Graph Variational autoencoder for fast end to end detector simulation

Dinesh PR

December 13, 2023

Abstract

This work is produced towards partial fulfillment of course requirements for PH 561, Nuclear and particle physics (Fall 2023) at the University of Alabama

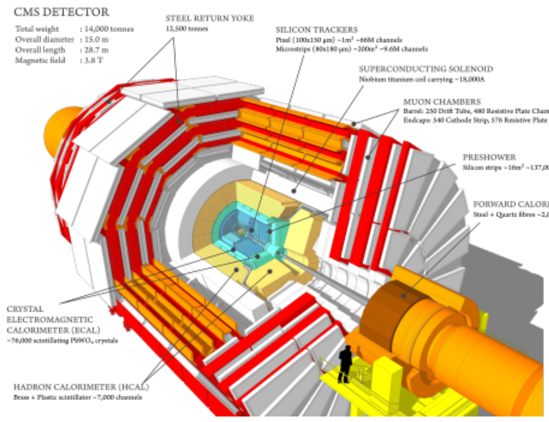

CMS DETECTOR
Total weight : 14,000 tonnes
Overall diameter : 15.0 m
Overall length : 28.7 m
Magnetic field : 3.8 T
STEEL RETURN YOKE
12,500 tonnes
SILICON TRACKERS
Pixel (100x150 μm) ~1m² ~66M channels
Microstrips (80x180 μm) ~200m² ~9.6M channels
SUPERCONDUCTING SOLENOID
Niobium titanium coil carrying ~18,000A
MUON CHAMBERS
Barrel: 250 Drift Tube, 480 Resistive Plate Chambers
Endcaps: 540 Cathode Strip, 576 Resistive Plate Chambers
PRESHOWER
Silicon strips ~16m² ~137,000 channels
FORWARD CALORIMETER
Steel + Quartz fibres ~2,000 Channels
CRYSTAL ELECTROMAGNETIC CALORIMETER (ECAL)
~76,000 scintillating PbWO₄ crystals
HADRON CALORIMETER (HCAL)
Brass + Plastic scintillator ~7,000 channels

Figure 1: Schematics of the CMS detector at LHC, CERN. Source: `https://cms.cern/detector`

0.1 Introduction and Background

In the current work, we build and study a generative model based on graph variational autoencoders to accelerate simulations of boosted top quark production in the Large Hadron Collider (LHC)[33]. While it trains on dataset obtained from existing Monte Carlo-based generation and detector simulation models, we aim to accelerate re-sampling the same without compromising on the accuracy and fidelity. Generative and reconstruction capabilities of variational autoencoders means they can train on simulated datasets from known physical processes and detect anomalies due to unknown or new physical processes behind actual detector data.

0.1.1 CMS detector and collision "images"

The CMS detector[10] in the LHC is arranged in cylindrical sections of different types of subdetectors (see fig 1), with the central axis along the line of collision between proton beams. This includes both barrel and endcap (see figure) sections of each subdetector. We look at 3 particular subdetectors as relevant to this study- the inner tracking system (Tracker), the electromagnetic calorimeter (ECAL) and the hadronic calorimeter (HCAL)[3].

Tracker is the innermost series of finely segmented silicon wafers that (non-destructively) detect positional tracks left by particles formed from a collision beam, deciphering practically nothing about their energies. The "pixels" in the Tracker are labelled using (z, ϕ) and (ρ, ϕ) for the barrel and endcap sections respectively- where z denotes length along the axis, ρ denotes the perpendicular radius and ϕ denotes the azimuthal angle.

Surrounding the Tracker is the ECAL subdetector which captures the incoming photons and electrons, and measures their energies using scintillating lead tungstate crystals. In the barrel section, it is segmented by pseudorapidity

($i\eta_{EB}$) and azimuthal angle ($i\phi_{EB}$) and spans within $|\eta| < 1.479$ making it a 170×360 grid of "pixels". The endcap is labelled using (iX, iY) as a Cartesian grid of pixels arranged in a circle. Psudorapidity is an alternative to the polar angle (θ) frequently used in spherical coordinates and they're related by:

$$\eta = -\ln\left[\tan\frac{\theta}{2}\right]$$

The HCAL subdetector encloses the ECAL and measures the energies of hadronic particles (mostly charged pions and kaons) using scintillating brass towers. The barrel section encloses the pseudorapidity range $|\eta| < 3$. The pixels in the barrel section of HCAL produce an image that's more coarsely segmented with roughly 1 HCAL pixel per 5 ECAL pixels along both the $i\phi$ and $i\eta$ directions.

