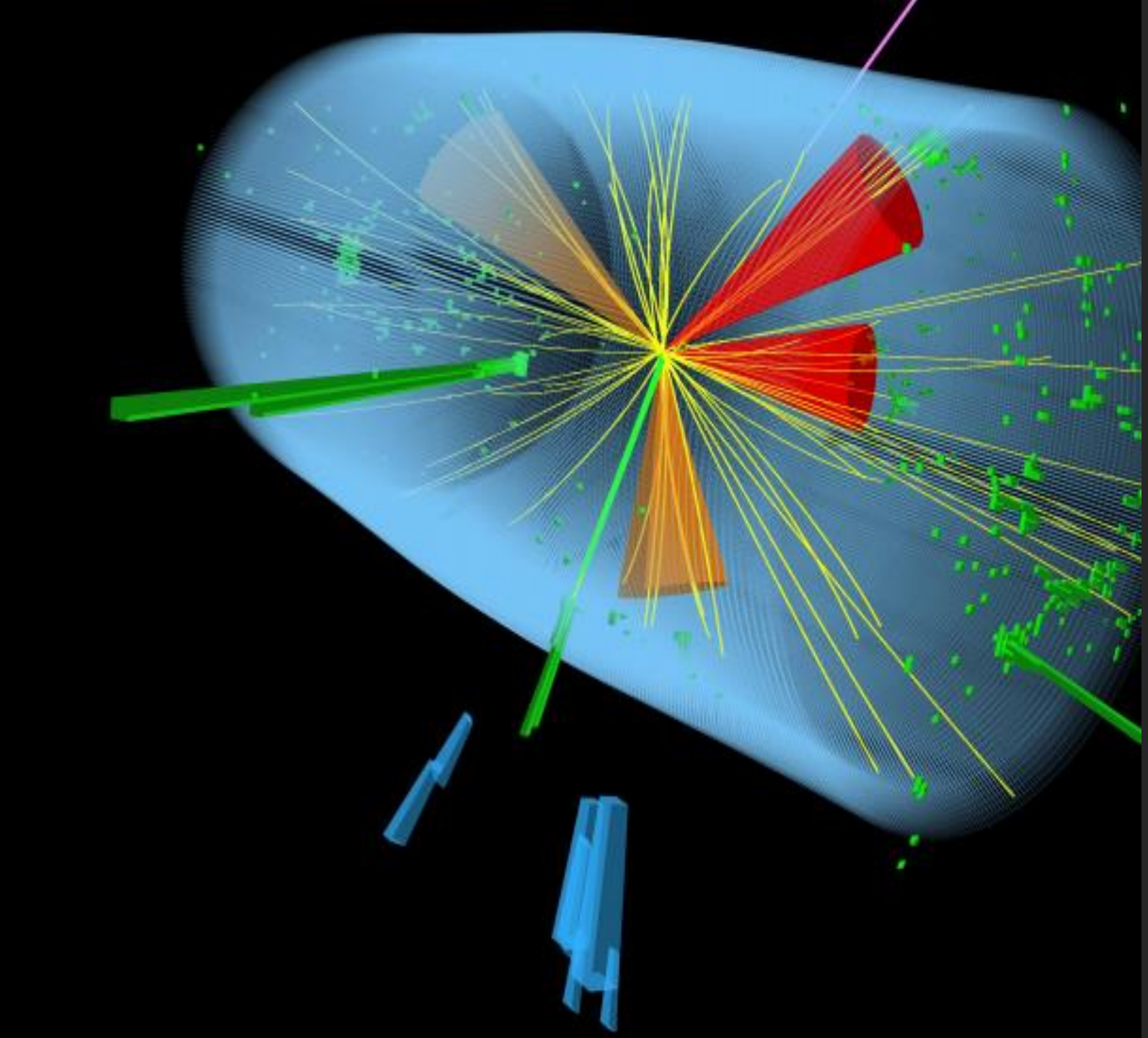


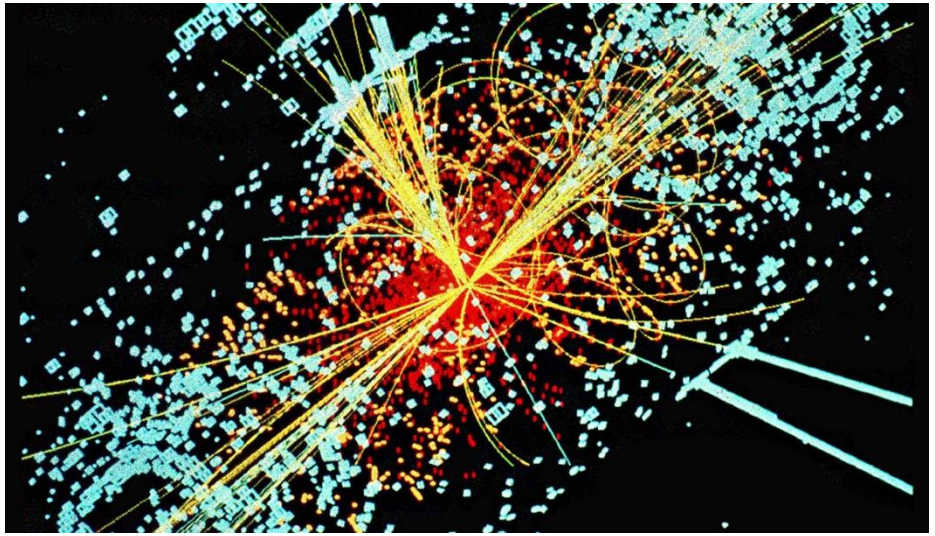
Graph Variational Autoencoder for Detector Reconstruction and Fast Simulation in High-Energy Physics

Authors:

- *Ali Hariri (American University of Beirut)*
- *Darya Dyachkova (Minerva Schools at KGI)*

Conventional physics simulations

- Particle collisions taking place at the LHC are very complex. There exists a good first principles model of how particles interact with detectors based on Monte Carlo methods.
- Packages like GEANT were developed to simulate the passage of particles through matter. However, this process **is difficult to parallelize** and is takes significant time to simulate each event.
- We distinguish 2 types of simulation frameworks:
  - Parametric simulations: Fast simulation tools such as Delphes were developed which offer a user interface to specify detector geometries and other simulation environment variables
  - Non-parametric simulations: Other methods are non-parametric: they rely on detector response simulation using a builder and a simulator.



