SuperCalo

Calorimeter shower super-resolution

Ian Pang
Nov 6, 2023
ML4Jets, Hamburg

RUTGERS
UNIVERSITY | NEW BRUNSWICK

[2308.11700] **IP**, J. Raine, D. Shih

Fast Calorimeter Simulation

Fast Calorimeter Simulation

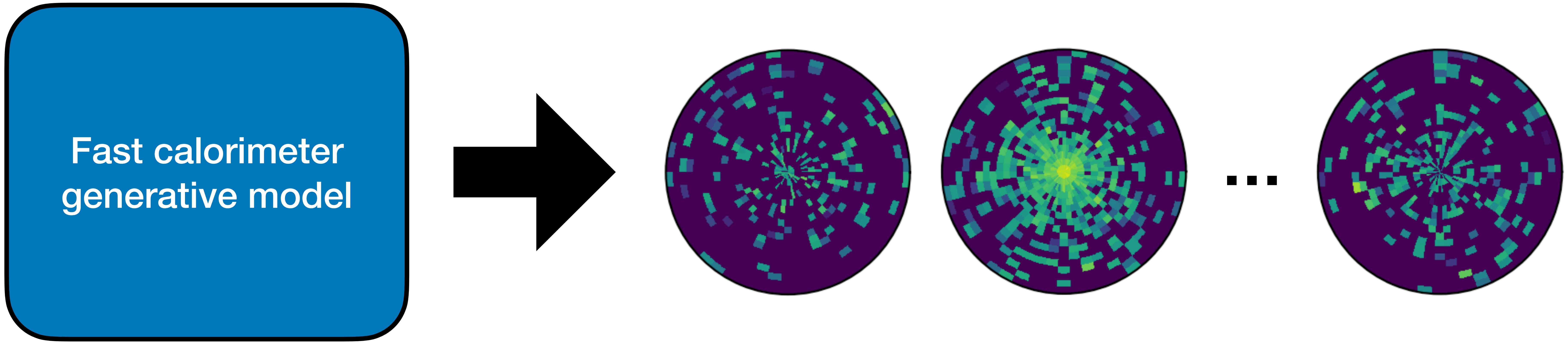
- Calorimeter shower simulation is major bottleneck in LHC computational pipeline!

Fast Calorimeter Simulation

- Calorimeter shower simulation is major bottleneck in LHC computational pipeline!
- Surrogate modeling to speed up generation of expensive GEANT4 calorimeter showers

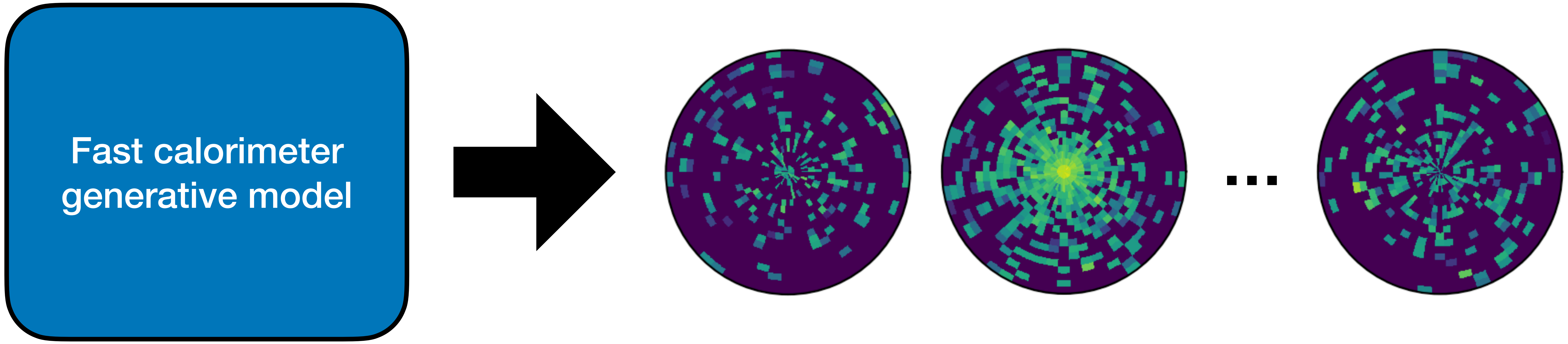
Fast Calorimeter Simulation

- Calorimeter shower simulation is major bottleneck in LHC computational pipeline!
- Surrogate modeling to speed up generation of expensive GEANT4 calorimeter showers
- Most approaches directly simulate the full-dim showers (3D image) in a single step



Fast Calorimeter Simulation

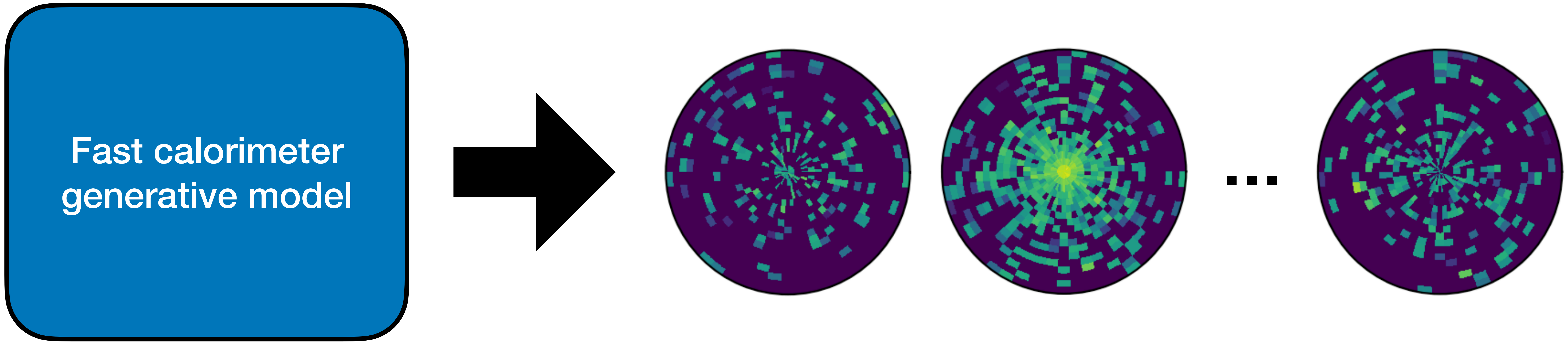
- Calorimeter shower simulation is major bottleneck in LHC computational pipeline!
- Surrogate modeling to speed up generation of expensive GEANT4 calorimeter showers
- Most approaches directly simulate the full-dim showers (3D image) in a single step



- Computationally prohibitive for high-dim (e.g. $\mathcal{O}(10^4)$)!

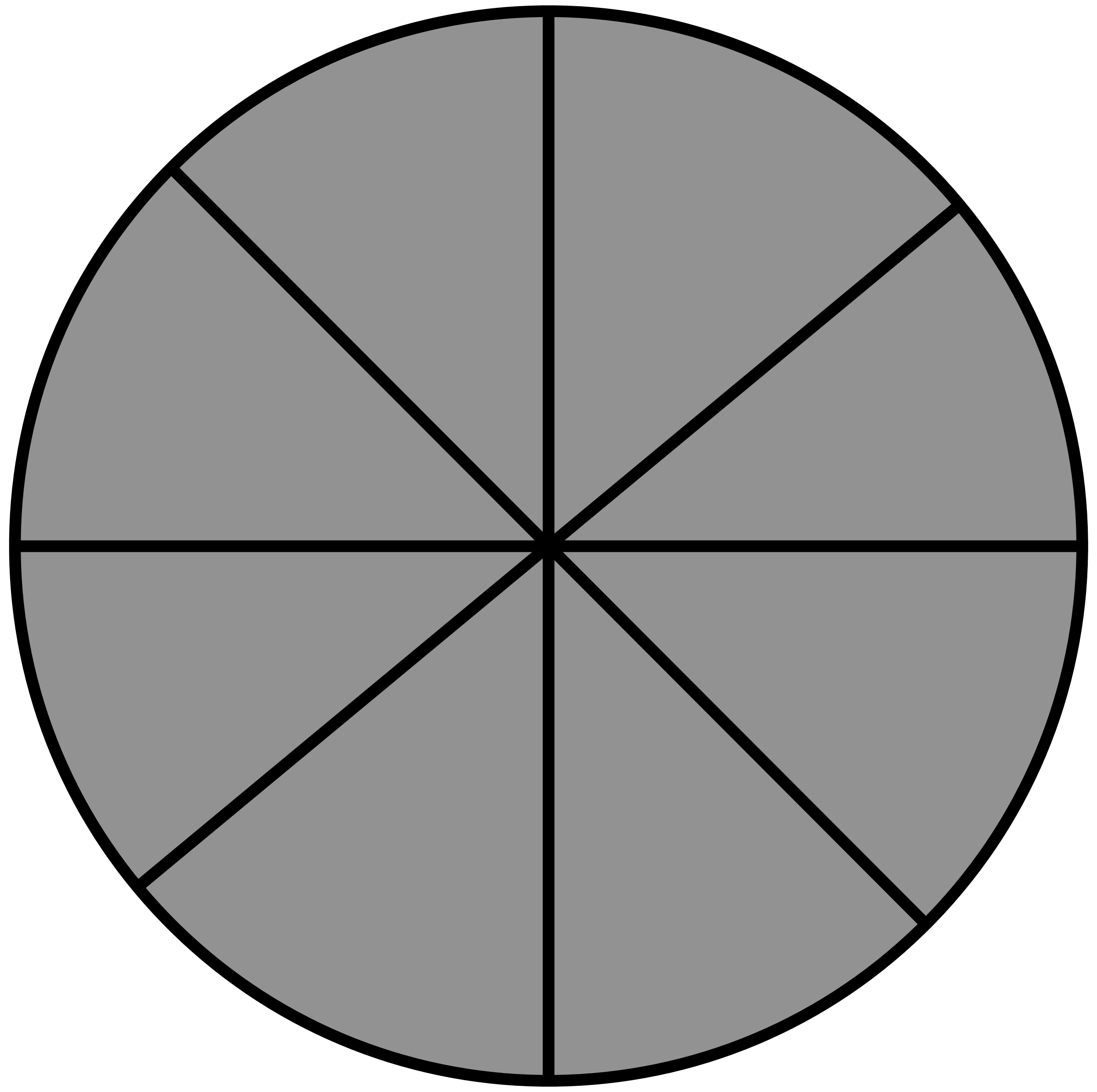
Fast Calorimeter Simulation

- Calorimeter shower simulation is major bottleneck in LHC computational pipeline!
- Surrogate modeling to speed up generation of expensive GEANT4 calorimeter showers
- Most approaches directly simulate the full-dim showers (3D image) in a single step



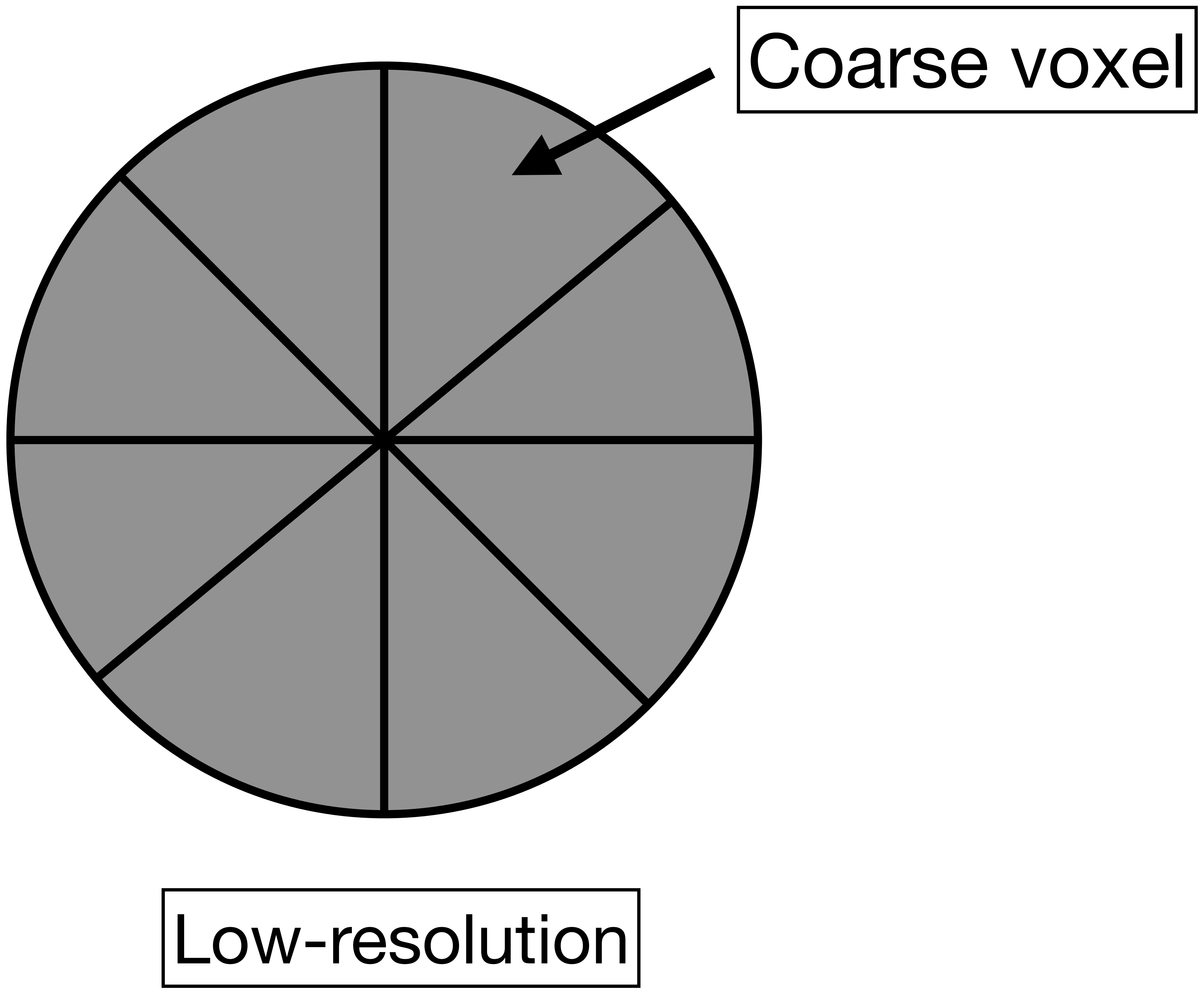
- Computationally prohibitive for high-dim (e.g. $\mathcal{O}(10^4)$)!
- **Enter SuperCalo!**

Main idea!