0.1.2 Event generation for the Large Hadrom Collider

Quantum field theory[36, 35, 31] is the mathematical framework based on which theoretical as well as numerical computations of physical quantities observed in high energy experiments are performed. In particular, the values of Green's functions or S matrices determine the various scattering *cross sections* for the final states of particles observed in different detectors. The simulation pipeline can be divided into several steps[7].

Hard scattering The *hard scattering* component probes the smallest length scales around the collision center. The cross sections are usually determined using ab inito (first principle), perturbative QFT calculations at the highest order of accuracy feasible in terms of the coupling constants. Terms of the perturbation series can be represented visually using Feynman diagrams[16] and their values are often plagued by problems with IR divergences in loop integrals which often cancel analytically with tree-level counterparts. If analytic solutions are not available, Monte Carlo-based simulations[20, 26, 2] are used for which there are ever-increasing quests for improvements in efficiency and numerical stability.

Secondary interactions and decays Since detectors are present at much larger distances from the center, there can be additional collisions and decays beyond the hard scattering which need to simulated accurately. Factorization theorems[12, 11] give rise to correlations between the momenta of partonic (quarks/gluons) currents of different flavours, which are encoded semi-classically as integrals of parton distribution functions (PDF). Their time-evolution is modelled using Monte Carlo-based algorithms like parton showers [27]and dipole showers[21].

Numerical libraries and open simulated datasets Many generic software libraries that simulate parts of the event generation chain we briefly summarized are available. Most popular general-purpose packages available for the same include SHERPA[17], HERWIG[13] and PYTHIA[32]. They're run in combination with software that simulate the behavior of particular detectors (in say, CMS) like GEANT[1]. While the Monte Carlo-based generators are highly accurate, they often scale poorly with the collision energy, luminosity and detector count. Faster detector simulation libraries like DELPHES[15] tend to make approximations that trade off accuracy for speed. Full simulation of collider experiments using theory-driven collision event generators and experimentally optimized detector simulators is referred to as end-to-end event generation.

0.1.3 Generative machine learning and Graph variational autoencoders

Generative networks[28] are machine learning tools used to sample from a learned data distribution of a training dataset. While Monte Carlo generators asymptotically (in the limit of infinite samples) converge to the actual distribution for averaged quantities, they can suffer from large autocorrelations between successive samples and poor accuracies for non-averaged quantities for practically finite number of samples[9]. On the other hand, most generative ML models sample with no autocorrelations but their asymptotic closeness to the target distribution depends on the model architecture, training parameters, loss function and the quality of training data. Most popular classes of generative networks include variational autoencoders[22] (VAE), generative adversarial networks[14] (GAN) and normalizing flows (NF)[29]. VAEs and GANs generally offer highly expressive models and have already been used in many high energy physics applications including end to end event generation[6]. NFs are known to be less expressive but allow us to explicitly compute likelihoods of different samples and can be inverted for reverse simulations, which have been exploited in many HEP applications.

Variational autoencoders VAEs use neural networks to *encode* or compress high dimensional input data to lower dimensional latent space which is also the parameter space for a generic probability distribution (like a Gaussian) and these samples are then *decoded* by other neural networks back to the original input space. They are typically trained to minimize a distance between the input and the (decoded) output data, like mean squared error (MSE), along with a prior likelihood term into an evidence lower bound (ELBO) estimator. In our work, we use graph-based neural networks for encoders and decoders in a VAE, an idea which has already been explored[19].

Graph convolution and pooling operations In the geometric deep learning framework[5], we tend to use operations that preserve the structure and symmetries of the input data. In the case of graph data, we'd like to preserve the neighborhood of each node and require that the output be invariant to permutations within the neighborhoods. Spectral convolutions[23] were initially developed as compatible operations on graph data but they tend to scale poorly with the number of nodes and edges. Message passing layers like GCN [24]and SAGE[18] were developed to compute convolutions in polynomial time on the size of input data. We convert the sparse detector image by taking the non-zero pixels as nodes and putting edges between k nearest neighbors (knn) of every node. We use SAGE convolution layers, which is ideal for aggregating information from the neighborhood of every node when the edge count ($\sim k$)is very small. The operation is mathematically described using

$$\boldsymbol{x}'_i = \boldsymbol{W}_1 \boldsymbol{x}_i + \boldsymbol{W}_2 . \mathrm{agg}_{j \in \mathcal{N}(i)} \boldsymbol{x}_j$$

where $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ are weights of the layer (which are optimized during training) and 'agg' is an aggregating operation like mean or max.

