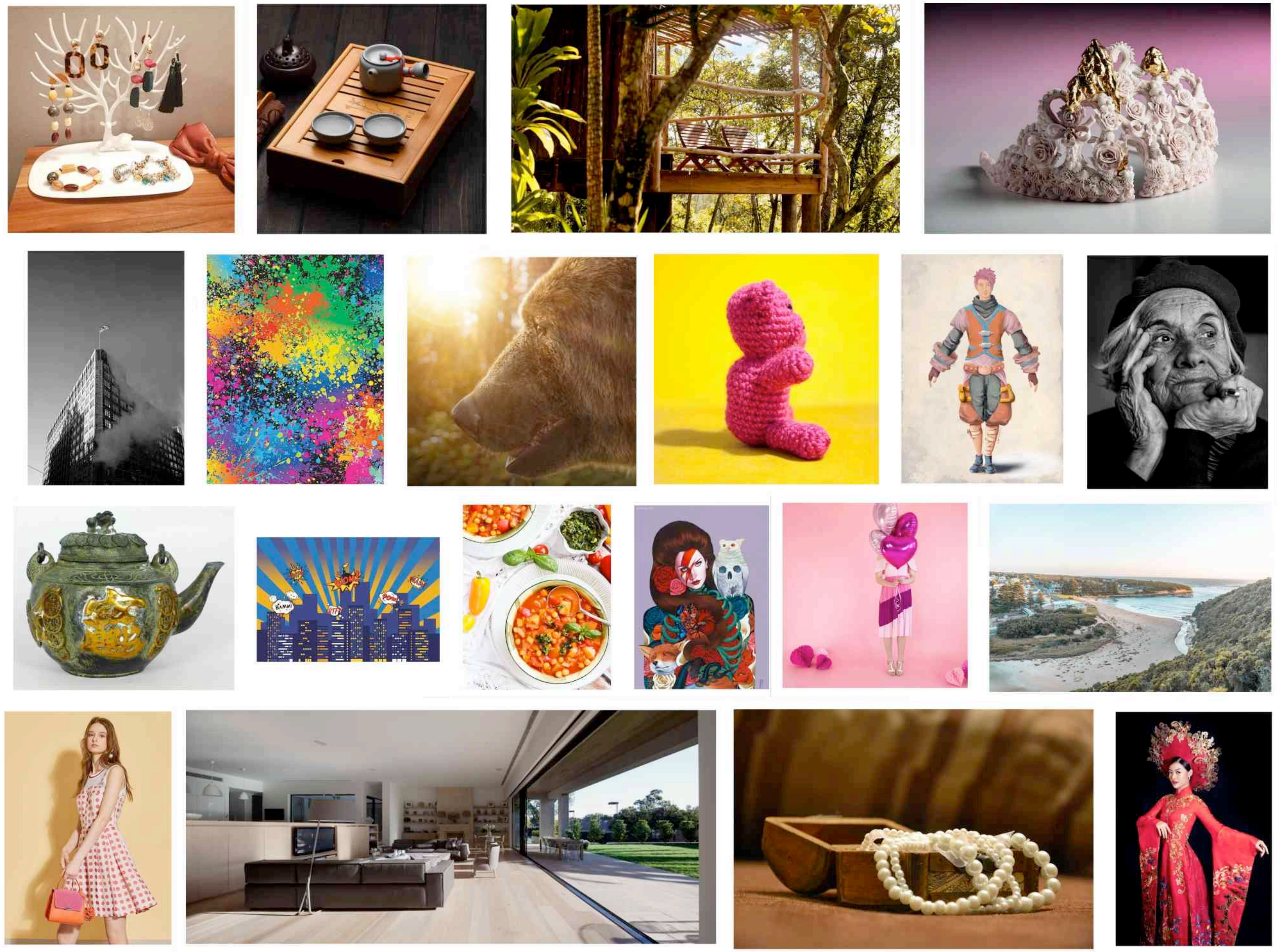
HELMHOLTZAI | ARTIFICIAL INTELLIGENCE COOPERATION UNIT

SAMPLING NEUTRONS WITH ARTIFICIAL INTELLIGENCE

José I. Robledo
Jülich Supercomputing Centre (JSC)
Jülich Centre for Neutron Science (JCNS)
Forschungszentrum Jülich (FZJ)

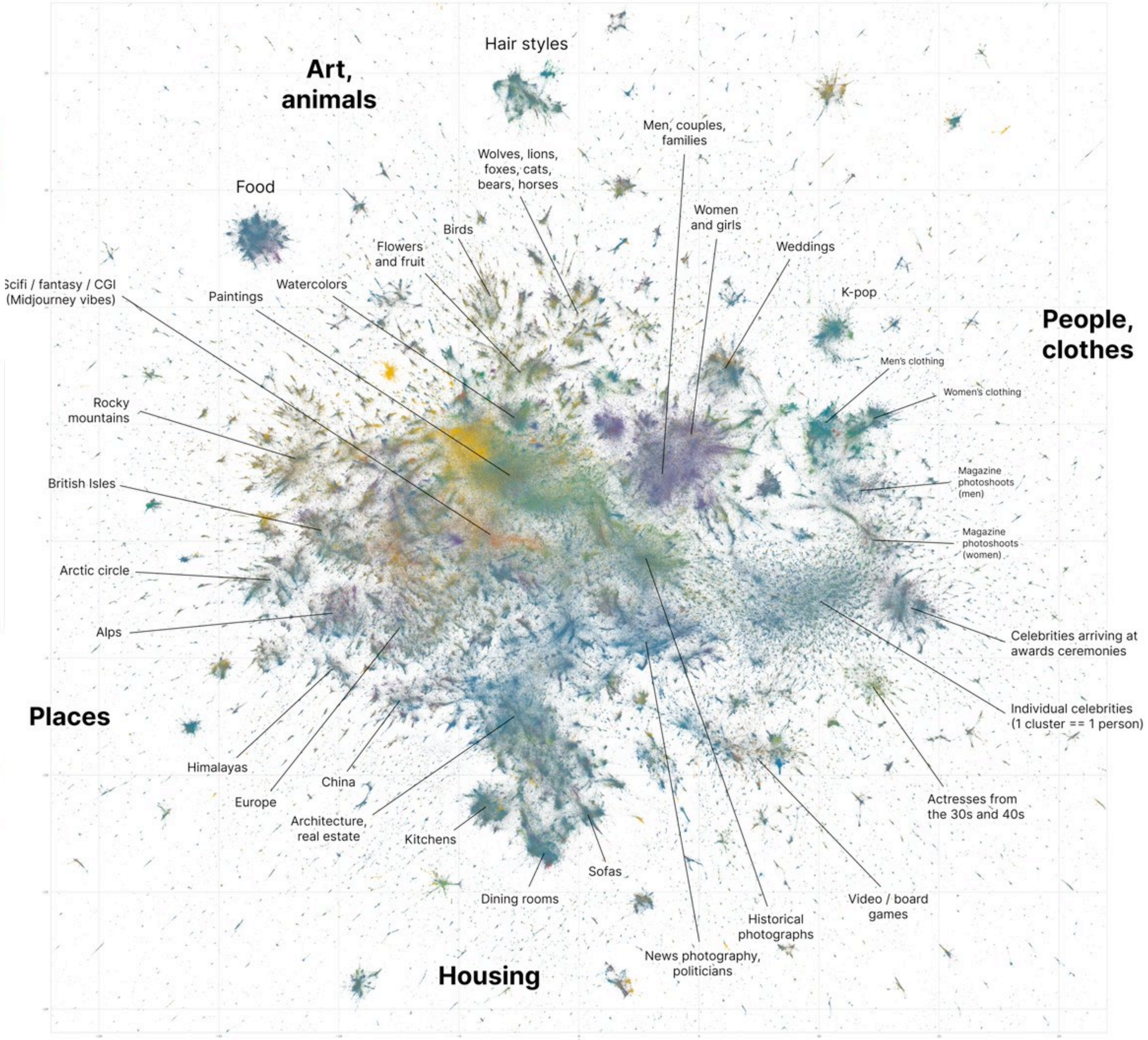
JÜLICH FORSCHUNGSZENTRUM

INTRODUCTION



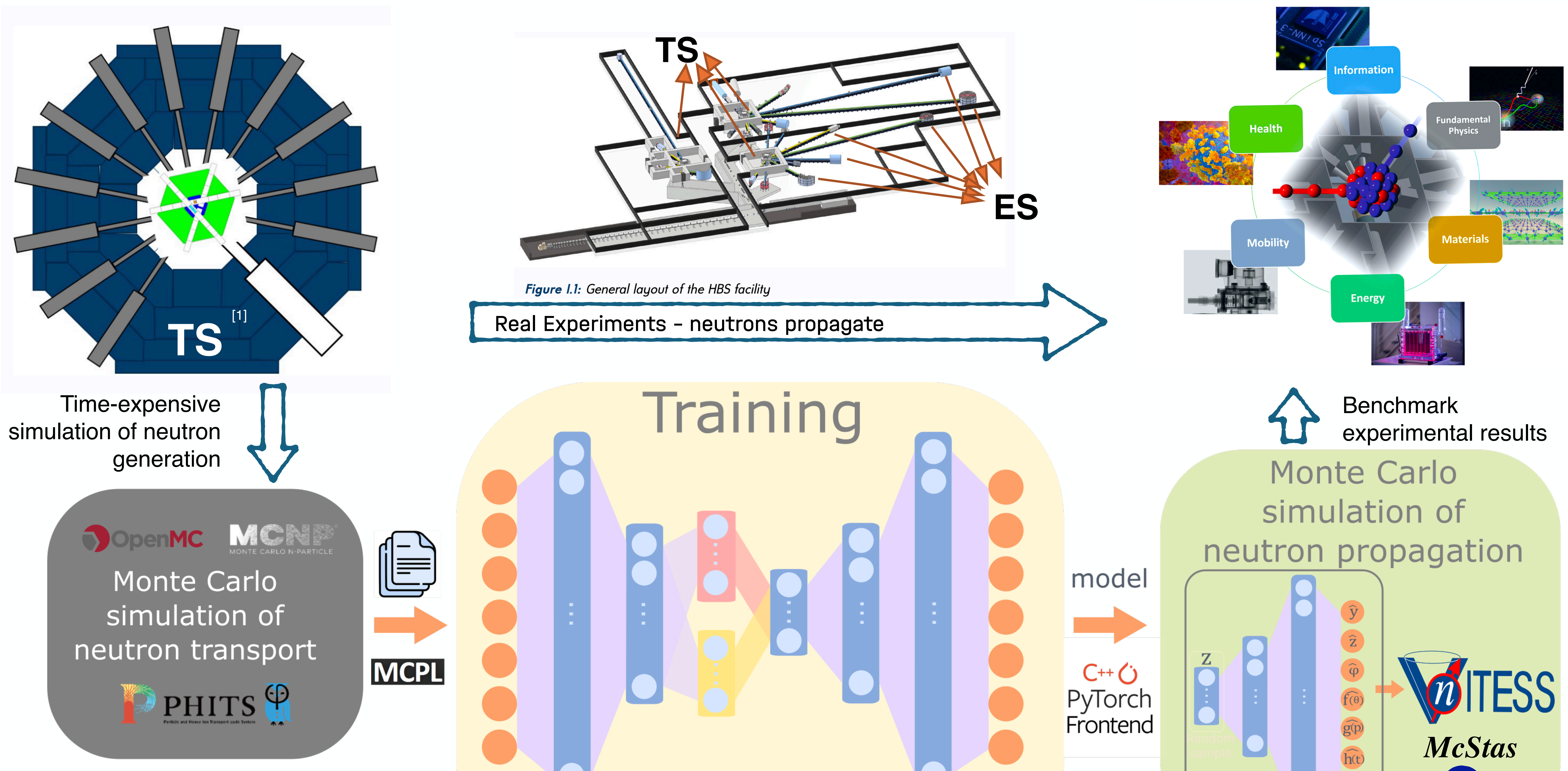
https://laion.ai/blog/laion-pop/

All captions from LAION-Aesthetics with score > 6 (n=12M)
Embedded with CLIP, UMAP to 2d



Why can't we generate neutrons?

