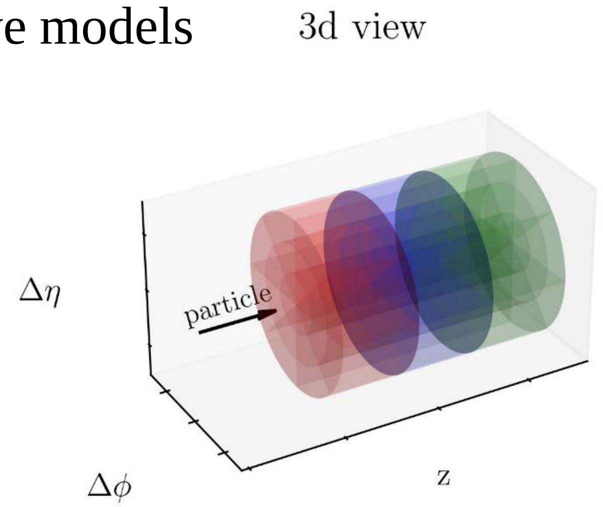
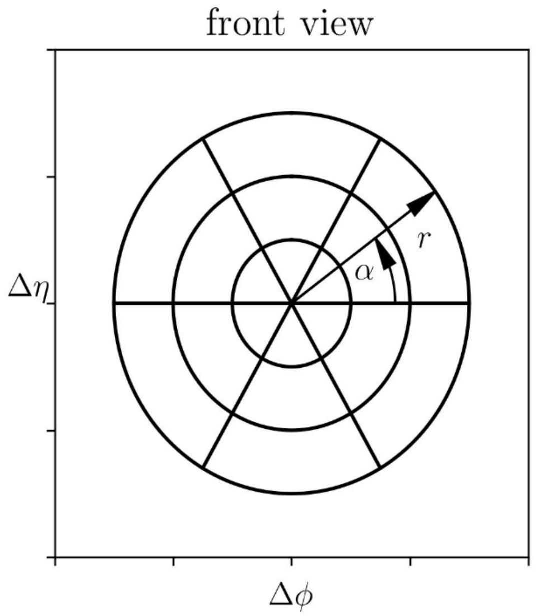
CaloDiffusion with GLaM for High Fidelity Calorimeter Simulation

Oz Amram, **Kevin Pedro**
(Fermilab)
November 7, 2023

Calorimeter Simulation

- CaloChallenge: common datasets for evaluation & comparison of generative models
  - Dataset 1: ATLAS calorimeter, irregular
    - Photons (368 voxels), 242K events
    - Pions (533 voxels), 241.6K events
  - Dataset 2: silicon-tungsten, 45 layers
    - Electrons (6480 voxels), 200K events
  - Dataset 3: silicon-tungsten, 45 layers
    - Electrons (40500 voxels), 200K events
- Preprocessing: (E_i = voxel energy)
  - Logit transform: $u_i = \log(\frac{x}{1-x})$, $x \equiv \delta + (1 - 2\delta)E_i$
  - Standardization: $u_i' = (u_i - \overline{u})/\sigma_u$

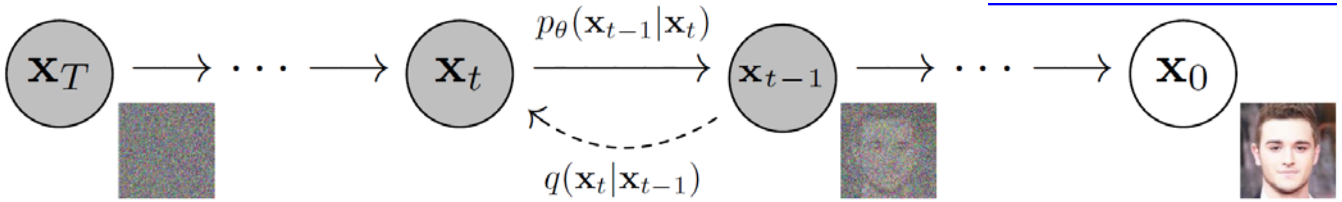




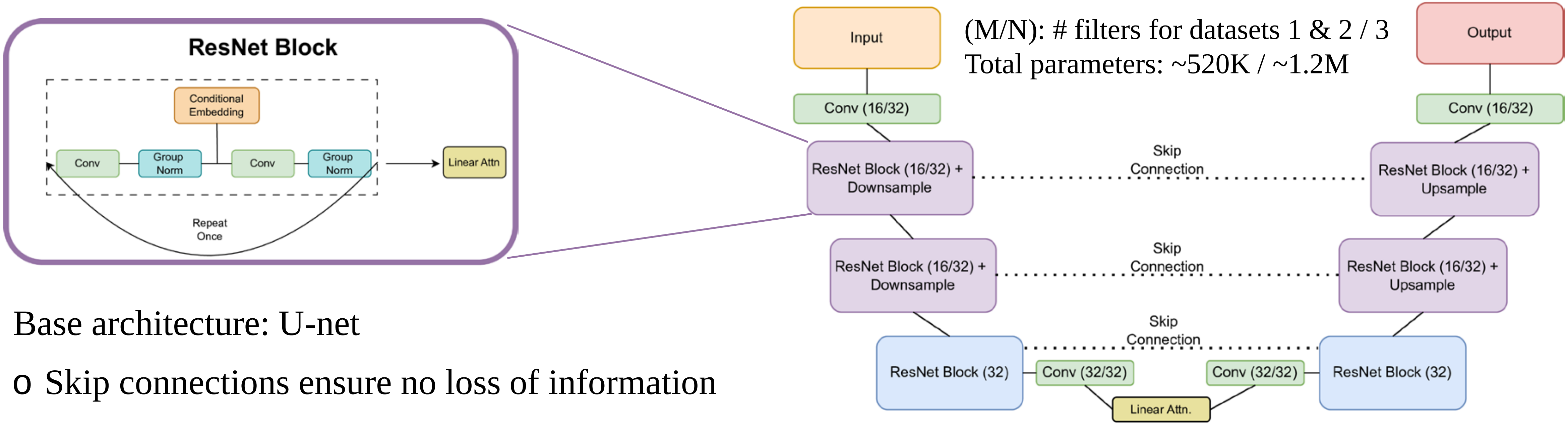
Diffusion Models

- Learn to reverse a “noising process” that iteratively adds Gaussian noise to image
  - Here: learn denoising directly
    - More sophisticated version of early denoising approaches e.g. arXiv:2202.05320
  - Alternative: score-based, learn gradient of probability density
    - Equivalent to denoising for “variance-preserving” score formulation
- Generate image from pure noise by iteratively applying learned denoising
  - Conditioned on relevant properties
- Rapidly adopted for image generation
  - Now dominant after just ~2 years

arXiv:2006.11239



CaloDiffusion



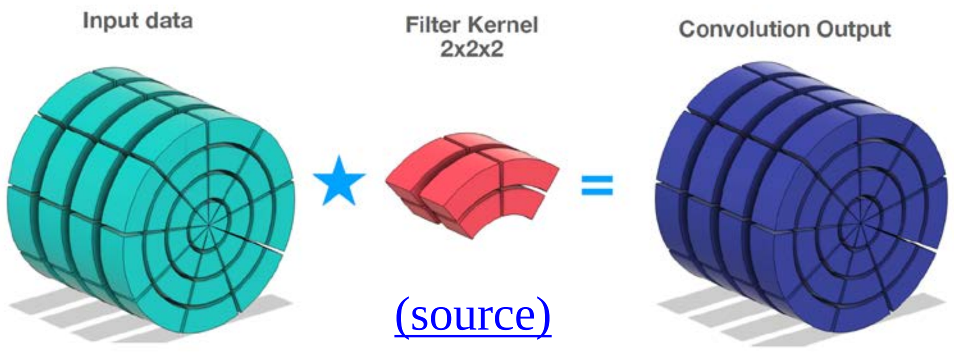
(M/N): # filters for datasets 1 & 2 / 3
Total parameters: ~520K / ~1.2M

