# 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 (12/2)
  - What is the problem we're facing
  - FastSim approach - using a generative neural network
- Issues with FastSim methods and how they were addressed
- Results
- Further steps

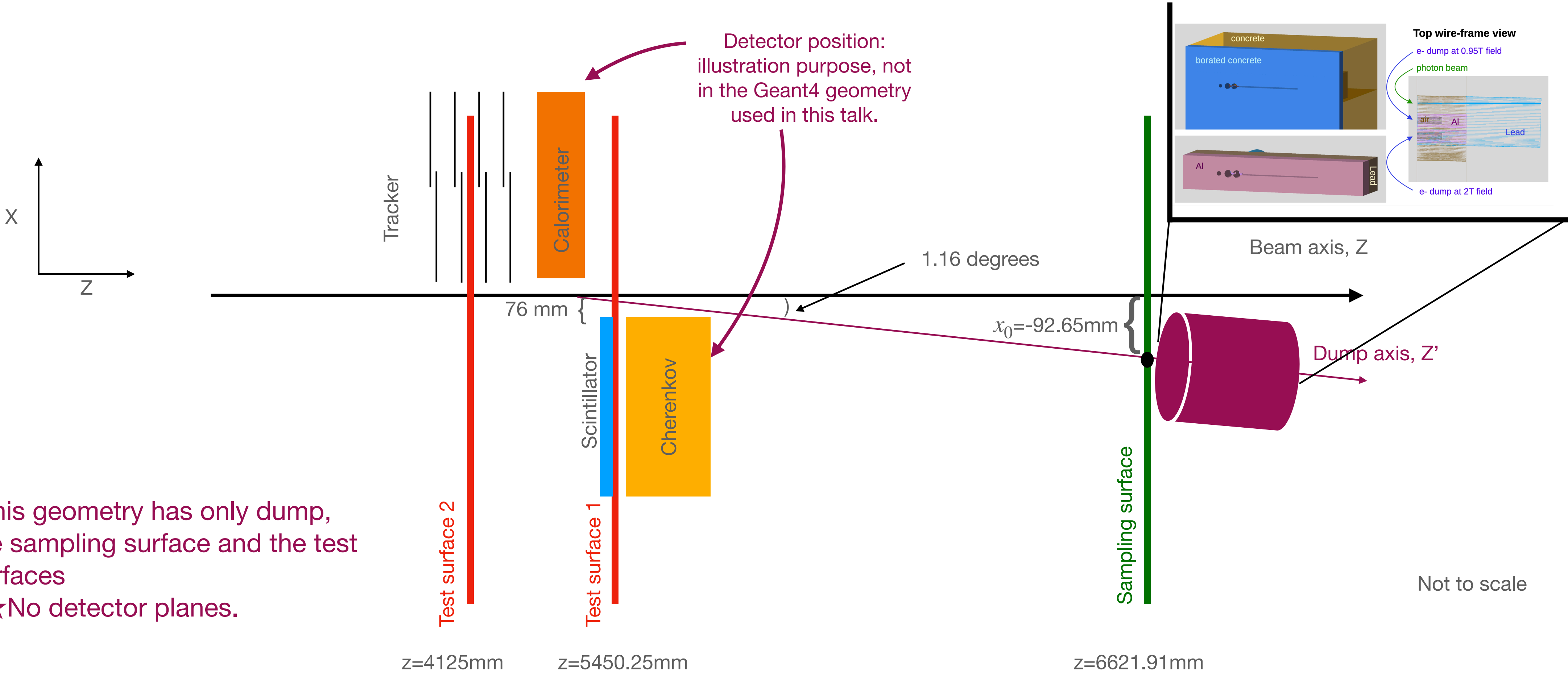
# Recap

- Ideally - use Geant4 to understand the background
- In practice:
  - We have $1.5$ [nC] $\sim 10^{10}\, e^-$ per BX
  - Most of them end up in the beam dump
  - Each particle may generate secondary particles
  - Results in very long computation times: $2.6$ hours for only $10^4\, e^-$ !
  - What we generated using Geant4 is $\sim 1\,\%$ of 1BX which requires 25k hours of computation time

# Approaches to background simulation

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

# Schematic diagram of the dump in the LUXE geometry




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

★No detector planes.



