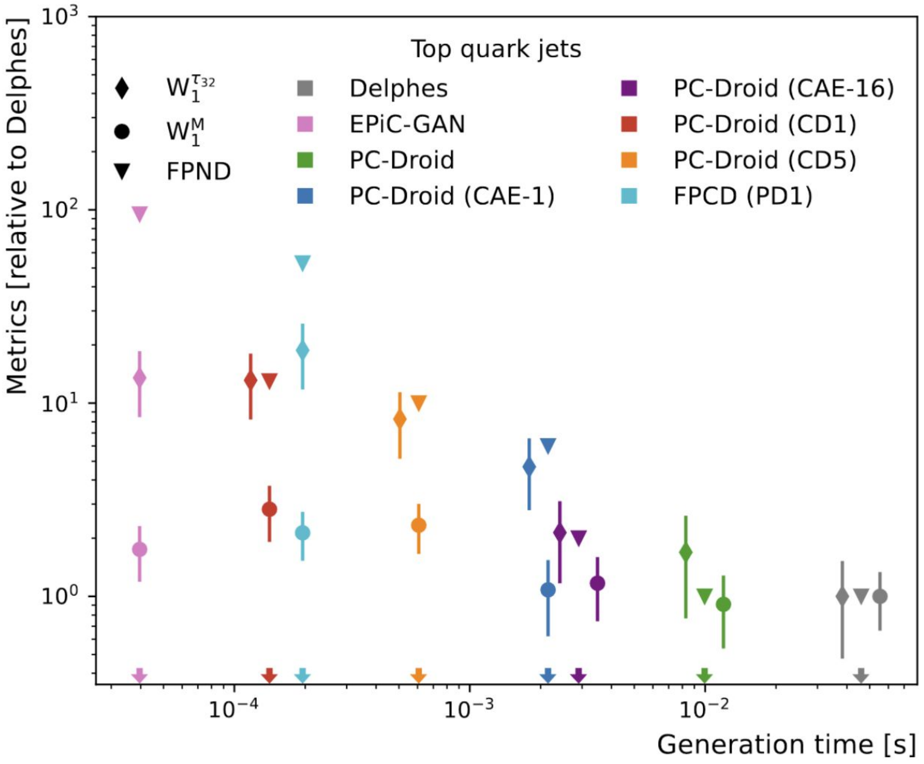




https://arxiv.org/abs/2307.06836

Time vs Fidelity Trade-Off

- Comparison with other generative models on 150 dataset
  - [FPCD]
  - [EPIC-GAN]