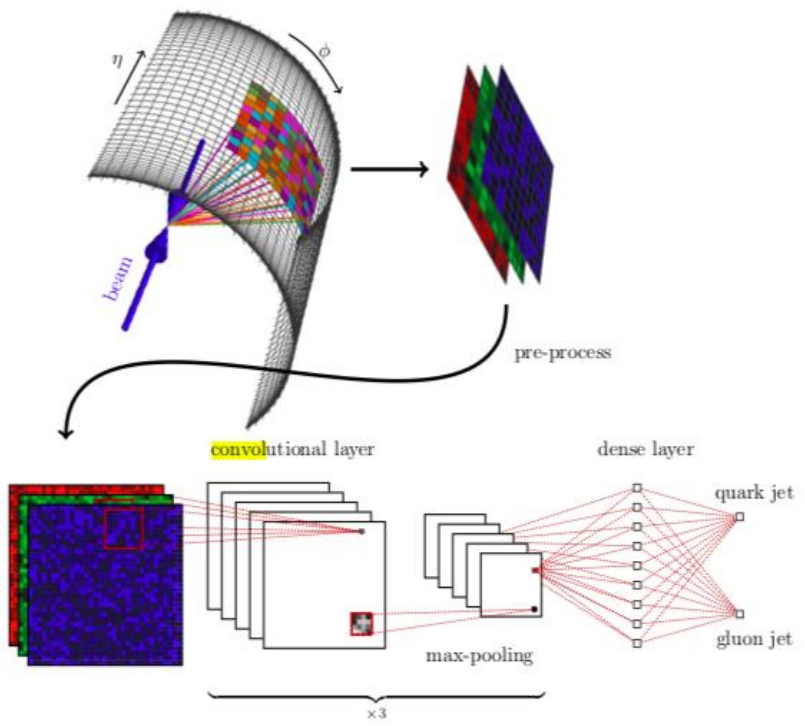
The rise of Deep Generative Models

- Deep generative models showed high potential in computer vision applications such as image generation and segmentation.
- Their ability to model complex distributions and dealing with unlabeled data makes them an interesting candidate for detector physics applications.
- The most widely used generative models are Generative Adversarial Networks and Variational Auto-Encoders.
- Our work sheds light on Variational Auto-Encoders and their ability to learn the properties of collider simulation events in a latent M-dimensional space.

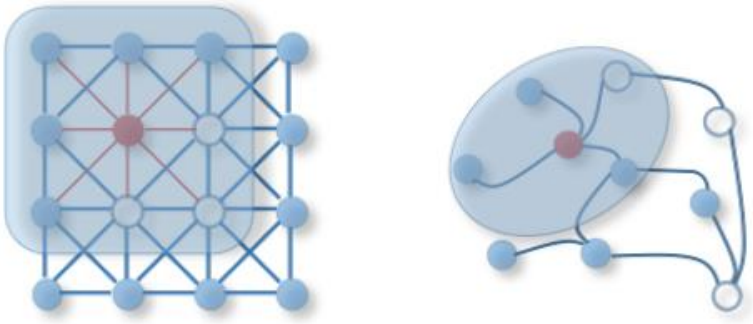
thispersondoesnotexist.com

Graph Neural Networks

- Data is frequently transformed to be treated as a regular structure in the Euclidean domain.
- Numerous types of data are irregular in shape, which rises the need for new approaches to analyze data having a non-Euclidean geometric structure.
- Graphs are an example of a data type whose structure is suitable to represent various complex problems involving interactions between entities.



Source: "Deep learning in color: Towards automated quark/gluon jet discrimination" Komiske et.al



Source: "A Comprehensive Survey on Graph Neural Networks", Wu et.al

Application: Graph Generative Model for Fast Simulation of collider events

This work discusses the potential provided by graph neural networks in high energy physics applications.

The ability of GNNs to learn on isolated hits while disregarding the empty cells surrounding them makes them suitable for sparse detector data.

