

# CaloDiT-2 updates

Peter McKeown<sup>1,\*</sup>, Piyush Raikwar<sup>1</sup>, Anna Zaborowska<sup>1</sup>

<sup>1</sup> CERN

\*peter.mckeown@cern.ch

**HSF / IML CaloChallenge Meeting**

**01.12.2025**



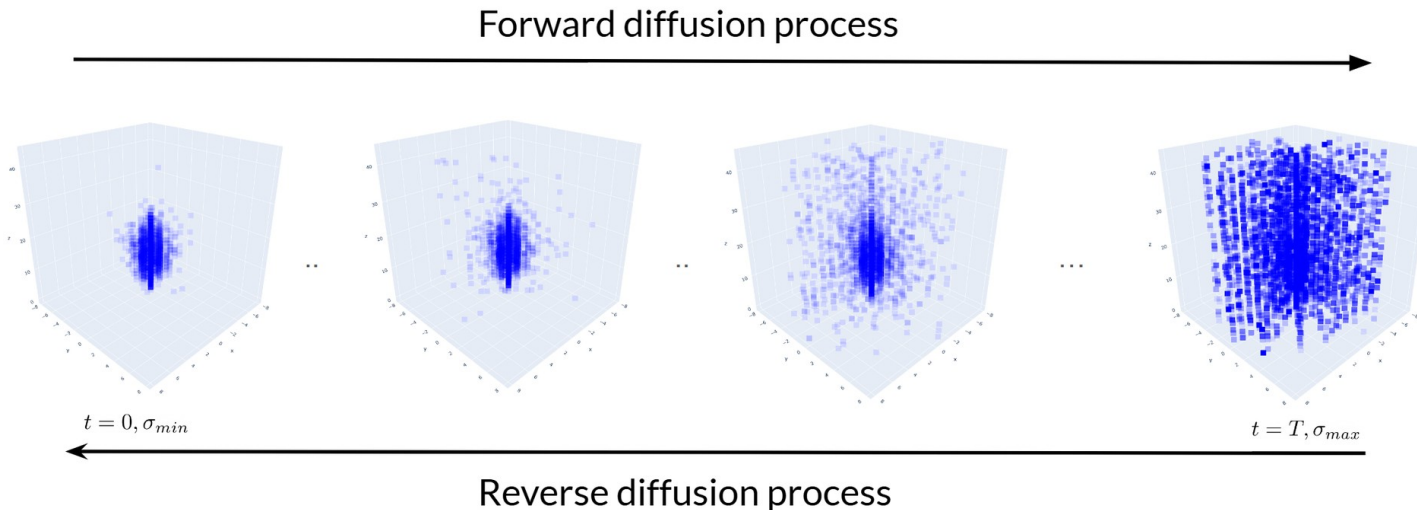
EP-SFT  
Software Frameworks and Tools



# Diffusion Models

- Class of generative model that relies on mapping data to a latent space via the **gradual addition of noise**
- **Diffusion process** adds noise to data:  $dx_t = \mu(x_t, t)dt + \sigma(t)dw_t$ 
  - ← Wiener Process
  - ← Drift & diffusion coeff.
- **Reverse diffusion**: trained model removes noise at each step to generate a new image

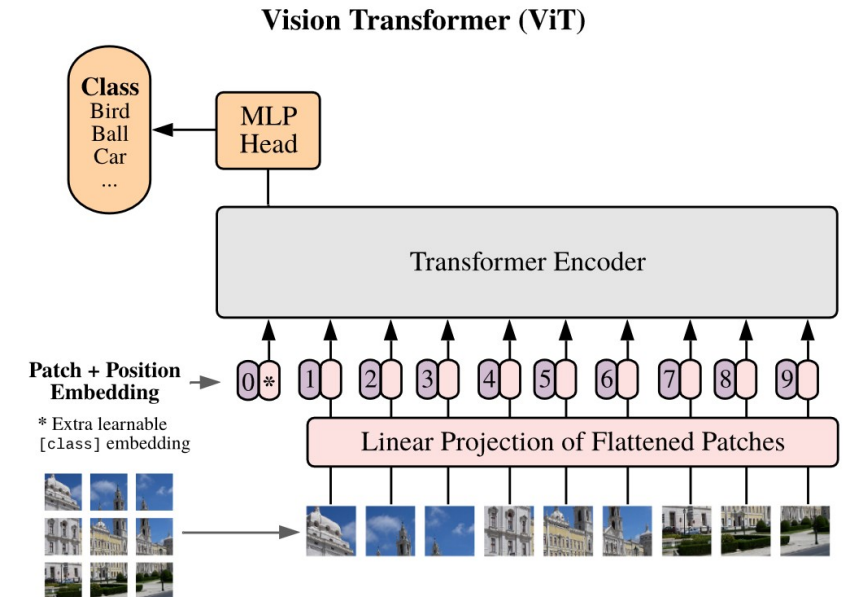
→ “Probability Flow” ODE:  $dx_t = \left[ \mu(x_t, t) - \frac{1}{2} \sigma(t)^2 \nabla \log p_t(x_t) \right] dt$



# Vision Transformers

- **Attention** mechanism highly successful in Natural Language Processing applications – **Transformers**
- **Vision Transformer (ViT)** approach allows scaling up to images:
  - Create **patches** over image
  - Create **tokens**: Combine **patches** with information about **position** in image
- **CaloDiT-2**: uses a combined [diffusion-transformer](#) approach

Figure: [arXiv:2010.11929](https://arxiv.org/abs/2010.11929)



# CaloDiT-2

- Initial CaloDiT implementation included in the CaloChallenge was very preliminary
- **CaloDiT-2**: Multiple important architectural improvements including:
  - Switching from **DDPM** (discrete-time) to **EDM** (continuous time) diffusion process
  - Incorporation of **consistency distillation (CD)**
    - Enables **single step** inference
- More details in **paper**: [arXiv:2509.07700](https://arxiv.org/abs/2509.07700)

