Pay Attention to Mean Fields for Point Cloud Generation

ML4Jets - 6.10.2023

Artwork(s) by DALL – E · 3

Benno Käch[1], Isabell Melzer-Pellmann, Dirk Krücker, Moritz Scham, Simon Schnake

(1) Funded through Helmholtz AI grant, number ZT-I-PF-5-064

HELMHOLTZ AI

CLUSTER OF EXCELLENCE
QUANTUM UNIVERSE

DESY.

Generative Modelling for Detector Simulation

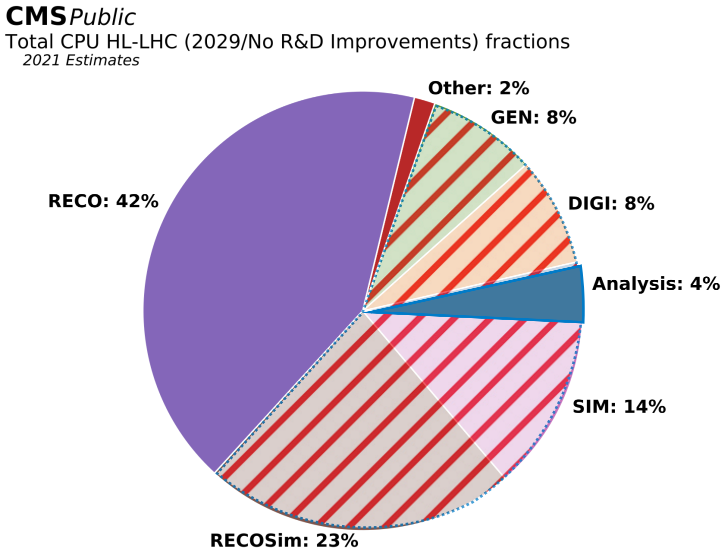
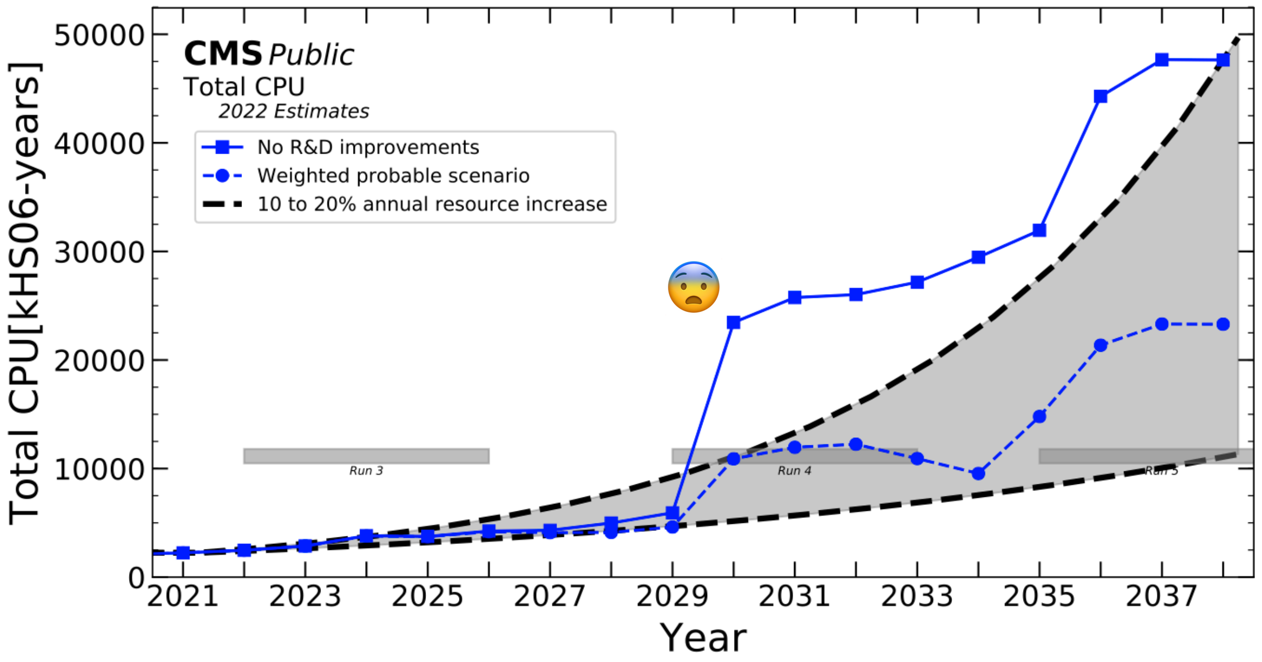


Figure from CMS Offline Computing Results

- Rely on experiment simulation in High Energy Physics (Digital Twin)
- Classically generated with Monte Carlo simulation
  → slow and computing intense
- Already > 50 % of computing budget
- Coming High Luminosity upgrades makes MC approach challenging



CMS Experiment

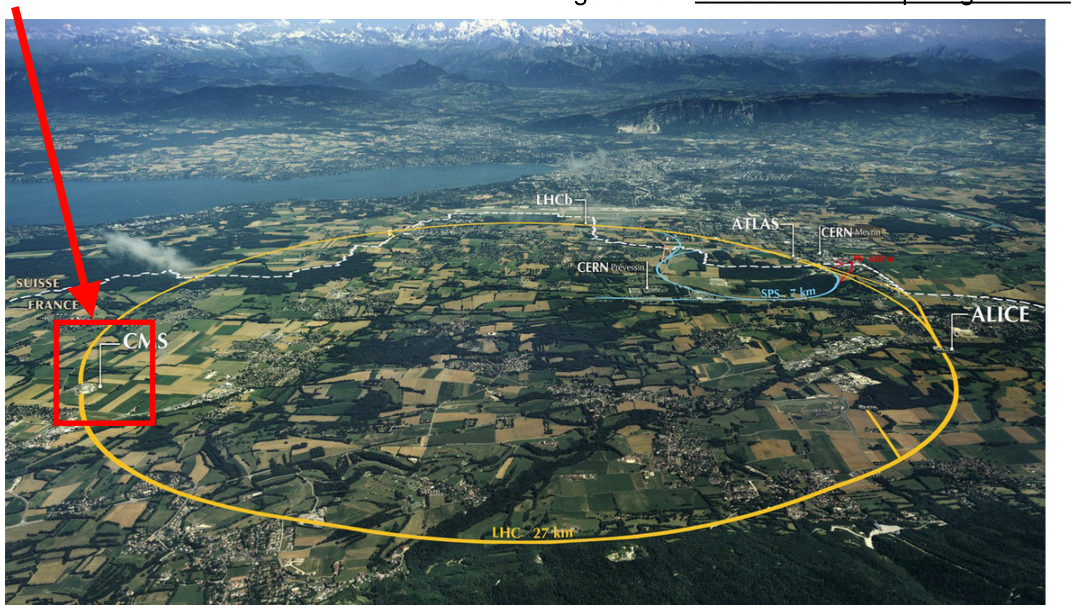


Figure from CMS Offline Computing Results

Generative Modelling for Detector Simulation

- Rely on experiment simulation in High Energy Phy
- Classically generated with Monte Carlo simulation
  → slow and computing intense
- Already > 50 % of computing budget
- Coming High Luminosity upgrades makes

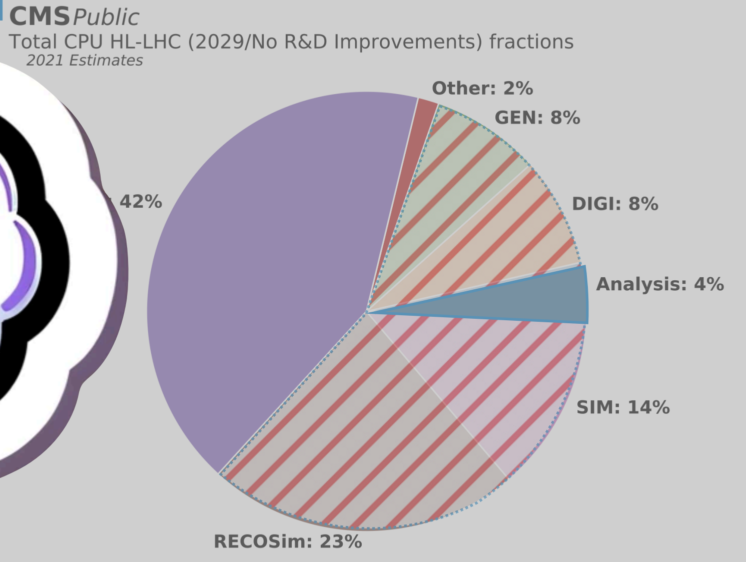
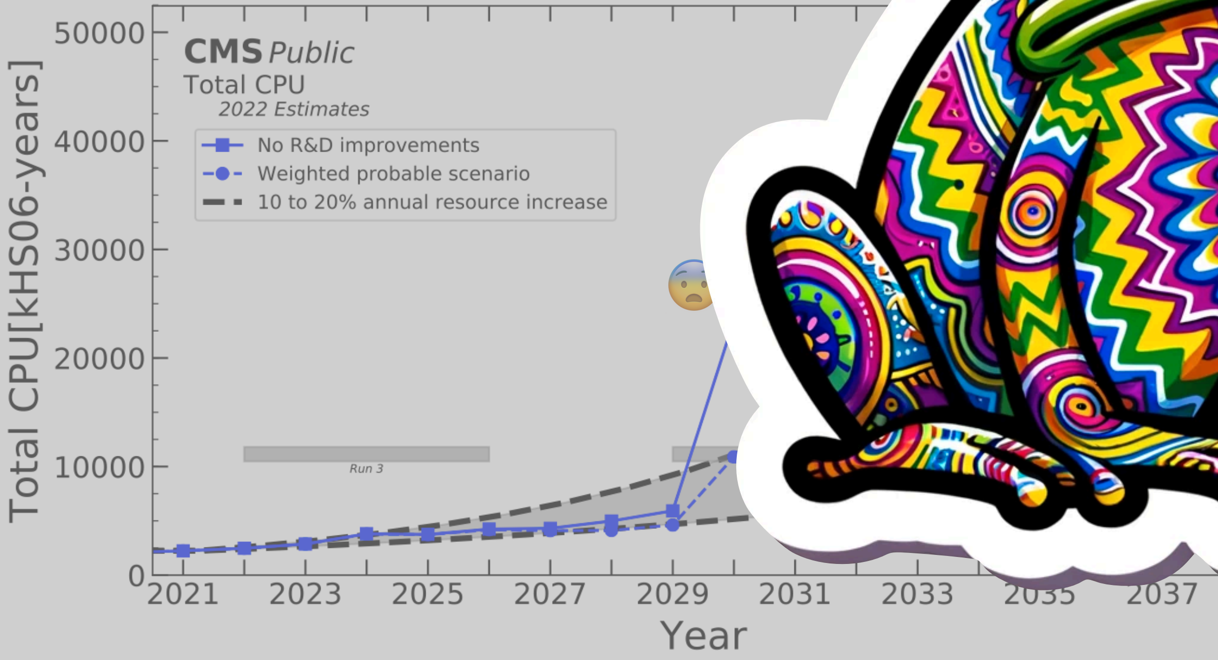