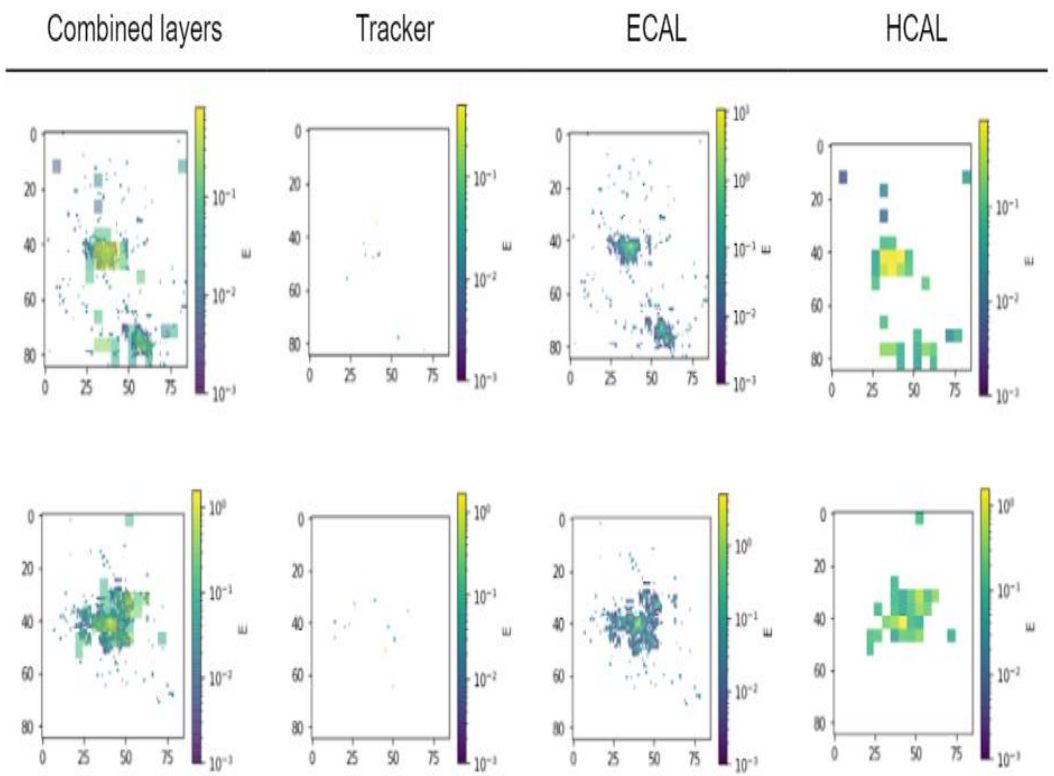
The learning process takes place on the particle hits exclusively !

We propose geometric deep learning as an alternative methodology in particle physics by using it in Fast simulation

Data Description

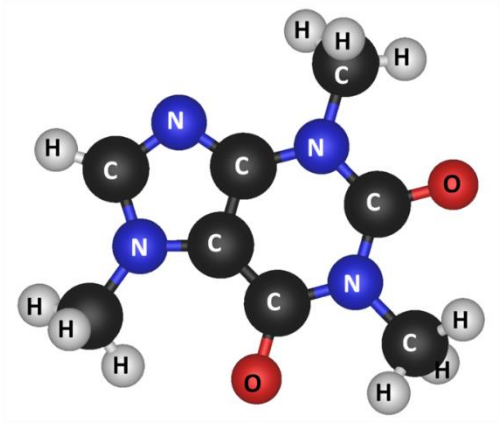
Key ▲	Type	Size	Value
X_jets	float32	(3, 125, 125)	[[[0. 0. 0. ... 0. 0. 0.] [0. 0. 0. ... 0. 0. 0.]

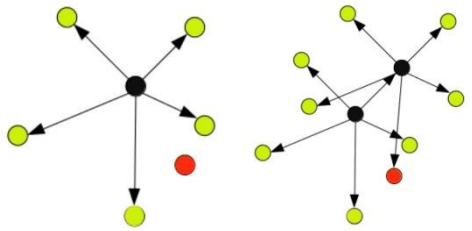
- In this work, we make use of the CMS Open Data release publicly accessible data from the LHC experiments.
- We consider the top quark jets produced using Pythia 6, a program for generating particle collisions events.
- After pre-processing, the data consists of almost 30000 samples of 3x125x125 arrays representing the mesh and the segmentation of 3 detector stages: Tracker, ECAL and HCAL subdetectors, respectively.

