FastSim Parametrization of Beam Dump using Generative Adversarial Network (GAN)

Oleksander Borysov, Alon Levi, Arka Santra, Noam Tal Hod

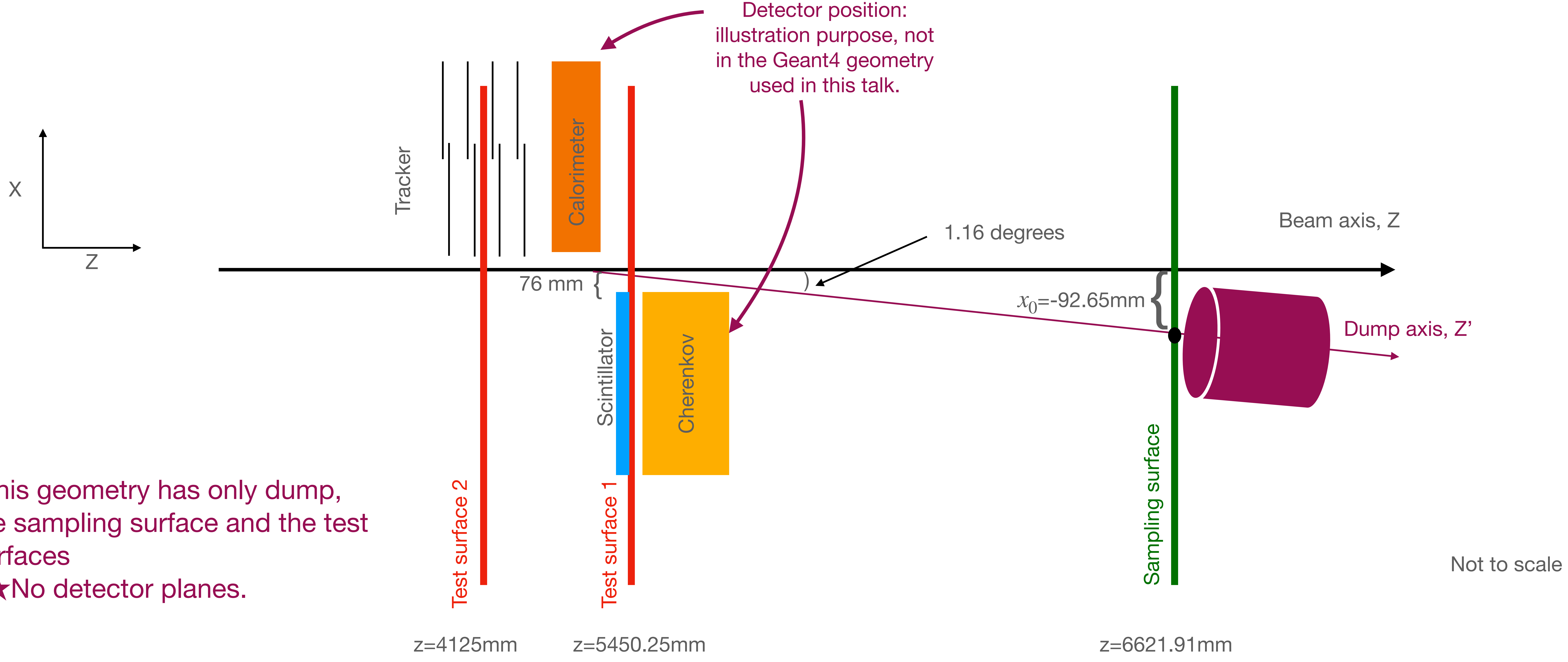
Feb 12, 2024

LUXE SAS meeting

Overview

- Recap of previous talk given by Arka Santra (Oct 30)
- Updates and changes
- Summary

Schematic diagram of the dump in the LUXE geometry



★This geometry has only dump, the sampling surface and the test surfaces

★No detector planes.

Taken from Arka's slides*

Approaches to background simulation

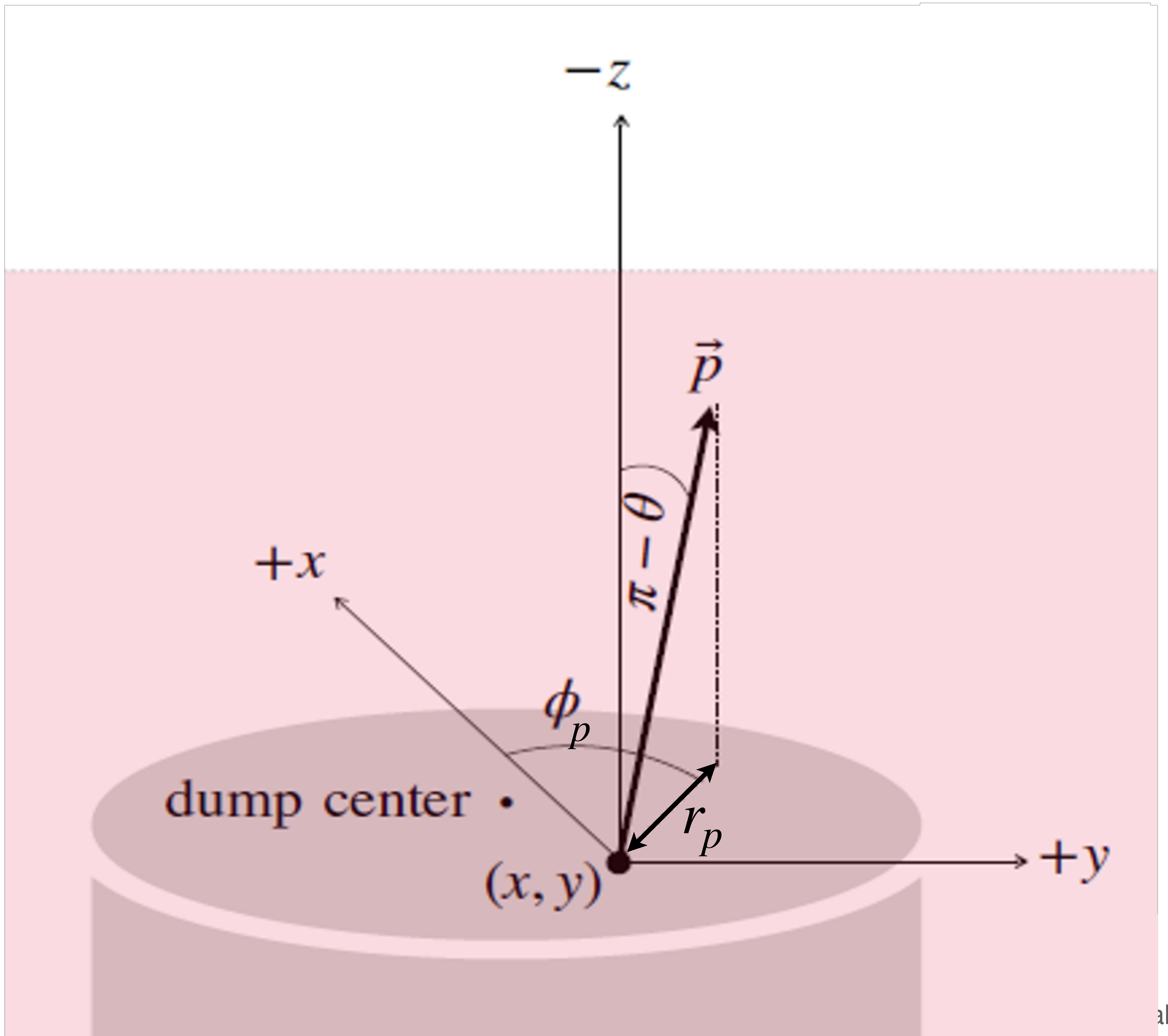
- FullSim - full Geant4 simulation, computationally heavy
- FastSim (correlation sampling) - lacks accuracy for backscattered particles
- FastSim using Wasserstein GAN

