Generative modelling for Particle-Clouds with Discrete Features using Markov Jump Processes

Darius A. Faroughy

THE STATE UNIVERSITY OF NEW JERSEY
RUTGERS

ML4Jets 4-8 Nov 2024, Paris

- DAF, 2411.XXXXX [HEP-PH]
- DAF, Cesar Ojeda, Manfred Opper 250X.XXXXX [ML]

Introduction

- Elementary particles are characterized by several fundamental quantities:

$$\text{mass} \in \mathbb{R}^+$$
$$p_\mu = (p_x, p_y, p_z, E)$$

continuous

$$\text{spin} \in \{0, 1/2, 1, ...\}$$
$$\text{charge} \in \{-1, 0, +1\}$$
$$\text{flavor} \in \{e, \mu, \tau, ...\}$$
$$\text{parity} \in \{-1, +1\}$$
$$...$$

discrete quantum numbers

.

→ understood through *symmetries* and representation theory.

- At High Energy colliders the detectors are capable of infering some of these discrete quantities.

@ LHC:

$$\text{reconstructed particle} = (p_T, \eta, \phi) \otimes \{\gamma, e^-, e^+, \mu^-, \mu^+, \text{h}^0, \text{h}^-, \text{h}^+\} \otimes ...$$

kinematics *particle-id*

→ data points live in a “hybrid” feature space that includes both continuous and discrete features.

In this talk:

- I present a new generative modelling framework for datasets with hybrid feature spaces.
- I’ll focus on **jets** represented as **particle-clouds**, i.e. unordered set of constituents.

Particle-clouds datasets

- In recent years, a lot of interest in Deep Generative Models for **jet constituents** represented as **particle-clouds**.

  → Permutation equivariant architectures e.g. deep sets, transformers

  → Almost all overlook discrete features, focus exclusively on kinematics.

  → Previous open datasets did not include discrete features, e.g. JetNet

MP-GAN [2106.11535]
PC-JeDI [2303.05376]
EPiC-GAN [2301.08128]
FPCD [2304.01266]
MDMA-GAN [2305.15254]
EPIC-FM [2310.00049]
...

SOTA

Recent exceptions:
Birk et al. [2312.00123] Araz et al. [2410.22421]

Kansal et al. [2106.11535]

Particle-clouds datasets

- In recent years, a lot of interest in Deep Generative Models for **jet constituents** represented as **particle-clouds**.

  → Permutation equivariant architectures e.g. deep sets, transformers

  → Almost all overlook discrete features, focus exclusively on kinematics.

  → Previous open datasets did not include discrete features, e.g. JetNet

MP-GAN [2106.11535]
PC-JeDI [2303.05376]
EPiC-GAN [2301.08128]
FPCD [2304.01266]
MDMA-GAN [2305.15254]
EPIC-FM [2310.00049]
...

SOTA

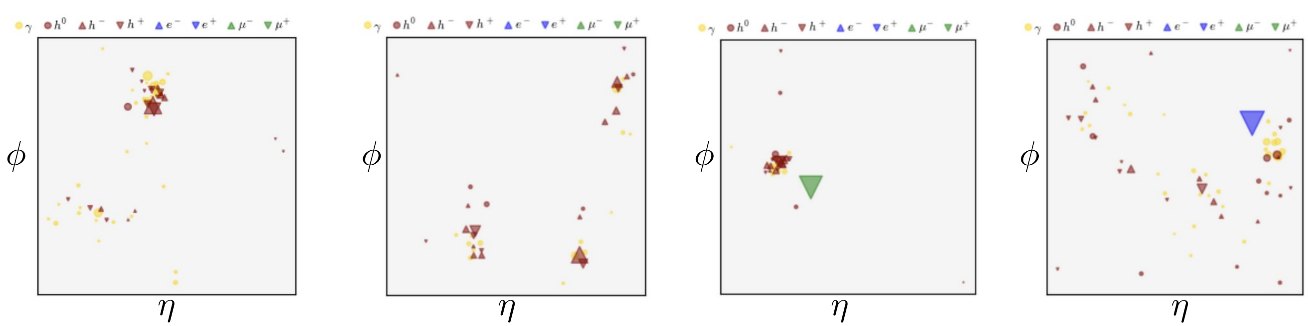
Recent exceptions:
Birk et al. [2312.00123] Araz et al. [2410.22421]

Kansal et al. [2106.11535]

- JetClass: Qu et al. [2022]

$$\text{jet constituent} = (p_T, \eta, \phi) \otimes (d_0, d_z) \otimes \{\gamma, e, \mu, \text{had}\} \otimes \{-, 0, +\}$$

kinematics *trajectories* *particle-id* *electric charge*



QCD $t \to bqq$ $t \to b\ell\nu$ $h \to WW \to qq\ell\nu$

- AspenOpenJets (soon to appear): 180M jets from CMS open data! (~QCD jets) Ian Peng [ML4Jets talk] for more details

Dynamical Generative models

- The model we propose is a dynamical generative model like Diffusion and Flow-Matching.

- These models learn a continuous-time dynamics that evolves source data at t = 0 (e.g. noise) into your target data at t = 1.

- Generation: simulate the EOM with the source data as initial condition using numerical ODE/SDE solvers.

  → output at t = 1 is taken as your generated sample.

Dynamical Generative models

- The model we propose is a dynamical generative model like Diffusion and Flow-Matching.

- These models <u>learn</u> a continuous-time dynamics that evolves source data at t = 0 (e.g. noise) into your target data at t = 1.

- <u>Generation</u>: simulate the EOM with the source data as initial condition using ODE/SDE solvers.

  → output at t = 1 is taken as your generated sample.

