Google Summer of Code Project Proposal 2024

Adithya Penagonda

March 31, 2024

Implementation of Quantum Generative Adversarial Networks to Perform High Energy Physics Analysis at LHC

adithyapenagonda@gmail.com
+91-6302 722 627
VNR Vignana Jyothi Institute of Technology, Hyderabad
Bachelor of Technology, Computer Science (Artificial Intelligence and Data Science)
India/GMT +5:30
Medium *GitHub* *LinkedIn*
To give feedback, please contact the email address.

Abstract

Generative Adversarial Networks (GANs) have transformed unsupervised machine learning by producing identical data. Jets—particle showers produced by high-energy collisions—provide insight into the behavior of gluons and quarks, the fundamental constituents of matter, in the realm of HEP. However, due to the massive amounts of data generated by experiments such as the Large Hadron Collider (LHC), traditional GANs have computational and scalability issues. Quantum Generative Adversarial Networks (QGANs) take advantage of quantum physics to promise faster training times while using fewer resources. The project aims to implement different GAN architectures ranging from classical models to fully quantum models, including hybrid models as well. And compare their performance with the goal of proving the quantum advantage. We work with different datasets that belong to High energy physics to prove the practicality of these quantum models in the NISQ era.

1 Introduction

Discovery of new physics phenomenon requires identification of rare signals against immense backgrounds. Machine learning techniques have helped quite a bit to identify these phenomenons. LHC generates 15PB of data annually, to put this in perspective, a 4K resolution image typically contains around 8.3 million pixels, so you would need around 1.8 million 4K images to represent 15 PB of data. Current computational resources would not be able to keep up with the increasing volume of data and Quantum algorithms offer significant advantages over classical algorithms by providing the potential for exponential speed-ups in numerical simulations. One such algorithm that promises to show an advantage over it's classical counterpart is Quantum Generative Adversarial Networks.

2 Datasets

2.1 Photon-Electron Electromagnetic Calorimeter (ECAL) Dataset

The dataset contains images from two types of particles: photons (0) and electrons (1) captured by the ECAL detector.

- Each pixel corresponds to a detector cell.
- The amount of energy measured in that cell is correlated with the pixel's intensity.

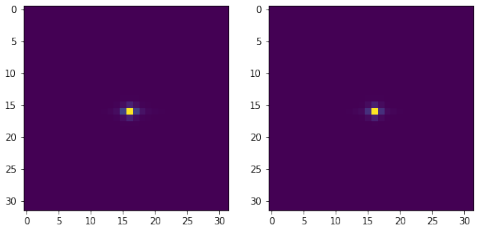

Figure 1: Averages of Electron (left) and Electron (right) image samples.

- In total, there are 498,000 samples, equally distributed between the two classes.
- The size of the images are 32x32.

2.2 Quark-Gluon Dataset

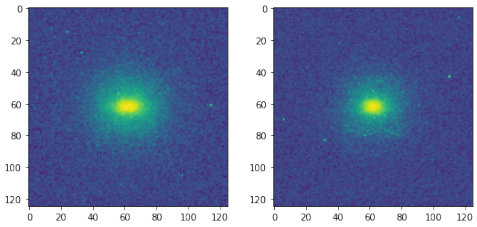

Figure 2: Averages of Gluon (left) and Quark (right) image samples of the track channel.

The dataset contains images of simulated quark and gluon jets. The image has three channels, the first channel is the reconstructed tracks of the jet, the second channel is the images captured by the electromagnetic calorimeter (ECAL) detector, and the third channel is the images captured by the hadronic calorimeter (HCAL) detector.

- The images have a resolution of 125 x 125 pixels (for every channel).
- The original size of 125 x 125 pixels is too large for quantum computing simulation, we cropped the images into certain size. For now, we limit the current size to 40 x 40 pixels.(recommended by Ericardo Muten)

2.3 Jet Mass Image Dataset

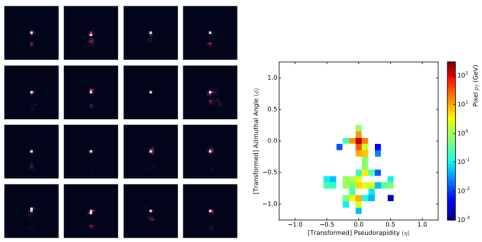

Figure 3: A sample of jet images(left) and a normalised jet image(right).

Dataset contains 872666 jet images, Simulated using Pythia 8.219 at $\sqrt{s} = 14$ TeV

- Each images is a 25x25 containing pixel intensities
- 'Signal' represents binary array to identify signal (1, i.e. W boson) vs background (0, i.e. QCD)

Figure 4: An image sample for each class from MNIST dataset.

