Cross-Geometry Fast Electromagnetic Shower Simulation

F. Gaede[2], G. Kasieczka[1],
L. Valente[1*]

[1] University of Hamburg, UHH
[2] Deutsches Elektronen-Synchrotron, DESY

KISS

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

CLUSTER OF EXCELLENCE
QUANTUM UNIVERSE

DESY.

* lorenzo.valente@uni-hamburg.de

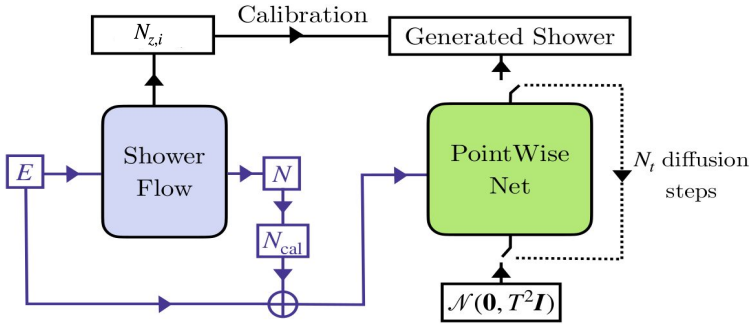
CaloClouds2 Model

1. Shower Flow

- Predicts point counts per layer for given incident energy

2. PointWise Net

- Generates the **4D point cloud** using Shower Flow's point counts

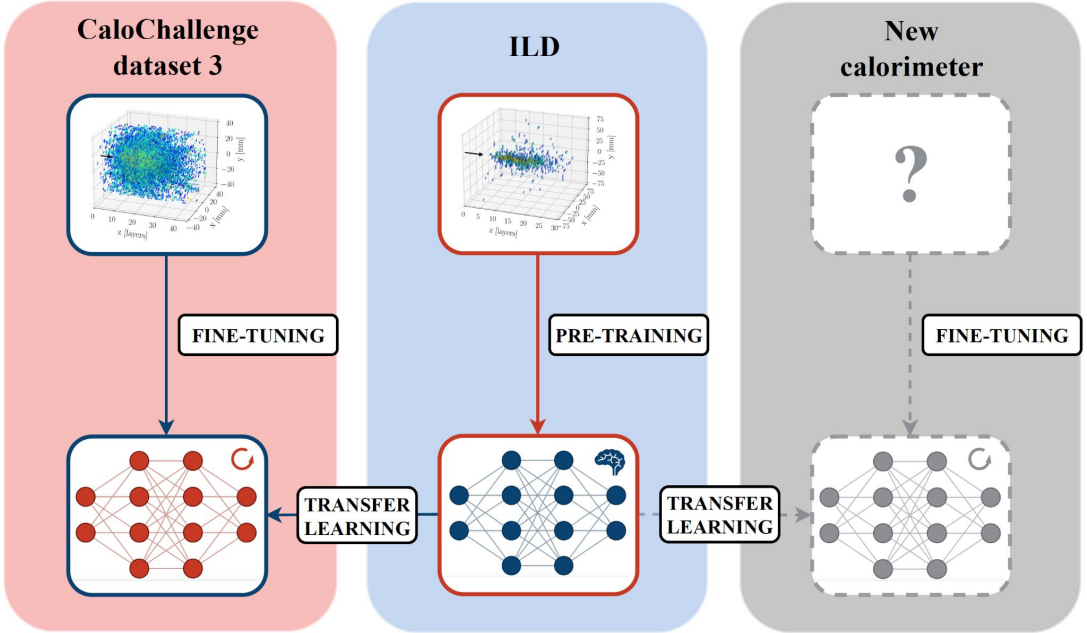


Sampling

E. Buhmann et al.
ArXiv: 2309.05704

Transfer Learning

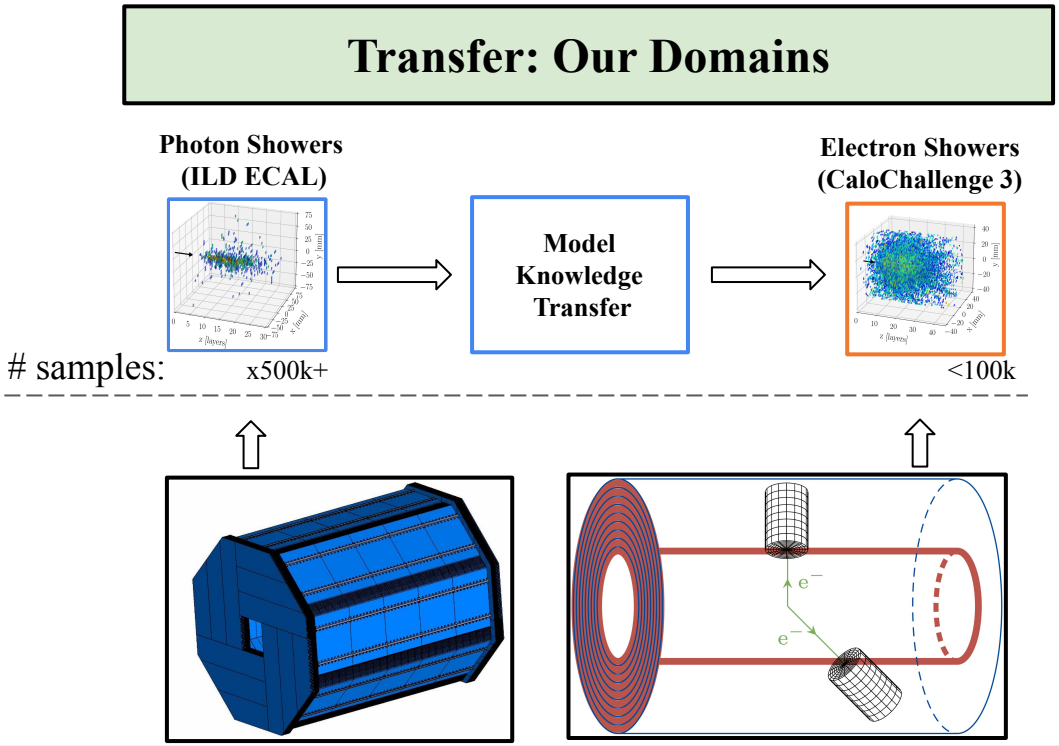