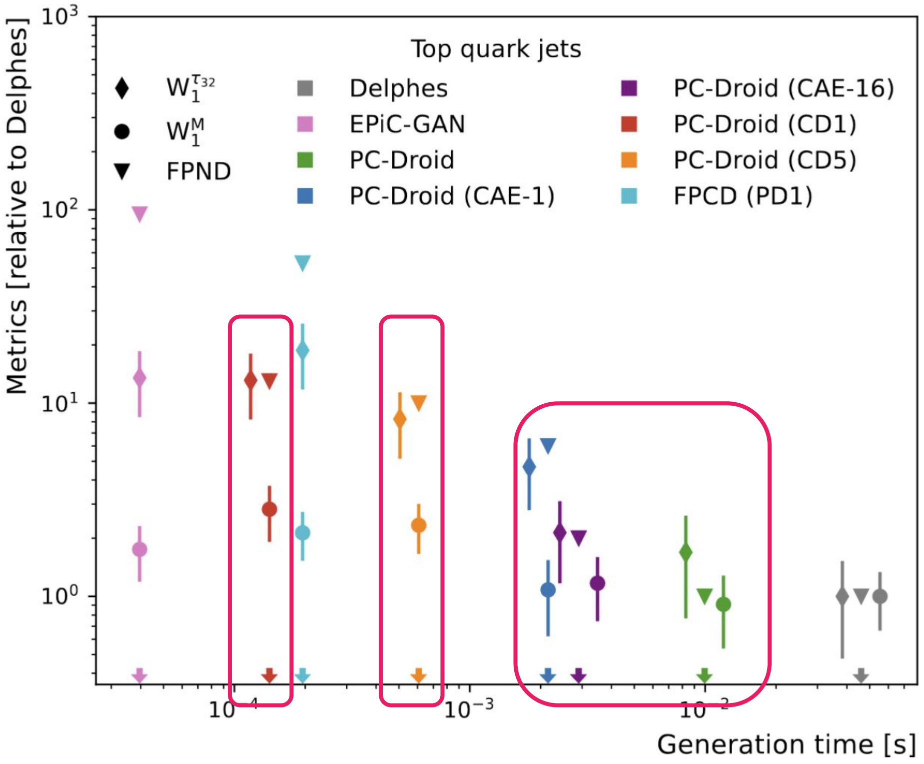


https://arxiv.org/abs/2307.06836

Time vs Fidelity Trade-Off

- Comparison with other generative models on 150 dataset
  - FPCD
  - EPIC-GAN

- PC-Droid performance on higher end is now close to ideal and 5 times faster
- Can sacrifice fidelity to get up to 100 times faster



https://arxiv.org/abs/2307.06836

Conclusion

- Introduced diffusion models into HEP for point cloud generation with PC-JeDI
- Significantly improved quality with PC-Droid
- At time of writing PC-Droid is SOTA for jet generation in most established metrics
- Looked at all models in terms of time-vs-quality trade off
- We are now looking at new ways to use such models beyond fast-sim (Next talk!)

Thank You

Backup

Denoising Learning Objective

Approximating the **score function** with a network is **impossible**

$$\min_\theta \underbrace{\mathbb{E}_{t\sim U(0,1)}}_{\text{time}} \underbrace{\mathbb{E}_{\boldsymbol{x}_t\sim p(\boldsymbol{x}_t)}}_{\text{diffused data}} \|\underbrace{\boldsymbol{s}_\theta(\boldsymbol{x}_t, t)}_{\text{neural network}} - \underbrace{\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)}_{\text{score of diffused data}}\|^2$$

Denoising Learning Objective

Approximating the **score function** with a network is **impossible**

$$\min_{\theta} \mathbb{E}_{t \sim U(0,1)} \mathbb{E}_{\boldsymbol{x}_t \sim p(\boldsymbol{x}_t)} \|\boldsymbol{s}_\theta(\boldsymbol{x}_t, t) - \nabla_{\boldsymbol{x}_t} \log \underline{p(\boldsymbol{x}_t)}\|^2$$

marginal diffused densities are intractable

Denoising Learning Objective

Approximating the **score function** with a network is **impossible**

$$\min_{\theta} \mathbb{E}_{t\sim U(0,1)} \mathbb{E}_{\boldsymbol{x}_t\sim p(\boldsymbol{x}_t)} \left\| \boldsymbol{s}_\theta(\boldsymbol{x}_t, t) - \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t) \right\|^2$$

Instead we look at the diffusion process of a **single sample $\mathbf{x_0}$**

$$\min_{\theta} \underbrace{\mathbb{E}_{t\sim U(0,1)}}_{\text{time}} \underbrace{\mathbb{E}_{\boldsymbol{x}_0\sim p(\boldsymbol{x}_0)}}_{\text{data sample}} \underbrace{\mathbb{E}_{\boldsymbol{x}_t\sim p(\boldsymbol{x}_t|\boldsymbol{x}_0)}}_{\text{diffused sample}} \left\| \boldsymbol{s}_\theta(\boldsymbol{x}_t, t) - \underbrace{\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{x}_0)}_{\text{score of diffused sample}} \right\|^2$$

Denoising Learning Objective

Approximating the **score function** with a network is **impossible**

$$\min_\theta \mathbb{E}_{t\sim U(0,1)}\,\mathbb{E}_{\boldsymbol{x}_t\sim p(\boldsymbol{x}_t)}\,\|\boldsymbol{s}_\theta(\boldsymbol{x}_t,t)-\nabla_{\boldsymbol{x}_t}\log p(\boldsymbol{x}_t)\|^2$$

Instead we look at the diffusion process of a **single sample $\mathbf{x}_0$**

$$\min_\theta \mathbb{E}_{t\sim U(0,1)}\,\mathbb{E}_{\boldsymbol{x}_0\sim p(\boldsymbol{x}_0)}\,\mathbb{E}_{\boldsymbol{x}_t\sim p(\boldsymbol{x}_t|\boldsymbol{x}_0)}\,\|\boldsymbol{s}_\theta(\boldsymbol{x}_t,t)-\nabla_{\boldsymbol{x}_t}\log \underline{p(\boldsymbol{x}_t|\boldsymbol{x}_0)}\|^2$$