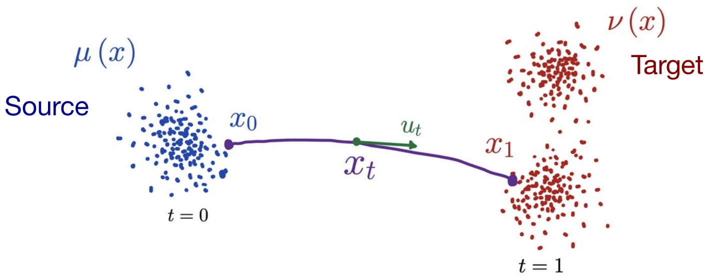
- Flow-Matching: Lipman et al. [ICLR 2023]

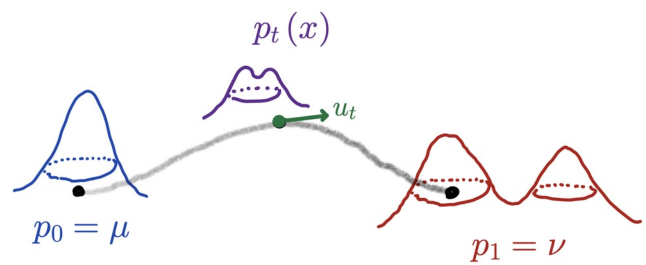
Flow (ODE):

$$\mathrm{d}\mathbf{x}_t = u_t(\mathbf{x}_t)\,\mathrm{d}t$$

Probability Flow (PDE)

$$\partial_t p_t(\mathbf{x}) = -\nabla_{\mathbf{x}} \cdot \left[u_t(\mathbf{x}) p_t(\mathbf{x})\right]$$





→ Regress the velocity vector-field u_t^θ with a NN by matching a *conditional process*: $\tilde{u}_t(\mathbf{x}_t|\mathbf{x}_0, \mathbf{x}_1)$

$\Downarrow$

Conditional Flow-Matching loss (MSE):

$$\mathcal{L}_{\mathrm{CFM}}(\theta) \equiv \mathbb{E}_{t,\mathbf{x}_0,\mathbf{x}_1,\mathbf{x}_t} \left\| u_t^\theta(\mathbf{x}_t) - \tilde{u}_t(\mathbf{x}_t|\mathbf{x}_0, \mathbf{x}_1) \right\|^2 \quad \Longleftarrow \quad \tilde{u}_t = \mathbf{x}_1 - \mathbf{x}_0$$

e.g. Cond-OT

Diffusion and Flow-Matching can't **directly** handle data with discrete features.

- Trick: one-hot encode and float the discrete features. Birk et al. [2312.00123] Araz et al. [2410.22421]

$$\mathrm{pid} \in \{\gamma, e, \mu, \mathrm{had}\} \xrightarrow{\text{one-hot}} \begin{bmatrix} 1.0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1.0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1.0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1.0 \end{bmatrix}$$

PID "probability"

- After generation, apply a hard assigment to get a unique PID feature using **`ArgMax()`**.

→ Works very well!

Diffusion and Flow-Matching can't **directly** handle data with discrete features.

- Trick: one-hot encode and float the discrete features. Birk et al. [2312.00123] Araz et al. [2410.22421]

$$\text{pid} \in \{\gamma, e, \mu, \text{had}\} \xrightarrow{\text{one-hot}} \begin{bmatrix} 1.0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1.0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1.0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1.0 \end{bmatrix}$$

PID "probability"

- After generation, apply a hard assigment to get a unique PID feature using **`ArgMax()`**.

→ Works very well!

- We propose an alternative approach:

→ Cook up a generative model that preserves the data's representation.

$$(\mathbf{x}, k) \in \mathcal{D}_{\text{continuous}} \otimes \mathcal{D}_{\text{discrete}} \qquad \begin{cases} \mathcal{D}_{\text{continuous}} = \mathbb{R}^D & \textit{vector space} \\ \mathcal{D}_{\text{discrete}} = \{1, 2, ..., K\} & \textit{countable "state" space} \end{cases}$$

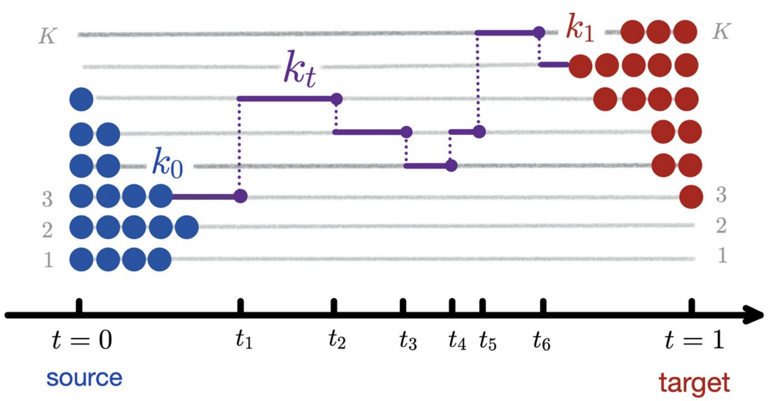
Our model consitsts of two dynamical generative models:

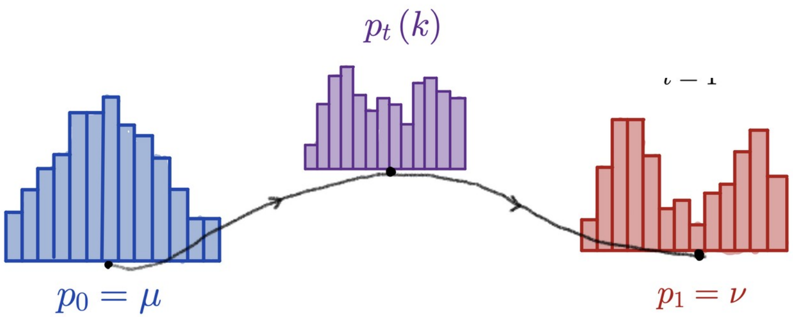
$\{\mathbf{x}_t\}_{t\in[0,1]}$ ← handled by Flow-Matching (could also use diffusion).