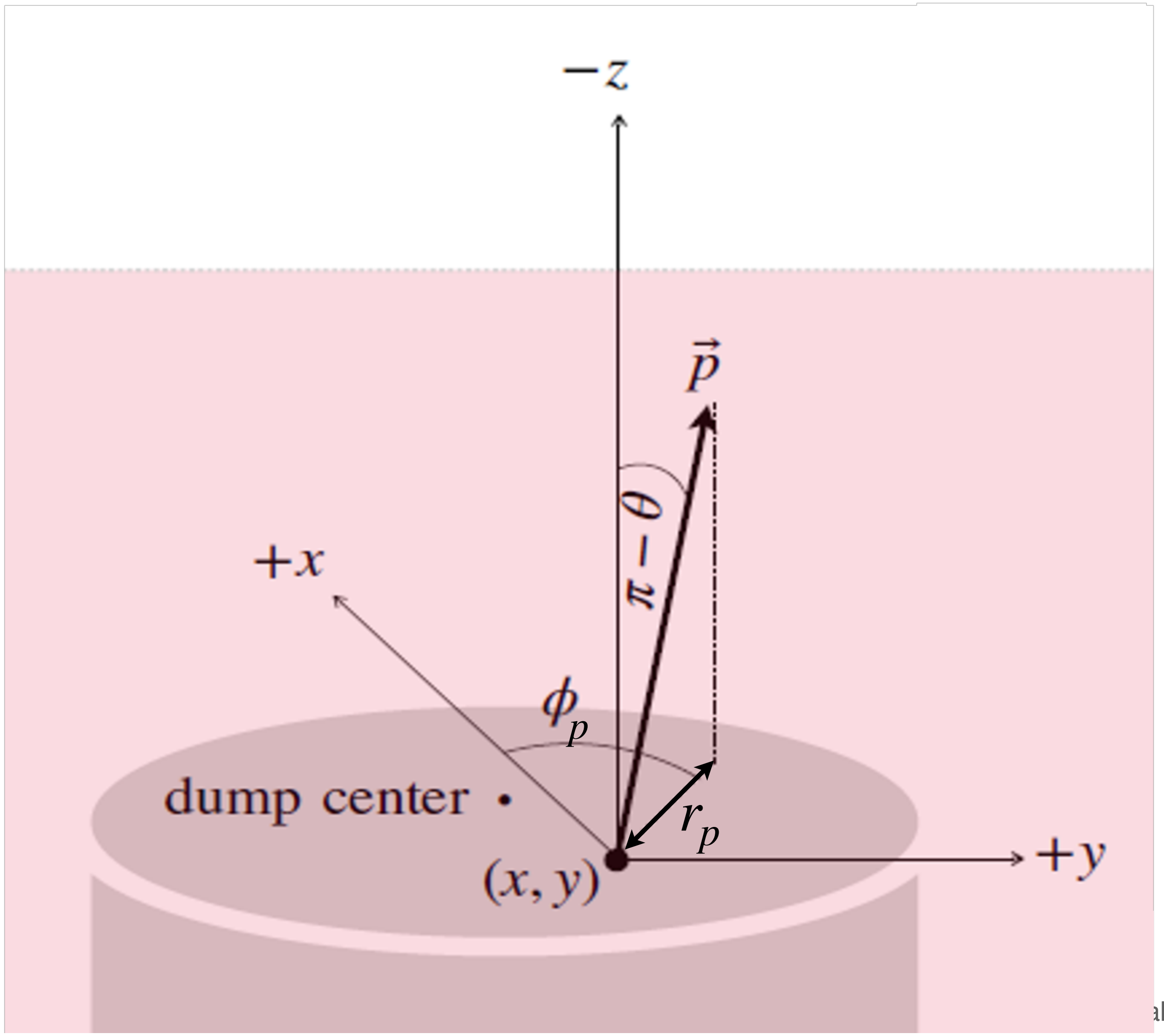
Taken from Arka's slides*

# Particle Features

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

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

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



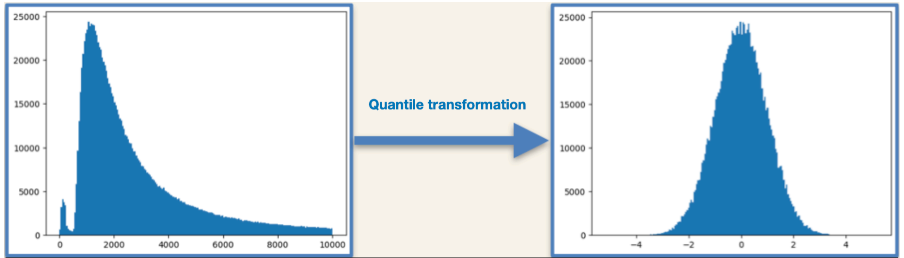

aken from Arka's slides*

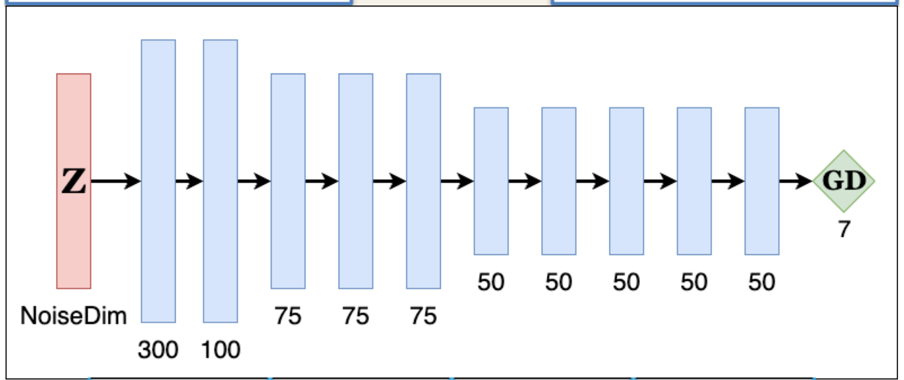
# WGAN method

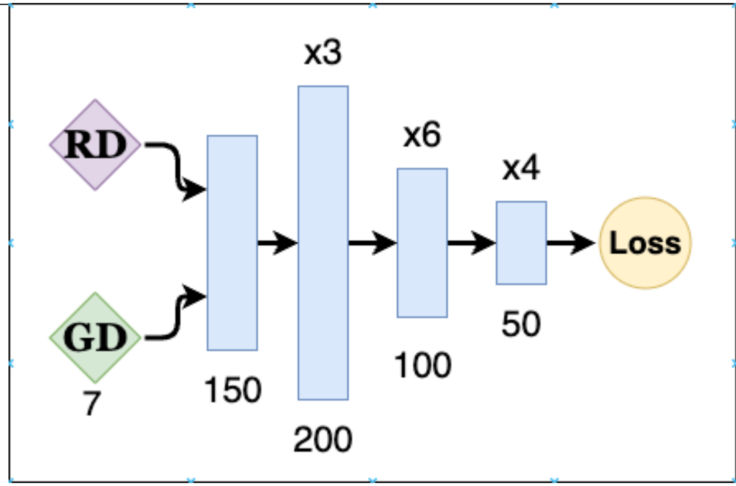
1. Apply quantile transformation to the data
2. Generate data points from random noise
3. Pass real and generated data through critic
4. Determine loss via Wasserstein metric
5. Monitor Progress with KL divergence
6. Apply inverse quantile transformation

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

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



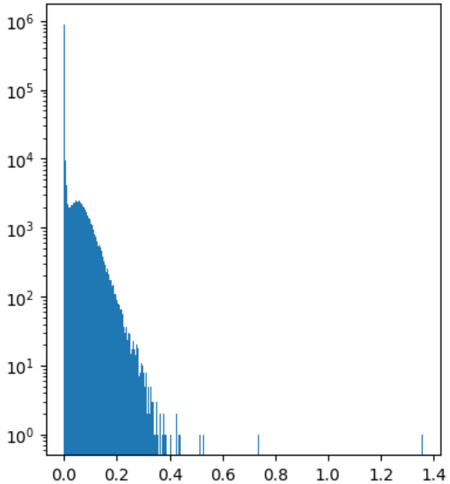



# Quantile Transformation

- Consider energy for example

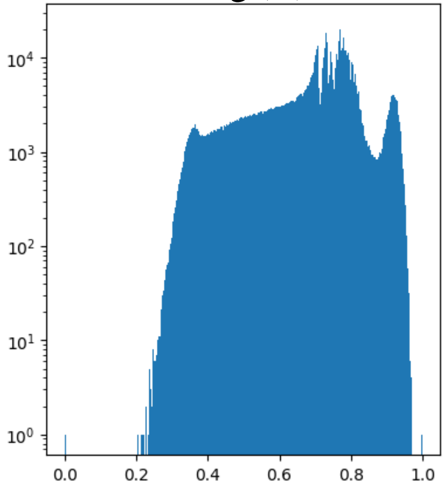
Raw Data

$E$



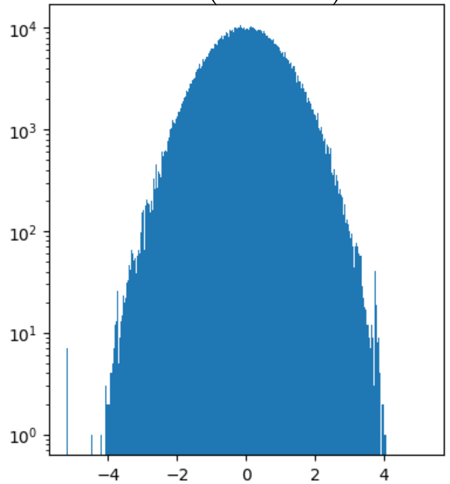

Some function applied
to make quantiles “less sharp”