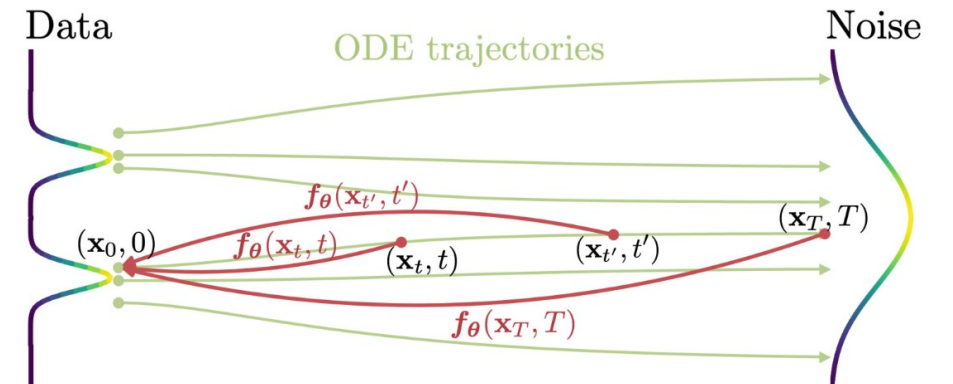
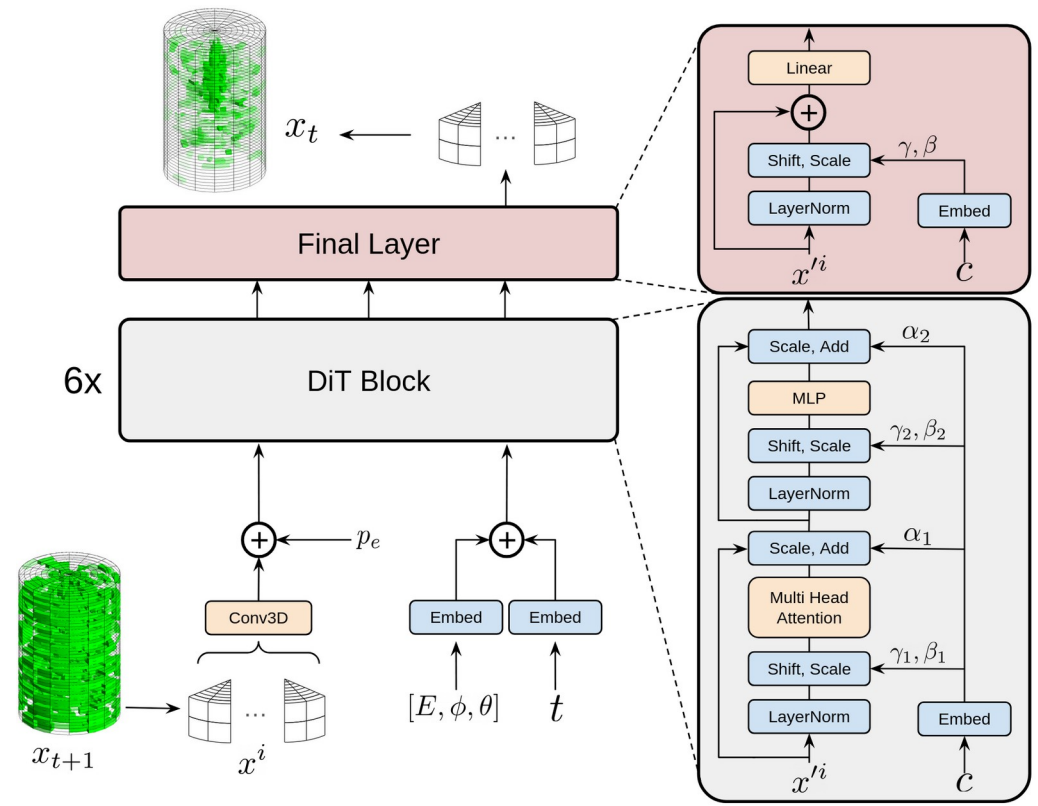


Figure: [arXiv:2303.01469](https://arxiv.org/abs/2303.01469)

# Updated CaloChallenge Dataset 2 Results

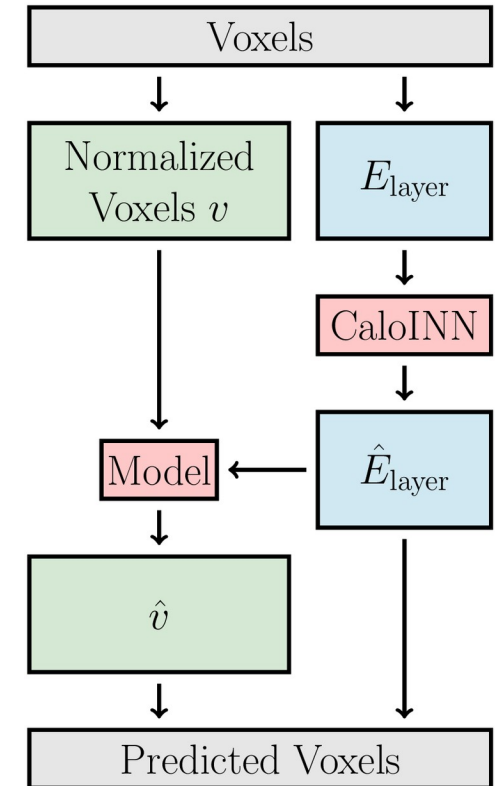
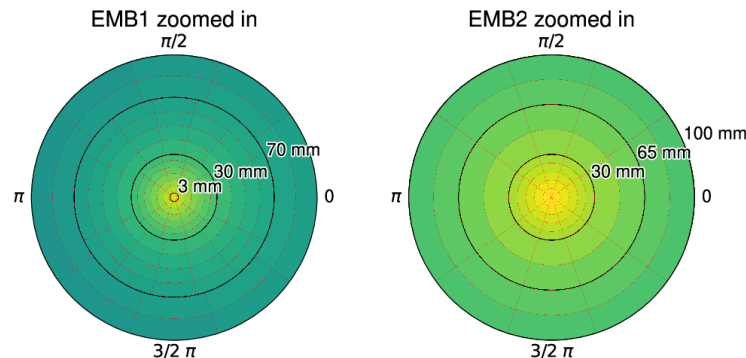
- **CaloDiT-2** model shows **major improvements** with respect to original **CaloDiT-1** included in the CaloChallenge
- CaloDiT-2 trained on **dataset 2** now **competitive**
- Comparison with standard CaloChallenge metrics. CPU/GPU benchmark now performed with an **AMD EPYC 9334 CPU** and an **NVIDIA RTX 6000 Ada GPU** in a pythonic environment. CPU benchmark performed on a single core with a batch size of one

paper: [arXiv:2509.07700](https://arxiv.org/abs/2509.07700)

Models \ Metrics	AUC low-level	AUC high-level	FPD ( $\times 10^3$ )	KPD ( $\times 10^3$ )	CPU time (ms)	GPU time (ms)
CaloDiT-2 EDM	$0.5939 \pm 0.002$	$0.5598 \pm 0.005$	$20.06 \pm 0.74$	$0.089 \pm 0.09$	$6349.0 \pm 7.6$	$171.8 \pm 0.13$
CaloDiT-2 CD	$0.5975 \pm 0.002$	$0.6947 \pm 0.004$	$68.83 \pm 3.37$	$0.39 \pm 0.15$	$101.2 \pm 0.04$	$2.9 \pm 0.02$
CaloDiT-1 DDPM	$0.984 \pm 0.001$	$0.912 \pm 0.001$	$1690.98 \pm 6.77$	$11.03 \pm 0.43$	$17322.9 \pm 33.9$	$639.96 \pm 2.4$

# CaloDiT-2 For ATLAS

- Collaboration with the **ATLAS** Simulation Group to **improve** the current **ATLFast3** (AF3) fast simulation framework
- See details in [Florian's talk](#) and recent ATLAS **public note**: [ATL-SOFT-PUB-2025-003](#)
- **Dedicated configuration** of CaloDiT-2 developed for ATLAS
  - Trained on **regular dataset** (1680 voxels) ~ 4x more than non-regular voxelisation
  - Combined with **CaloINN** model for learning **layer energies**

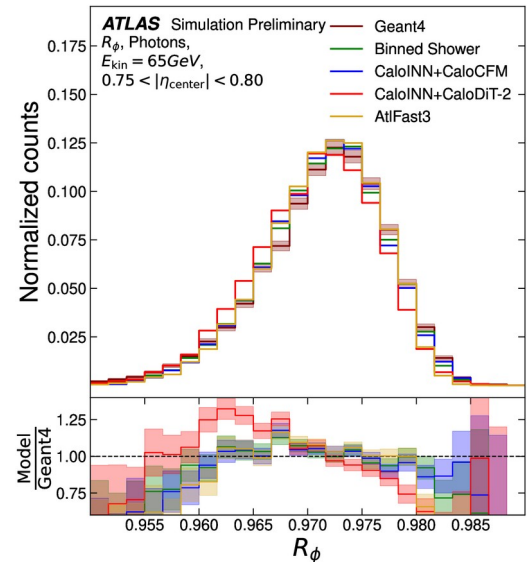
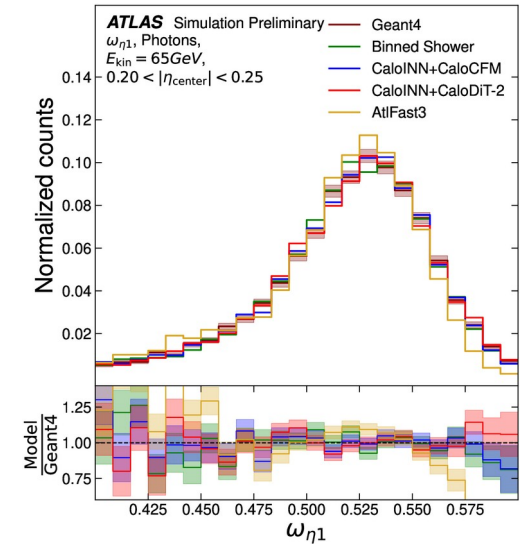