$\{k_t\}_{t\in[0,1]}$ ← handled by a continuous-time ***Markov Jump Process.***

Discrete dynamics: Markov Jump Processes

- Sequence of discrete jumps $\{k_t\}_{t\in[0,1]}$ $k_t \in \{1, 2, ..., K\}$ *K possible "states"*





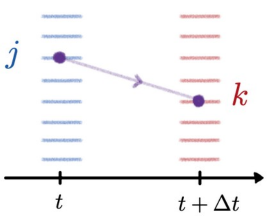
- Probability flow: *Kolmogorov Equation*

$$\partial_t\, p_t(k) = \sum_{j\neq k} \left[W_t(k|j)p_t(k) - W_t(j|k)p_t(j)\right]$$

in-flow prob *out-flow prob*

Jump Rate matrix: $W_t \in \mathbb{R}^{K\times K}$

→ discrete analog of velocity field



instantaneous transition probability

- Generation: simulate the Kolmogorov eq. with *tau-leaping method*

[Gillespie 2001]
[Campbell et al. 2205.14987]

Conditional Jump Process

- We define a jump process conditioned on the data: $\begin{cases} \tilde{p}_t(k\,|k_0, k_1) & \textit{Conditional Probability path} \\ \tilde{W}_t(k|j, k_0, k_1) & \textit{Conditional Rate} \end{cases}$

Unconditionals are recovered by marginalizing:

$$p_t(k) = \sum_{k_0, k_1} \tilde{p}_t(k|k_0, k_1)\, \mu(k_0)\, \nu(k_1)$$

$$W_t(k|j) = \sum_{k_0, k_1} \tilde{W}_t(k|j, k_0, k_1)\, q_t(k_0, k_1|j)$$

Posterior probability

$$q_t(k_0, k_1|k) = \frac{\tilde{p}_t(k|k_0, k_1)\, \mu(k_0)\, \nu(k_1)}{p_t(k)}$$

Conditional Jump Process

- We define a jump process conditioned on the data: $\begin{cases} \tilde{p}_t(k\,|k_0, k_1) & \textit{Conditional Probability path} \\ \tilde{W}_t(k|j, k_0, k_1) & \textit{Conditional Rate} \end{cases}$

Unconditionals are recovered by marginalizing:

$$p_t(k) = \sum_{k_0,k_1} \tilde{p}_t(k|k_0, k_1)\, \mu(k_0)\, \nu(k_1)$$

$$W_t(k|j) = \sum_{k_0,k_1} \tilde{W}_t(k|j, k_0, k_1)\, q_t(k_0, k_1|j)$$

Posterior probability

$$q_t(k_0, k_1|k) = \frac{\tilde{p}_t(k|k_0, k_1)\, \mu(k_0)\, \nu(k_1)}{p_t(k)}$$

- To generate data we will need to compute the rate matrix $W_t(k|j)$:
  - 1. Choose tractable conditional process: *Random Telegraph Process* (defined next slide)
  - 2. Approximate the posterior probability with a Neural Network.

  q_t^θ $\leftarrow$ time-dependent state classifer trained with cross-entropy loss!

- **Conditional Jump Process** objective:

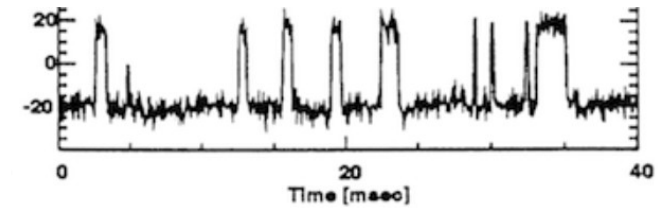
$$\mathcal{L}_{\text{CJP}} = \mathbb{E}_{t,k_0,k_1,k_t} \left[\log q_t^\theta(k_0, k_1|k_t)\right]$$

Expectation over: $\begin{cases} t \sim \mathcal{U}(0,1) \\ k_0, k_1 \sim \mu, \nu \\ k_t \sim \tilde{p}_t \end{cases}$

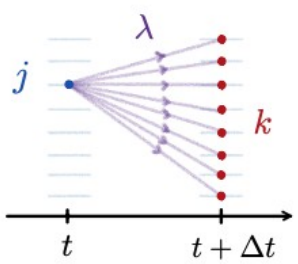
Random Telegraph Process

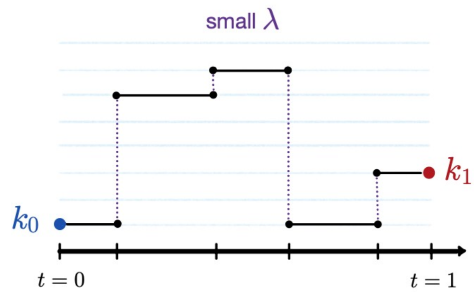
- Stochastic process that models random bit-flips in 2-state sytems

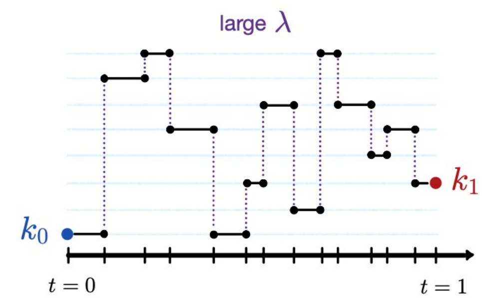
  e.g. burst noise in semi-conductors



- Multivariate generalization: $k \in \{1, ..., K\}$



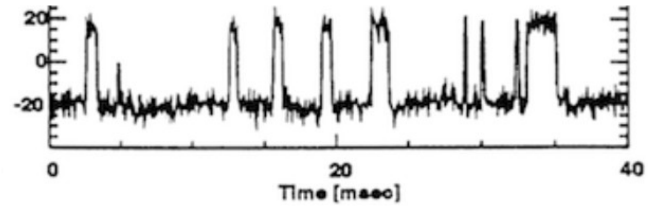




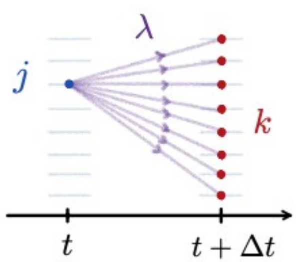
Random Telegraph Process

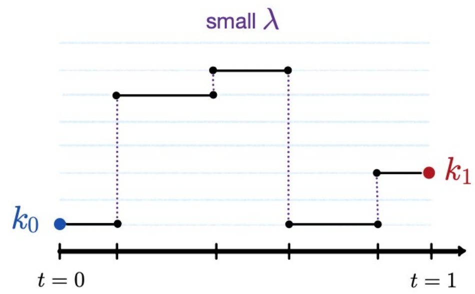
- Stochastic process that models random bit-flips in 2-state sytems