- Base architecture: U-net
  - Skip connections ensure no loss of information
- Linear self-attention layers applied to each convolutional ResNet block
  - Allows dimensionality reduction in *z* to handle longitudinal correlations in showers
- + numerous geometric innovations (next slide)
- Cosine noise schedule for training
- Stochastic sampling algorithm for generation

- Objectives:
  - Datasets 1 & 2: predict (normalized) noise
  - Dataset 3: predict weighted average of noise and denoised image
- Aim for highest achievable quality first
  - Then focus on improving speed
  - Wrong answers can be obtained infinitely fast

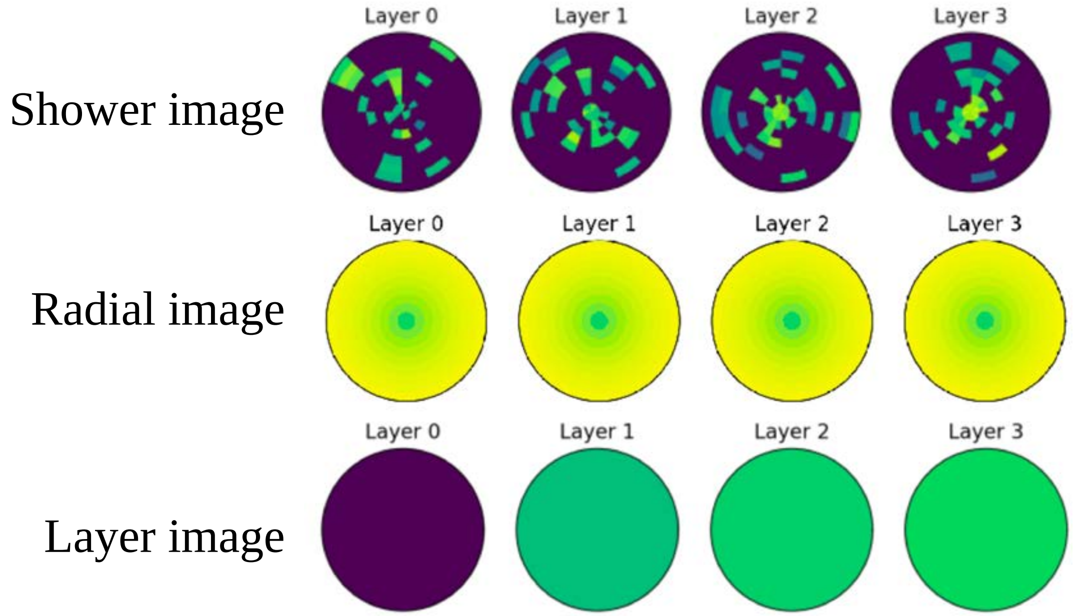
Geometric Innovations

- Particle showers are invariant & periodic in φ
  - Pad in φ so convolutions "wrap around"



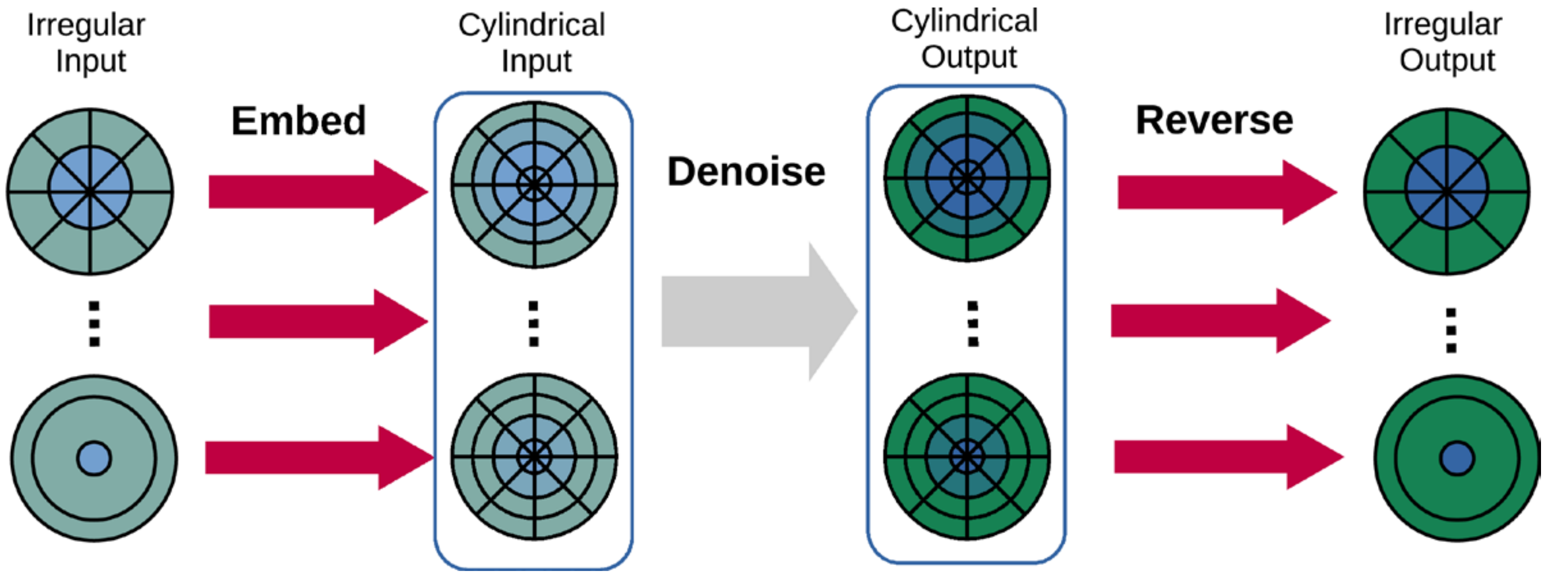
[(source)]

- Particle showers are *not* invariant in *r* or *z*
  - Provide *r* and *z* (layer) as extra per-pixel channels (input features)
  - Convolutions become *conditional*



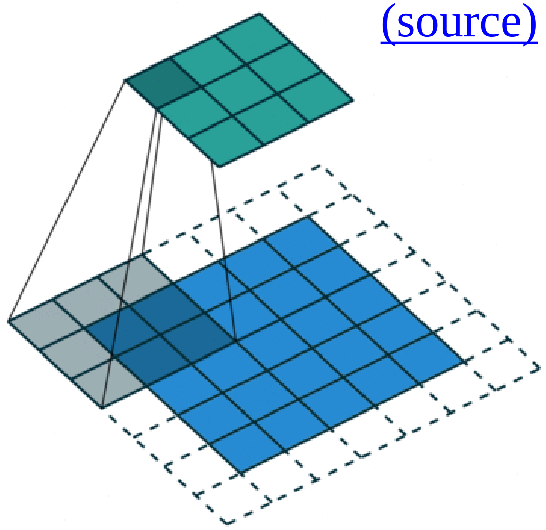
➢ *Conditional cylindrical convolutions*
  - Handle inherent features of particle detector geometry, distinct from rectangular images

Geometry Latent Mapping: GLaM



- Dataset 1 has different radial/angular bins in each layer
  - Can't directly apply convolutions, which require regular neighbor structure
- Learn forward and reverse embeddings to and from a regular geometry
  - Simple matrices C (NxM) and D (MxN)
    - C initialized to split or merge cells based on overlap between original and embedded geometries
    - D initialized as Moore-Penrose pseudoinverse of C
- Inspired by "latent diffusion" approach
  - But not necessarily lower-dimensional representation; actually higher-dimensional here

Why Convolutions?