$\log(E)$

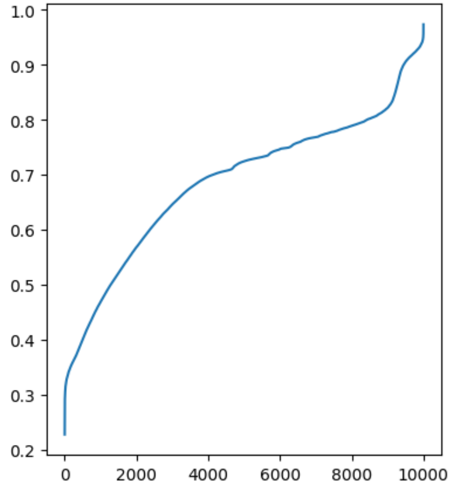



Post transformation data
Ideally $\sim N(0,1)$

$QT\left(\log(E)\right)$

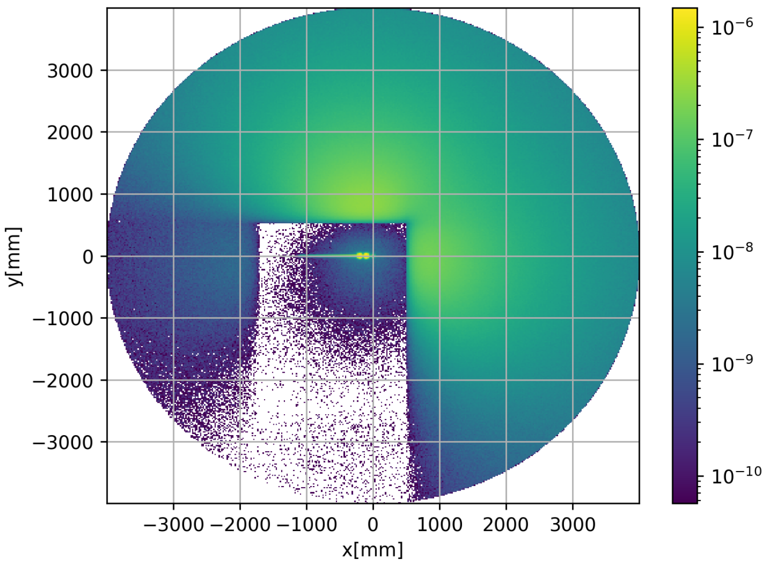


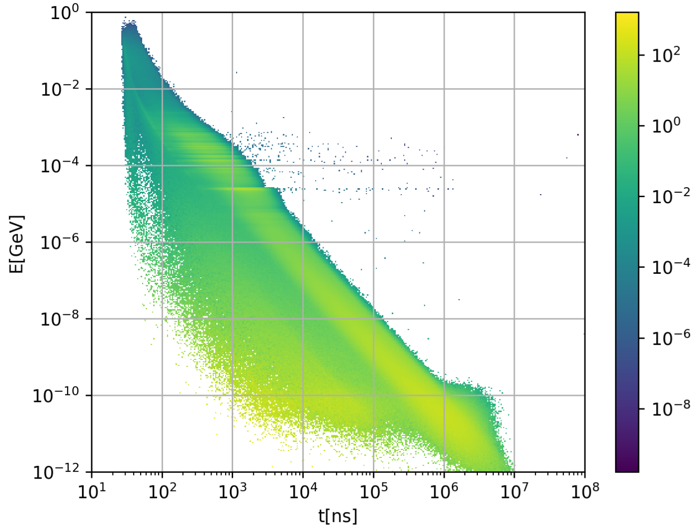

The CDF used
for transformation

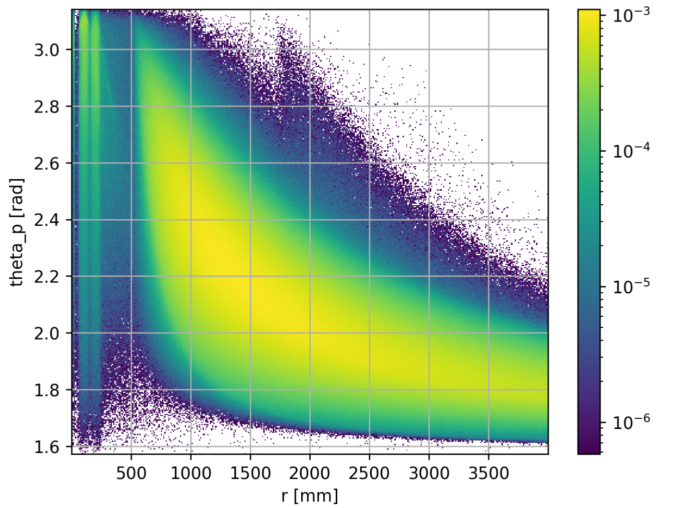


# Data splitting

- GAN struggles to learn sharp feature -> Split the data
- We'll discuss the distribution for outgoing neutrons
- The case is similar for $e^{-}, \gamma$ etc.