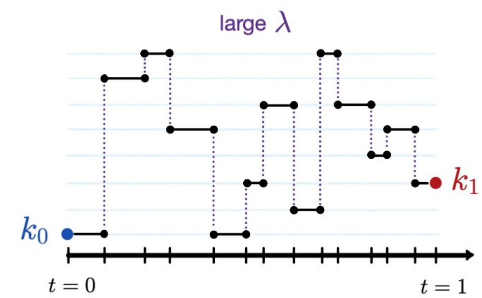
  e.g. burst noise in semi-conductors



- Multivariate generalization: $k \in \{1, ..., K\}$







Kolmogorov equation for *transition probability density*: $p_{t|s}(i|j) \equiv \mathrm{Prob}(k_t = i | k_s = j)\,, \quad t > s$

$$\partial_t \tilde{p}_{t|s}(k|j) = \lambda \left[1 - K\, \tilde{p}_{t|s}(k|j)\right]$$

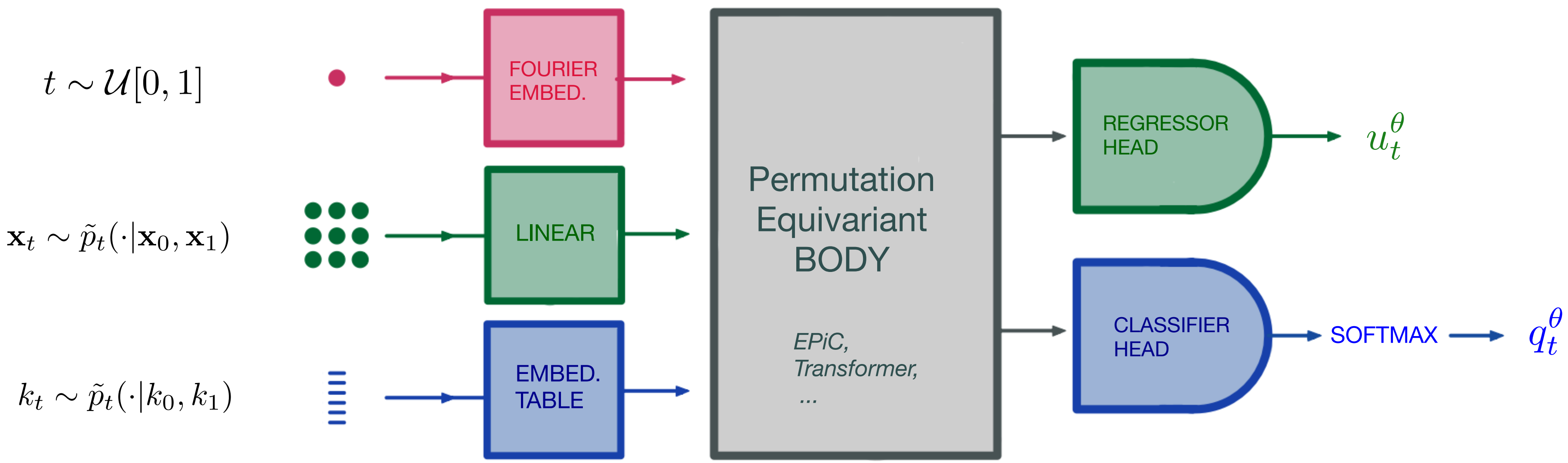
Analytical solution: $\tilde{p}_{t|s}(k|j) = \frac{1}{K} - \left(\frac{1}{K} + \delta_{kj}\right) e^{-\lambda K(t-s)}$

We obtain an analytic expression for the conditional rate:

$$\Longrightarrow \qquad \tilde{W}_t(k|j, k_0, k_1) = 1 + K\, \frac{\omega_t}{1-\omega_t}\, \delta_{k k_1} + \omega_t\, \delta_{j k_1} \qquad \text{with} \qquad \omega_t \equiv e^{-\lambda K(1-t)}$$

A Generative Model for Hybrid Data

- Hybrid data: $(\mathbf{x}, k) \in \mathbb{R}^D \otimes \{0, 1, ..., K\}$
- Hybrid generative model: Conditional Flow-Matching + Conditional Jump Process
- We parametrize a single NN to learn the vector field and the posteriors: $h_t^\theta = u_t^\theta \otimes q_t^\theta$
- Architecture:



- Hybrid Loss: $\mathcal{L}_{\text{hybrid}}(h^\theta; \alpha) = \mathcal{L}_{\text{CFM}}(u_t^\theta) + \alpha\, \mathcal{L}_{\text{CJP}}(q_t^\theta)$