Figure from CMS Offline Computing Results



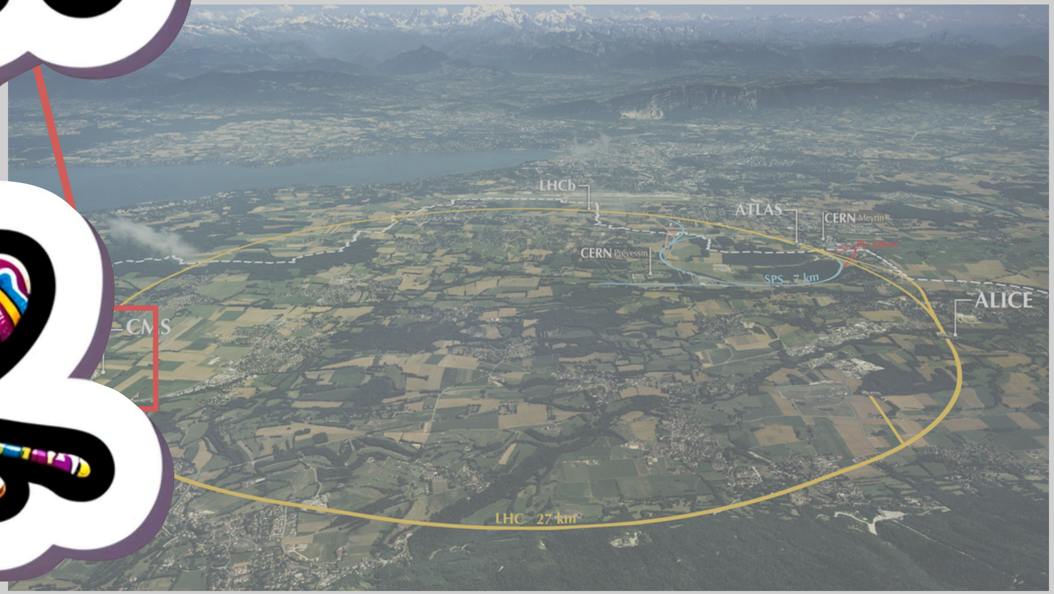


Figure from CMS Offline Computing Results

Point Clouds, their Symmetries

- Most natural representation of collider data
- Handles sparsity
- Detector independent
- Point clouds are permutation invariant → Use permutation-equivariant aggregations

[1]: Kansal et al., Particle Cloud Generation with Message Passing Generative Adversarial Networks
[2]: Krause et al., CaloChallenge

Point Clouds, their Symmetries and Requirements

- Most natural representation of collider data
- Handles sparsity
- Detector independent
- Point clouds are permutation invariant → Use permutation-equivariant aggregations

- Aggregations should scale **linearly** with point cloud cardinality [3]
  → otherwise memory goes 👋🏻 → already a problem at 150 points on one GPU
  → Showstopper for (Self-Attention-based) model I presented at last ML4Jets from scaling from 30 to 150 particles
- Datasets for point clouds: (JetNet [1], preprocessed CaloChallenge [2])

[1]: Kansal et al., Particle Cloud Generation with Message Passing Generative Adversarial Networks
[2]: Krause et al., CaloChallenge
[3]: Kaech et al., Point Cloud Generation using Transformer Encoders and Normalising Flows

Building a permutation equivariant model

- Universal approximation for functions from **Deep Sets** [1]: Sum pooling latent representation of particles:

$$z = \sum_{i=1}^{N} \Phi(x_i)$$

  - Disclaimer: latent space dimension $=$ max set cardinality $\times$ features per point
  - **EPiC**-GAN [2] by Buhmann et al. first to use equivariant architecture to obtain promising results on Jetnet150

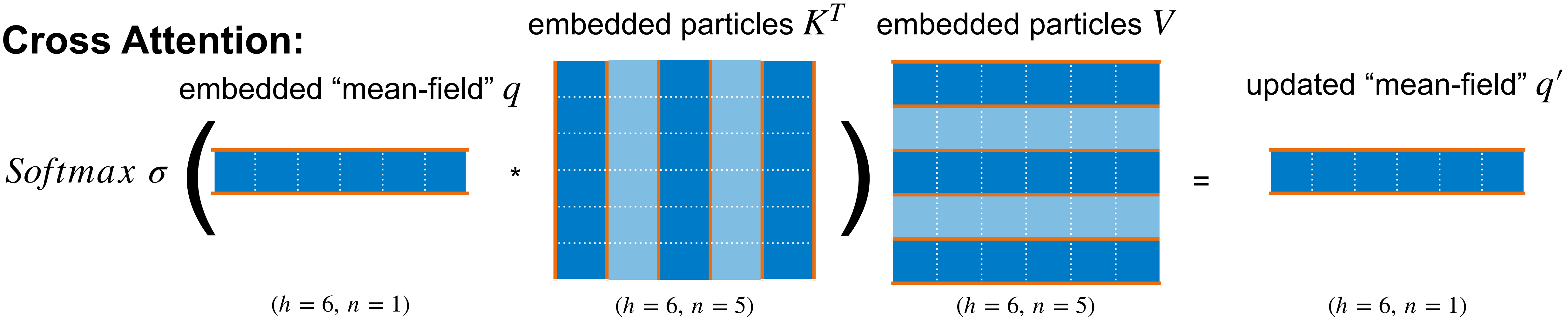
  $\rightarrow$ used combination of sum and mean pooling to update global latent state of jet given which particles are conditionally independent $\rightarrow$ scales $O(n)$

- But I really wanted to keep Attention! - Attention is permutation-equivariant, although most commonly known from (sequential) NLP

[1]: Wagstaff et al., On the Limitations of Representing Functions on Sets
[2]: Buhmann et al., EPiC-GAN: Equivariant Point Cloud Generation for Particle Jets

Cross Attention to the rescue

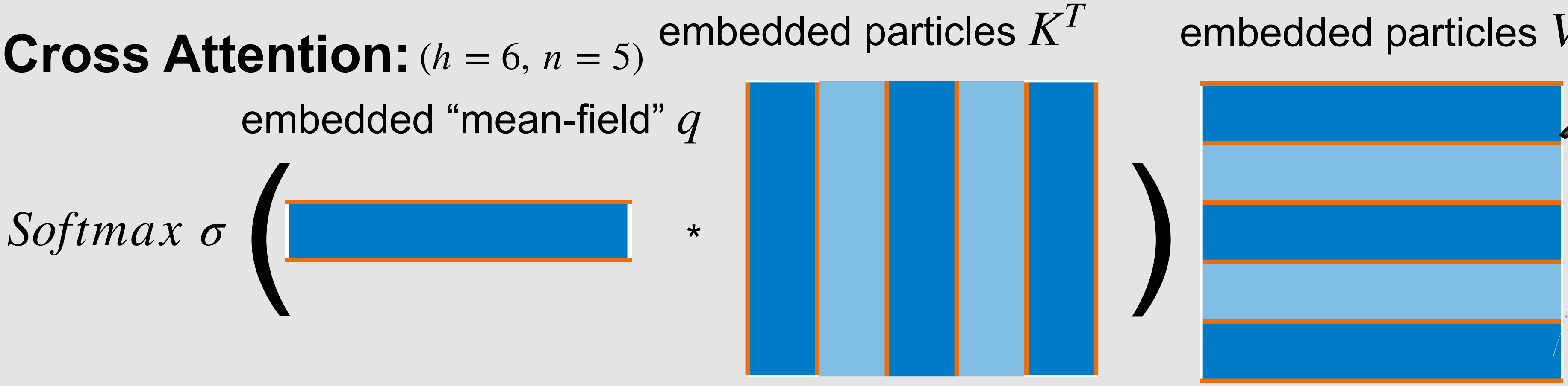
Cross Attention:



$$q' = \sigma\left(\frac{qK^T}{\sqrt{h}}\right) V$$

Similar to Sum-Pooling but more General

Cross Attention: $(h = 6,\ n = 5)$

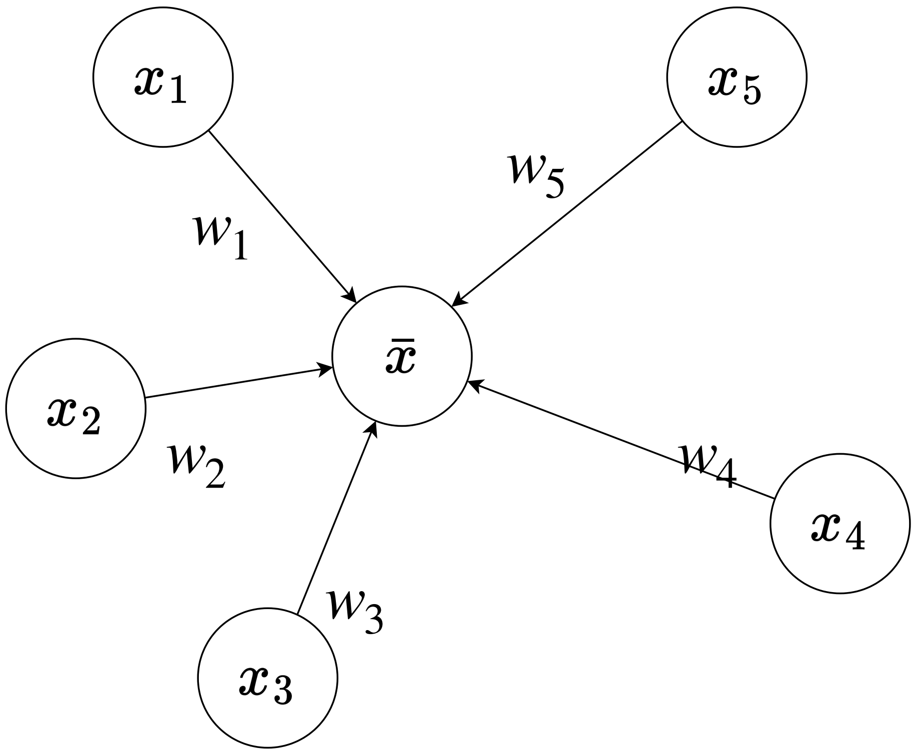


- “Mean-field” $q \in \mathbb{R}^{1\times h}$, h hidden dimension
- K: n embedded particles $K = (W_K x)^T$, $x \in \mathbb{R}^{n\times 4}$, $W_K \in \mathbb{R}^{h\times n}$
- V: n embedded particles, $V = W_V x$, $x \in \mathbb{R}^{n\times 4}$, $W_V \in \mathbb{R}^{h\times n}$