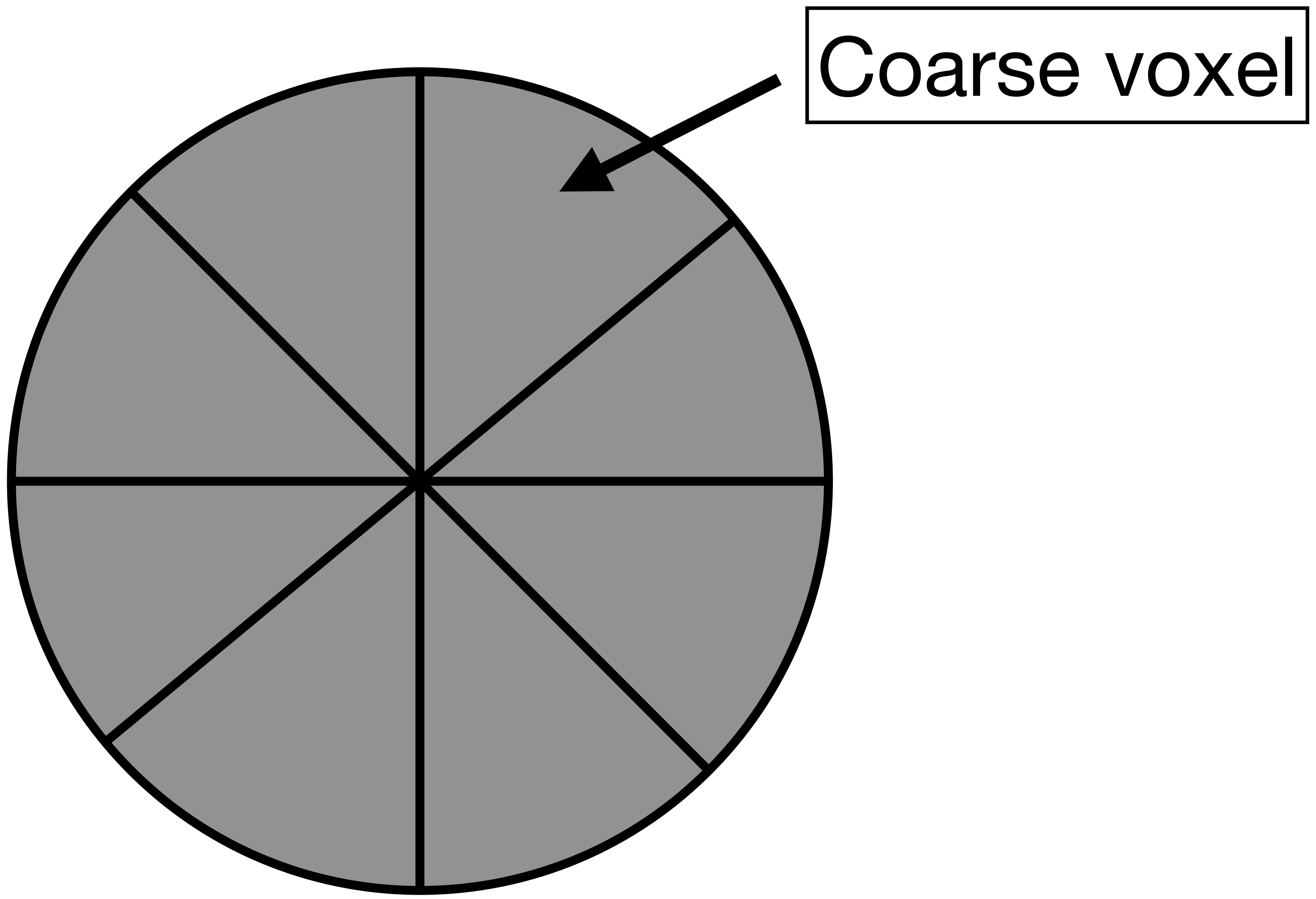


Low-resolution

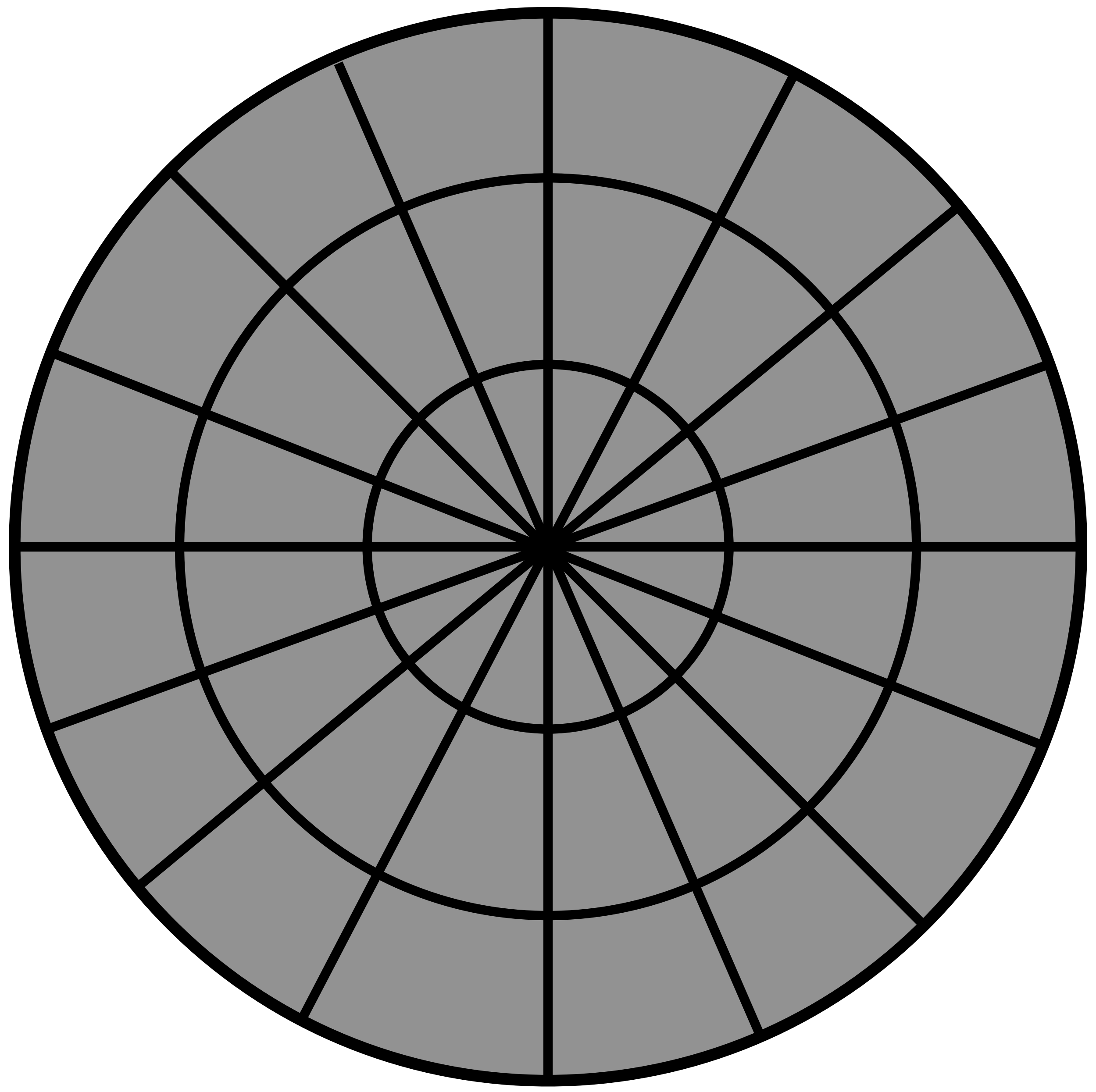
Main idea!



Main idea!

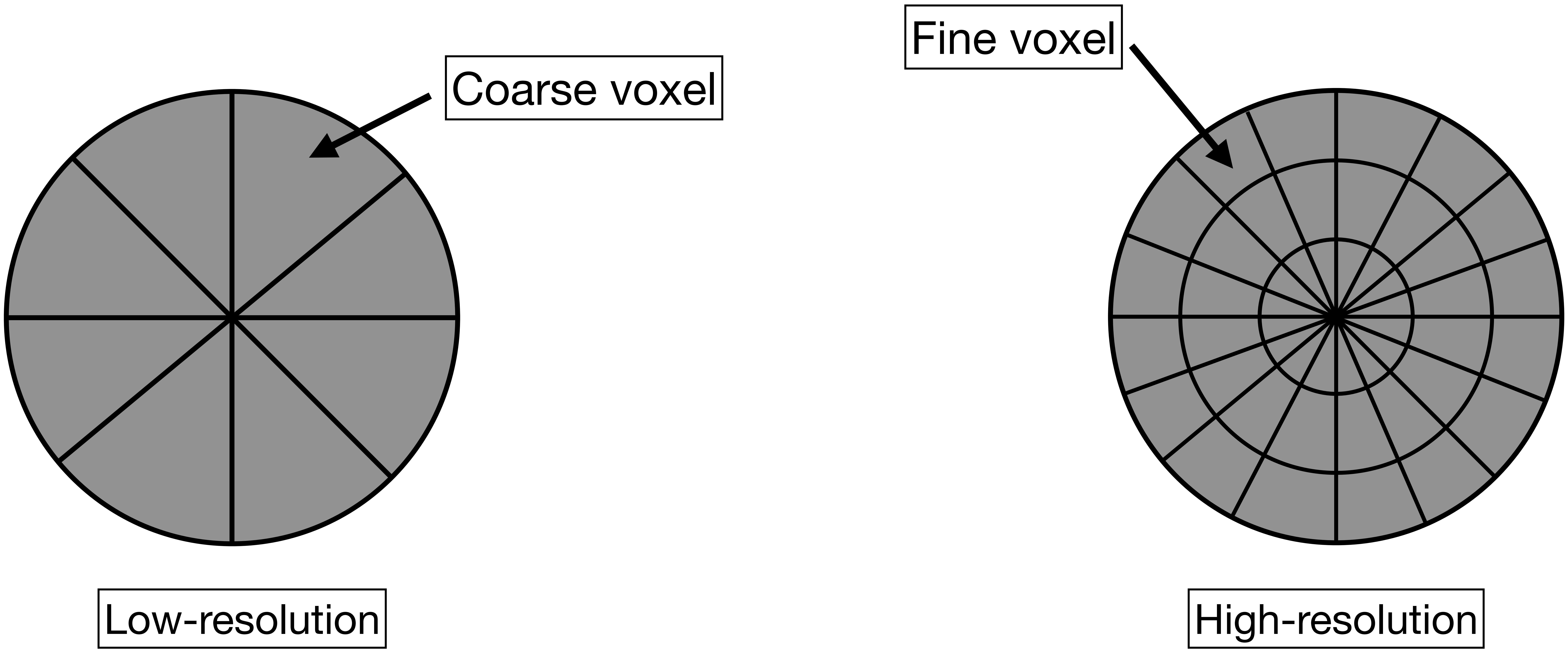


Low-resolution

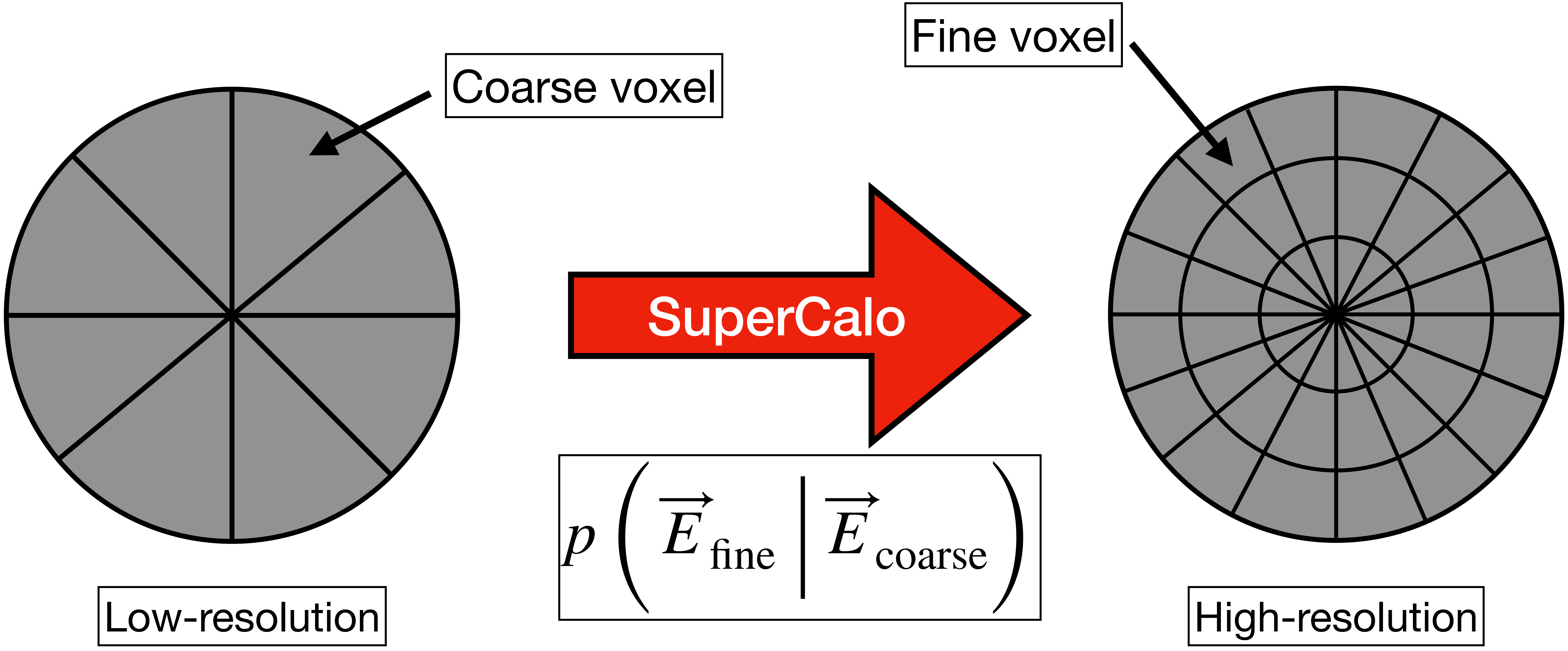


High-resolution

Main idea!

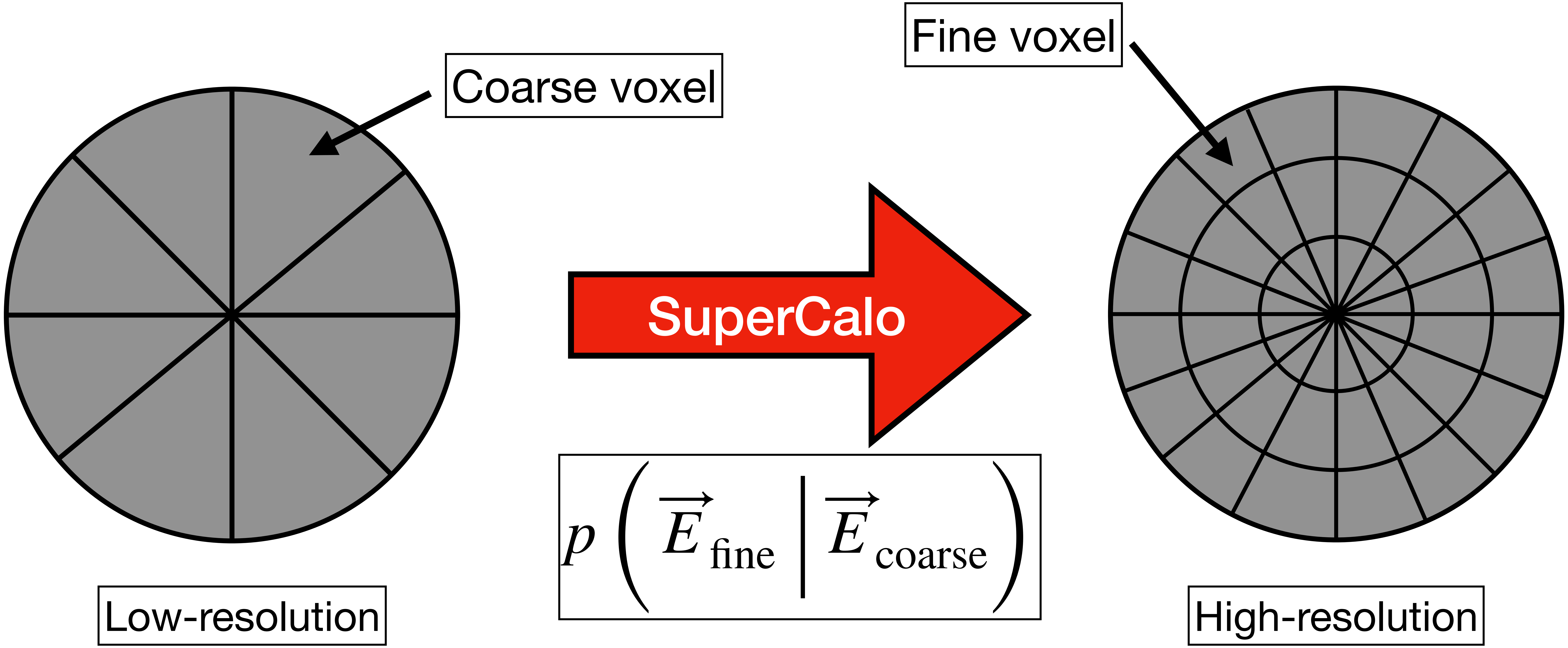


Main idea!



$$p\left(\vec{E}_{\text{fine}} \,\middle|\, \vec{E}_{\text{coarse}}\right)$$

Main idea!