2.4 MNIST Dataset

The dataset contains images of grayscale (8 bit) handwritten digits, 28 x 28 pixels, has a training set of 60,000 examples, and a test set of 10,000 examples. This dataset will be used for validating all the model.

3 Pre-Processing and Data Encoding

3.1 Dimensionality Reduction

In the era of Noisy Intermediate Scale Quantum(NISQ), the number of qubits we can utilize is constrained. We have to reduce the total number of features while keeping the variance high. Having a low variance would reduce the diversity present in the data while training GANs. Another major issue would be mode collapse when the generator fails to capture the complete diversity. One common way to reduce dimensions is to use Principal Component Analysis (PCA), By using the cumulative sum of all principal components' eigenvalues, we can find out how much of variance is left in the dataset.We can further reduce dimensions by cropping images, Images from Quark-Gluon Dataset can be cropped to 40 x 40 instead of using the original 125 x 125 images. Pooling is another proven way to reduce the number of features without much loss of information. We would be using different techniques based on the dataset.

3.2 Data Encoding

Classical data must be encoded into quantum states, for quantum machines to work on. There are numerous data encoding techniques but among QGANs, basis encoding, amplitude encoding and angle encoding are most popular[]

3.2.1 Basis Encoding

This is the simplest way to convert classical data into quantum states where binary inputs are converted into corresponding computational basis of the qubit system. Suppose a collection of inputs D consists of M binary strings x^m of length n. Then, all inputs in D can be embedded in the quantum state

$$|D\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |x^m\rangle$$

This is not a recommended method for large dataset, so we wouldn't be preferring this.

3.2.2 Amplitude Encoding

This method can embed N-dimensional classical data points into quantum states and each dimension of the data point becomes the amplitude of a corresponding basis state. A data point $x = (x_k)$ for $k = 1, ..., N$ is encoded as

$$|x\rangle = \frac{1}{\sqrt{\sum_{k=1}^{N} x_k^2}} \sum_{k=1}^{N} x_k \, |i_k\rangle$$

where $|i_k\rangle$ is the k-th computational basis. We need at least $log_2(N)$ qubits to encode N Dimensions. Though amplitude encoding is sensitive to noise, it might be a very usable method of encoding for our project scope due to much lower qubit requirement.

3.2.3 Angle Encoding

Each dimension of data point $x = (x_k)$ for $k = 1, ..., N$ becomes the angle of a rotation gate $R(.)$

$$|x\rangle = \bigotimes_{k=1}^{N} R(x_k) |0\rangle .$$

Here the data must be normalised between 0 and 1, and then transform a data point x into an angle $\theta = 2arcsin\sqrt{x}$, which would go through R_y gates. For my previous experience training images on QGANs, this was the best method to encode data and we would preferably using this method of encoding.

4 Classical GANs

Generative Adversarial Networks is an algorithms that works on optimizing two competing neural networks to produce new data distributions similar to the one they were trained on. Though the scope of project is limited to testing Quantum models, it would be ideal if we explore classical models as well. Classical GAN models have been mainstream since 2016, and they have proven to produce exceptional images. While training them with large datasets they often run into problems like mode collapse, vanishing or exploding gradients, mode dropping, slow convergence, etc. Here are two models that I believe would make a good benchmark for comparing with the quantum models.

4.1 Vanilla GAN

This is the simplest form of GAN, which involves a generator and discriminator. The generator learns to map random noise to data samples, aiming to generate realistic data. The discriminator learns to distinguish between real and generated data.

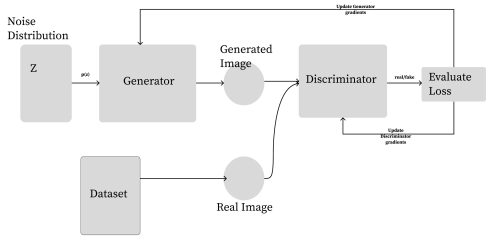

Figure 5: Architecture of Vanilla GAN

This will be our foundation model for comparison, From my knowledge this model might have high chances of mode collapse or overfitting.

4.2 Deep Convolutional GANs

DCGANs use convolutional neural networks (CNNs) in the generator and discriminator architectures. The pooling layers are replaced by strided convolutions (discriminator) and fractional-strided convolutions (generator). BatchNorm is used in both the generator and discriminator, which makes it a really good for large image datasets.

We will be testing our Quantum models on this architecture as well.