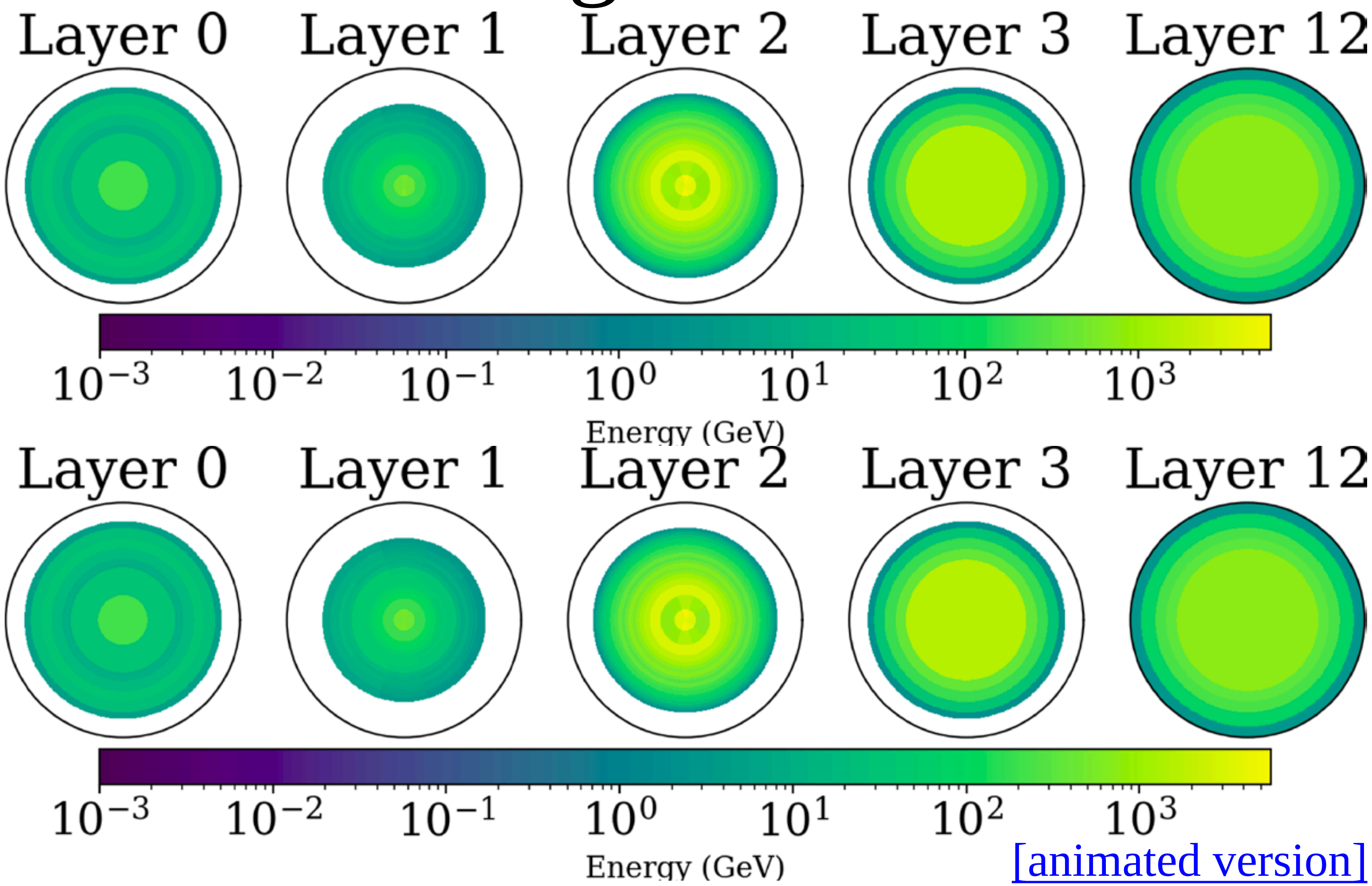


(source)

- Convolutions started the modern machine learning revolution (AlexNet, 2012)
  - *Spatial locality* and translational invariance
  - Shared weights → fewer parameters, *better scaling*
  - Highly *efficient* on GPUs: spatial locality implies memory locality
- Ideally suited for computer vision with rectangular images
  - Application to irregular geometries requires innovations
- Graph neural networks?
  - **Pro**: natural representation for irregular geometries
  - **Cons**: adjacency matrices consume substantial memory; operations less local/efficient; hard to generate arbitrary output (masking technique exists, but difficult to scale)
- Point clouds or transformers?
  - **Pro**: no adjacency matrix consuming memory
  - **Con**: discards useful geometric information, which then must be learned from (often sparse) inputs

➢ For generative applications, convolutions still have a lot to offer!
  - And they can keep up with transformers when trained properly… arXiv:2310.16764