This change is allowed because after expectations

$$\boldsymbol{s}_\theta(\boldsymbol{x}_t,t)\sim\nabla_{\boldsymbol{x}_t}\log p(\boldsymbol{x}_t)$$

Denoising Learning Objective

Approximating the **score function** with a network is **impossible**

$$\min_\theta \mathbb{E}_{t\sim U(0,1)}\,\mathbb{E}_{\boldsymbol{x}_t\sim p(\boldsymbol{x}_t)}\,\|\boldsymbol{s}_\theta(\boldsymbol{x}_t,t)-\nabla_{\boldsymbol{x}_t}\log p(\boldsymbol{x}_t)\|^2$$

Instead we look at the diffusion process of a **single sample $\mathbf{x}_0$**

$$\min_\theta \mathbb{E}_{t\sim U(0,1)}\,\mathbb{E}_{\boldsymbol{x}_0\sim p(\boldsymbol{x}_0)}\,\mathbb{E}_{\boldsymbol{x}_t\sim p(\boldsymbol{x}_t|\boldsymbol{x}_0)}\,\|\boldsymbol{s}_\theta(\boldsymbol{x}_t,t)-\nabla_{\boldsymbol{x}_t}\log \underline{p(\boldsymbol{x}_t|\boldsymbol{x}_0)}\|^2$$

This change is <u>allowed</u> because after expectations

$$\boldsymbol{s}_\theta(\boldsymbol{x}_t,t)\sim\nabla_{\boldsymbol{x}_t}\log p(\boldsymbol{x}_t)$$

This change is <u>useful</u> because the conditional density is tractable

$$p(\boldsymbol{x}_t|\boldsymbol{x}_0)=\mathcal{N}(\boldsymbol{x}_t;\gamma(t)\boldsymbol{x}_0,\sigma(t)^2\,\boldsymbol{I})$$

Denoising Learning Objective

Old Learning Objective

$$\min_\theta \mathbb{E}_{t\sim U(0,1)}\, \mathbb{E}_{\boldsymbol{x}_0\sim p(\boldsymbol{x}_0)}\, \mathbb{E}_{\boldsymbol{x}_t\sim p(\boldsymbol{x}_t|\boldsymbol{x}_0)}\, \|\boldsymbol{s}_\theta(\boldsymbol{x}_t, t) - \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{x}_0)\|^2$$

Conditional Density:

$$p(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \gamma(t)\boldsymbol{x}_0, \sigma(t)^2\, \boldsymbol{I})$$

Diffused Sample Score:

$$\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{x}_0) = \nabla_{\boldsymbol{x}_t} \frac{(\boldsymbol{x}_t - \gamma(t)\boldsymbol{x}_0)^2}{2\sigma(t)^2} = -\frac{\boldsymbol{x}_t - \gamma(t)\boldsymbol{x}_0}{\sigma(t)^2} = -\frac{\gamma(t)\boldsymbol{x}_0 + \sigma(t)\boldsymbol{\epsilon} - \gamma(t)\boldsymbol{x}_0}{\sigma(t)^2} = -\frac{\boldsymbol{\epsilon}}{\sigma(t)}$$

Neural Network Parameterisation:

$$\hat{\boldsymbol{\epsilon}}_\theta(\boldsymbol{x}_t, t) = -\sigma(t) s_\theta(\boldsymbol{x}_t, t)$$

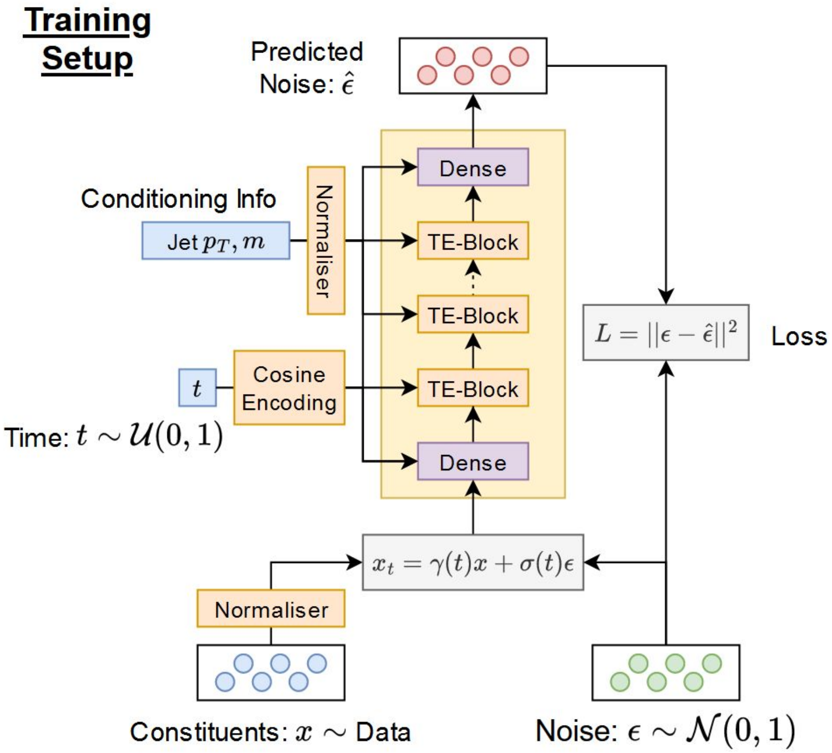
New Learning objective:

$$\min_\theta \mathbb{E}_{t\sim U(0,1)}\, \mathbb{E}_{\boldsymbol{x}_0\sim p(\boldsymbol{x}_0)}\, \mathbb{E}_{\boldsymbol{\epsilon}\sim \mathcal{N}(\boldsymbol{0},\boldsymbol{I})}\, \frac{1}{\sigma(t)^2} \|\hat{\boldsymbol{\epsilon}}_\theta(\boldsymbol{x}_t, t) - \boldsymbol{\epsilon}\|^2$$

PC-Jedi: Paper 1

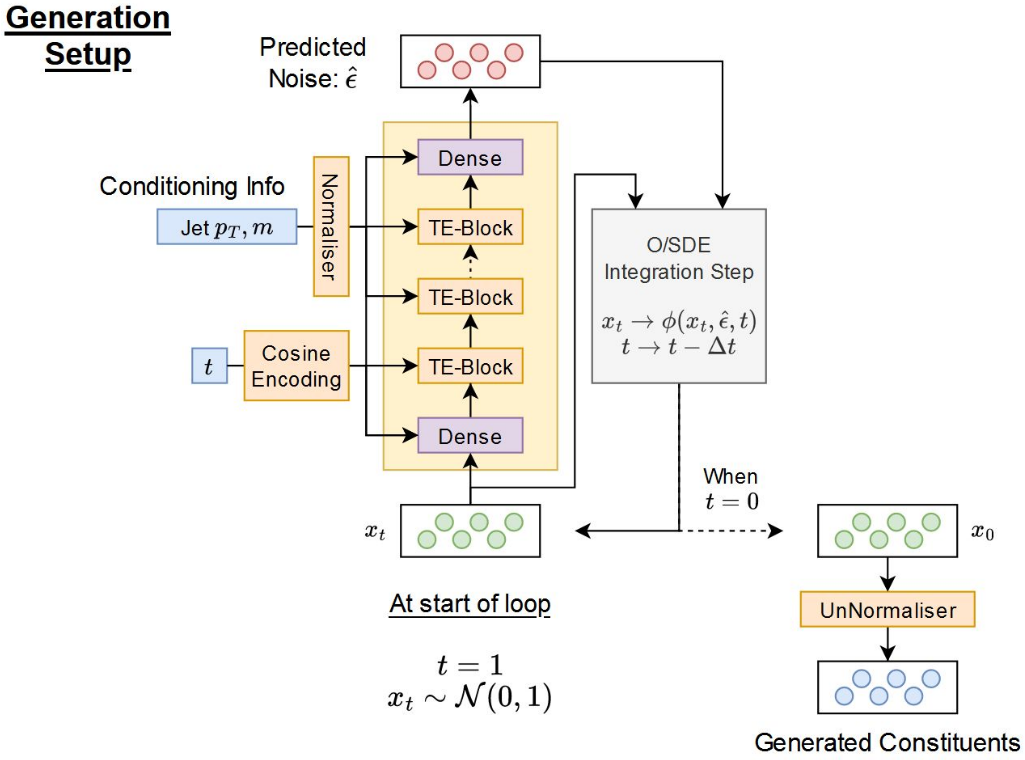
PC-Jedi Setup

- Based on a transformer
- Trained **separate models** for gluon and top
- Network always aware of current **timestep**
- Conditional generation based on jet $\mathbf{p_T}$ and **mass**