# CaloDiT-2 For ATLAS

ATL-SOFT-PUB-2025-003



- Both CaloDiT-2 and CaloCFM combined with CaloINN show **consistent improvements** with respect to current AF3
- In both cases, only **2 separate models** were required to cover the barrel ( $-1.35 < \eta < 1.35$ )
  - Current **AF3 requires 27 models** – improved memory efficiency
- Future directions:
  - Some more noticeable **deviations for higher energies** and more **complex detector regions**
    - Exploring **improved distillation** scheme
    - Implement **irregular patching** – operate on non-regular dataset (reduces required voxels by  $\sim x4$ )
  - Study use for **hadrons** – patching in CaloDiT-2 allows scaling to **significantly more voxels**



EP-SFT

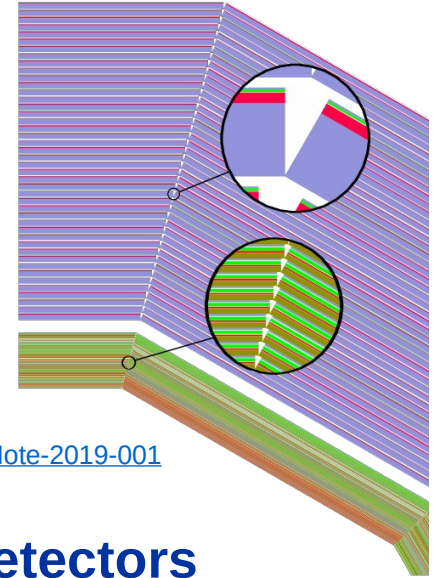
Software Frameworks and Tools

# Motivation for a Generic Approach

- Currently: development of a fast sim model for a new detector requires:
  - Lots of **compute resources**:
    - Producing Geant4 **training data**
    - **Developing model** (training, hyperparameter scans, etc.)
  - Dedicated **expertise and manpower**
- A generic approach requires:
  - A **generic shower representation**- capable of handling very **different detector geometries** – use virtual scoring mesh
  - A **generalisable ML model**
- Strategy:
  - **Pre-train** on a sufficiently large and diverse dataset of **different detectors**
  - **Adapt** quickly to new detectors

## CLD ECAL:

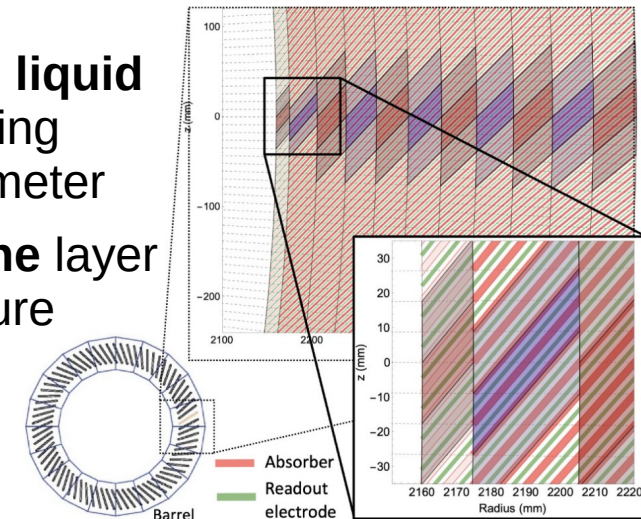
- **W-Si** sampling calorimeter
- **Planar** layer structure



E.g. FCC-ee detectors

## ALLEGRO ECAL: [EPJ Web Conf. 320 \(2025\) 00022](https://arxiv.org/abs/2501.00022)

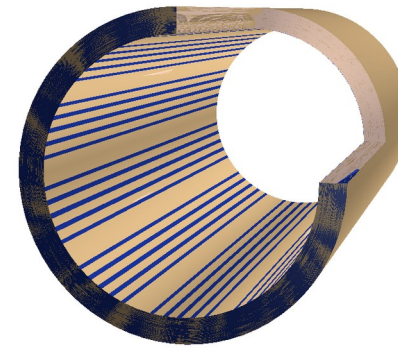
- **Nobel liquid** sampling calorimeter
- **Turbine** layer structure



# LEMURS Dataset



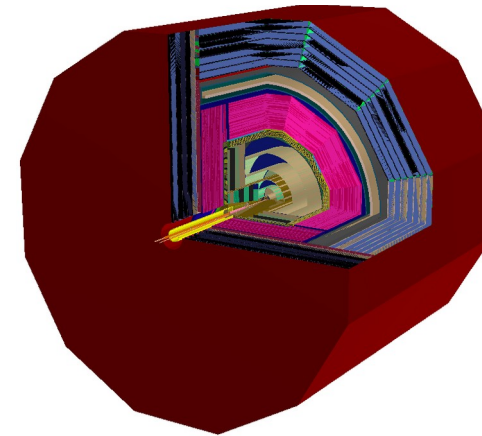
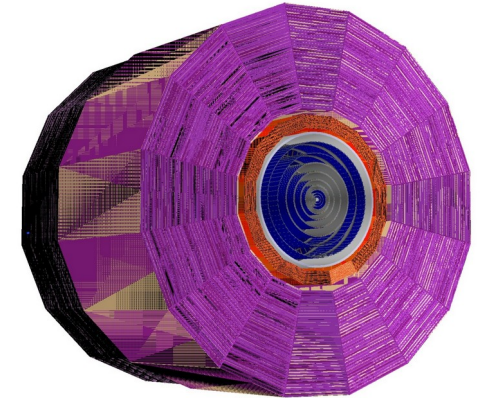
- **5 detectors** in total:
  - Par04 (Si-W ECAL)
  - Par04 (Sci-Pb ECAL)
  - The Open Data Detector (ODD) (Si-W ECAL)
  - FCC-ee CLD (Si-W ECAL)
  - FCC-ee ALLEGRO (LAr-Pb ECAL)
- **All datasets available:**  
[10.5281/zenodo.17076360](https://zenodo.org/record/17076360)
- Vary **incident energy** and (global) incident **angles** uniformly:
  - $E = 1 \text{ GeV} - 1 \text{ TeV}$  for Par04 + ODD: Hadron colliders
  - $E = 1 \text{ GeV} - 100 \text{ GeV}$  for FCCee:  $e^+e^-$  colliders
  - $\phi = -\pi - \pi$ ;  $\theta = 0.87 - 2.27 \text{ rad}$ .
- **1 million photon showers per detector**
- **8 sets of 1,000 showers discrete points per detector**



Par04  
(Si-W + Sci-Pb)