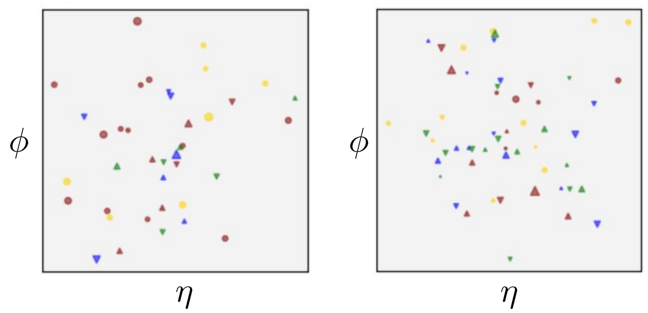
MSE *Cross-Entropy*

Generating particle-clouds with PID

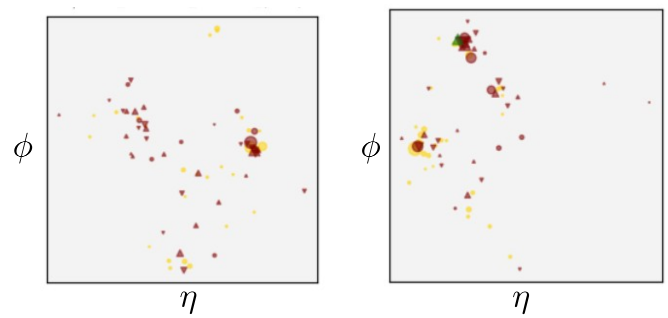
- Dataset: JetClass

$$\mathcal{D} = (p_T, \eta_{\text{rel}}, \phi_{\text{rel}}) \otimes \text{pid}$$ where $\text{pid} \in \{\gamma, h^0, h^-, h^+, e^-, e^+, \mu^-, \mu^+\}$ $\quad N = 8$ states

Source: Noise $\mathcal{N}(0, \mathbb{1}) \otimes \text{Cat}\,(p = 1/8)$



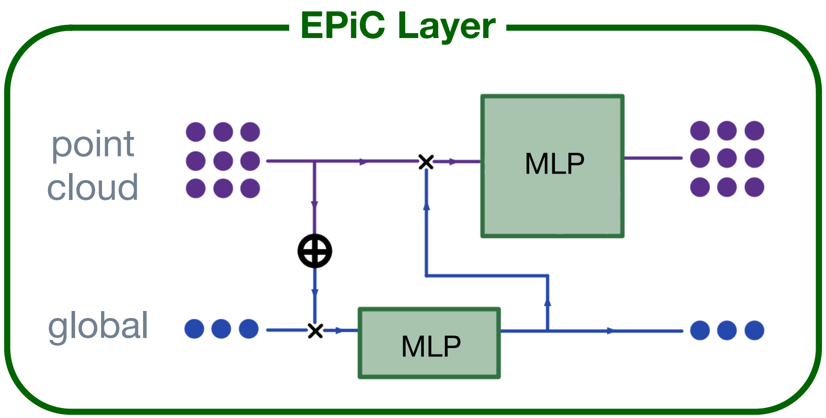
Target: hadronic top-jets



- Permutation equivariant network:

$$h^\theta = \tilde{u}_t^\theta \otimes q_t^\theta$$ EPiC Network

EPIC-FM [2310.00049]



EPiC-GAN [2301.08128]

- Model hyper-parameters:

  continuous: CFM parameter $\sigma = 10^{-4}$

  discrete: telegraph rate $\lambda = 1/8 = 0.125$

- Training:

  - 300k jets (270k train + 30k validation)
  - 10 EPiC Layers (# params: ~ 850K)
  - 500 epochs

- Generation:

  - 1000 time-steps
  - Euler method (continuous)
  - Tau-Leaping (discrete)

- Results:

- Training:

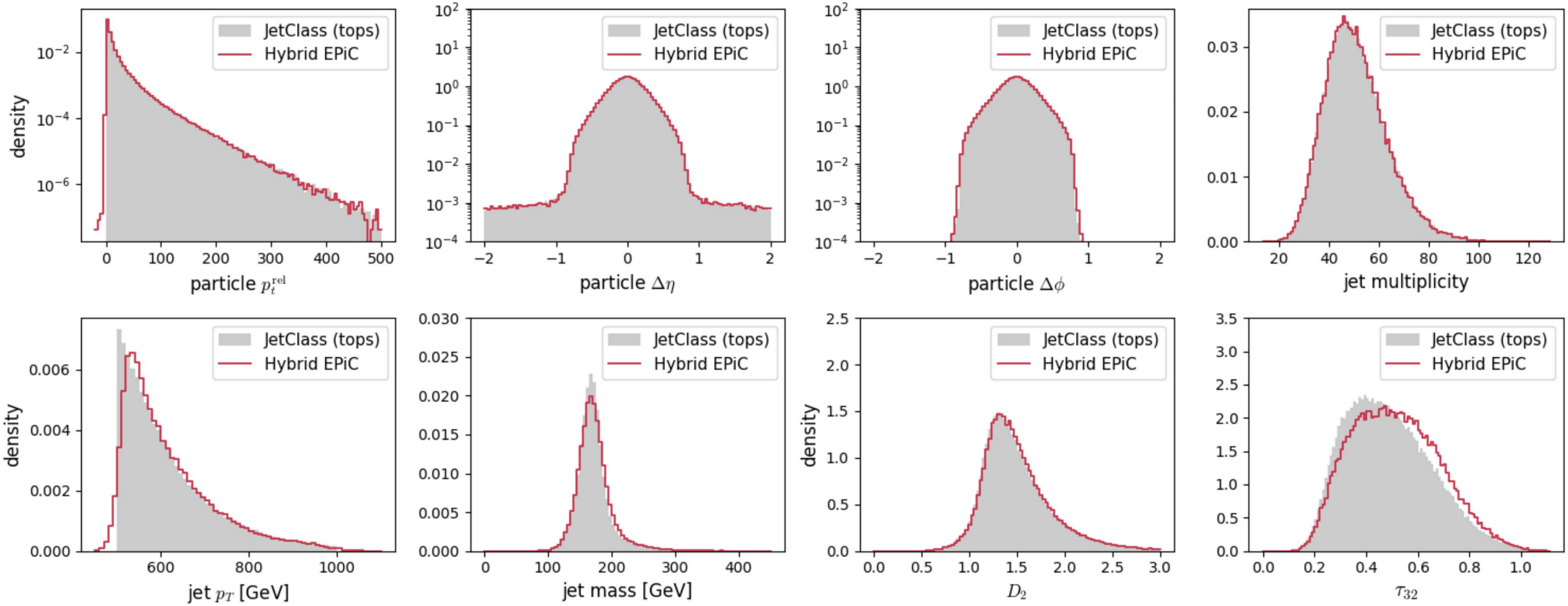
- 300k jets (270k train + 30k validation)
- 10 EPiC Layers (# params: ~ 850K)
- 500 epochs