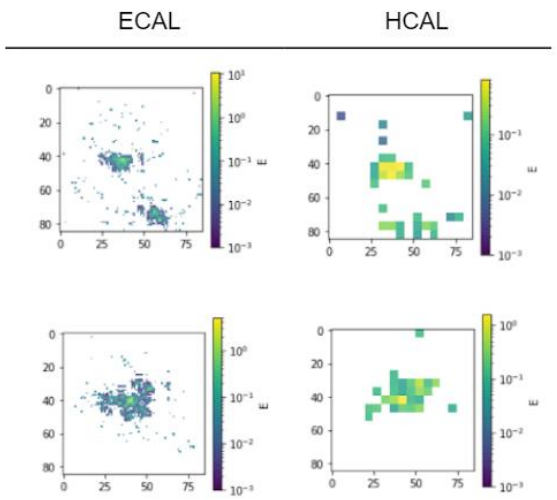


Graph VAE for Jet Reconstruction with Spectral (Mincut) Pooling

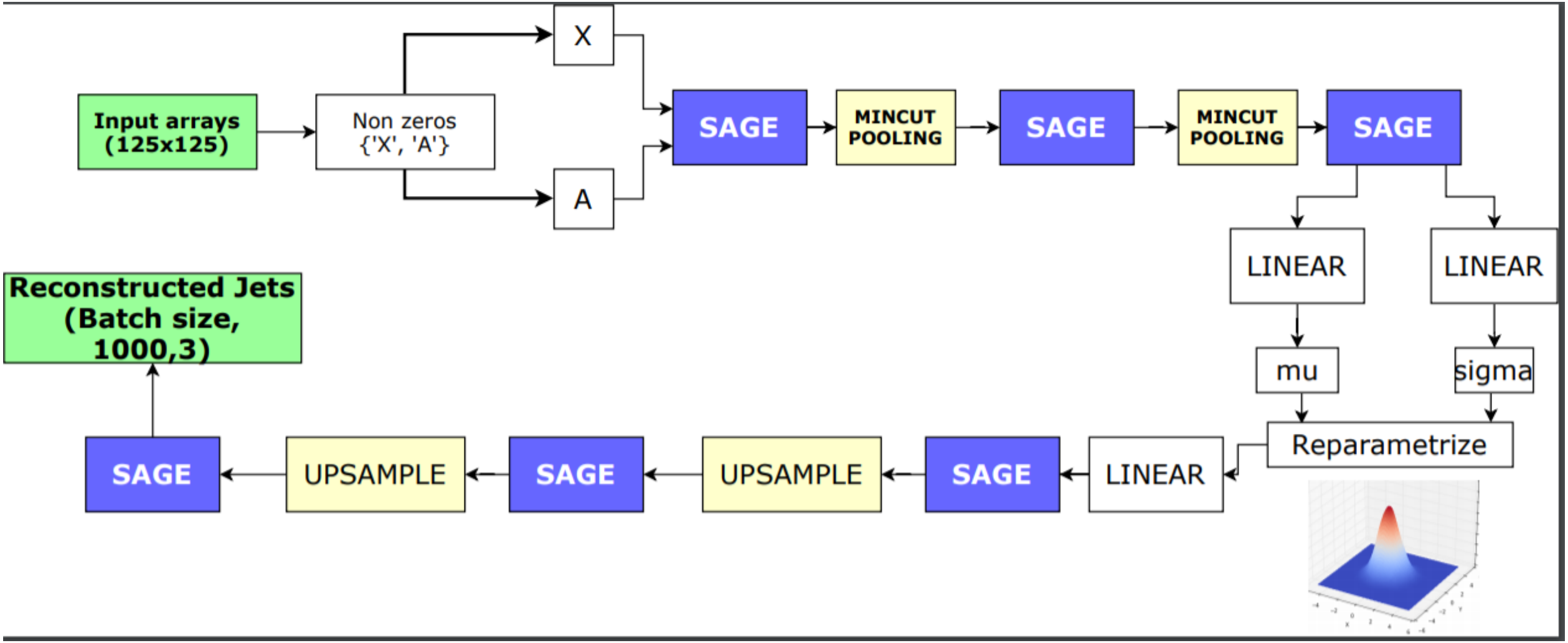
- In contrast to molecular chemistry, jets in particle collisions are not characterized by pre-defined topology.
- We proceed by connecting each node to its k-nearest neighbors based on Euclidean distance given by $\sqrt{(x-x_i)^2+(y-y_i)^2}$ with x_i and y_i referring to this node's coordinates.
- We aim to learn the properties of these jets as graphs in addition to their compressed representation.
- We develop a Graph VAE architecture whose encoder embeds the node features into latent space dimensions through Dense GraphSAGE layers.