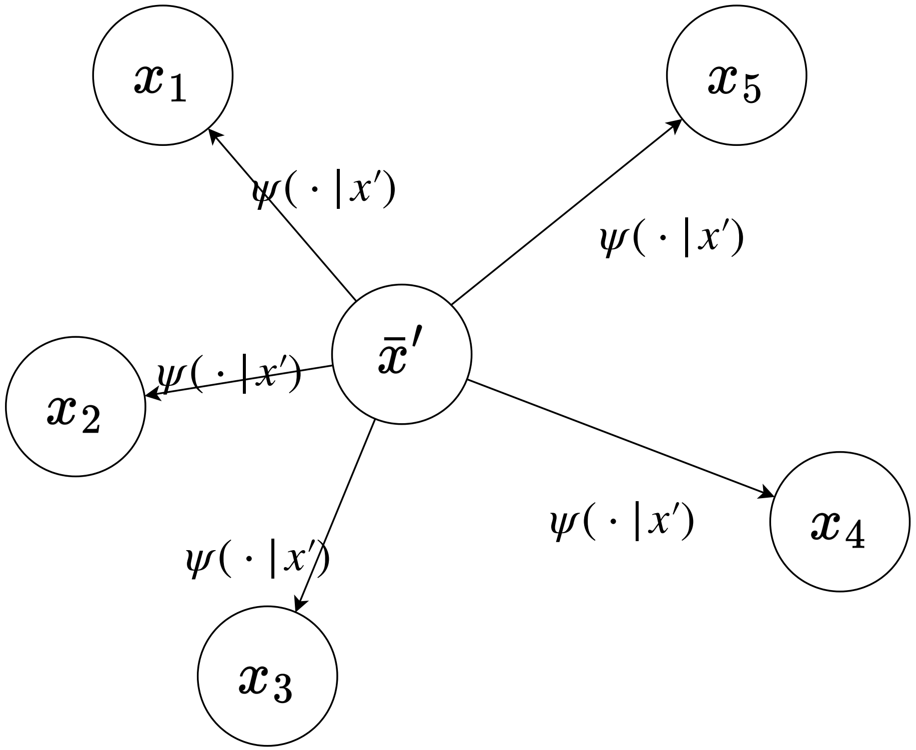
$$\bar{\mathbf{x}}' = \sigma\left((\mathbf{q} \cdot K)/\sqrt{h}\right) V = \sum_{i=1}^{n} w_i W_V \mathbf{x}_i$$

Mean-Field Aggregation

- Introduce "artificial" mean-field $\bar{x}$: proxy for particle-particle interaction
- Particles update mean-field via cross-attention, mean-field concatenated to particles
- Computation scales with $O(n)$



$\longrightarrow$



The Missing Piece

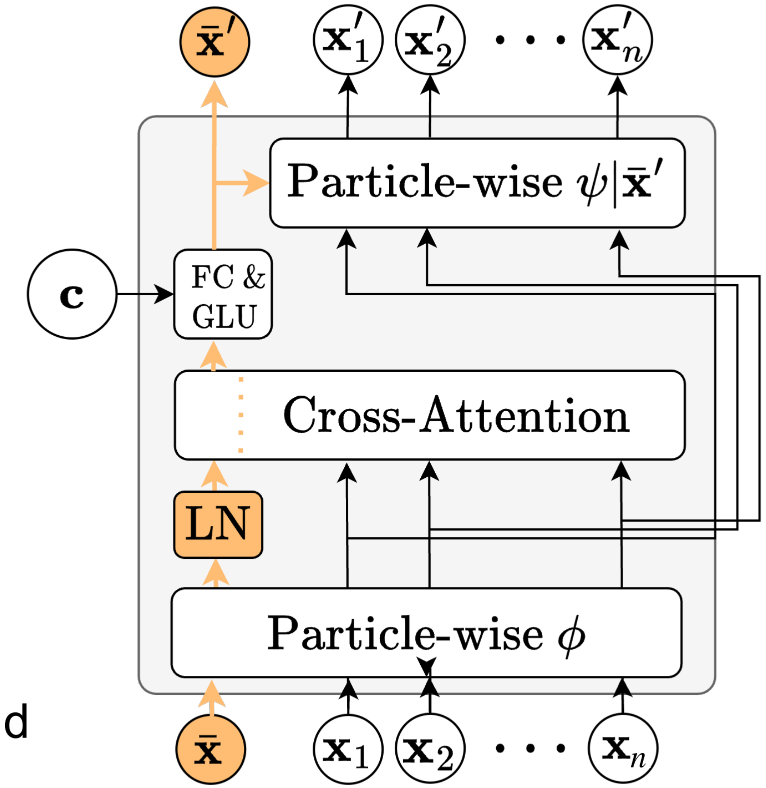
- Although having found a fancy aggregation results were still not acceptable for higher set cardinalities
- Model performs very well up to 50 particles - then stops working
- Why does EPiC-GAN work and my model does not? 😢
- Also is mean and sum pooling not very similar?

The Missing Piece

- Although having found a fancy aggregation results were still not acceptable for higher set cardinalities
- Model performs very well up to 50 particles - then stops working
- Why does EPiC-GAN work and my model does not? 🥲
- Also is mean and sum pooling not very similar?
  - Only difference: the **number of constituents** n!
  - Also: My model is **agnostic** of the number of constituents!

Main Architecture Block

- Architecture motivated by Transformer Encoder architecture used on JetNet 30 [1]
- IN: embedded particles $x_i \in \mathbb{R}^l$, embedded mean-field $\bar{x} \in \mathbb{R}^l$, OUT: embedded particles $x_i \in \mathbb{R}^l$, embedded mean-field $\bar{x} \in \mathbb{R}^l$
  1. Particle-wise ϕ mapping from latent dimension $\mathbb{R}^l \rightarrow \mathbb{R}^h$
  2. Layer Norm applied to mean-field
  3. Multi Headed Cross-Attention between mean-field and particles
  4. Cloud multiplicity & incoming energy (only for CaloChallenge) conditioned fully-connected layer updates mean-field
  5. Particle-wise FC ψ conditioned on mean-field updates particles
- **Permutation-equivariant**
- Not shown here: residual connection between in/out particles & mean-field



[1] Kaech et al., Point Cloud Generation using Transformer Encoders and Normalising Flows

GAN Training

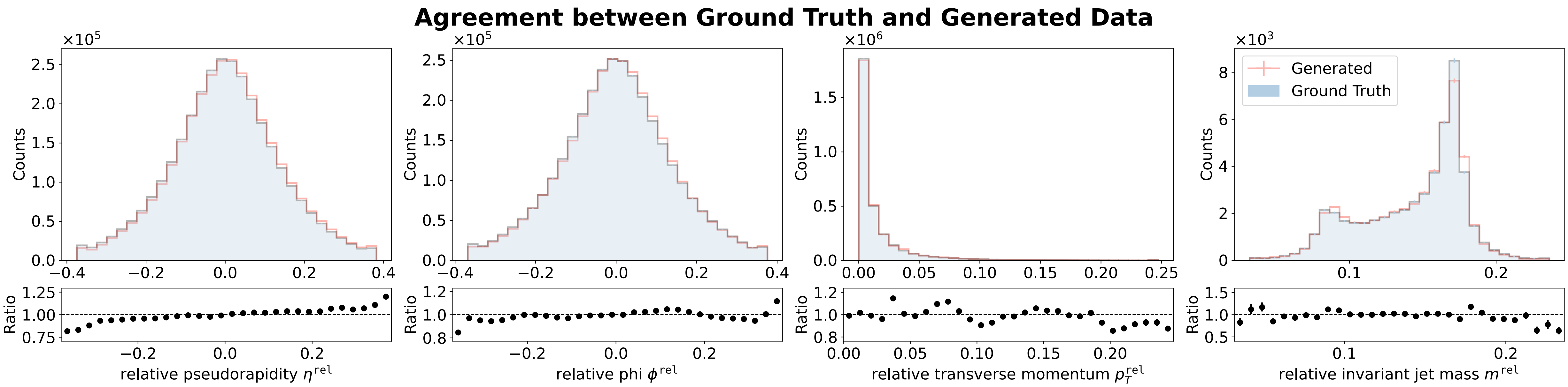
- WGAN GP Loss: $\begin{cases} L_C = -C(x_{real}) + C(x_{gen}) + GP & \text{Critic} \\ L_G = -C(G(z)) & \text{Generator} \end{cases}$
- Gradient Penalty: $GP = (\nabla_{\hat{x}}(C(\hat{x}) - 1)^2, \begin{cases} \hat{x} = \lambda x_{real} + (1-\lambda)x_{gen} \\ \lambda \sim U(0,1) \end{cases}$

  → Only interpolate between same sized clouds
- Deep Mean-field Matching: Generator L2 loss between mean-field in last critic layer for real and fake showers

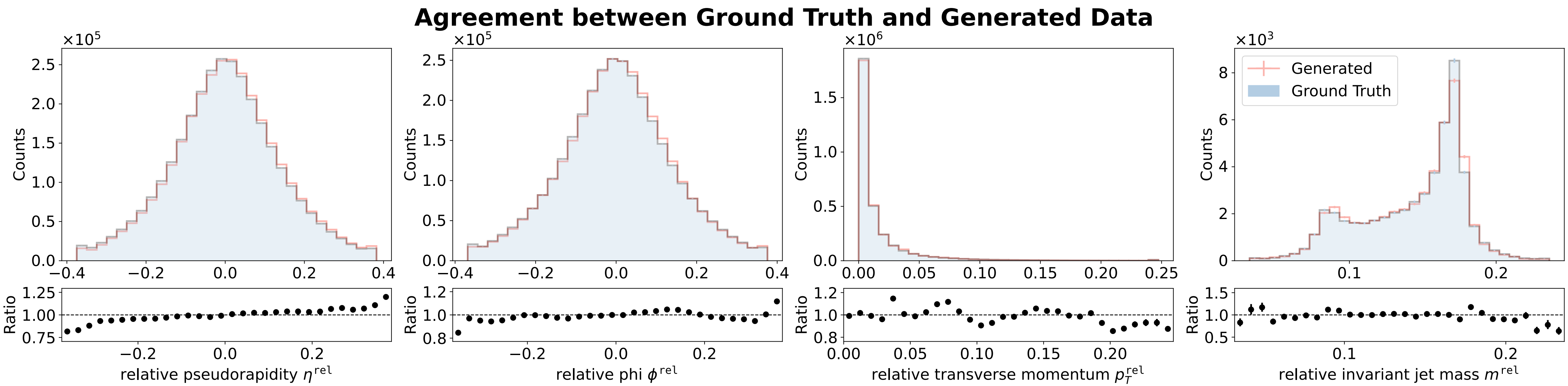
$$L_{MF} = \left| \bar{x}'_{fake} - \bar{x}'_{real} \right|^2$$

- Hence the name: **M**atching **D**eep **M**ean-fields **A**ttentive (**MDMA**) GAN