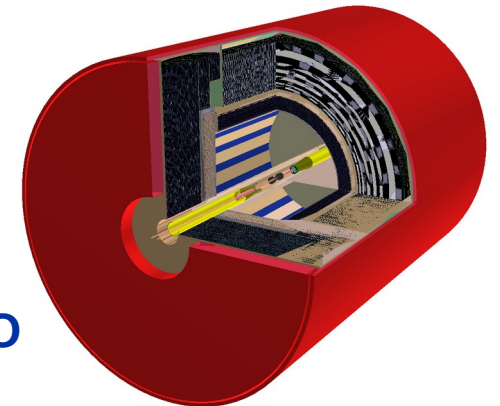
Using  
CaloChallenge  
dataset 2 mesh  
granularity

ODD



FCC-ee  
CLD

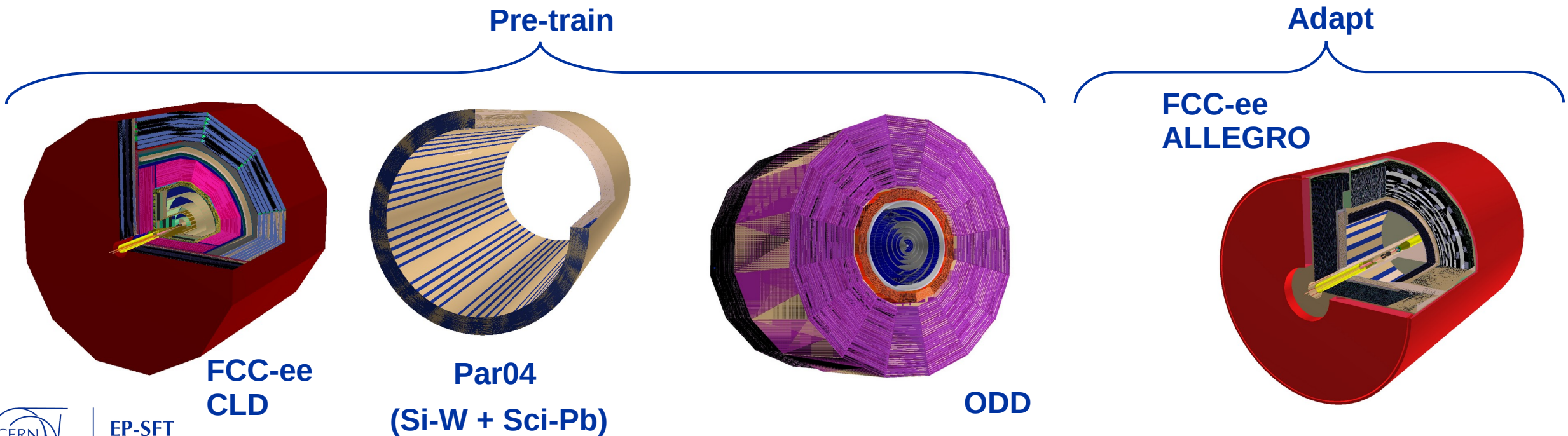
FCC-ee  
ALLEGRO



# Adaptation Study

paper: [arXiv:2509.07700](https://arxiv.org/abs/2509.07700)

- Need to pass information about detector to model- use additional **geometry condition G**
  - **Categorical** (one-hot) **encoding**
  - For **K detectors**, G has size **K + 1** to allow for new detector
- As demonstrator: **pre-train** on **4** detector geometries, and study **adaptation** performance onto **one** (FCC-ee ALLEGRO)



FCC-ee  
CLD

Par04  
(Si-W + Sci-Pb)

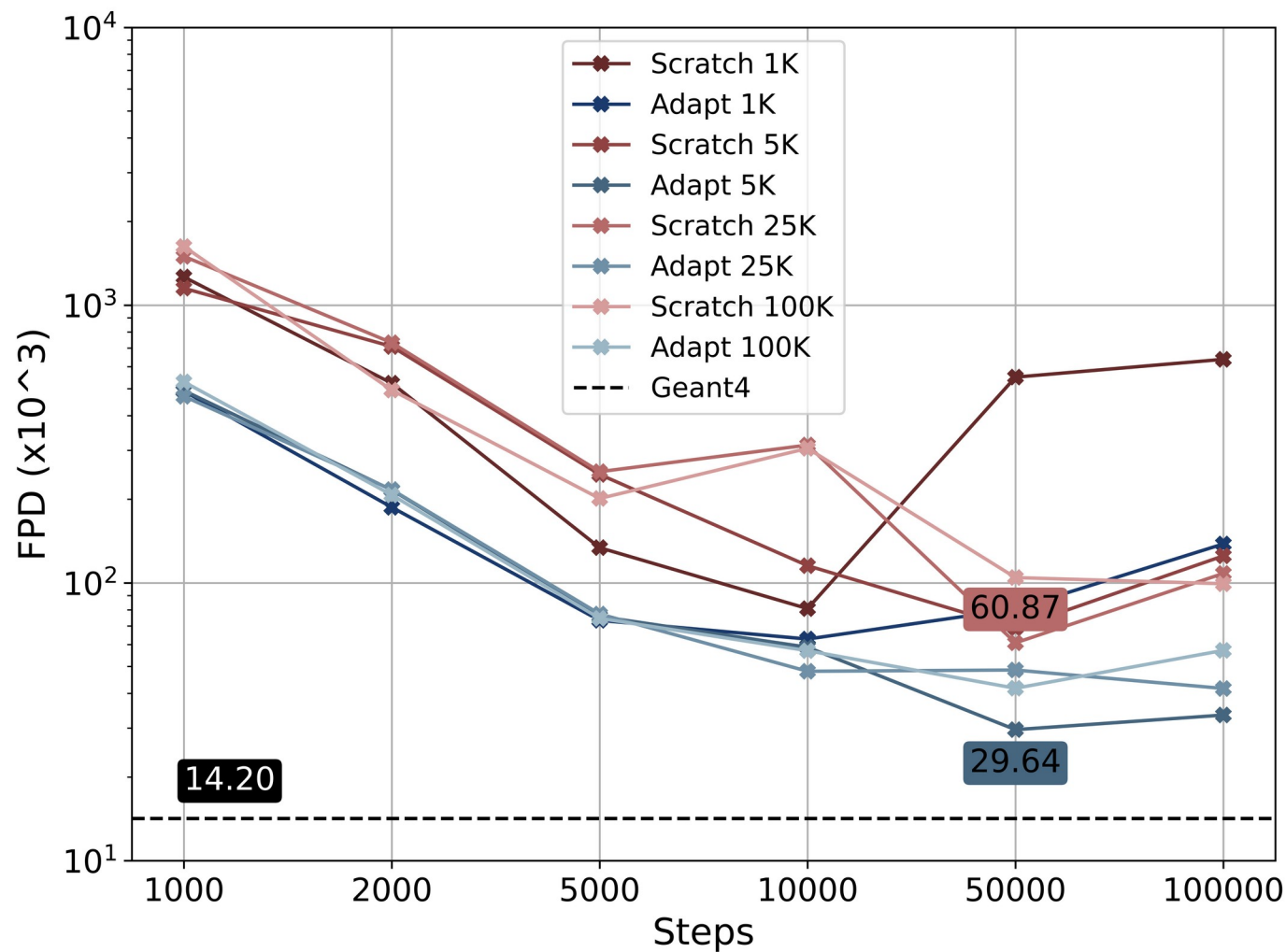
ODD

FCC-ee  
ALLEGRO

# Adaptation Results

- Compare **number of training steps** for **adaptation** vs training from **scratch** for **different dataset sizes**
- **Evaluating showers is complicated**: as example proxy metric, use **Fréchet Physics Distance (FPD)** for shower quality
- For a comparable FPD to training from scratch, adaptation requires:
  - **~20x less training steps\***
  - **~25x less data**
- Disclaimer: mileage may vary for different training/architecture configurations

Note: FPD is a relative measure!