Particle Features

$$\begin{bmatrix} x \\ y \\ r_p \\ \phi_p \\ p_z \\ t \end{bmatrix}$$

$$r_x = \sqrt{x^2 + y^2}$$

$$\phi_x = \arctan \frac{y}{x}$$



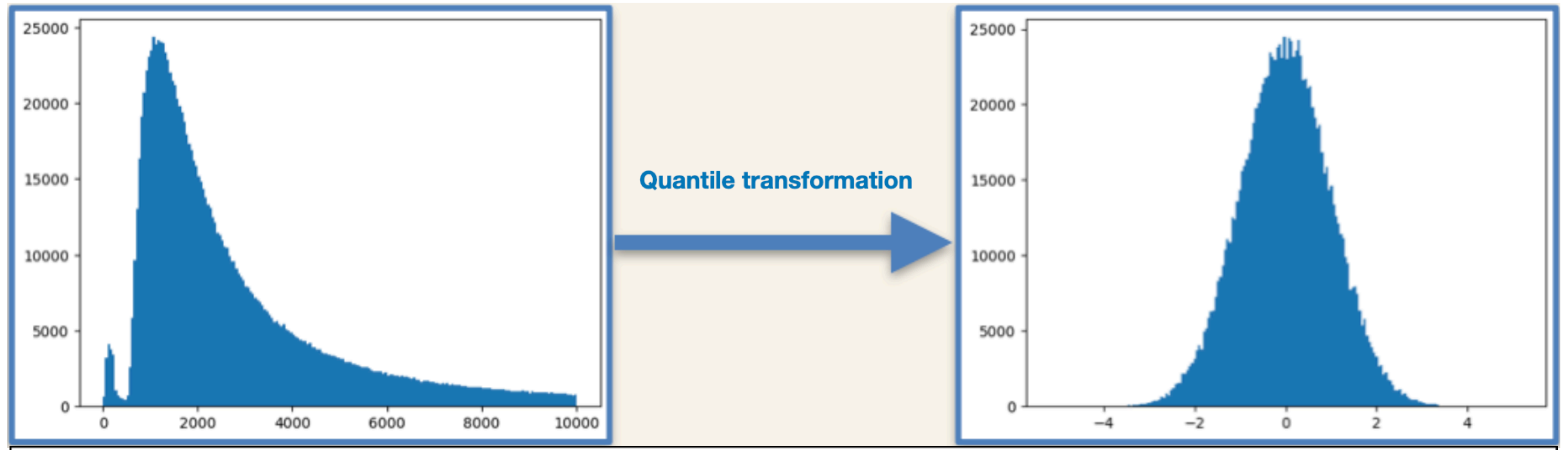
aken from Arka's slides*

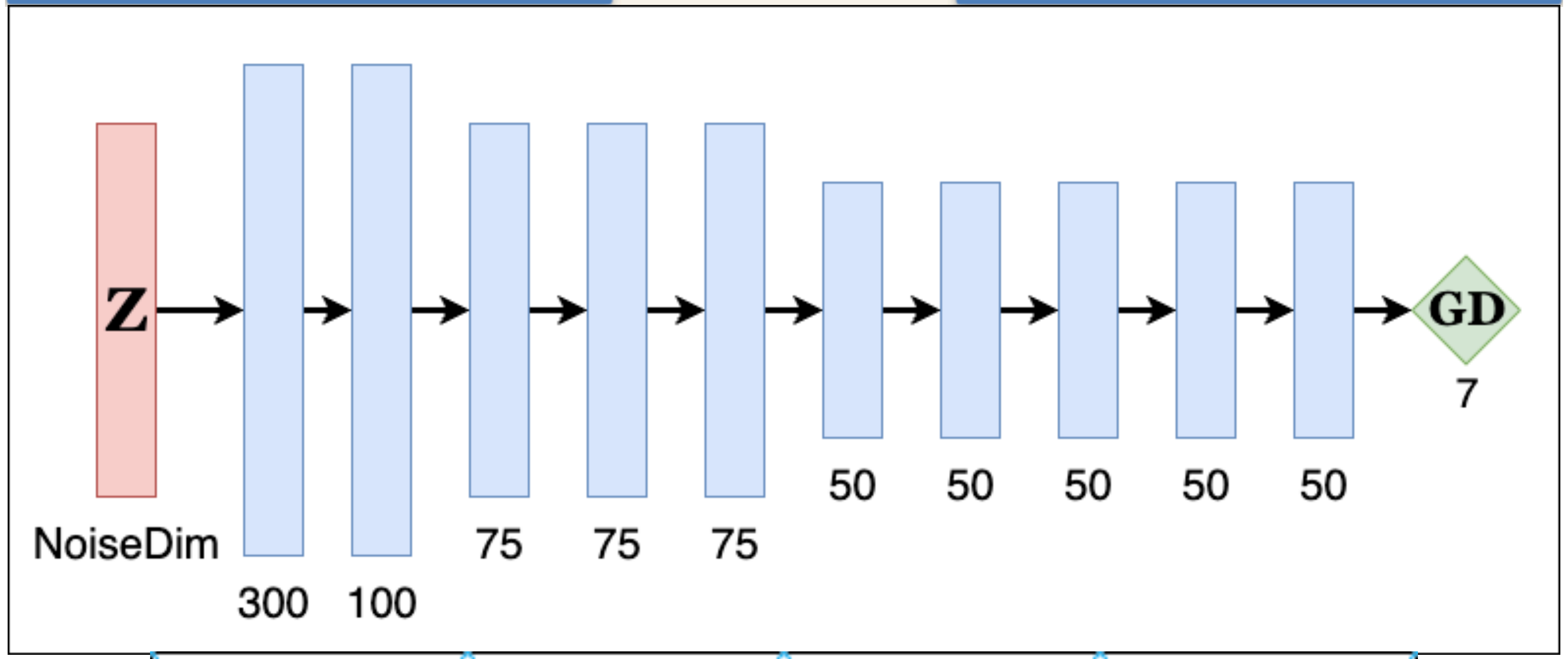
WGAN method

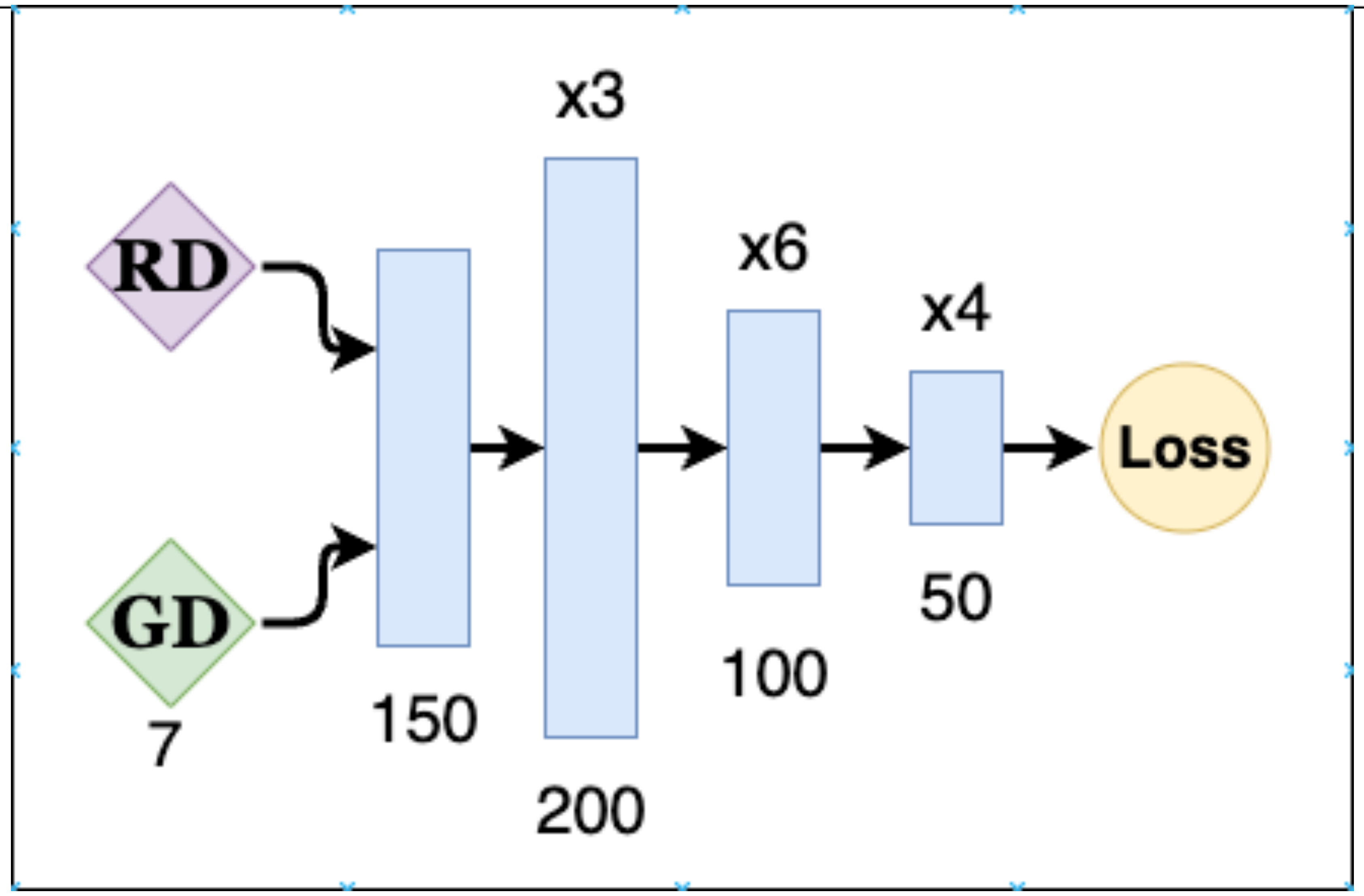
1. Apply quantile transformation to the data
2. Generate data points from random noise
3. Pass real and generated data through critic
4. Determine loss via Wasserstein metric
5. Monitor Progress with KL divergence
6. Apply inverse quantile transformation

$$\mathscr{L}(p_r, p(z)) = \max_{w\in W, \theta\in\Theta} \left[\underbrace{\mathbb{E}_{x\sim p_r}[f_w(x)]}_{\text{Expectation value from the original distribution}} - \underbrace{\mathbb{E}_{x\sim p(z)}[f_w(g_\theta(z))]}_{\text{Expectation value from the generated distribution}} \right]$$