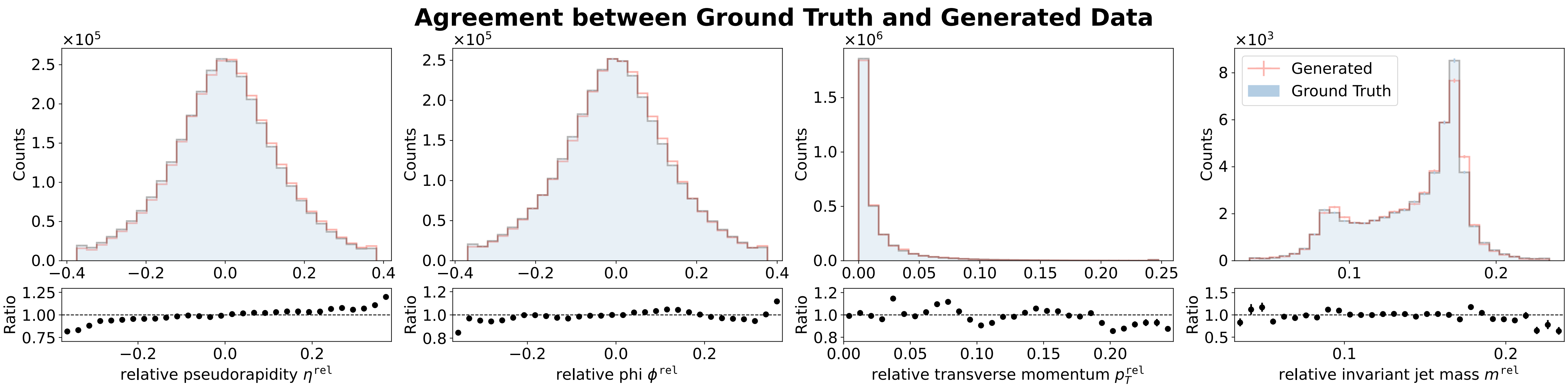
Results: Top-quark JetNet150



Results: Top-quark JetNet150



Results: JetNet150 - quantitative Results

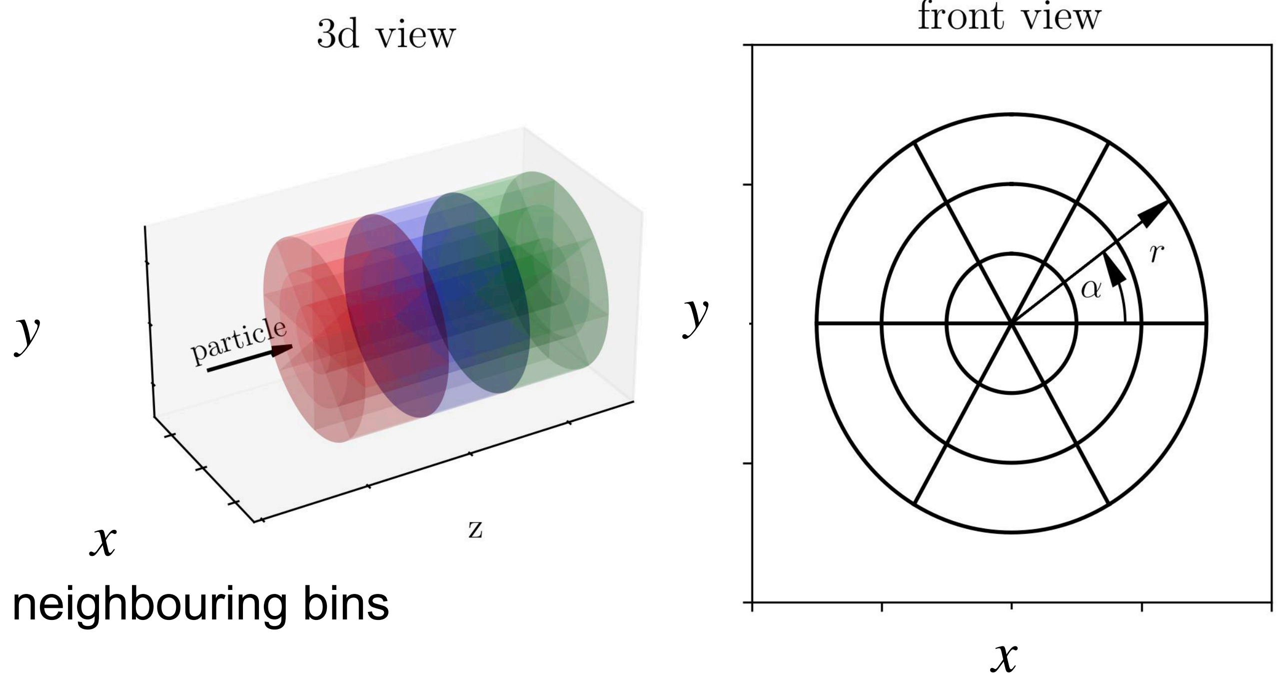


Jet Class	Model	$W_1^M (\times 10^3)$	$W_1^P (\times 10^3)$	$W_1^{EFP} (\times 10^5)$	$KPD (\times 10^4)$	$FPD (\times 10^4)$
Light Quark	EPiC	$\mathbf{0.5 \pm 0.1}$	1.25 ± 0.09	0.81 ± 0.16	-0.0 ± 0.1	5 ± 1
	MDMA	0.7 ± 0.1	$\mathbf{0.37 \pm 0.05}$	0.82 ± 0.14	-0.05 ± 0.07	$\mathbf{3.8 \pm 0.8}$
	IN	0.5 ± 0.1	0.15 ± 0.04	0.59 ± 0.16	-0.0 ± 0.2	3 ± 2
Gluon	EPiC	0.5 ± 0.1	1.13 ± 0.04	$\mathbf{0.93 \pm 0.14}$	0.06 ± 0.05	4.2 ± 0.7
	MDMA	0.5 ± 0.1	$\mathbf{0.27 \pm 0.02}$	1.12 ± 0.13	$\mathbf{-0.05 \pm 0.03}$	$\mathbf{1.4 \pm 0.5}$
	IN	0.6 ± 0.2	0.048 ± 0.009	0.79 ± 0.24	-0.1 ± 0.1	3.2 ± 0.8
Top Quark	EPiC	0.69 ± 0.08	0.65 ± 0.03	2.67 ± 0.39	1.7 ± 1.0	22 ± 1
	MDMA	$\mathbf{0.57 \pm 0.09}$	$\mathbf{0.10 \pm 0.02}$	$\mathbf{2.12 \pm 0.64}$	$\mathbf{-0.0 \pm 0.2}$	$\mathbf{5.3 \pm 0.9}$
	IN	0.42 ± 0.09	0.12 ± 0.04	1.22 ± 0.32	-0.1 ± 0.2	1.2 ± 0.6

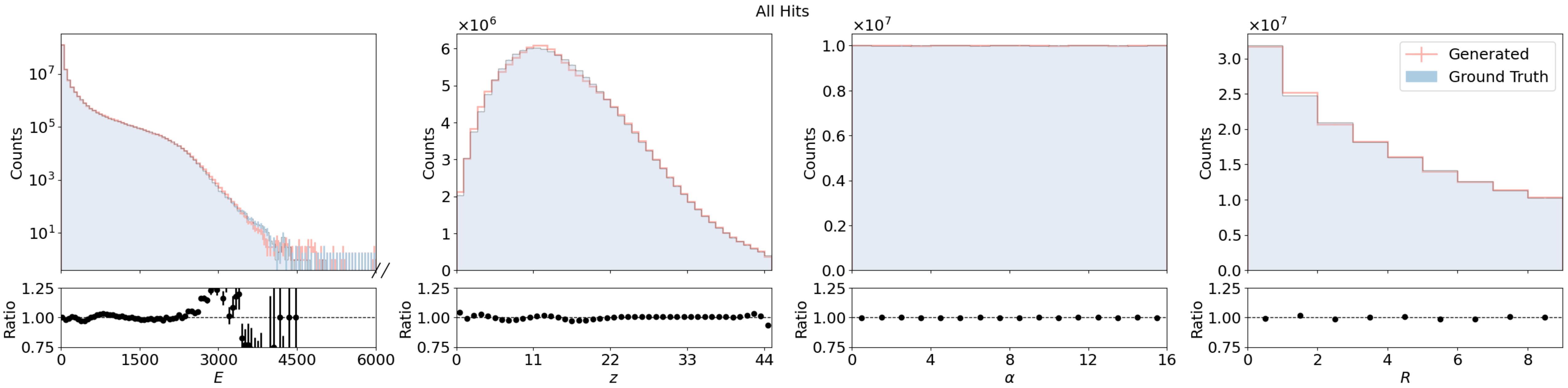
Best written in **bold**

Conclusion: **MDMA** is quite **EPiC** as well

Results: CaloChallenge



- Points: voxels with $E > 0$
- Coordinates: (z, α, R) index
- Dequantisation: Random noise interpolating between neighbouring bins



Classifier Metric, Detector Response & Cross Section

<u>Dataset 2</u>

High-level classifier:

AUC: 0.89

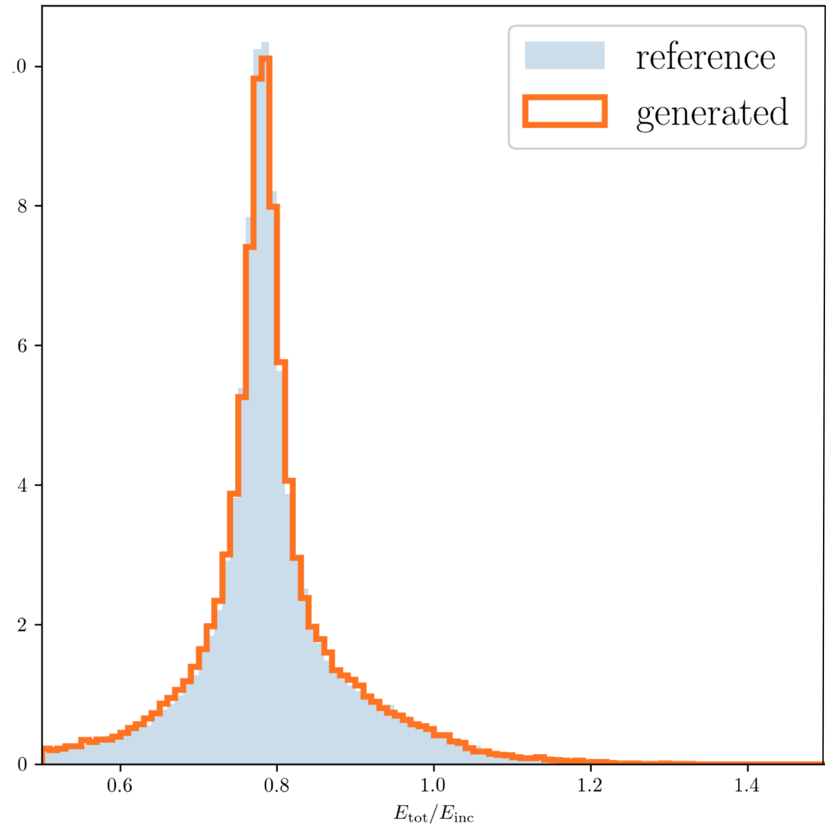
JSD: 0.39

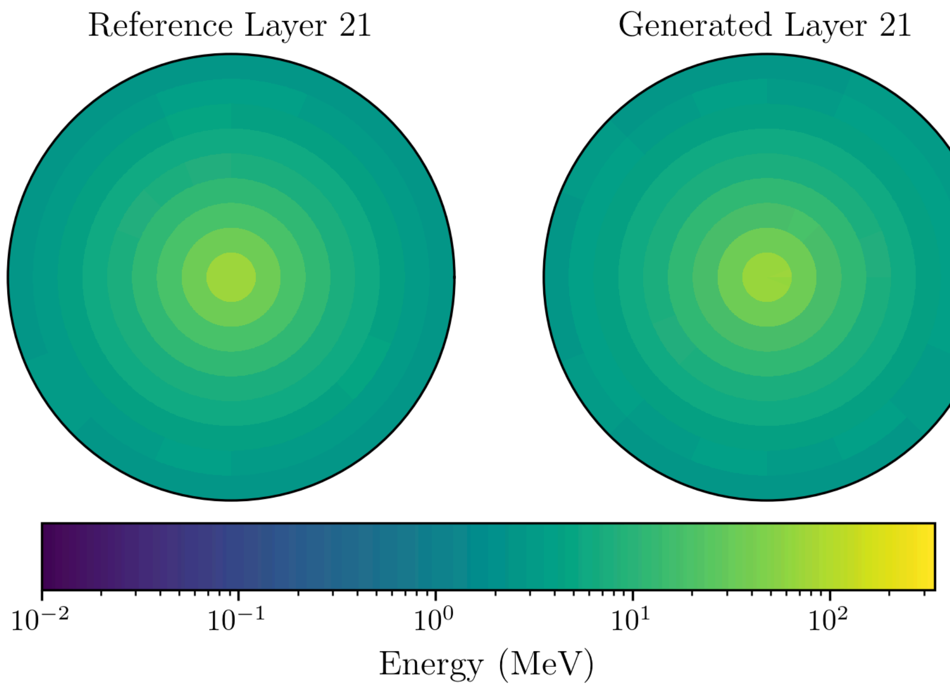
Low-Level classifier:

AUC: 0.97

JSD: 0.71

Target: AUC=0.5





But wait there's more

GANs are Dead - let's Match Flows on MDMA

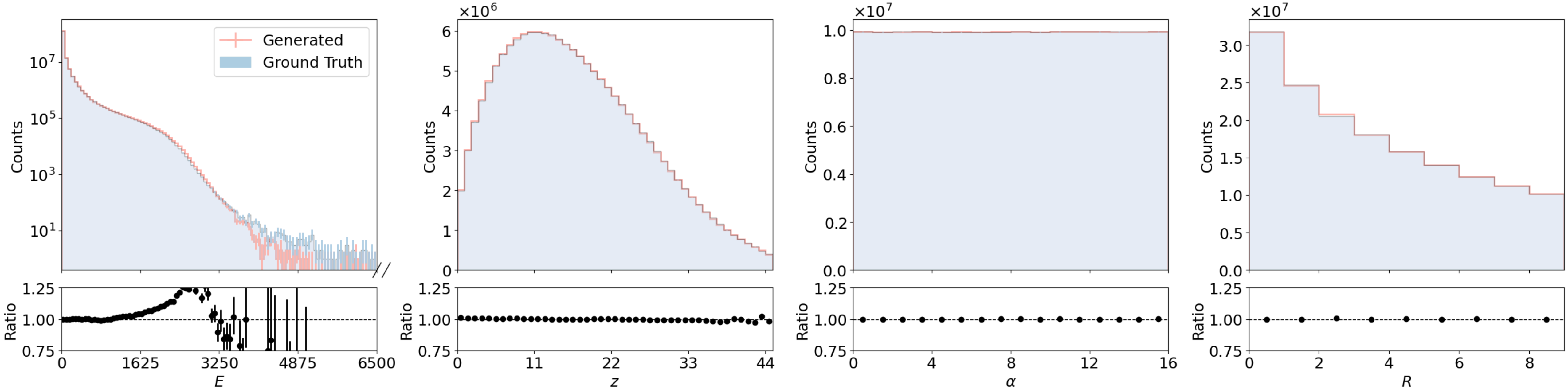
Based on work from Erik Buhmann, Cedric Ewen, Anatoli Karol and Gregor Kasieczka

- Flow Matching [1] novel generative framework, recently proposed for JetNet150 by Buhmann et al. [2]
- They used the EPiC block → replace it with MDMA-block
- Works out of the Box on JetNet150 → never spend months of GPU time on hyperparameter optimisation again!

[1]: Lipman et al., Flow Matching for Generative Modeling
[2]: Buhmann et al., EPiC-ly Fast Particle Cloud Generation with Flow-Matching and Diffusion

Promising First Results on CaloChallenge

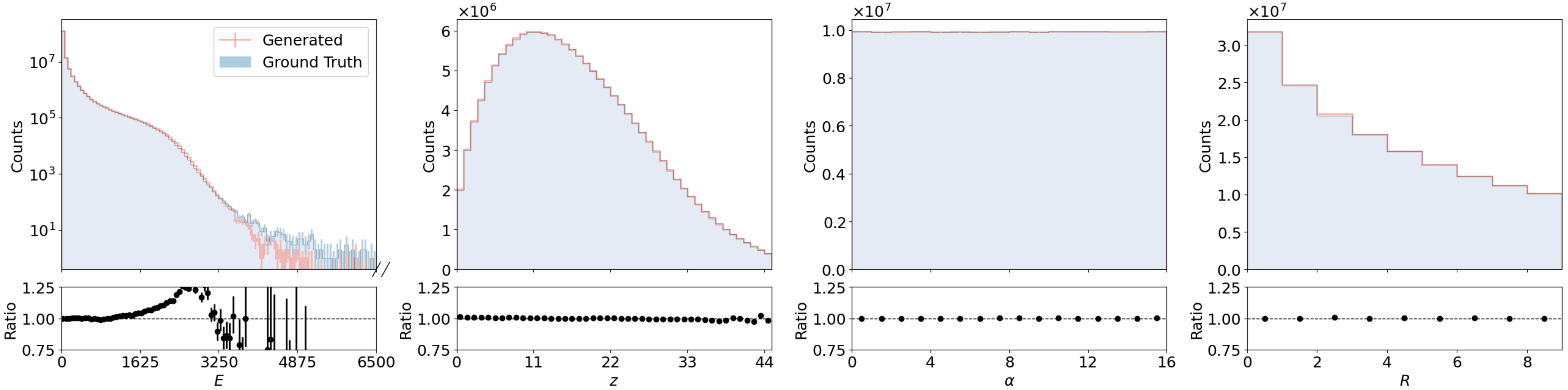
Downside: it is a lot slower but please don't ask



<u>Dataset 2</u>
High-level classifier:
AUC: 0.69
JSD: 0.08
Low-level classifier:
AUC: 0.90
JSD: 0.47

Promising First Results on CaloChallenge

Downside: it is a lot slower ~~but please don't ask~~



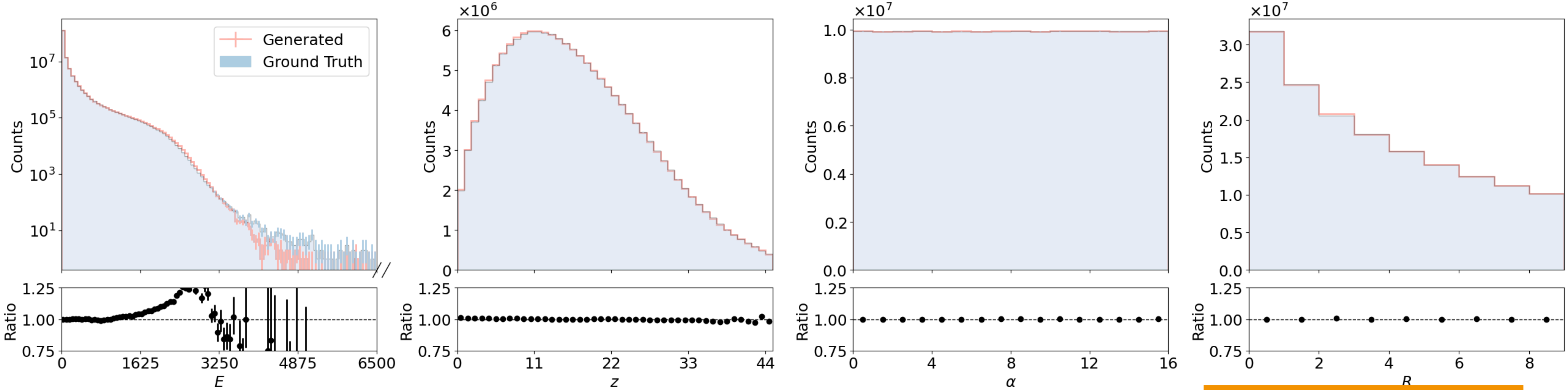
Generation Time:

670 μs→ 270 ms

GAN Results		Flow Matching
High-level classifier:		High-level classifier:
AUC: 0.89		AUC: 0.69
JSD: 0.39	→	JSD: 0.08
Low-level classifier:		Low-level classifier:
AUC: 0.97		AUC: 0.94
JSD: 0.71		JSD: 0.57

Promising First Results on CaloChallenge

Downside: it is a lot slower ~~but please don't ask~~



Generation Time:

$670\ \mu s \rightarrow 270\ ms$

GAN Results
High-level classifier:
AUC: 0.89
JSD: 0.39
Low-level classifier:
AUC: 0.97
JSD: 0.71

→

Flow Matching
High-level classifier:
AUC: 0.69
JSD: 0.08
Low-level classifier:
AUC: 0.94
JSD: 0.57

→

Shifting Double Hits
High-level classifier:
AUC: 0.69
JSD: 0.08
Low-level classifier:
AUC: 0.90
JSD: 0.47

See Moritz's Scham talk:
caloutils (Thursday)

Conclusion

- Point Clouds representation of choice for colliders (sparsity, geometry independent)
- Model using attention-based aggregation mechanism that scales linearly
- Linear scaling achieved by proxying particle-particle interactions by mean-field interaction
- Attention sounds complicated - inherently in our case just a weighted sum
- Results presented for 2 datasets: JetNet150-Top and CaloChallenge Dataset 2
- Although main building block was designed for GAN also works with Flow Matching → improves results on CaloChallenge Dataset 2
- Albeit Flow Matching orders slower, a lot more pleasurable to work with - speed up also possible!

Questions?