- Generation:

- 1000 time-steps
- Euler method (continuous)
- Tau-Leaping (discrete)

- Results:

Continuous Features



Results consistent with EPIC-FM

- Training:

- 300k jets (270k train + 30k validation)
- 10 EPiC Layers (# params: ~ 850K)
- 500 epochs

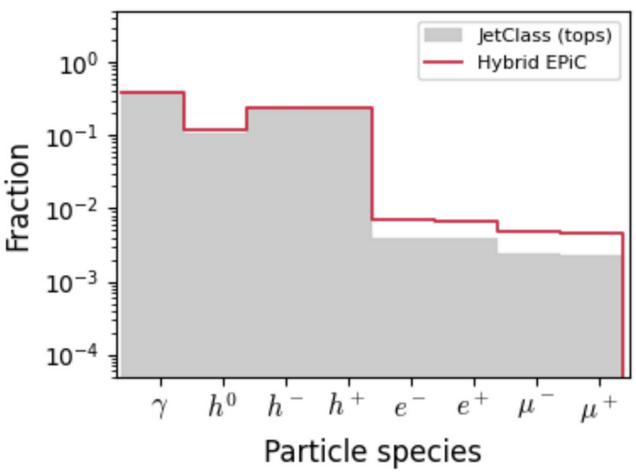
- Generation:

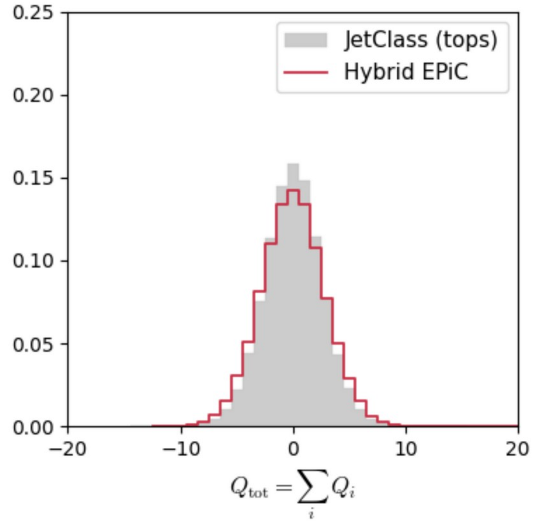
- 1000 time-steps
- Euler method (continuous)
- Tau-Leaping (discrete)

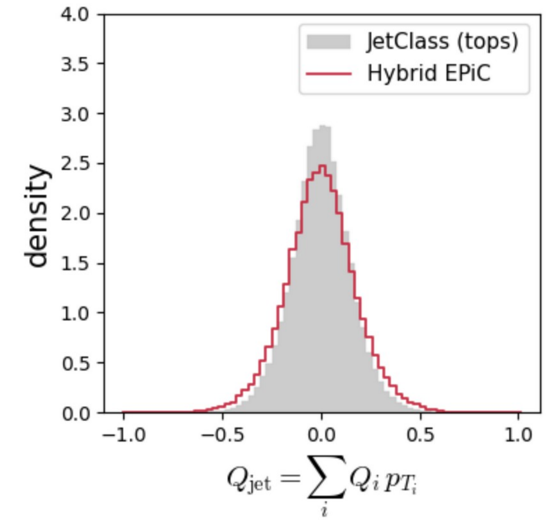
- Results:

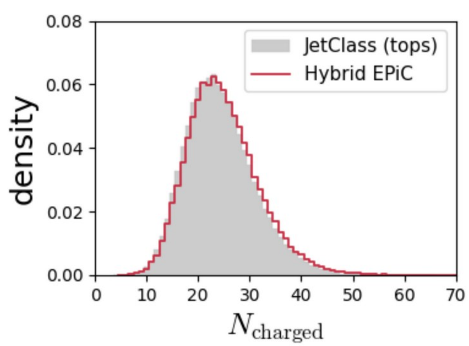
Discrete features

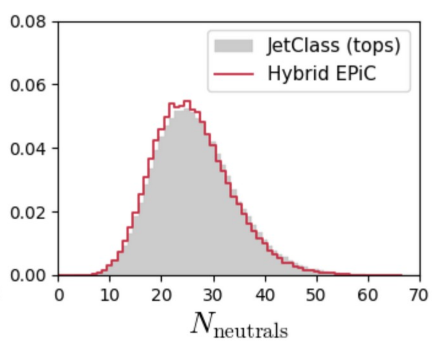
Hybrid feature

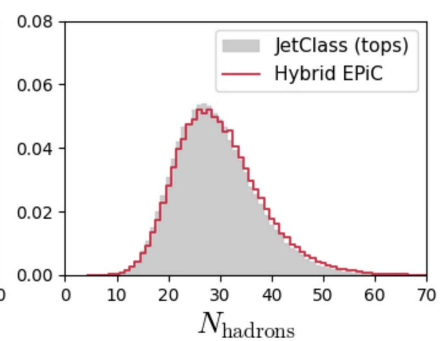












Conclusions

- In this talk we've presented a new generative model for particle-clouds with discrete features.

  Based on training two dynamical generative models in parallel:

  Conditional Flow-Matching for kinematics → (p_T, η, ϕ)

  Conditional Jump Process for particle-id → $\{\gamma, e^{\pm}, \mu^{\pm}, \mathrm{h}^0, \mathrm{h}^{\pm}\}$

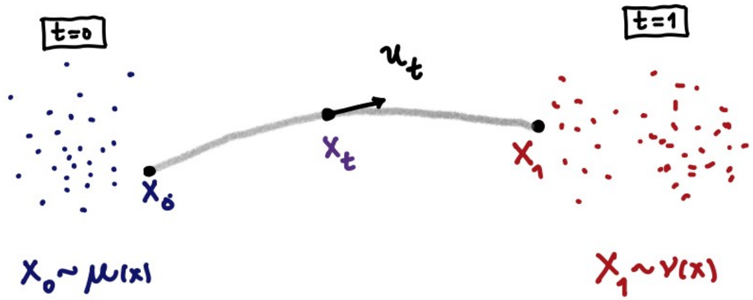
- We showed that our model can give good results for the **kinematics** and **particle-id** distributions for JetClass.