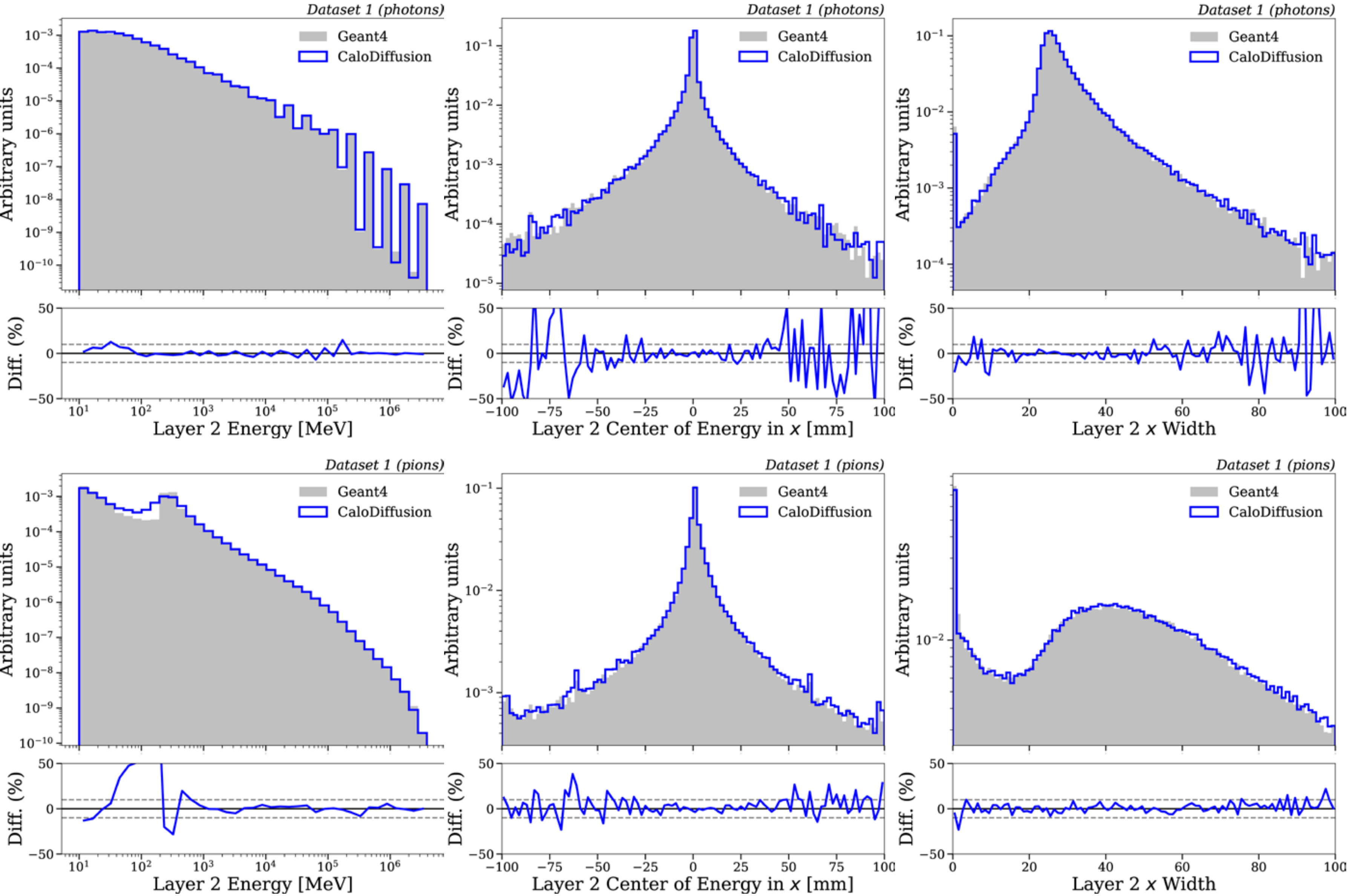
Average Showers



[animated version]

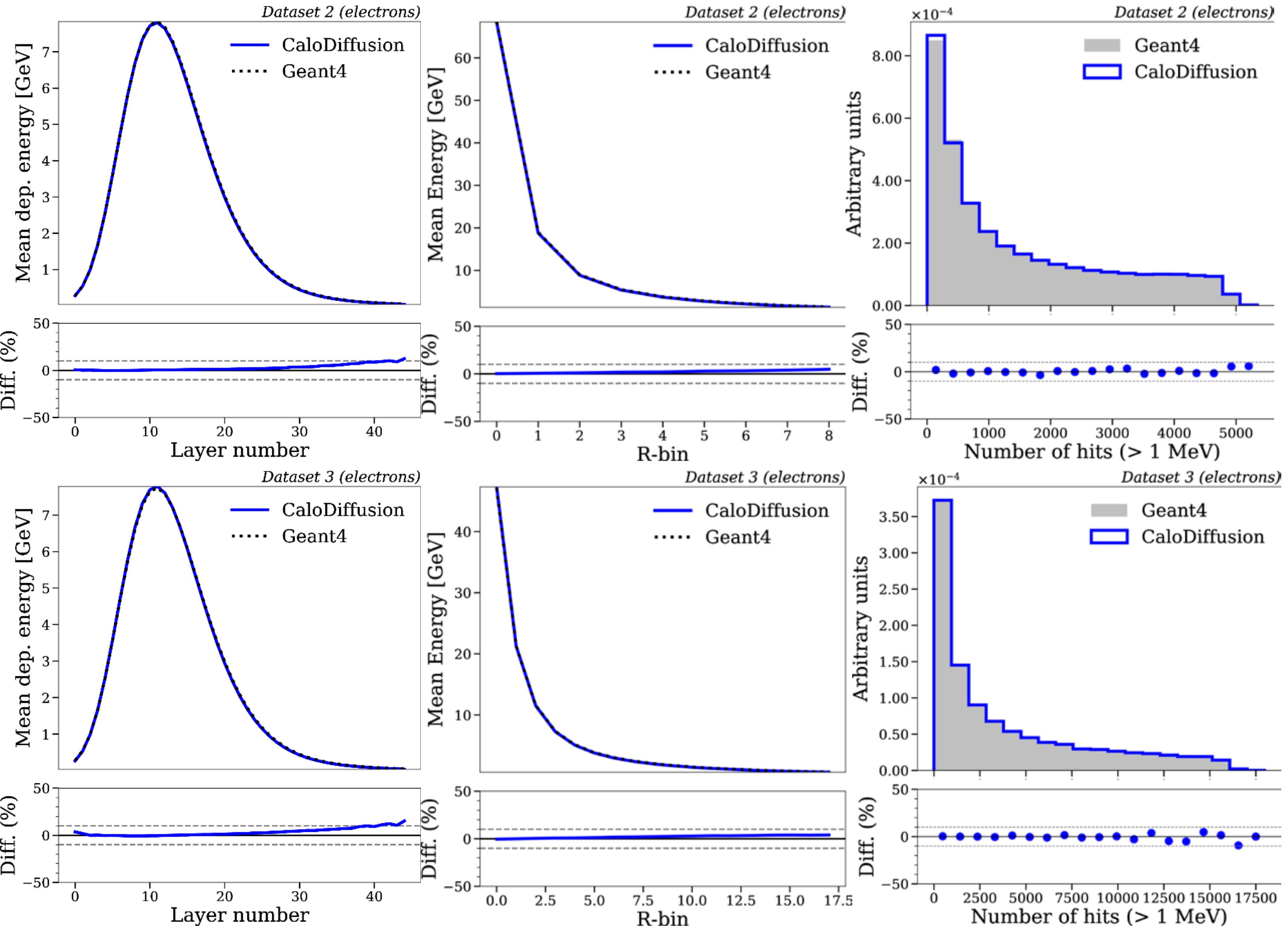
- Top: Geant4; bottom: CaloDiffusion (dataset 1, photons)
  - … or is it the other way around? Can you tell?

Dataset 1



- Excellent modeling for photon showers
- Some mismodeling of low-energy pions
  - Could be resolved by dedicated training/conditioning
  - No significant impact on shower shape variables

Datasets 2 & 3



- Very good agreement in shower shapes and physically important quantities
- So far, have only shown 1D comparisons
- Next: further and higher-dimensional quantification

Metrics

- Classifier AUC: train a binary classifier to distinguish between Geant4 and generative model
  - 2 hidden layers, 2048 neurons each; 20% dropout after each layer
  - Two flavors w/ different inputs: (incident particle energy included in both)
    - Low-level: full showers (all voxels)
    - High-level: energy in each layer, center of energy and shower width in η and φ
  - Compared to CaloScore v2 (undistilled), (i)CaloFlow (teacher)
- Integral probability metrics: Fréchet Particle Distance (FPD), Kernel Particle Distance (KPD)
  - High-level shower features used as input