INTRODUCTION

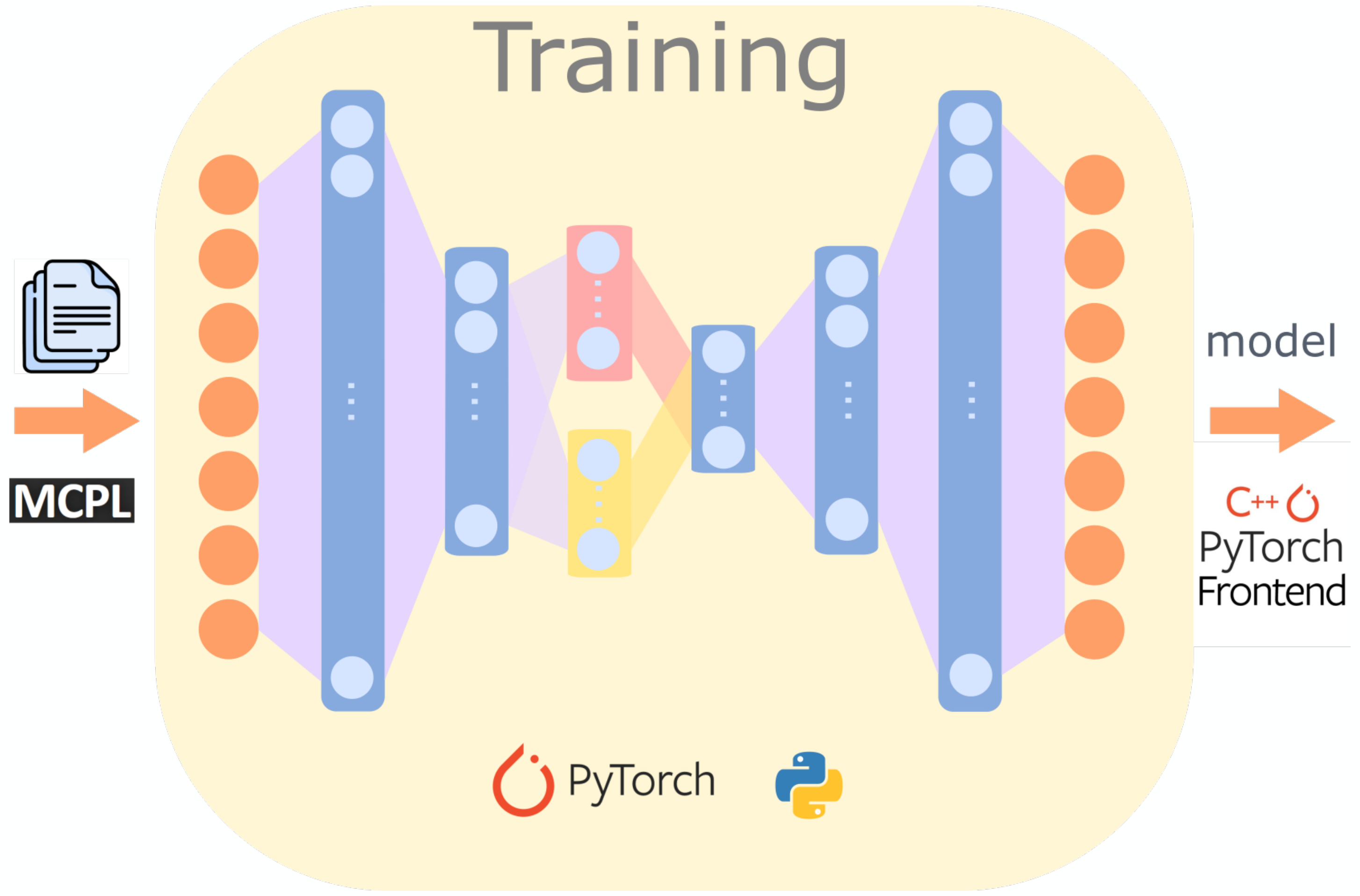


TDR MCPL file

700.000 Neutrons

https://www.fz-juelich.de/en/jcns/jcns-2/news/announcements/2023/hbs-tdr

TRAINING DATA: MCPL FILES



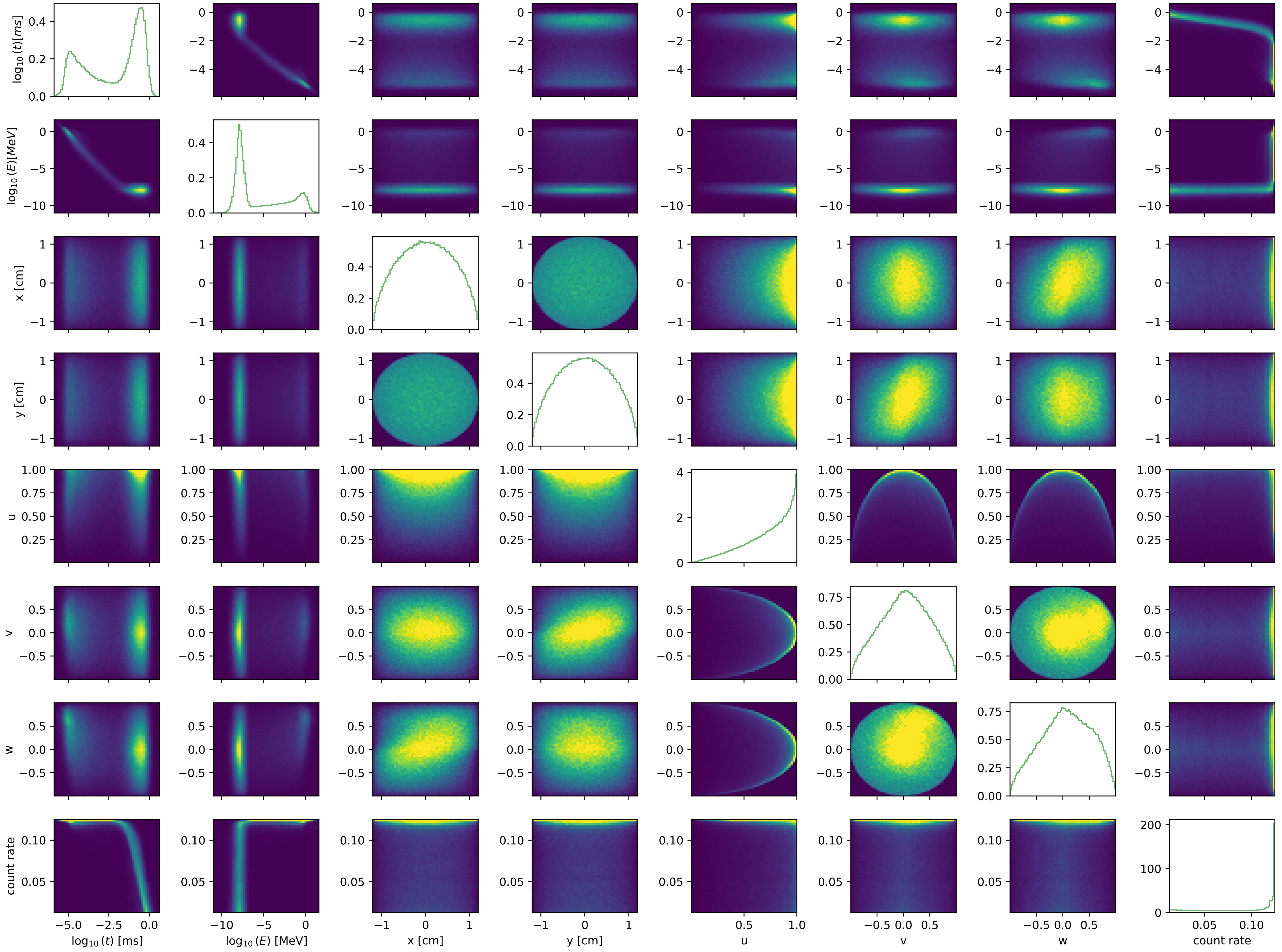
PyTorch DataLoader

Variables

- y [cm] and z [cm]
- u, v, and w
- count rate (weight)
- $\log_{10}(t)$ [ms]
- $\log_{10}(E)$ [MeV]

	pos_y	pos_z	u	v	w	count_rate	TOF	E
0	-1.157290	-0.156233	0.974585	-0.203838	-0.092924	0.124997	0.000059	9.301730e-02
1	-0.675784	0.331821	0.534119	-0.218695	0.816633	0.074287	0.157121	8.421972e-08
2	0.598687	-0.840697	0.647142	0.740285	-0.182169	0.074748	0.156539	8.232719e-08
3	0.775426	-0.258992	0.508785	0.752556	0.418089	0.075199	0.156472	1.423273e-07
4	-0.778382	0.114899	0.702195	-0.399447	-0.589376	0.122384	0.007535	5.222545e-08
...	...	...	...	...	...	...	...	...
676553	-0.711618	-0.400495	0.916834	-0.103239	-0.385690	0.124989	0.000039	8.381045e-03
676554	-0.052427	0.881021	0.818265	0.572754	-0.048949	0.124998	0.000005	2.565348e-01
676555	-0.042421	0.981967	0.545910	-0.667388	0.506533	0.025299	1.222780	2.869063e-09
676556	0.244454	-0.181313	0.743137	-0.051606	0.667146	0.080170	0.105039	2.322999e-08
676557	-1.063570	-0.350045	0.633484	-0.657062	-0.408617	0.032235	0.657213	6.777769e-09