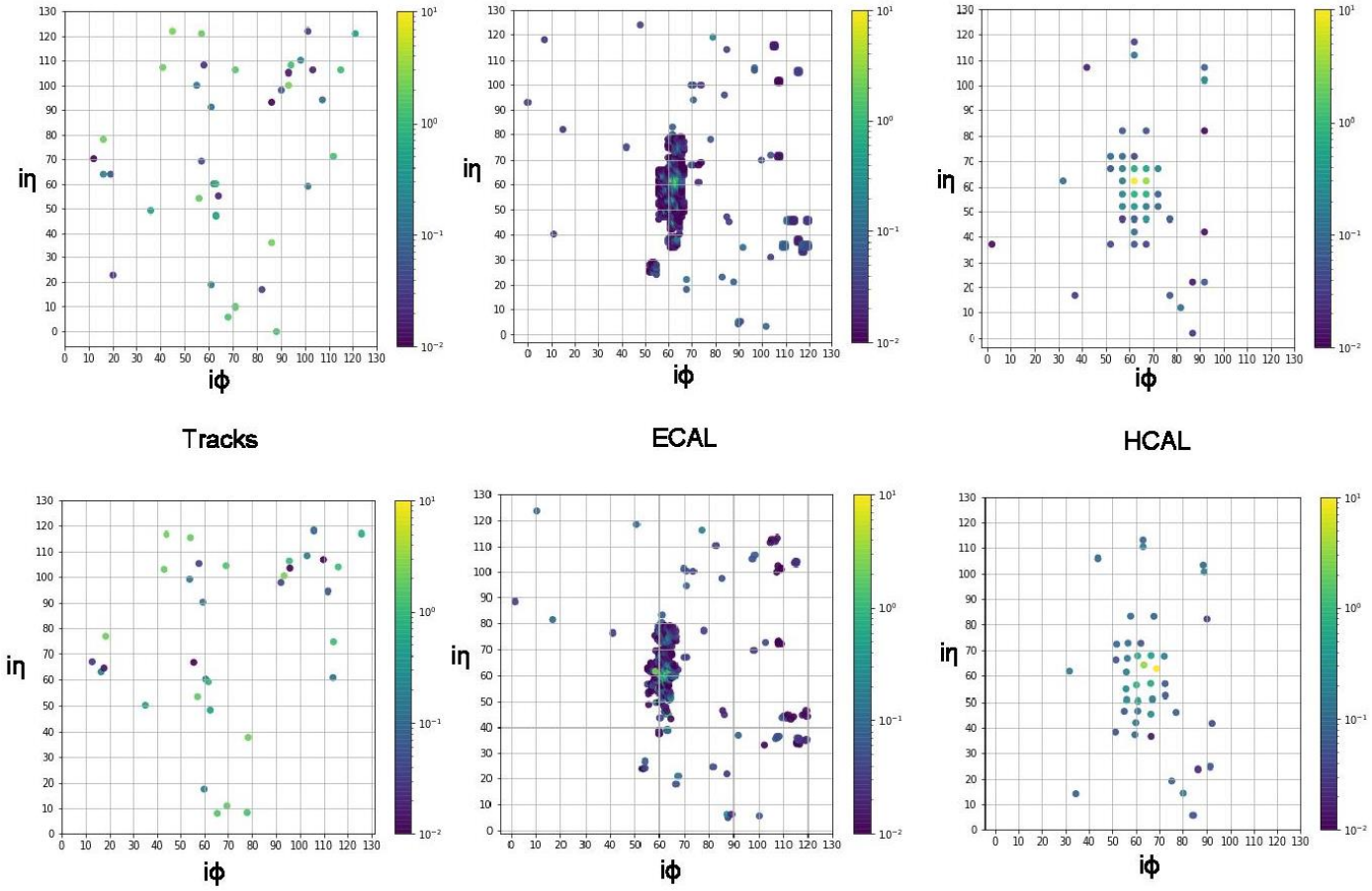
Graph Autoencoder model



- **<u>Loss Function:</u>** Reconstruction Loss is MSE + KL Divergence between latent space distribution and normal distribution.