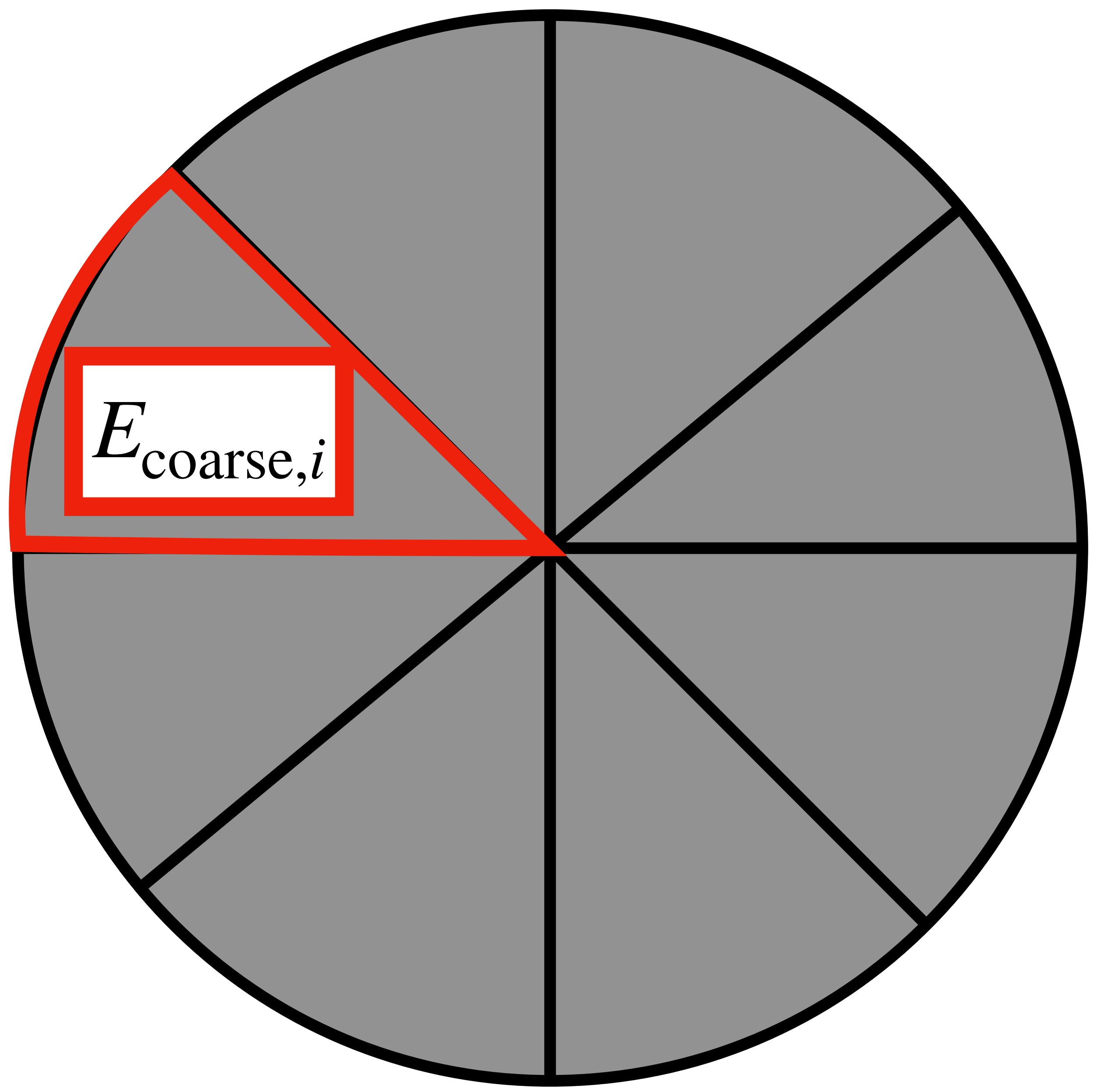


$$p\left(\vec{E}_{\text{fine}} \,\middle|\, \vec{E}_{\text{coarse}}\right)$$

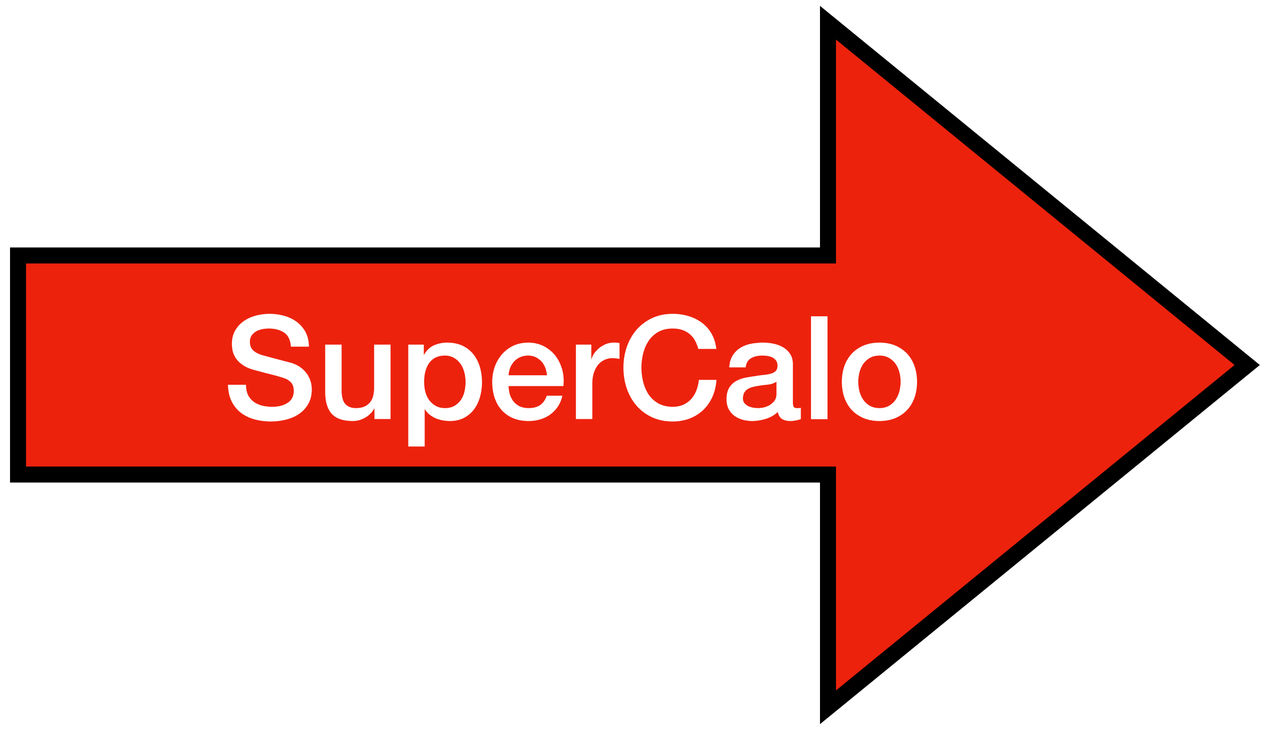
Ansatz: $$p\left(\vec{E}_{\text{fine}} \,\middle|\, \vec{E}_{\text{coarse}}\right) = \prod_{i=1}^{N_{\text{coarse}}} q\left(\vec{e}_{\text{fine},i} \,\middle|\, E_{\text{coarse},i} \;, \ldots\right)$$

- Coarse voxel index i

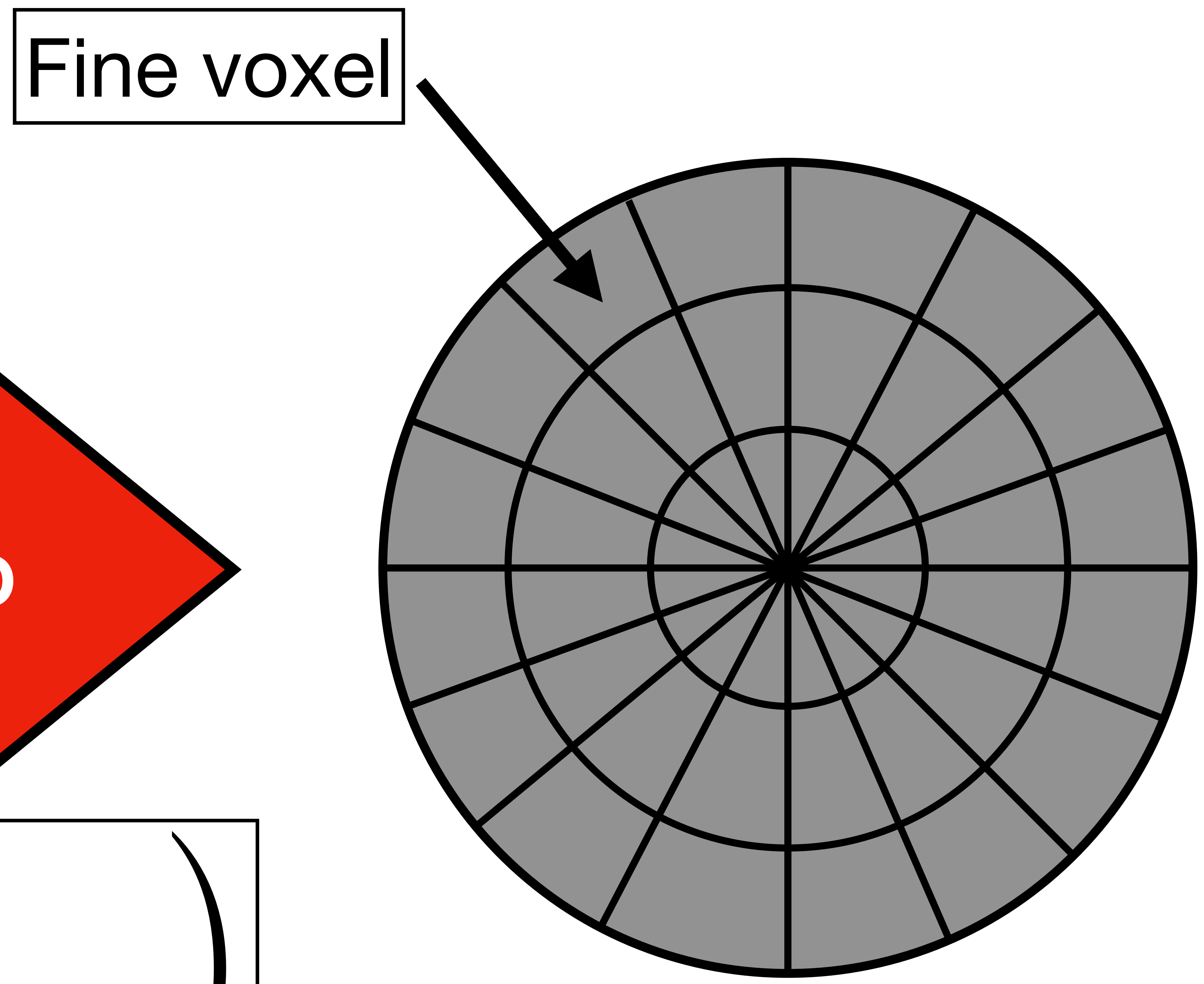
Main idea!



Low-resolution



$$p\left(\vec{E}_{\text{fine}} \,\middle|\, \vec{E}_{\text{coarse}}\right)$$

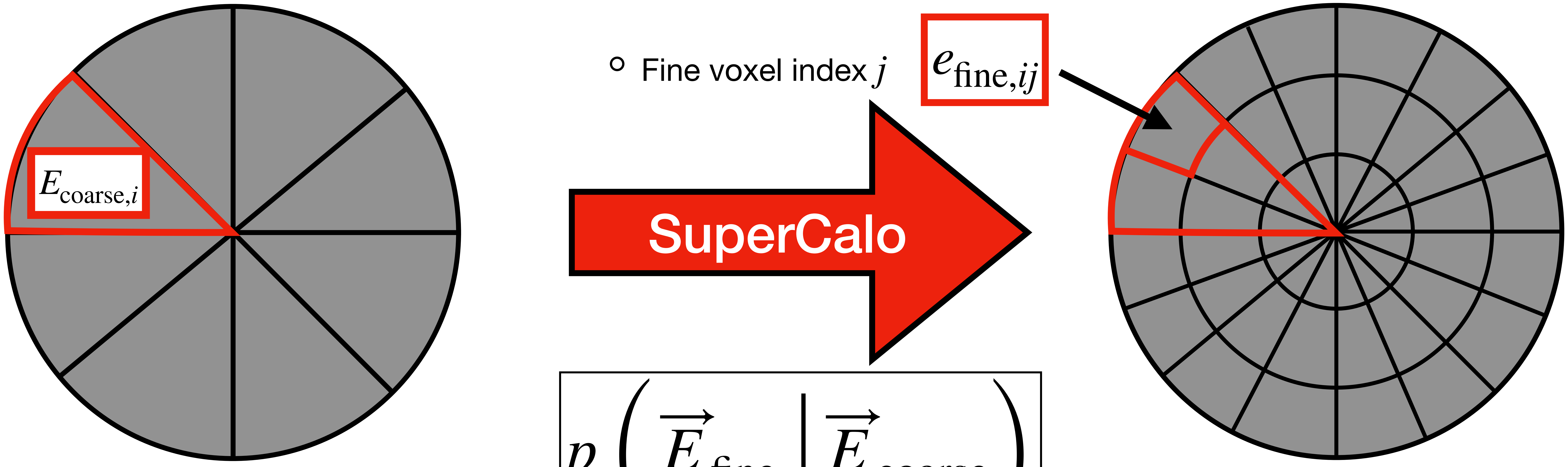


High-resolution

Ansatz: $p\left(\vec{E}_{\text{fine}} \,\middle|\, \vec{E}_{\text{coarse}}\right) = \prod_{i=1}^{N_{\text{coarse}}} q\left(\vec{e}_{\text{fine},i} \,\middle|\, \boxed{E_{\text{coarse},i}}, \ldots\right)$

- Coarse voxel index i

Main idea!



$$p\left(\vec{E}_{\mathrm{fine}} \,\middle|\, \vec{E}_{\mathrm{coarse}}\right)$$

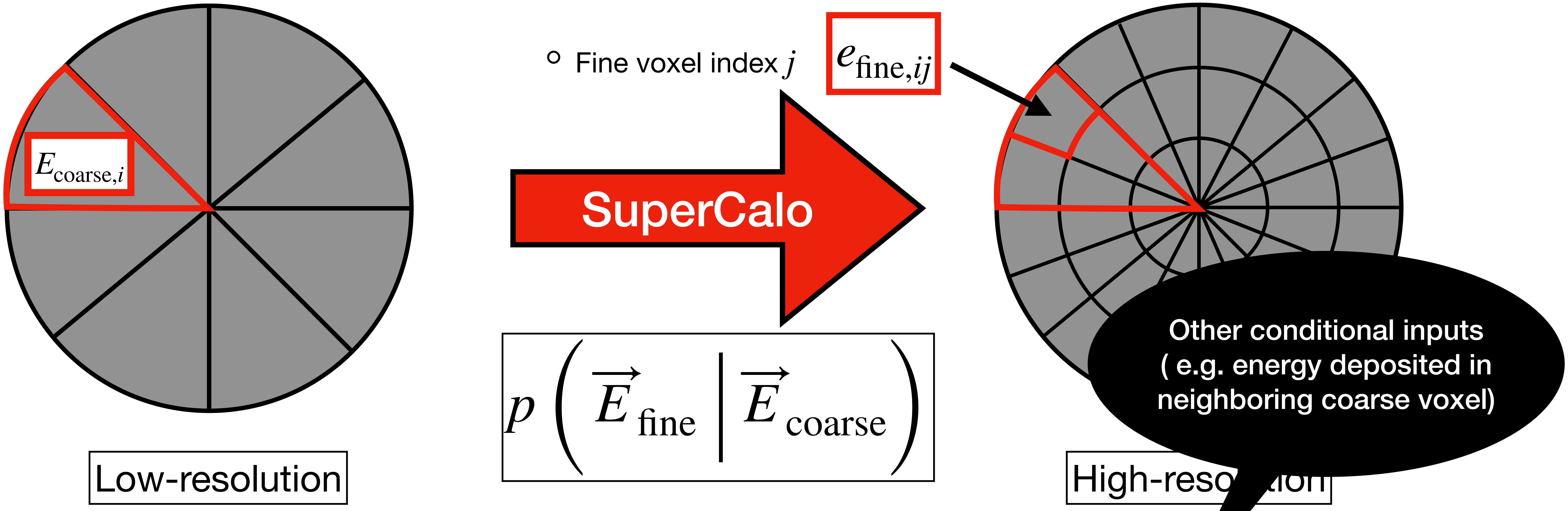
Low-resolution

High-resolution

Ansatz: $p\left(\vec{E}_{\mathrm{fine}} \,\middle|\, \vec{E}_{\mathrm{coarse}}\right) = \prod_{i=1}^{N_{\mathrm{coarse}}} q\left(\boxed{\vec{e}_{\mathrm{fine},i}} \,\middle|\, E_{\mathrm{coarse},i}\ , \dots\right)$

- Coarse voxel index i

Main idea!

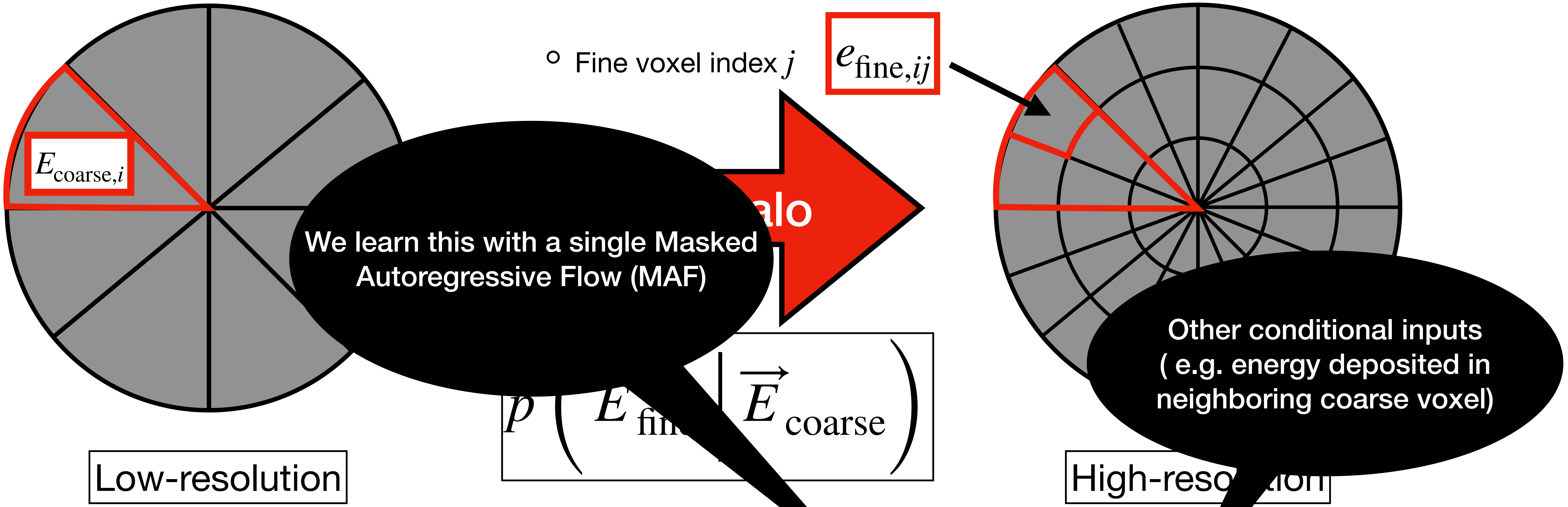


$$p\left(\vec{E}_{\text{fine}} \,\middle|\, \vec{E}_{\text{coarse}}\right)$$

Ansatz: $$p\left(\vec{E}_{\text{fine}} \,\middle|\, \vec{E}_{\text{coarse}}\right) = \prod_{i=1}^{N_{\text{coarse}}} q\left(\vec{e}_{\text{fine},i} \,\middle|\, E_{\text{coarse},i} \;, \ldots\right)$$

- Coarse voxel index i

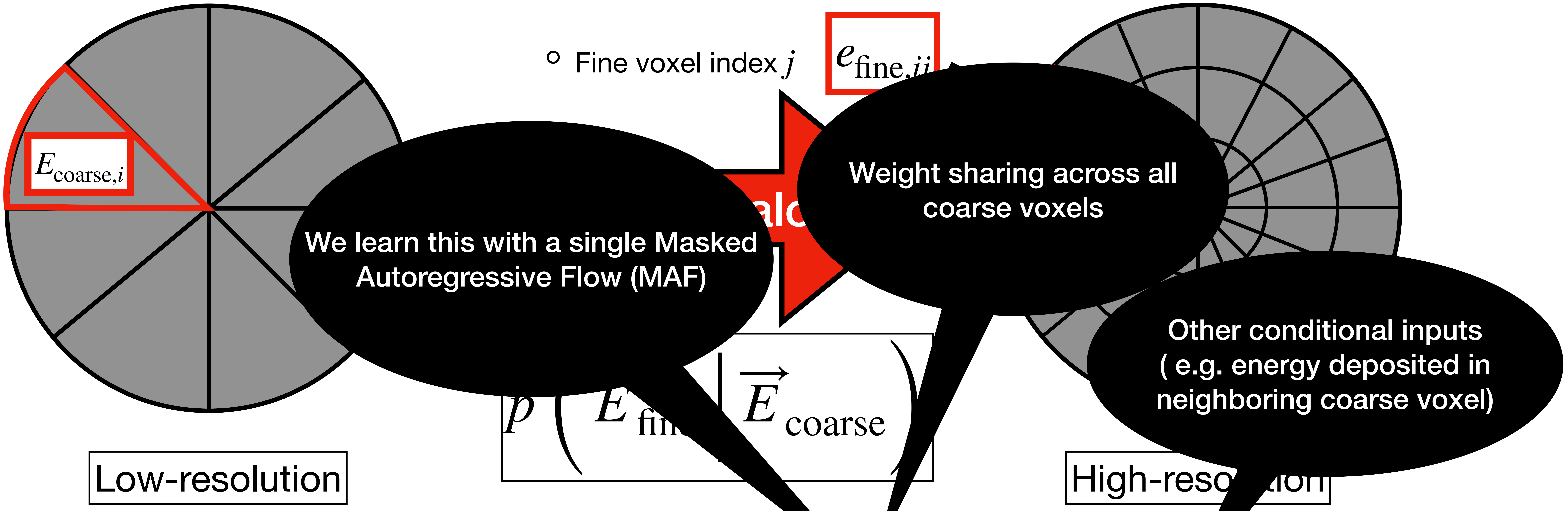
Main idea!