5 Quantum GANs

When it comes to Quantum Generative Adversarial networks, there are several ways to build them. They may be fully quantum, hybrid, or based on a tensor network.

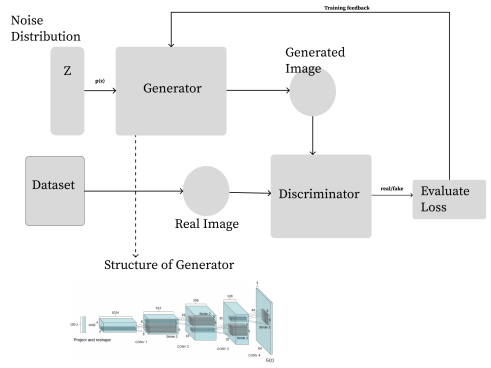

Figure 6: Architecture of DCGAN

5.1 Hybrid Quantum GANs

Hybrid Quantum Architectures combine quantum and classical computing to leverage strengths of each, We can either have a quantum generator or a quantum discriminator paired with a classical generator or a classical discriminator respectively.

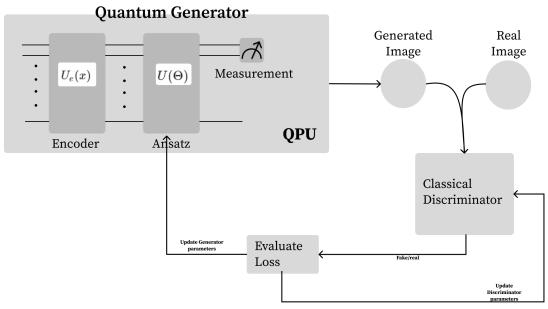

Figure 7: An example architecture of Hybrid QGAN with quantum Generator.

A quantum generator consists of encoder and a parameterized quantum circuit(PQC) that generates quantum states based on a set of parameters. These parameters will be updated iteratively during training to improve the model. While the Classical discriminator evalutes the quantum states generated by the quantum generator and determines if they are real or fake and update the parameters accordingly. We train both generator and discriminator to achieve Nash equilibrium. We would be training these models for all the datasets and compare them with classical and fully quantum models.

5.2 Fully Quantum GANs

Hybrid QGANs have atleast one component, either generator or discriminator which are not exactly compatible, meaning the discriminator cannot be classical if the data for training is generated by a quantum system, which in turn has a quantum advantage. A fully quantum GAN would demostrate that Quantum Computing can be the new paradigm. According to the description given, we have two reference papers. One architecture is EQGAN and IQGAN. EQGANs have been implemented by previous contributors(Amey Bhatuse), at GSoC 2022 so I'll mainly be focusing on IQGAN.

The IQGAN architecture introduced a trainable multiqubit quantum encoder, while having a compact quantum generator that reduces circuit depth by quite a bit. This architecture is proven to have produce high quality outputs while having low computational costs. The authors used Angle encod-

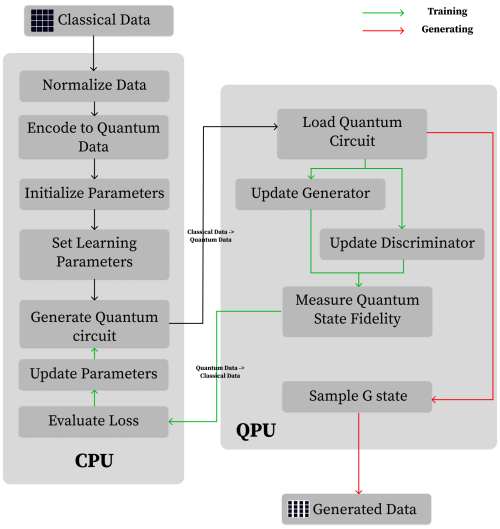

Figure 8: A typical fully QGAN architecture.

ing and VQC circuit ansatz constructed by parameterized single qubit rotation gates and followed by nearest-neighbor coupling of qubits using fixed two-qubit CNOT gates. The paper proposed a new version of angle encoding, where the variational encoder S($arcsin(x.\theta_s)$) and also improved the generator architecture to remove two-qubit entanglement gates to make it more efficient. As per the ref[],

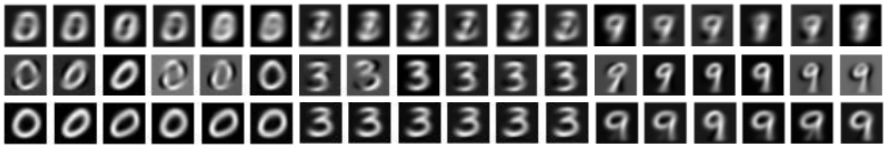

Figure 9: MNIST images generated by QuGAN (1st row), EQ-GAN(2nd row), and IQGAN (bottom row) (image from original paper)

IQGAN shows to perform better than both QuGAN, EQ-GAN. If the mentors also wish to implement EQGAN or suggest other models, we can add them as well.