Results

Reconstruction



GPU Hackathon results

Helmholtz GPU Hackathon – October 2020

- Five days of intensive hands-on events designed to help computational scientists port their applications to GPUs with dedicated mentors experienced in GPU programming and development.

- This event is jointly organized by Helmholtz-Zentrum Dresden-Rossendorf (HZDR) and Jülich Supercomputing Centre (JSC) in association with the Helmholtz Federated IT Services Software Cluster (HIFIS).

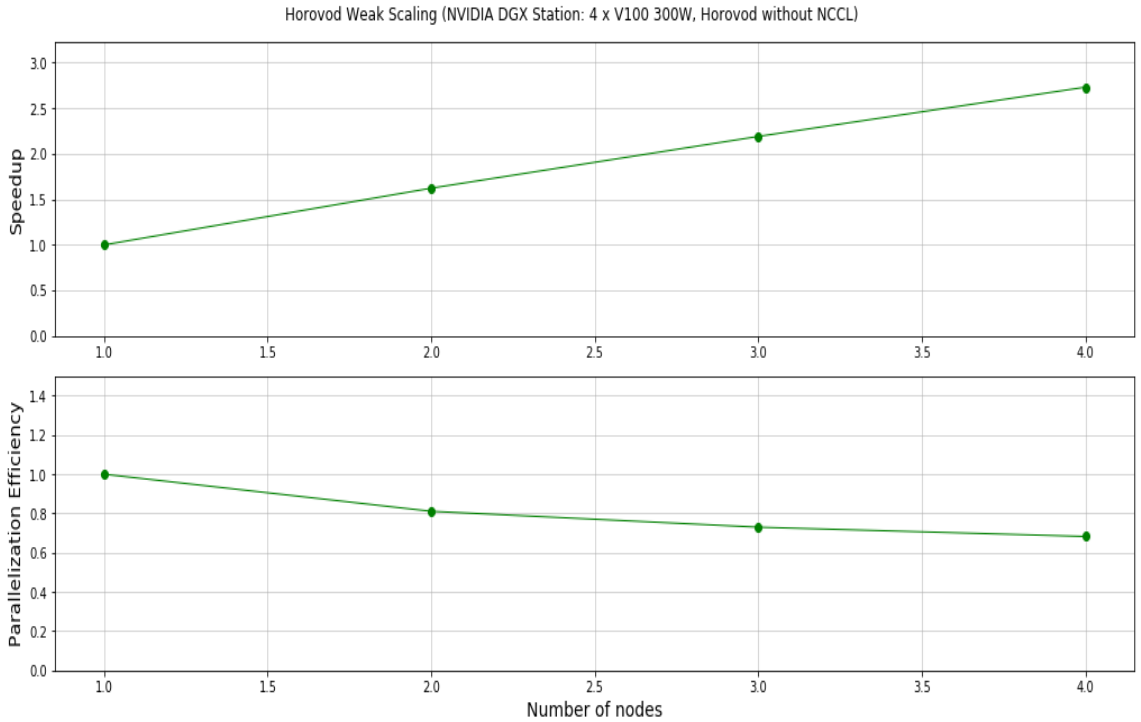
- Special thanks to our team's mentors Christian Hundt and Giuseppe Fiameni as well as Mozhgan Kabiri Chimeh.