0.2 Data and Pre-processing

In this work, we focus on the quark-gluon shower dataset for the CMS detector, available on the CERN open data portal[8] where the event generation of stable particle arriving at the detectors were done using PYTHIA. GEANT4 is used for detector simulation, digitization, image generation and reconstruction. It contains 125×125 images corresponding to the Tracker, ECAL and HCAL subdetector, labelled 0 for quark jet and 1 for gluon jet responses. Out of roughly 130K images (equally split between quark and gluon jets), we train a GraphVAE quark jet ECAL images alone, for simplicity as well as benchmarking the model against corresponding images of gluon jets- investigating the utility of GraphVAE for detecting anomalies in input data. To convert the images to *graph* format, we pool all the hits with energies greater that 10^{-3}[1]compress them to form 3-feature nodes (pixel positions and energies). This considerably reduces the size of the dataset file in the computer's memory- from ≈ 600 Mb to ≈ 100 Mb, in the HDF5 dataset format. The positions and energies are further normalized to values in the $(0, 1)$ interval to finally prepare the training dataset. Training and test datasets consist 50,000 and 10,000 of the graph-formatted ECAL images respectively. We compiled another test dataset of 10,000 gluon images for benchmarking.

0.3 Results

Pytorch deep learning framework[30] coupled with Pytorch geometric library (which specializes in operations on graphs) were used to implement the Graph-VAE model used in this work. The encoder network consists of 3 SAGE convolution layer with 2 MinCut Pool layers[4] sandwiched in between, gradually reducing the dimensionality of the latent space to 20 from the input space dimensionality of 400. The decoder network exactly mirrors this operation with similar

[1]An upper bound of 400 hits was chosen, sorted according to energy in descending order

Dataset (model)	Hit MSE	Energy MSE
Quark training (flat)	0.4043	0.0013
Quark training (periodic)	0.0917	0.0004
Quark test (flat)	0.3786	0.0013
Quark test (periodic)	0.0864	0.0004
Gluon test (flat)	0.4144	0.0014
Gluon test (periodic)	0.0942	0.0005

Table 1: This table describes the Mean Square Error (MSE) metric of hit and energy reconstruction by models trained on flat/periodic geometries on different datasets (lower values are better)

graph operations that upscale the dimensions back to 400. Degree normalization[4] is applied to the adjacency matrix between every upscaling/downscaling operation and masking out padded nodes has been carefully applied to ensure there are no "leaks" in the graph operations. Since this is a toy model, fairly modest layer widths of upto 64 were used and it remains to be seen how scaling to bigger neural networks affects the results.

Mean squared error (MSE) loss between the input and output features (both hits and their energies) were used along with a KL-divergence loss for the reparameterization mean and variance, as well as a regularization term for optimizing cluster assignments while downscaling/upscaling. ADAM optimizer has been used to update weights via gradient descent during training with a learning rate of 10^{-3}. A publicly accessible codebase of our model can be found here: `github.com/dinesh110598/Quark_Gluon_Data/main` and all data/figures in this work are reproducible.

0.3.1 Cylindrical geometry of image data

Assigning the position features values between 0 and 1 imposes a "flat" or *Cartesian* geometry for the detector hit images, while the $i\phi$ direction horizontal to the images is circular or periodic which can lead to loss of information at the ends near $\phi = 0, 2\pi$ during training. To take this periodic nature into account, we translate the x coordinates of some hits by ± 1 depending on the position of predicted hit positions[2]. So we train our model using both a naive flat geometry loss and a periodic one to compare their MSE metrics on both training and test datasets in Table 1. Training using periodic geometry clearly outperforms the flat geometry counterpart of the Graph VAE model in the given metric across different datasets as well as for both hit positions and the corresponding energies. Figures 2 and 3 compare the true and reconstructed "images" of randomly chosen samples from each dataset. It's generally hard to visually determine the performance of the flat model against the periodic one from looking through the plots.

[2]The train.py file in our code repository contains a precise implementation

0.3.2 Gluon jet data as anomalies

An earlier work[3] demonstrates classifying images of quark and gluon jets using convolutional neural networks where the ML model trains to discriminate between different labels. In the current work, we attempt to train a variational autoencoder to reconstruct samples of quark jets alone, so it has no prior information about gluon jet samples. Hence, we expect the model to perform worse in terms of various metrics when queried to reconstruct gluon jets, compared to quark jets and thus be treated as anomalies beyond a threshold value of these metrics. Referring to table 1 seems to indicate similar metrics for both flat and periodic trained Graph VAE models for quark and gluon jets. In a closer look, training dataset has worse MSE metric compared to the gluon test dataset, which are in turn marginally higher than the quark test dataset for both hits and energies. Visual comparisons between figures 2 and 3 offer no insight either.