5.3 Tensor network based QGANs

Huggins et al. introduced a new type of QuGAN architecture based on Tensor Networks. The idea is to factorize a high-rank tensor that describes a multi-partite system's quantum wavefunction over low-rank tensors. This is distinct from fully quantum or hybrid architectures, and it provides a unified framework in which classical environments can be transferred into quantum environments for optimization without modification. After consulting with the mentors, we can decide whether or not to implement this architecture for comparison purposes.

6 Optimization and Training

6.1 Loss Function

Loss function helps trace gradients, so that we can adjust model weights to reach Nash equilibrium. Generally we use the output of the discriminator to evaluate loss.

1. Maximum Likelihood, mainly used in classical GANs is one of the most popular loss functions. Just like traditional GANs there are potential issues like quick convergence, which leads to insufficient gradients for the generator to learn because the loss function saturates. Another potential issue is that this cost function does not help in overcoming with mode collapse. This still makes for a good choice to start with.

2. Wasserstein Distance, was converted to a quantum Wasserstein semi-metric by Chakrabarti et al. It is defined between two states $P \in D(x)$ and $Q \in D(y)$ is

$$qW(P,Q) = \min_{\pi} \mathrm{Tr}(\pi C)$$

. This loss function can be utilised as well.

3. Total variation, usually used as a metric for evaluating the performance of a network. The reference[] used it as a cost function. The total variation can be written as the trace of the target quantum state p_R and generated quantum state p_G:

$$D(p_R, p_G) = \frac{1}{2}\mathrm{Tr}(p_R - p_G)$$

where the network converges as the distance reaches 0. This is totally experimental and based on mentors opinion we can discuss.

6.2 Gradients

Parameterized quantum circuits consists of various quantum gates and unitaries with tunable continous parameters. Classical models usually use optimizing algorithms like gradient descent, Adagrade or Adam. A few methods include Parameter shift Method and Linear Combination of Unitaries.

6.3 Evaluation

To evaluate the performance of the GAN, we can use various metrics. A few include Kullback-Leibler divergence, Jensen-Shannon Divergence, Wasserstein distance can be used. State fidelity is also a popular metric where a swap test is used to find the difference between two quantum states. In my previous work, I've used KL divergence and it showed plausible results.

7 GSoC Timeline

7.1 Application Review period (April 2 - May 1)

Try to learn about the Pennylane framework, Qiskit, Tensorflow Quantum, and Cirq. Learn about the current state of Quantum Machine Learning and Quantum Advantage while also gaining practical experience by reviewing and implementing previous work.

7.2 Community Bonding (May 1 - May 26)

7.2.1 Week 1 (May 1 - May 8): Review and Plan

Discuss the entire proposal with the mentors, get feedback, and revise the required goals and deliverables. Finalize the datasets and models on which I will be working, and make the necessary changes. Also, discuss other details such as loss functions, gradient algorithms, optimizer and evaluation networks, as well as the hardware I own versus cloud platforms for execution and changing the timeline.

7.2.2 Week 2-3 (May 8 - May 17): Pre-Processing(Quite literally)

Prepare the working environment by installing all packages without any dependency issues. In a jupyter notebook, perform all of the pre-processing steps, including cropping or pooling, applying PCA, and normalizing the values.

7.2.3 week 3 (May 18 - May 26): Bonding!

I'd like to learn more about the mentors, organizations, and their work. Potentially network with other contributors.

7.3 Let's get coding(May 27 - Aug 26)

7.3.1 Week 1 (May 27 - Jun 3): Functional Dependency

Define all necessary functions for training, testing, and visualization. Decide on all performance metrics with your mentors and code them as well. As of now, I'm going for model accuracy and AUC score.

7.3.2 Week 2 (Jun 3 - Jun 10): Classical stuff

Implement the two classical models described in Section 4 (Classical GANs). I'm familiar with these models, so building them shouldn't take long. The plan is to implement Vanilla GAN first, followed by Deep Convolutional GAN, and then compare the two. Test these models on the MNIST dataset to ensure that no-mode collapse occurs, then apply them to the three datasets defined in Section 2 (Datasets), evaluate the results, and use them as benchmarks. Get feedback on the work style.