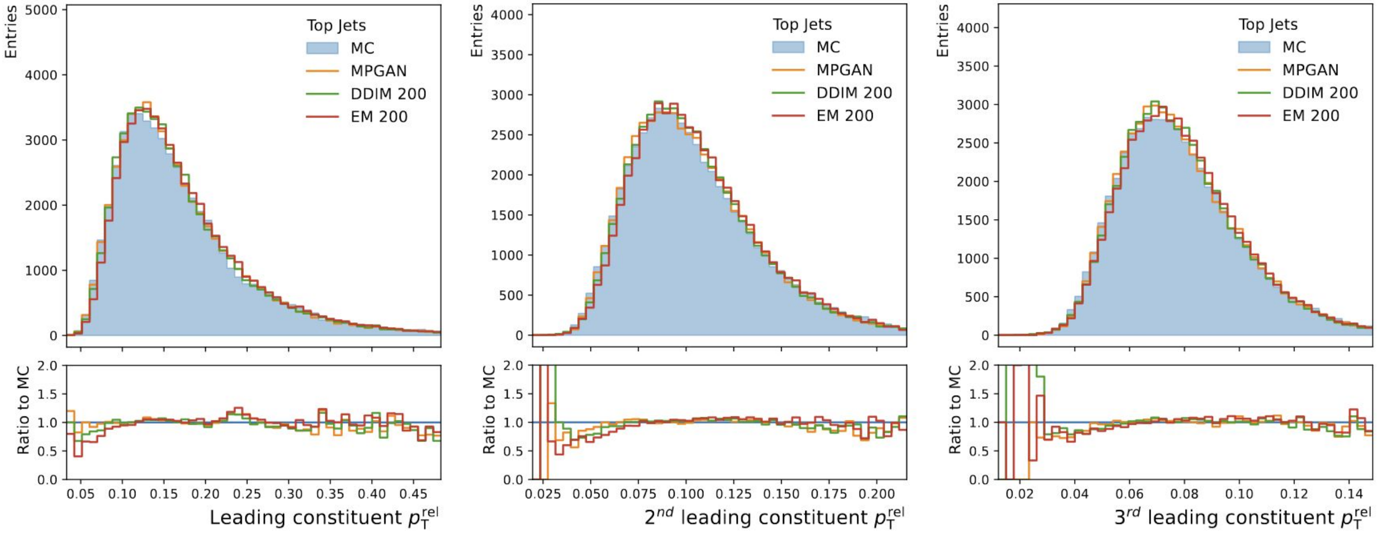
PC-Jedi Setup

- For generation we tested:
  - **Euler**
  - **E**uler-**M**aruyama (SDE)
  - **RK4**
  - **DDIM**