Ansatz: $$p\left(\vec{E}_{\text{fine}} \,\middle|\, \vec{E}_{\text{coarse}}\right) = \prod_{i=1}^{N_{\text{coarse}}} q\left(\vec{e}_{\text{fine},i} \,\middle|\, E_{\text{coarse},i}\ ,\ldots\right)$$

○ Coarse voxel index i

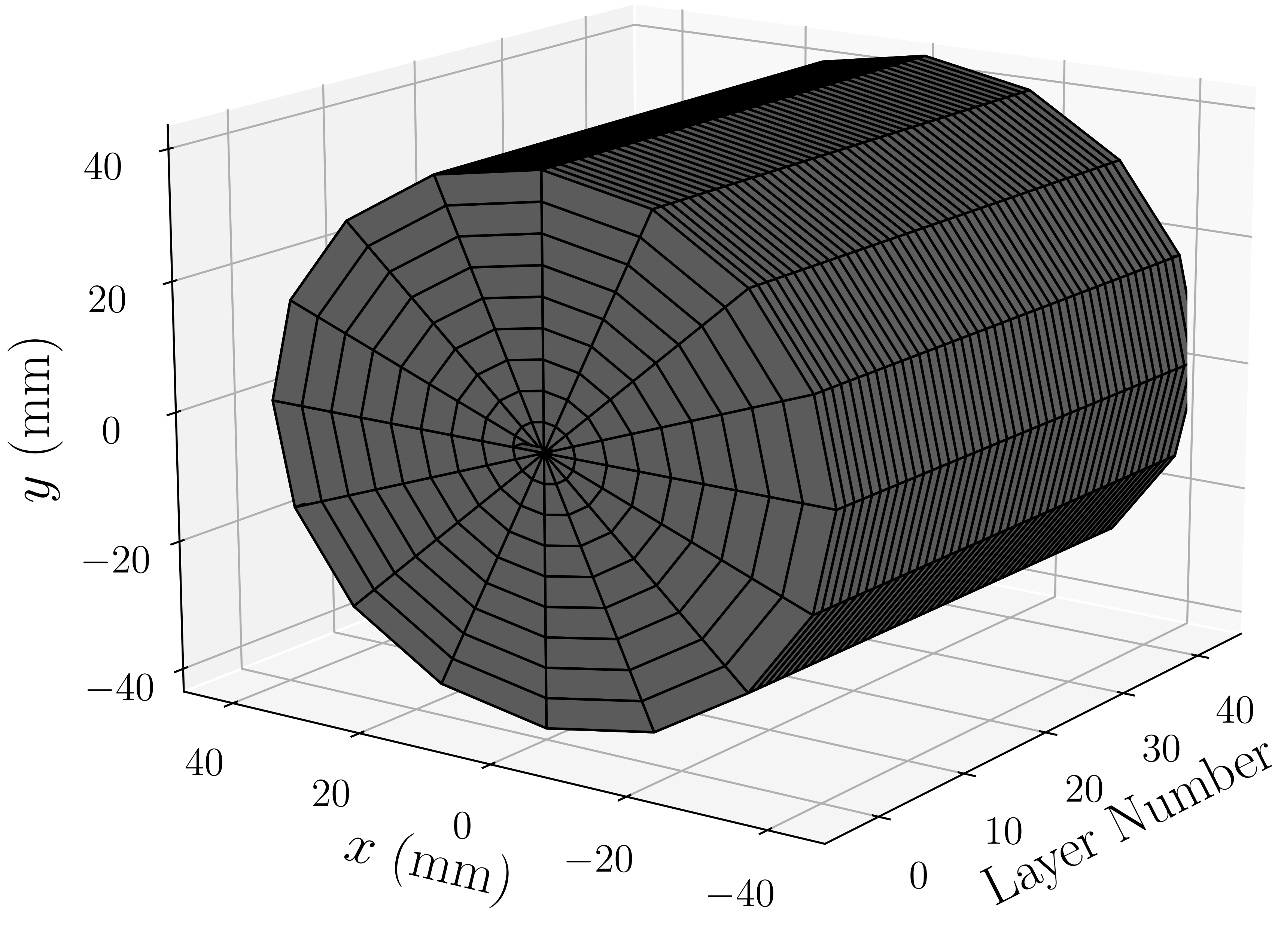
Main idea!



$$\text{Ansatz:} \quad p\left(\vec{E}_{\text{fine}} \,\middle|\, \vec{E}_{\text{coarse}}\right) = \prod_{i=1}^{N_{\text{coarse}}} q\left(\vec{e}_{\text{fine},i} \,\middle|\, E_{\text{coarse},i}\ , \ldots\right)$$

- Coarse voxel index i

Dataset



- Dataset 2 of CaloChallenge

  M.F. Giannelli, G. Kasieczka, C. Krause, B. Nachman, D. Salamani, D. Shih, A. Zaborowska
  https://doi.org/10.5281/zenodo.6366271

- Idealized calorimeter with concentric cylinders of absorber (W) and active material (Si)

Geometry:

- **45** layers (z)
- **16** angular (α) bins
- **9** radial (r) bins

} **6480** voxels

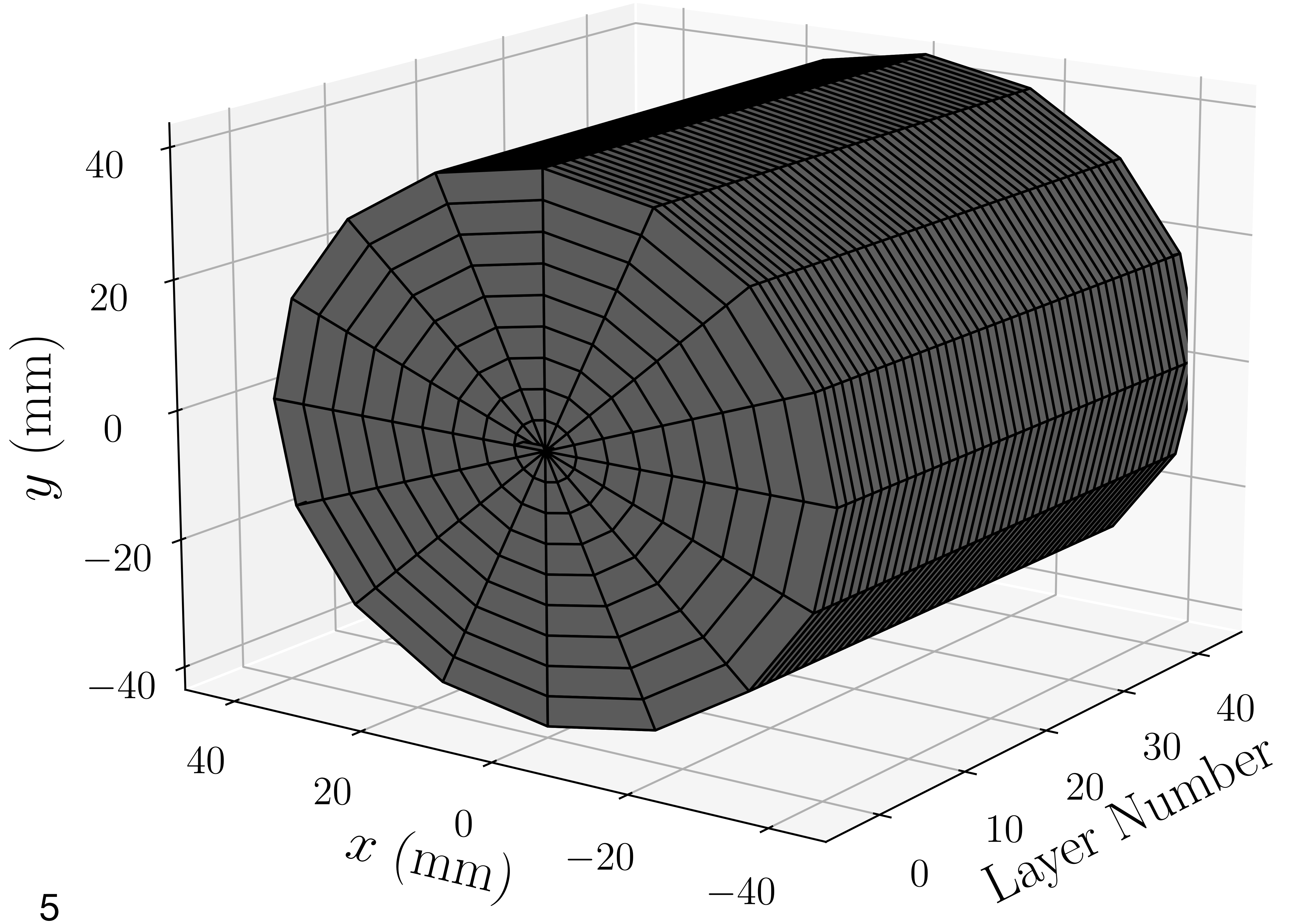
Format:

Total **100k** e^- showers

- Incident energy (1-dim) ~ [1 GeV, 1 TeV]
- Fine voxel energies (6480-dim)

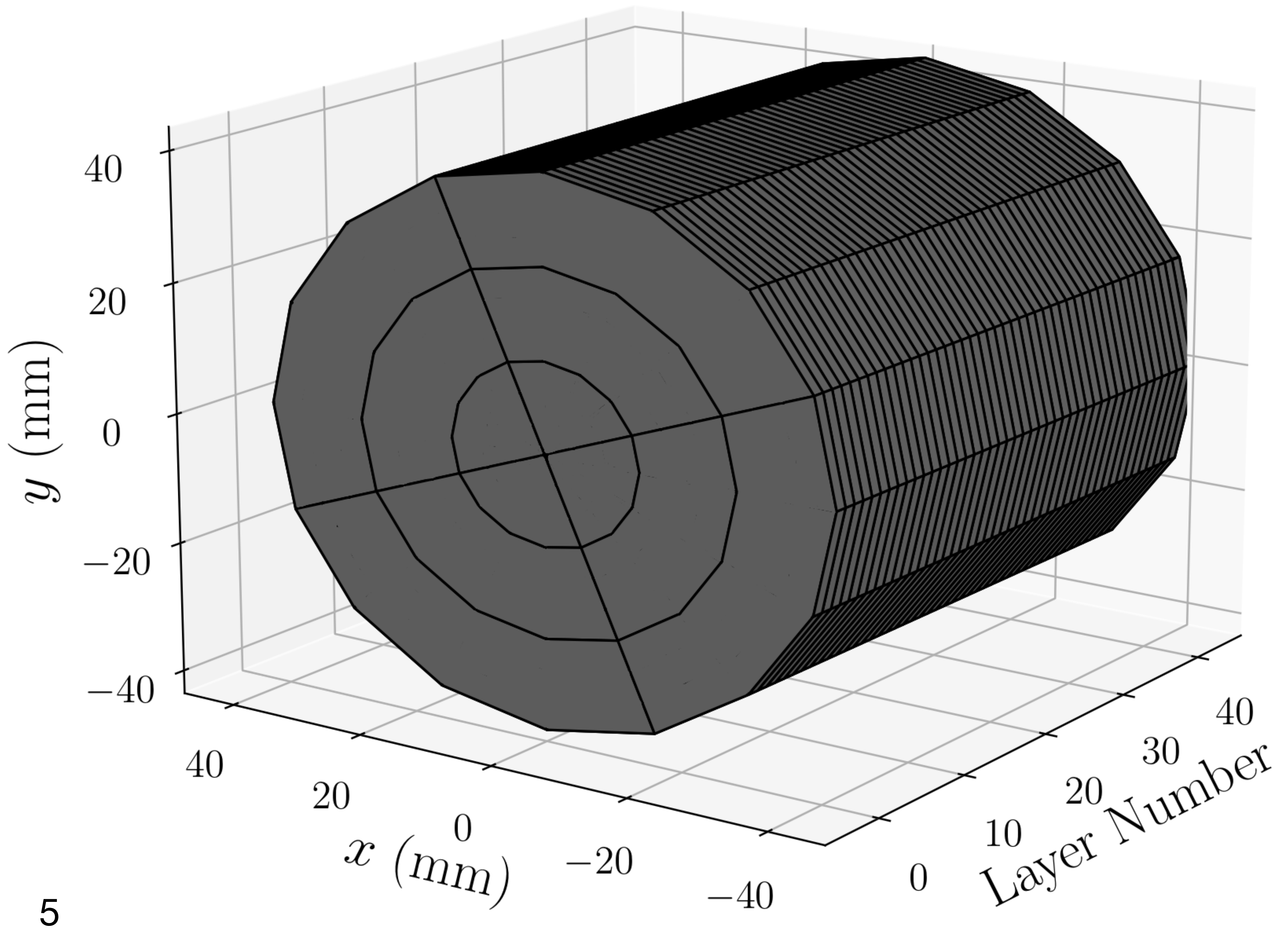
Coarse voxelization

- **Many possible choices** with no obvious best choice!



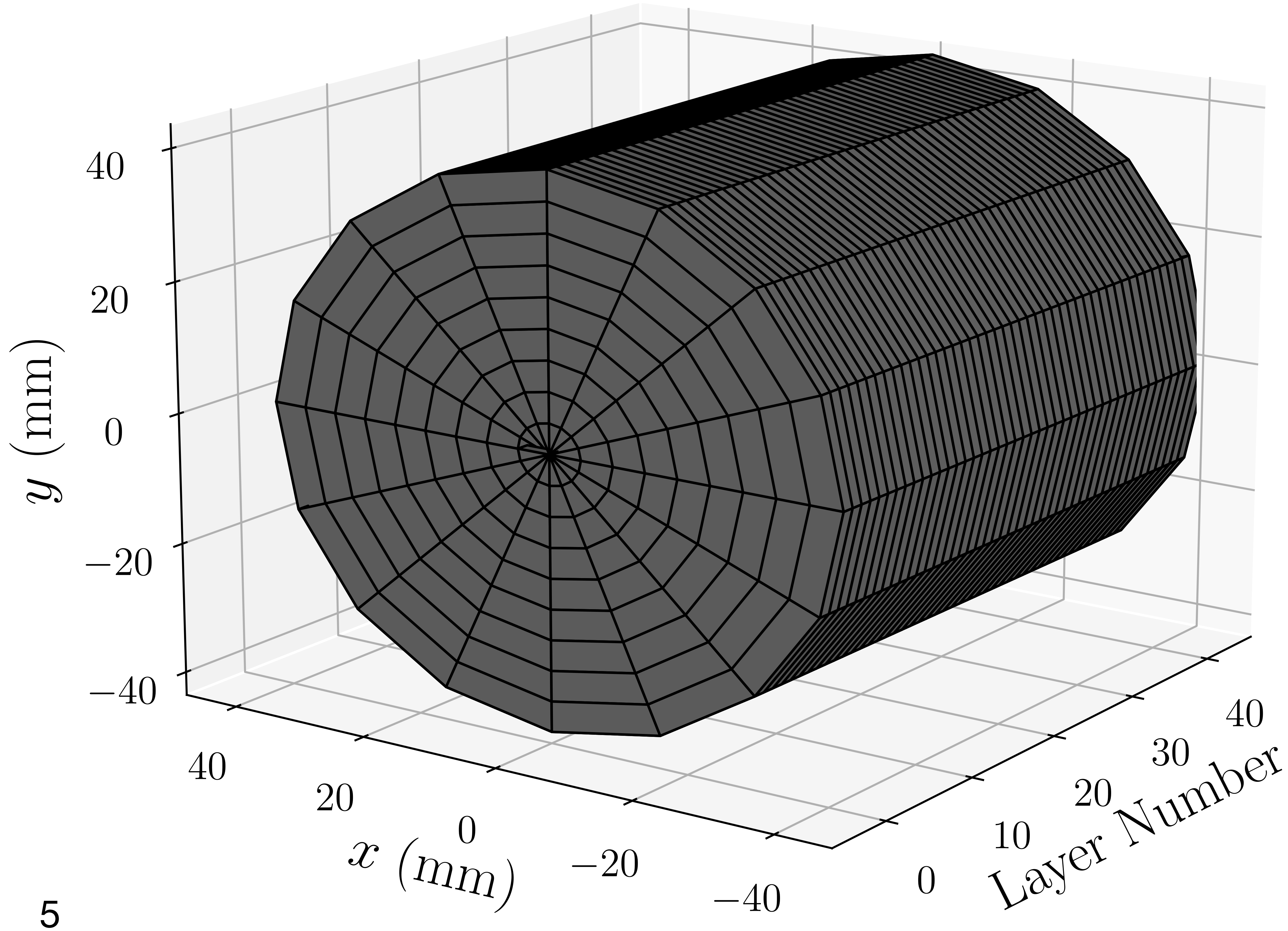
Coarse voxelization

- **Many possible choices** with no obvious best choice!



Coarse voxelization

- **Many possible choices** with no obvious best choice!



Coarse voxelization

- **Many possible choices** with no obvious best choice!