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x\in\mathcal{X}} P(x)\, \log\left(\frac{P(x)}{Q(x)}\right)$$





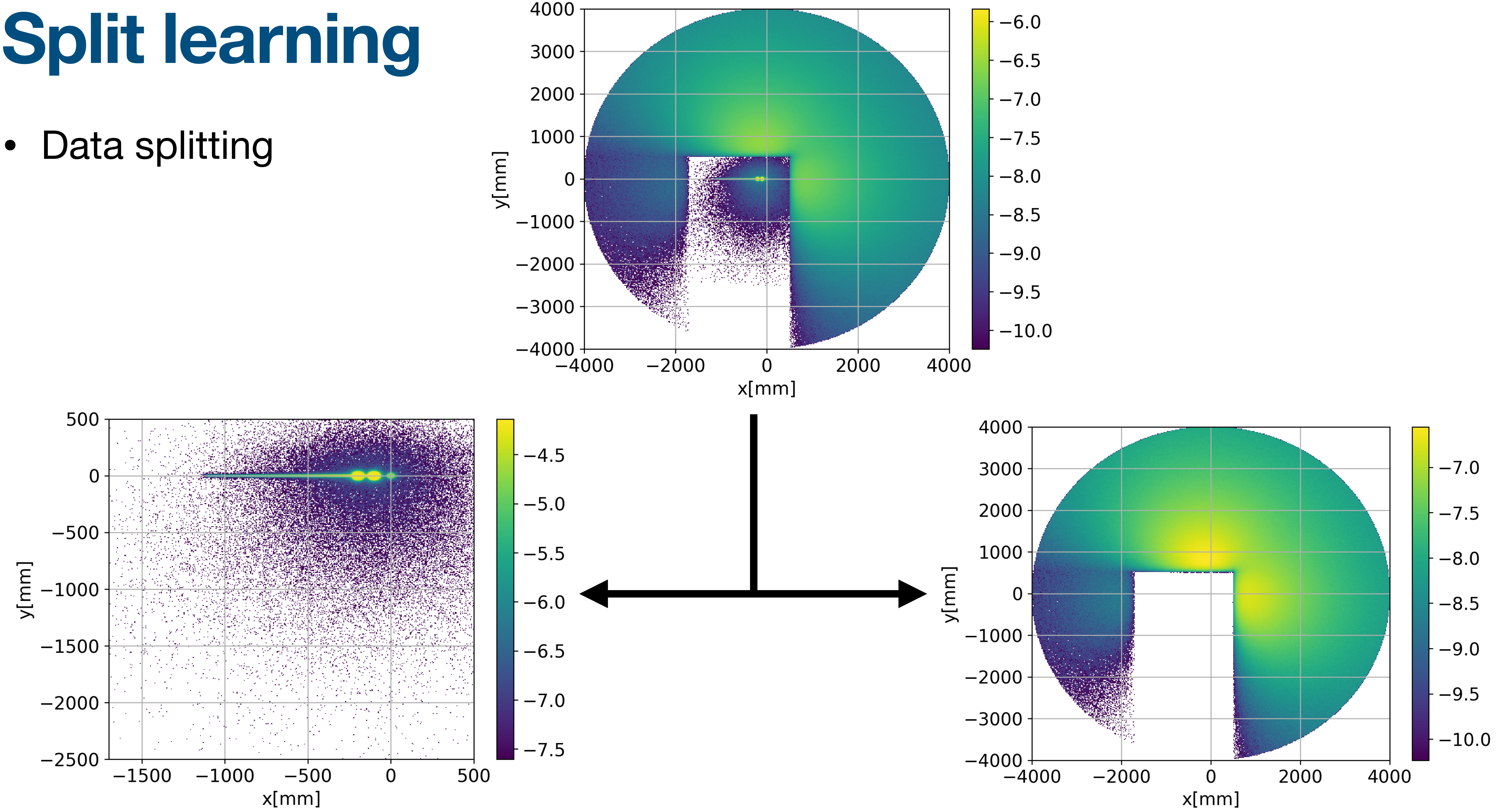


Improvements

- GAN struggles to learn sharp features
  - Larger networks
  - Decrease radius - focused learning
  - Split learning