\* including distillation

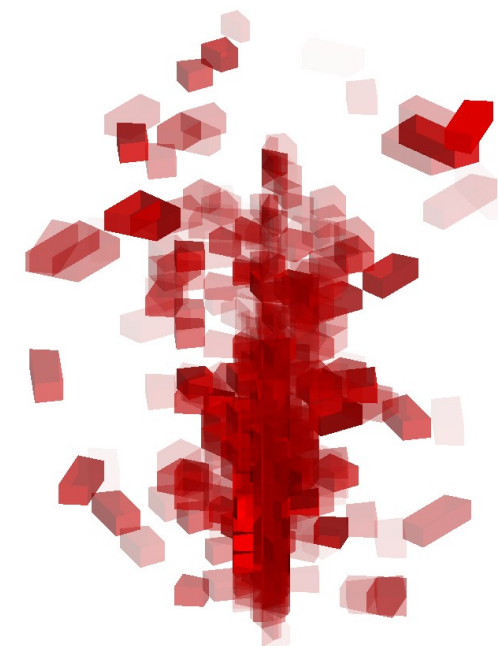
paper: [arXiv:2509.07700](https://arxiv.org/abs/2509.07700)



# Geant4 Integration

Par04 introduced  
in Geant4 11.0

- Our **pre-trained CaloDiT-2** has been released with **Geant4 11.4.0.beta**
  - In [examples/extended/parameterisations/Par04](#)
- Options to perform inference with both **OnnxRuntime** and **LibTorch**
- The **complete code** for model **training** and **conversion** for inference is available [here](#)
- To **apply** CaloDiT-2 to a **new** detector geometry, use the workflow provided to:
  - **Generate** (a reduced amount of) **new training data**, following Par04
  - **Adapt** the pre-trained diffusion model
  - Follow the **distillation** procedure, using the adapted diffusion model
  - **Convert** the distilled model into **ONNX** or **TorchScript** format with the provided scripts
  - **Integrate into Geant4** following the Par04 example
  - **Hit placement** has to be implemented by user (geometry specific) following given framework



**CaloDiT-2 shower  
from Geant4 Par04  
example**

# DD4hep Integration: DDML

- **DD4hep** toolkit widely used by future collider projects via common **Key4hep** turnkey software stack
- **Generic library** for running ML-based fast sim models in DD4hep: [DDML](#)
  - Follow Par04 for interfacing with Geant4
  - Can be used with **realistic, detailed detector geometries**
- Aim for easy to use library which can **accommodate all types of ML architectures**
- Library **included** in the **Key4hep** software stack

## Trigger

- Fast Sim trigger
  - e.g. particle type, energy, geometry

## Inference

- Concrete inference in C++
  - ONNX, LibTorch etc...

## Model

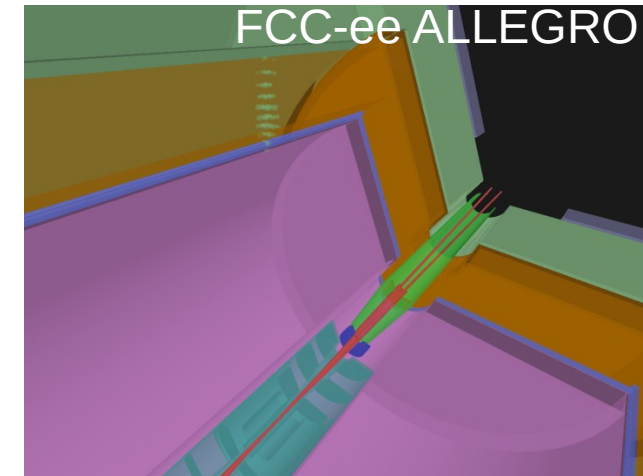
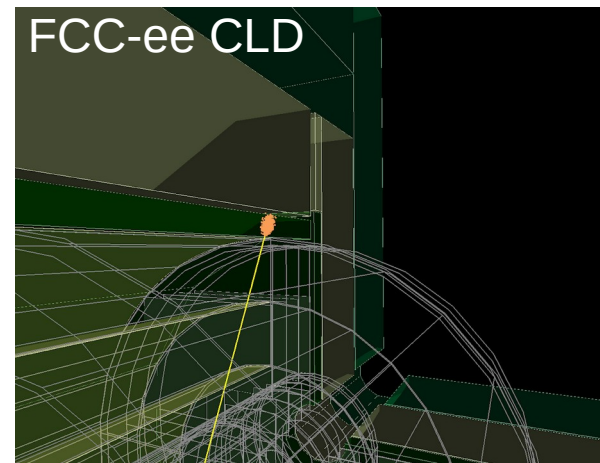
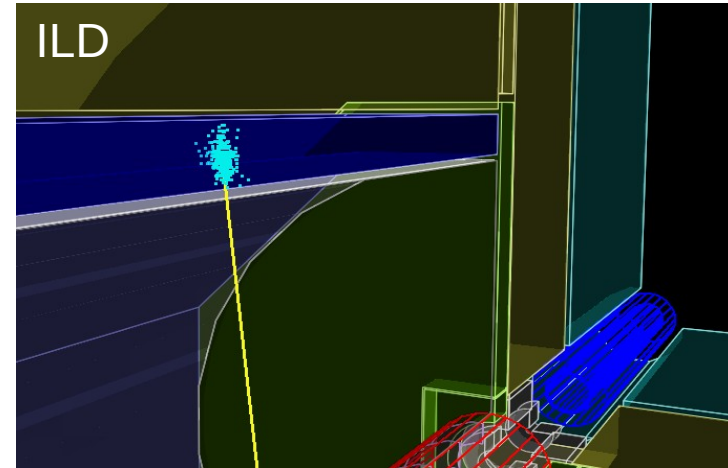
- Model-specific implementation of ML architecture
  - e.g. BIB-AE, Flow, Diffusion model

## Geometry

- Concrete placement in detector geometry
  - Endcap, barrel etc...

# DD4hep Integration: DDML

- Newly added support for **CaloChallenge-like mesh placements**
  - In addition to original layer-wise placement (ILD)
- **CaloDiT-2** validation for **FCC-ee CLD** and **ALLEGRO** ongoing
- Provides straightforward access to **realistic simulation applications** and **reconstruction workflows**
  - See [Martina's talk](#) for studies in ILD using DDML