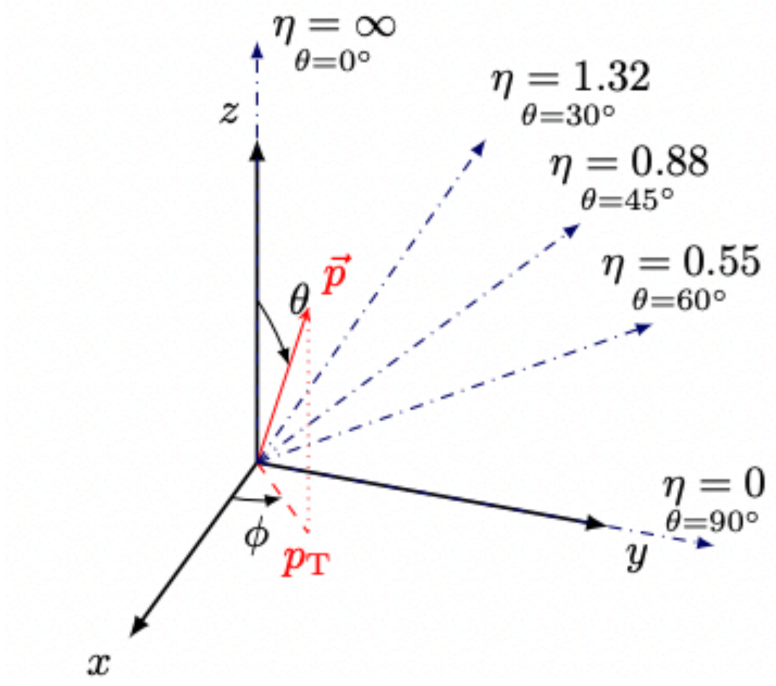
JetNet

JetNet [1] Datasets

- Jets: unordered sprays of particles
- Particles: tuples of $(\eta^{\mathrm{rel}}, \phi^{\mathrm{rel}}, p_T^{\mathrm{rel}})$ relative to jet axis
- Constrained to max 150 particles/jet

  → Goal: generate $\boldsymbol{X} = \left\{ \left(\eta_{(i)}^{\mathrm{rel}}, \phi_{(i)}^{\mathrm{rel}}, p_{T,(i)}^{\mathrm{rel}} \right) \right\}_{(i \leq n)} \sim p_{data}(\boldsymbol{X})$

- Invariant jet mass: $m_{rel}{}^2 = \left(\sum_{i=1}^{n} |\boldsymbol{p}_i| \right)^2 - \left(\sum_{i=1}^{n} \boldsymbol{p}_i \right)^2$
- Size $\sim 178'000$ Samples
- 70% used for training
- Benchmarking possible



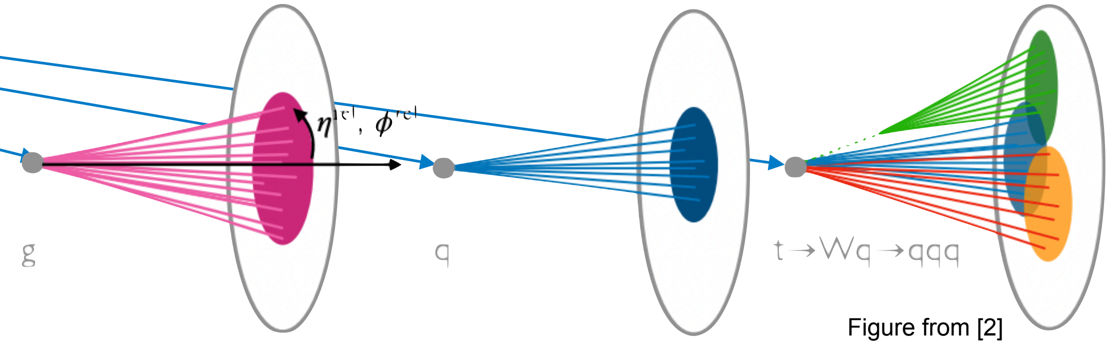


Figure from [2]

[1] Kansal et al, JetNet, PyPi
[2] Kansal et al., Particle Cloud Generation with Message Passing Generative Adversarial Networks, arxiv.org/abs/2106.11535

Point Cloud Representation of Collider Data

- Handle sparsity in the detector by representing hits as Point Cloud [1]
- Reduces batch dimension to (batch size,~4k hits,4) → fits on one GPU
- Model learns physics ~decoupled of detector geometry
- Mapping detector cells to point clouds:

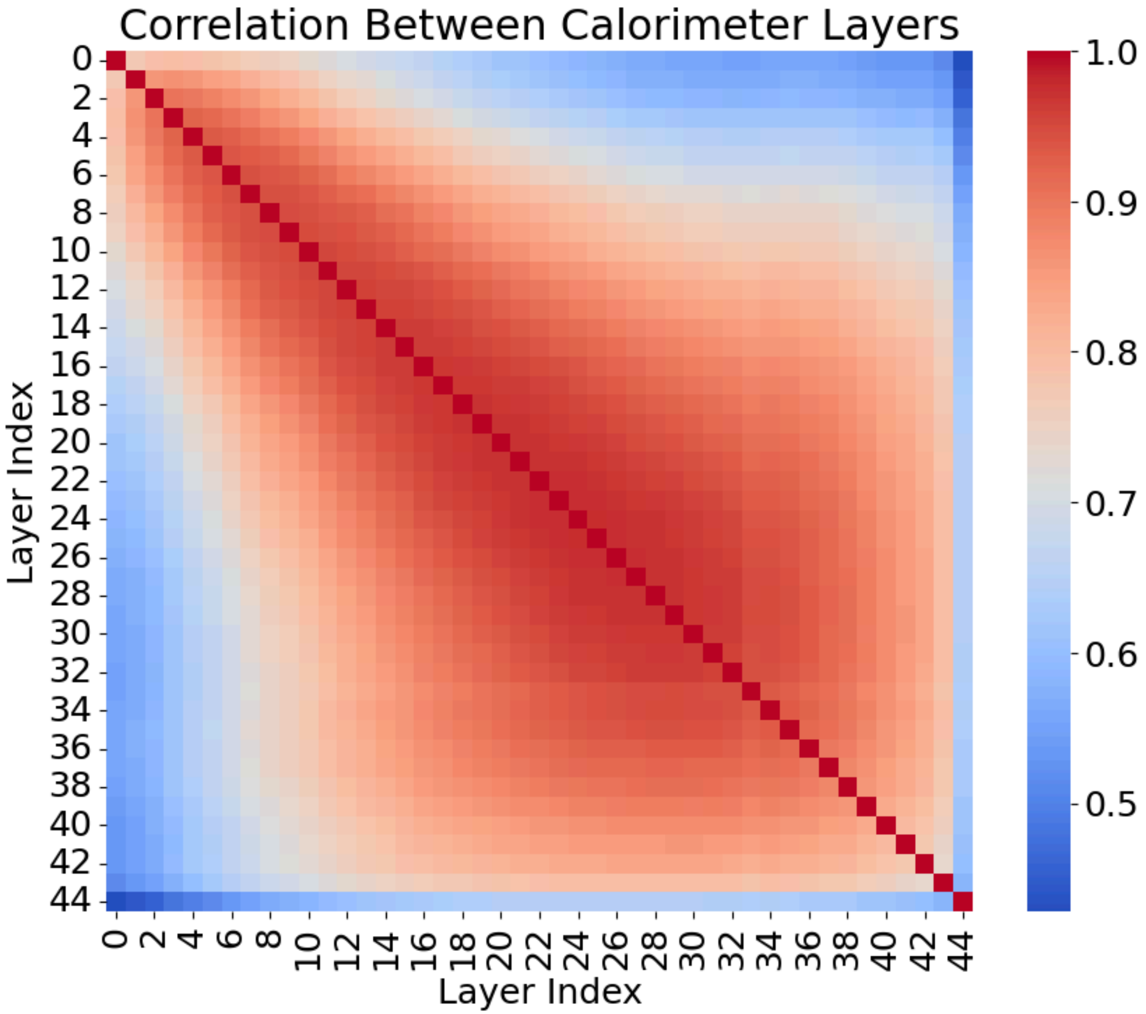
```
hits=detector[E>0]
for coordinate  in (z, alpha, R):
    for cellnumber in enumerate(cells):
            coordinate(cellnumber) x=cellnumber + DQ(0,1)
    x=MinMaxScaler(x)
    x=Logit(x)
    x=StandardScale(x)
for hit in hits:
    E(hit)=BoxCoxTransform(E(hit))
```

- Gives about Gaussian distribution for different variables (except α)
- α periodcity → problematic
- Volume of space **not** respected in particle cloud definition