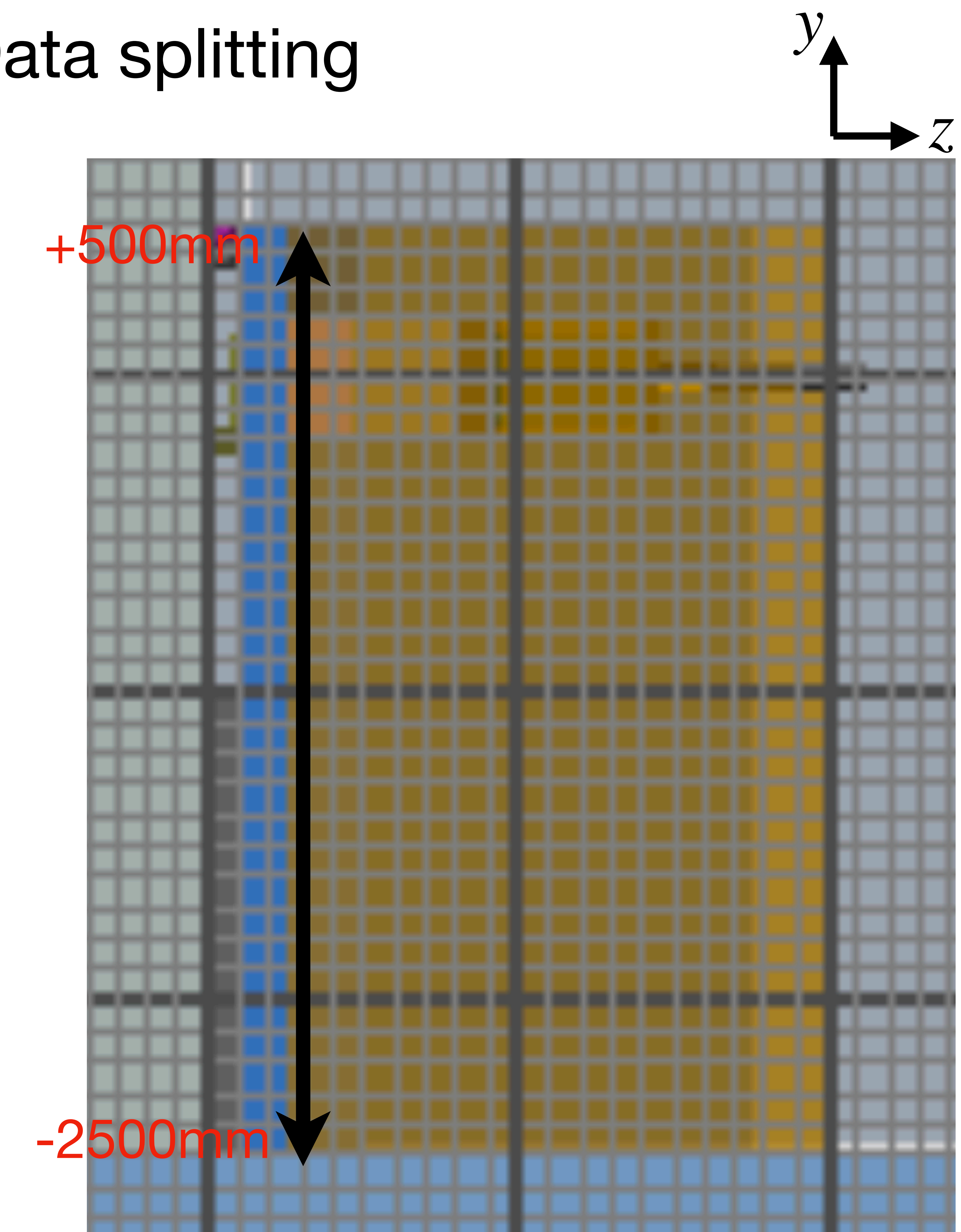
Split learning

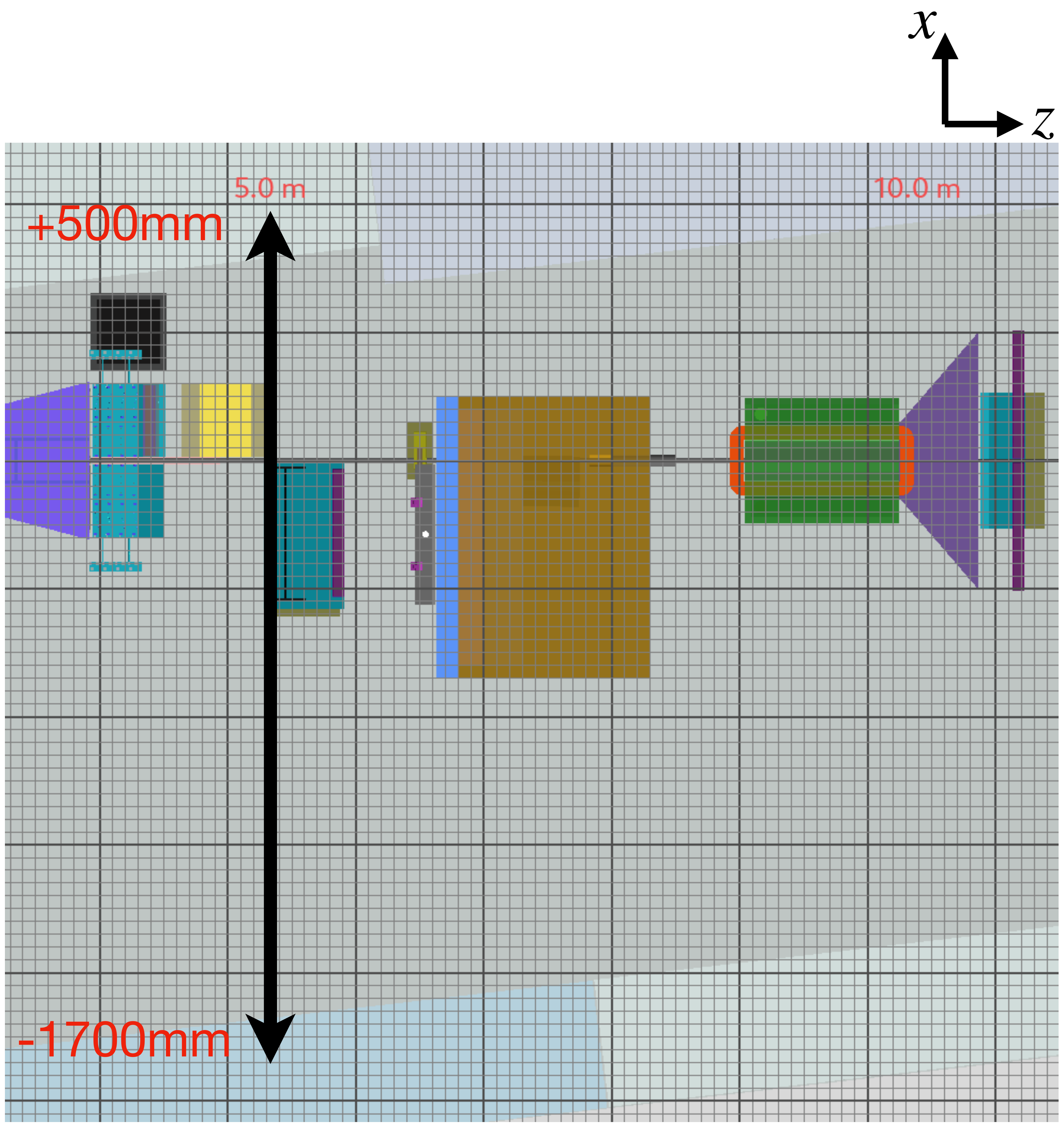
- Data splitting



Split learning

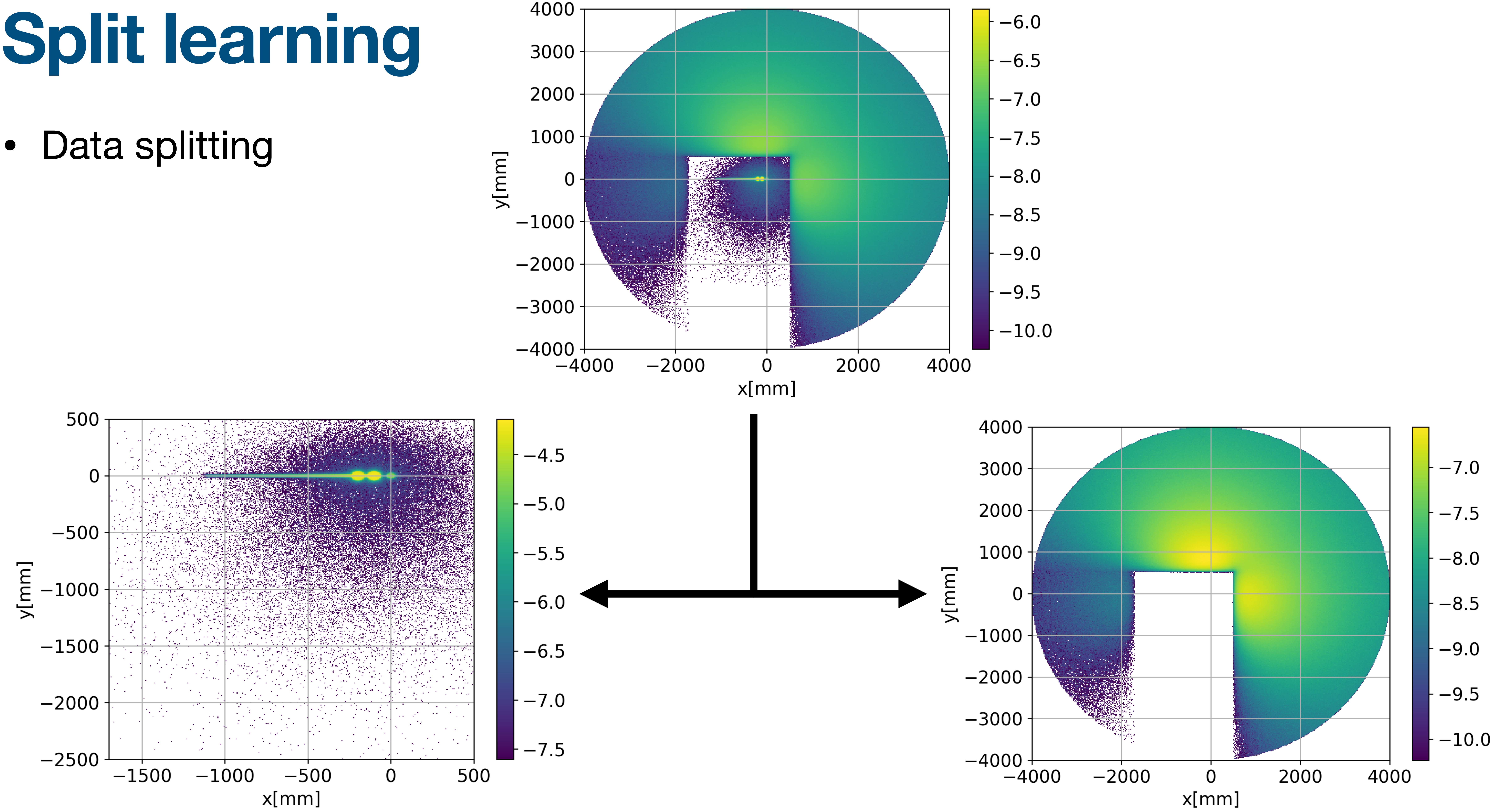
- Data splitting



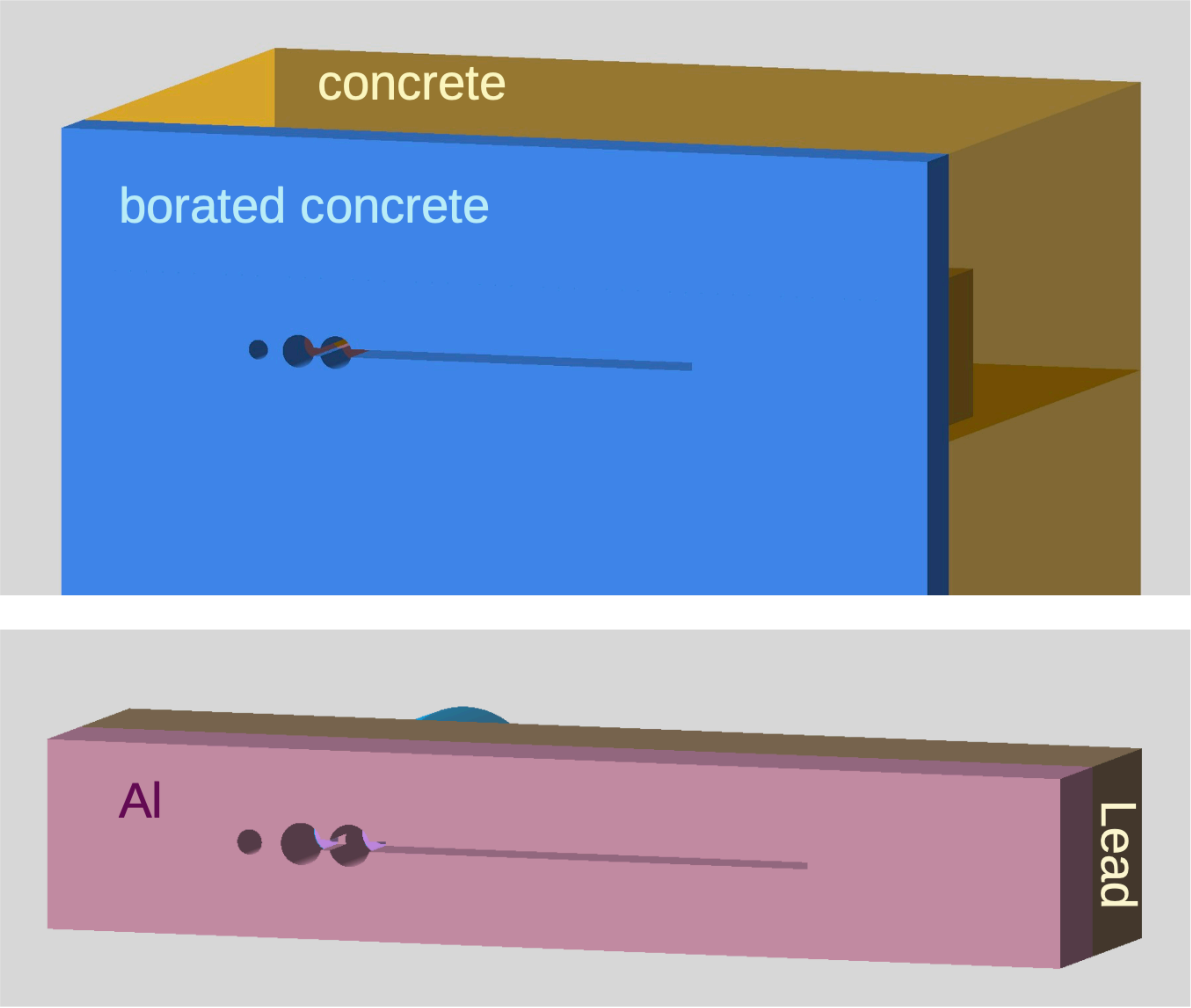


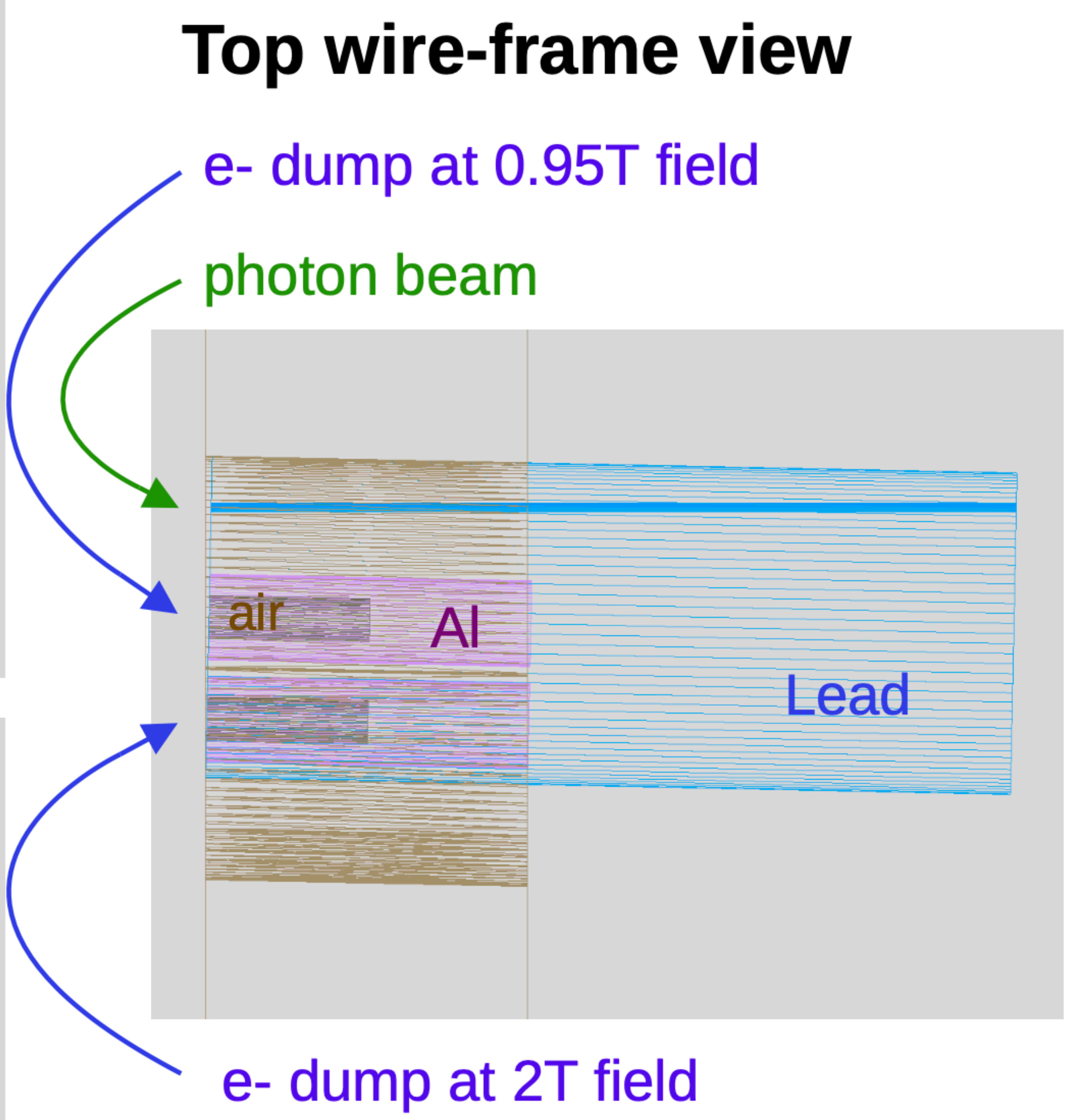