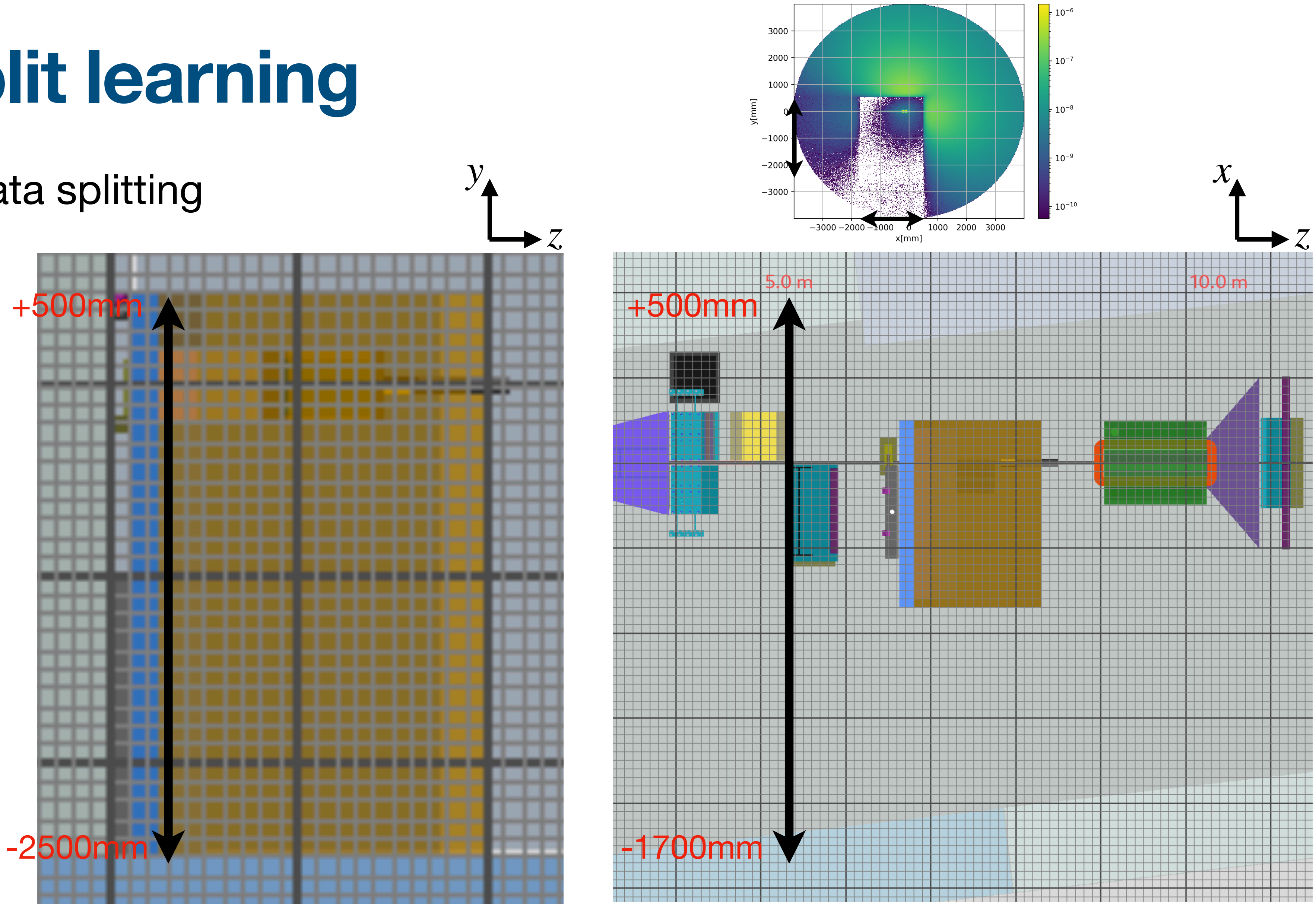


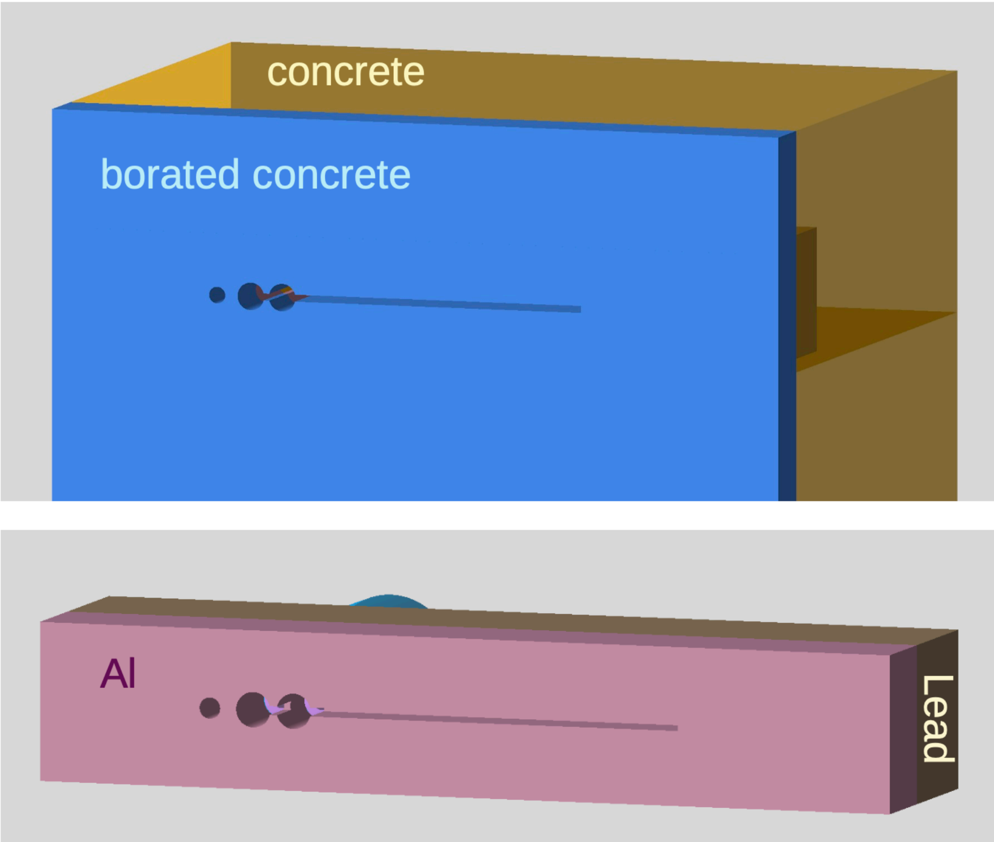



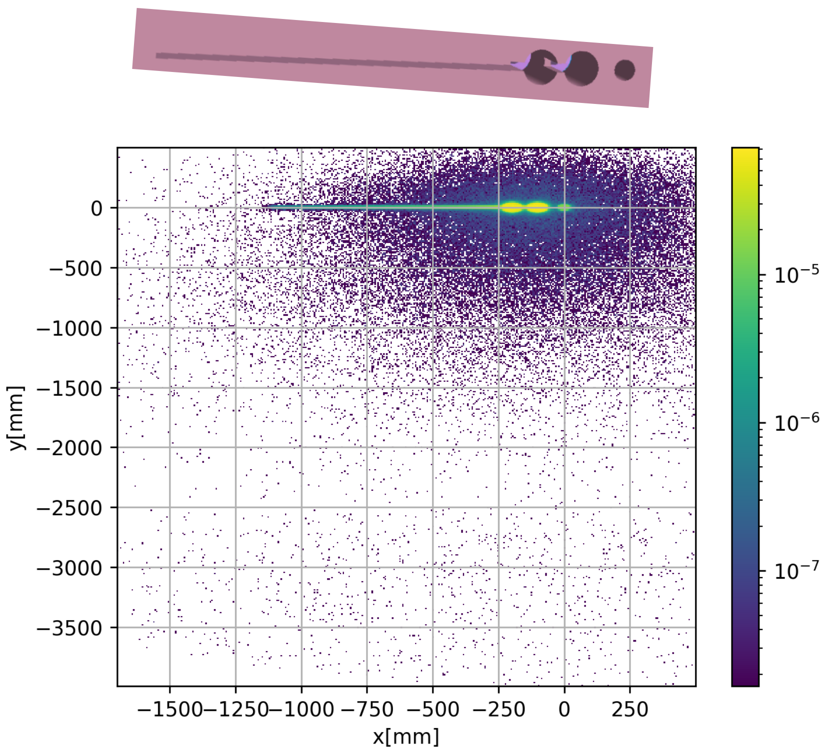
# Split learning

- Data splitting



# Beam dump and shielding

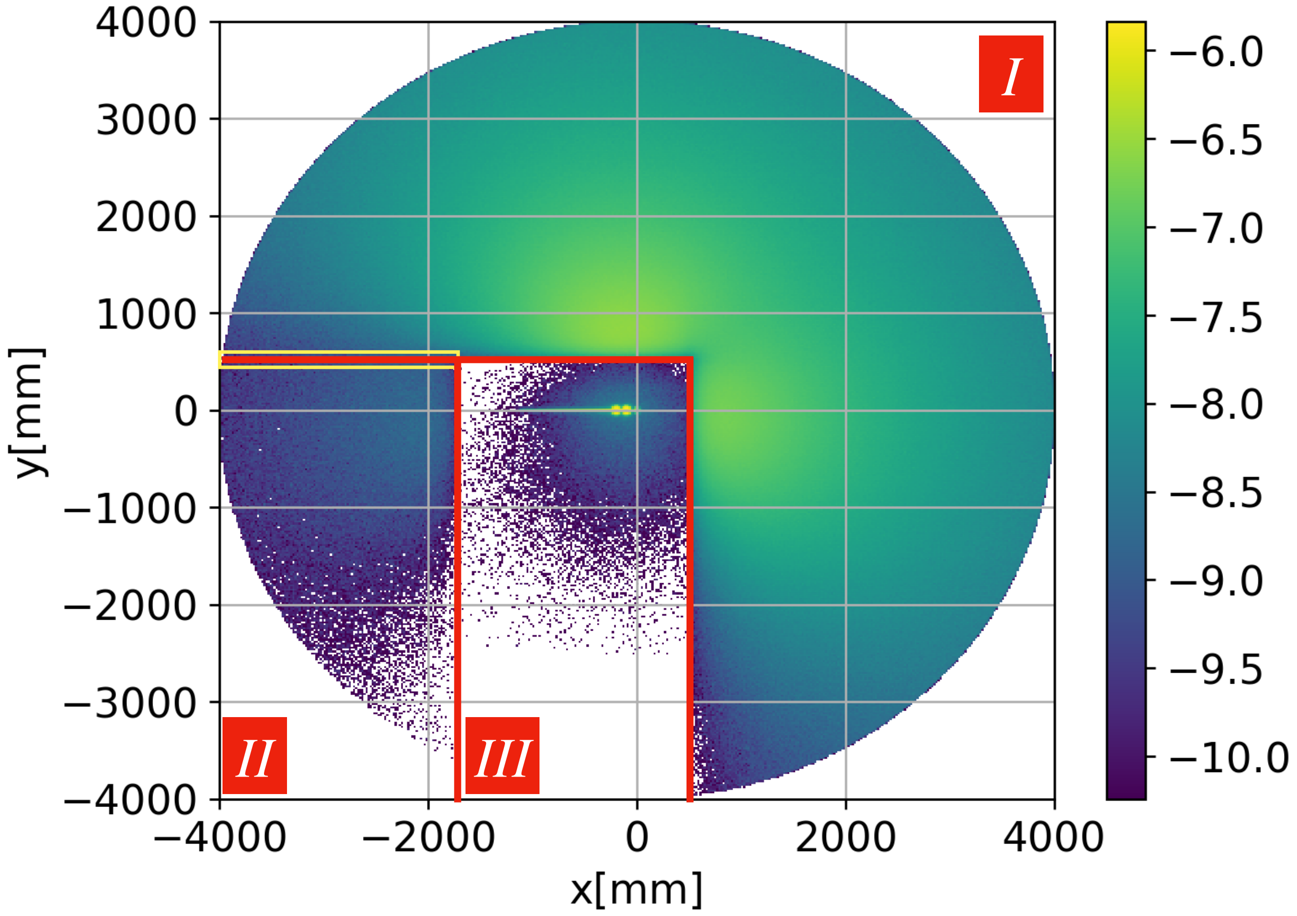


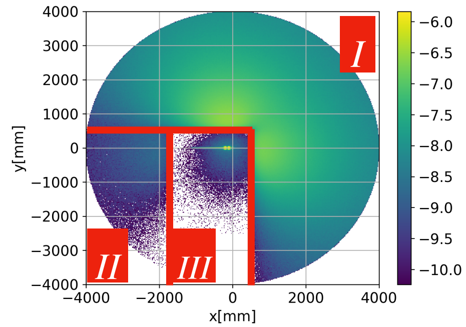

Made by Oleksander Borysov

# Split learning

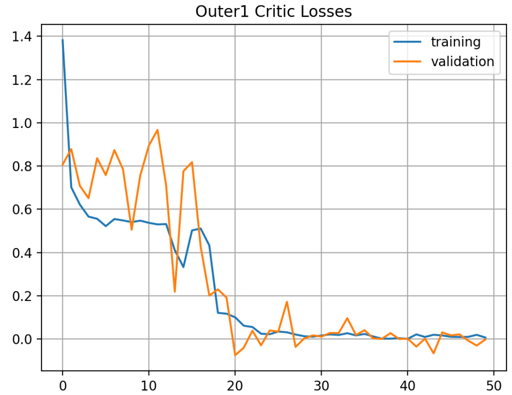
- Treat each region separately
- Overlap in training data
- During training we allow networks to produce events anywhere - “Leakage”
- No “Leakage” in production