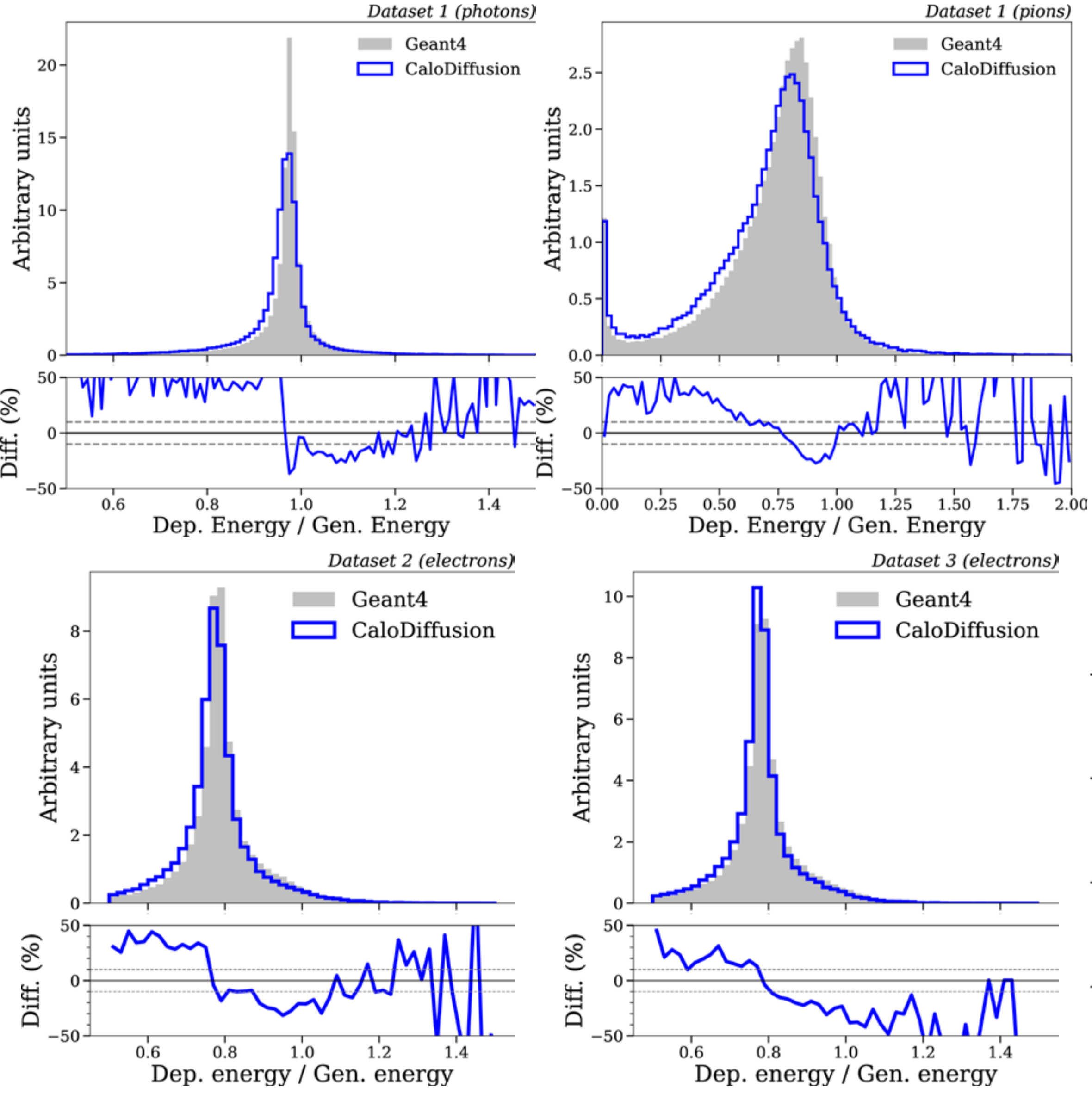
	Classifier AUC (low / high)		
Dataset	`CaloDiffusion`	`CaloFlow`	`CaloScore v2`
1 (photons)	**0.62** / 0.62	0.70 / **0.55**	0.76 / 0.59
1 (pions)	**0.65** / **0.65**	0.78 / 0.70	- / -
2 (electrons)	**0.56** / **0.56**	0.80 / 0.80	0.60 / 0.62
3 (electrons)	**0.56** / **0.57**	0.91 / 0.95	0.67 / 0.85

Dataset	FPD†	KPD
1 (photons)	0.014(1)	0.004(1)
1 (pions)	0.029(1)	0.004(1)
2 (electrons)	0.043(2)	0.0001(2)
3 (electrons)	0.031(2)	0.0001(1)

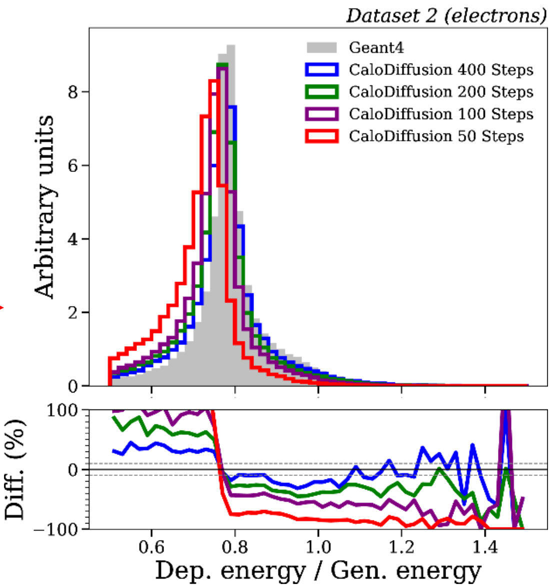
- CaloDiffusion wins in almost all comparisons, with very small distance values
  - Generated showers almost indistinguishable from Geant4
  - Further comparisons to come in CaloChallenge summary

† Geant4 self-comparison values subtracted (0.008, 0.0005, 0.008, 0.011)

Areas for Improvement



- Deficit in total energy modeling
- Need 400 diffusion steps to get acceptable quality
  - o Still faster than Geant4 (~100s) w/ batching on GPU
- Fewer steps:
  - o Linear speed improvement
  - o But even less accurate in this quantity

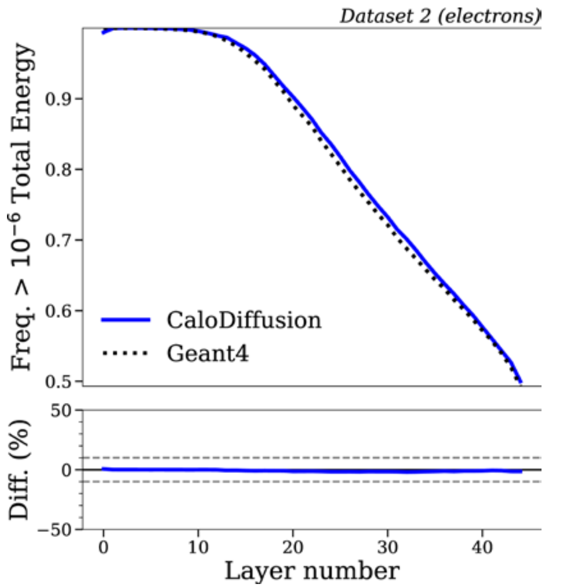


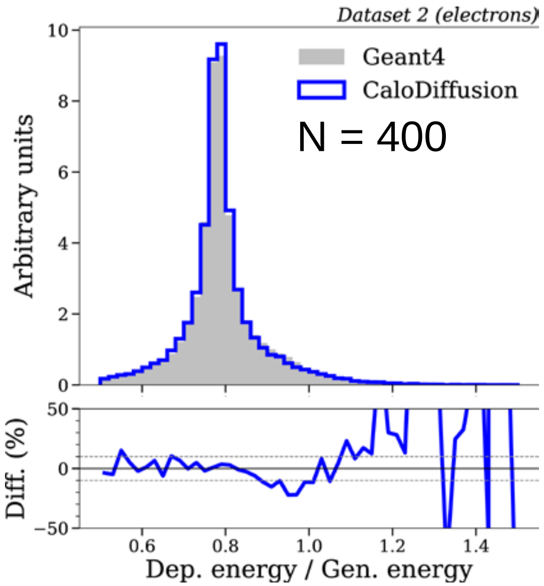
Dataset	Batch Size	Time/Shower [s] CPU	GPU
1 (photons)	1	9.4	6.3
(368 voxels)	10	2.0	0.6
	100	1.0	0.1
1 (pions)	1	9.8	6.4
(533 voxels)	10	2.0	0.6
	100	1.0	0.1
2 (electrons)	1	14.8	6.2
(6.5K voxels)	10	4.6	0.6
	100	4.0	0.2
3 (electrons)	1	52.7	7.1
(40.5K voxels)	10	44.1	2.6
	100	-	2.0