Leveraging knowledge from a pre-trained model (**source** task) to improve performance on a related task (**target** task)



Domains

Key Adaptation Challenges:

1. **Geometry mismatch**: Regular → Cylindrical
2. **Incident energy:**
   10-90 GeV → 1-1000 GeV
   Uniform → Log-normal
3. **Max number of points:** ~ 6k → ~ 20k
4. **Particle difference**: Photon → Electron showers
5. **Layer depth mismatch:** 30 → 45



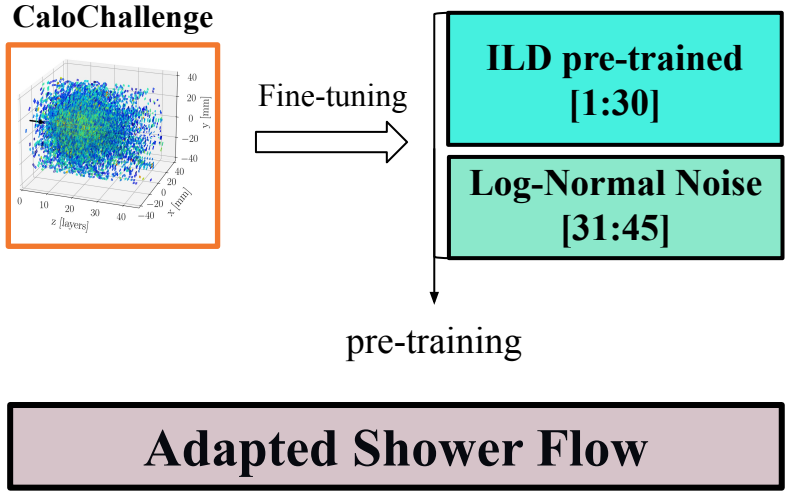
Fine-Tuning: Shower Flow

Challenges:

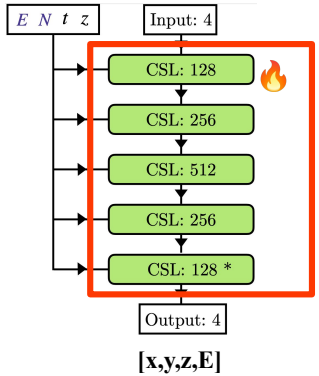
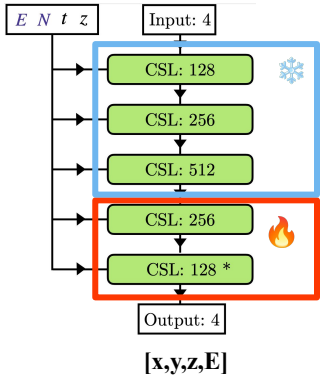
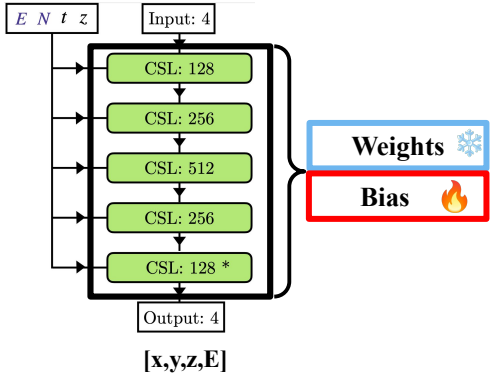
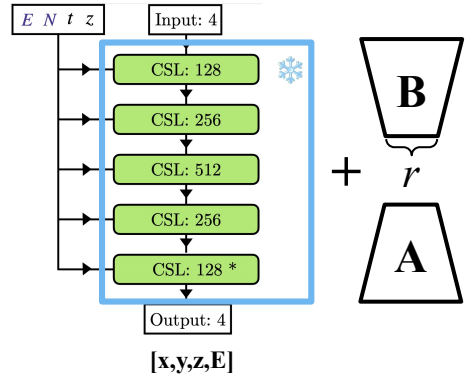
- Shower Flow architecture is **rigid by design** for adapting to different geometries