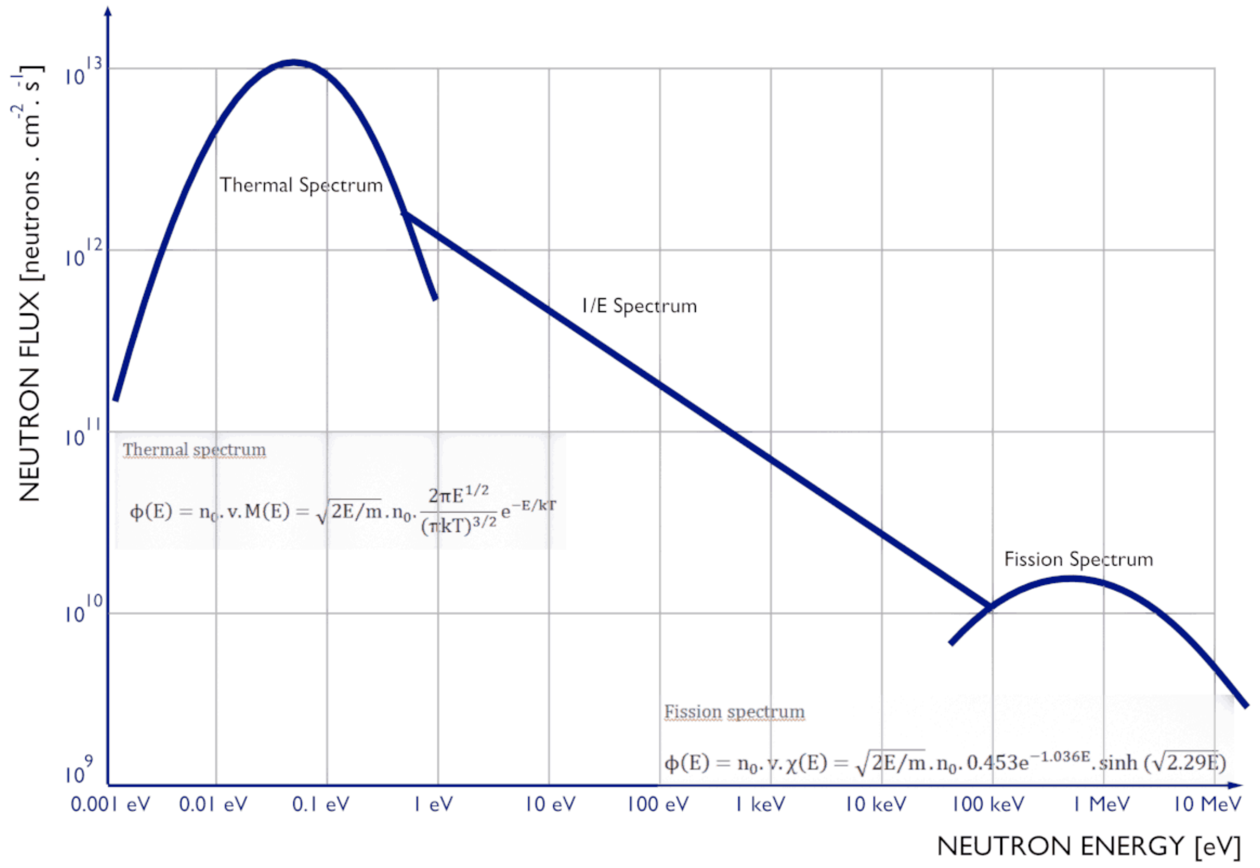
676558 rows × 8 columns

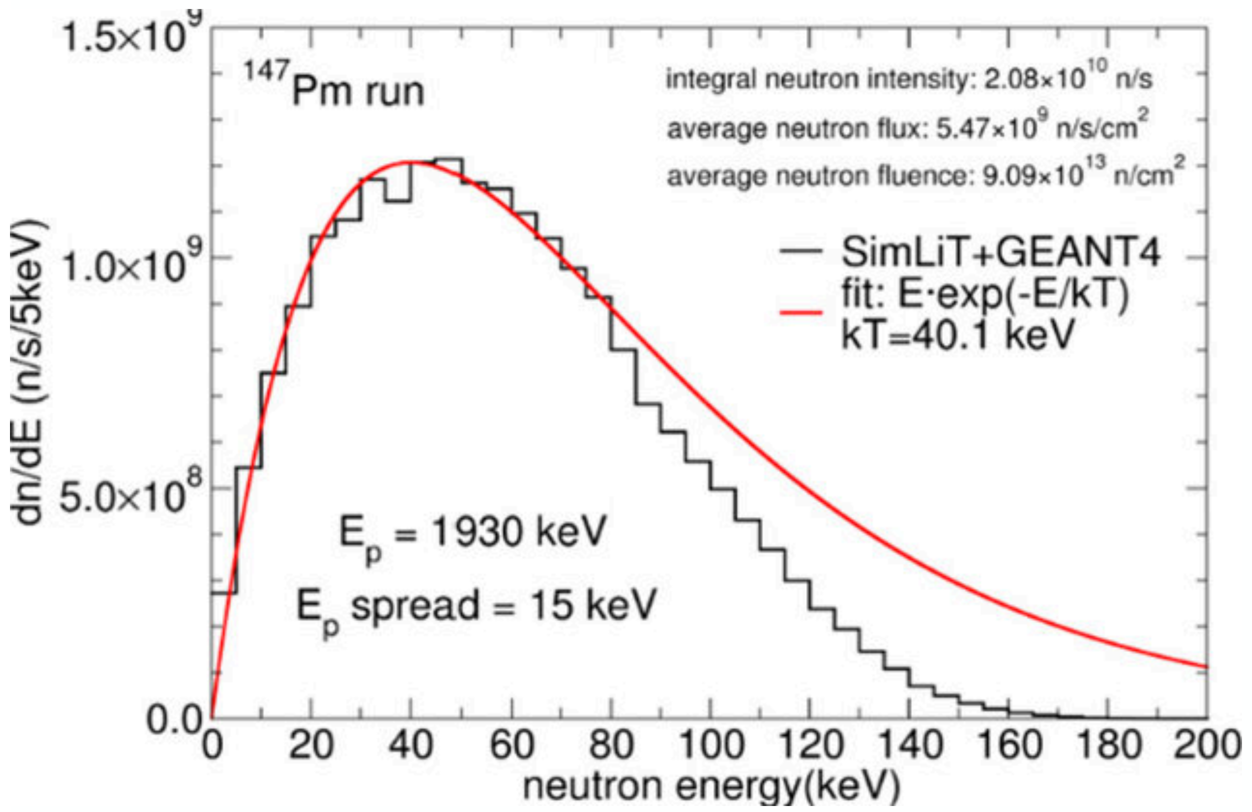


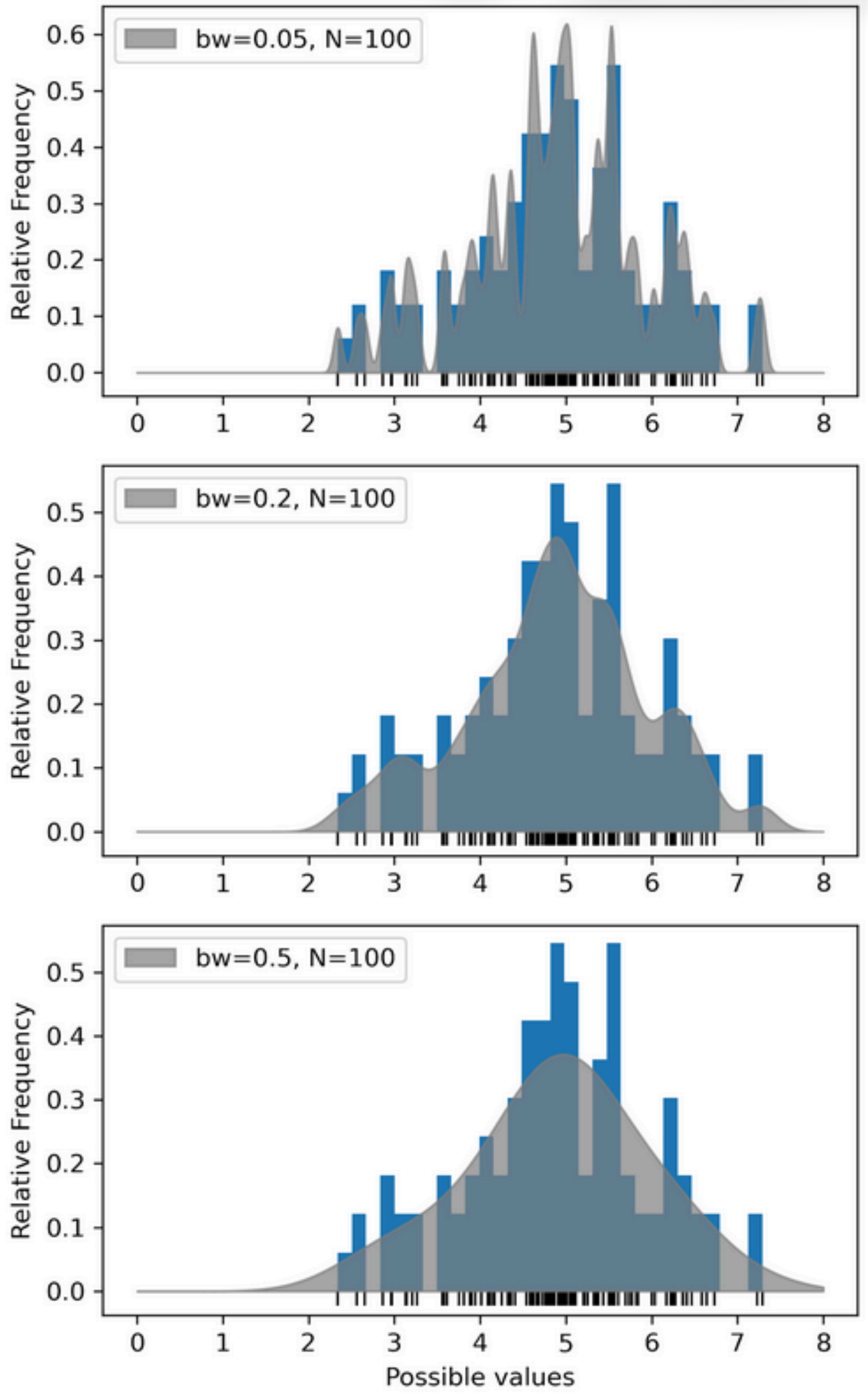
T. Kittelmann et al. Monte Carlo particle lists: MCPL. *Computer Physics Communications*, *218*, 17-42.

CURRENT APPROACHES TO SOURCE ESTIMATION

- Analytical approximation based on theory / fitting to observed data







- Kernel Density Estimation

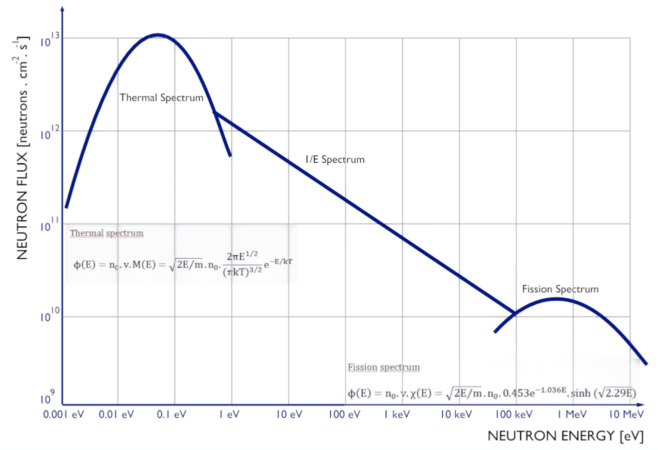
KDSource

$$\hat{f}(\mathbf{x}) = \hat{f}(x_1, x_2, \ldots, x_D) = \sum_{\mathrm{i}=1}^{N} w_{\mathrm{i}} \left\{ \prod_{\mathrm{j}=1}^{D} \frac{1}{h} K\left(\frac{x_{\mathrm{j}} - (\tilde{p}_{\mathrm{i}})_{\mathrm{j}}}{h} \right) \right\}$$

Schmidt, N. S. et al. (2022), KDSource, a tool for the generation of Monte Carlo particle sources using kernel density estimation. *Annals of nuclear energy*, *177*, 109309.

PROS & CONS

- Analytical approximation based on theory / fitting to observed data
  - Reliable when assumptions are adequate (Theoretical foundation)
  - It is an approximation and assumptions are needed
  - simple and fast for sampling
  - lack features specific to individual cases
  - Fitting parameters are characteristic of the distribution (interpretability)



- Kernel Density Estimation
  - Non-parametric approach, making it adaptable (flexibility)
  - Although, hyper-parameter: Bandwidth and kernel
  - Data-driven, potentially providing a better representation of the distribution
  - Computational cost for high dimensional spaces
  - Fast sampling
  - **Data dependency for sampling**

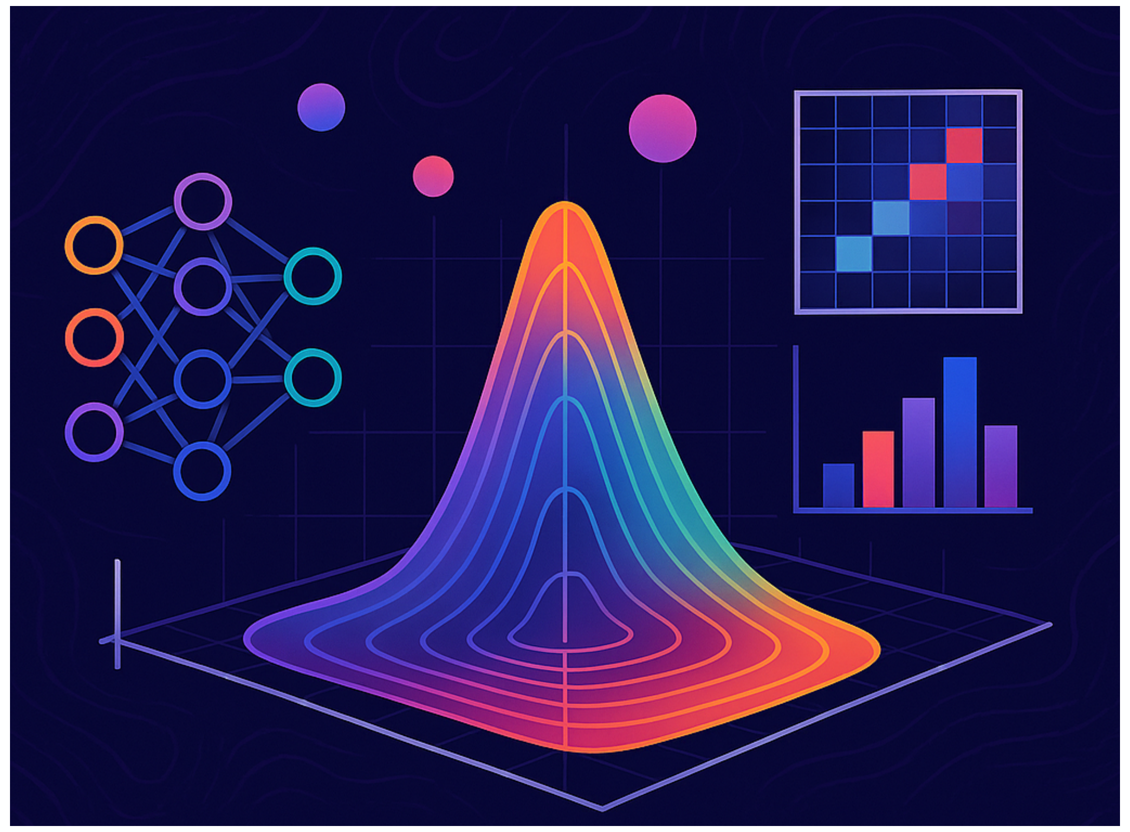
KDSource

GENERATIVE MODELS

- **Objective:** Learn the underlying patterns and distributions of a dataset and generate new data points that resemble the original

We can use generative models to learn the multivariate phase-space distribution of neutrons from a Monte Carlo Particle List (MCPL file)

- *Easy to generate new neutrons to propagate inside tracing software.*
- *capable of learning and generating data with complex patterns and structures between phase-space variables! (High fidelity and realism)*
- *Not limited to specific types of data (Flexibility)*
- ***Once trained, no need to keep the original dataset used to train it.***
- *High-computational cost for training*
- *Large datasets*
- *Lack of interpretability*
- *Depending on model size, slow for sampling*