Split learning

- Data splitting



Beam dump and shielding

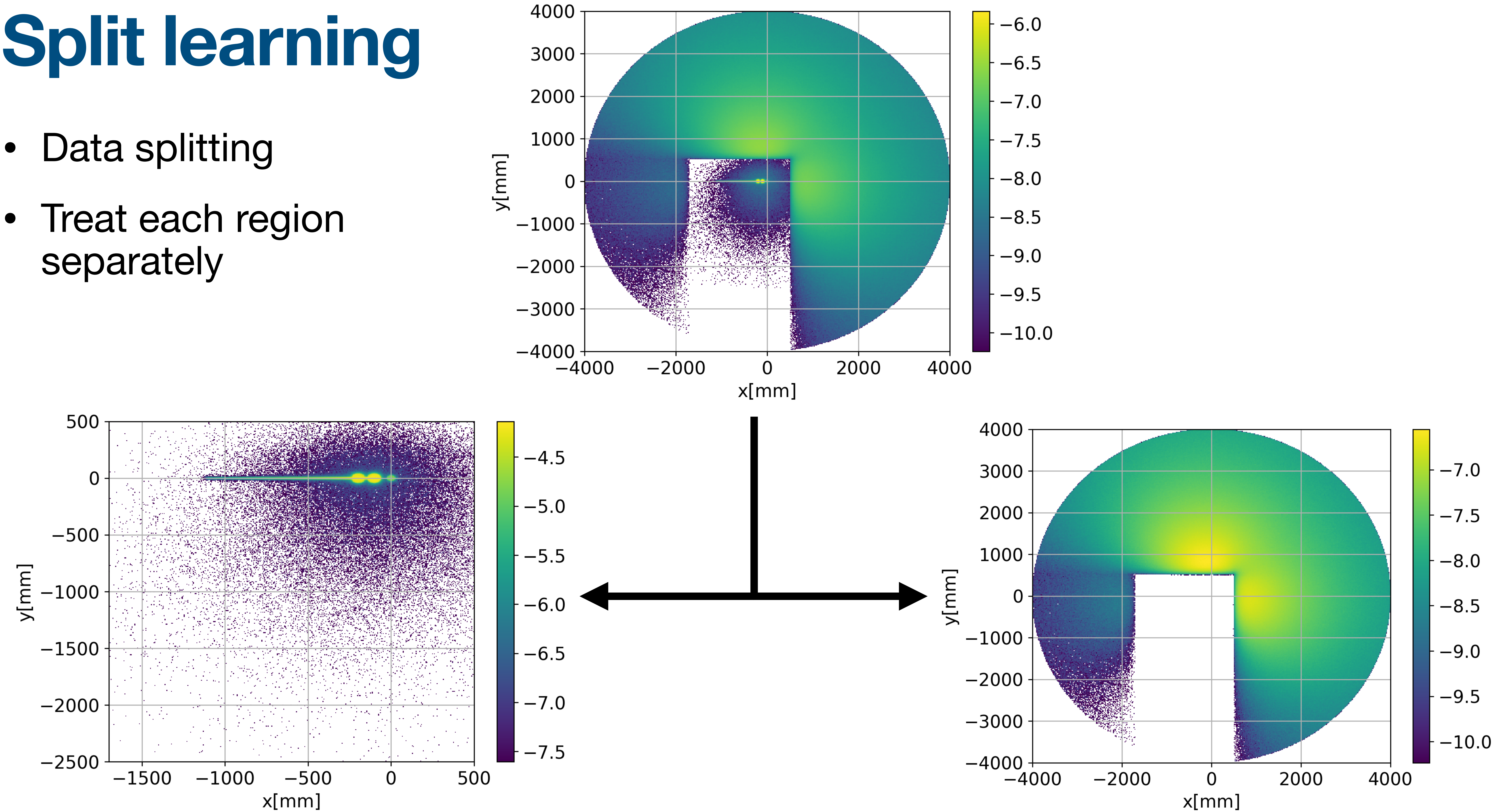




Made by Oleksander Borysov

Split learning

- Data splitting
- Treat each region separately

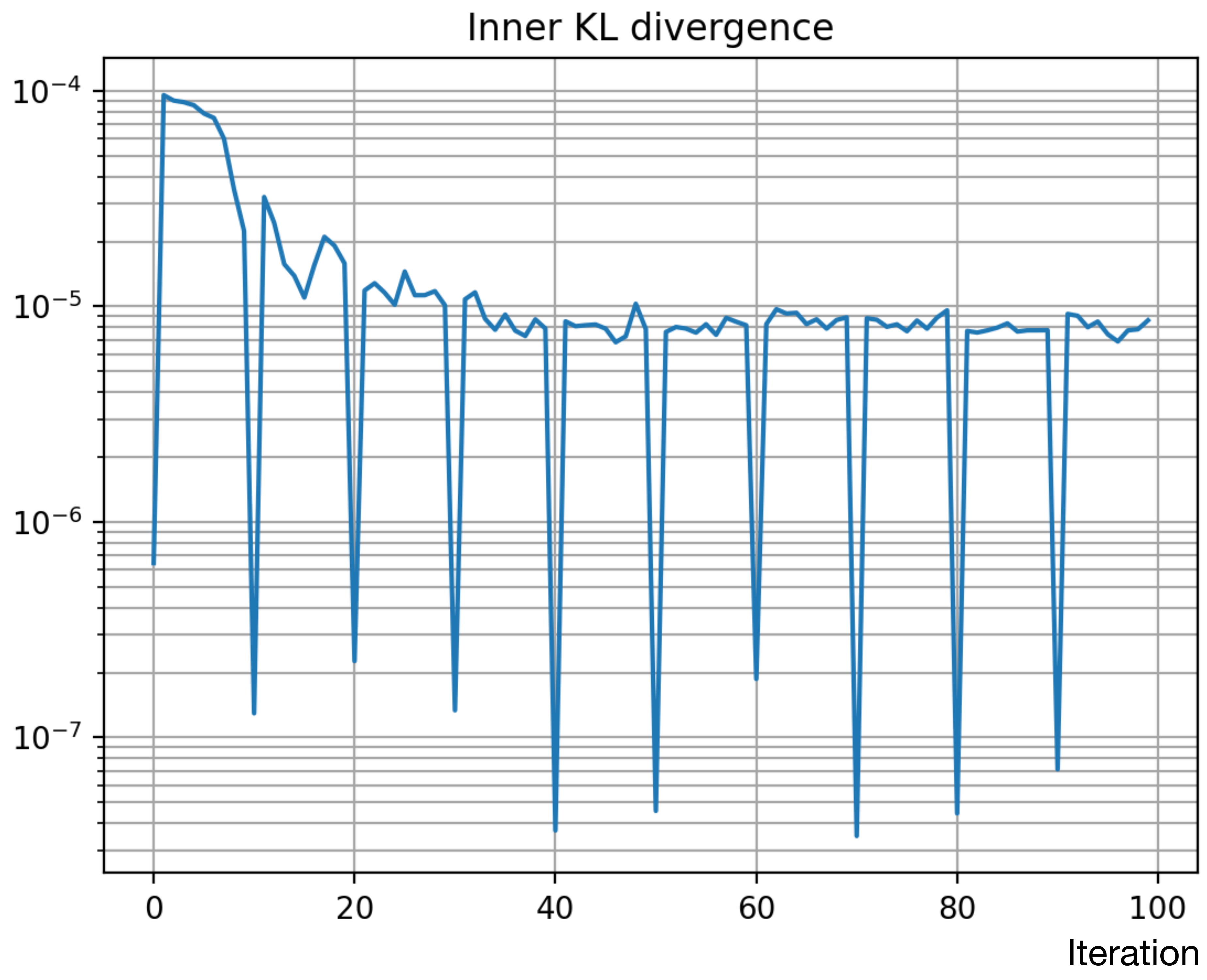