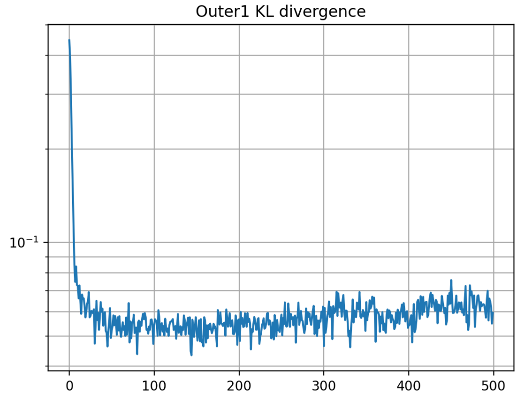


# Learning process

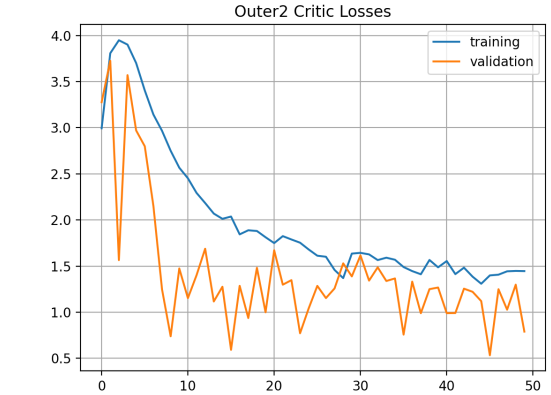


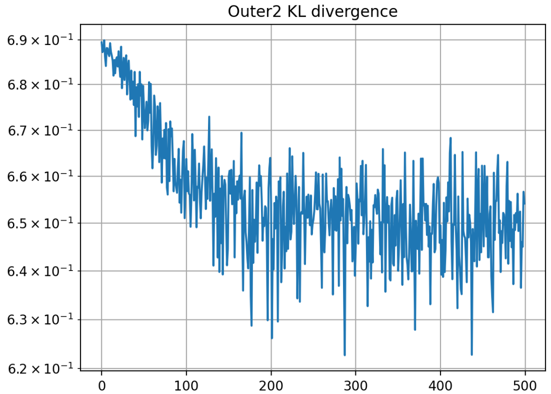

## I



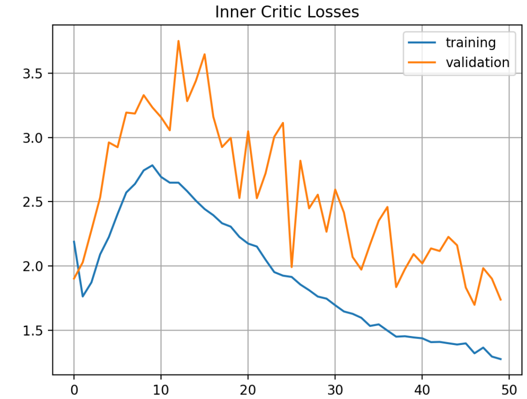



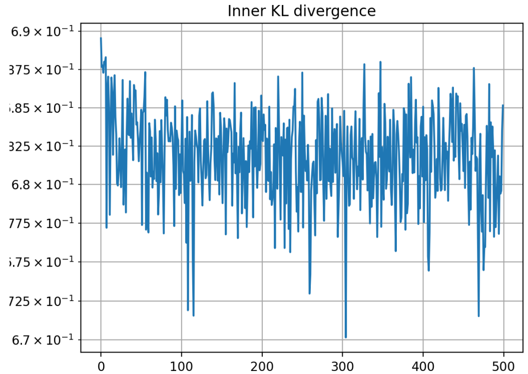

## II



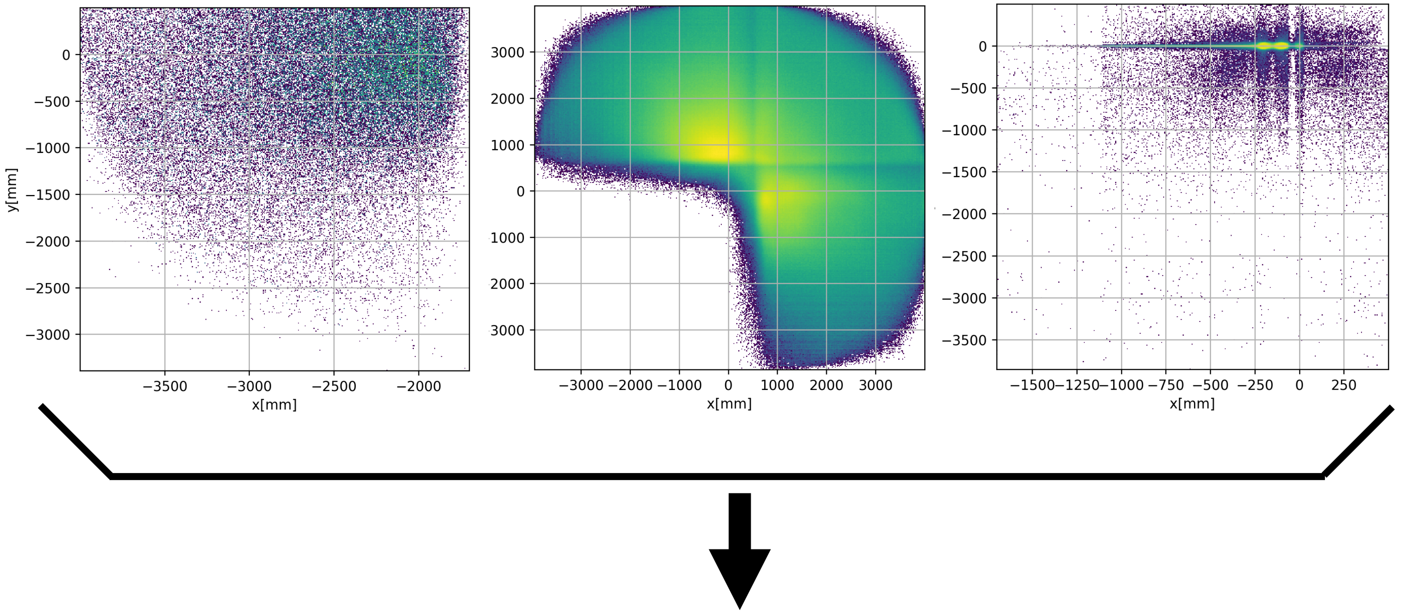