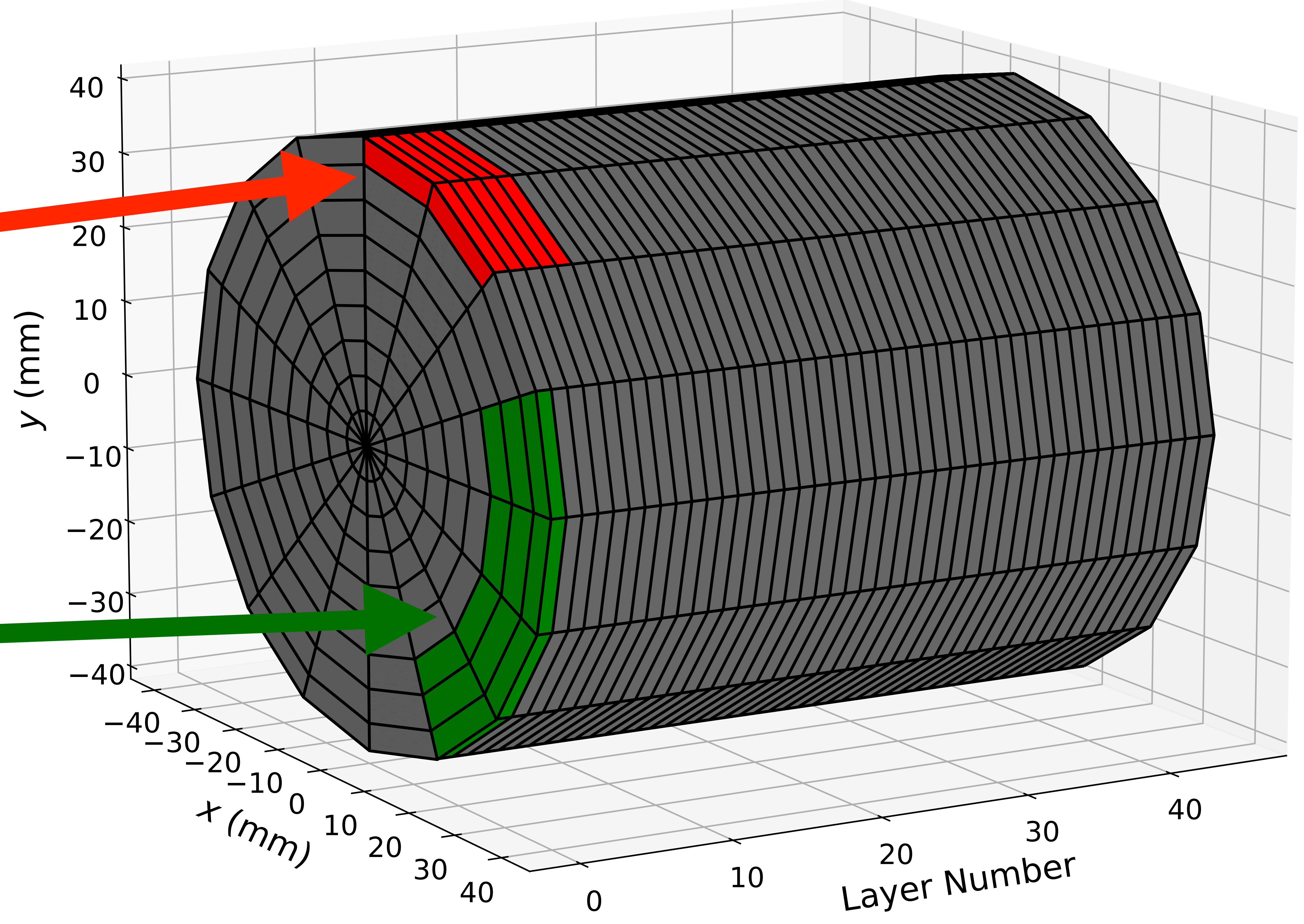
- Experimented with **two** choices:

Choice A (red) :

1 coarse voxel = 1 r × 2 α × 5 z

Choice B (green) :

1 coarse voxel = 3 r × 4 α × 1 z



Comparing coarse voxelization choices

Choice B (only group in r and α) fails to capture inter-layer correlations!

Comparing coarse voxelization choices

Choice B (only group in r and α) <u>fails to capture inter-layer correlations</u>!

Generate two sets of **<u>high-res showers (fine voxels)</u>** using choices A & B respectively

Comparing coarse voxelization choices

Choice **B** (only group in r and α) <u>fails to capture inter-layer correlations</u>!

Generate two sets of **<u>high-res showers (fine voxels)</u>** using choices A & B respectively

Compute ρ

$$\rho \equiv \frac{\vec{e}^{\,(n)}_{\text{fine}} \cdot \vec{e}^{\,(n+1)}_{\text{fine}}}{\left|\vec{e}^{\,(n)}_{\text{fine}}\right| \left|\vec{e}^{\,(n+1)}_{\text{fine}}\right|}$$

Comparing coarse voxelization choices

Choice **B** (only group in r and α) fails to capture inter-layer correlations!

Generate two sets of **high-res showers (fine voxels)** using choices A & B respectively

Compute ρ

$$\rho \equiv \frac{\vec{e}^{\,(n)}_{\text{fine}} \cdot \vec{e}^{\,(n+1)}_{\text{fine}}}{\left|\vec{e}^{\,(n)}_{\text{fine}}\right| \left|\vec{e}^{\,(n+1)}_{\text{fine}}\right|}$$

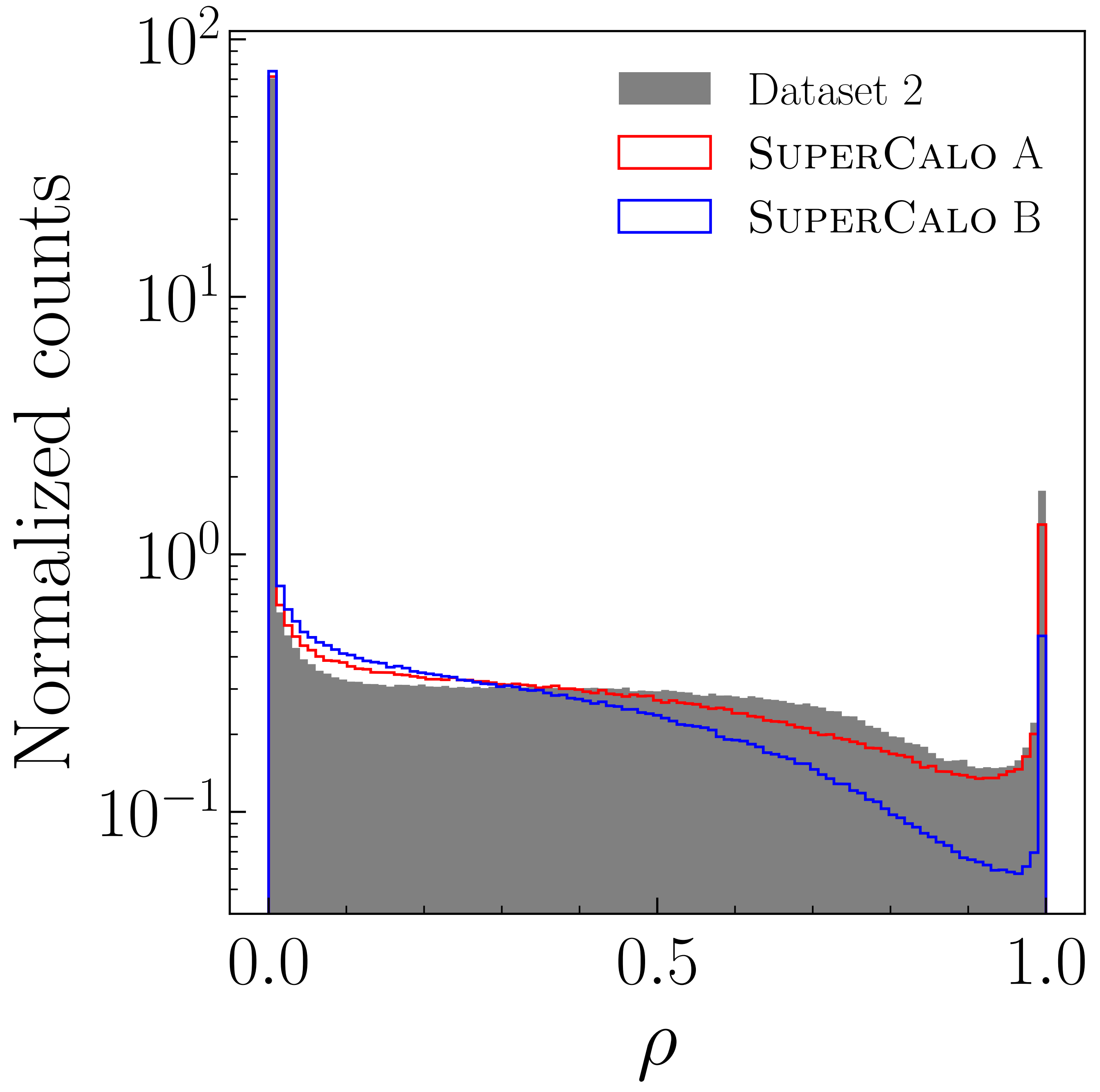
Dot product of fine voxels contained in adjacent layers*

Comparing coarse voxelization choices

Choice B (only group in r and α) <u>fails to capture inter-layer correlations</u>!

Generate two sets of **<u>high-res showers (fine voxels)</u>** using choices A & B respectively

Compute ρ



$$\rho \equiv \frac{\vec{e}^{\,(n)}_{\text{fine}} \cdot \vec{e}^{\,(n+1)}_{\text{fine}}}{\left|\vec{e}^{\,(n)}_{\text{fine}}\right| \left|\vec{e}^{\,(n+1)}_{\text{fine}}\right|}$$

Dot product of fine voxels contained in adjacent layers*

Comparing coarse voxelization choices

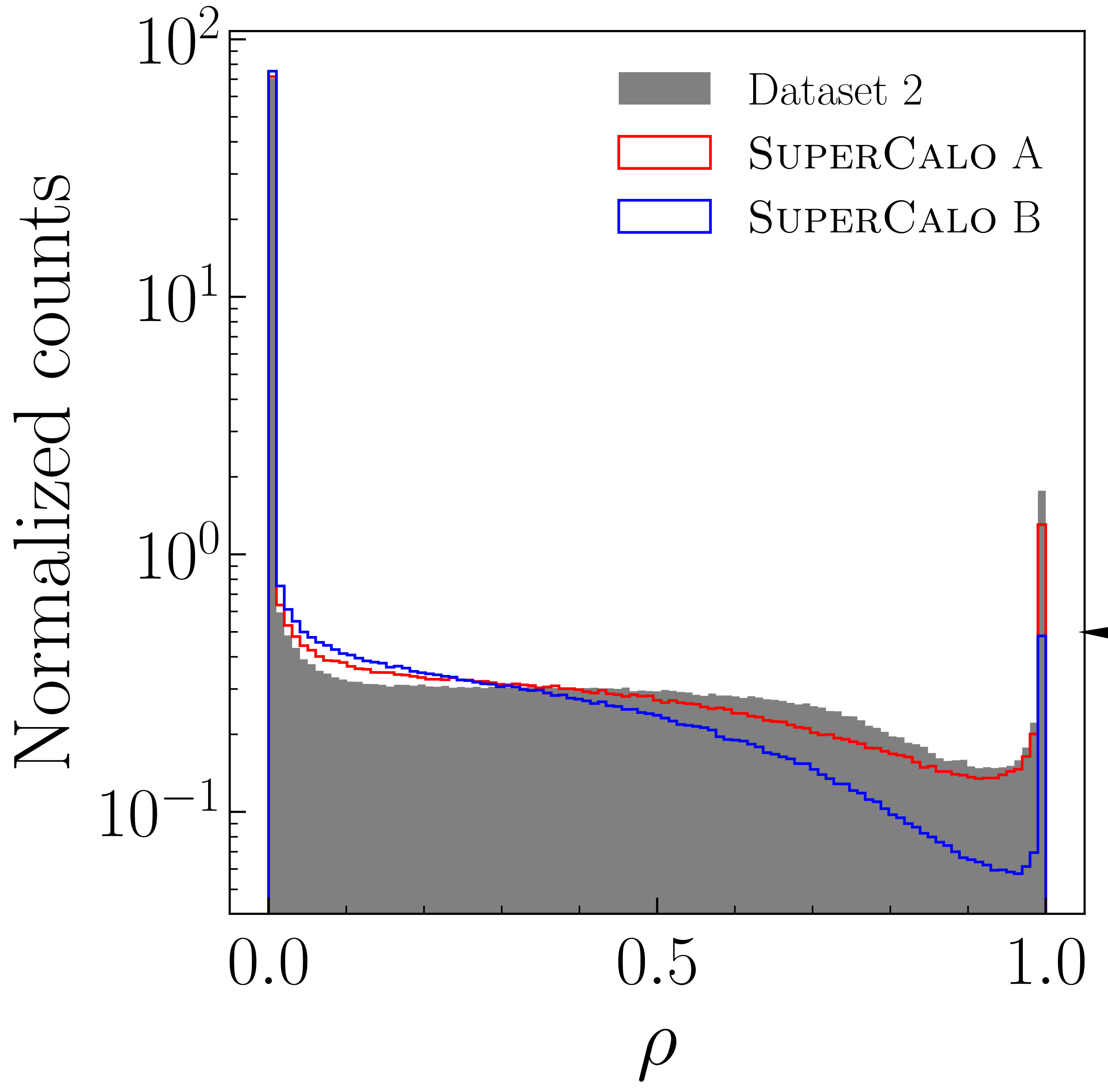
Choice B (only group in r and α) fails to capture inter-layer correlations!

Generate two sets of **high-res showers (fine voxels)** using choices A & B respectively

Compute ρ

Dot product of fine voxels contained in adjacent layers*

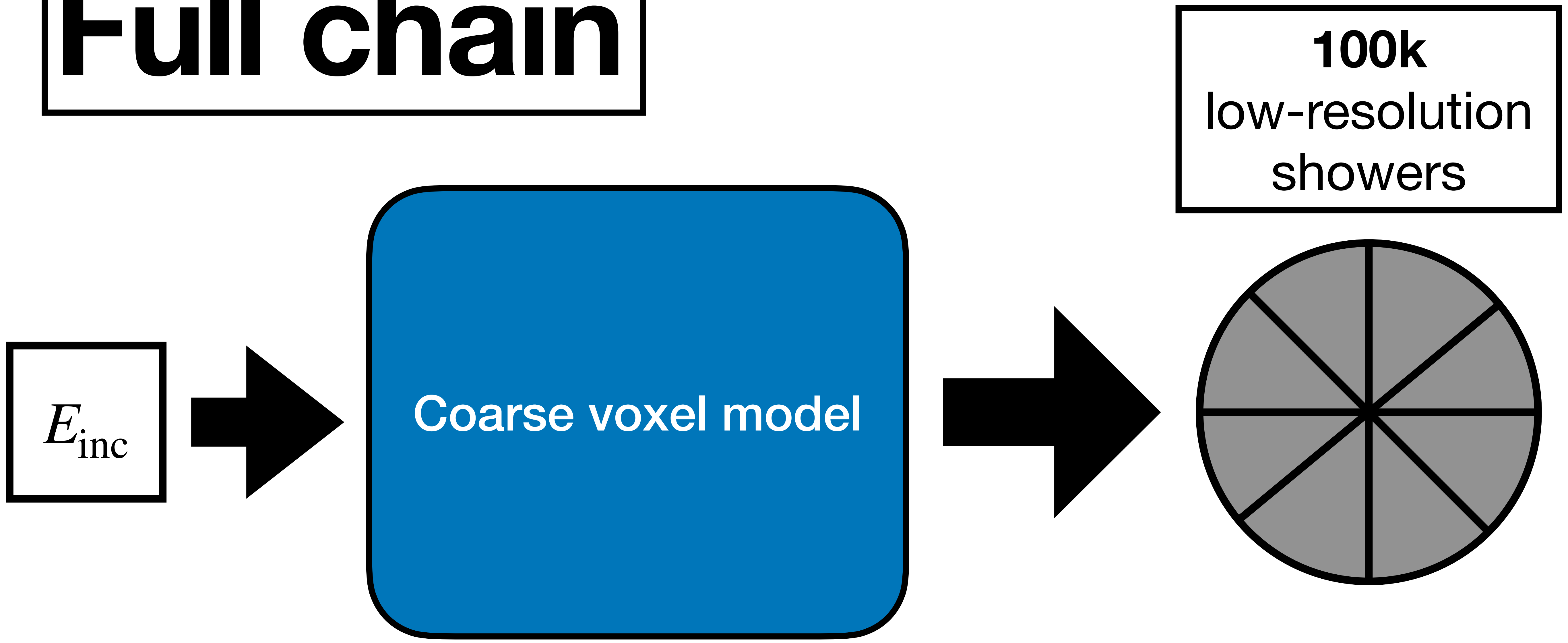
$$\rho \equiv \frac{\vec{e}^{\,(n)}_{\rm fine} \cdot \vec{e}^{\,(n+1)}_{\rm fine}}{\left|\vec{e}^{\,(n)}_{\rm fine}\right| \left|\vec{e}^{\,(n+1)}_{\rm fine}\right|}$$



Stick with Choice A for rest of talk!