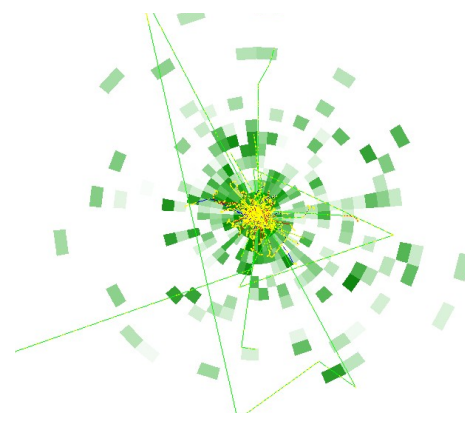
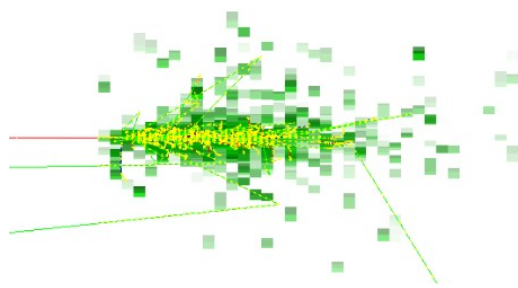
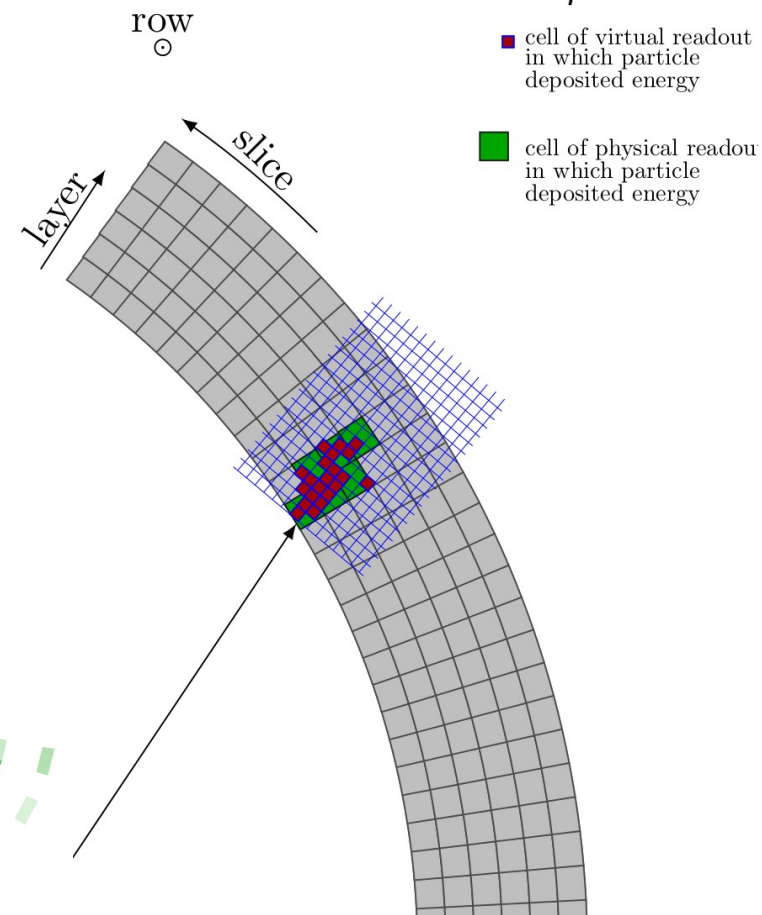
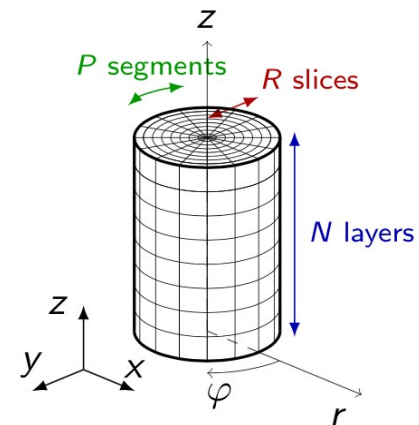
# Summary

- **CaloDiT-2** has **improved significantly** since original CaloChallenge
  - Competitive on dataset 2
- **Dedicated implementation** explored for **ATLAS**
  - **Initial work** also underway for **CMS HGCal** and **LHCb**
- **Generic approach** explored and **LEMURS** dataset containing 5 different detector geometries released
  - Demonstrated potential of **significantly reducing** required amount of **training steps** and **data**
  - **First generic** ML fast sim model **included** in the **Geant4** toolkit
- Developments in the **DDML** library for including **scoring mesh placements**
  - Provides access to **realistic simulation** and **reconstruction workflows**

# Backup

# Shower Representation

- Use **representation detached** from **readout geometry** of detector
  - **Consistent representation across detector** concepts
- **Virtual scoring mesh** to record energy deposits
  - From [Geant4 Par04](#) example- **basis of CaloChallenge** datasets 2 and 3
  - **Overlays** actual detector geometry
  - CaloChallenge dataset 2 size mesh:  $N \times P \times R = 45 \times 9 \times 16$
- **Cylindrical** mesh, aligned along the direction of the incident particle
- Offload difficulty from **model training** to **hit placement**



# Consistency Distillation

- Major bottleneck in inference time for model is **number of diffusion steps**
- **Consistency distillation (CD)**: transfer “knowledge” to target model requiring **fewer diffusion steps**
  - We reduce the model to a **single evaluation step**
- “**Self-consistent**”- function evaluation at any point on trajectory provides the same result:

$$f(x_t, t) = f(x_{t'}, t') \forall t, t' \in [\epsilon, T]$$

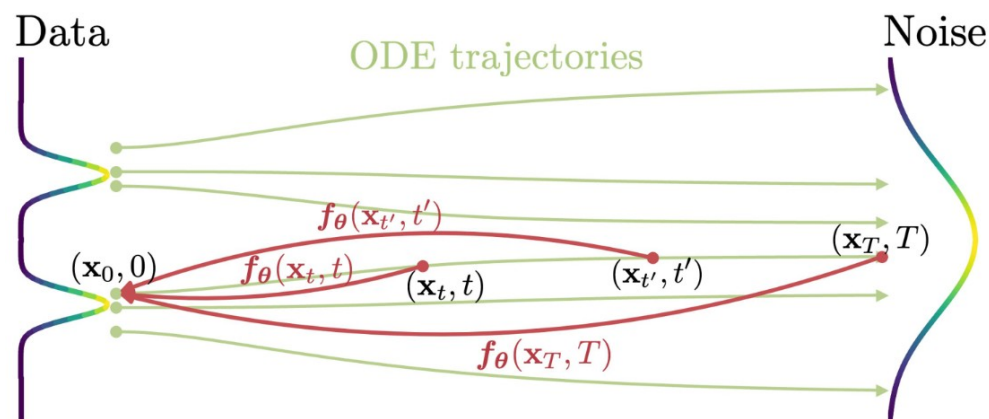
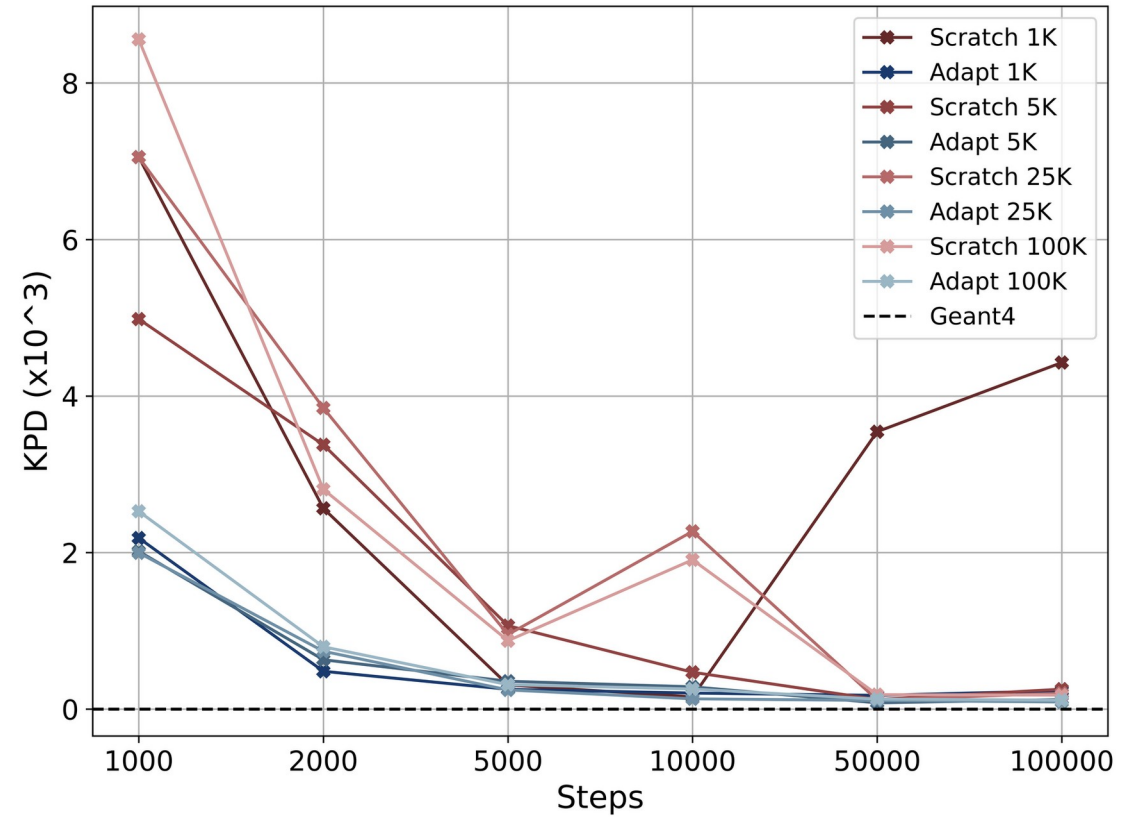
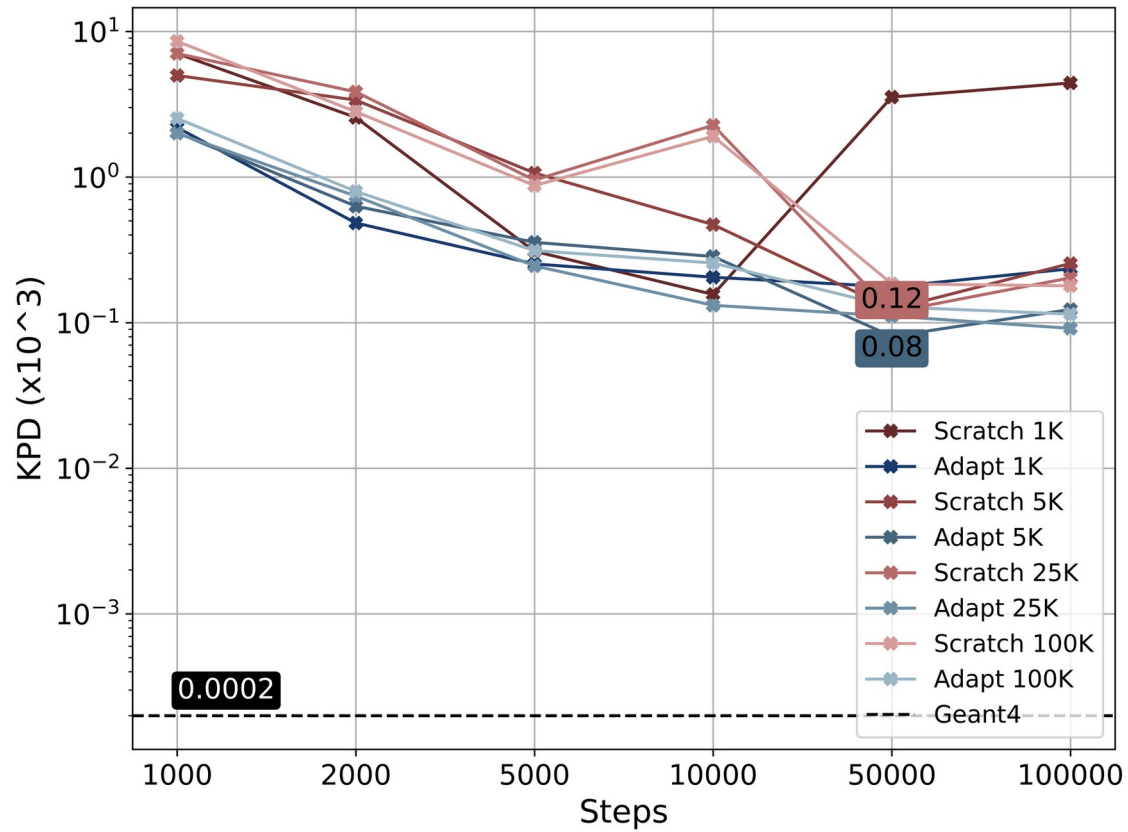


Figure: [arXiv:2303.01469](https://arxiv.org/abs/2303.01469)

# Updated CaloChallenge Dataset 2 Results

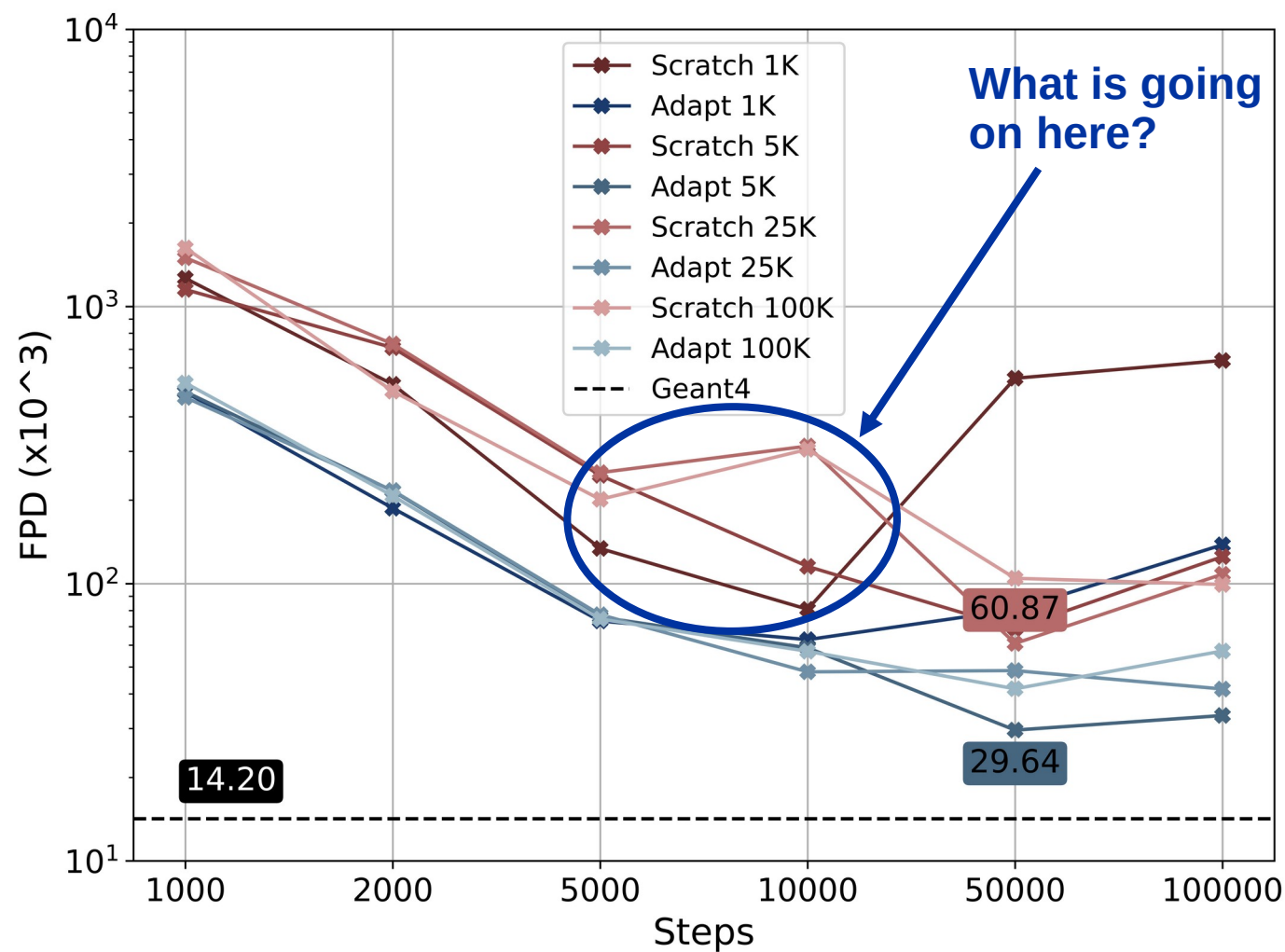
Models \ Metrics	AUC low-level	AUC high-level	FPD ( $\times 10^3$ )	KPD ( $\times 10^3$ )
CaloDiT-2 EDM	$0.5939 \pm 0.002$	$0.5598 \pm 0.005$	$20.06 \pm 0.74$	$0.089 \pm 0.09$
CaloDiT-2 CD	$0.5975 \pm 0.002$	$0.6947 \pm 0.004$	$68.83 \pm 3.37$	$0.39 \pm 0.15$
CaloDiT-1 DDPM [28]	$0.984 \pm 0.001$	$0.912 \pm 0.001$	$1690.98 \pm 6.77$	$11.03 \pm 0.43$
CaloDiffusion [23]	$0.577 \pm 0.004$	$0.591 \pm 0.009$	$146.933 \pm 0.87$	$0.174 \pm 0.04$
CaloDREAM [26]	$0.531 \pm 0.003$	$0.521 \pm 0.002$	$24.65 \pm 1.035$	$0.023 \pm 0.036$
CaloINN [19]	$0.743 \pm 0.002$	$0.865 \pm 0.003$	$732.83 \pm 5.33$	$2.82 \pm 0.42$
Calo-VQ [15]	$0.986 \pm 0.001$	$0.994 \pm 0.0$	$1315.72 \pm 7.03$	$8.52 \pm 0.5$