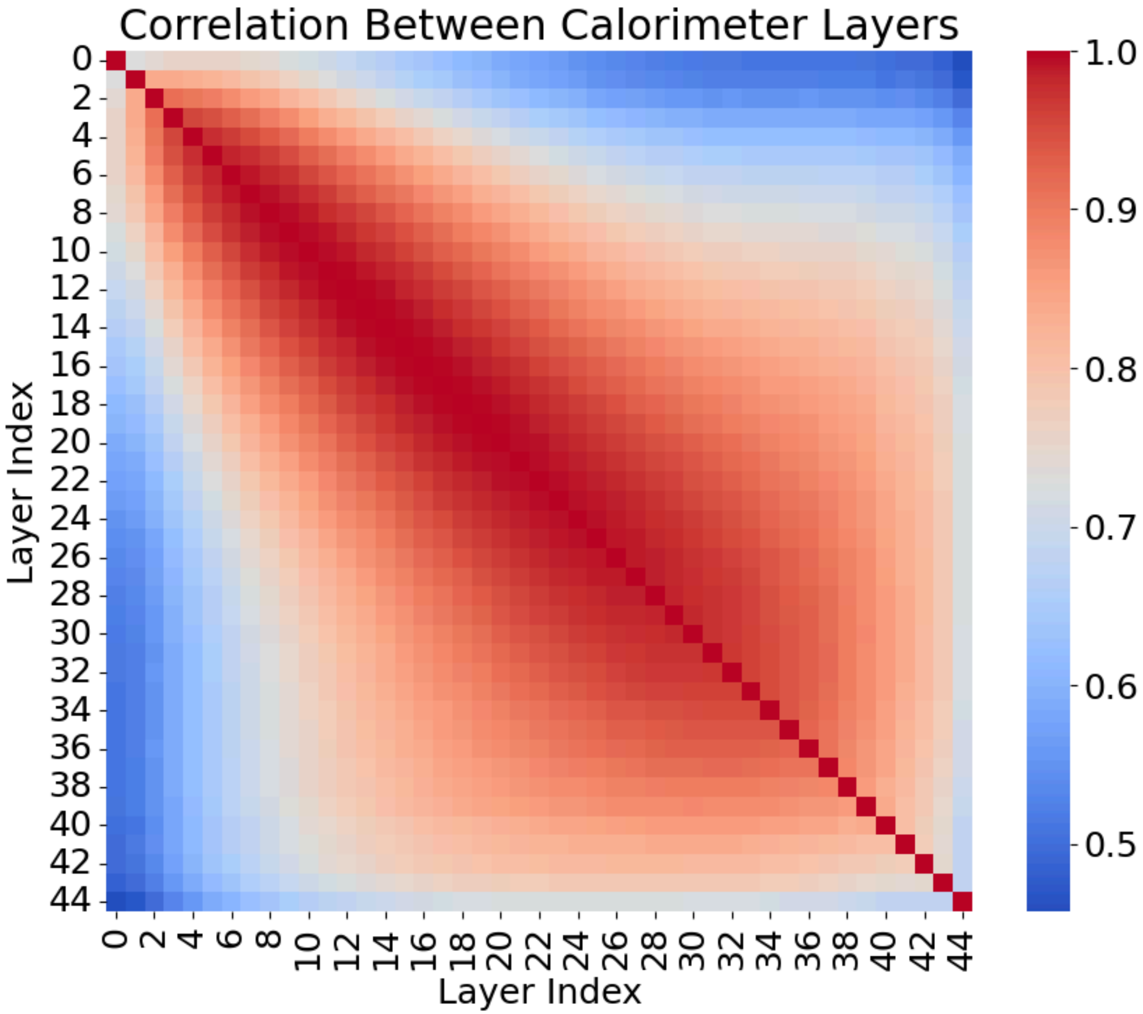
[1] Simon Schnake et al., 2022

CaloChallenge Correlation between Layers

Generated

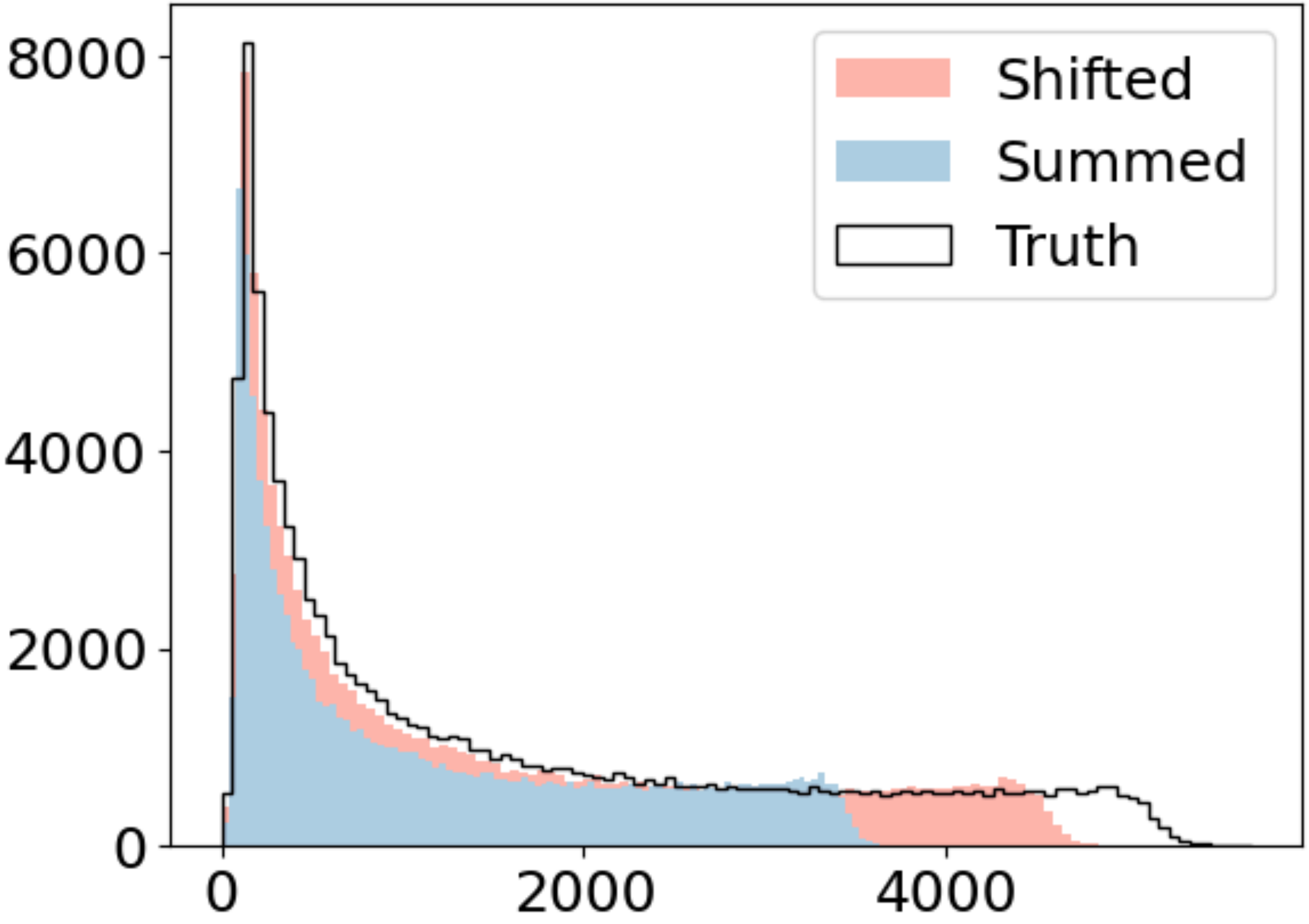


Ground Truth

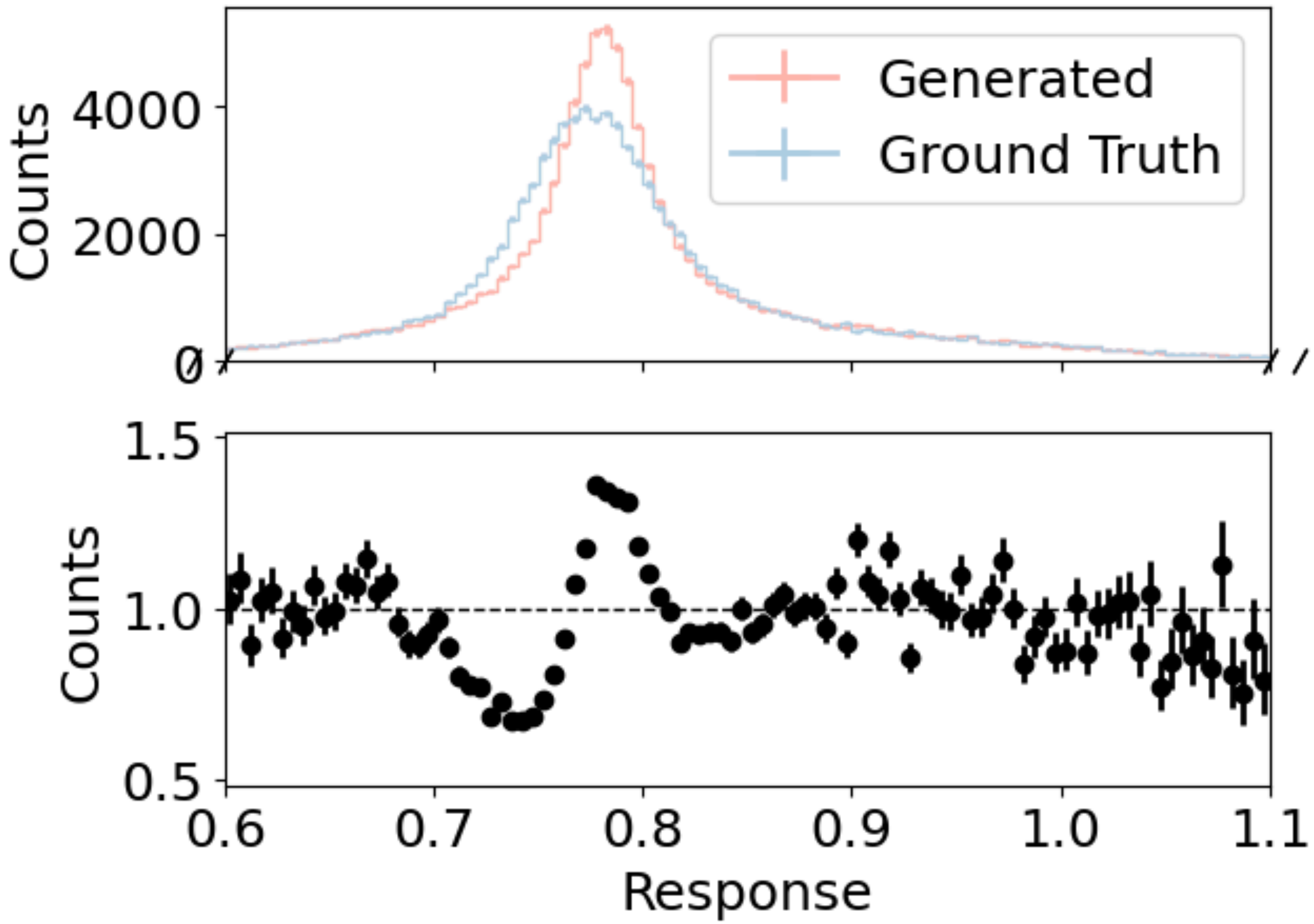


Double Hits

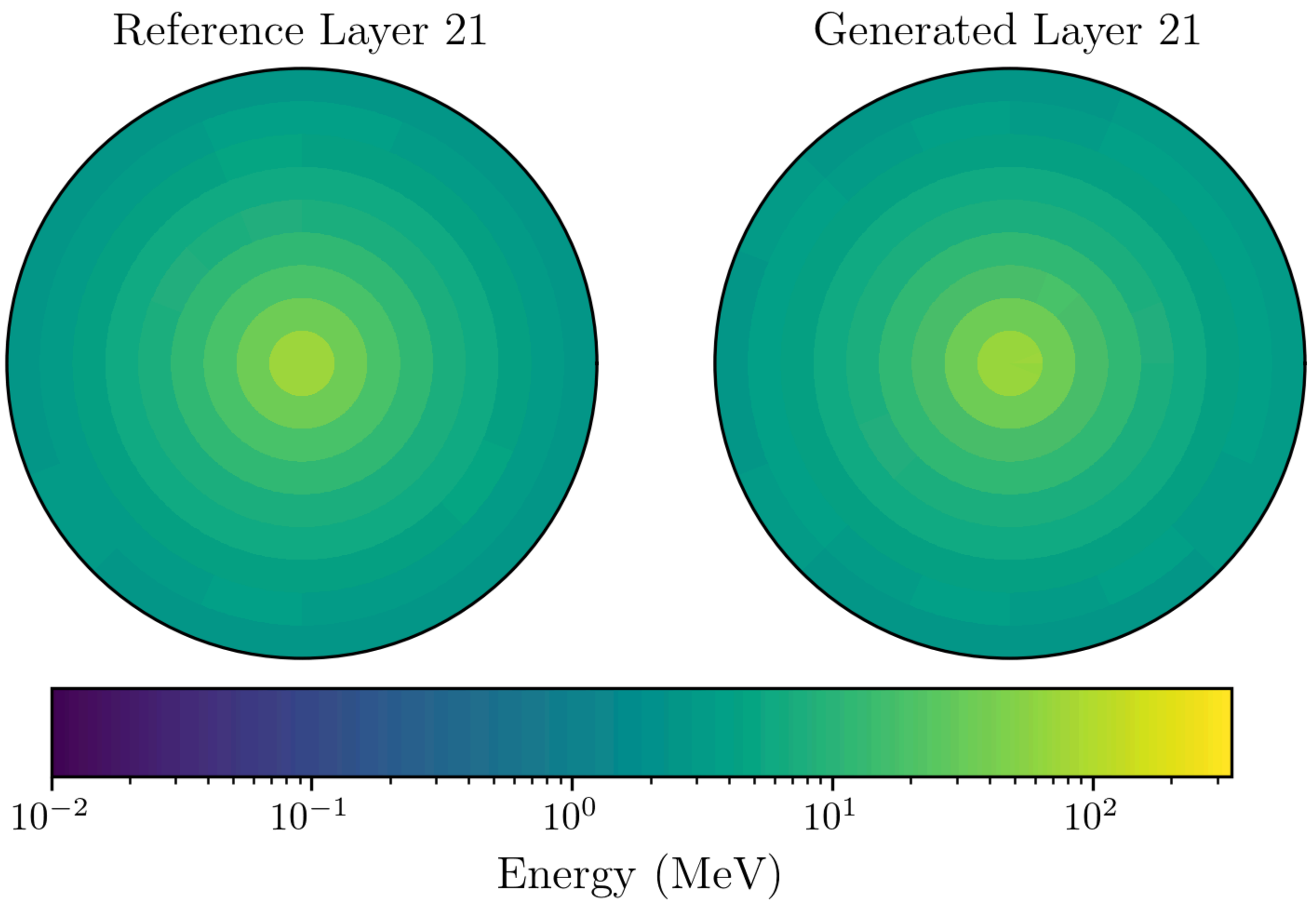
- Multiple hits can get assigned to same detector cell
- Simplest approach: just summing energies
- Check Moritz Schams presentation about our repository *caloutils* for a more sophisticated approach!