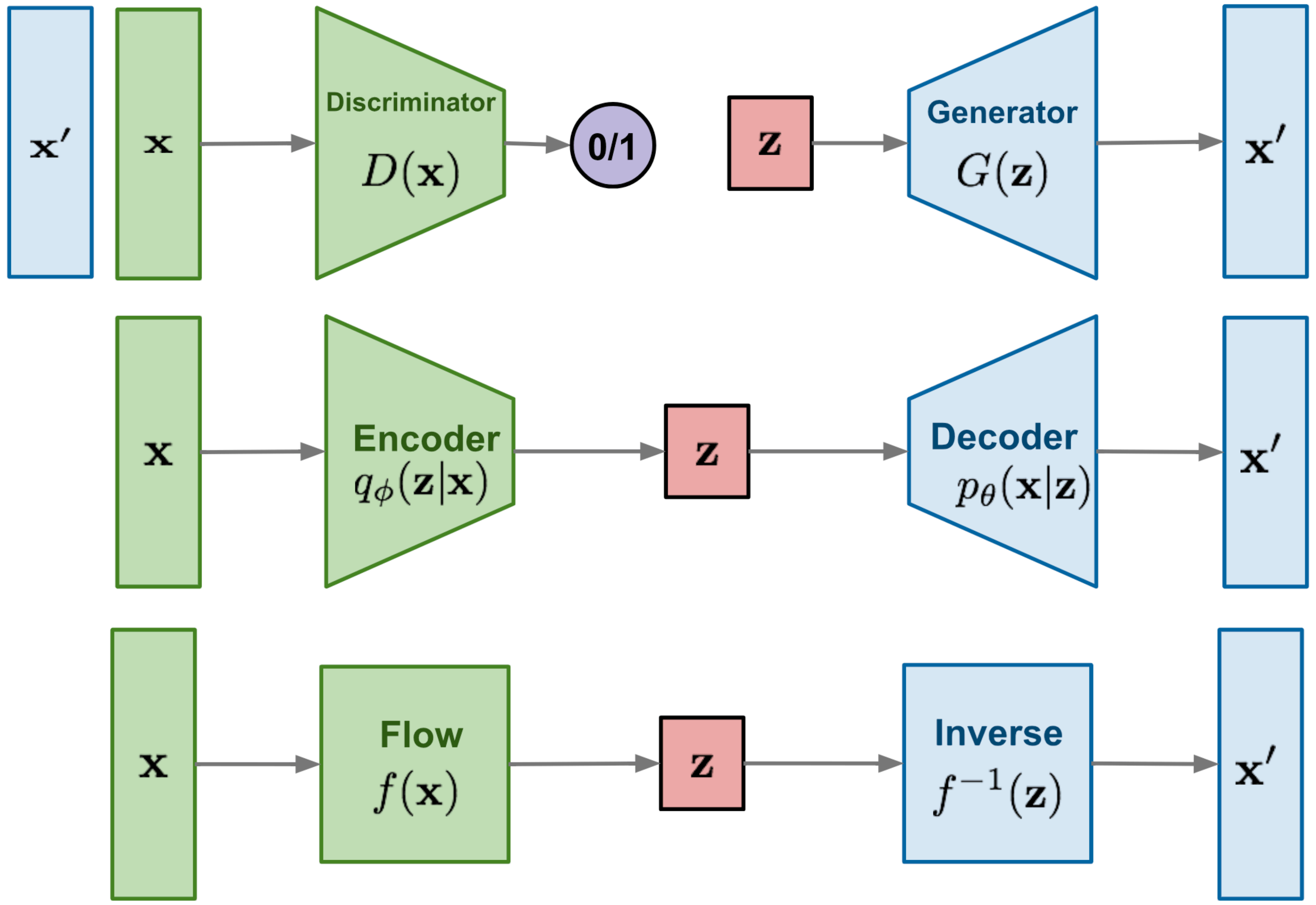


GENERATIVE MODELS

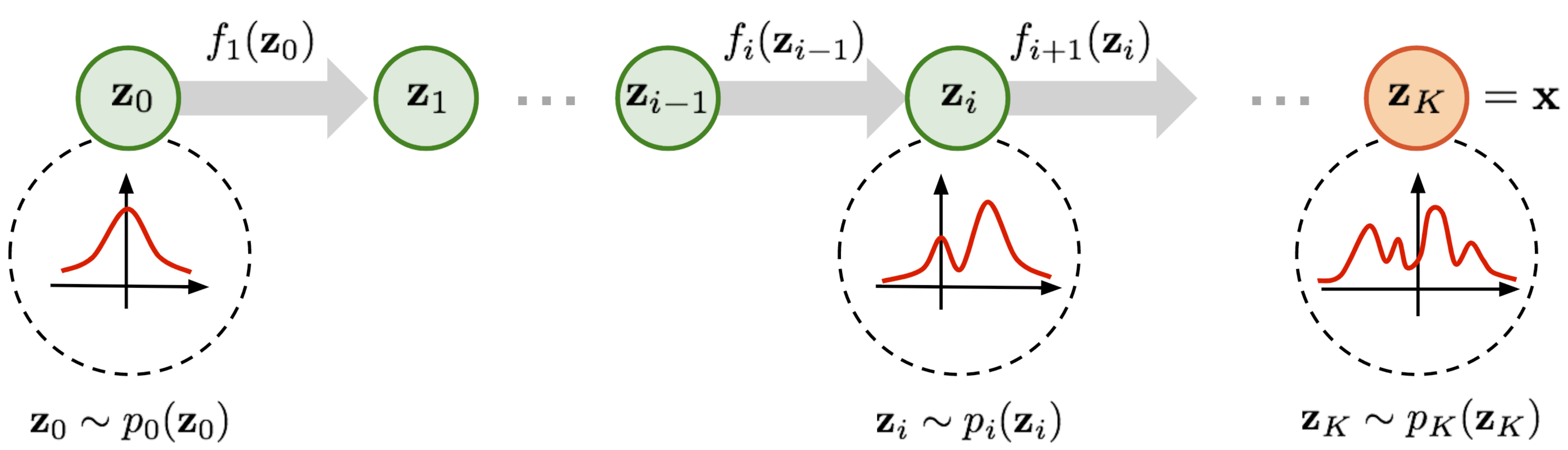
GAN: minimax the classification error loss.

VAE: maximize ELBO.

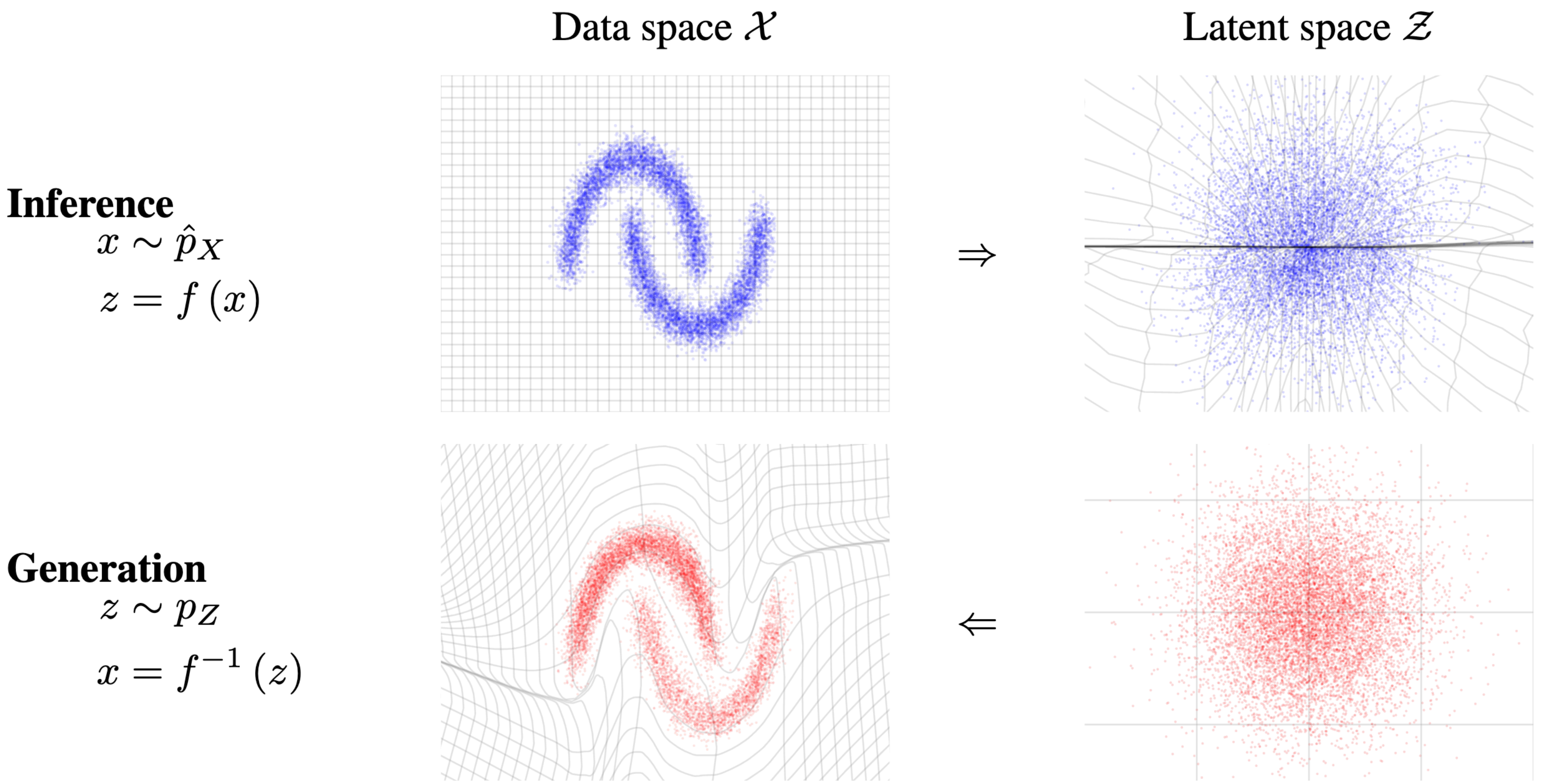
Flow-based generative models: minimize the negative log-likelihood



Normalizing Flows



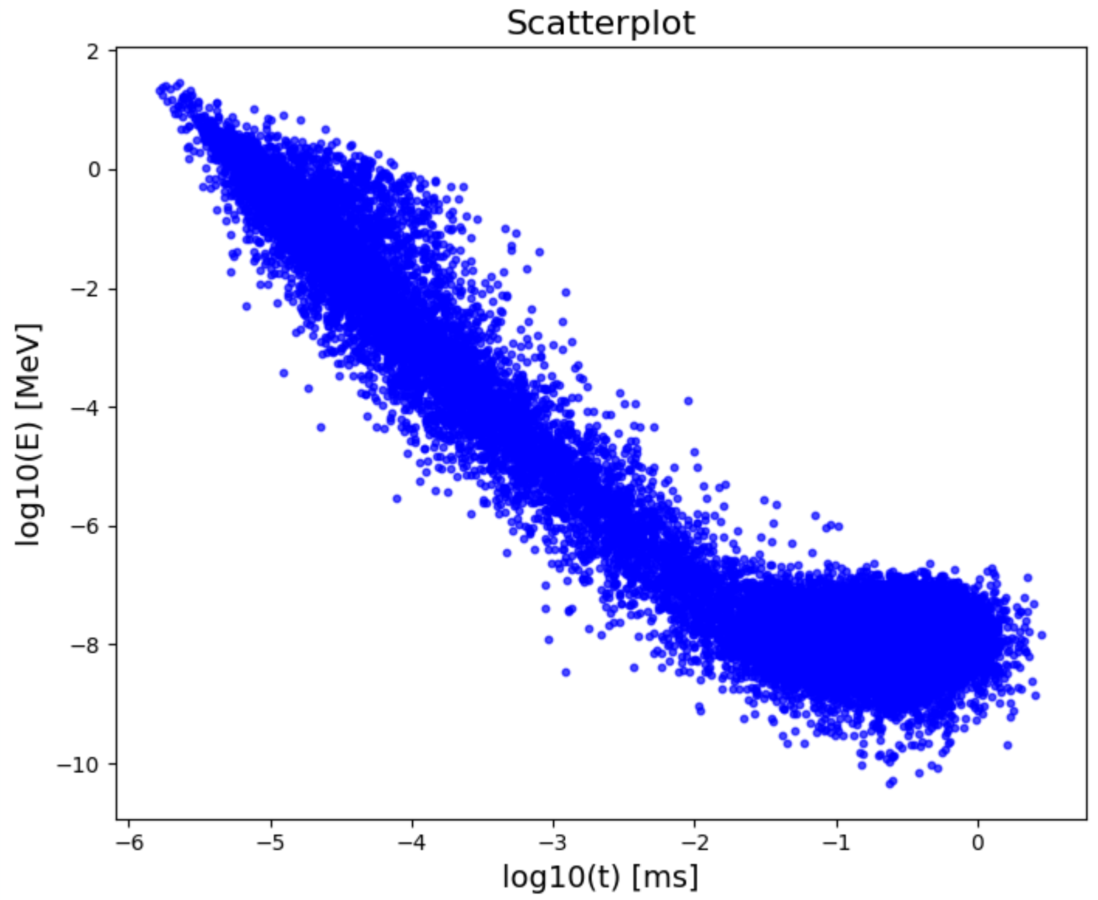
$$p_x(x) = p_z(z)\left|\frac{dz}{dx}\right|$$