Our Solution:

- **Preserved** original 30 calorimeter layer structure and weights from **ILD pre-trained backbone**
- **Extended** with 15 additional calorimeter layers to match **CaloChallenge's 45 layer geometry**
- **Fine-tuning** on CaloChallenge dataset



Fine-Tuning: PointWise Net

	Full fine-tuning	Top2 fine-tuning	BitFit	LoRA
				
# trainable parameters	**524K** (100%)	**221K** (44%)	**87K** (17%)	Depends on r , e.g. for $r = 106$: **272K** (52%)

Results

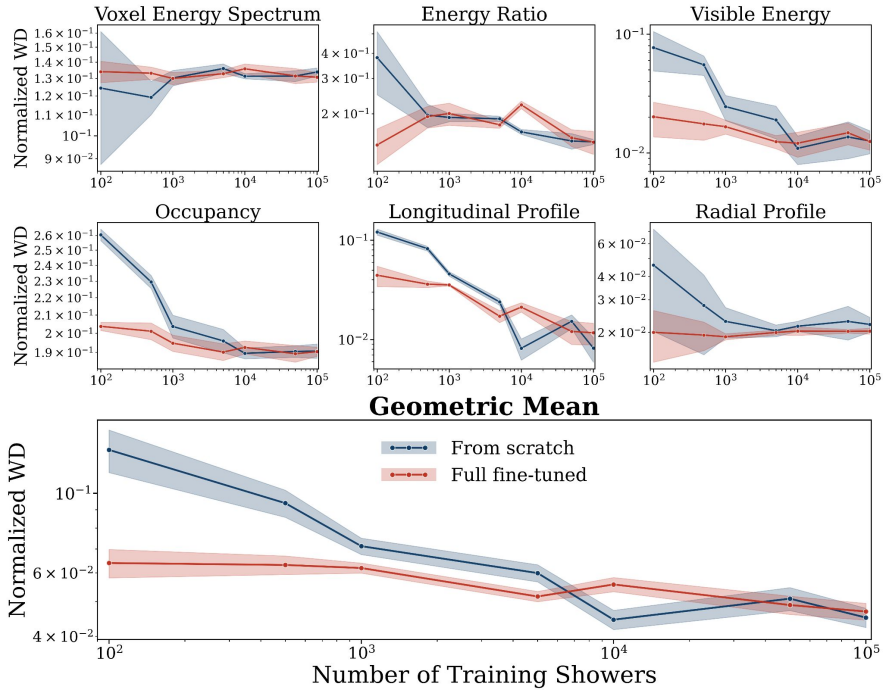
Geometric Mean:

$$\bar{y}_{jk} = \left(\prod_{i=1}^{6} y_{ijk} \right)^{1/6},$$

where each training method j across the six physical observables i is calculated for different training shower sizes k

✅ Data Efficiency

Fine-tuned versions matches *from scratch* performance with orders of magnitude fewer training samples



Results

✅ Consistent Performance

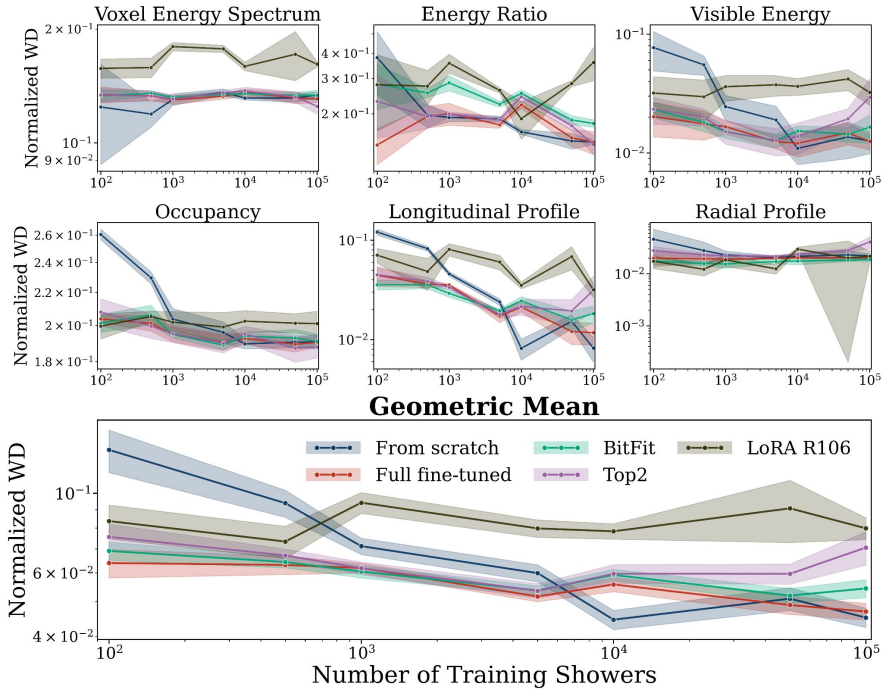
Parameter-efficient fine-tuning matches *full fine-tuned* performance with limited training data

