Point Cloud Generation using Transformer Encoders and Normalising Flows

Benno Käch
Deutsches Elektronen-Synchrotron DESY
Hamburg, Germany
benno.kaech@desy.de

Dirk Krücker
Deutsches Elektronen-Synchrotron DESY
Hamburg, Germany
dirk.kruecker@desy.de

Isabell-A. Melzer-Pellmann
Deutsches Elektronen-Synchrotron DESY
Hamburg, Germany
isabell.melzer@desy.de

Abstract

Data generation based on Machine Learning has become a major research topic in particle physics. This is due to the current Monte Carlo simulation approach being computationally challenging for future colliders, which will have a significantly higher luminosity. The generation of collider data is similar to point cloud generation, but arguably more difficult as there are complex correlations between the points which need to be modelled correctly. A refinement model consisting of normalising flows and transformer encoders is presented. The normalising flow output is corrected by a transformer encoder, which is adversarially trained against another transformer encoder discriminator/critic. The model reaches state-of-the-art performance while yielding a stable training.

1 Introduction

High Energy Physics (HEP) has benefited from the advances in Machine Learning (ML) since the analysis of HEP data is a high-dimensional multivariate problem. There have been multiple applications of ML to HEP [1], although most of them were in the supervised approach to ML. In HEP, detailed simulations of the physical processes, which almost perfectly describe the details of the experimental measurement, are commonly available. These Monte Carlo simulations (MC) provide labeled data and are needed in large numbers to cover the extreme areas of the physical phase space. The simulations for the CMS detector at the Large Hadron Collider, for example, require about 50% [2] of the current CMS computing budget. An even larger number of simulations will be needed for the coming high-luminosity phase of the LHC [3]. Therefore, generative modelling with Deep Learning (DL) has sparked great interest in the HEP community.

In this study, the generation of jets is investigated using Normalising Flows (NFs) [4; 5; 6; 7] and an adversarially trained refinement setup consisting of two transformer [8; 9] encoder networks. The Jets in the dataset are described as point clouds in momentum space $\{(\eta^{(i)}, \phi^{(i)}, p_T^{(i)})_{i\leq 30}\}$, which motivates the use of transformers as attention is permutation invariant. One of the transformer encoders acts as a refinement model to enhance the NF output. The use of NFs is motivated by their stable convergence thanks to Maximum Likelihood training, however NFs often struggle to model all correlations present in the training data correctly and are not particularly well suited for point clouds.

Machine Learning and the Physical Sciences workshop, NeurIPS 2022.

The use of the refinement network enhances the performance in the modeling of the correlations significantly.

This extended abstract first gives an overview over the models used, then the different training paradigms are briefly discussed. Finally, the results for different models are presented and compared.

2 Dataset

In this first study, the *JetNet* [10] top quark dataset [11] is used. We restrict ourselves to top quark jets due to their more complex sub-structure. The dataset contains $\sim 180,000$ samples and and a 70/30 train/test split is applied. The data consists of top quark jets with an energy of about 1 TeV, with each jet containing up to 30 particles. These jet constituents are considered to be massless and can therefore be described by their 3-momenta or equivalently by transverse momentum p_T, pseudorapidity η, and azimuth angle ϕ. In the *JetNet* dataset, these variables are given relative to the jet momentum: $\eta_i^{rel} := \eta_i^{particle} - \eta^{jet}$, $\phi_i^{rel} := (\phi_i^{particle} - \phi^{jet}) \bmod 2\pi$, and $p_{T,i}^{rel} := p_{T.i}^{particle} / p_T^{jet}$, where i runs over the particles in a jet.

The invariant mass m_{jet} of a jet is an essential high-level feature containing important physics information. It is a global variable that depends on the correlations between the single jet constituents and provides therefore an important metric for the performance of the generative model. For the relative quantities above, we can define the scaled jet mass as $(m^{rel})^2 = (\sum_i {E_i}^{rel})^2 - (\sum_i \vec{p}_i^{\,rel})^2 = m_{jet}^2 / p_{T,jet}^2$.

An additional set of metrics is constructed in [10] by using the Energy Flow Polynomials (EFP) [12], which form a complete set of jet substructure observables.

The *JetNet* paper additionally proposes a Message-Passing Generative Adversarial Network (MP-GAN) that outperforms other state-of-the-art models at the time of publication and to which we compare our results.