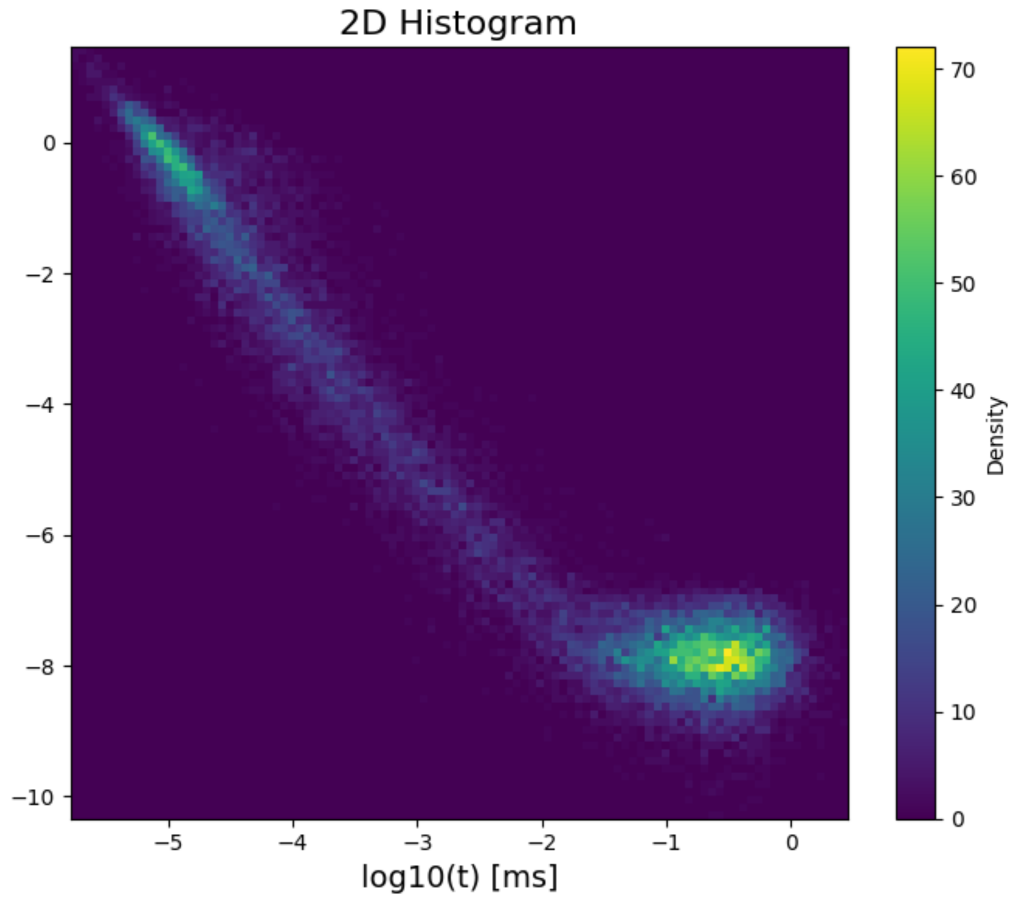


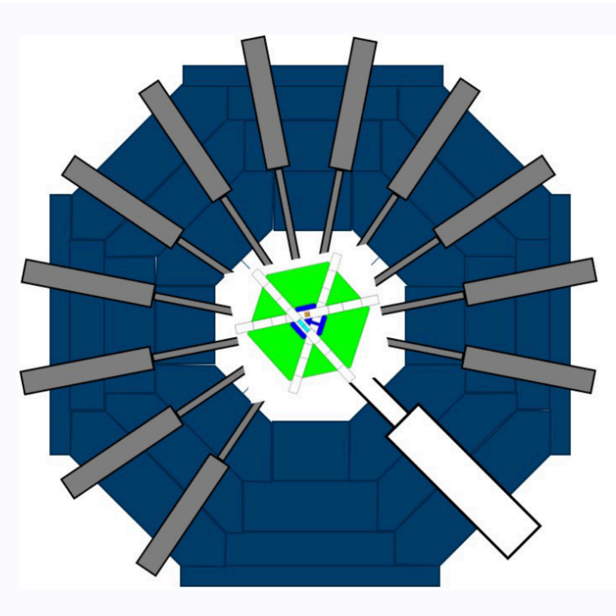
https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/tutorial11/NF_image_modeling.html