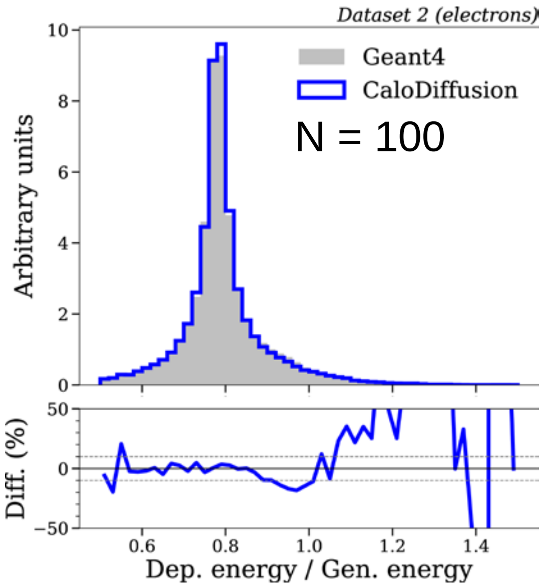
Num. Steps	Classifier AUC (low / high)	FPD	E Ratio Sep. Power
400	0.56 / 0.55	0.043(1)	0.011
200	0.61 / 0.56	0.046(1)	0.036
100	0.69 / 0.59	0.065(3)	0.079
50	0.83 / 0.67	0.110(4)	0.251

Improvement: More Diffusion!

- Train LayerDiffusion to predict energy deposited per layer (1D diffusion)
  - Negligible inference time (200 steps) compared to CaloDiffusion
- Normalize CaloDiffusion output based on LayerDiffusion
  - Only if both models predict sufficiently non-zero deposited energy in a layer
- ➢ Substantial improvement in total energy modeling
- Number of CaloDiffusion steps can be reduced with no loss of quality
  - 4× speedup for Dataset 2! (8× for Dataset 1 & improves low-energy pions)





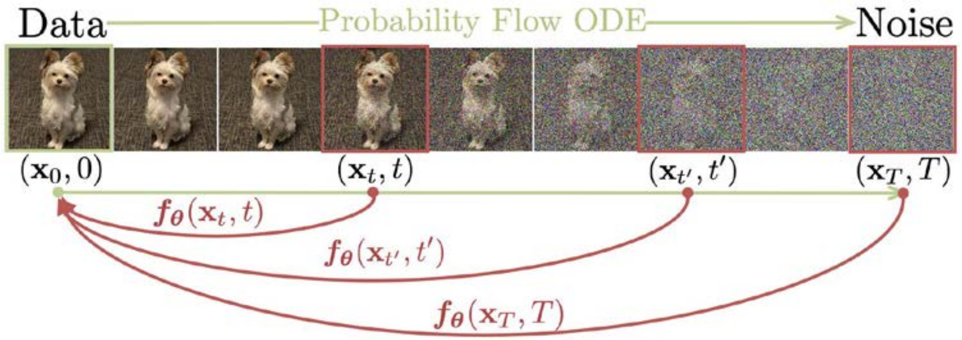


Model (2, electrons)	AUC (low / high)	FPD	KPD	E Ratio Sep. Power
Orig. (N = 400)	0.56 / 0.56	0.043	0.0001	0.011
Layer (N = 400)	0.54 / 0.58	0.045	0.00005	0.0017
Layer (N = 100)	0.54 / 0.60	0.076	0.0003	0.0017

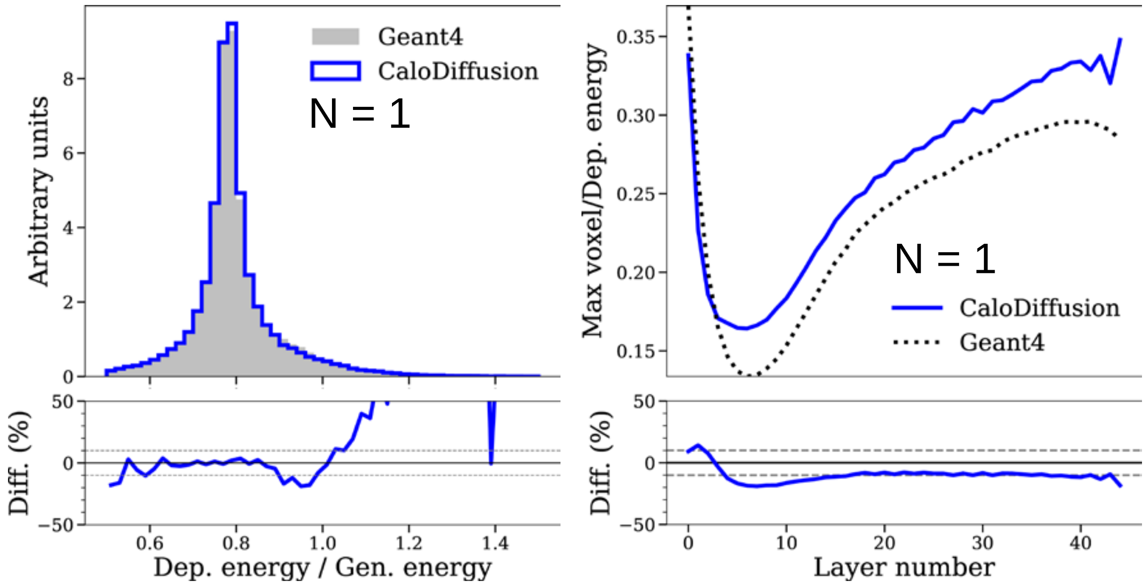
Consistency Model

arXiv:2303.01469

- Map any time step in diffusion process to origin
  - Trained via distillation from full diffusion model
- Allows single-step sampling with high quality output
  - Can improve quality by increasing sampling steps: "add back" some noise after each step
- Good agreement in some distributions, but worse in others
  - Disagreements mostly occur in higher-order quantities
  - Metrics worsen accordingly:
    - AUC 0.73 / 0.85, FPD 0.75, KPD 0.007
- Work in progress!
  - Multistep sampling for consistency models requires some optimization
  - Other techniques also being investigated



dataset 2 (electrons)



Outlook

- CaloDiffusion: bleeding-edge industry models and techniques + particle physics domain knowledge
  - Denoising diffusion architecture; sophisticated objectives, training schedule, sampling algorithm
  - Conditional cylindrical convolutions and GLaM for irregular geometries
  - Published in *Phys. Rev. D* 108 (2023) 072014
- *Leading performance* on virtually every CaloChallenge metric assessed so far
- Already significant improvement in a few initially suboptimal areas
  - LayerDiffusion for energy modeling
    - Enables substantial reduction in diffusion steps: quality impacts speed!
  - Consistency models for single-step generation
    - O(100)× speedup; stay tuned for further quality improvements…
- Future work:
  - Explore other speedup methods, such as progressive distillation or latent diffusion
  - Scale up to even higher-dimensional datasets, e.g. CMS HGCal

Backup

Acknowledgments etc.