## III

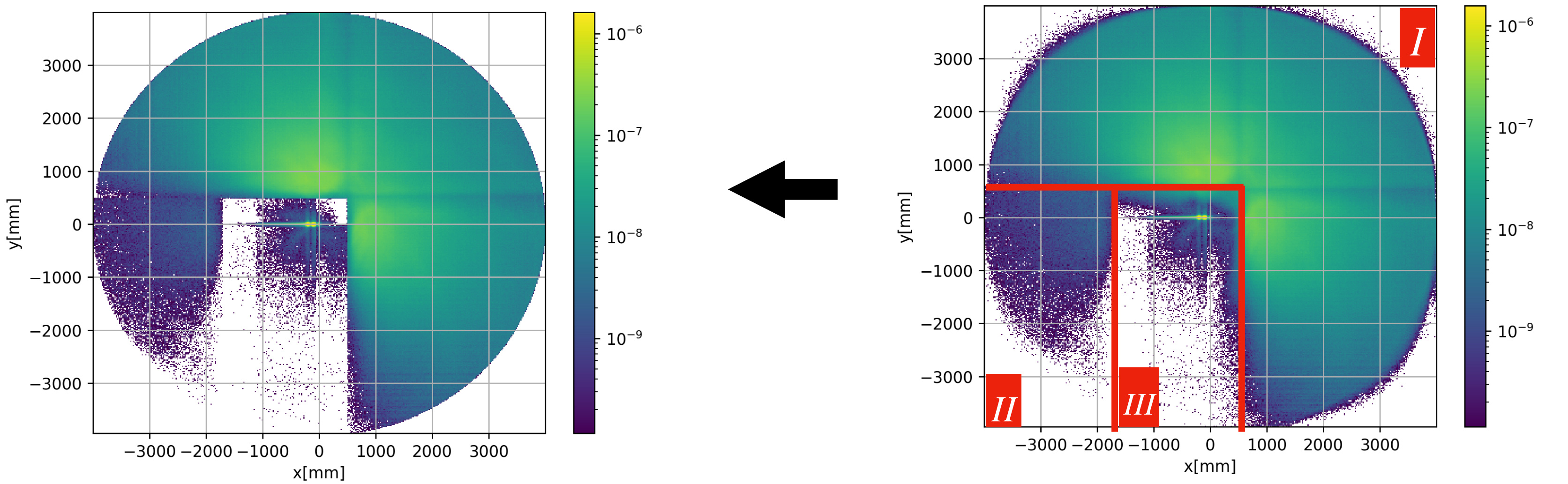


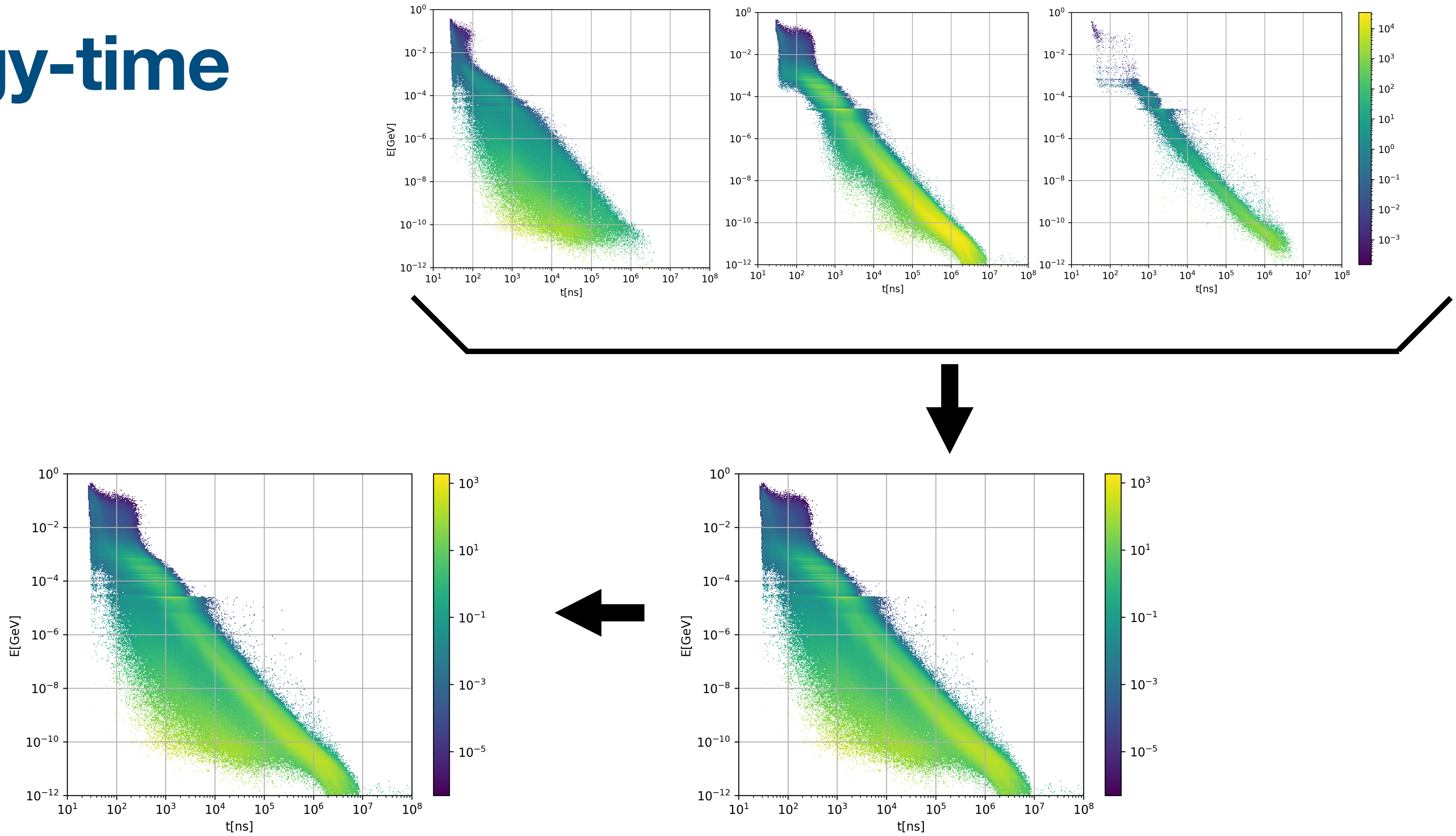
# Data Production

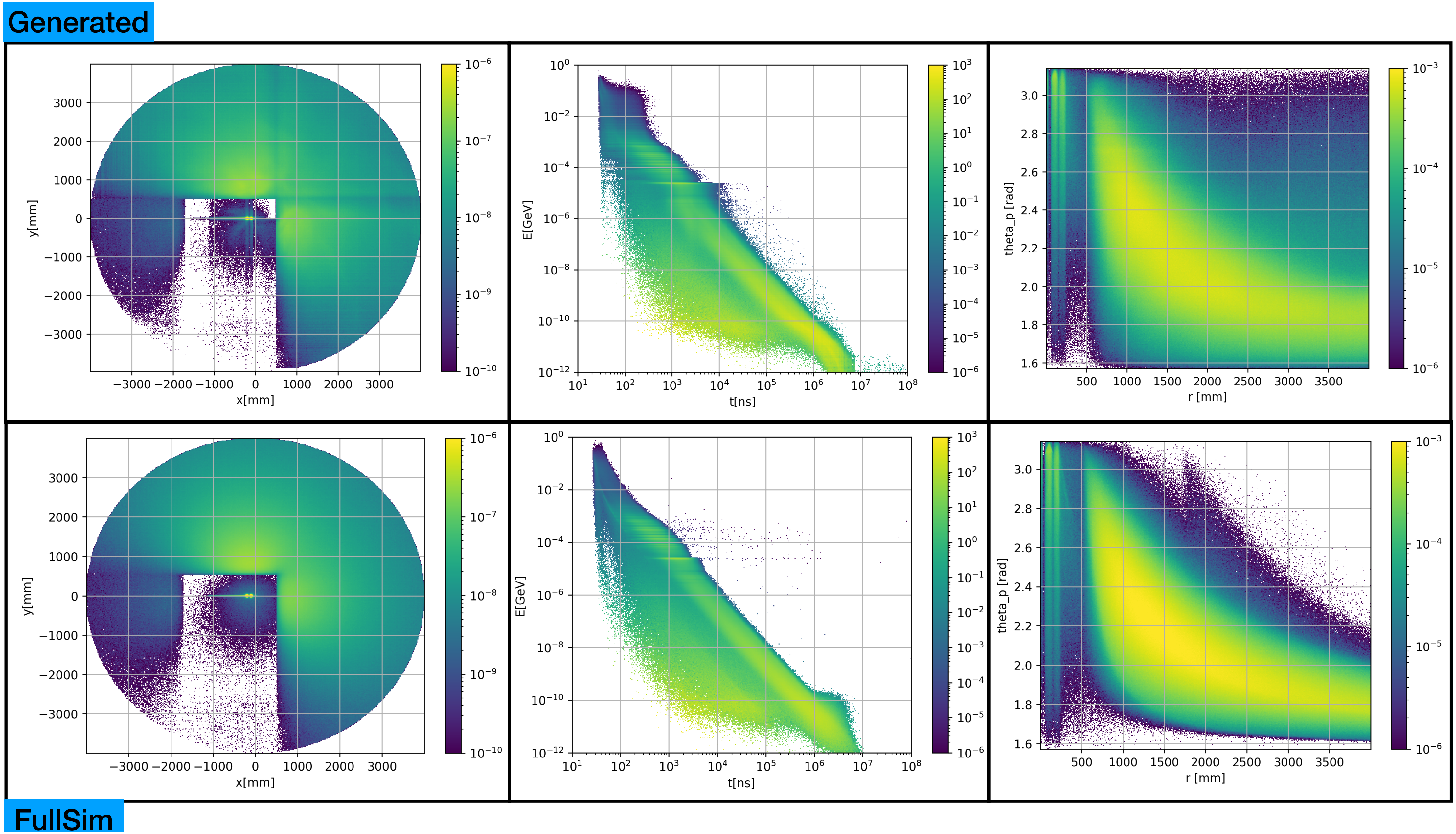
- Concatenate the 3 regions
- Clean each region from points generated by other networks
- Enforce $r \leq 4000$