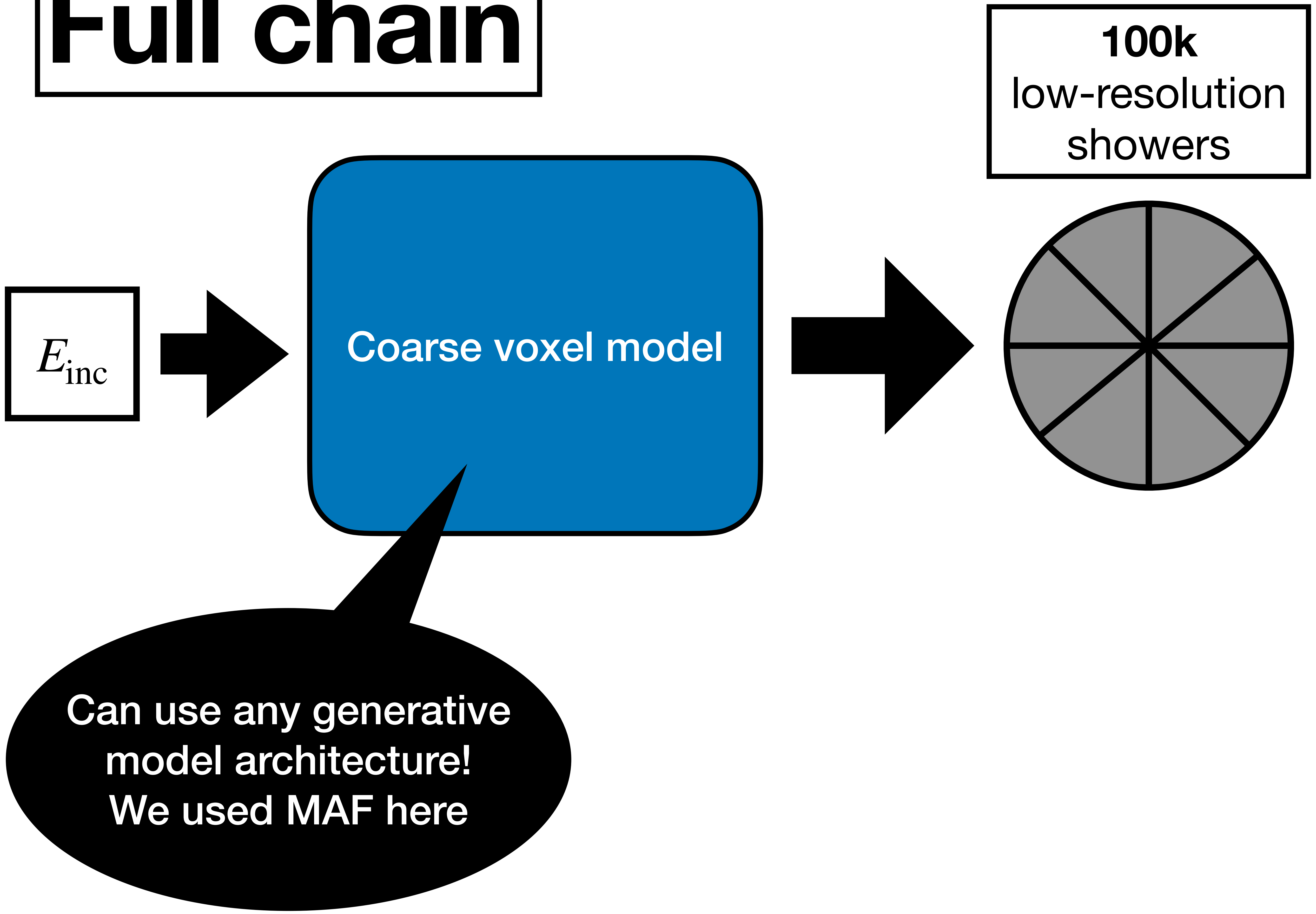
Full chain

Full chain

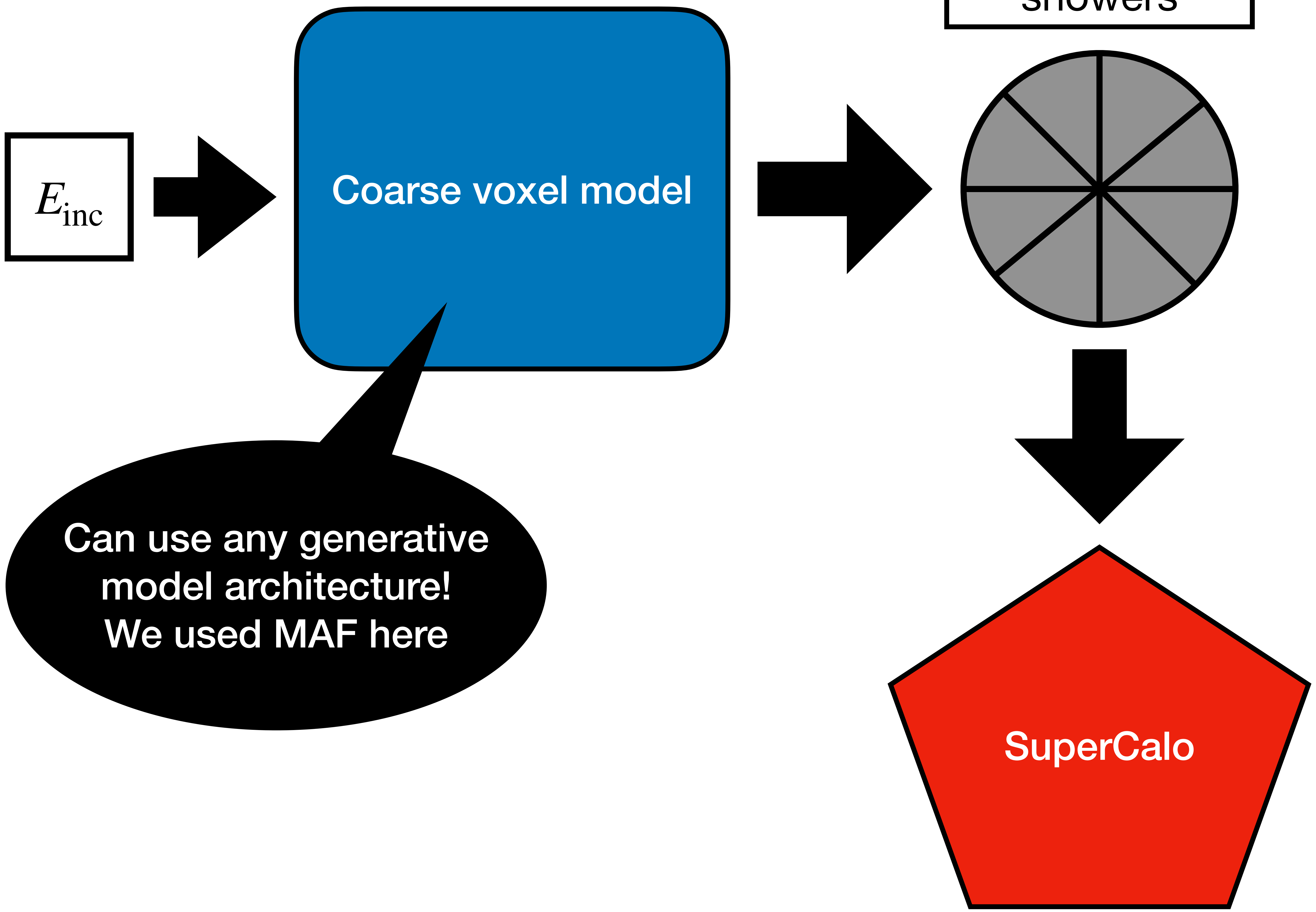


100k
low-resolution
showers
E_{inc}
Coarse voxel model

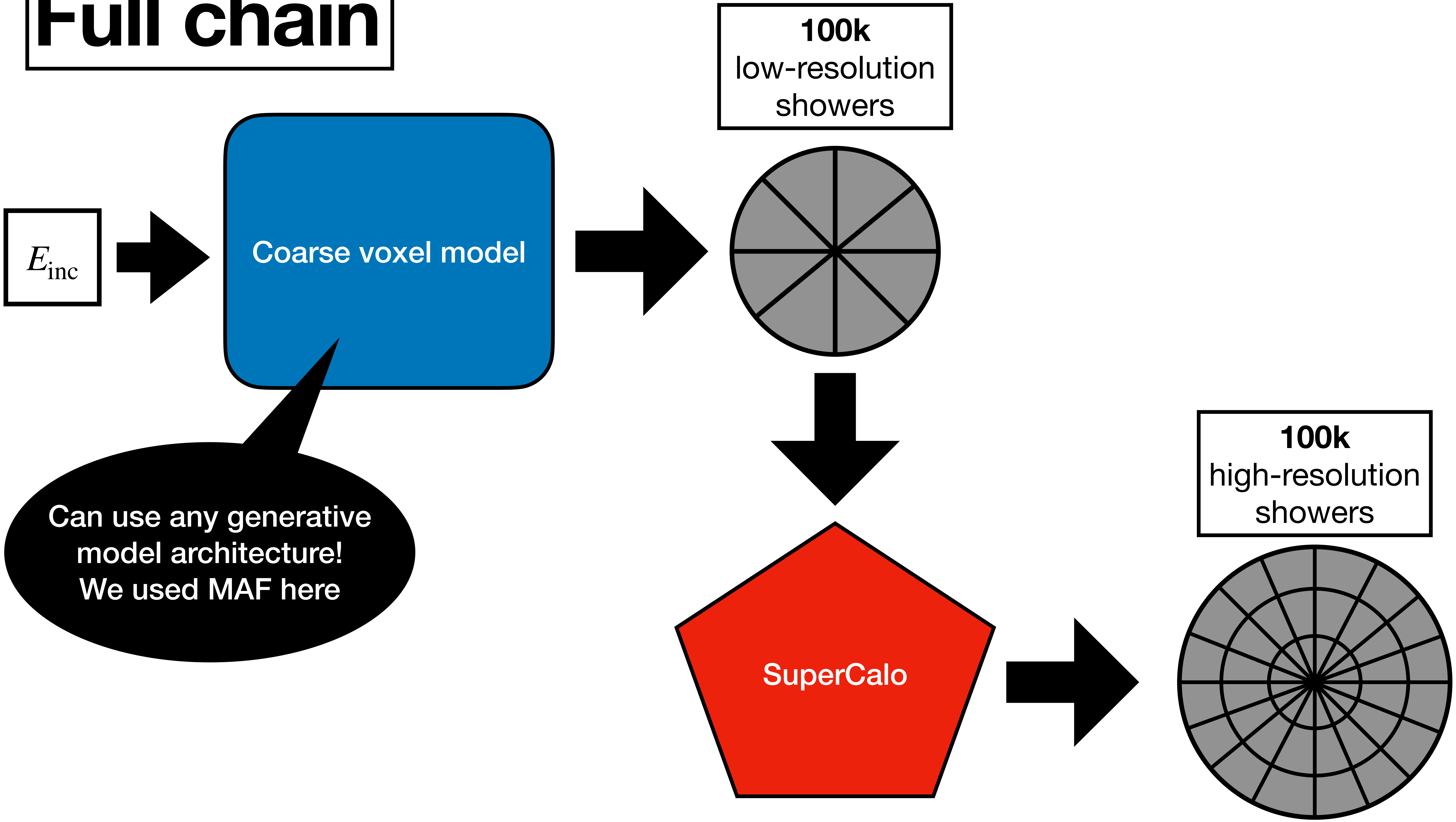
Full chain


100k
low-resolution
showers
E_{inc}
Coarse voxel model
Can use any generative
model architecture!
We used MAF here

Full chain


E_{inc}
Coarse voxel model
Can use any generative model architecture! We used MAF here
100k low-resolution showers
SuperCalo

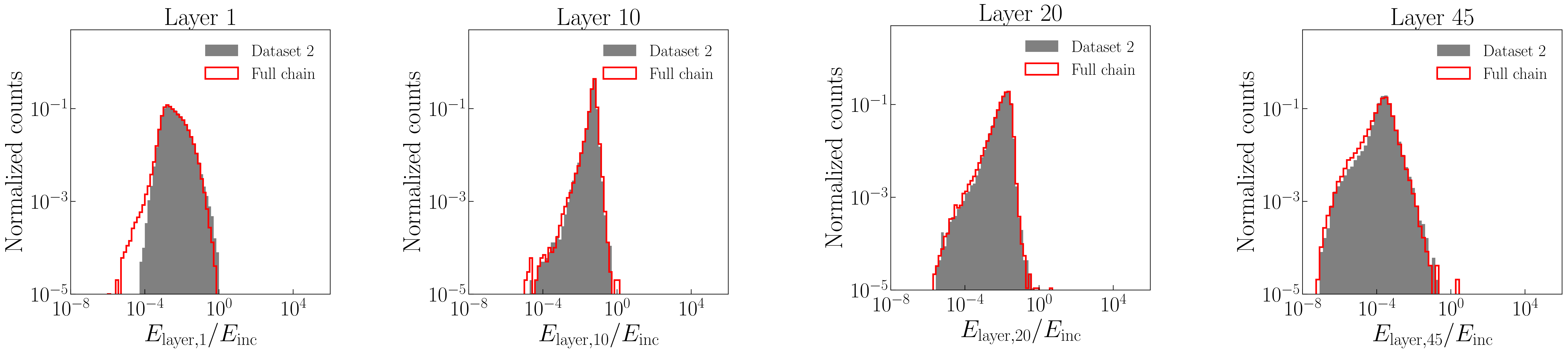
Full chain



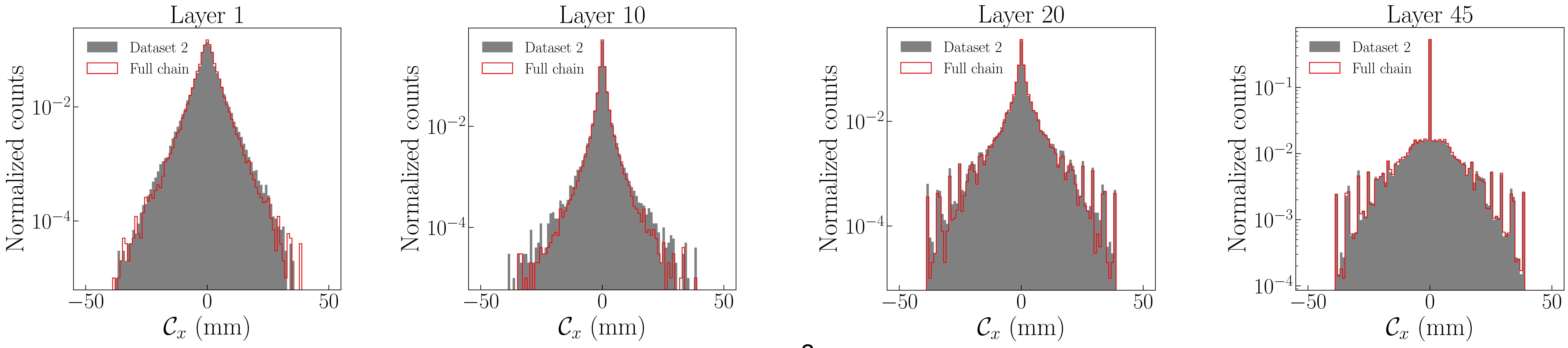
E_{inc}
Coarse voxel model
Can use any generative model architecture! We used MAF here
100k low-resolution showers
SuperCalo
100k high-resolution showers

Histograms of HLFs

Total deposited energy per layer

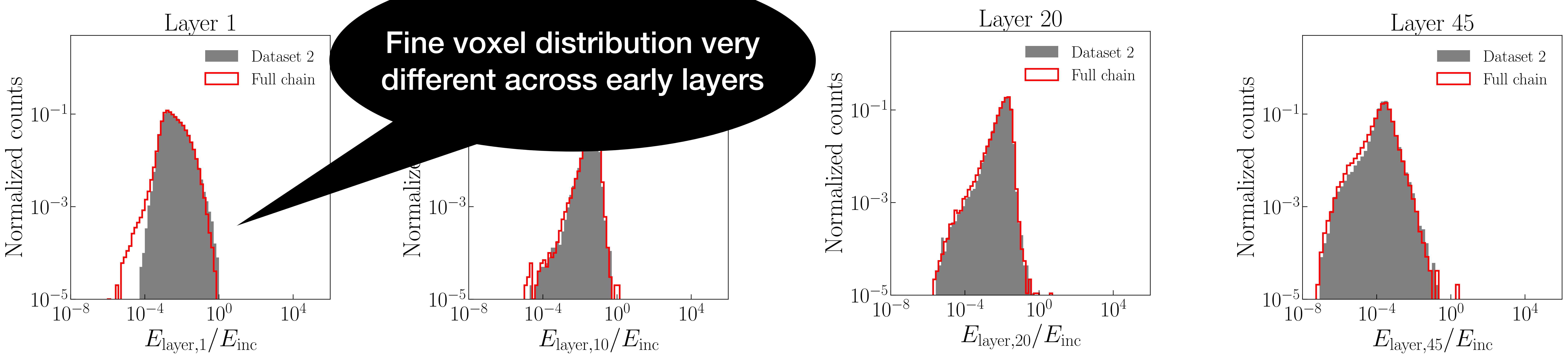


Shower shape (center of energy)