TOY EXAMPLE ON 2D

20.000 neutrons, only time and energy

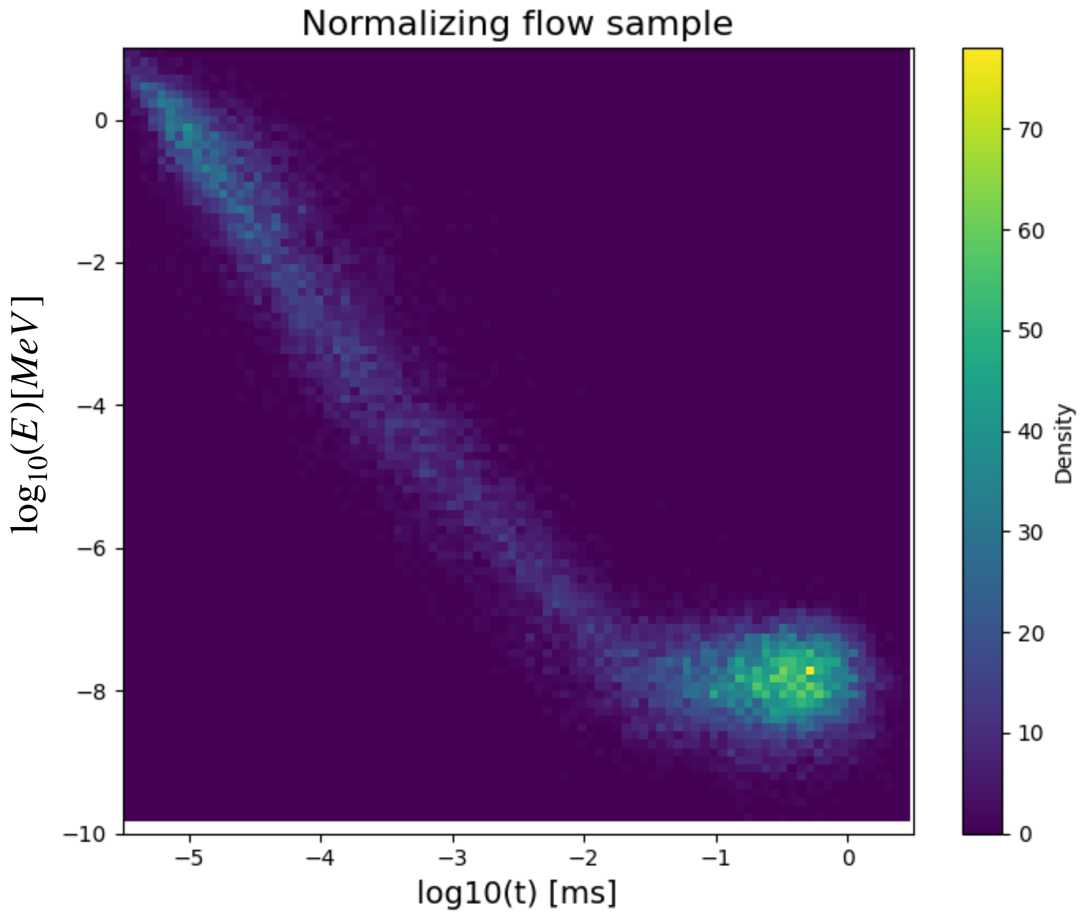


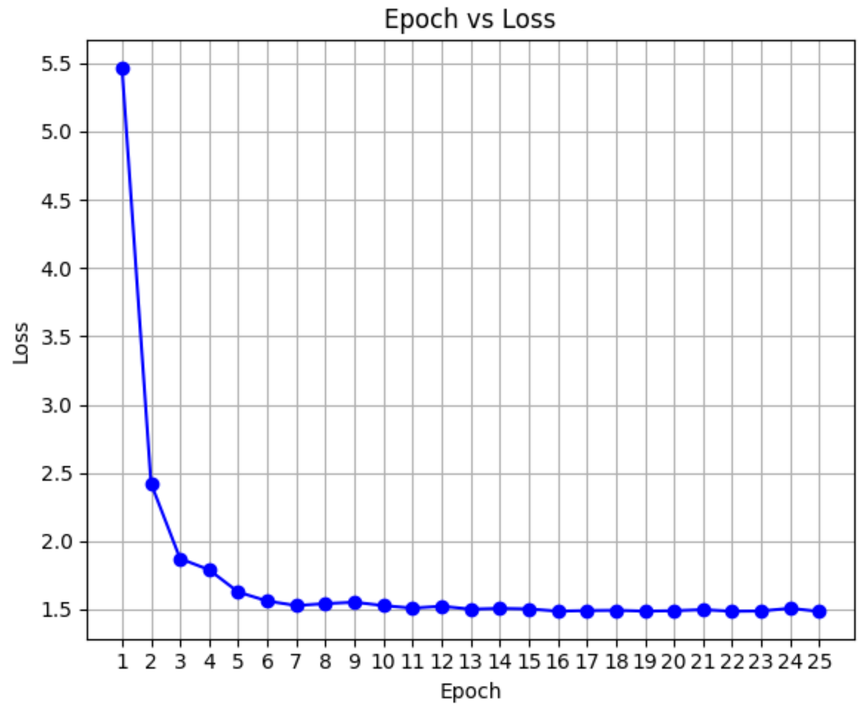




After 25 epochs

Sampling 20000

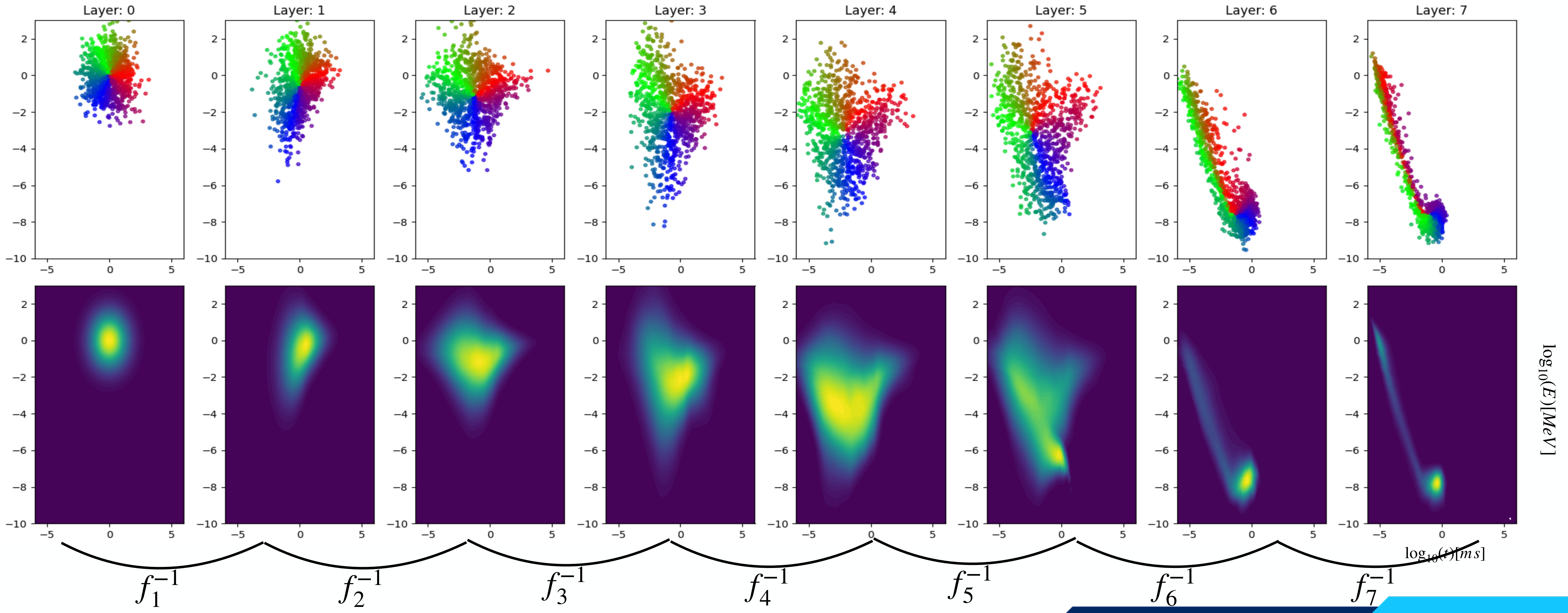




Invertible transformations

2D Normal distribution

Easy to sample



Artificial sample

HELMHOLTZ AI Artificial Intelligence Cooperation Unit

JÜLICH FORSCHUNGSZENTRUM

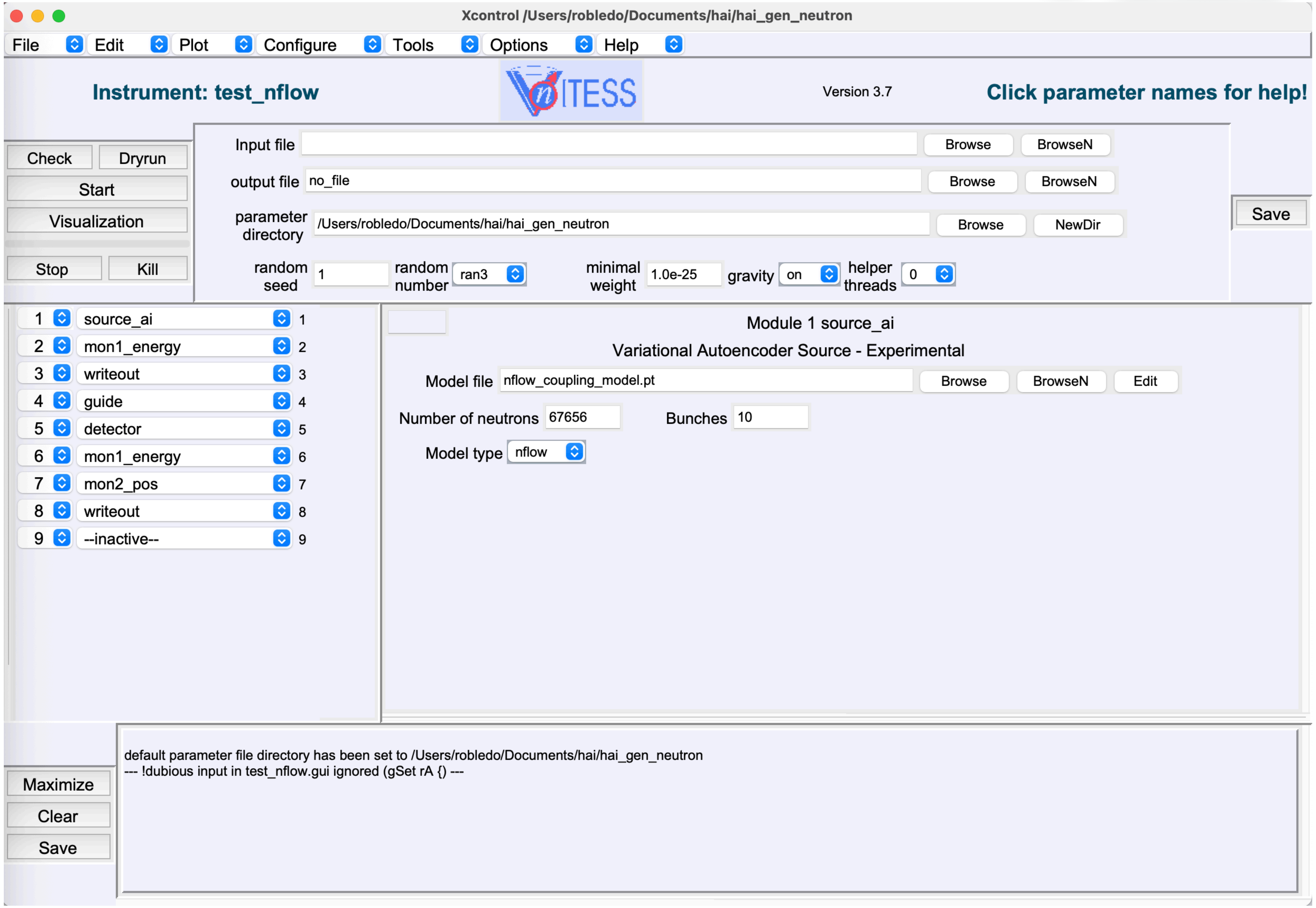
VITESS

New source_AI module

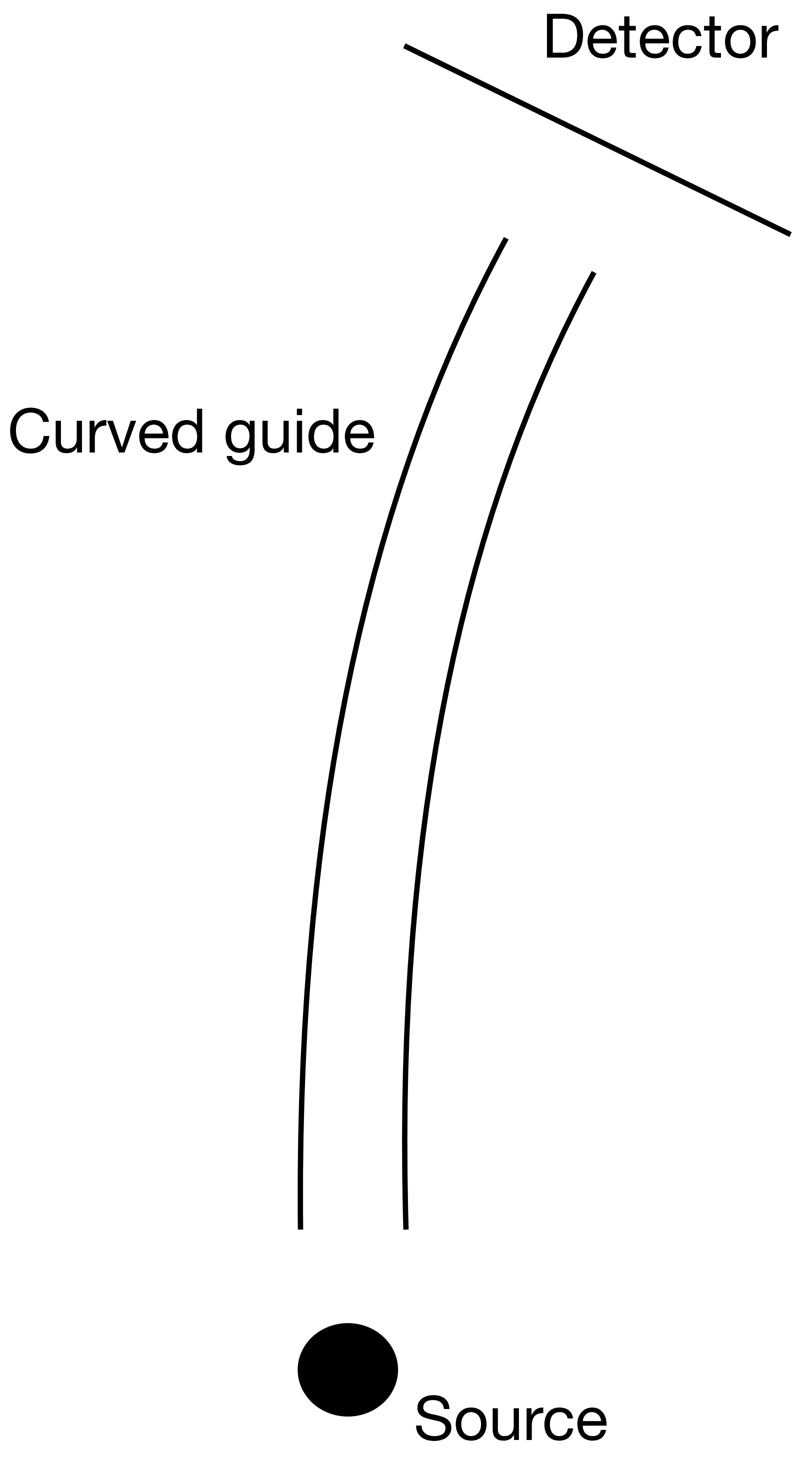
PyTorch

C++ Frontend

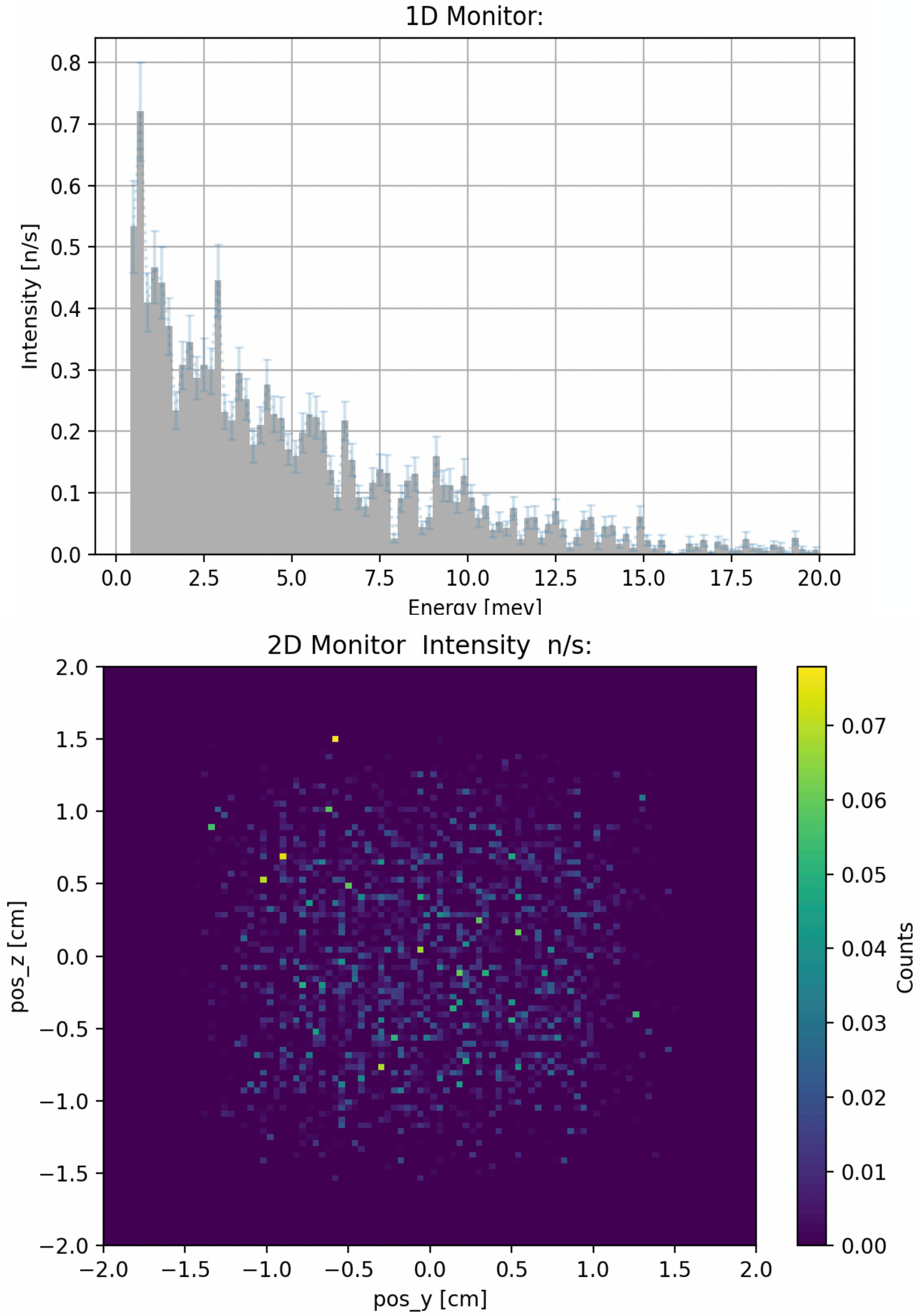
Model needs to be jit compiled



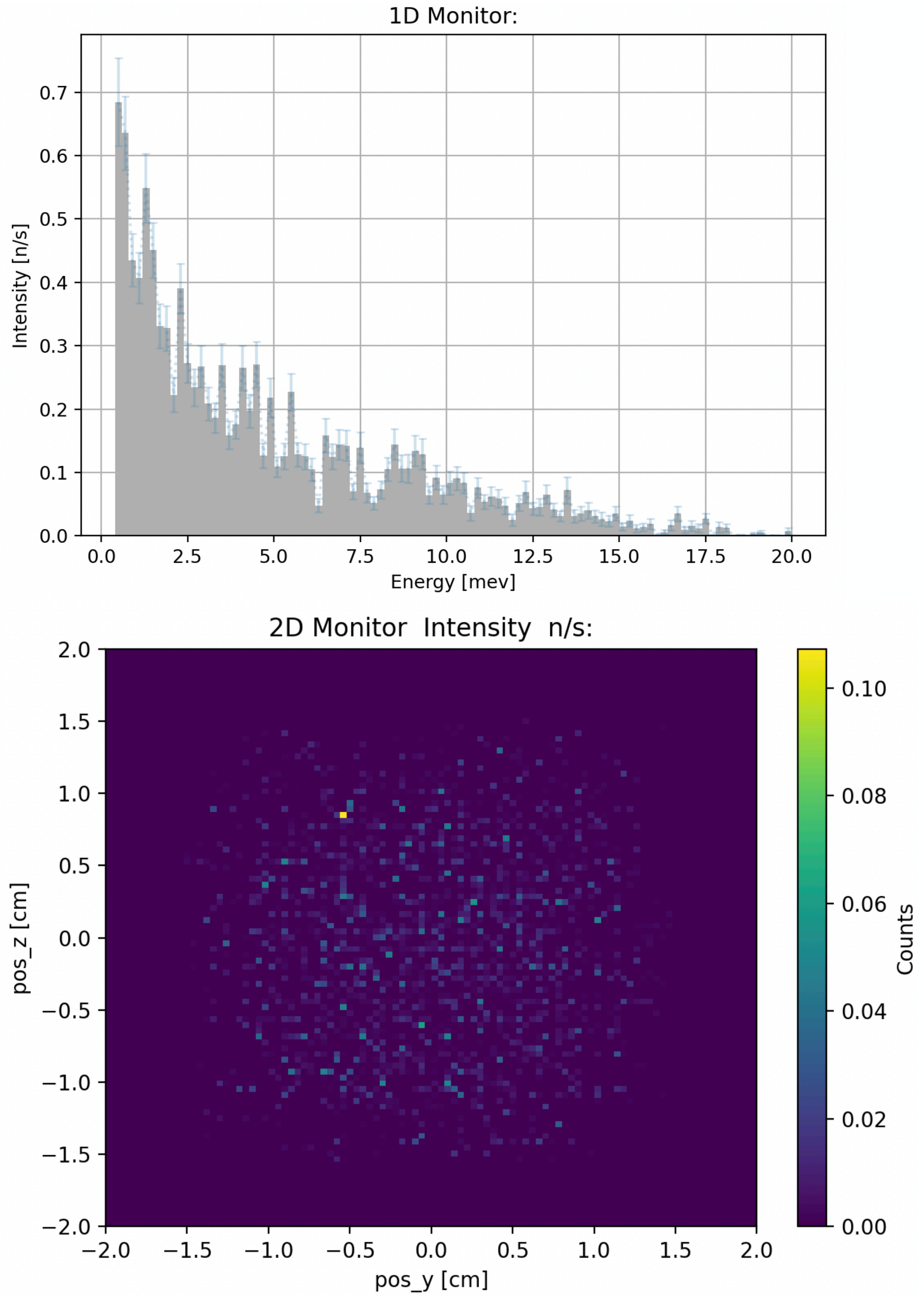
SAMPLING ON VITESS



mcpl

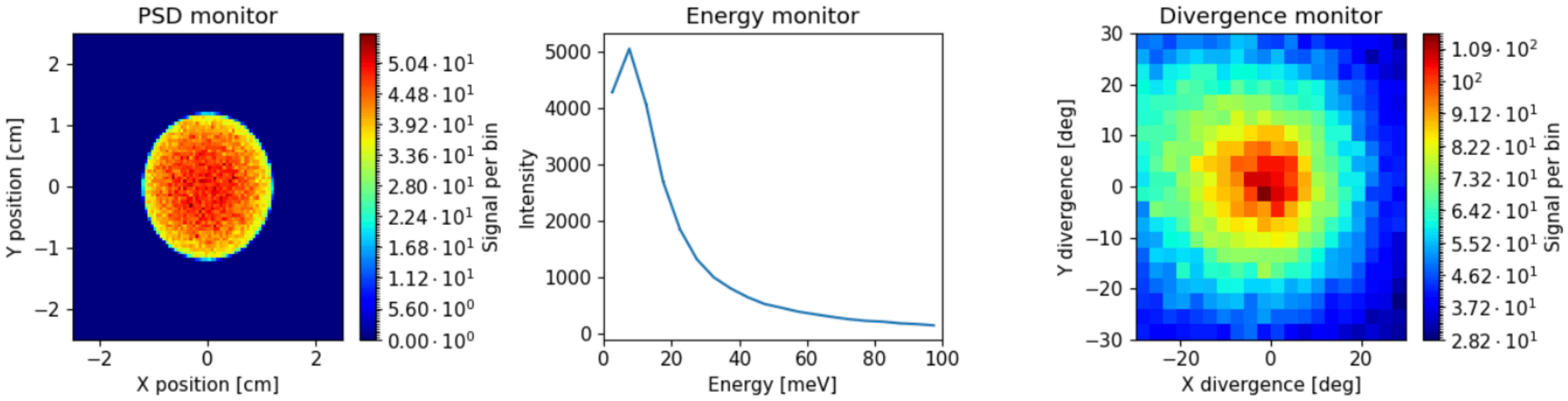


NFlow



MCSTAS

```python
import np2mcpl

# Load model
model = torch.load("nflows_model.model", map_location=torch.device('cpu'), weights_only=False)