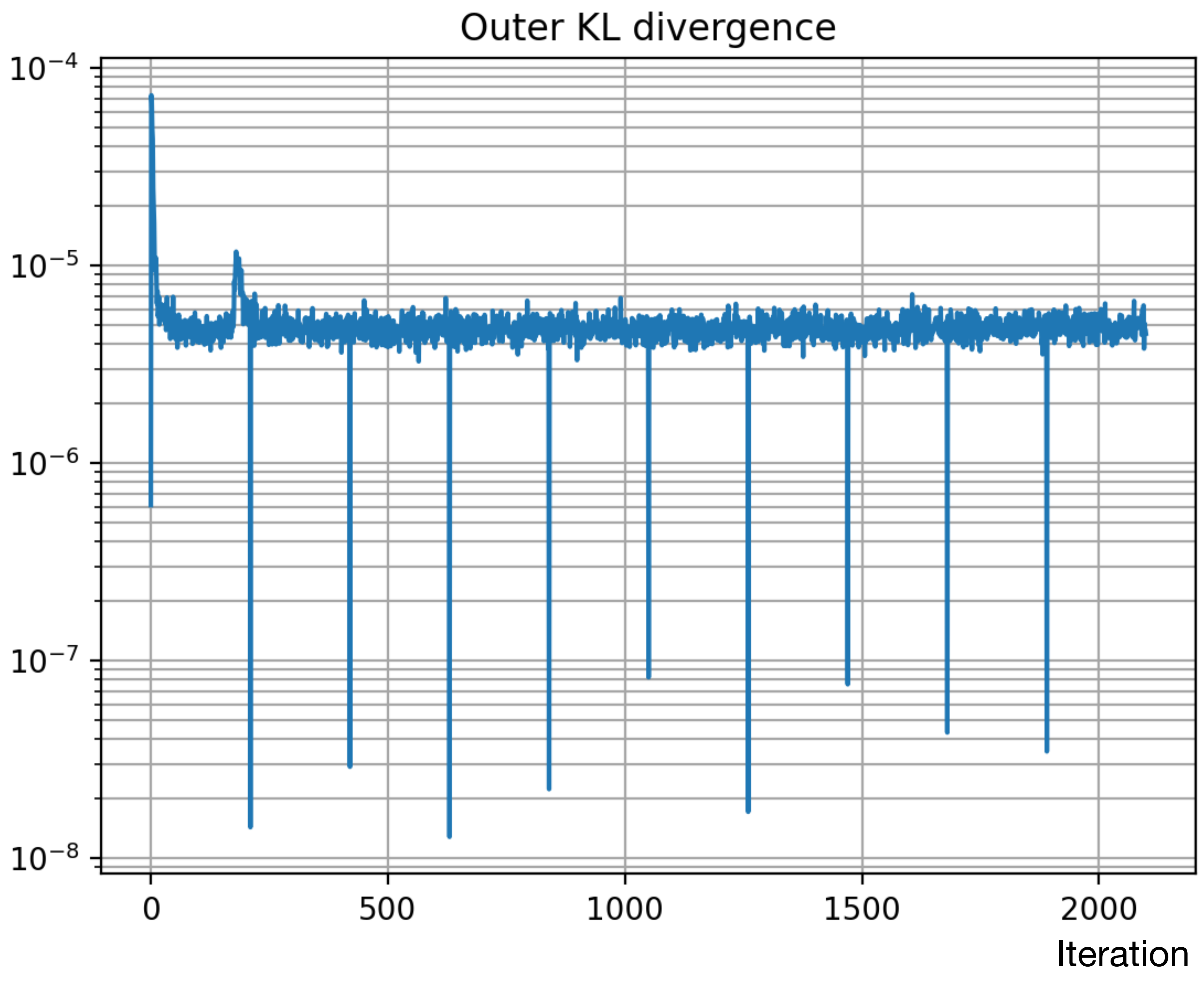
Hyperparameters and testing

- Larger batch sizes
- Different coordinate systems
- Training redundancy

Train with: $\begin{bmatrix} r_x = \sqrt{x^2 + y^2} \\ x \\ y \end{bmatrix}$, Generate only: $\begin{bmatrix} x \\ y \end{bmatrix}$