3 Generative Modelling

Modeling an unknown complex probability function only from a set of data points is challenging. In addition, it is often not straightforward to assess the quality of generated data. New learning paradigms are needed to be able to train such a model. GANs, as one of the most popular approaches, employ a setup of two networks, a generator and a discriminator that tries to distinguish generated and real data. Their training is notoriously difficult. A more stable approach to generative modelling are NFs, allowing Maximum Likelihood training, which is typically both fast and reproducible.

3.1 Normalising Flow Models

In NFs, the objective is to transform the training data distribution to a Normal distribution by mapping the training data with a series of invertible functions. This setup then allows for Maximum Likelihood optimization of the parameters of the transformation. Once the NF can transform the training data to a Normal distribution sufficiently well, new data is generated by sampling from a Normal distribution and applying the inverted transformations. This study used the `nflows` [13] library for NFs.

3.1.1 Rational Quadratic Spline Coupling Layers

In NFs with coupling layers, a clever trick is used to guarantee the invertibility of the transformation: the input dataset is split into two disjoint sets along its feature dimensions. The first set is mapped with the identity function, whereas the second set is transformed with a usually elementwise transformation. In the case of Rational Quadratic Splines (RQS) [14], the quotient of two monotonic quadratic splines is used, which is by construction invertible. To increase the expressivity of the transformation, the parameters of the RQS for the second feature set are the output of a neural network applied to the first set. In our model, multiple of such RQS coupling layers are stacked, and after each layer a different split for the two sets is chosen. The Neural Networks used for producing the parameters of the RQS are fully connected feed-forward networks, using skip connections.

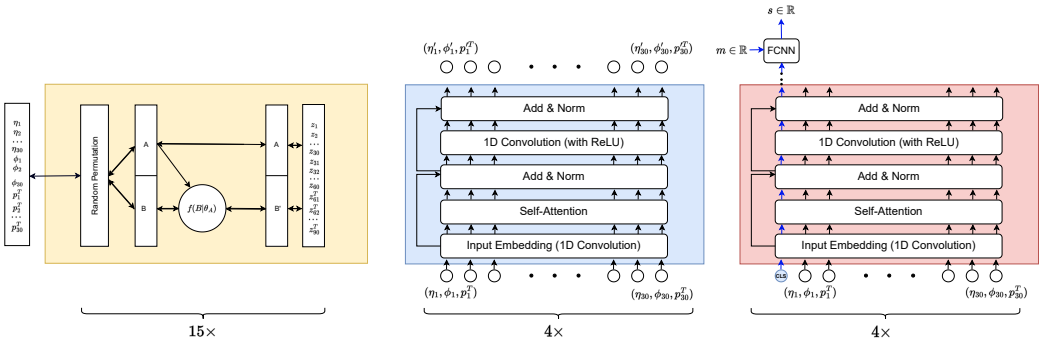

Figure 1: An overview of all components used in the model. The schematic on the left describes the NF, made up of 15 coupling layers. The middle one shows the refinement network, which is made up of 4 transformer encoder blocks and the one on the right similarly illustrates the critic network. All three models are used during the training, while only the first two are used during generation. The number of layers are given for the model architecture that gives the results presented in this study.

3.2 Transformers

Transformers were originally proposed as sequence-to-sequence models for machine translation [9]. However, they have been widely adopted in various fields like computer vision and speech processing. On an abstract level, the main ingredients of a transformer are the encoder block and the decoder block. In this study, similarly as for most vision transformers, only an encoder block is used [15]. The attention architecture [8] is typically permutation invariant, which is a useful feature for modelling particles in jets. The encoder block consists of two main components, one being 1×1 convolutions applied token-wise (here particle-wise, since in this case a token corresponds to a particle) and the second being multi-headed self attention. Important for our use case is that transformers can practically take a variable number of particles as input. This is done by zero-padding particles, which are then masked in the calculation of the attention: their respective values are set to negative infinity in the softmax function of the attention calculation such that they do not have any influence on the output at all.

In this study the output of the NFs is corrected with a post-processing transformer encoder network. An adversarial setup is used to train the refinement network which provides an additive correction to the output of the NF. A discriminator or critic is used to judge whether the corrected NF generated jets resemble real data. This refinement setup is used instead of a setup of only a generator and discriminator as the training of adversarial architectures and transformers is already difficult enough without combining them.

3.2.1 Post-processing network

The post-processing network is a transformer encoder, which takes the output of the NF and reshapes it to $\mathbb{R}^{N \times D}$, where N is the maximum number particles per jet and D is the features per particle, in the case of this study $D = 3$, and $N = 30$. It uses Layer Normalization [16] and residual connections to connect the pre-attention features with the post-attention layer features (skip connections in fig. 1). The normalization is applied after the addition (Add&Norm in fig. 1).

3.2.2 Discriminator/Critic Network

Depending on whether a more Vanilla GAN-like [17; 18] or a WGAN [19; 20] setup is employed, a discriminator or critic is used. The latter is another transformer encoder that uses a BERT style classification token [15] to assess the realness of a jet. After the classification token passes through the second transformer encoder network, a fully-connected network is applied, which additionally takes the invariant mass as input. The rest of the architecture is the same as for the post-processing network. A schematic of all parts of the architecture is shown in Fig. 1.

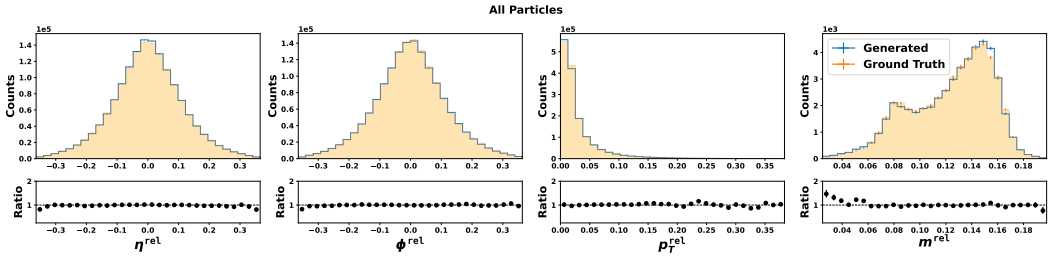

Figure 2: Distribution of all particles $(\eta^{rel}, \phi^{rel}, p_T^{rel})$ and the invariant mass. The generated data is compared to a hold out data set, and a ratio is shown below the plot.

3.3 Training

Different Training procedures are tested and compared, including the Vanilla GAN [17] training, an LSGAN [18] and a WGAN training with gradient penalty [19; 20]. Optimization is done with RMSprop [21], the learning rate was varied with a two-cycle cosine scheduling after a linear warm-up that is commonly used with transformers. The most stable results were obtained with the LSGAN training. The presented model was trained on a NVIDIA P100 GPU after around 11 hours of training time. To produce results which can be compared to Ref. [10], we perform the same (70/30) train/test split and evaluate with the functions that are provided by the *JetNet* library.

4 Results

The results on the top-quark dataset from Ref. [22] are shown in Table 1. The first two entries are from the best performing models of Ref. [10], while VNF is the evaluation of the NF output and TF is the transformer-refined results. The best performing model used 15 RQS coupling layers and 4 transformer encoder layers in the refinement network and the critic. The performance of the proposed models is comparable or better with respect to all considered metrics. Of particular interest is the FPND metric, where the proposed model performs considerably better. In Fig. 2 the results are visualized by comparing the generated particles with the holdout set. The marginal distribution of all features of all particles and the distribution of the invariant mass of the jet are shown. The model needs $\sim 7.6\mu s$ for the generation of one jet, which is 4.6 times faster than the model in Ref. [10] on a `NVIDIA A100`.

Model	$W_1^M(\times 10^{-3})$	$W_1^P(\times 10^{-3})$	$W_1^{EFP}(\times 10^{-5})$	FPND	COV ↑	MMD
MP-MP [10]	$\mathbf{0.6 \pm 0.2}$	2.3 ± 0.3	$\mathbf{2 \pm 1}$	0.37	**0.57**	**0.071**
MP_LFC-MP [10]	0.9 ± 0.3	2.2 ± 0.7	$\mathbf{2 \pm 1}$	0.93	0.56	0.073
VNF	5.0 ± 0.2	2.5 ± 0.6	14 ± 2	5.61	0.56	0.072
TF	0.78 ± 0.09	$\mathbf{1.3 \pm 0.3}$	$\mathbf{2 \pm 1}$	**0.08**	**0.57**	0.072

Table 1: Comparison between the best performing models from Ref. [10], with different models from this study on the top quark dataset. Scores written in bold highlight the best value.

5 Conclusion

In this study a particle cloud generation model consisting of normalising flows and transformer encoders is proposed. The transformer encoder acts as a refinement model on the output of the normalising flow, and is trained in an adversarial fashion using another transformer encoder as critic. Intuitively one would not expect that NFs are well suited for point cloud generation, however they provide a stable starting point, as adversarial setups and transformer models are notoriously difficult to optimize. The model reaches state-of-the-art performance on the publicly available *JetNet* [23] top quark dataset. It is important to note that particle clouds are similar to point clouds, possibly

even more complex. As such it is expected that the model also would perform well on point cloud generation tasks. For future studies it is planned to use the model on the other *JetNet* datasets [22] as well as on a dataset with more particles, and to evaluate its scaleability. It is expected that inputs with a higher dimensionality are difficult for the attention calculation as it scales with the number particles squared. But as transformers are an active field of research, there seem to be possible ways to overcome this [24; 25].

6 Acknowledgements

Benno Käch is funded by Helmholtz Association's Initiative and Networking Fund through Helmholtz AI (grant number: ZT-I-PF-5-64). This research was supported in part through the Maxwell computational resources operated at Deutsches Elektronen-Synchrotron DESY (Hamburg, Germany). The authors acknowledge support from Deutsches Elektronen-Synchrotron DESY (Hamburg, Germany), a member of the Helmholtz Association HGF.

References

[1] A living review of machine learning for particle physics. 2022. URL https://iml-wg.github.io/HEPML-LivingReview/.

[2] CMS Collaboration. CMS offline and computing public results. URL https://twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults.

[3] G Apollinari, O Brüning, T Nakamoto, and L Rossi. High luminosity large hadron collider hl-lhc, 2015. URL https://cds.cern.ch/record/2120673.

[4] E. G. Tabak and Cristina V. Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164. doi:https://doi.org/10.1002/cpa.21423.

[5] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1530–1538. PMLR, 2015. URL https://proceedings.mlr.press/v37/rezende15.html.

[6] Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979, nov 2021. doi:10.1109/tpami.2020.2992934. URL https://arxiv.org/abs/1908.09257.

[7] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. 2019. doi:10.48550/ARXIV.1912.02762. URL https://arxiv.org/abs/1912.02762.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

[9] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

[10] Raghav Kansal, Javier M. Duarte, Hao Su, Breno Orzari, Thiago Tomei, Maurizio Pierini, Mary Touranakou, Jean-Roch Vlimant, and Dimitrios Gunopoulos. Particle cloud generation with message passing generative adversarial networks. *CoRR*, abs/2106.11535, 2021. URL https://arxiv.org/abs/2106.11535.

[11] Raghav Kansal, Javier Duarte, Hao Su, Breno Orzari, Thiago Tomei, Maurizio Pierini, Mary Touranakou, Jean-Roch Vlimant, and Dimitrios Gunopulos. Jetnet(1.1)[topquark], May 2021. URL https://doi.org/10.5281/zenodo.6302454.

[12] Patrick T. Komiske, Eric M. Metodiev, and Jesse Thaler. Energy flow polynomials: a complete linear basis for jet substructure. *Journal of High Energy Physics*, 2018(4), apr 2018. doi:10.1007/jhep04(2018)013. URL https://doi.org/10.1007%2Fjhep04%282018%29013.

[13] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. nflows: normalizing flows in PyTorch, November 2020. URL https://doi.org/10.5281/zenodo.4296287.

[14] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows, 2019. URL https://arxiv.org/abs/1906.04032.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. URL https://arxiv.org/abs/1810.04805.

[16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.

[17] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. URL https://arxiv.org/abs/1406.2661.

[18] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, and Zhen Wang. Multi-class generative adversarial networks with the L2 loss function. 2016.

[19] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. 2017.

[20] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of Wasserstein GANs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, page 5767. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7159-improved-training-of-wasserstein-gans.pdf.

[21] Tijmen Tieleman and Geoffery Hinton. Rmsprop gradient optimization, 2014. URL http://www.cs.toronto.edu/tijmen/csc321/slides/lecture_slides_lec6.pdf.

[22] Raghav Kansal, Javier Duarte, Hao Su, Breno Orzari, Thiago Tomei, Maurizio Pierini, Mary Touranakou, Jean-Roch Vlimant, and Dimitrios Gunopulos. JetNet, 05 2021.

[23] Raghav Kansal. jetnet library. URL https://github.com/jet-net/JetNet.

[24] Markus N. Rabe and Charles Staats. Self-attention does not need $o(n^2)$ memory, 2021. URL https://arxiv.org/abs/2112.05682.

[25] Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena, and Chinmay Hegde. On the computational complexity of self-attention, 2022. URL https://arxiv.org/abs/2209.04881.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Scalability is unclear
   (c) Did you discuss any potential negative societal impacts of your work? [No] The model brings the same negative impact as most generative models - but due to the limited space in this extended abstract it has been left out.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [No] Again, left out due to limited space
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Again, left out due to limited space
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] The libraries that were used are *nflows* and *JetNet*
   (b) Did you mention the license of the assets? [Yes] Both libraries use the MIT license
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]