⚠️ Known Limitation

LoRA shows performance degradation in high-data training regimes

⌛ Zero Inference Overhead

Parameter-efficient methods maintain original model inference speed



Conclusions

✅ What Works

- **44% better performance at 10^2 training samples** than *from scratch*
- **Some observables benefit most** - Visible Energy & Longitudinal/Radial profile transfer well
- **Full fine-tuning > parameter-efficient methods** - LoRA underperform

⚠️ Key Findings

- **Crossover at** $\sim 10^3 - 10^4$ **samples** - beyond this, *from scratch* catches up

Paper out soon! → 2511.XXXXX

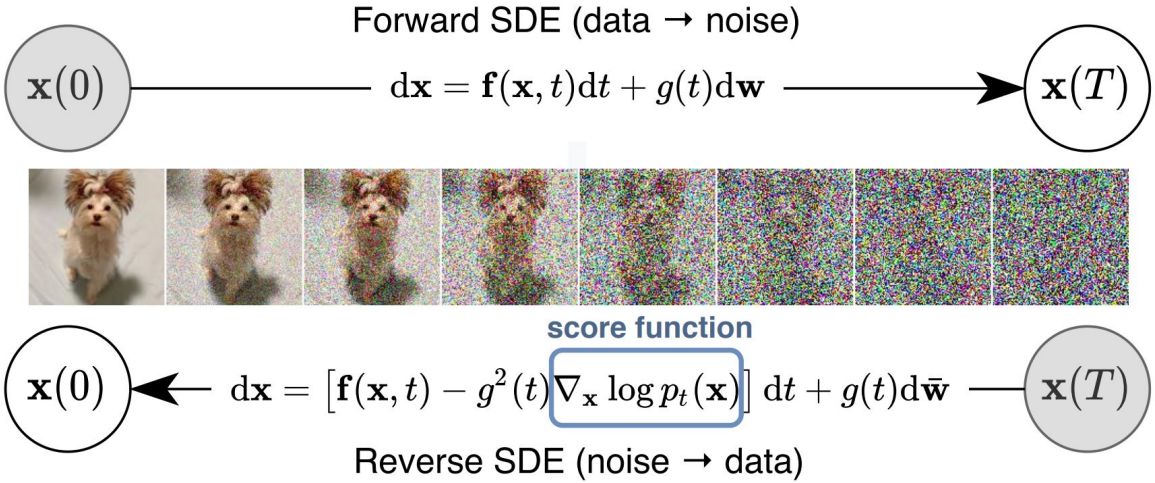
Thanks for the attention!

Contact:

lorenzo.valente@uni-hamburg.de

Backup

Diffusion Models



Song, Yang, et al.
ArXiv: 2011.13456

- **Forward Process:** Adding noise to input data → pure randomness
- **Reverse Process:** Removing noise step-by-step → generating structured data

Quantitative Measures

- **Normalised Wasserstein Distance:**

$$l_1(P,Q) = \inf_{\pi \in \Gamma(P,Q)} \int_{\mathbb{R}\times\mathbb{R}} |x-y|\, d\pi(x,y)$$

Where $\Gamma(P,Q)$ is the set of (probability) distributions on $\mathbb{R} \times \mathbb{R}$ whose marginals are P and Q on the first and second factors respectively. The final metric is normalised by the standard deviation σ_P of the Geant4 reference.