0.4 Conclusion

Variational autoencoders combined with graph neural network architectures generally offer an efficient solution to generate and reconstruct sparse datasets like detector hits in the LHC. From the results, it seems we are headed in the right direction with the toy model consisting of modestly sized neural networks. The image plots in figures 2 and 3 do suggest scope for improvement both in terms of reconstruction quality and anomaly detection. In particular, bigger hidden channels, latent space and a deeper architecture with residual connections could offer better approximation capacities to the hit positions and energies. The mathematical similarities between graph convolution and transformer architectures[34] used in natural language processing can provide insights into improving upon this work as well. Additional metrics like Earth Mover Distance[25] can throw more light in interpreting results of our models compared to MSE since the former is a natural energy-based metric for distances between particle configurations.

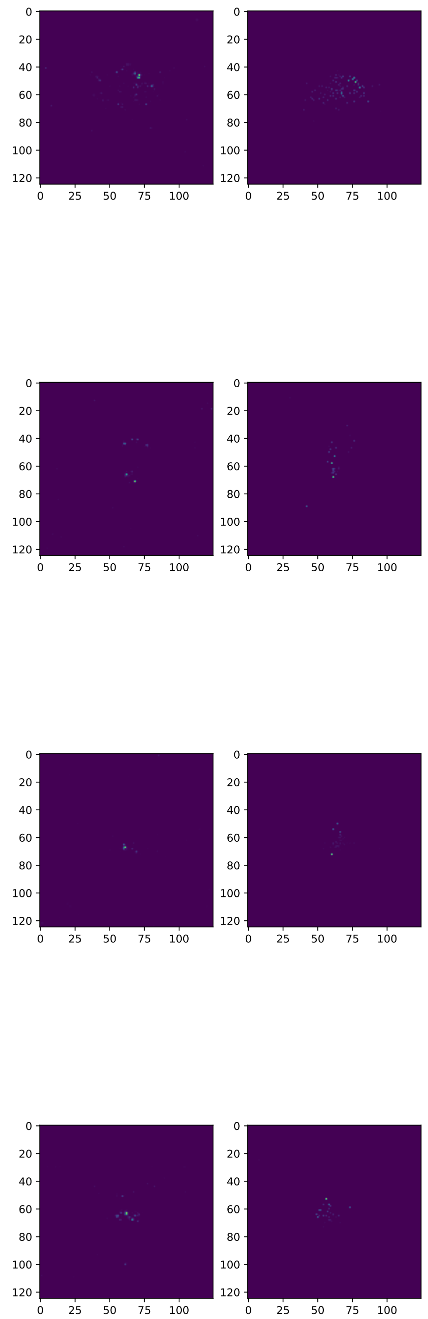
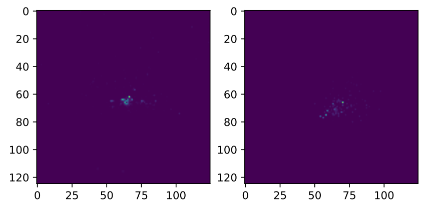


Figure 2: Original and reconstructed samples from different quark jet datasets are plotted side-by-side presented in the same (row wise) order as initial four rows of table 1



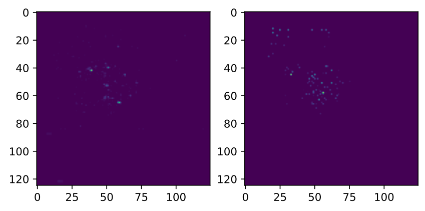


Figure 3: Original and reconstructed samples from different gluon jet datasets are plotted side-by-side presented in the same (row wise) order as final two rows of table 1

Bibliography

[1] Sea Agostinelli et al. “GEANT4, a simulation toolkit”. In: *Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 506.3 (2003), pp. 250–303.

[2] Johan Alwall et al. “MadGraph 5: going beyond”. In: *Journal of High Energy Physics* 2011.6 (2011), pp. 1–40.

[3] M Andrews et al. “End-to-end event classification of high-energy physics data”. In: *Journal of Physics: Conference Series*. Vol. 1085. 4. IOP Publishing. 2018, p. 042022.

[4] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. “Spectral clustering with graph neural networks for graph pooling”. In: *International conference on machine learning*. PMLR. 2020, pp. 874–883.

[5] Michael M Bronstein et al. “Geometric deep learning: going beyond euclidean data”. In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.

[6] Anja Butter et al. “Machine learning and LHC event generation”. In: *SciPost Physics* 14.4 (2023), p. 079.

[7] JM Campbell et al. “Event generators for high-energy physics experiments”. In: *arXiv preprint arXiv:2203.11110* (2022).

[8] *CERN Open data portal*. URL: `http://opendata.cern.ch`.

[9] Siddhartha Chib and Edward Greenberg. “Understanding the metropolishastings algorithm”. In: *The american statistician* 49.4 (1995), pp. 327–335.

[10] CMS Collaboration et al. “The CMS experiment at the CERN LHC”. In: *Jinst* 3 (2008), S08004.

[11] John Collins. *Foundations of perturbative QCD*. Cambridge University Press, 2011.

[12] John C Collins, Davison E Soper, and George Sterman. “Factorization of hard processes in QCD”. In: *Perturbative QCD*. World Scientific, 1989, pp. 1–91.

[13] Gennaro Corcella et al. “HERWIG 6: an event generator for hadron emission reactions with interfering gluons (including supersymmetric processes)”. In: *Journal of High Energy Physics* 2001.01 (2001), p. 010.

[14] Antonia Creswell et al. “Generative adversarial networks: An overview”. In: *IEEE signal processing magazine* 35.1 (2018), pp. 53–65.

[15] J De Favereau et al. “DELPHES 3: a modular framework for fast simulation of a generic collider experiment”. In: *Journal of High Energy Physics* 2014.2 (2014), pp. 1–26.

[16] Richard Phillips Feynman. *Selected papers of Richard Feynman: with commentary*. Vol. 27. World Scientific, 2000.

[17] Tanju Gleisberg et al. “Event generation with SHERPA 1.1”. In: *Journal of High Energy Physics* 2009.02 (2009), p. 007.

[18] Will Hamilton, Zhitao Ying, and Jure Leskovec. “Inductive representation learning on large graphs”. In: *Advances in neural information processing systems* 30 (2017).

[19] Ali Hariri, Darya Dyachkova, and Sergei Gleyzer. “Graph generative models for fast detector simulations in high energy physics”. In: *arXiv preprint arXiv:2104.01725* (2021).

[20] Aggeliki Kanaki and Costas G Papadopoulos. “HELAC: a package to compute electroweak helicity amplitudes”. In: *Computer physics communications* 132.3 (2000), pp. 306–315.

[21] Hamid Kharraziha and Leif Lönnblad. “The linked dipole chain Monte Carlo”. In: *Journal of High Energy Physics* 1998.03 (1998), p. 006.

[22] Diederik P Kingma, Max Welling, et al. “An introduction to variational autoencoders”. In: *Foundations and Trends® in Machine Learning* 12.4 (2019), pp. 307–392.

[23] Thomas N Kipf and Max Welling. “Semi-supervised classification with graph convolutional networks”. In: *arXiv preprint arXiv:1609.02907* (2016).

[24] Thomas N Kipf and Max Welling. “Semi-supervised classification with graph convolutional networks”. In: *arXiv preprint arXiv:1609.02907* (2016).

[25] Patrick T Komiske, Eric M Metodiev, and Jesse Thaler. “Metric space of collider events”. In: *Physical review letters* 123.4 (2019), p. 041801.

[26] Frank Krauss, Ralf Kuhn, and Gerhard Soff. “AMEGIC++ 1.0, a matrix element generator in C++”. In: *Journal of High Energy Physics* 2002.02 (2002), p. 044.

[27] G Marchesini and Bryan R Webber. “Monte Carlo simulation of general hard processes with coherent QCD radiation”. In: *Nuclear Physics B* 310.3-4 (1988), pp. 461–526.

[28] Achraf Oussidi and Azeddine Elhassouny. “Deep generative models: Survey”. In: *2018 International conference on intelligent systems and computer vision (ISCV)*. IEEE. 2018, pp. 1–8.

[29] George Papamakarios et al. "Normalizing flows for probabilistic modeling and inference". In: *The Journal of Machine Learning Research* 22.1 (2021), pp. 2617–2680.

[30] Adam Paszke et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019).

[31] Michael E Peskin. *An introduction to quantum field theory.* CRC press, 2018.

[32] Torbjörn Sjöstrand et al. "High-energy-physics event generation with PYTHIA 6.1". In: *Computer Physics Communications* 135.2 (2001), pp. 238–259.

[33] *The Large Hadron Collider at CERN.* URL: `https://home.web.cern.ch/science/accelerators/large-hadron-collider`.

[34] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[35] Steven Weinberg. *The quantum theory of fields.* Vol. 2. Cambridge university press, 1995.

[36] Anthony Zee. *Quantum field theory in a nutshell.* Vol. 7. Princeton university press, 2010.