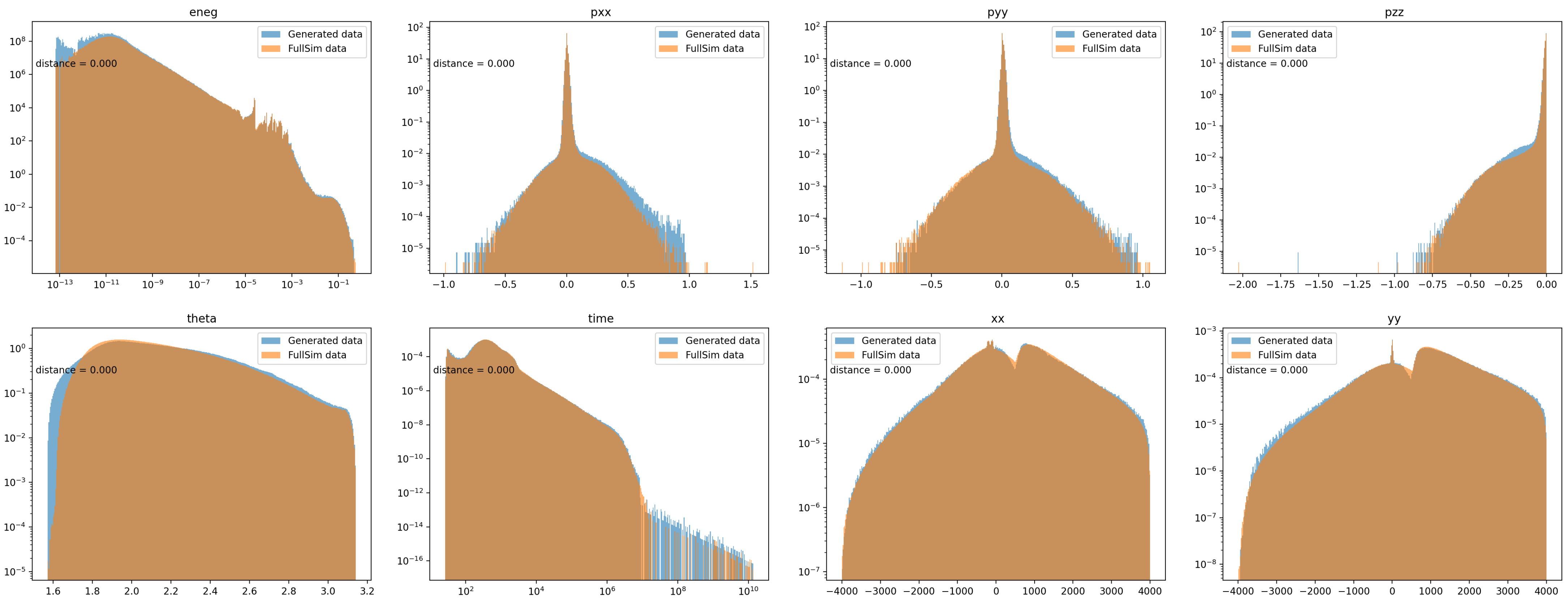
# Energy-time





Generated
FullSim
y[mm]
x[mm]
E[GeV]
t[ns]
theta_p [rad]
r [mm]

# 1D distributions



# Computation times

- Generating $50M$ neutrons takes $\sim 0.5h$
- Advancing them via Geant4 will take $\sim 1h$
- Compared to Geant4 generation times we will save up to 3 orders of magnitude in computation time

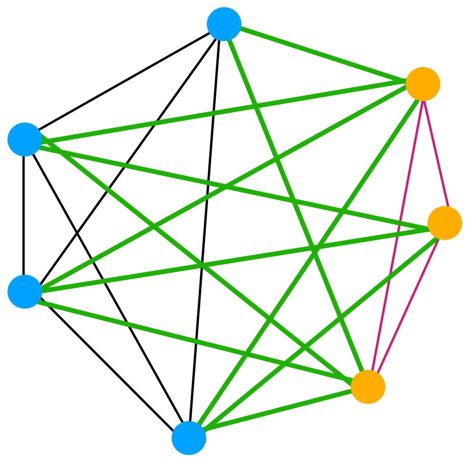
# Quantifying similarity to data

- Initial hyper-parameter tuning and testing was done until we saw a good match qualitatively, now we require a quantitative approach
- We have two 6-dimensional datasets, one generated by our WGAN: $X \sim H_X$, and another generated by Geant4: $Y \sim H_Y$
- Null hypothesis $H_X \neq H_Y$, we wish to see if $H_X = H_Y$
- Batched Energy Distance

# Batched Energy Distance

1. Start with generated dataset X, “real” (Geant4) dataset Y
2. Split X and Y into 2 sets of batches
3. Make a 3rd set by permuting Y batches
4. Define Energy Distance

$$A := \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}\underline{\|x_i - y_j\|}, B := \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\underline{\|x_i - x_j\|}, C := \frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m}\underline{\|y_i - y_j\|}$$



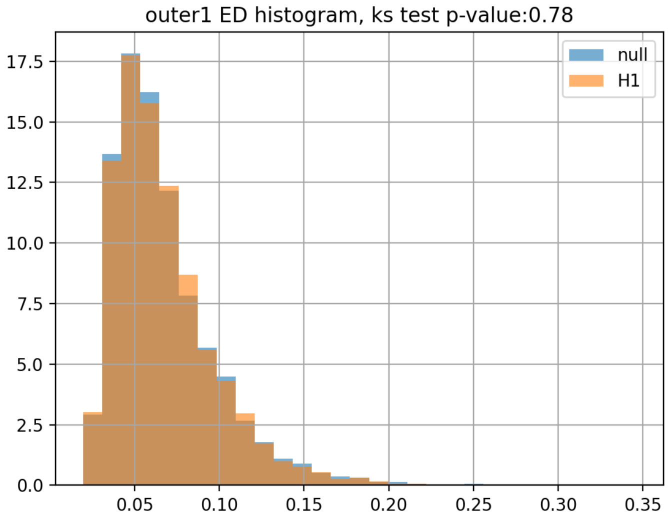
$$E_{n,m}(X, Y) := 2A - B - C$$

# Batched Energy Distance

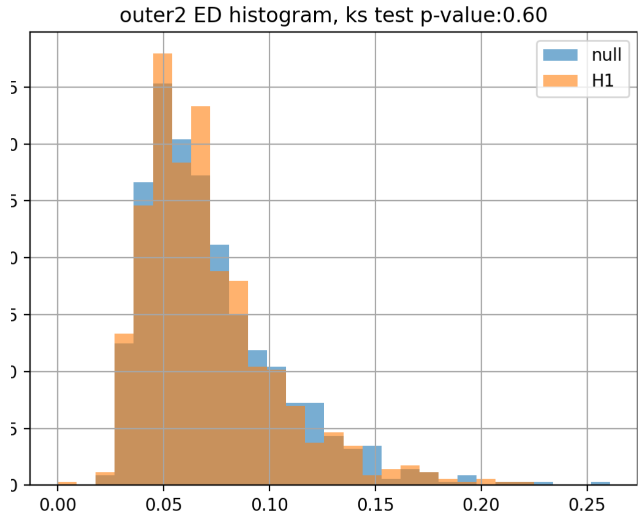
5. Make 2 histograms: $ED(X, Y), ED(Y', Y)$

6. Perform two-sample-KS test on 1D distributions -> p-value !

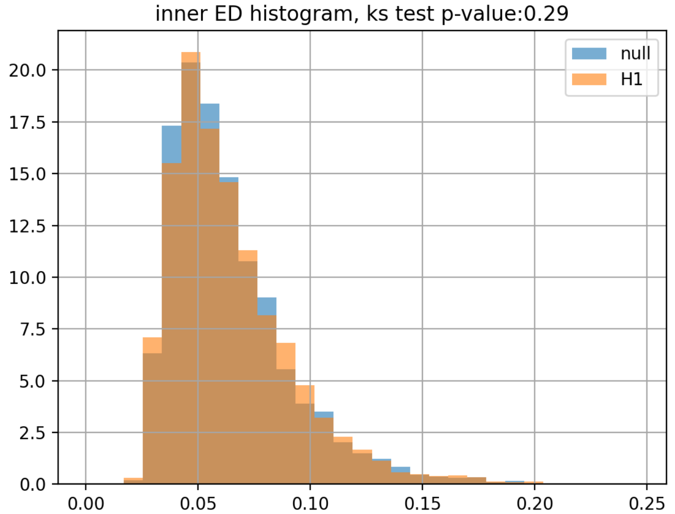
*I*




*II*




*III*



# Further steps

- Applying the same method to different particle types
- Extracting actual background from generated events

**Thank you!**