- **Quantile Kullback-Leibler divergence:**

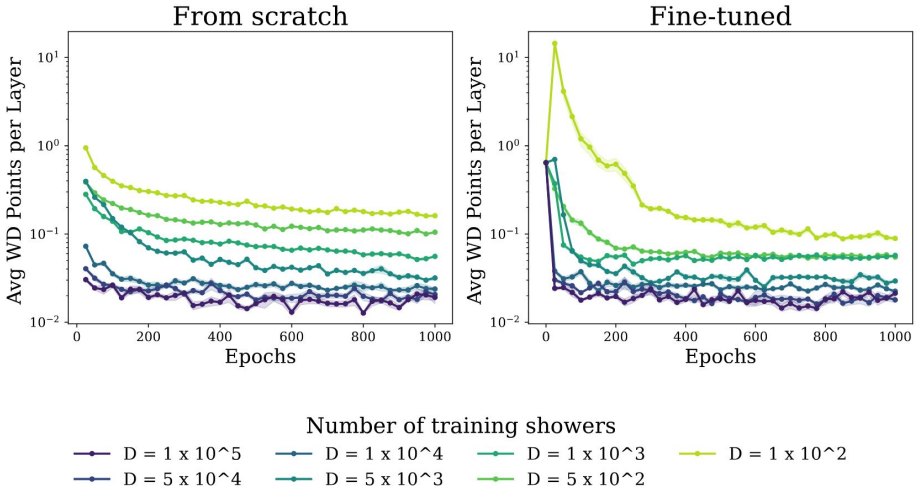
$$D_{\mathrm{KL}}^{\mathrm{quant}}(P\|Q) = \sum_{i=1}^{N} P_i \log\left(\frac{P_i}{Q_i}\right)$$

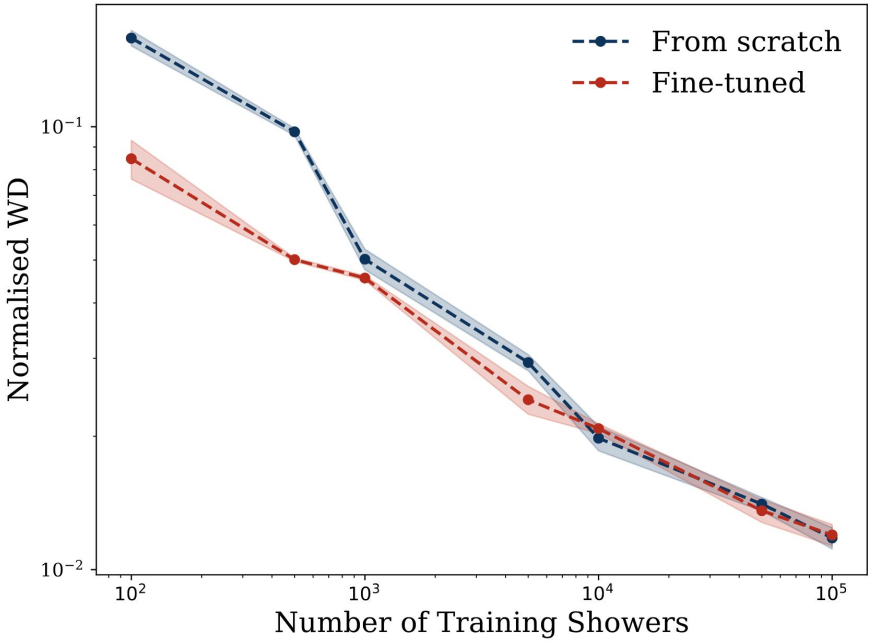
  - P_i and Q_i are the probabilities of the Geant4 samples and generated sample falling into the i-th bin, respectively.
  - The bins are defined by the quantiles of the reference sample:

$$\text{bin\_edges} = \left\{-\infty, q_{\frac{0}{N}}, q_{\frac{1}{N}}, \ldots, q_{\frac{N-1}{N}}, +\infty\right\}$$

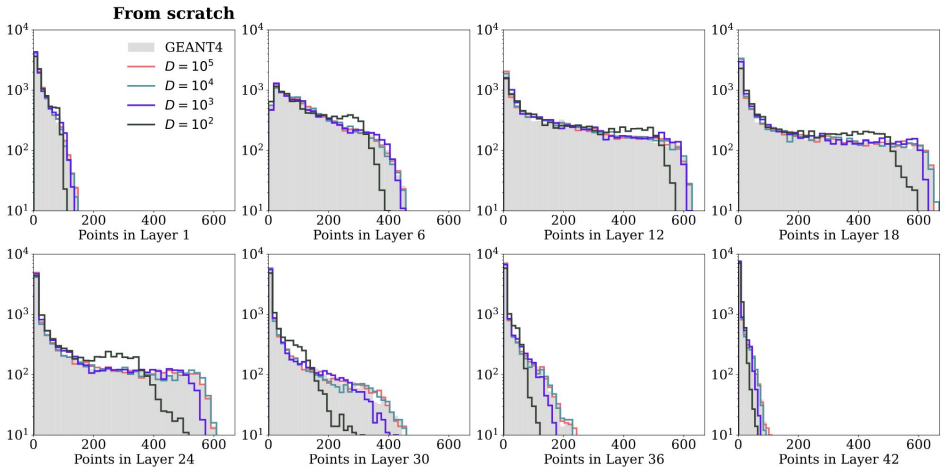
    With q_α being the α-quantile of Geant4 sample.