https://www.gpuhackathons.org/blog/helmholtz-hackathon-went-virtual

Helmholtz GPU Hackathon

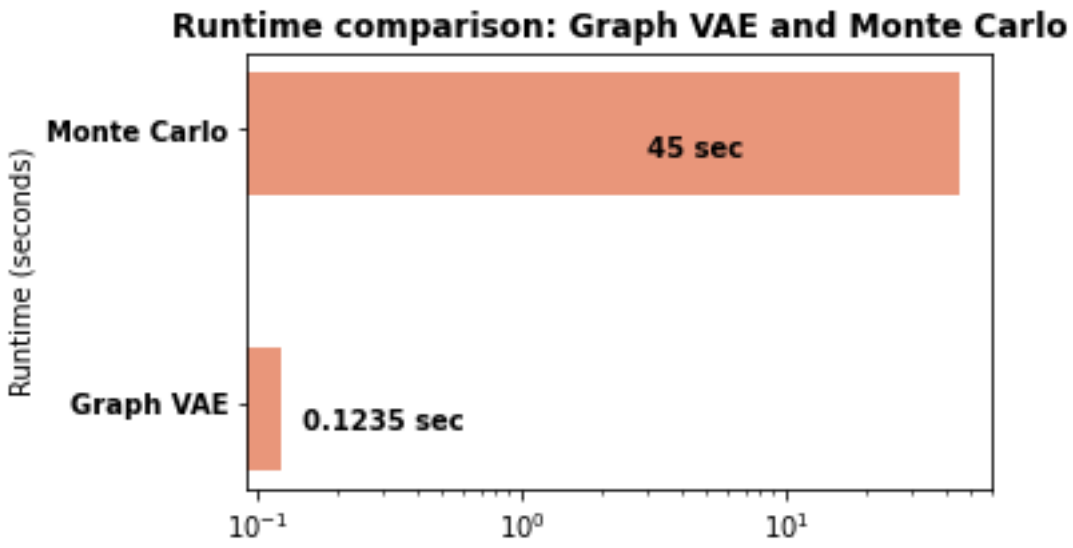
- This hackathon was a crucial part of my thesis work and had the following
  - Code optimization.
  - Porting code to multiple GPUs.
  - Being mentored by NVIDIA experts.
- For this purpose, we apply a profiling run on our code for a few training iterations using the **NVIDIA Nsight Systems's Visual Profiler.**
- When scaling we see an increase in performance as we scale to more GPU devices.
- To calculate the resulting speedup from scaling, we take as reference the Mean Execution Time (MET) resulting from one GPU.
- The speedup for 2 GPUs is given by: $\frac{MET_{singleGPU}}{METperGPU_{using\ N-GPUs}} = \frac{69.34}{\frac{85.48}{2}} = 1.62.$

Horovod Weak Scaling (p in 1, 2, 3, 4; b = 32 * p; 100 iterations * p) NVIDIA DGX Station: 4 x V100 300W, horovod without NCCL)				
Checkpoint	**GPU Processes execution time in seconds of 3200 samples**			
	One	**Two**	**Three**	**Four**
1	71.5	83.9	95.1	100.6
2	64.6	85.7	97.2	102.7
3	69.1	88.2	91.6	100.5
4	69	83.6	94.1	102.4
5	72.5	86	96.8	101.5
Mean Execution Time	**69.34**	**85.48**	**94.96**	**101.54**
Stdev Execution Time	**3.05**	**1.85**	**2.26**	**1.00**
Speedup	**1.00**	**1.62**	**2.19**	**2.73**
Parallelization Efficiency	**1.00**	**0.81**	**0.73**	**0.68**



Thank you

GVAE model timing comparison



- We compare the inference time of that of the conventional Monte Carlo simulations (MC) that approximate PDFs.
- MC takes 45 seconds for the same event batch. Our graph model takes around 0.1 second for the inference, which is over x400 speedup.