Multi-channel particle physics detector simulation with graph variational autoencoders

Anonymous authors
Paper under double-blind review

Abstract

Accurate and fast simulation of particle physics processes is crucial for the high-energy physics community. Simulating particle interactions with the detector is both time consuming and computationally expensive. With its proton-proton collision energy of 13 TeV, the Large Hadron Collider is uniquely positioned to detect and measure the rare phenomena that can shape our knowledge of new interactions. The High-Luminosity Large Hadron Collider (HL-LHC) upgrade will put a significant strain on the computing infrastructure and budget due to increased event rate and levels of pile-up. Simulation of high-energy physics collisions needs to be significantly faster without sacrificing the physics accuracy. Machine learning approaches can offer faster solutions, while maintaining a high level of fidelity. We introduce a graph generative model that provides effective reconstruction of LHC events on the level of calorimeter deposits and tracks, paving the way for full detector level fast simulation.

1 Introduction

Simulations of particle interactions are crucial for particle physics research. The Large Hadron Collider is delivering the highest energy proton-proton collisions ever recorded in the laboratory, permitting a detailed exploration of elementary particle physics. It is uniquely positioned to detect and measure the rare phenomena that can shape our knowledge of new interactions and possibly resolve the present tensions of the Standard Model. LHC experiments have already observed the long-sought after Higgs boson CMS Collaboration (2012); ATLAS Collaboration (2012), and it is hoped that they will further reveal evidence for new physics beyond the Standard Model.

After the next LHC run and the ensuing shutdown, the LHC will begin its 'High-Luminosity' (HL-LHC) operations around 2027 Apollinari et al. (2015), targeting a five-fold increase in yearly data collection, with an additional ten years of operation. Advanced computational paradigms, including advanced machine learning techniques and dedicated hardware that accelerates their application for detector simulation and event reconstruction, will be necessary to attain the main physics goals of the HL-LHC Albertsson et al. (2018) due to significant challenges posed by increased levels of pile-up and the rarity of sought-after signals.

Probabilistically, the reconstruction of collision events is modeled using the equation:

$$p(\text{r-particles}|\theta) = \int R(\text{r-particles}|\text{particles}) H(\text{particles}|\text{partons}) \times P(\text{partons}|\theta)\, d\text{particles}\, d\text{partons} \tag{1}$$

where P is the probability density of observing a set of reconstructed objects given a point in the parameter space, H refers to the hadronization process where mapping from the parton to the particle level occurs and R(particles) is the detector response Gleyzer et al. (2016). The latter is estimated using Monte Carlo-based full simulation or parametric methods. This poses a computational bottleneck for current simulation frameworks that model collision events using Monte Carlo methods. The current forward simulation models Brun et al. (1978) are difficult to parallelize and do not scale well with the rising computational complexity. Parametrized models such as de Favereau et al. (2014) are faster but suffer from a loss in fidelity compared to full simulation which limits their utility.

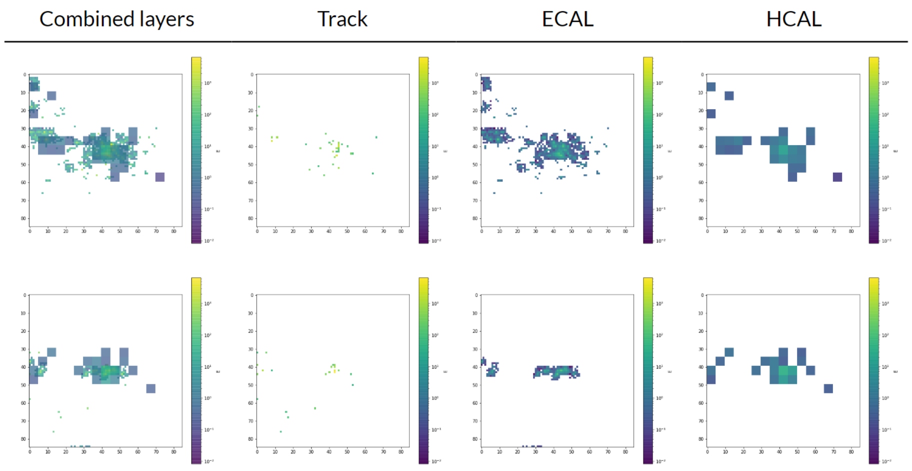


Figure 1: The visualization of the energy deposits in ECAL and HCAL calorimeter layers for two distinct *ttbar* events (top and bottom). The tracks layer shows the tracks projected to the ECAL surface. The leftmost image is the combined one from all the three layers.

2 Data and Pre-Processing

For the study, we use a representative sample of top quark pair events generated with Pythia 6 and available on the CERN Open Data Portal cms. The input data is processed by following the recipe in Andrews et al. (2020). The input data contains a total of 30000 samples, each containing 3 channels: Tracks, Electromagnetic Calorimeter (ECAL) and Hadronic Calorimeter (HCAL), respectively. To visualize the events in different sub-detectors, we display them on a mesh with 85 x 85 segmentation (Fig. 1), shown at ECAL resolution.

We pre-process the data by selecting the non-zero hit locations within the array, providing their respective x and y locations as per the calorimeter segmentation. Afterwards, we concatenate the x,y locations with their corresponding reconstructed hit energy at that location. At this stage, each sample has the shape Nx3 where N is the number of non-zero hits within the detector. In the next section we show that N is also the number of nodes within one graph representing a jet.

3 Graph Representation of Events

A graph is denoted by $G = (V, E, A)$ where V is the set of vertices composing the graph, E the set of edges connecting these vertices, and A being an N x N adjacency matrix describing the relationship between the nodes, where N is the total number of nodes in the graph. Associated with each vertex V is a set of features describing it.

We consider particle hits within a detector to be interconnected nodes in a graph. Therefore, each event is characterized by a set of nodes whose features include x-y locations in 2D space in addition to their energies. We use the k-nearest neighbours algorithm in 2D space to connect the nodes, i.e each node is connected by an edge to k-nodes that are closest in terms of 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. To learn the graph properties of these showers we develop a Graph VAE architecture whose encoder embeds the node features into latent space dimensions.

After defining the graph topology and node features, we also need to address the graph embeddings. We use GraphSAGE Hamilton et al. (2017), an inductive learning framework that assumes nodes within the same neighbourhood to have similar embeddings and proceed by aggregating information from higher-level neighbourhoods. Hence, we use dense GraphSAGE layers in a graph variational encoder-decoder model that learns to reconstruct graphs from a learned compressed representation in latent space. The latter is obtained at the level of the encoder by means of spectral clustering of the graph's nodes with similar latent features. More specifically, the graph is compressed using

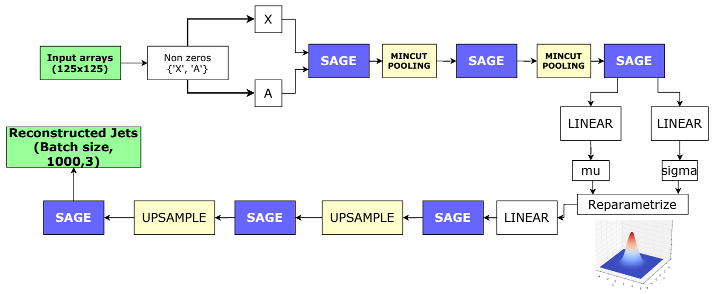


Figure 2: Model architecture of the Graph Variational Autoencoder showing GraphSAGE layers and pooling blocks

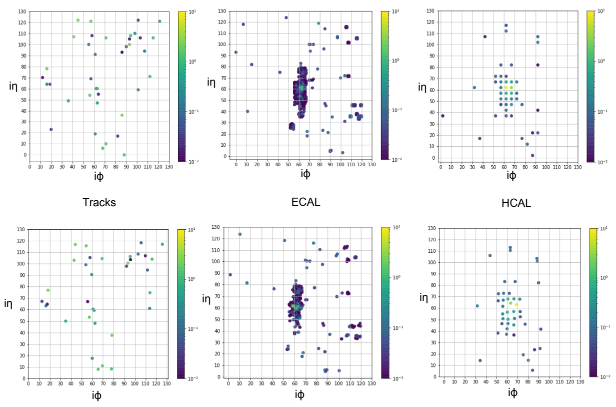


Figure 3: Original simulated top quark initiated jet (left) compared to the GVAE-reconstructed jet (bottom) in each of the three channels. The energy range is log-scaled for better visualization.

the MinCut pooling technique inspired by Bianchi et al. (2010). Finally, the decoder upsamples the feature and adjacency matrices as follows:

$$X^{rec} = S\,X^{Pooled};\, A^{rec} = S\,A^{Pooled}S^{T} \tag{2}$$

where S is a learned cluster assignment matrix similar to the one defined in Bianchi et al. (2010). A pictorial representation of our model is given in Figure 2.

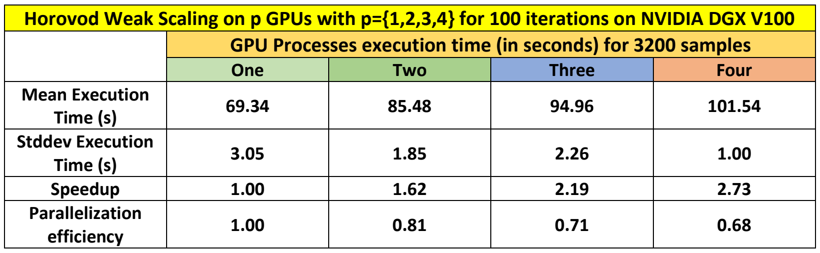
4 Results

Our Graph Variational Autoencoder was trained on a Tesla T4 GPU on Google Colab Bisong (2019) to reconstruct top quark-initiated jets. A clear visualization of our results is shown in Figure 3, where the fully-simulated events are compared to their corresponding GVAE reconstruction. Our results indicate a high-level of agreement in 2D hit distributions and their respective energy scales. Finally, several trial runs showed an average inference time of 0.13088 seconds for GVAE reconstruction on a batch size of 64, orders of magnitude below typical full reconstruction times for simulation of top events, while maintaining a very high level of fidelity.

4.1 Multi-GPU Scaling

We next investigate how the GVAE model scales on multiple GPU devices. We first profile the code to monitor the process load taking place in CPUs and GPUs. To avoid a potential data-loading bottleneck, we opted to generate batches of graphs on the spot in the CPU prior to sending them to the training model on the GPU. This change has reduced the training time by 50%. Having trained on a single Tesla V100 GPU as a baseline, we proceeded by scaling the model using the Horovod library Sergeev & Del Balso (2018), a deep learning library for distributed training of deep learning models.

We train our GVAE model on a cluster using Volta V100 GPUs having 16 GB of RAM. Following profiling and code optimization for enhanced CPU performance, we scale the training on multiple GPUs using the Horovod library. To get a baseline performance, we compare the results to the training on a single GPU with a batch size of 32 for 100 iterations, i.e a total of 3200 graph samples (Figure 4).



Horovod Weak Scaling on p GPUs with p={1,2,3,4} for 100 iterations on NVIDIA DGX V100				
	GPU Processes execution time (in seconds) for 3200 samples			
	One	Two	Three	Four
Mean Execution Time (s)	69.34	85.48	94.96	101.54
Stddev Execution Time (s)	3.05	1.85	2.26	1.00
Speedup	1.00	1.62	2.19	2.73
Parallelization efficiency	1.00	0.81	0.71	0.68

Figure 4: Table showing scaling efficiency

Throughout scaling we notice an increase in the performance with an increase in the GPU devices used. To calculate the resulting speedup from scaling, we take as reference the Mean Execution Time (MET) resulting from training on one GPU. For $N_{GPUs} = 2$, the mean execution time is 85.48 seconds. Bearing in mind that we are training twice the amount of data overall, we get MET per GPU = 42.74 seconds. The resulting speedup for $N_{GPUs} = 2$ is given by

$$\frac{MET_{singleGPU}}{MET perGPUusingN_{GPUs}} = \frac{69.34}{42.74} = 1.62 \tag{3}$$

Applying the same speedup calculation, going from 2 to 4 GPUs, we get speedups of 1.62, 2.19 and 2.73, respectively. Therefore, we conclude that our model scales well on multiple GPUs using Horovod, which is useful for future work and similar graph generative models. However, there is still some loss in scaling performance due to parallelization efficiency that can be explained by an inefficient communication between the GPUs once parallel batch training has been done. This issue can be overcome by programming the kernel directly. Further optimization could be deferred to future work.

5 Conclusion

In this proof-of-concept work we shed light on the potential of graph-based architectures for learning the representation of high-energy collision events. Graph neural networks tackle the issue of data sparsity in particle detectors by allowing the model to directly learn from the hits deposited in the detector. Through spatial convolution our model was able to learn the interactions between hits in the detector and we sequentially compressed it by means of mincut pooling to preserve the most representative nodes in latent space. Finally, a trained decoder can upsample the compressed vectors to the original reconstruction, leading to a proof-of-concept simulator. We additionally benchmarked our model on single and multiple gpus, demonstrating low latency times and efficient multi-gpu scaling performance.

Acknowledgments

We would like to thank Michael Andrews, Bjorn Burkle, Christian Hundt and Giuseppe Fiameni for useful discussions. We thank them for their expertise and support. Ali Hariri was a participant in the Google Summer of Code 2020 Program. We additionally thank NVIDIA, Helmholtz Association and Mozhgan Kabiri Chimeh for support during a gpu hackathon that enabled us to study multi-gpu scaling.

References

CERN Open Data Portal. http://opendata.cern.ch.

K. Albertsson et al. Machine Learning in High Energy Physics Community White Paper. *arXiv e-prints*, Jul 2018.

M. Andrews, M. Paulini, S. Gleyzer, and B. Poczos. End-to-end physics event classification with the cms open data: Applying image-based deep learning on detector data to directly classify collision events at the lhc. *Computing and Software for Big Science*, 4(1):1–14, 2020.

G Apollinari, I Béjar Alonso, O Brüning, M Lamont, and L Rossi. High-Luminosity Large Hadron Collider (HL-LHC) : Preliminary Design Report. *CERN-2015-005, FERMILAB-DESIGN-2015-02*, 2015. doi: 10.5170/CERN-2015-005,10.2172/1365580.

ATLAS Collaboration. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Physics Letters B*, 716(1):1–29, Sep 2012. doi: 10.1016/j.physletb.2012.08.020.

F. Bianchi, D. Grattarola, and C. Alippi. Spectral clustering with graph neural networks for graph pooling. *Proceedings of Machine Learning Research*, 2010.

Ekaba Bisong. *Google Colaboratory*, pp. 59–64. 09 2019. ISBN 978-1-4842-4469-2. doi: 10.1007/978-1-4842-4470-8_7.

R. Brun, R. Hagelberg, M. Hansroul, and J. Lassalle. Simulation program for particle physics experiments, geant : user guide and reference manual. *Report Number CERN-DD-78-2*, 1978.

CMS Collaboration. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Physics Letters B*, 716(1):30–61, Sep 2012. doi: 10.1016/j.physletb.2012.08.021.

J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaitre, A. Mertens, and M. Selvaggi. Delphes 3, a modular framework for fast simulation of a generic collider experiment. *Journal of High Energy Physics*, 2014(2), 57., 2014.

S. Gleyzer, Prosper Orlando R. D., S. H.B., Sekmen, and O.A. Zapata. Physics at tev colliders. new physics working group report. contribution 15. falcon: towards an ultra fast non-parametric detector simulator. *arXiv preprint arXiv:1203.1488*, 2016.

W. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. *Conference Notes from 31st Conference on Neural Information Processing Systems*, 2017.

Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in tensorflow. *arXiv preprint arXiv:1802.05799*, 2018.