# Adaptation Results- KPD



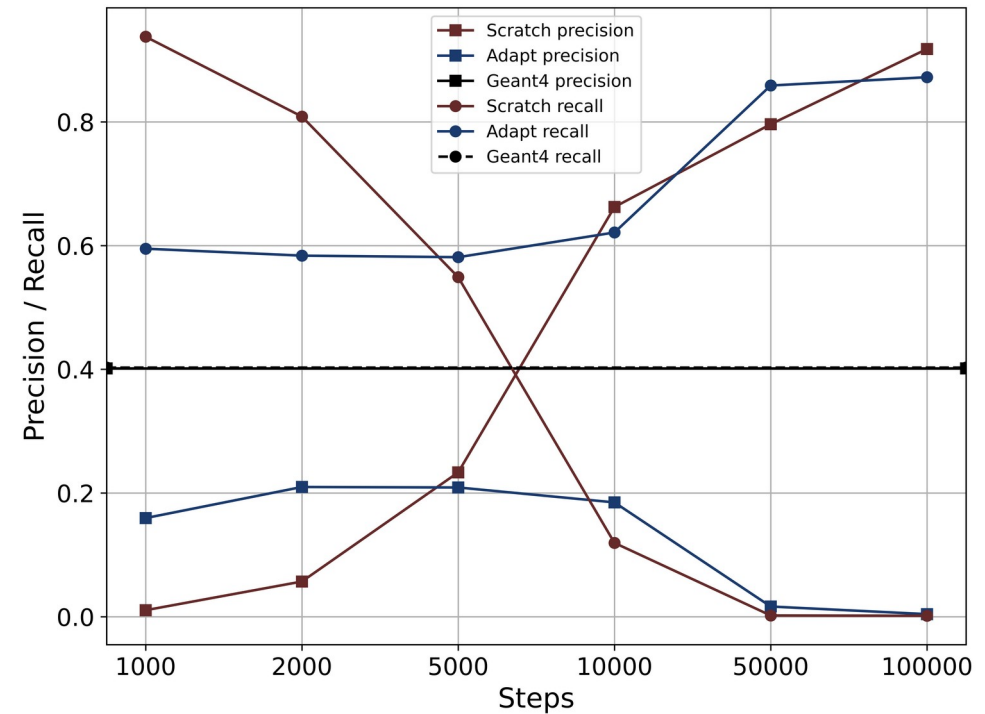
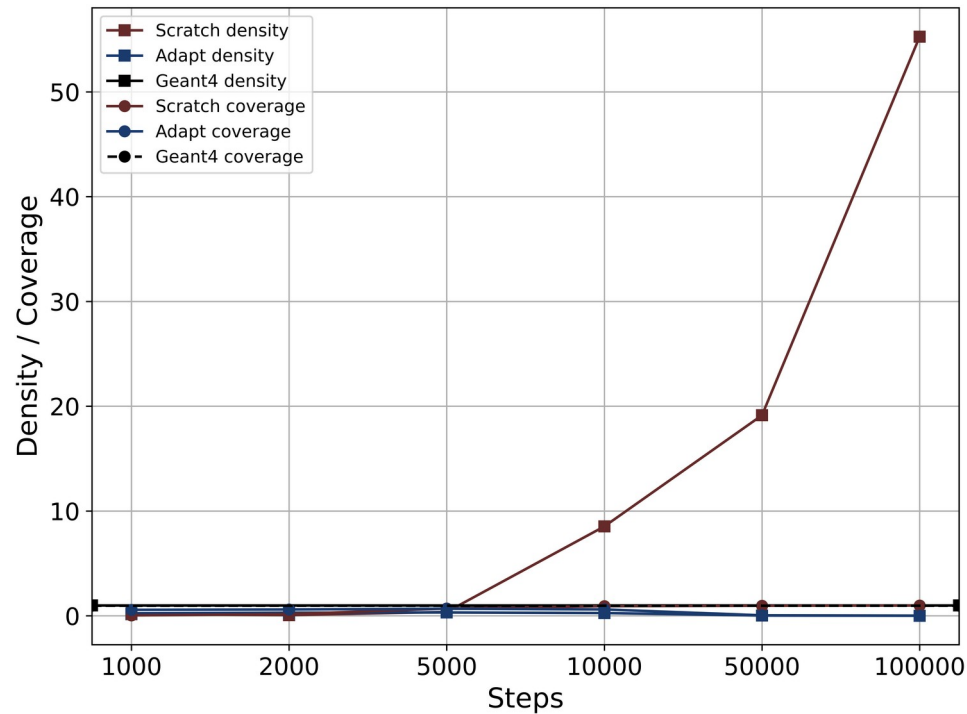
# Adaptation Results

- Compare **number of training steps** for **adaptation** vs training from **scratch** for **different dataset sizes**
- **Evaluating showers is complicated:** use **Fréchet Physics Distance (FPD)** as proxy metric for **shower quality**
- For a comparable FPD to training from scratch, adaptation requires:
  - **~20x less training steps**
  - **~25x less data**
- Disclaimer: mileage may vary for different training/architecture configurations



# Adaptation Results

- Precision = number of generated points on real manifold
- Recall = number of real points on generated manifold



- Model trained from **scratch** on very **limited data** (1K) quickly **memorises** showers- giving the appearance of better physics performance
- **Adaptation** much **more robust** against these effects