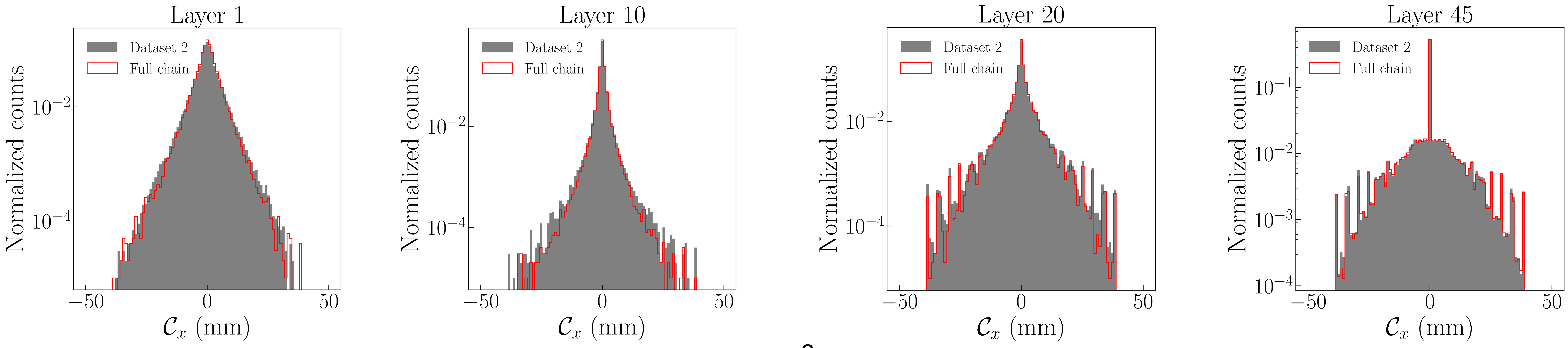


Histograms of HLFs

Total deposited energy per layer

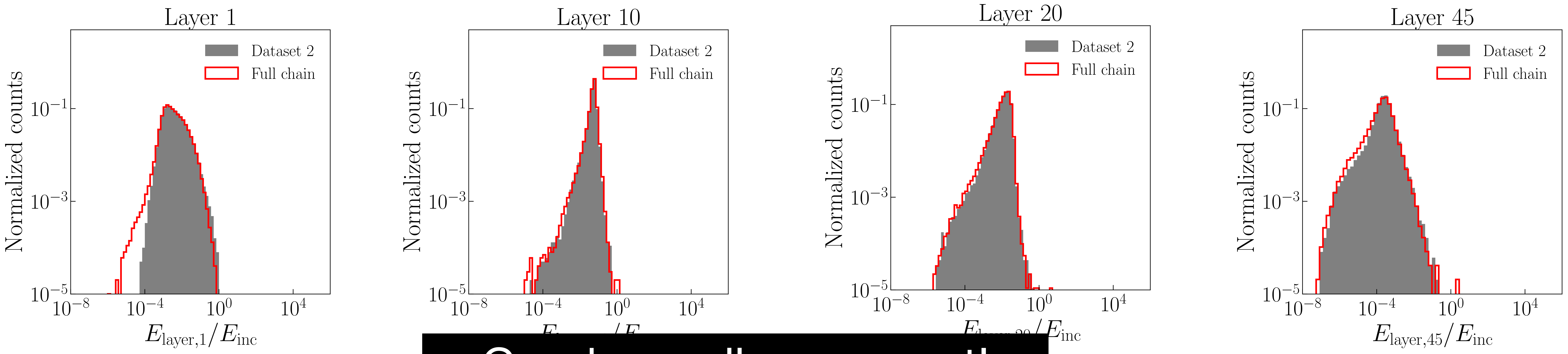


Shower shape (center of energy)

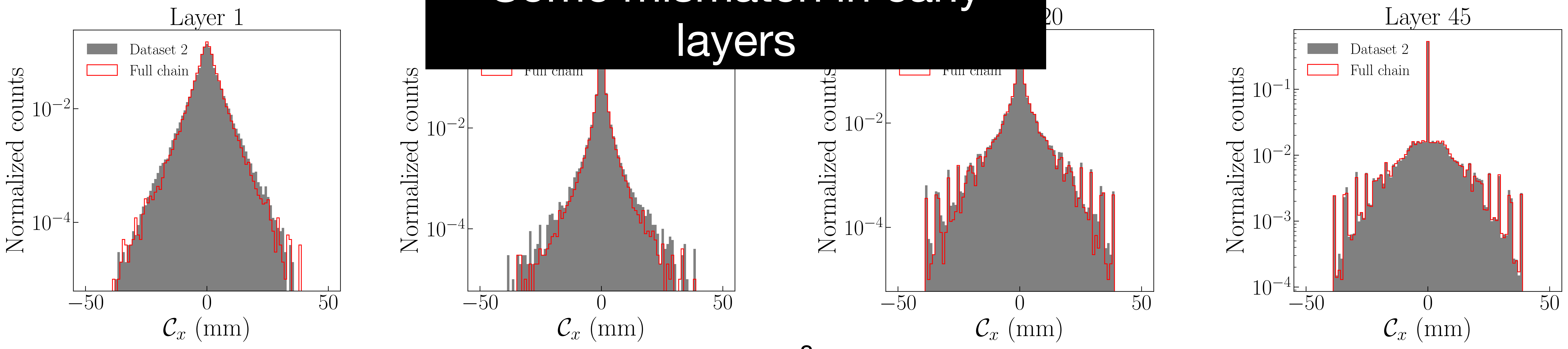


Histograms of HLFs

Total deposited energy per layer



Good overall agreement!
Some mismatch in early layers



Classifier test

- According to Neyman-Pearson lemma, we have $p_{\text{GEANT4}}(x) = p_{\text{generated}}(x)$ if optimal classifier cannot distinguish between two datasets
- Trained binary classifier directly on low-level (voxels) and high-level features of SuperCalo and Geant4 samples.

Classifier test

- According to Neyman-Pearson lemma, we have $p_{\mathrm{GEANT4}}(x) = p_{\mathrm{generated}}(x)$ if optimal classifier cannot distinguish between two datasets

- Trained binary classifier directly on low-level (voxels) and high-level features of SuperCalo and Geant4 samples.

	low-level features		high-level features	
Model	AUC	JSD	AUC	JSD
Full chain	**0.726(19)**	**0.117(19)**	**0.715(3)**	**0.110(4)**
iCaloFlow	0.797(5)	0.210(7)	0.798(3)	0.214(5)

iCaloFlow [2305.11934]: M. Buckley, C. Krause, **IP**, D. Shih

Classifier test

- According to Neyman-Pearson lemma, we have $p_{\text{GEANT4}}(x) = p_{\text{generated}}(x)$ if optimal classifier cannot distinguish between two datasets

- Trained binary classifier directly on low-level (voxe[illegible]evel features of SuperCalo and Geant4 samples.

All AUC and JSD < 1
$\Longrightarrow$High-fidelity samples

	low-level features		[illegible]	
Model	AUC	JSD	AUC	JSD
Full chain	**0.726(19)**	**0.117(19)**	**0.715(3)**	**0.110(4)**
iCaloFlow	0.797(5)	0.210(7)	0.798(3)	0.214(5)

iCaloFlow [2305.11934]: M. Buckley, C. Krause, **IP**, D. Shih

Classifier test

- According to Neyman-Pearson lemma, we have $p_{\mathrm{GEANT4}}(x) = p_{\mathrm{generated}}(x)$ if optimal classifier cannot distinguish between two datasets

- …ifier directly on low-level (voxels) and high-level features of …4 samples.

Better performance compared to another MAF-based model

Model	low-level features		high-level features	
	AUC	JSD	AUC	JSD
Full chain	**0.726(19)**	**0.117(19)**	**0.715(3)**	**0.110(4)**
iCaloFlow	0.797(5)	0.210(7)	0.798(3)	0.214(5)

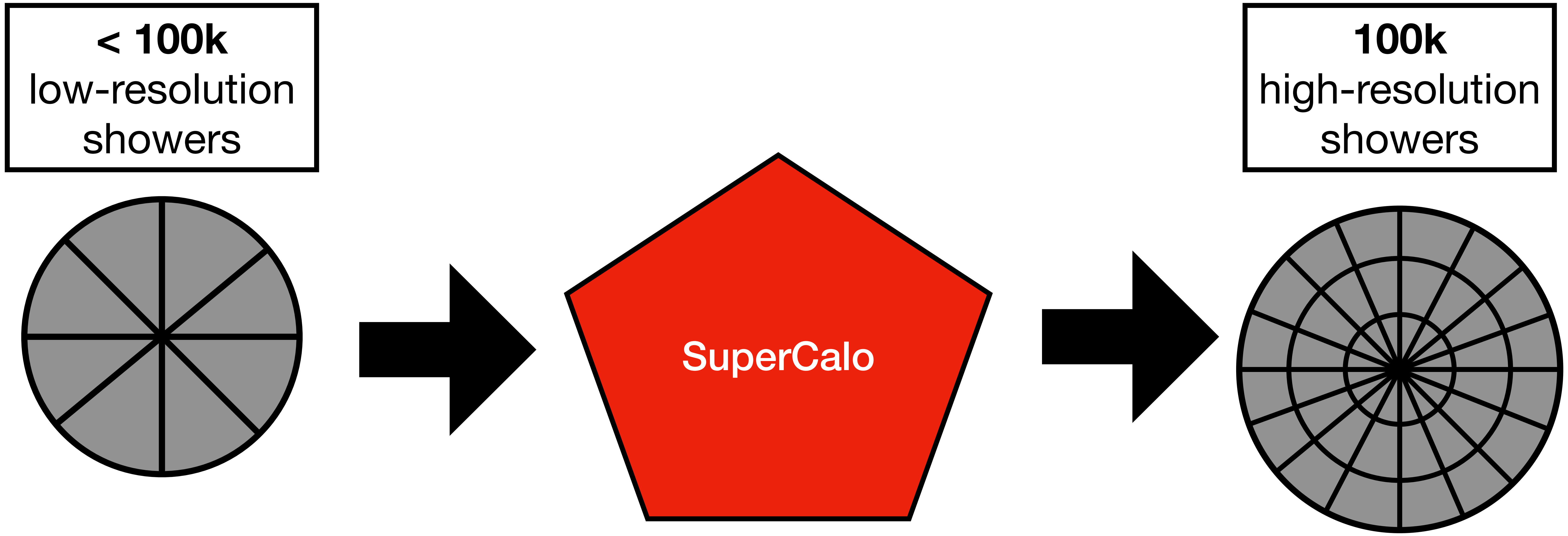
iCaloFlow [2305.11934]: M. Buckley, C. Krause, **IP**, D. Shih

Variation study

How much variation do we get from upsampling?

Variation study

How much variation do we get from upsampling?



Variation study

How much variation do we get from upsampling?

Number of coarse showers	low-level features		high-level features	
	AUC	JSD	AUC	JSD
2×10^4	0.762(3)	0.160(4)	0.724(2)	0.119(3)
1×10^4	0.795(4)	0.208(6)	0.738(4)	0.135(5)
5×10^3	0.852(4)	0.310(6)	0.759(3)	0.162(3)
2×10^3	0.938(2)	0.556(7)	0.818(3)	0.255(6)
1×10^3	0.980(1)	0.769(4)	0.887(4)	0.408(10)

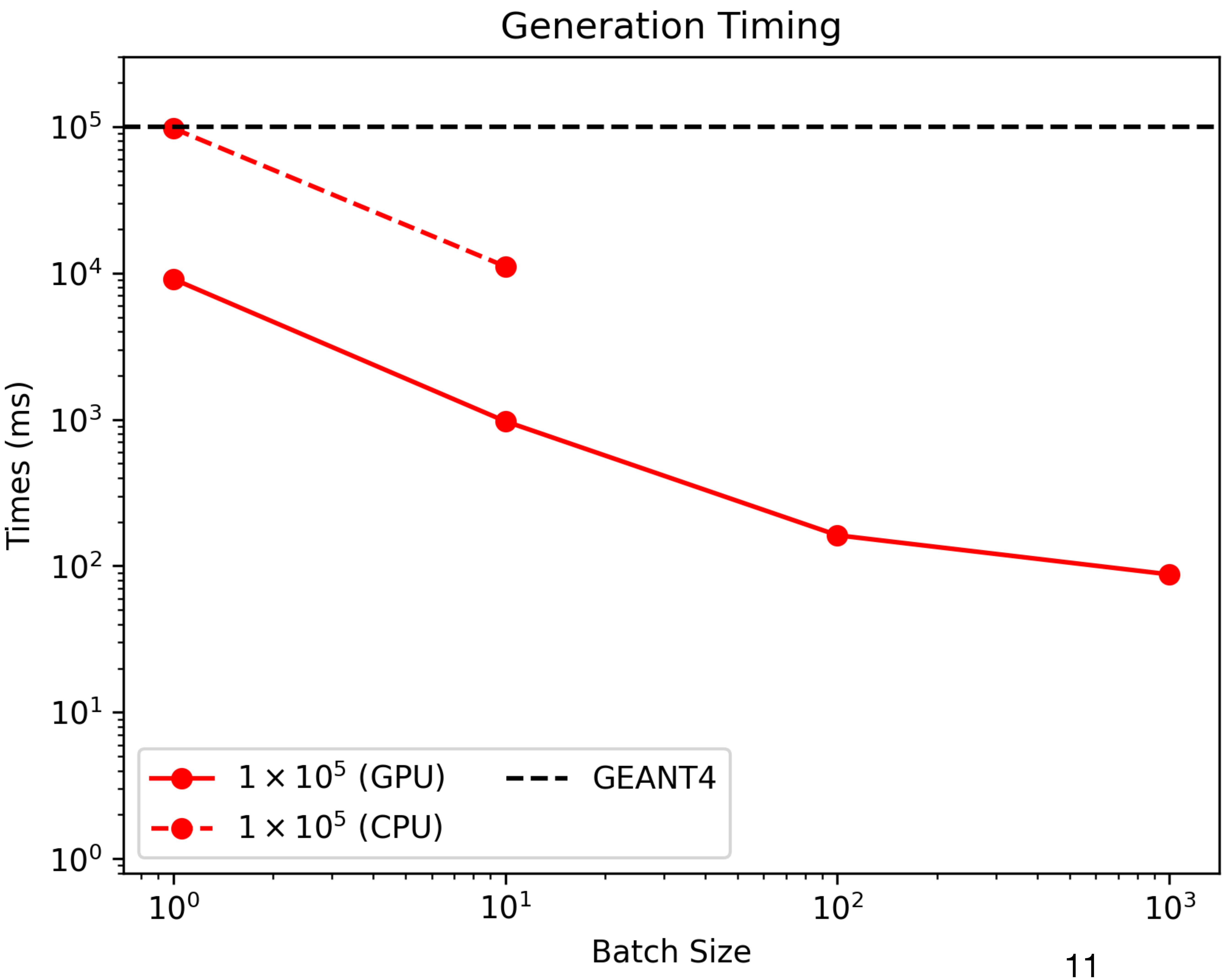
Variation study

How much variation do we get from upsampling?

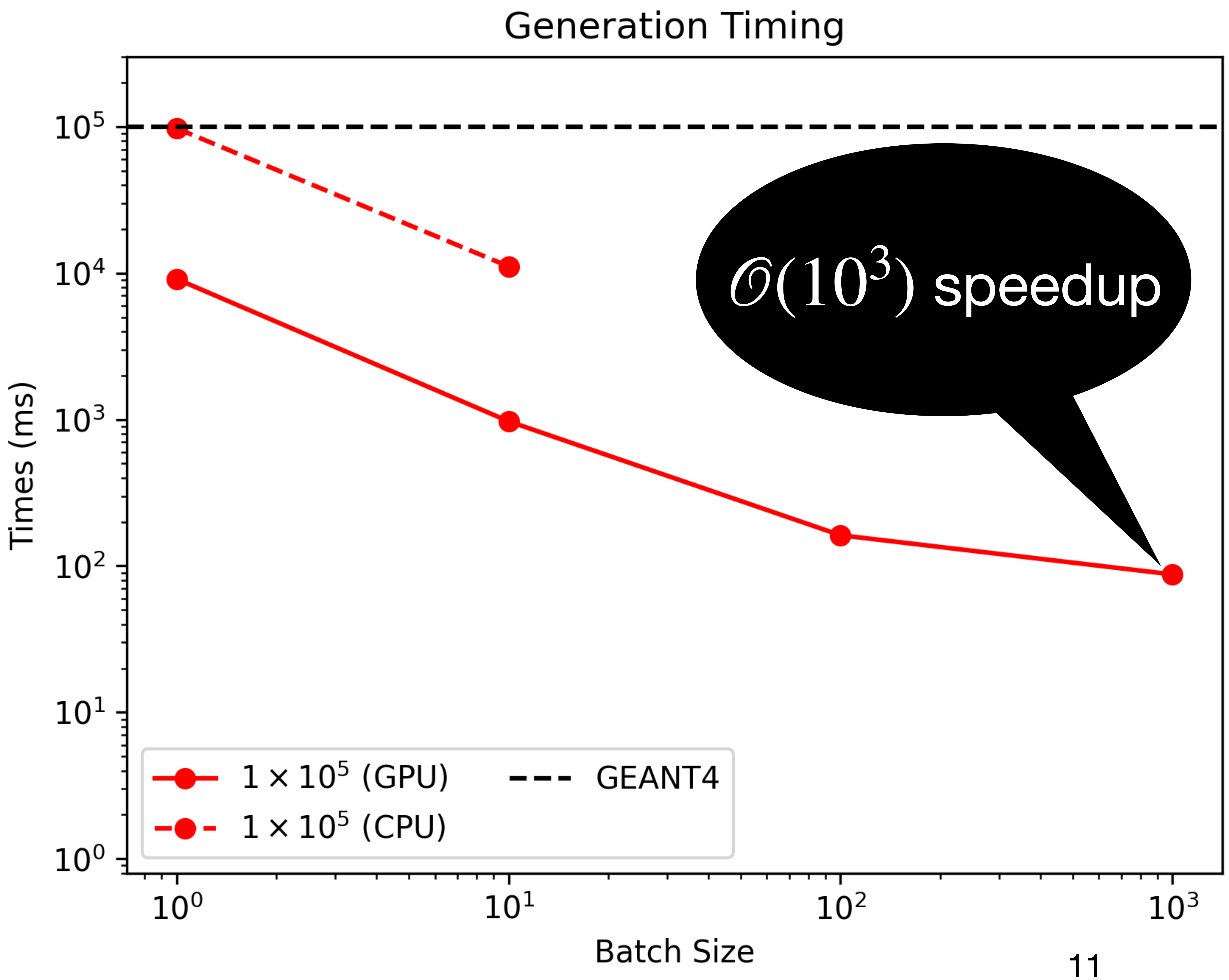
Number of coarse showers	low-level features AUC	low-level features JSD	high-level features AUC	high-level features JSD
2×10^4	0.762(3)	0.160(4)	[illegible]	[illegible]
1×10^4	0.795(4)	0.208(6)	[illegible]	[illegible]
5×10^3	0.852(4)	0.310(6)	0.759(3)	[illegible]
2×10^3	0.938(2)	0.556(7)	0.818(3)	0.255(6)
1×10^3	0.980(1)	0.769(4)	0.887(4)	0.408(10)

Classifier only fully catches on when we start from 1000 coarse showers!