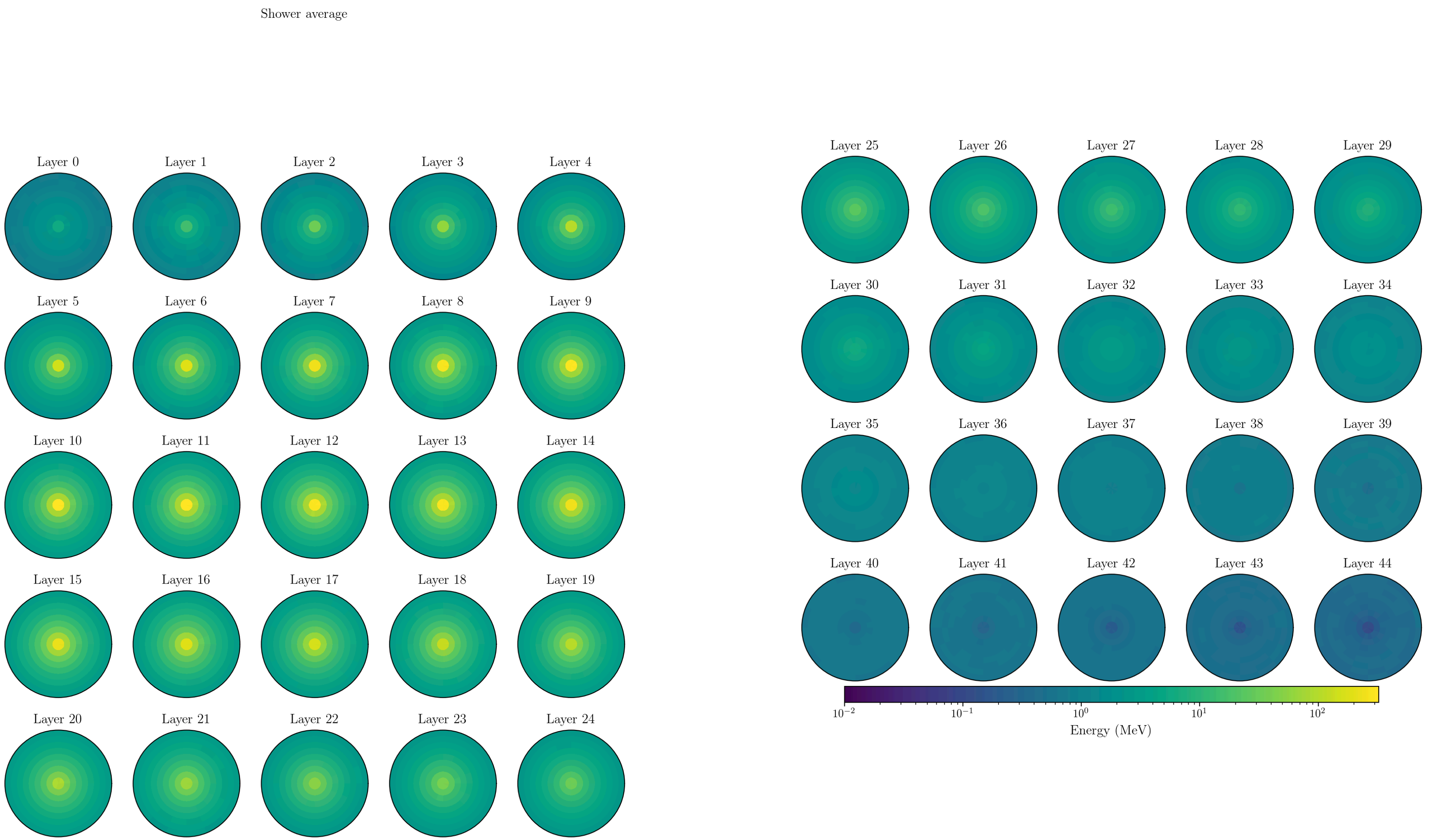
Todo: Improve Incoming Energy Conditioning



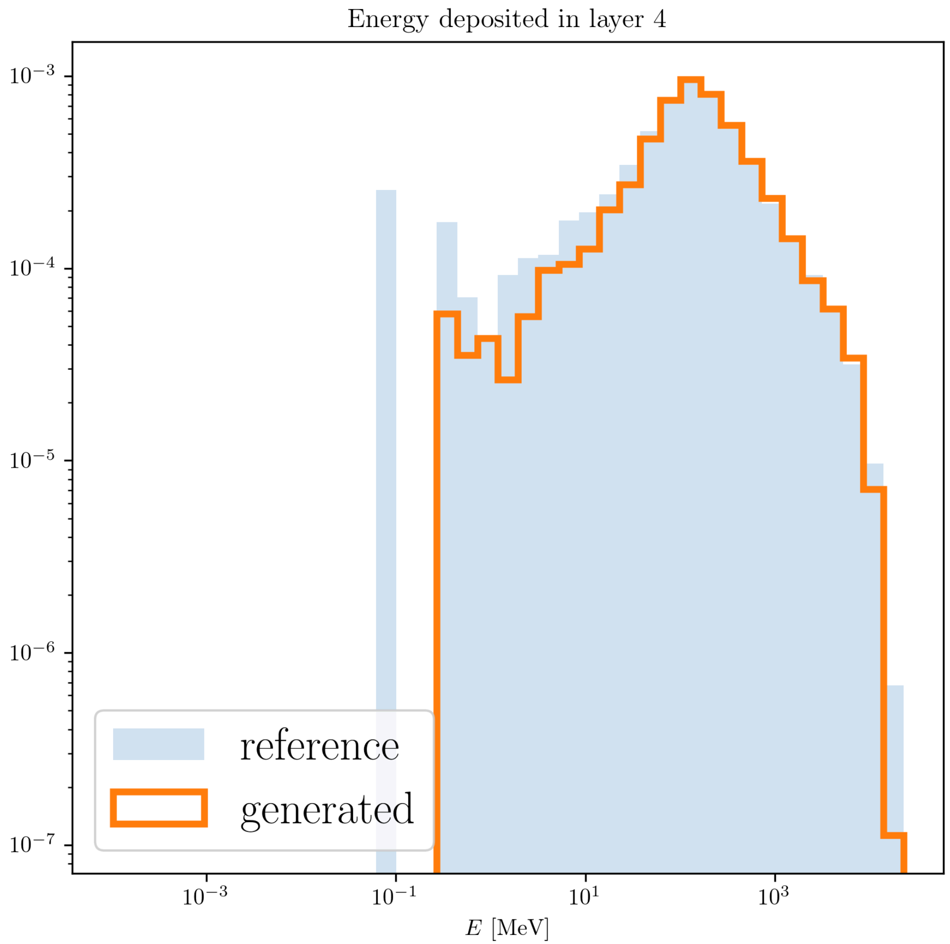
Energy Per Layer

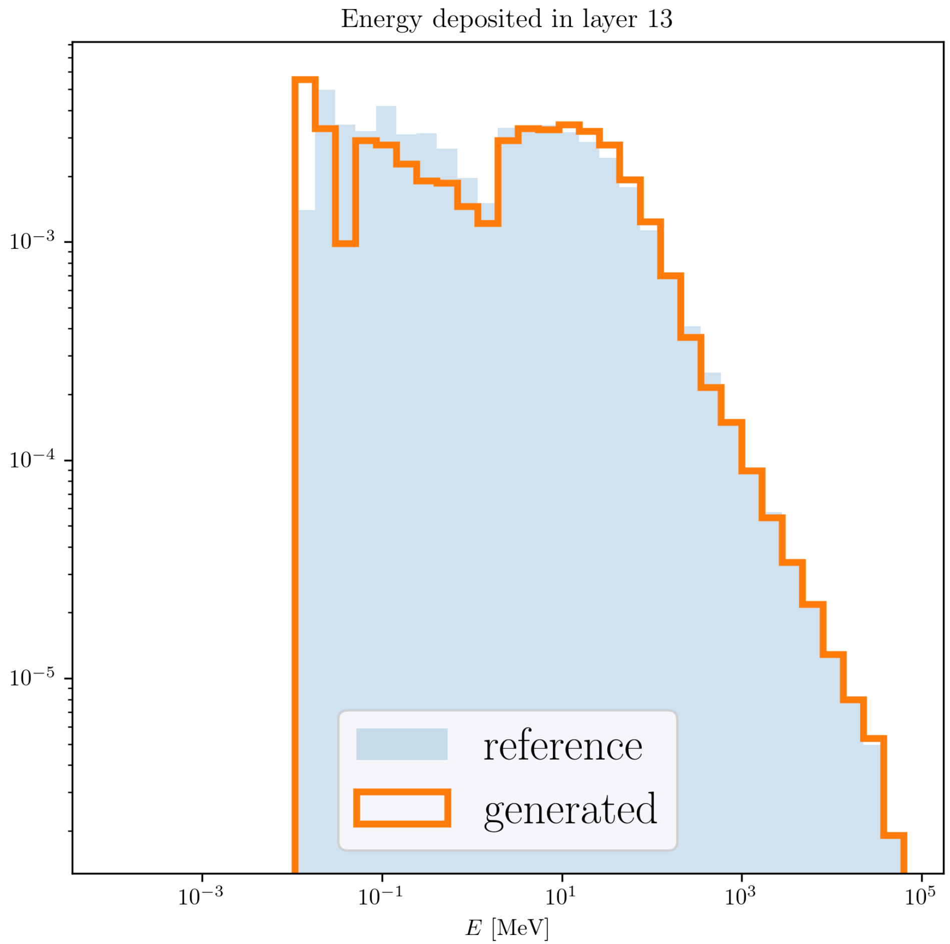


All Layers


Shower average
Layer 0
Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
Layer 6
Layer 7
Layer 8
Layer 9
Layer 10
Layer 11
Layer 12
Layer 13
Layer 14
Layer 15
Layer 16
Layer 17
Layer 18
Layer 19
Layer 20
Layer 21
Layer 22
Layer 23
Layer 24
Layer 25
Layer 26
Layer 27
Layer 28
Layer 29
Layer 30
Layer 31
Layer 32
Layer 33
Layer 34
Layer 35
Layer 36
Layer 37
Layer 38
Layer 39
Layer 40
Layer 41
Layer 42
Layer 43
Layer 44
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
Energy (MeV)

Energy Deposition per Layer





Center of Energy