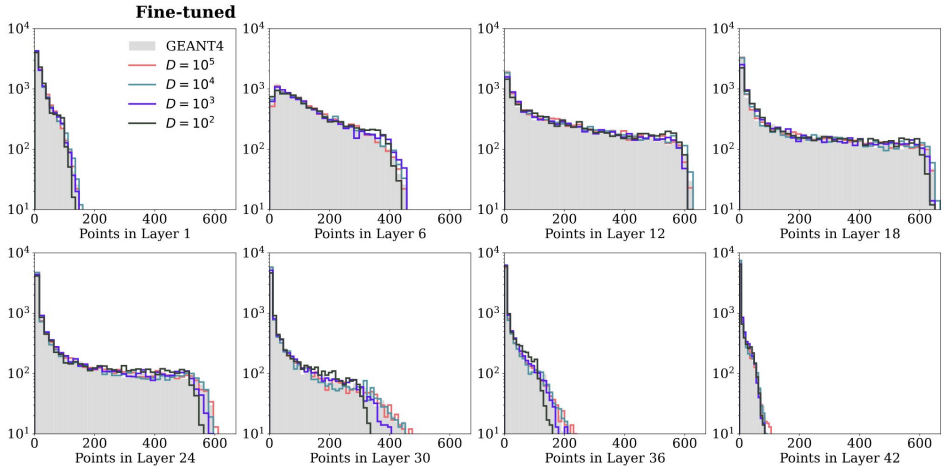
ShowerFlow


From scratch
Fine-tuned
Avg WD Points per Layer
Epochs
Number of training showers
D = 1 x 10^5
D = 1 x 10^4
D = 1 x 10^3
D = 1 x 10^2
D = 5 x 10^4
D = 5 x 10^3
D = 5 x 10^2

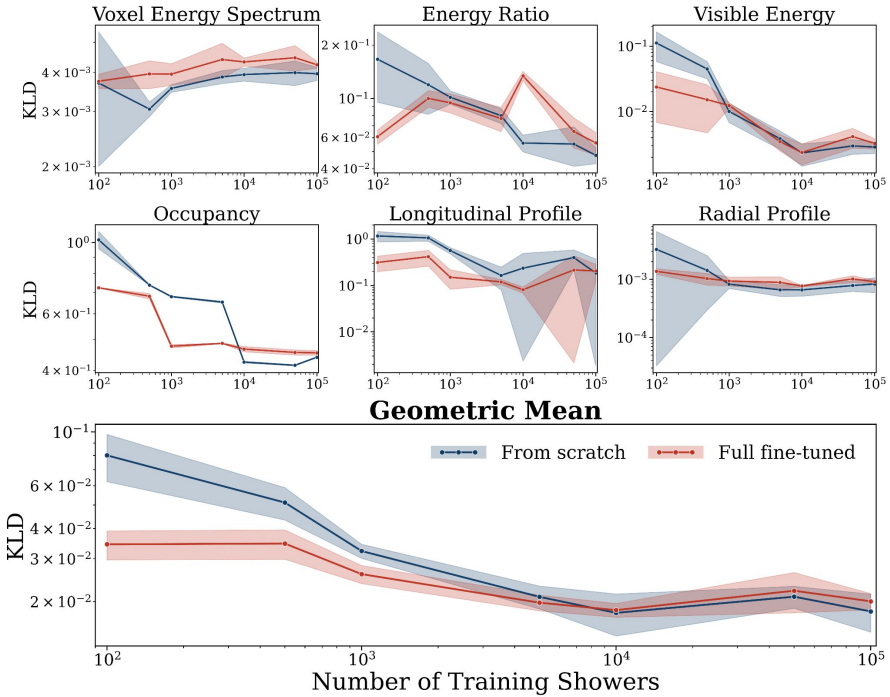

From scratch
Fine-tuned
Normalised WD
Number of Training Showers

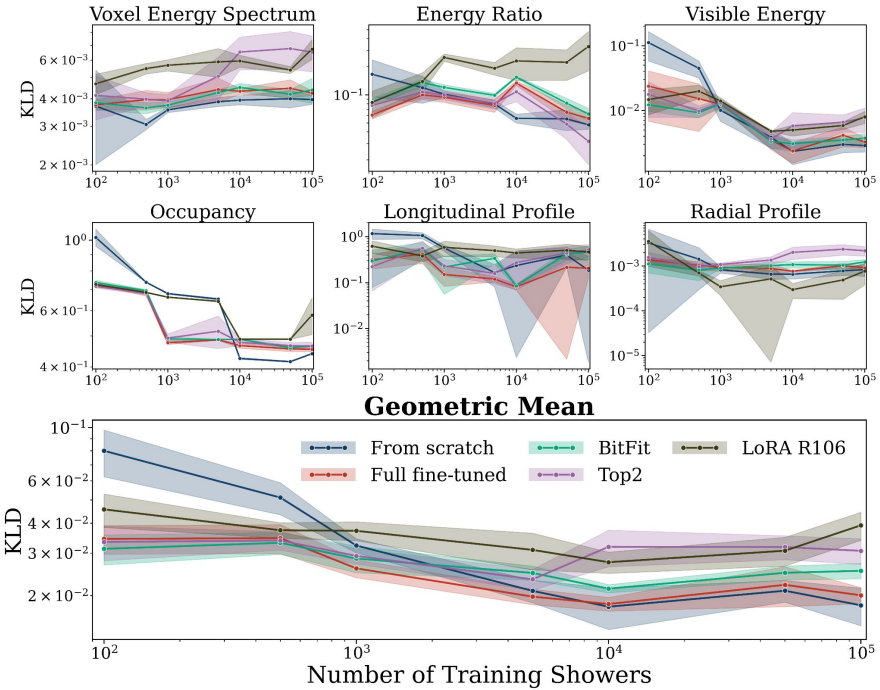
ShowerFlow: points per layer distributions