- This work was performed with support of the U.S. CMS Software and Computing Operations Program under the U.S. CMS HL-LHC R&D Initiative.
- Additional support provided by the Fermi National Accelerator Laboratory, managed and operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy
- Thanks to Raghav Kansal for assistance in computing FPD and KPD
- Thanks to the CaloChallenge organizers for providing datasets and evaluation code

- Code for *Phys. Rev. D* 108 (2023) 072014 can be found at: https://github.com/OzAmram/CaloDiffusionPaper

Metrics

- Speed only matters if needed accuracy is achieved
  - Wrong answers can be obtained infinitely fast
- Looking at 1D histograms: not good enough!
  - Can miss high-dimensional correlations
- Best category: **integral probability metrics**

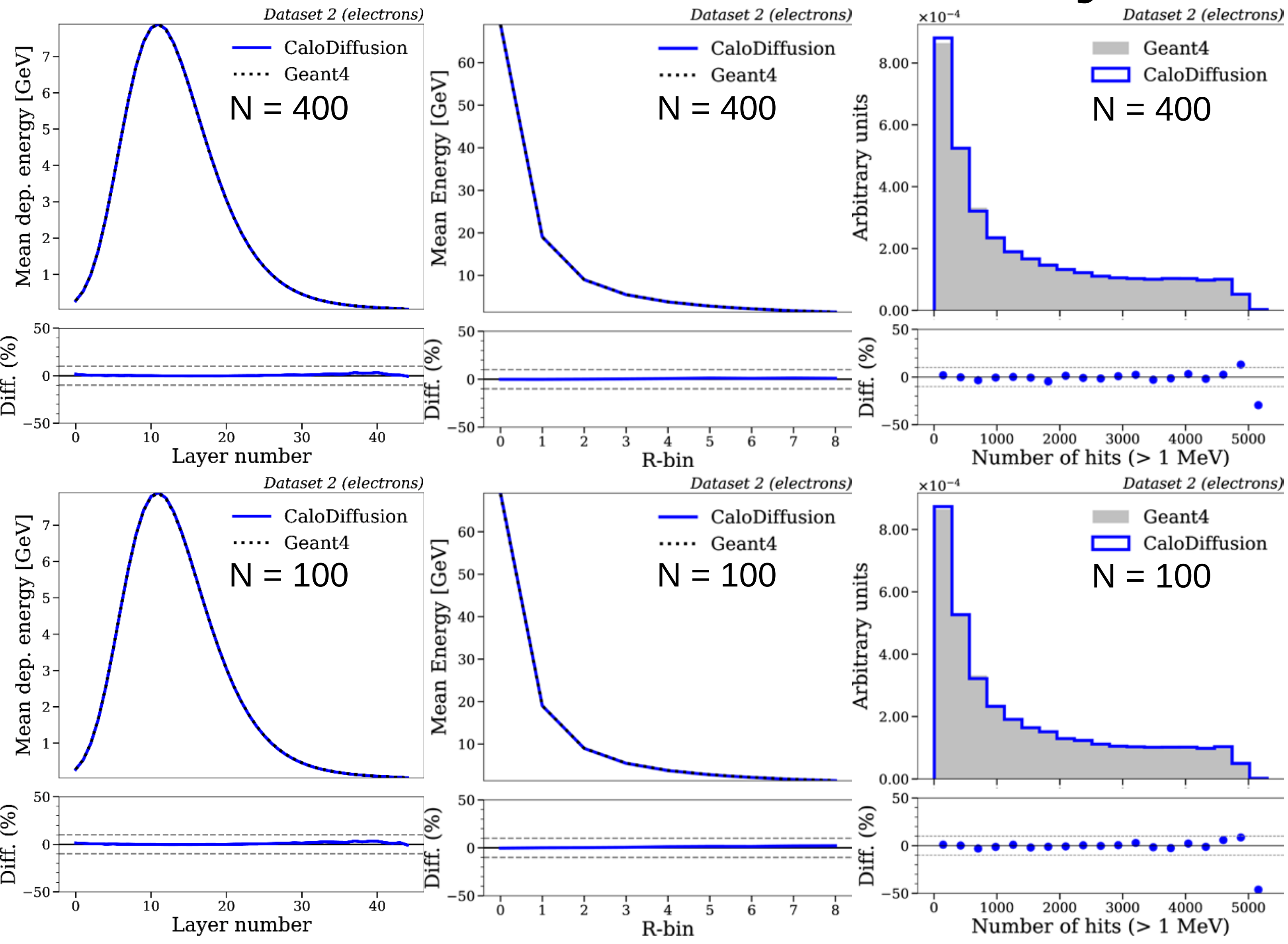
$$D_{\mathcal{F}}(p_{\text{real}}, p_{\text{gen}}) = \sup_{f \in \mathcal{F}} \left| \mathbb{E}_{\mathbf{x} \sim p_{\text{real}}} f(\mathbf{x}) - \mathbb{E}_{\mathbf{y} \sim p_{\text{gen}}} f(\mathbf{y}) \right|$$

  - *Wasserstein distance* W_1: $\mathcal{F}$ is set of all K-Lipschitz functions
    - Only works well in 1D, biased in high-D
  - *Maximum mean discrepancy* (MMD): $\mathcal{F}$ is unit ball in reproducing kernel Hilbert space
    - Depends on choice of kernel
  - *Fréchet distance*: W_2 distance between Gaussian fits to (high-D) feature space
    - Features can be hand-engineered or obtained from NN activations
- Another interesting category: *classifier scores*
  - Train NN to distinguish real vs. generated
  - AUC score ranges from 0.5 to 1.0
- *Fréchet Particle Distance* most clearly distinguishes between two similar approaches (message passing GAN and generative adversarial particle transformer)

	FPD $\times 10^3$	KPD $\times 10^3$	W_1^M $\times 10^3$
Truth	0.08 ± 0.03	-0.006 ± 0.005	0.28 ± 0.05
MPGAN	$\mathbf{0.30 \pm 0.06}$	$\mathbf{-0.001 \pm 0.004}$	$\mathbf{0.54 \pm 0.06}$
GAPT	0.66 ± 0.09	0.001 ± 0.005	0.56 ± 0.08