7.3.3 Week 3 - Week 6 (Jun 10 - Jul 1): Hybrid mode

In Section 5.1, we discussed having both a classical generator and a quantum discriminator. The first possibility is ruled out because a classical generator cannot learn a quantum target distribution. So we'll be working with a quantum generator and a classical discriminator. I'll need to practice a variety of encoding techniques, particularly angle or amplitude encoding. Try out different ansatz.
Once the model is functional, run it on the MNIST dataset. If everything goes well, we can try it on other datasets as well. Otherwise, we would have fine-tuned the hyperparameters. In my previous experience, hybrid GANs would only begin to converge after approximately 1000 epochs. Mentors would be given a weekly update on the work, and the approach would be updated accordingly.

7.3.4 Week 7 (Jul 1 - Jul 8): the Hybrid test

Visualise all results, generate new outputs, and validate all metrics for the hybrid model. Finish the work and compare it to both classical models.
Also, submit a report prior to the initial evaluation. Mentors can review all work.

Mid-Term Evaluation

7.3.5 Week 8 - Week 11 (Jul 8 - Jul 29): Full Quantum

Start working on the crux of the project, the fully quantum GANs. As of now my thought is to implement IQGANs but mentors can suggest other architectures as well. A thought has to be given around three main factors which are Loss function, Gradient and Network evaluation. I've gone through some papers, here is the summary of them.

Year	Name	Loss Function	Gradient	Network Evaluation
2019	Quantum Wasserstein Generative Adversarial Networks	Quantum Wasserstein distance with regularizer	Parameter Shift	State Fidelity
2021	QuGAN: A Quantum State Fidelity based Generative Adversarial Network	Log-likelihood	Parameter Shift	Hellinger distance
2022	Learning Quantum Data with the Quantum Earth Mover's Distance	Quantum Wasserstein distance	Parameter Shift	State Fidelity

Table 1: Example of different QGANs

I would be implementing IQGAN and possibly try to tweak it by using a different network evaluation or a different metric to measure convergence like KL divergence. Our goal here is to prove quantum advantage.

Some classical models may need to be retrained because their number of trainable parameters is too small or too large in comparison to the best IQGAN model. To ensure fairness, we want to compare models with roughly the same number of trainable parameters.

7.3.6 Week 12 (Jul 29 - Aug 13): The final buildup

Verify the completeness of the fully quantum model and finish up with all the hyper-parameter tuning as we need models to benchmark. Output all the required visualizations and compare it with both classical and hybrid models, notably with the number of parameters

7.3.7 Week 13 (Aug 13 - Aug 20): Write, Write, Write...

Finish writing the documentation, Make comments and markdowns in the notebook so that the mentors can verify and maybe other potential contributors find it easy to add.

Add everything to GitHub as well with a proper readme file. The documentation would include everything from the project motivation to all the datasets used, different models tested and results of all these models along with visualizations.

7.4 Students Submit Code and Final Evaluations(Aug 20 - Aug 26)

7.4.1 Week 1 (Aug 20 - Aug 26): Almost done

Use this time to complete all expected changes from in the documentation as per the mentors' feedback. This would be more of a buffer week in case the timeline goes further.

8 About me

I am Adithya Penagonda, a junior year engineering student at VNRVJIET in Hyderabad, India. I began coding at my freshman year and was always fascinated by machine learning; I self-taught myself using every resource available to me. During my sophomore year, I interned with a company to help them develop a chatbot. The majority of it was built with scalability in mind, which I accomplished using Azure. Speaking of Azure, I'm also a Microsoft Learn student ambassador. I'm in charge of leading a technical student community at my campus.

Nevertheless, it required a full year before I could at last contribute to ML4SCI. I completed the IBM Qiskit summer school and afterwards worked on several projects involving the use of quantum annealing, quantum phase estimation, and QAOA. I completed the identical project to which I will return this summer. I have been working on QGANs, trying to prove that there is no mode collapse by implementing a fully quantum GAN with the jet mass image dataset. My entire research was presented at the ANTIC conference at IIT BHU in the paper "Enhanced Simulation of Collision Events Using Quantum GANs for Jet Images Generation." Among India's top universities for technical research are the IITs. Only a small portion of the project I'm proposing has been worked on by me. Even though I completed my assignments last year, I thought it was crucial to retake the most of them and get better results. It would also be a great addition to my career to work for ML4SCI at GSoC' 24.

You can find the tasks on my GitHub under GSoC-QMLHEP-Tasks.