Learning process

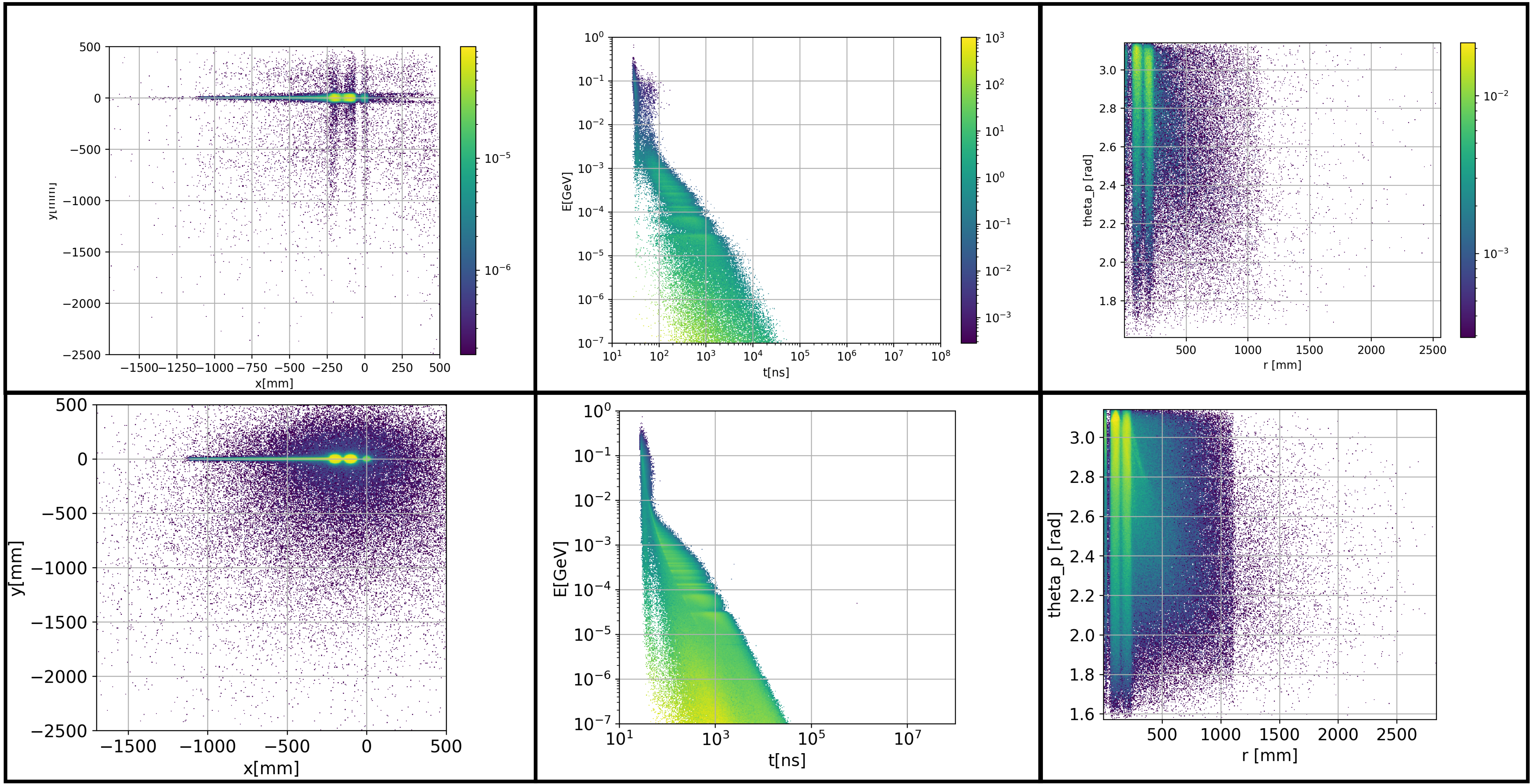




“Inner” points

Generated Data

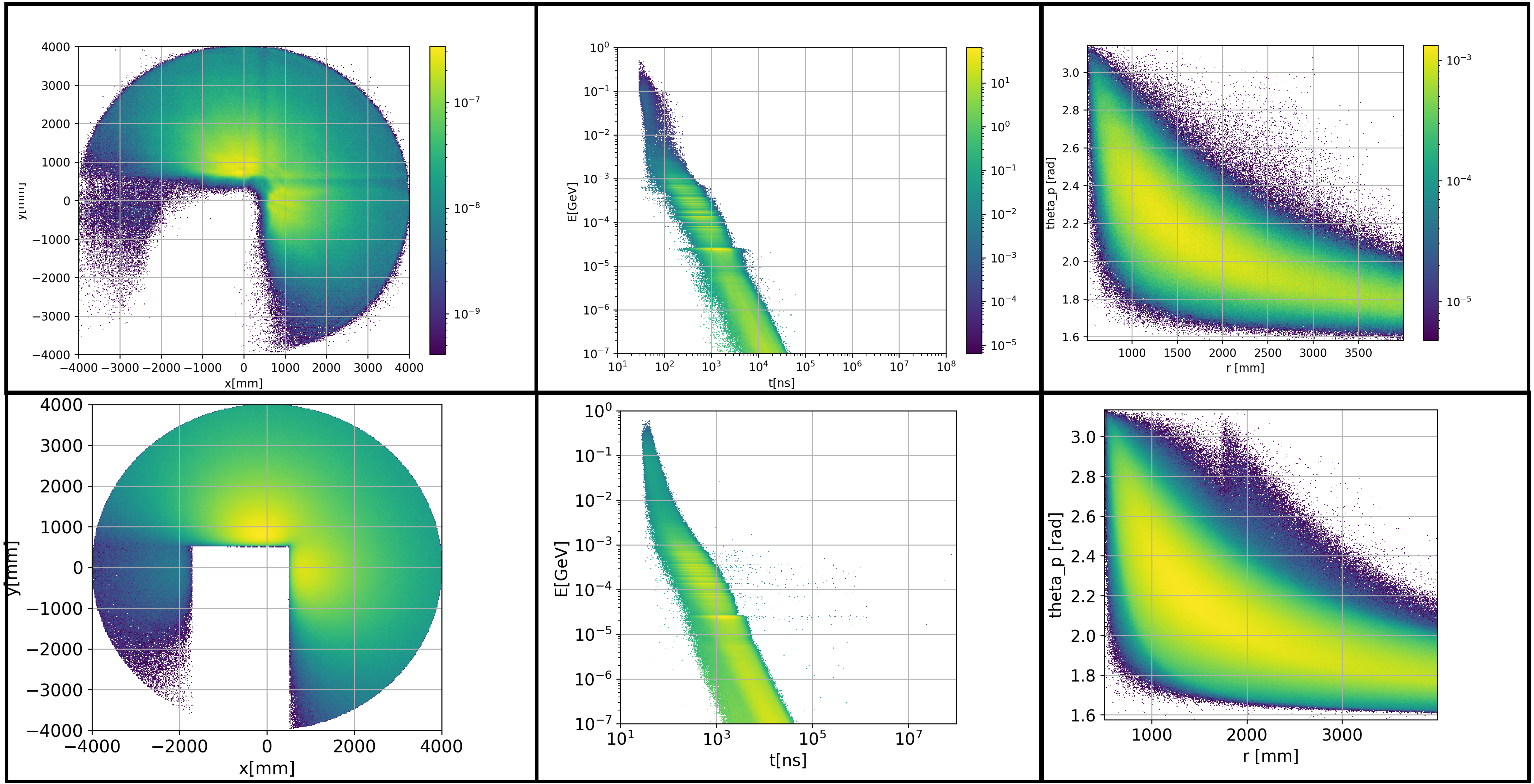
FullSim

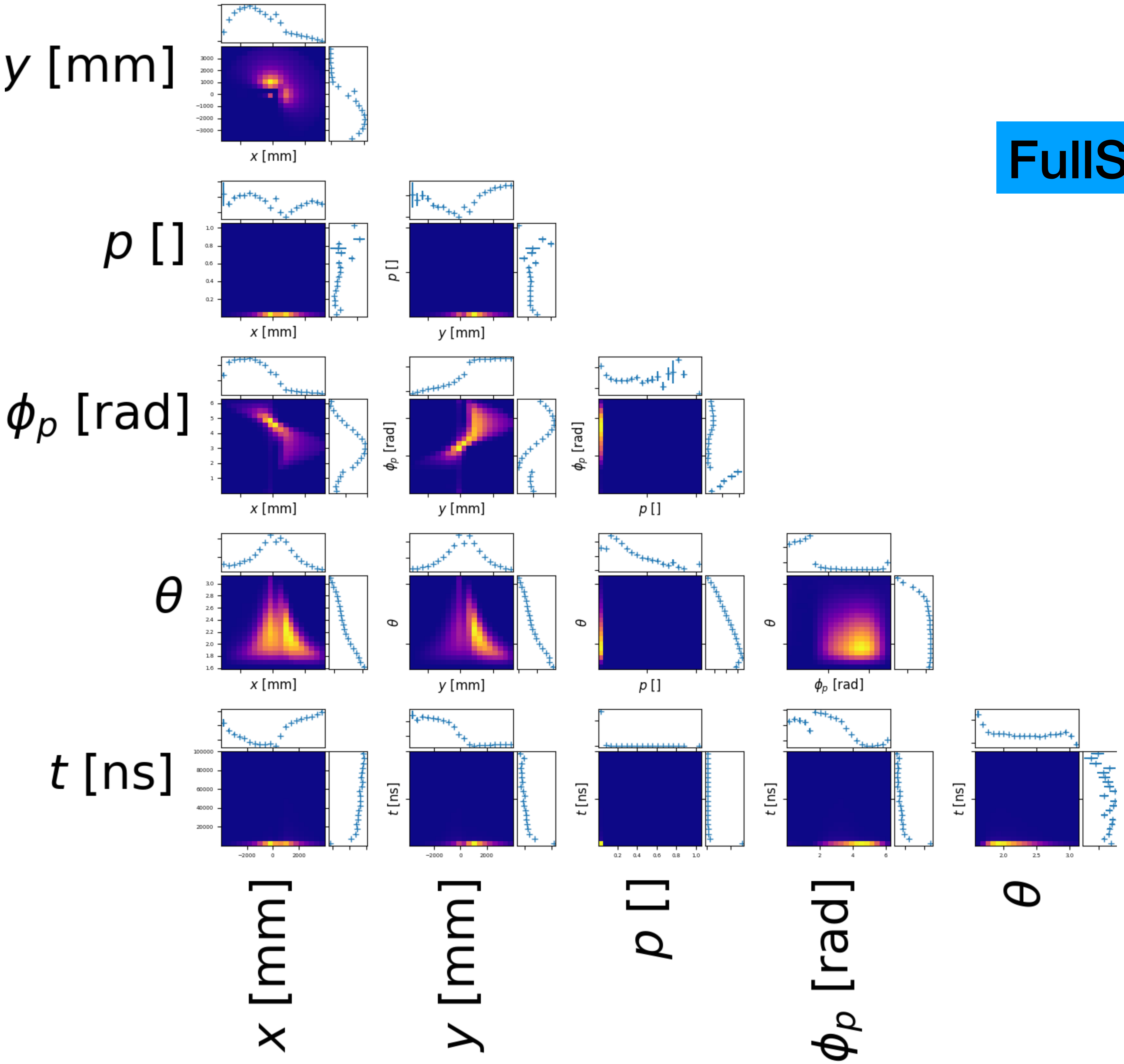


“Outer” points

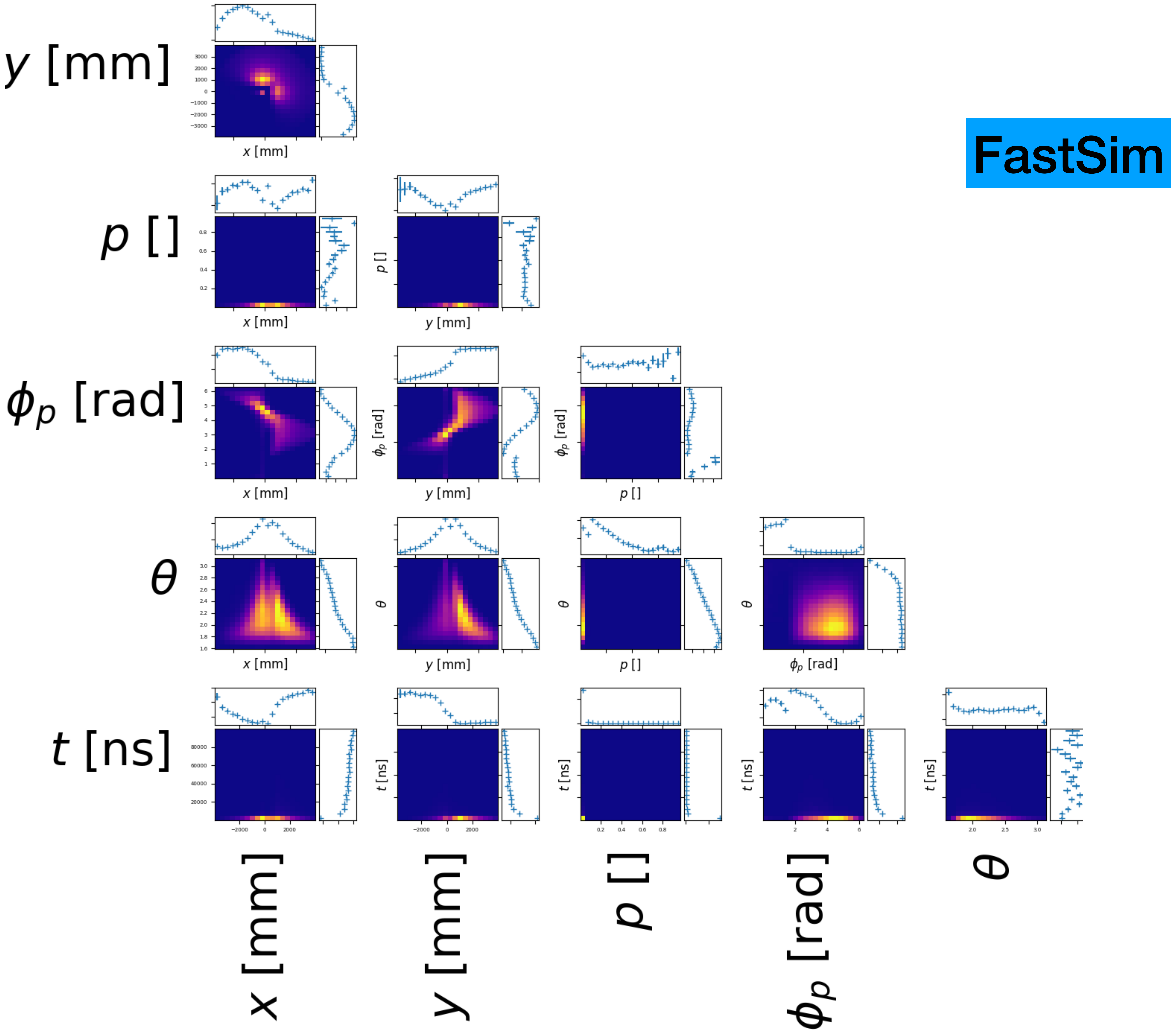
Generated Data

FullSim



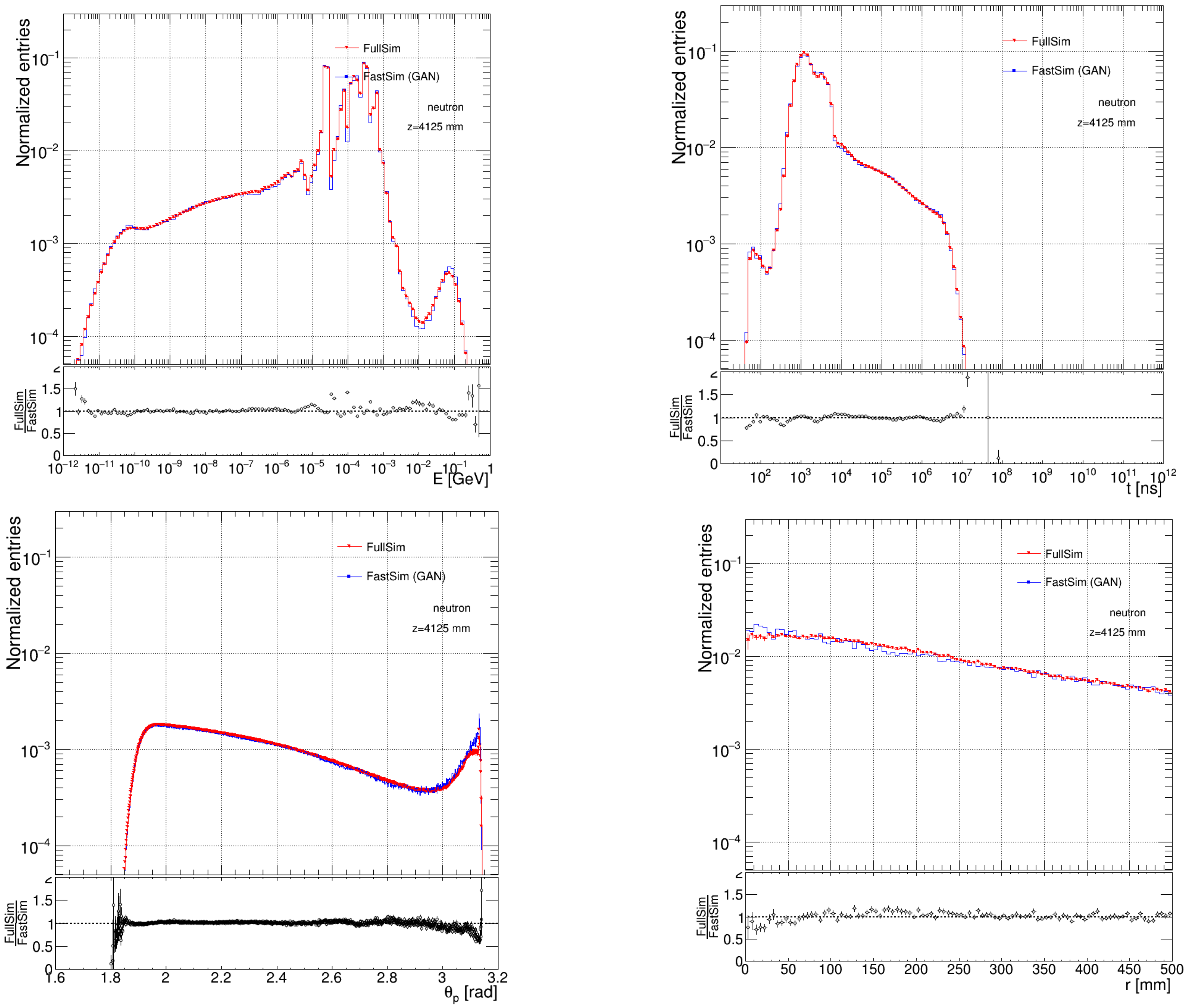


FullSim



FastSim
y [mm]
p []
ϕ_p [rad]
θ
t [ns]
x [mm]
y [mm]
p []
ϕ_p [rad]
θ

At tracker surface



Further steps

- Quantifying how well the distributions match
- Applying the same method to different particle types
- Extracting actual background from generated events

Thank you!