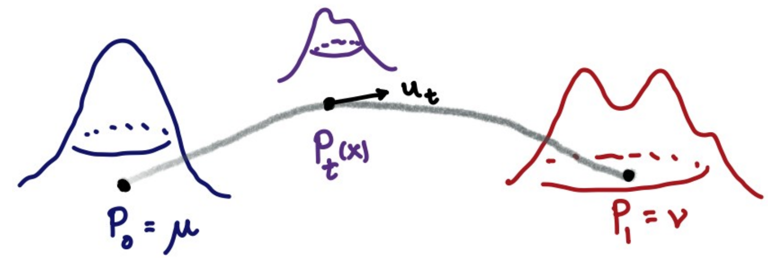
For now proof of concept...

- We will look into other dynamics besides the Telegraph process
- We need to optimize our training (e.g. scan over hyperparamas)
- Train on larger datastets.
- We still need apple-to-apple comparison with other methods.

THANKS!

Continuous dynamics: Flow-Matching





Flow (ODE)

$$\mathrm{d}\mathbf{x}_t = u_t(\mathbf{x}_t)\,\mathrm{d}t$$

Probability Flow (PDE)

$$\partial_t p_t(\mathbf{x}) = -\nabla_{\mathbf{x}} \cdot [u_t(\mathbf{x}) p_t(\mathbf{x})]$$

- Consider a *conditional dynamics*: $\begin{cases} \tilde{p}_t(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) & \textit{conditional probability path} \\ \tilde{u}_t(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) & \textit{conditional velocity field} \end{cases}$

Gaussian probability paths:

$$\tilde{p}_t(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) = \mathcal{N}(\mathbf{x}|\mu_t,\, \sigma_t^2) \qquad \begin{cases} \mu_t = t\,\mathbf{x}_0 + (1-t)\,\mathbf{x}_1 \\ \sigma_t = \sigma = \text{small const.} \end{cases} \implies \tilde{u}_t = \mathbf{x}_1 - \mathbf{x}_0$$

- **Conditional Flow-Matching** objective:

$$\mathcal{L}_{\mathrm{CFM}}(\theta) \;\equiv\; \mathbb{E}_{t,\mathbf{x}_0,\mathbf{x}_1,\mathbf{x}_t} \left\| u_t^\theta(\mathbf{x}_t) - \tilde{u}_t(\mathbf{x}_t|\mathbf{x}_0, \mathbf{x}_1) \right\|^2$$

Conditional Flows

- Define a process conditioned on the data: $\begin{cases} \tilde{p}_t(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) \\ \tilde{u}_t(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) \end{cases}$ *conditional probability path* / *conditional velocity field*

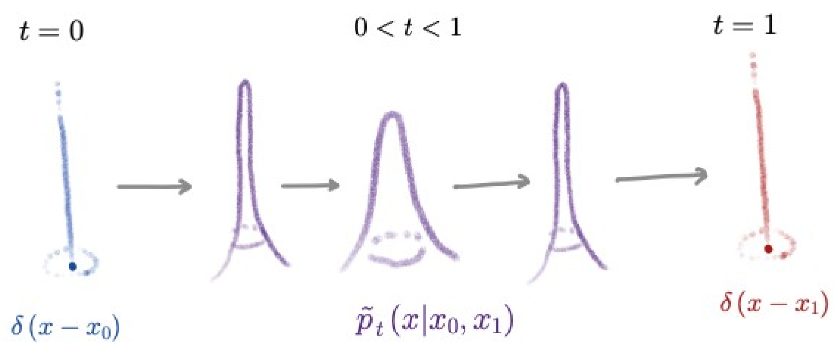
$$p_t(\mathbf{x}) = \int \mathrm{d}\mathbf{x}_0 \mathrm{d}\mathbf{x}_1 \, \tilde{p}_t(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) \, \mu(\mathbf{x}_0) \, \nu(\mathbf{x}_1)$$

$$\begin{cases} p_0 = \mu \quad \Longrightarrow \quad \tilde{p}_0(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) = \delta(\mathbf{x} - \mathbf{x}_0) \\ p_1 = \nu \quad \Longrightarrow \quad \tilde{p}_1(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) = \delta(\mathbf{x} - \mathbf{x}_1) \end{cases}$$

$$u_t(\mathbf{x}) = \int \mathrm{d}\mathbf{x}_0 \mathrm{d}\mathbf{x}_1 \, \tilde{u}_t(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) \, q_t(\mathbf{x}_0, \mathbf{x}_1|\mathbf{x})$$

Posterior probability:

$$q_t(\mathbf{x}_0, \mathbf{x}_1|\mathbf{x}) = \frac{\tilde{p}_t(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) \, \mu(\mathbf{x}_0) \, \nu(\mathbf{x}_1)}{p_t(\mathbf{x})}$$



Gaussian probability paths:

$$\tilde{p}_t(\mathbf{x}|\mathbf{x}_0, \mathbf{x}_1) = \mathcal{N}(\mathbf{x}|\mu_t, \, \sigma_t^2) \qquad \mu_t = \mu_t(\mathbf{x}_0, \mathbf{x}_1)$$

- **Conditional flow-matching** objective: $\nabla_\theta \mathcal{L}_{\mathrm{CFM}} = \nabla_\theta \mathcal{L}_{\mathrm{FM}}$

$$\mathcal{L}_{\mathrm{CFM}}(\theta) \equiv \mathbb{E}_{t, \mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_t} \left\| u_t^\theta(\mathbf{x}_t) - \tilde{u}_t(\mathbf{x}_t|\mathbf{x}_0, \mathbf{x}_1) \right\|^2$$