# sample model
samples = model.sample(int(1e7))

# Transform to adapt to mcpl format if necessary

...

# Save data
np2mcpl.save("output", samples)

# Load data
import mcstasscript as ms
instrument = ms.McStas_instr("sample_normalizing_flow")
source = instrument.add_component("source", "MCPL_input")
source.filename = '"output.mcpl.gz"'

PSD = instrument.add_component("PSD", "PSD_monitor")
PSD.set_AT([0, 0, 0.2], RELATIVE=source)
PSD.set_parameters(xwidth=1, yheight = 1, filename='"PSD.dat"')
data = instrument.backengine()
ms.make_sub_plot(data)
```

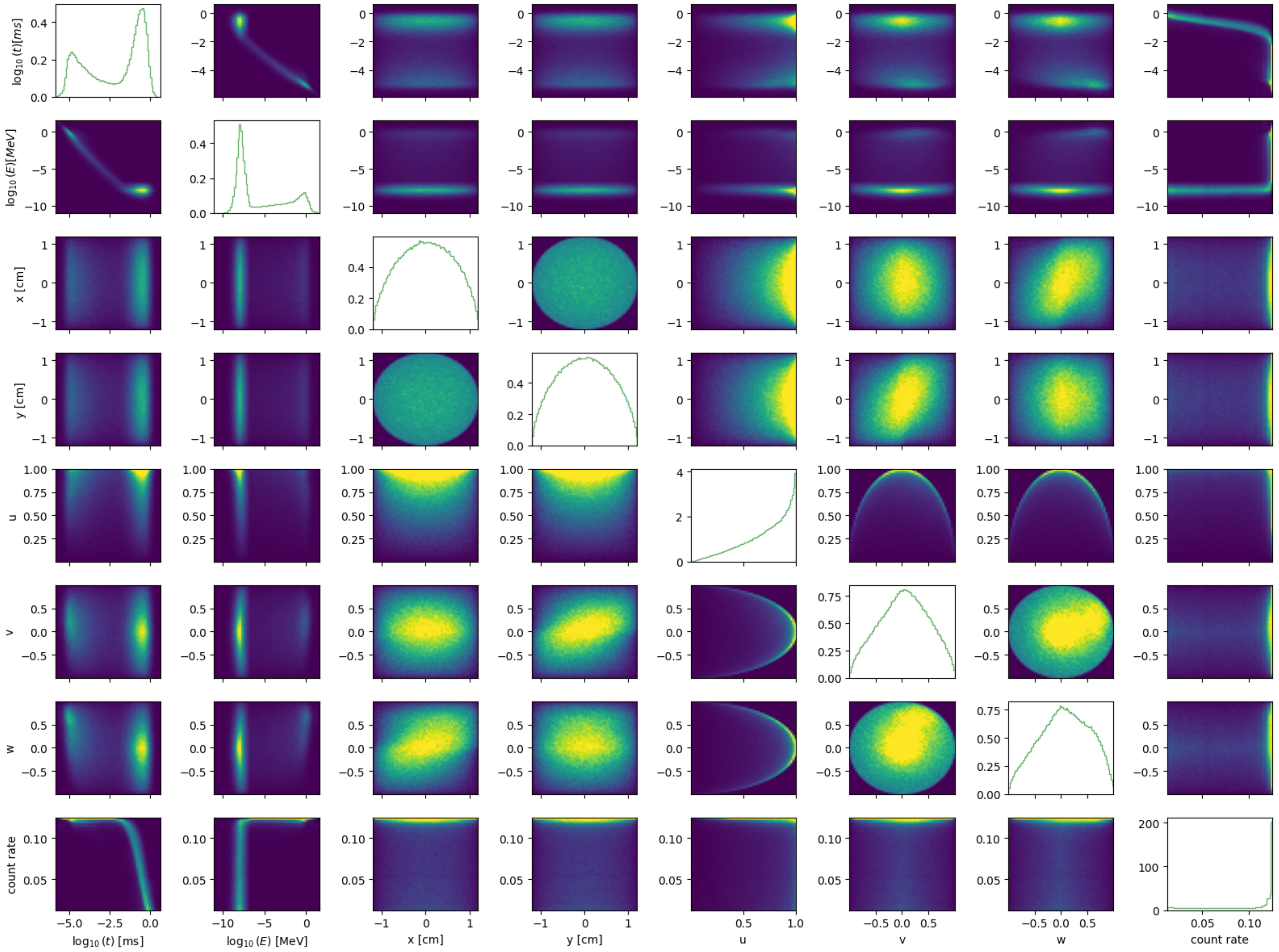


*Same procedure can be done in **vitess-python***

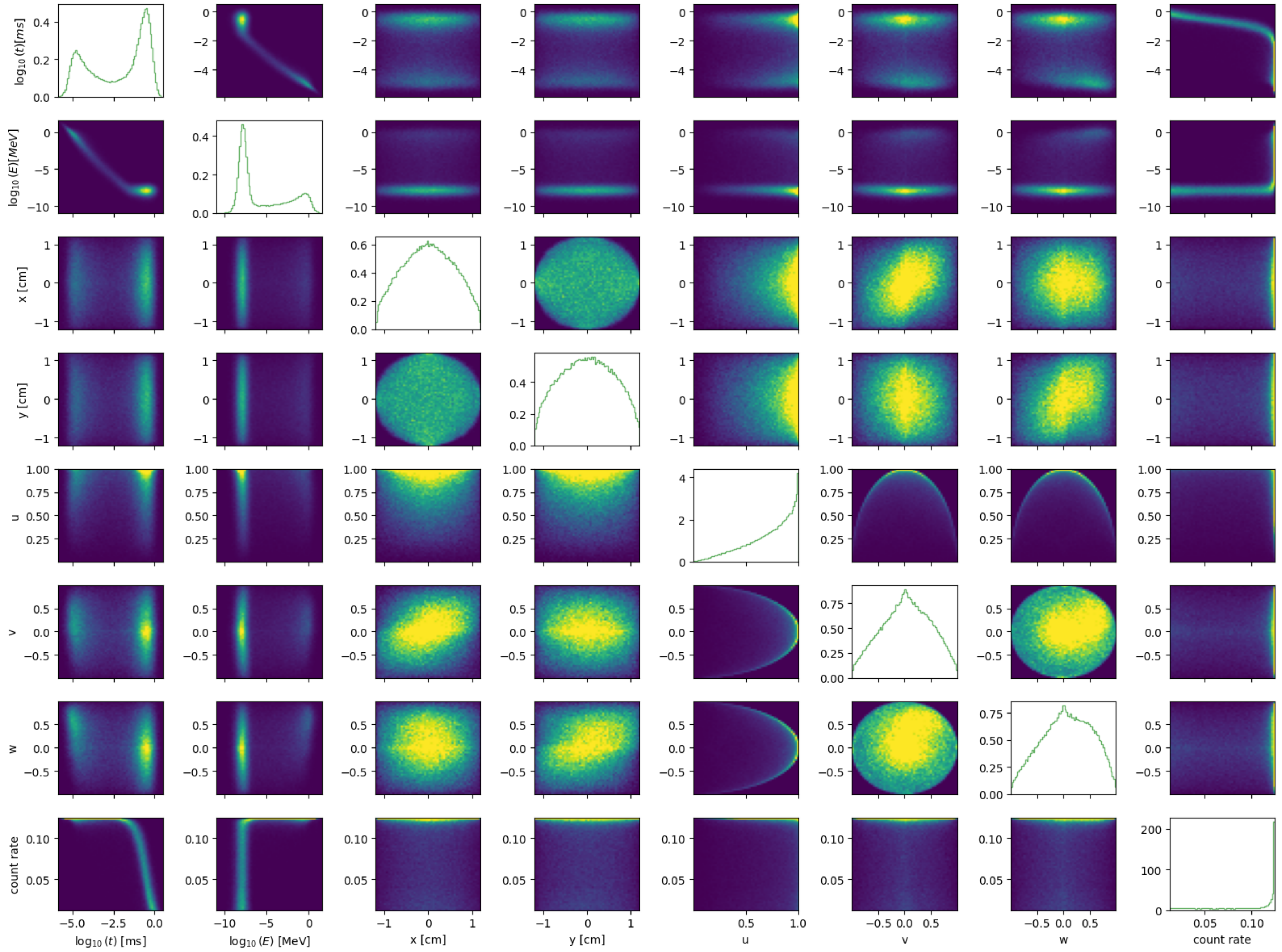
*Poster: "**Vitess-Python: A Python API for VITESS Instruments**" - Fabian Beule*