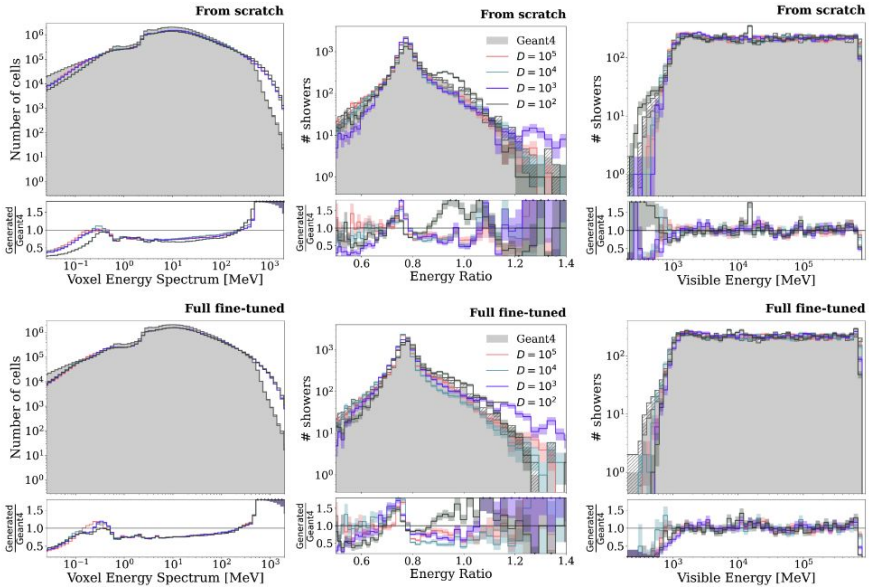


Results : KLD

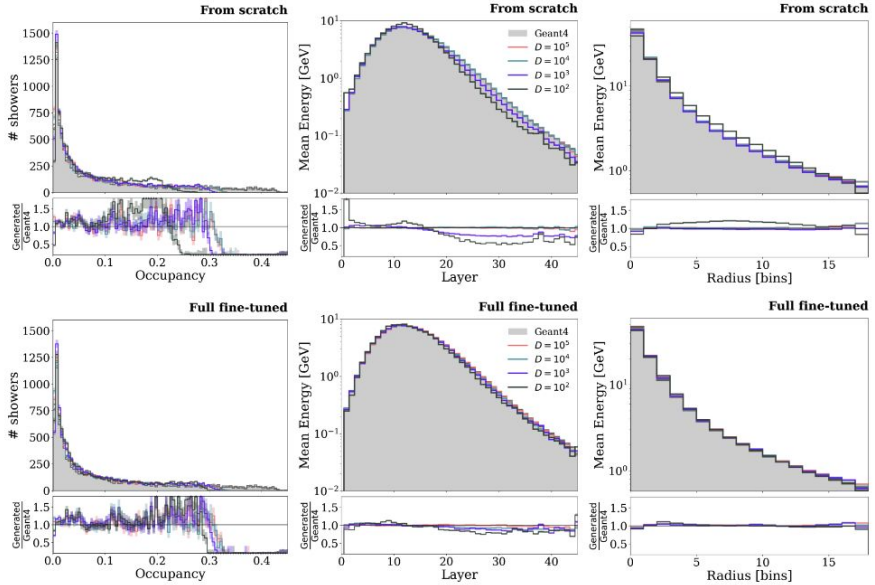

Voxel Energy Spectrum
Energy Ratio
Visible Energy
Occupancy
Longitudinal Profile
Radial Profile
KLD
Geometric Mean
From scratch
Full fine-tuned
Number of Training Showers


Voxel Energy Spectrum
Energy Ratio
Visible Energy
Occupancy
Longitudinal Profile
Radial Profile
KLD
Geometric Mean
From scratch
Full fine-tuned
BitFit
Top2
LoRA R106
Number of Training Showers

Observable Distributions



(a) Distributions: cell energy spectrum (left), total deposited energy over incident energy (center), visit energy (right).



(b) Distributions: occupancy (left), longitudinal (center), radial profile (right).