Timing

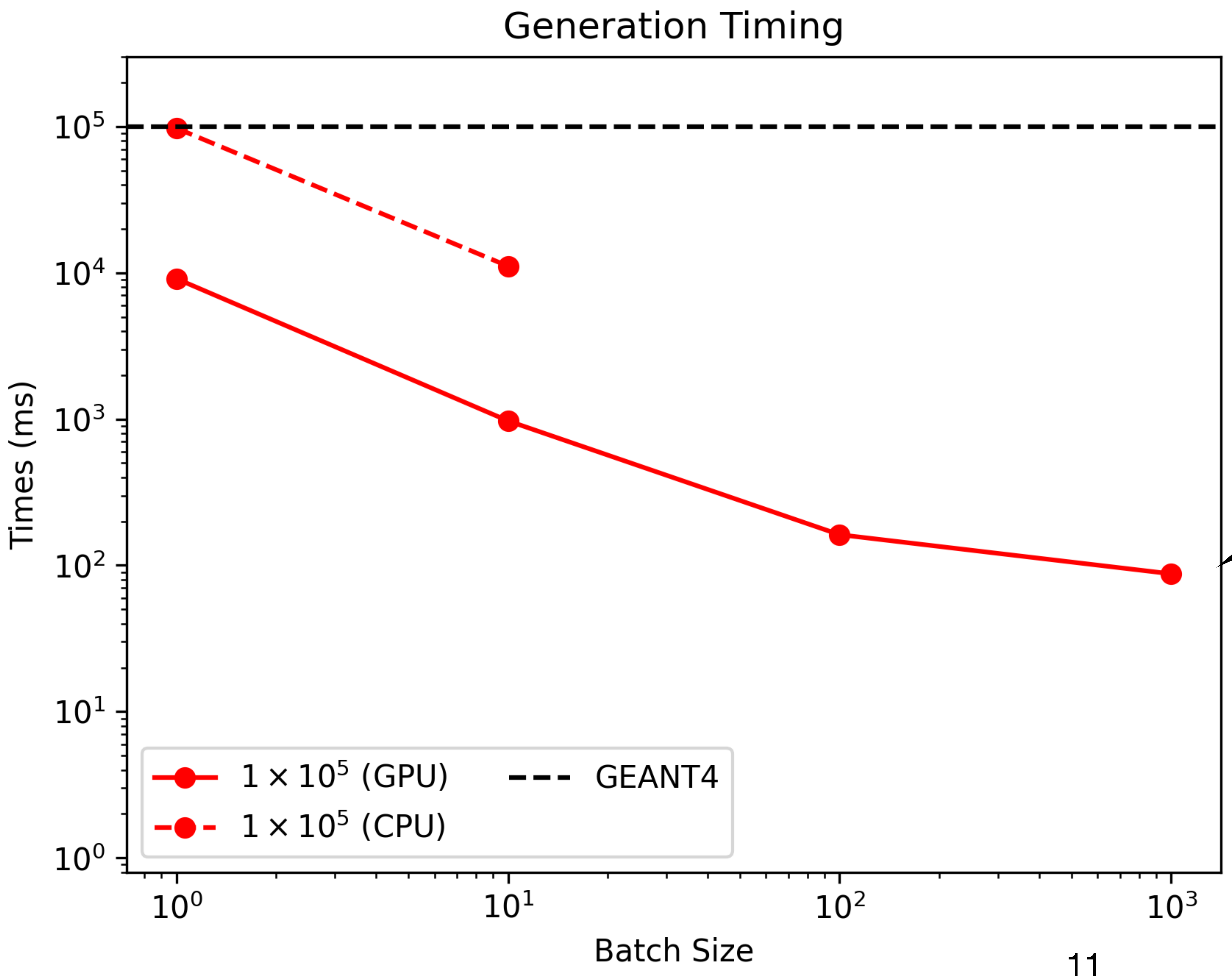

Generation Timing
Times (ms)
Batch Size
10^0
10^1
10^2
10^3
10^4
10^5
1×10^5 (GPU)
1×10^5 (CPU)
GEANT4

Timing



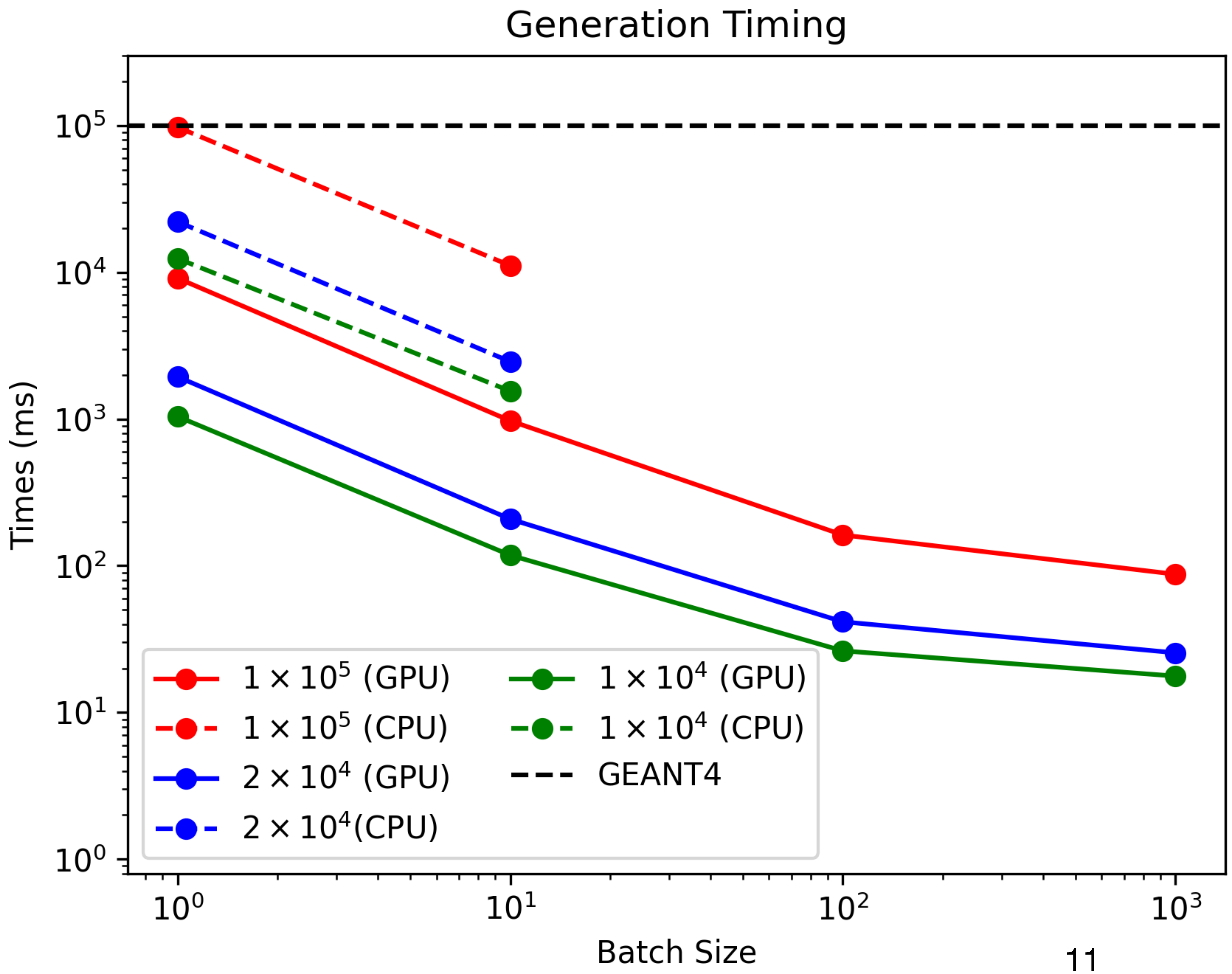
Generation Timing
10^5
10^4
10^3
10^2
10^1
10^0
Times (ms)
10^0
10^1
10^2
10^3
Batch Size
1×10^5 (GPU)
1×10^5 (CPU)
GEANT4
$\mathcal{O}(10^3)$ speedup

Timing



Factor of ~1.9 speedup compared to iCaloFlow MAF

Timing



Additional speedup by resampling low-res

Conclusions and Outlook

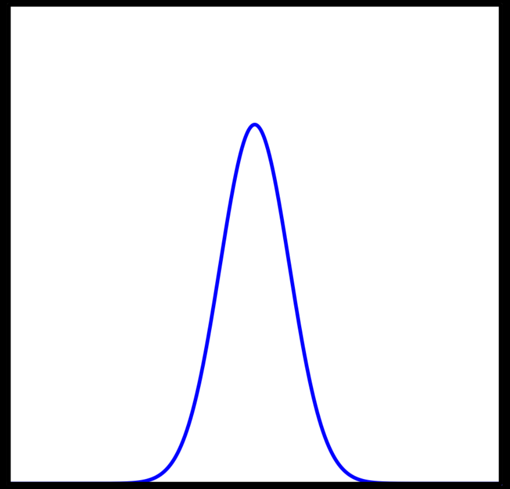
- Generate high-dim showers by **upsampling coarse showers** with SuperCalo
- Achieved $\mathcal{O}(10^3)$ **speedup** vs GEANT4 and **~1.9 speedup** vs approach with similar architecture
- SuperCalo generates high-dim showers with **substantial variation**
- SuperCalo approach can be **generalized to any** generative model architecture

Thank you!

Backup

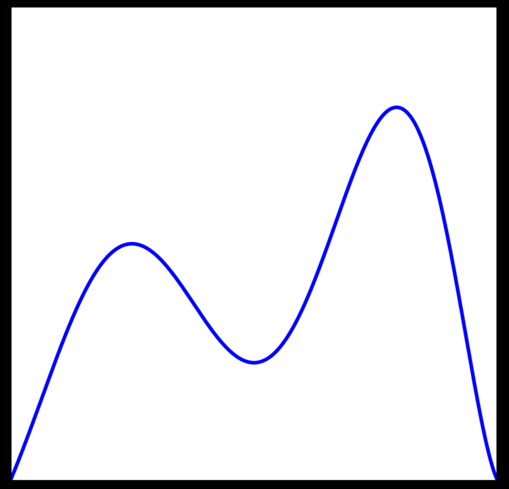
Normalizing Flows

Latent space
with simple base distribution
z ~ Z



Bijective transformation f

Data space
with more complex distribution
x ~ X



$$p(x) = \pi(f(x)) \left| \det \frac{\partial f(x)}{\partial x} \right|$$

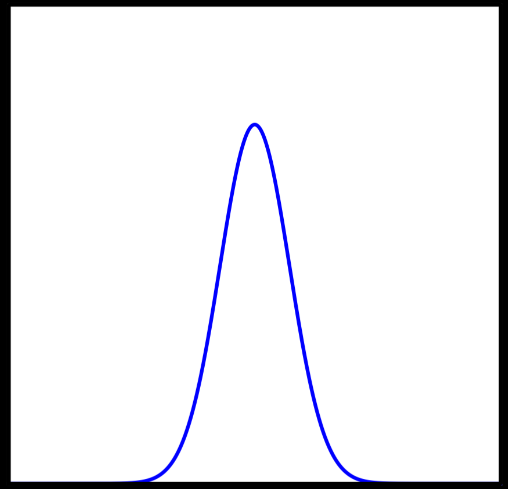
where $f(x) = z$

Density estimation, $p(x)$

Sample generation

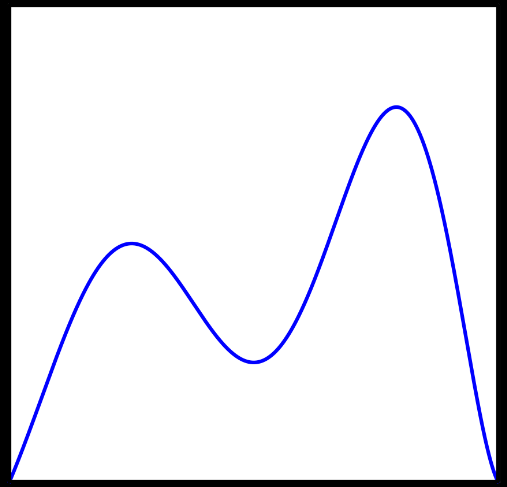
Normalizing Flows

Latent space
with simple base distribution
z ~ Z



Bijective transformation f

Data space
with more complex distribution
x ~ X



$$p(x) = \pi(f(x)) \left| \det \frac{\partial f(x)}{\partial x} \right|$$

where $f(x) = z$

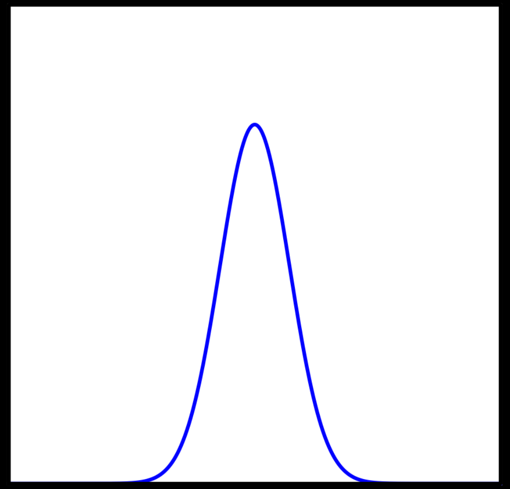
Needs to be tractable!

Density estimation, $p(x)$

Sample generation

Normalizing Flows

Latent space
with simple base distribution
z ~ Z

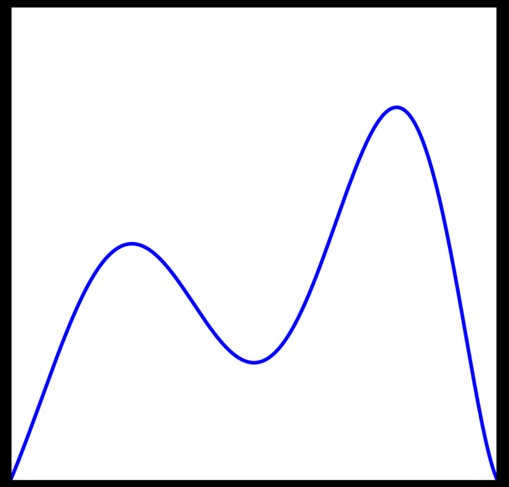


Bijective transformation f

$$p(x) = \pi(f(x)) \left| \det \frac{\partial f(x)}{\partial x} \right|$$

where $f(x) = z$

Data space
with more complex distribution
x ~ X



Density estimation, $p(x)$

Sample generation

Conditional inputs

1. Incident energy of the incoming particle, E_{inc}
2. Deposited energy in coarse voxel i, $E_{\text{coarse},i}$
3. Fine layer energies of layers spanned by coarse voxel i
4. Deposited energy in neighboring coarse voxels in α, r and z directions
5. One-hot encoded coarse layer number
6. One-hot encoded coarse radial bin

Architecture and training

- **MAF-RQS** flows for all models
- Noise added to voxels during training
  - Uniform random of noise **[0, 1] keV**
- LR schedule: **OneCycle LR**

Model	dim of base distribution	number of MADE blocks	layer sizes: input	layer sizes: hidden	layer sizes: output	number of RQS bins	RQS tail bound
Flow-1	45	8	256	1×256	1035	8	14
Flow-2	648	8	648	1×648	14904	8	6
SuperCalo A	10	8	128	2×128	230	8	14
SuperCalo B	12	8	128	2×128	276	8	14

Pre-processing

- Flow-1
  - **Log**-transformed E_{inc} (E_{inc} normalized by constant) $E_{\mathrm{inc}} \to \log_{10} \frac{E_{\mathrm{inc}}}{10^{4.5}\ \mathrm{MeV}} \in [-1.5, 1.5].$
  - **Logit**-transformed $E_{\mathrm{layer},i}$ ($E_{\mathrm{layer},i}$ normalized by constant) $E_{\mathrm{layer},i} \to x_i \equiv (E_{\mathrm{layer},i} + \mathrm{rand}[0, 5\ \mathrm{keV}])/65\ \mathrm{GeV}$

$$y_i = \log \frac{u_i}{1-u_i}, \quad u_i \equiv \alpha + (1-2\alpha)x_i,$$

- Flow-2
  - **Log**-transformed $E_{\mathrm{coarse},i}$ ($E_{\mathrm{coarse},i}$ normalized by constant) $E_{\mathrm{coarse},i} \to \log_{10}\left((E_{\mathrm{coarse},i} + \mathrm{rand}[0, 5\ \mathrm{keV}])/E_{\mathrm{coarse,max}}\right)+6$
  - **Log**-transformed $E^{(\mathrm{coarse})}_{\mathrm{layer},i}$ $E^{(\mathrm{coarse})}_{\mathrm{layer},i} \to \frac{1}{4} \log_{10}\left(E^{(\mathrm{coarse})}_{\mathrm{layer},i}\right)$

Pre-processing

- SuperCalo
  - **Logit**-transformed $E_{\mathrm{coarse},i}$ ($E_{\mathrm{coarse},i}$ normalized by constant)

$$E_{\mathrm{coarse},i} \to \tilde{x}_i \equiv (E_{\mathrm{coarse},i} + \mathrm{rand}[0, 1\ \mathrm{keV}])/E_{\mathrm{coarse,max}}$$

$$\tilde{y}_i = \log \frac{\tilde{u}_i}{1-\tilde{u}_i}, \quad \tilde{u}_i \equiv \alpha + (1-2\alpha)\tilde{x}_i$$

  - **Logit**-transformed e_{ij} (e_{ij} normalized by $E_{\mathrm{coarse},i}$)

$$e_{\mathrm{fine},ij} \to \hat{x}_{ij} \equiv (e_{\mathrm{fine},ij} + \mathrm{rand}[0, 0.1\ \mathrm{keV}])/E_{\mathrm{coarse,i}}$$

$$\hat{y}_{ij} = \log \frac{\hat{u}_{ij}}{1-\hat{u}_{ij}}, \quad \hat{u}_{ij} \equiv \alpha + (1-2\alpha)\hat{x}_{ij}$$