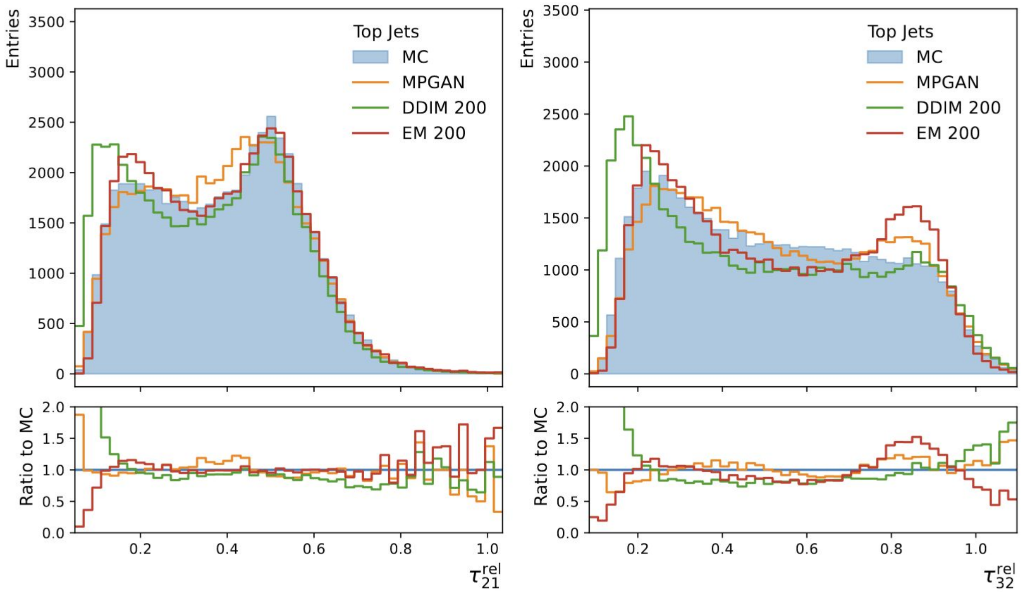
PC-Jedi Results

- Model was competitive to SOTA MPGAN



PC-Jedi Results

- Struggled recreating substructure variables for top jets



PC-Droid: Paper 2

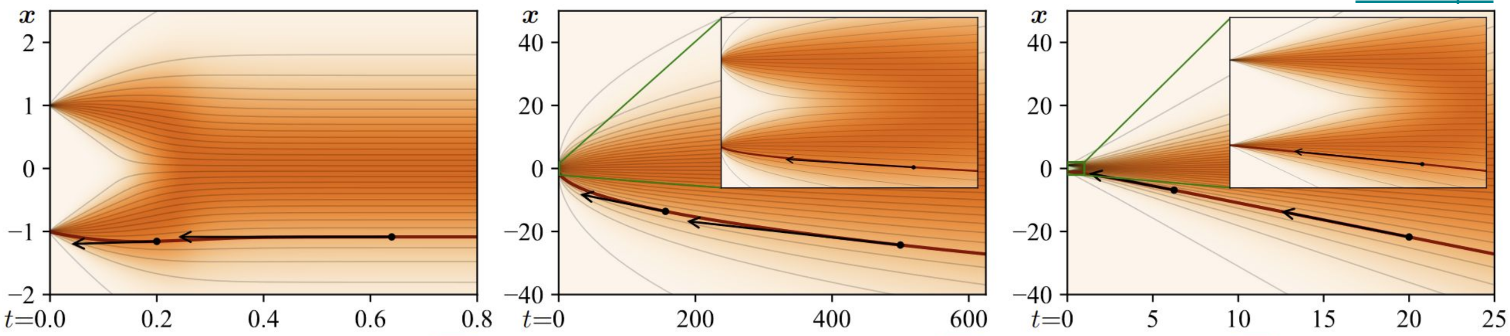
Improvements with PC-Droid

1. Change to EDM setup with preprocessing and sigma sampling

   SDE: $\mathrm{d}x_t = \sqrt{2t}\,\mathrm{d}w$

   ODE: $\mathrm{d}x_t = -t\nabla_x \log p(x;t)\,\mathrm{d}t$

[EDM Paper](#)

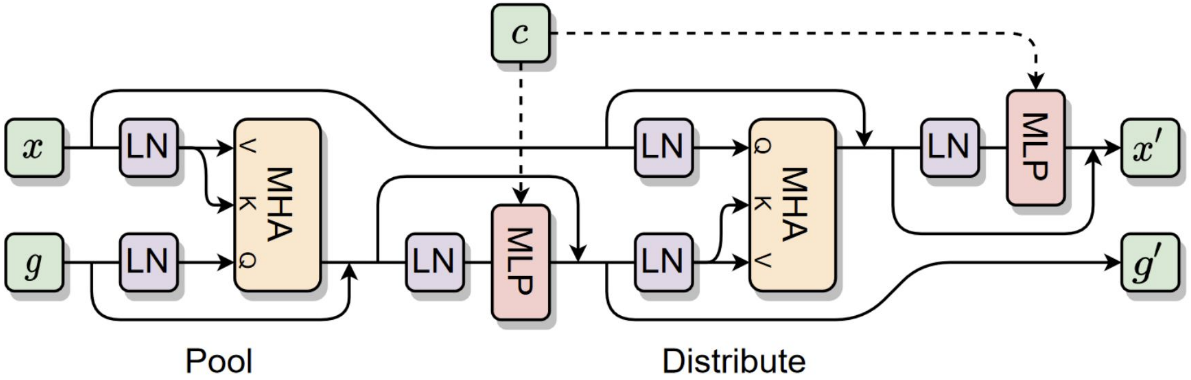


(a) Variance preserving ODE [49] (b) Variance exploding ODE [49] (c) DDIM [47] / Our ODE

Improvements with PC-Droid

2. Increase number of constituents from 30 to 150

- Introduced new network type: Cross Attention Encoder
- Bipartite graph between point cloud and collection of global tokens
  - Number of global tokens is a hyperparameter (M)
  - O(NM) computations compared to $O(N^2)$ of standard transformer



Improvements with PC-Droid

3. Access to new types of integration solvers designed for diffusion process

- Compatible with k-diffusion package