What is LoRA? [1]

Definition:

- LoRA *freezes* the original (*pretrained*) model weights and introduces a small number of *trainable low-rank matrices* into specific layers.

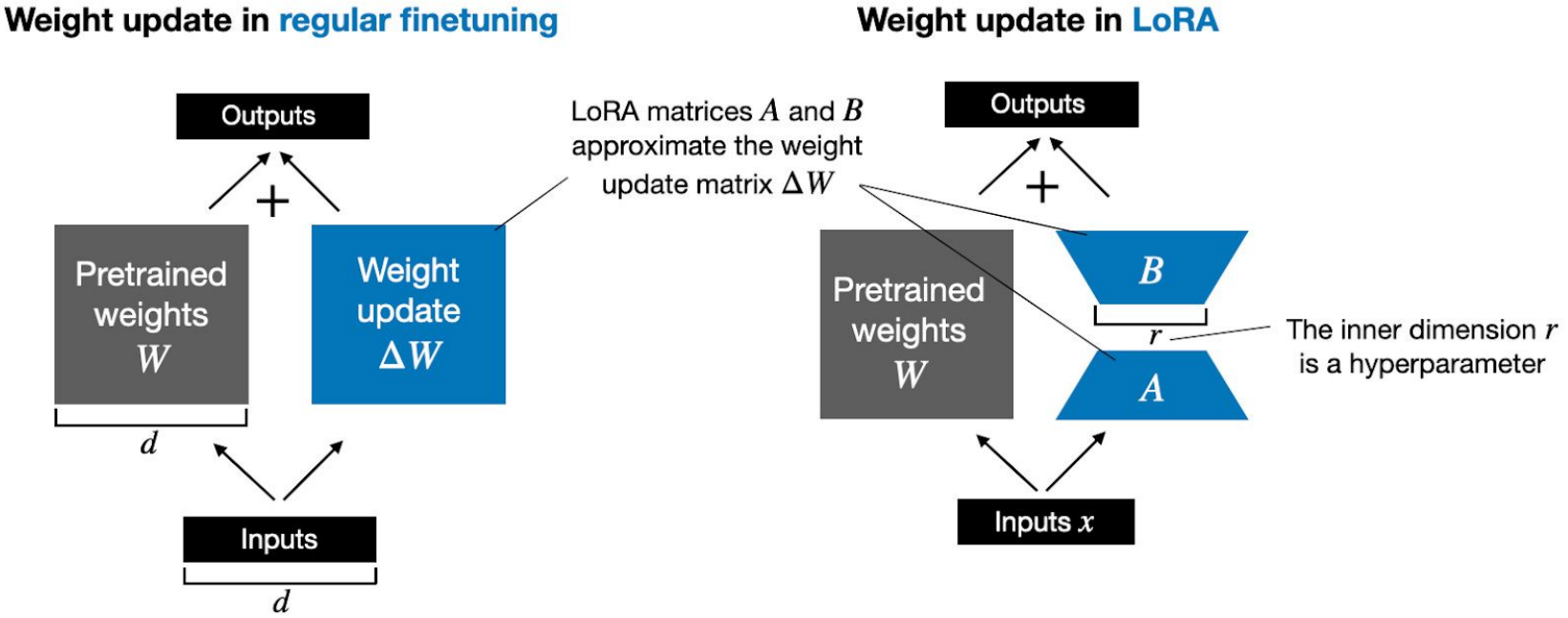
Key Ideas:

- Instead of updating the full weight matrix ***W***, LoRA *decomposes* the update into two smaller matrices ***A***, ***B***, such that the update is $\Delta W = A \times B$ such that:

$$W' = W + \frac{\alpha}{r}\Delta W = W + \frac{\alpha}{r} A \times B$$

- ***W*** is the pretrained weights (frozen)
- ***A***, ***B*****:** Low-rank matrices (trainable)
- α LoRA scaling factor (hyperparm)
- ***r*** is the rank of LoRA (hyperparm)

[1] LoRA


Weight update in regular finetuning
Outputs
Pretrained weights W
Weight update ΔW
d
Inputs
d
LoRA matrices A and B approximate the weight update matrix ΔW
Weight update in LoRA
Outputs
Pretrained weights W
B
r
A
The inner dimension r is a hyperparameter
Inputs x

Inverse LoRA study

$$\Delta W = W_{\text{ft}} - W_{\text{pre}} \in \mathbb{R}^{m \times n}$$

According to the Eckart-Young-Mirsky the optimal rank approximation minimizing Frobenius norm error is:

$$\Delta W = U \Sigma V^{\top} = \sum_{i=1}^{\rho} \sigma_i u_i v_i^{\top} \qquad \text{(SVD)}$$

where $\rho = \min(m, n)$

The reconstruction error is defined as the relative Frobenius norm:

$$\varepsilon_r = \frac{\|\Delta W - \Delta W_r\|_F}{\|\Delta W\|_F}$$