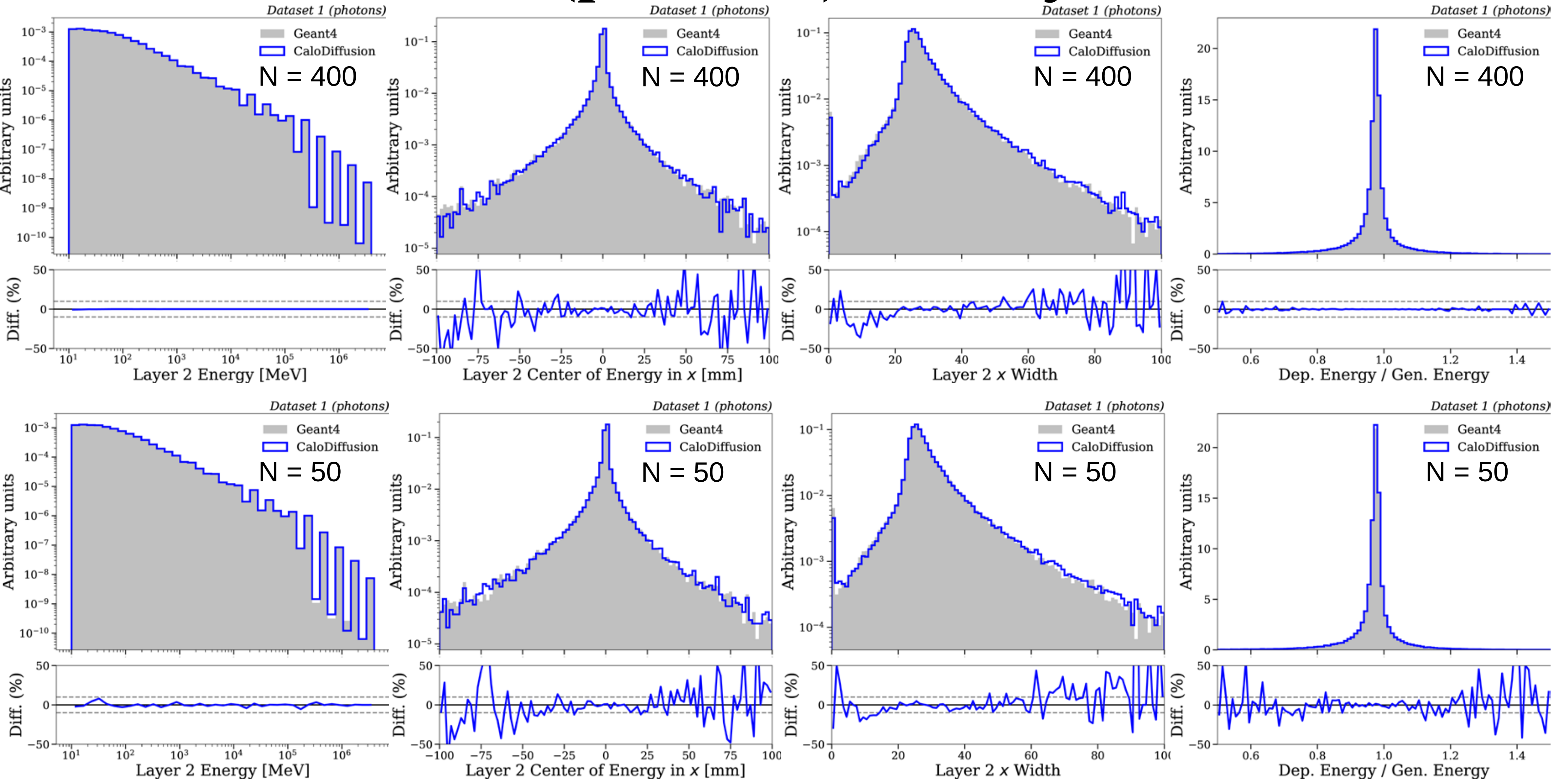
arXiv:2211.10295

Dataset 2 w/ LayerDiffusion

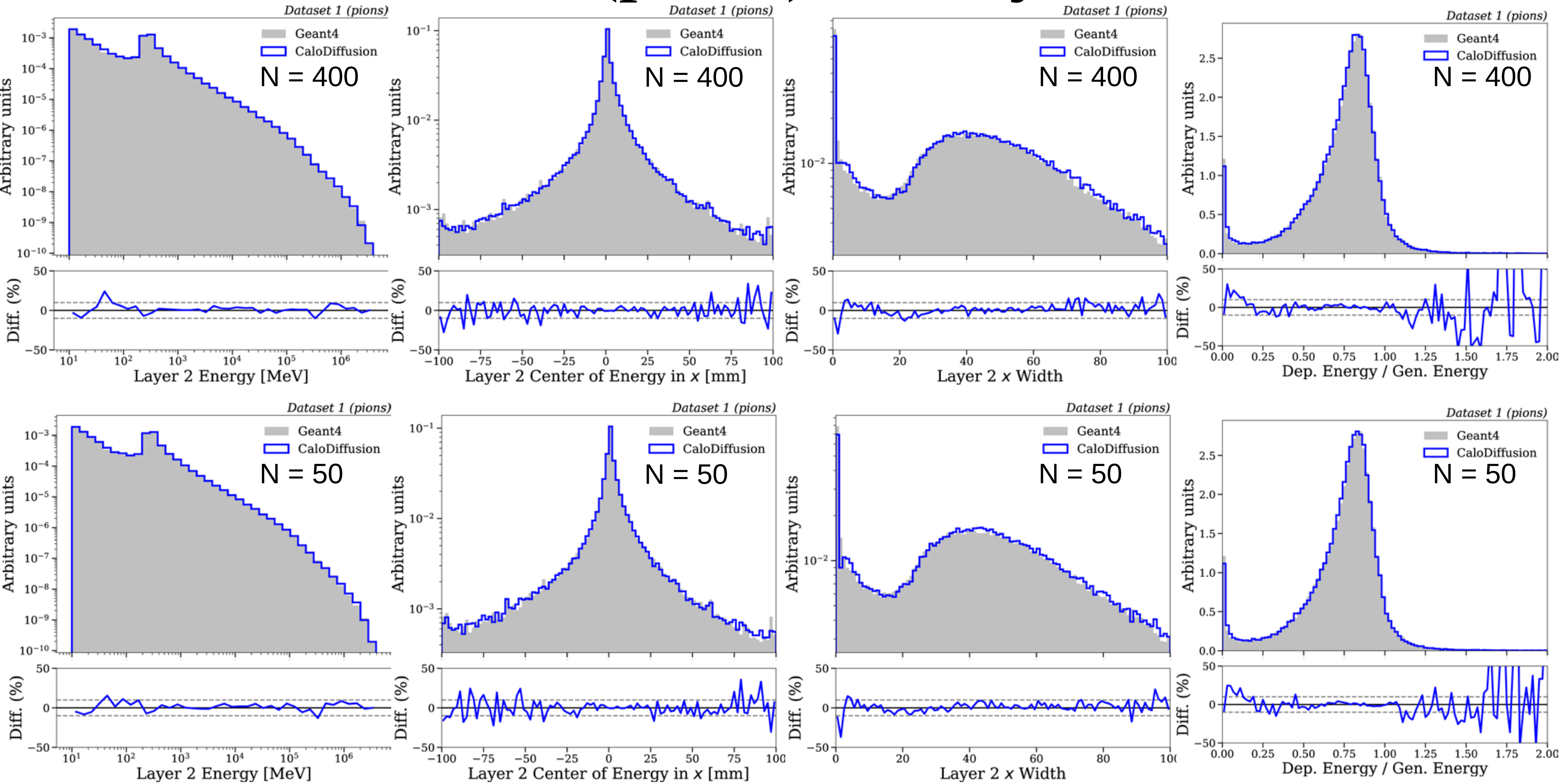


- Virtually indistinguishable for 4× fewer diffusion steps
- Improved agreement vs. original CaloDiffusion

Dataset 1 (photons) w/ LayerDiffusion



Dataset 1 (pions) w/ LayerDiffusion



Dataset 1 Metrics

Model (1, photons)	**AUC (low / high)**	**FPD**	**KPD**	**E Ratio Sep. Power**
Orig. (N = 400)	0.62 / 0.62	0.014	0.004	0.025
Layer (N = 400)	0.55 / 0.66	0.045	0.012	0.000005
Layer (N = 50)	0.60 / 0.65	0.038	0.010	0.0005

Model (1, pions)	**AUC (low / high)**	**FPD**	**KPD**	**E Ratio Sep. Power**
Orig. (N = 400)	0.65 / 0.65	0.029	0.004	0.010
Layer (N = 400)	0.63 / 0.65	0.040	0.004	0.0008
Layer (N = 50)	0.62 / 0.66	0.044	0.005	0.0007