Comparison between NFlow and MCPL

MCPL histograms



NFlow histograms



Comparison between NFlow and MCPL

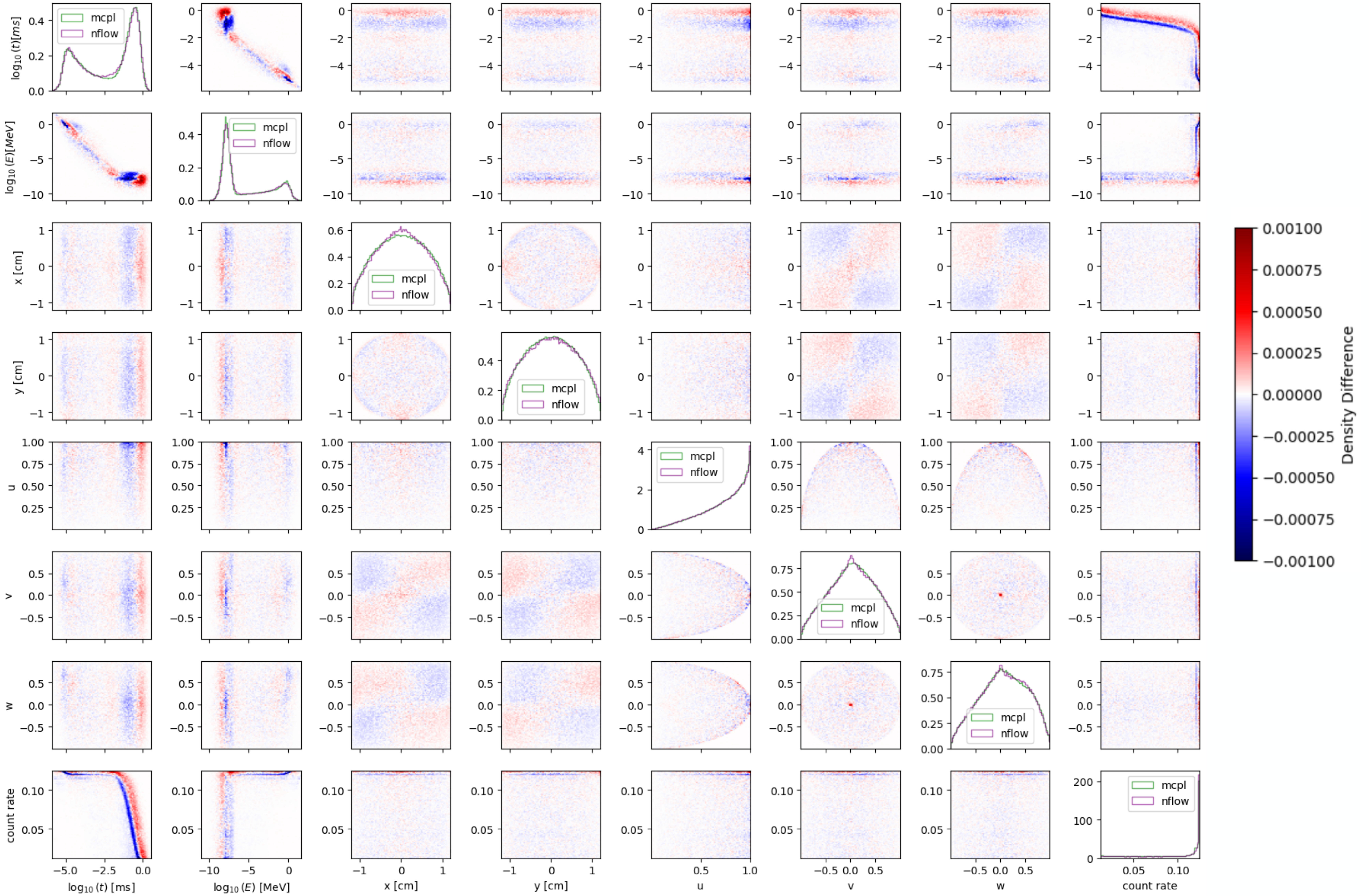
Blue: under-estimate

Red: over-estimate

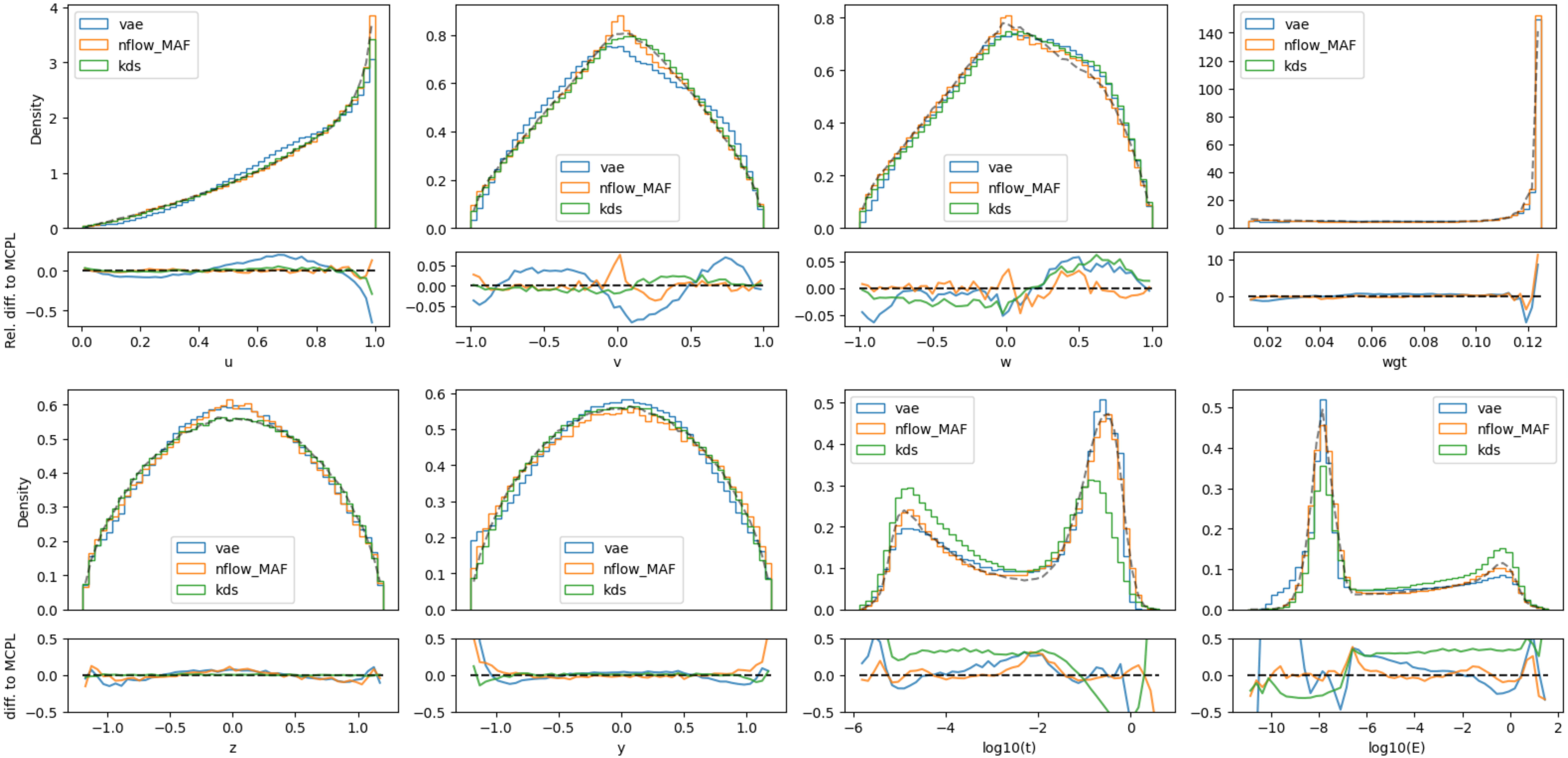
model can be stored in a file consisting of **few KB**

easily loadable through PyTorch API

we can sample from the latent space

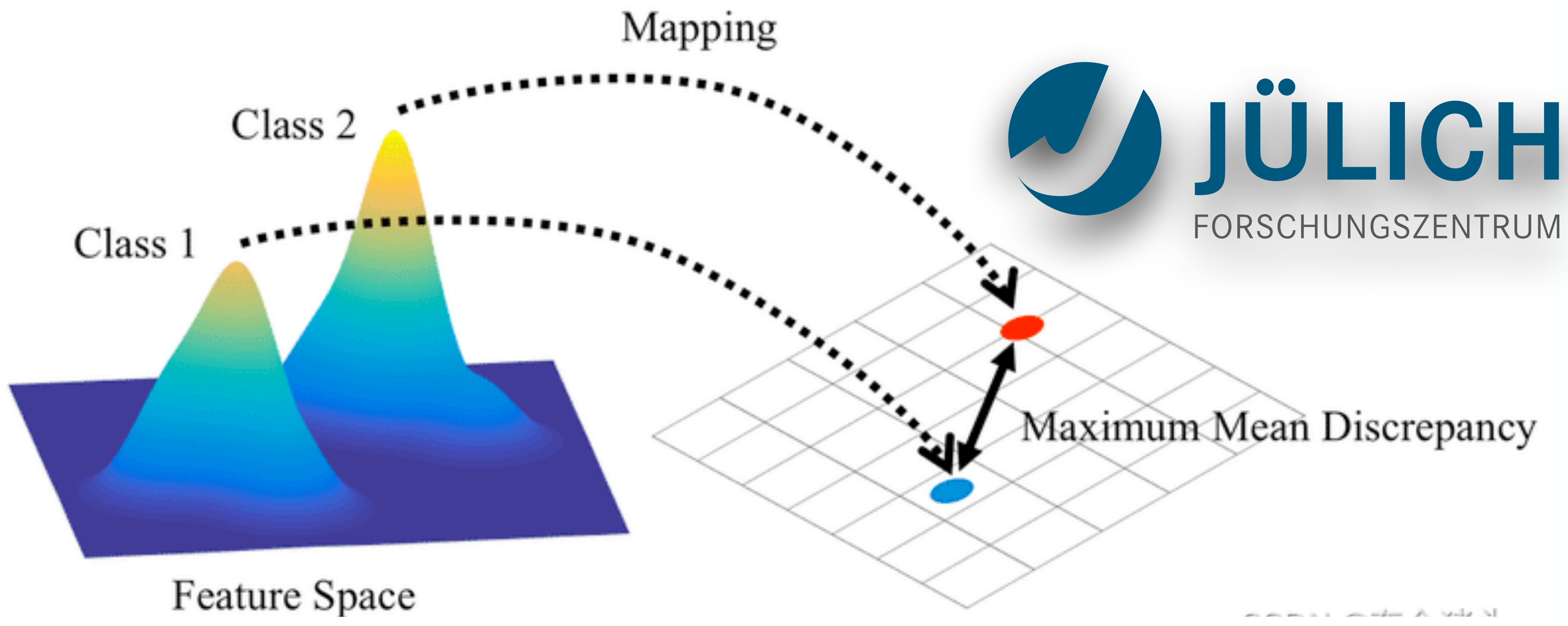


COMPARISON BETWEEN MODELS



HELMHOLTZ AI Artificial Intelligence Cooperation Unit

QUANTIFYING DIFFERENCES



JÜLICH FORSCHUNGSZENTRUM

Statistical measure: Maximum mean discrepancy (MMD)

Determine if two datasets are likely to have been drawn from the same distribution by embedding probability distributions into a Reproducing Kernel Hilbert Space (RKHS) and then calculating the distance between the means of these embeddings.

Model	Average MMD *
MCPL	0.00015 ± 0.00016
MAF NFlow	**0.00053 ± 0.00010**
Coupling Flow	0.00171 ± 0.00015
VAE	0.00308 ± 0.00017
KDSource	0.08014 ± 0.00143
Uniform	0.15231 ± 0.00149

* sample size = 10000, average over 10 samples

Model	Sampling time / 700.000 n (s)
KDSource	2
VAE	8
NFlow	7

SUMMARY

- Generative models can learn multivariate probability distributions from data
- MCPL files make great training data!
- Normalizing Flows show great potential in learning neutron phase-space variable distributions, but they can estimate poorly if distributions have sharp features.
- These sources can already be used in Vitess and McStas, and are easily extensible to other Monte Carlo software.
- There are multiple architectures of NFs, as well as VAEs and GANs. Exploring which model does best is still an art. Physical constraints can be added inside the loss function.
- Metrics for comparing multivariate distributions should be taken into account for model selection.

Thank you for your attention! Questions?

Jose Robledo

Forschungszentrum Jülich