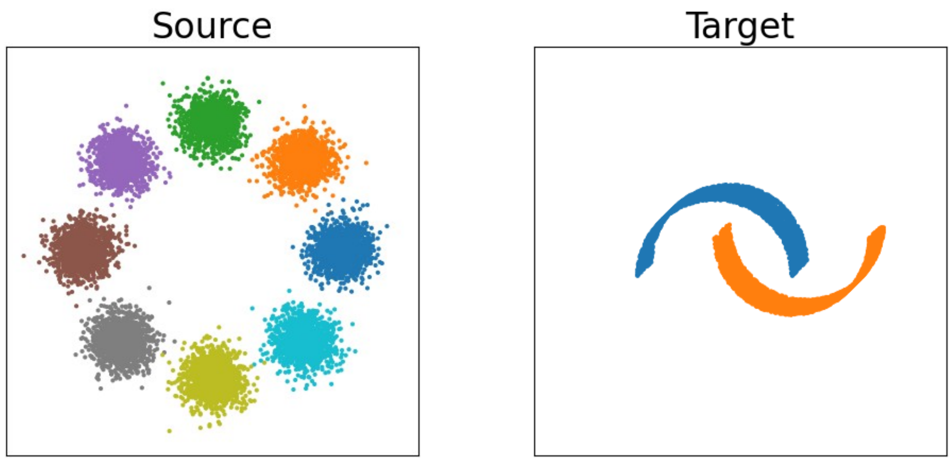
Expectation over: $\begin{cases} t \sim \mathcal{U}(0,1) \\ \mathbf{x}_0, \mathbf{x}_1 \sim \mu, \nu \\ \mathbf{x}_t \sim \tilde{p}_t(\cdot|\mathbf{x}_0, \mathbf{x}_1) \end{cases}$

1) Toy example for hybrid data:

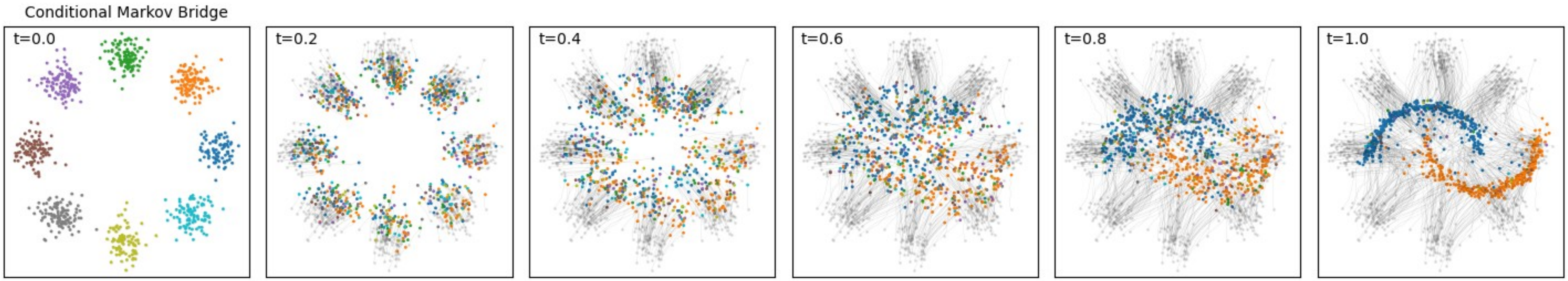
$$(x, y, \text{color}) \in \mathbb{R}^2 \otimes \{1, ..., 8\}$$

$h^\theta = \tilde{u}_t^\theta \otimes q_t^\theta$ MLP (3 layers)

$$\lambda_{\text{CJB}} = 1/N = 0.125$$
$$\sigma_{\text{CFM}} = 0.1$$
$$\alpha = 1$$



- Snapshots of source → target generation



EPiC Flow-Matching (EPiC-FM)

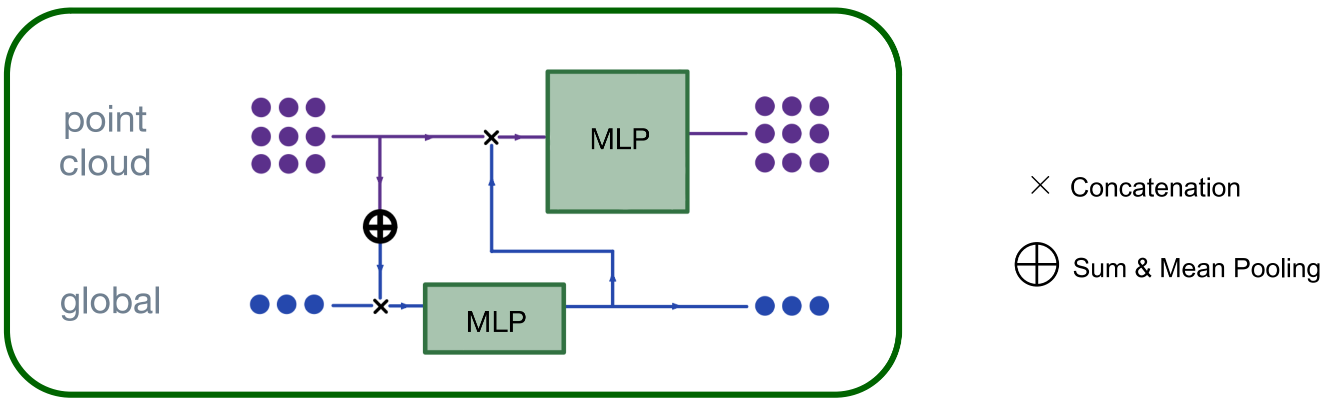
- Flow matching model: conditional optimal transport

$$u_t(X) = X_1 - X_0$$

$$\mathcal{L}(\theta) = \mathbb{E}||u_t^\theta(X_t) - (X_1 - X_0)||^2$$

Buhman, DAF et al [2310.00049]

EPiC Layer



EPiC Network: