Quantum GANs for $t\bar{t}H(b\bar{b})$ Process Data Generation

OCTOBER 2021

AUTHORS:
Togan Tlimakhov Yusuf
Department of EEE, Ankara University

Eraraya Ricardo Muten
Quantum Technology Lab., Institut Teknologi Bandung

Andrei Voicu Tomut
Faculty of Physics, Babes¸-Bolyai University

SUPERVISOR:
Sofia Vallecorsa

MENTORS:
Su Yeon Chang, Florian Rehm, Simon Schnake

CERN openlab

ABSTRACT

In this report, we present the Deep Learning generative model GAN for the Higgs boson $t\bar{t}H(b\bar{b})$ process data generation. Initially, a classical GAN model is considered, with Convolutional layers, Batch Normalization layers, and a Leaky ReLU activation function. The GAN aims to simulate the Higgs process precisely, capturing the crucial features in each b-jet produced. Two b-jets were considered in this work, each possessing four features that were resized to fit the Neural Network training process, where a relatively decent Wasserstein distance was obtained. Subsequently, a Quantum GAN model was considered, where the Quantum Circuit consisted of Gaussian gates as a continuous variable architecture per the nature of the dataset constraint. Xanadu's both PennyLane and Strawberry Fields Python libraries were used on a continuous variable quantum neural network-based, where obtaining comparable results with the classical benchmark was intended on the simulators, considering a smaller dataset with fewer features.

TABLE OF CONTENTS

1. INTRODUCTION

The experiments conducted in the field of High-energy physics (HEP) require the processing of large and complex datasets, which present a challenge to process and analyze these data with high accuracy. Particle transport simulation is one of the primary methods for understanding the outcomes of particle collisions such as the collisions at the Large Hadron Collider (LHC), where the most common kind of interaction between the two colliding protons is the scattering between quarks and gluons.

The conventional approach to simulate these events is based on the Monte Carlo method, which has become both time-consuming and computationally expensive as the produced data increases. For the past years, Monte Carlo simulations have represented more than 50% of the WLCG (LHC Computing Grid) workload and the simulation demands are expected to increase significantly in the next years for the High Luminosity LHC runs [1][2].

This motivated the use of machine learning techniques such as the applications of Deep Learning in different aspects to improve the performance and analysis of the data. Additionally, the Generative Adversarial Network (GAN) model, which is a class of machine learning frameworks, seems suited to replace the Monte Carlo methods, as it can model complicated probability functions and deal with multi-modal outputs.

Moreover, the field of quantum computing has been in the development phase over the past several years, with the promise of providing a significant computational speed-up, besides reducing the number of required calculation steps in certain tasks like unstructured search [3] and factoring large numbers [4]. Hence, considering that the experiments in high energy physics require large computing resources, one might wonder whether quantum machine learning (QML) techniques could help overcome this computational challenge. Like classical machine learning techniques, the ”advantage” that quantum computing may provide in certain tasks was a motivation for it to be used in high energy physics for tasks like data analysis [5], the identification of charged particle trajectories [6], and more. Many of these attempts to achieve specific tasks used techniques of quantum machine learning, which is like classical machine learning, yet has the potential to demonstrate a quantum advantage over classical algorithms [7].

The recent development of quantum computing platforms and simulators that are available for public experimentation such as Qiskit [8] and PennyLane [9] led to a general acceleration of research interest on quantum algorithms and their applications. As for high-energy physics, quantum algorithms were suggested to tackle the computational challenges faced in data processing and analysis [10].

2. RELATED WORK

The main objective of this project is to explore the applications of deep learning and quantum computing in the field of high-energy physics. Different approaches have been explored in this domain using both classical and quantum machine learning techniques.

In [11], a classical GAN was used to simulate the LHC QCD events, in which the architecture we developed to set a relatively good classical benchmark for the quantum model. In [12] on the other hand, a quantum classifier has been explored to classify the $t\bar{t}H(b\bar{b})$ dataset, with performance comparable to the classical counterparts (SVM, Random Forest, AdaBoost).

In [13] a quantum GAN model was explored with a continuous-variable architecture to simulate High Energy Physics detectors, wherein [14] a Dual-Parameterized Quantum Circuit GAN Model was implemented. In our work, we tried to explore some of the already implemented models and to improve on them.

3. THE DATASET

The presses of identifying the Higgs boson production in association with top quark-antiquark pairs in which the Higgs decays into a pair of bottom quark-antiquark is crucial for understanding the functioning mechanism of our universe and serve as a potential measure for undiscovered physics since the Higgs boson top quark Yukawa coupling carries information about the scale of new physics [15].

Experimentally, distinguishing between signal and background is extremely challenging since in both cases the final state is similar. The complex final state of the $t\bar{t}H(b\bar{b})$ process comes with a large number of jets but allows studying the purely fermionic Higgs production and decay.

The semi-leptonic channel is addressed in order to suppress the QCD background. Moreover, using the decay of the Higgs boson into a pair of bottom quarks balances the very low production cross-section leading to a larger number of events to be observed [12].

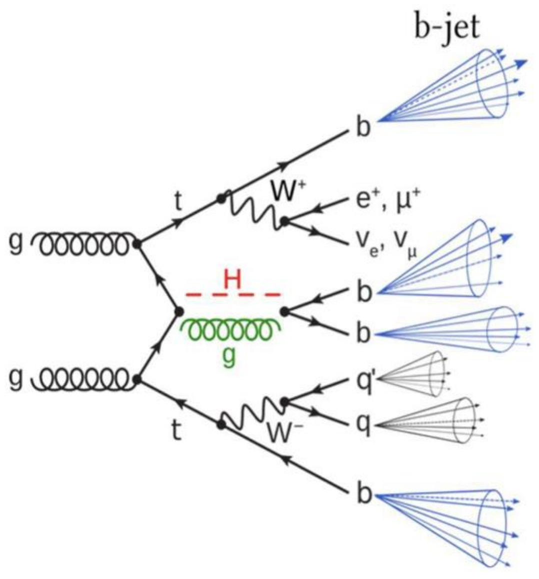


Figure 1. The Feynman diagram of the signal process (Higgs) in red and the dominant background process (gluon) in green

In our work, we implemented the simulation of the two b-jets from the Higgs, where the main focus was on simulating the Higgs event only, not the background (gluon).

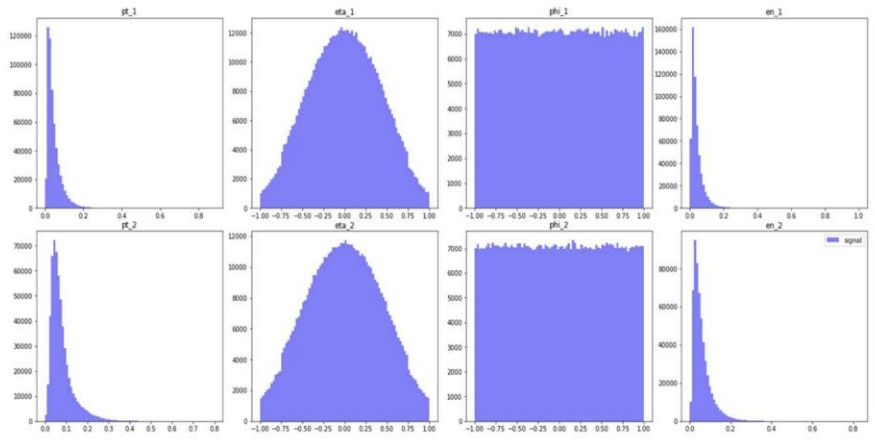


Figure 2. Presented dataset from the numerical Monte Carlo simulations

In figure 2 the features from the Monte Carlo simulations are given, where Pt is the Transverse momentum, η is Pseudo-rapidity, ϕ is Azimuthal angle, and E is the Energy of each b-jet. We limit the problem to only 4 features for each b-jet (from 8 initially) and normalized them into either [-1, 1] or [0, 1] ranges.

4. GENERATIVE ADVERSARIAL NETWORKS (GANs)

The state of generative models in Machine Learning has advanced dramatically in recent years, with Generative Adversarial Networks (GANs) being one of the most promising methods for unsupervised learning and at the forefront of attempts to generate high-fidelity, diverse images and signals with models learned directly from the presented data.

GANs are a class of machine learning models that consist of two networks, a Generator and a Discriminator competing with each other. The generative model G captures the data distribution, and the discriminative model D estimates the probability that a sample came from the training data rather than G.

The training procedure for the generator is to maximize the probability of the discriminator making a mistake. This framework corresponds to a minimax two-player game [16]. Where, the GAN’s objective, in its original form involves finding a Nash equilibrium to the following equation:

$$\min_{G}\max_{D} V(G,\ D) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1-D(G(z)))]$$

a. THE DISCRIMINATOR NETWORK

The discriminative model is simply a classifier that has two classes, real and fake. Given an input x, the discriminator calculates the probabilities and classifies the input x, where it tries to distinguish real data from the fake data created by the generator. It is possible to use any network architecture appropriate to the type of data it's classifying.

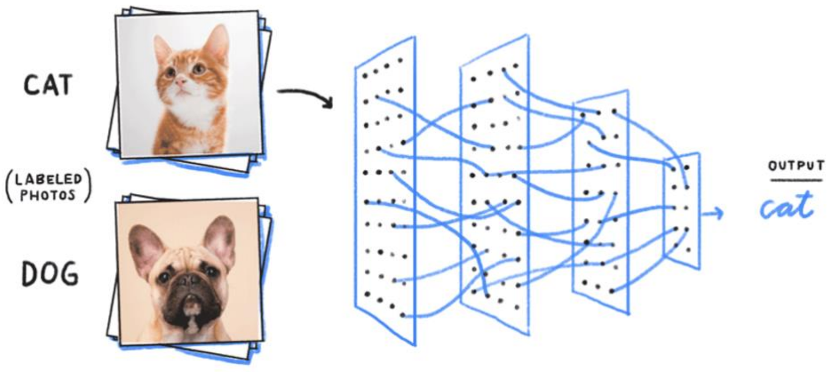


*Figure 3. An example of the Discriminative model**

The discriminator mainly connects to two loss functions. During discriminator training, the discriminator ignores the generator loss and only uses the discriminator loss, where it classifies both real data and fake data from the generator and the loss penalizes the discriminator for misclassifying a fake instance as real and vice versa.

Following, the weights are updated through backpropagation from the discriminator loss through the network. The training algorithm for the discriminative model is done by updating the discriminator's gradient descent is as follow:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))]$$

And considering the optimal discriminator D for any given generator G. Where G is fixed, the optimal discriminator D is:

$$D_G(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

b. THE GENERATOR NETWORK

The Generator model on the other hand uses the feedback from the Discriminator, where it tries to find all the features that represent the original input, $p(x \mid y)$, which is a much harder task than discrimination, since the output space of possible outcomes is relatively big, which makes it challenging for the generator.

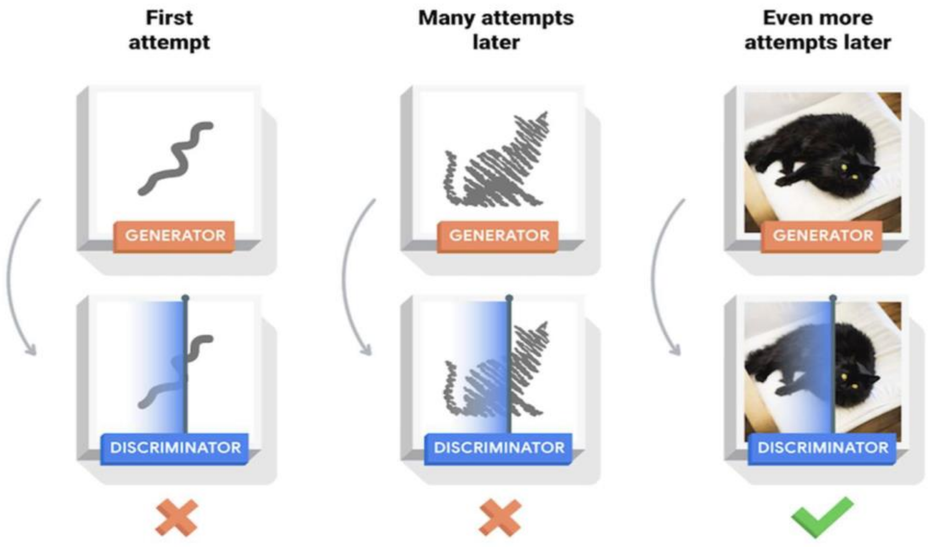


*Figure 4. An example of Generative model***

Typically, it takes the generator multiple steps to improve itself for every step the discriminator takes. It's easy to discriminate between two outputs than knowing exactly all the features of all possible outcomes of the feature space. Yet training the generative model in multiple steps can result in the generation of all kinds of different outputs, such as realistic images or signals.

Each block in the generator's network includes a linear transformation to map to another shape, a batch normalization for stabilization, and finally a non-linear activation function (Leaky ReLU) so the output can be transformed in complex ways. The training algorithm for the generative model using gradient descent is as follow:

$$\nabla_{\theta g} \frac{1}{m} \sum_{i=1}^{m} [\log(1 - D(G(z^{(i)})))]$$

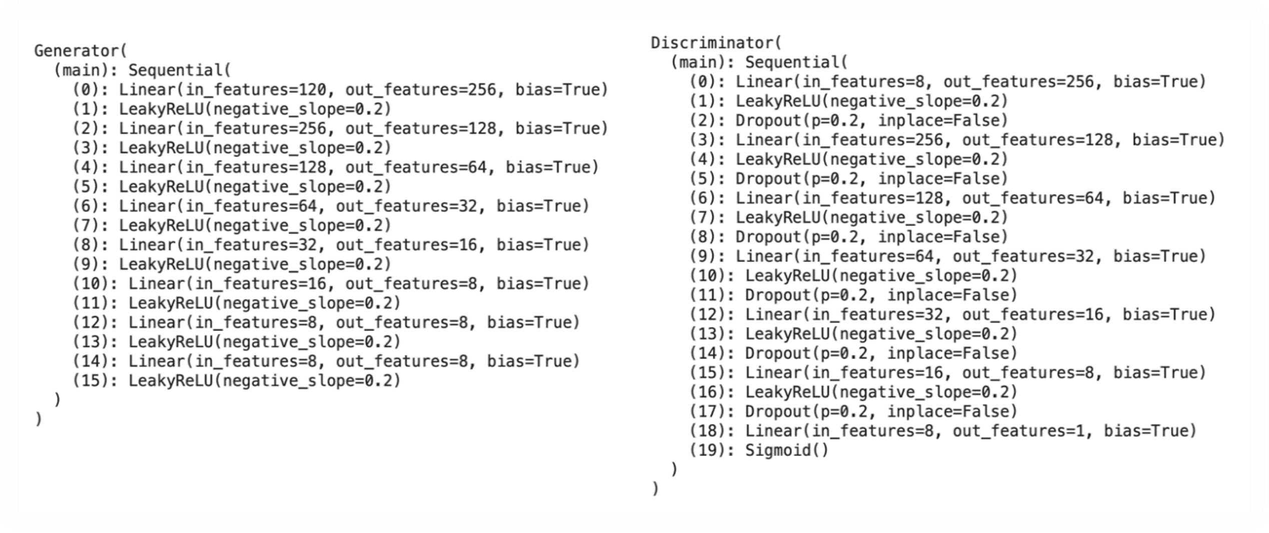
Another important difference in the generator model is the noise vector z, which has a crucial role in making sure that the signals generated from the same class are not the exact same.

The noise vector is usually generated by sampling random numbers either between 0 and 1 uniformly, or from the normal distribution with a mean of zero, and variance of 1, z~N(0,1).

5. CLASSICAL BENCHMARK

To evaluate the efficiency of a quantum model, we first set a classical GAN as a benchmark. In the beginning, we create a simple GAN with a generator composed only of linear layers and Leaky ReLU as an activation function.

Then, for the discriminator, we intercalate linear layers with Dropout layers, and for the last layer, we use a sigmoid activation. Finally, as a cost function, we used binary cross-entropy (BCE), and with Adam optimizer, we get the result from Figure 5 after 10000 epochs. As expected, the generated data's distributions are far from the targeted original ones.



```
Generator(
  (main): Sequential(
    (0): Linear(in_features=120, out_features=256, bias=True)
    (1): LeakyReLU(negative_slope=0.2)
    (2): Linear(in_features=256, out_features=128, bias=True)
    (3): LeakyReLU(negative_slope=0.2)
    (4): Linear(in_features=128, out_features=64, bias=True)
    (5): LeakyReLU(negative_slope=0.2)
    (6): Linear(in_features=64, out_features=32, bias=True)
    (7): LeakyReLU(negative_slope=0.2)
    (8): Linear(in_features=32, out_features=16, bias=True)
    (9): LeakyReLU(negative_slope=0.2)
    (10): Linear(in_features=16, out_features=8, bias=True)
    (11): LeakyReLU(negative_slope=0.2)
    (12): Linear(in_features=8, out_features=8, bias=True)
    (13): LeakyReLU(negative_slope=0.2)
    (14): Linear(in_features=8, out_features=8, bias=True)
    (15): LeakyReLU(negative_slope=0.2)
  )
)
```

```
Discriminator(
  (main): Sequential(
    (0): Linear(in_features=8, out_features=256, bias=True)
    (1): LeakyReLU(negative_slope=0.2)
    (2): Dropout(p=0.2, inplace=False)
    (3): Linear(in_features=256, out_features=128, bias=True)
    (4): LeakyReLU(negative_slope=0.2)
    (5): Dropout(p=0.2, inplace=False)
    (6): Linear(in_features=128, out_features=64, bias=True)
    (7): LeakyReLU(negative_slope=0.2)
    (8): Dropout(p=0.2, inplace=False)
    (9): Linear(in_features=64, out_features=32, bias=True)
    (10): LeakyReLU(negative_slope=0.2)
    (11): Dropout(p=0.2, inplace=False)
    (12): Linear(in_features=32, out_features=16, bias=True)
    (13): LeakyReLU(negative_slope=0.2)
    (14): Dropout(p=0.2, inplace=False)
    (15): Linear(in_features=16, out_features=8, bias=True)
    (16): LeakyReLU(negative_slope=0.2)
    (17): Dropout(p=0.2, inplace=False)
    (18): Linear(in_features=8, out_features=1, bias=True)
    (19): Sigmoid()
  )
)
```

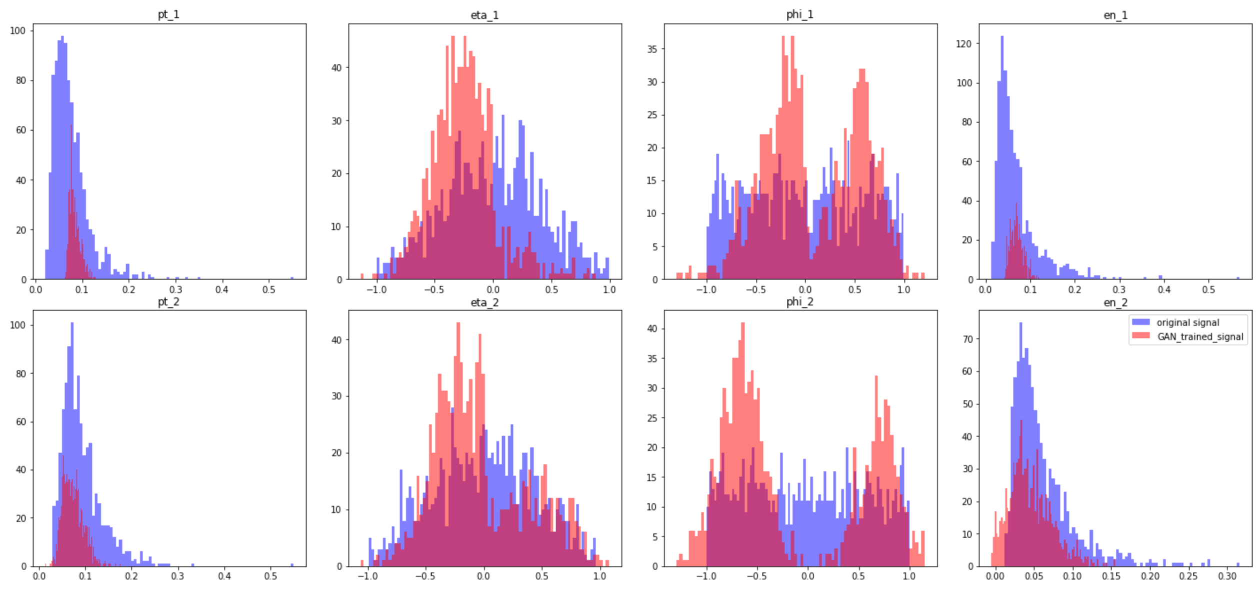


Figure 5. The generated data's distributions

Afterward, we created a convolutional GAN inspired by the model "DijetGAN: a Generative-Adversarial Network approach for the simulation of QCD dijet events at the LHC " [11] with the architecture summarized in Figure 6.

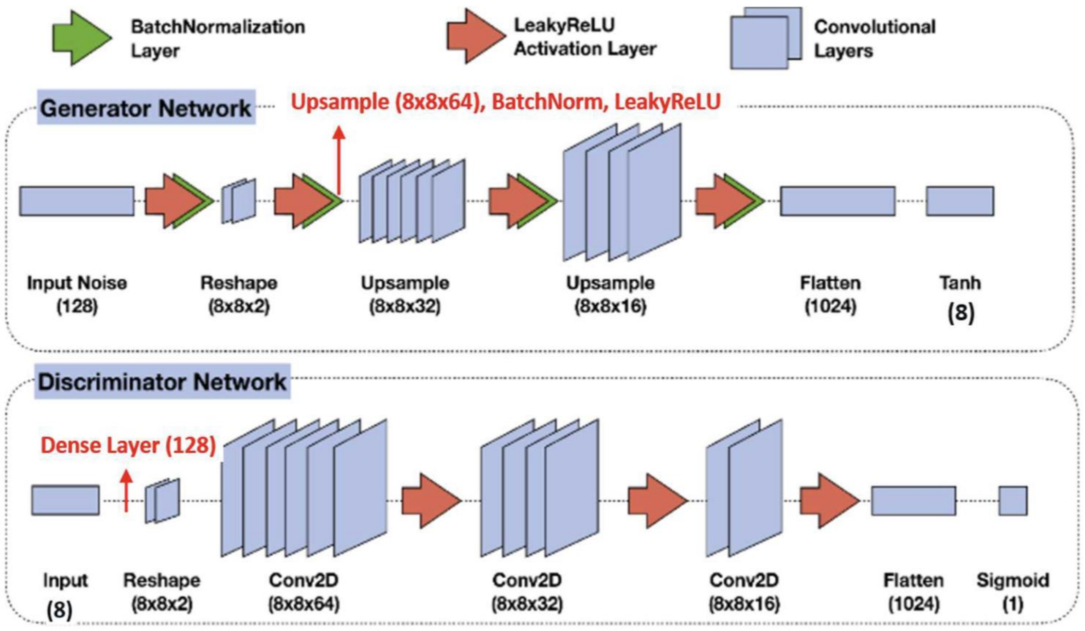


Figure 6. The modified DijetGAN architecture

In addition to the Dijet model, we added an Upsample(8x8x64) layer in the discriminator and a new dense layer in the discriminator. With filter size for Conv2D and Upsample: 3x3, with stride 1x1, and lr=10^(-5) and beta_1=0.5 and beta_2=0.9. With the modified model, we manage to get a Wasserstein distance of 0.0271705, where figure 7 shows the generated data's distributions.

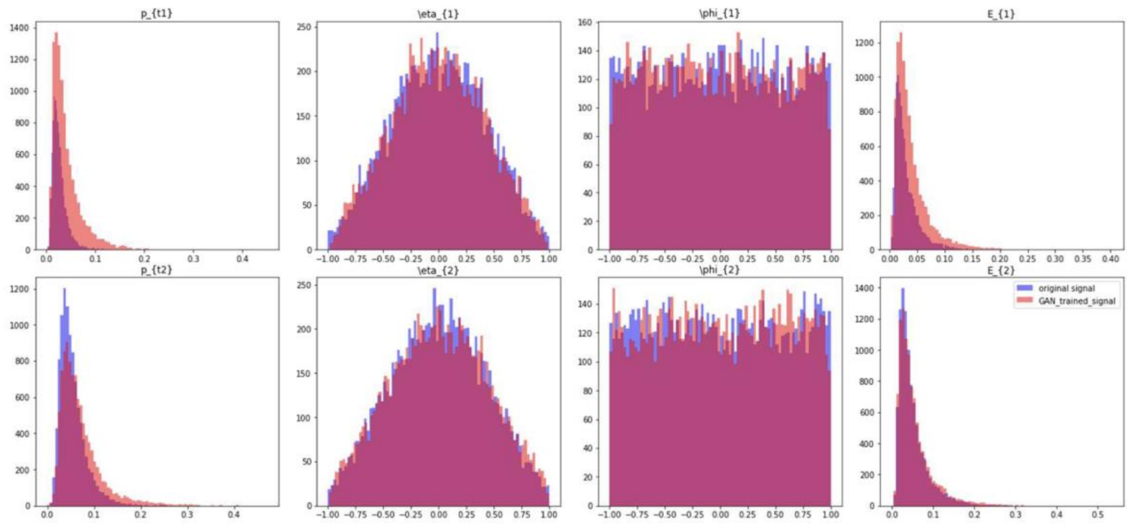


Figure 7. Generated data with the modified DijetGAN model

6. THE QGAN MODEL

The recent advances in machine learning and quantum computing allowed the merge of quantum machine learning models where the classical GAN paradigm can be extended to the quantum world. Unfortunately, the most widely known hybrid model proposed by IBM [12] with a classical discriminator and quantum generator is limited to the production of probability distributions over discrete variables, while the calorimeter outputs at the LHC are continuous in nature and require a continuous variable (CV) architecture.

CV quantum computation with hardware like photonics chips offers a solution to this problem. Unlike the discrete architecture, CV uses qumodes, showing its efficiency in several kinds of research that evaluated its applications [17].

In the CV architectures, the information is encoded in a continuous physical observable, such as the electromagnetic field. For the qumodes, the information is carried in quantum states of bosonic modes in the quantized electromagnetic field, where the quantum state of N qumodes is expressed as the superposition of the position basis, {|x⟩} or the Fock basis, {|n⟩}.

$$|\psi\rangle = \int_{\mathbb{R}^N} \psi(\mathbf{x})\,|\,\mathbf{x}\rangle \mathrm{d}\mathbf{x} = \sum_{i=0}^{\infty} \langle n|\,\psi\rangle\,|\,n\rangle$$

The CV model was examined for three qumodes, n = 3, due to the limitation of the simulator's computing time and memory. The original calorimeter outputs of size 25 × 25 are averaged over the longitudinal direction and then binned into 3 pixels as in [13]. Stability and convergence can be improved by increasing the latent space dimension.

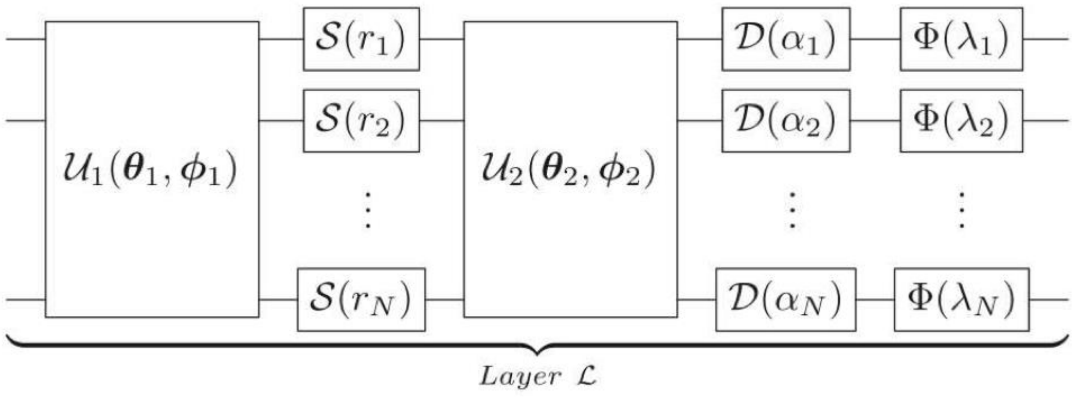


Figure 8. The circuit structure of a CV quantum neural network layer

Figure 8 shows the CV quantum neural network from [17] with an interferometer, local squeeze gates, local displacements, and local non-Gaussian gates. To further improve the model, various approaches can be made, including speeding up the training and trying to increase the number of qumodes, and hence the model's representational power. The current simulations require a tremendous amount of time for a small number of qumodes.

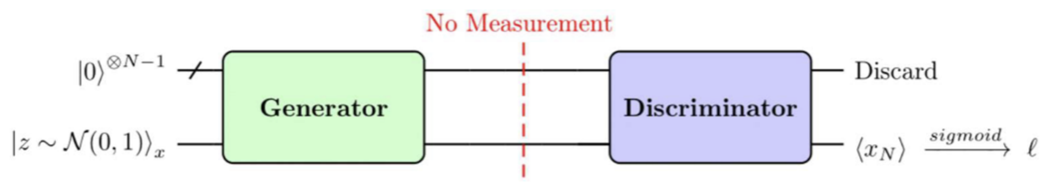

No Measurement
$|0\rangle^{\otimes N-1}$
$|z \sim \mathcal{N}(0,1)\rangle_x$
Generator
Discriminator
Discard
$\langle x_N \rangle \xrightarrow{sigmoid} \ell$

Figure 9. The schematics of fully quantum CV QGAN model manipulating embedded data

Further, different entanglement configurations can be tested together with mechanisms re-producing regularization that is usually applied to classical networks, in order to improve the QGAN model and reproduce the real image data accurately in future studies.

7. CONCLUSION

In this study, the Deep Learning framework of Generative Adversarial Networks was used to simulate and generate the Higgs Boson ttH(bb) process data which is one of the most crucial processes concluded at the large hadron collider. The GAN model considered was a modified and improved version of the DijetGAN. Multiple convolutional layers, Batch Normalization layers were implemented in the architecture, while reshaping, upsampling, and flattening the data entries, besides using the Leaky ReLU activation function for both the Generator and Discriminator. The training was conducted using both PyTorch and TensorFlow Libraries, and a Wasserstein distance of 0.0271705 was obtained as the best-measured outcome.

Subsequently, the Quantum GAN model was implemented with a continuous variable architecture as per the dataset nature requirement, where Gaussian gates; Displacement, Rotation, Squeezing, alongside the non-Gaussian Kerr gate to satisfy the continuous spectra of the dataset. Decreasing the size of the dataset and the numbers features due to the Fock backend computational limitation might lead to obtaining comparable results with the classical benchmark. Taking into account the usage of simulators and the current limitations of the Quantum Hardware, it can be concluded that QGANs could present a potential advantage over classical approaches in the field of High Energy Physics simulation due to the higher dimensional representation. A promising future direction is to train directly on quantum hardware, while exploring techniques to speed the training, increasing the number of qumodes at the same time, and hence the model's representational power. It is important to notice that a more comprehensive comparison between several modifications of the models might lead to a better understanding of the training strategies in general.

8. ACKNOWLEDGMENT

We would like to thank CERN for the opportunity to participate in the Summer Internship program and for all the instructors at the lecture series both at the CERN Summer lecture program and the openlab student events. Furthermore, we would like to sincerely thank our supervisor Dr. Sofia Vallecorsa for the commitment and support throughout the Internship. Additionally, we thank our Mentors on the project; Su Yeon Chang, Florian Rehm, and Simon Schnake for taking the time to join the meeting and provide us with the necessary guidance and support.

9. REFERENCES

[1] Cds.cern.ch. 2021. Update of the Computing Models of the WLCG and the LHC Experiments. [online] Available at: <https://cds.cern.ch/record/1695401/files/LCG-TDR-002.pdf> [Accessed 8 October 2021].

[2] Bird, I., Workshop Introduction, Context of the Workshop: Halfway through Run2; Preparing for Run3, Run4. (2016). https://indico.cern.ch/event/ 555063/contributions/2236372/ WLCG Workshop.

[3] L. K. Grover, “A fast quantum mechanical algorithm for database search,” in Proceedings of the Annual ACM Symposium on Theory of Computing, vol. Part F129452. New York, New York, USA: Association for Computing Machinery, jul 1996, pp. 212–219. [Online]. Available: http://portal.acm.org/citation.cfm?doid=237814.237866

[4] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” SIAM Review, vol. 41, no. 2, pp. 303–332, aug 1999.

[5] A. Mott, J. Job, J.-R. Vlimant, D. Lidar, and M. Spiropulu, “Solving a higgs optimization problem with quantum annealing for machine learning,” Nature, vol. 550, no. 7676, pp. 375–379, 2017.

[6] I. Shapoval and P. Calafiura, “Quantum associative memory in hep track pattern recognition,” EPJ Web of Conferences, vol. 214, p. 01012, 2019. [Online]. Available: http://dx.doi.org/10.1051/epjconf/201921401012

[7] J. Preskill, “Quantum Computing in the NISQ era and beyond,” Quantum, vol. 2, p. 79, Aug. 2018. [Online]. Available: https://doi.org/10.22331/q-2018-08-06-79

[8] G. Aleksandrowicz, T. Alexander, P. Barkoutsos, L. Bello, Y. Ben-Haim, D. Bucher, F. J. Cabrera-Hern´andez, J. Carballo-Franquis, A. Chen, C.- F. Chen, J. M. Chow, A. D. C´orcoles-Gonzales, A. J. Cross, A. Cross, J. Cruz-Benito, C. Culver, S. D. L. P. Gonz´alez, E. D. L. Torre, D. Ding, E. Dumitrescu, I. Duran, P. Eendebak, M. Everitt, I. F. Sertage, A. Frisch, A. Fuhrer, J. Gambetta, B. G. Gago, J. Gomez-Mosquera, D. Greenberg, I. Hamamura, V. Havlicek, J. Hellmers, Ł. Herok, H. Horii, S. Hu, T. Imamichi, T. Itoko, A. Javadi-Abhari, N. Kanazawa, A. Karazeev, K. Krsulich, P. Liu, Y. Luh, Y. Maeng, M. Marques, F. J. Mart´ın-Fern´andez, D. T. McClure, D. McKay, S. Meesala, A. Mezzacapo, N. Moll, D. M. Rodr´ıguez, G. Nannicini, P. Nation, P. Ollitrault, L. J. O’Riordan, H. Paik, J. P´erez, A. Phan, M. Pistoia, V. Prutyanov, M. Reuter, J. Rice, A. R. Davila, R. H. P. Rudy, M. Ryu, N. Sathaye, C. Schnabel, E. Schoute, K. Setia, Y. Shi, A. Silva, Y. Siraichi, S. Sivarajah, J. A. Smolin, M. Soeken, H. Takahashi, I. Tavernelli, C. Taylor, P. Taylour, K. Trabing, M. Treinish, W. Turner, D. Vogt-Lee, C. Vuillot, J. A. Wildstrom, J. Wilson, E. Winston, C. Wood, S. Wood, S. W¨orner, I. Y. Akhalwaya, and C. Zoufal, “Qiskit: An Open-source Framework for Quantum Computing,” jan 2019. [Online]. Available: https://zenodo.org/record/2562111

[9] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, M. S. Alam, S. Ahmed, J. M. Arrazola, C. Blank, A. Delgado, S. Jahangiri, K. McKiernan, J. J. Meyer, Z. Niu, A. Sz´ava, and N. Killoran, “Pennylane: Automatic differentiation of hybrid quantum-classical computations,” 2020.

[10] W. Guan, G. Perdue, A. Pesah, M. Schuld, K. Terashi, S. Vallecorsa, and J.-R. Vlimant, “Quantum machine learning in high energy physics,” Machine Learning: Science and Technology, vol. 2, no. 1, p. 011003, mar 2021. [Online]. Available: https://doi.org/10.1088/2632-2153/abc17d

[11] Di Sipio, R., Giannelli, M., Haghighat, S. and Palazzo, S., 2019. DijetGAN: a Generative-Adversarial Network approach for the simulation of QCD dijet events at the LHC. Journal of High Energy Physics, 2019(8).

[12] Belis, V., González-Castillo, S., Reissel, C., Vallecorsa, S., Combarro, E., Dissertori, G. and Reiter, F., 2021. Higgs analysis with quantum classifiers. EPJ Web of Conferences, 251, p.03070.

[13] Chang, S., Herbert, S., Vallecorsa, S., Combarro, E. and Carminati, F., 2021. Quantum Generative Adversarial Networks in a Continuous-Variable Architecture to Simulate High Energy Physics Detectors. arXiv:2101.11132

[14] Chang, S., Herbert, S., Vallecorsa, S., Combarro, E. and Duncan, R., 2021. Dual-Parameterized Quantum Circuit GAN Model in High Energy Physics. EPJ Web of Conferences, 251, p.03050.

[15] F. Bezrukov, M. Shaposhnikov, Journal of Experimental and Theoretical Physics 120, 335–343 (2015)

[16] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2020. Generative adversarial networks. Communications of the ACM, 63(11), pp.139-144.

[17] Killoran, N., Bromley, T., Arrazola, J., Schuld, M., Quesada, N. and Lloyd, S., 2019. Continuous-variable quantum neural networks. Physical Review Research, 1(3).

* Image Credit: Google ML
** Image Credit: TensorFlow