Google Summer of Code Project Proposal 2023

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

Adithya Penagonda
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.

Contents

1 Overview

1.1 Project Abstract

In many scientific domains, rare signals can be hidden within immense background noise, and identifying these signals is a critical task. In High-Energy Physics (HEP), identifying rare signals is important in the search for new particles or phenomena. Classical machine learning techniques such as Convolutional Neural Networks (CNNs) are effective for image classification, but Quantum Generative Adversarial Networks (QGANs) are a promising approach for identifying rare signals within backgrounds. As quantum computers promise many advantages over classical computing, comes the question of whether quantum machine learning (QML) can give any improvement in solving the problem.

This project aims to demonstrate quantum machine learning's potential, specifically the use of an Improved QGAN (IQGAN) for identifying rare signals in HEP datasets. We will concentrate on finding novel particle physics signals in the massive background noise of particle collision images by creating new data using QGANs. The IQGAN will be developed and trained to produce samples that are statistically similar to the rare signals but closely resemble the background.

1.2 Benefits to Community

Quantum Machine Learning is still in its infancy, and finding rare signals in High Energy Physics (HEP) might help researchers detect and learn new things about particle physics. Huge amounts of data are gathered during HEP experiments, and finding uncommon signals is essential for discovering new particles or phenomena. The ability to spot rare signals in noisy data, for instance, could be used in industries like finance where spotting unusual data patterns is essential to spotting fraud or other financial crimes.

The open-source community will benefit from both the study conducted and the source codes produced during the project. The community can use the source codes as a starting template for further investigation. Additionally, this project serves as a learning resource for those interested in the field because the documentation will also be written in a tutorial-like format.

I also believe that a good contribution to the open-source community uses existing open-source projects. I would be using libraries like TensorFlow, Pennylane, Keras, TensorFlow Quantum, Cirq, Scikit-Learn, etc.

2 Goals and Deliverables

2.1 Goals

Based on the project description page, this project aims to explore and demonstrate that quantum computing can be the new paradigm (Proof of Principle) for High Energy Physics. In addition, the project also would like to develop Quantum Machine Learning methods for the Large Hardon Collider High Energy Physics analysis based on e.g., the Pennylane framework. That will enhance the ability of the HEP community to use Quantum Machine Learning methods.

Considering the organization and the previous work done it can be concluded that the project's focus would be more toward scientific research rather than software or framework development.

2.2 Deliverables

Considering the goals and the time available, I sectioned the deliverables into two categories: required and optional. Required consists of the deliverables that must be finished during GSoC and are the project's main aims, while optional refers to the ones that will be completed if time permits or will be left for further development.

Required

1. Python code(s) that contains:
   - Code to train a QGAN model (including the fine-tuning) with several different architectures (IQGAN or HuQGAN).
   - Code to train a classical machine learning model and test it on the MNIST dataset.
   - Code to compare the performance of the IQGAN models to the classical machine learning models.

   The quantum programming frameworks that will be used for all the codes are Cirq[1] and TFQ[2] or Pennylane[15].
2. Train QGAN models (including the fine-tuning) using tutorial-like Jupyter Notebook(s) that used the Python codes above. This notebook doubles up as the documentation for code like double-Higgs production.
3. Compare the quantum machine learning performance to the classical machine learning performance.
4. A white paper contains a summary of previous related works, an explanation of the QGAN model implementation and how it is trained, and an analysis of their performance compared to the classical machine learning models. This white paper also doubles as GSoC final report.

Optional

1. If the project's final results are considered published-worthy by the mentors, the summary sheet will be documented as a research paper to be submitted to a conference, poster, talk/workshop, or journal.

3 Related Works and Recommendations

3.1 Related Works

This project has two essential parts: the dataset (including pre-processing) and the circuit architecture. The circuit architecture itself consists of a classical data encoding circuit and a variational circuit. This section will discuss some related works and my experience (including the experience from working on the evaluation test).

3.1.1 Dataset and Pre-processing

Dataset

We would prefer to conduct experiments on a well-known dataset since that would ensure a large number of papers to draw ideas from and compare the effectiveness of our models. Since the dataset(s) to be used are not listed on the project description page, I assumed that the applicants were free to suggest and go over project ideas with mentors.

CERN also has provided some public LHC datasets. The dataset that I believe would be appropriate for this project is Pythia Generated Jet Images for Location Aware Generative Adversarial Network Training[4]. This dataset is the $25x25$ simulated jet images that contain one of the two classes, either signal (i.e., W boson) or background (i.e., QCD) [4]. This dataset was used to train a GANs model to reproduce the images. Figure 1 shows an example of images from this dataset. There is another dataset that could for further complex is the Data for 3D jet images from [6].

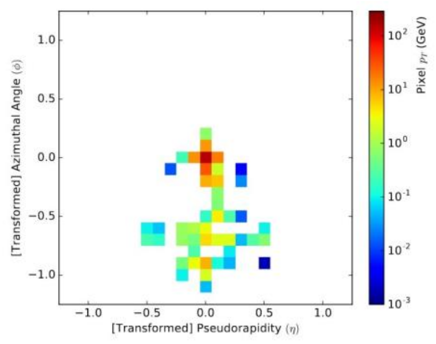

Figure 1: An example of images from [4].

As far as this preliminary investigation is concerned, I think the best dataset for this project would be images of particle jets (or a dataset that is similar to it). I will also be using MNIST dataset for verifying the model and that no mode collapse happens.

Dimensionality Reduction

Dimensionality reduction is essential because the number of qubits we can use is constrained. The ideal approach is to keep the dataset variance as high as possible while reducing the total number of features (pixels) to about 10–20 features. One of the simplest methods may be to crop the image's edge, but this is very dataset-dependent because the edge of the image likely still has significant pixel values. Additionally, even after cropping, we frequently still have images that are too large and require additional processing. Bilinear interpolation from TensorFlow is employed to decrease the size of the image in the TensorFlow MNIST Classification tutorial (and [11]).

Skolik et al.[7] and Mardirosian[8] have tried to use Principal Component Analysis (PCA) to reduce the dimension of the MNIST dataset[9] before classifying it with a quantum classifier, I observed that PCA is quite efficient. By calculating the cumulative sum of all used principal components' eigenvalues, we can quickly determine how much variance percentage is still present in the reduced dataset. There will be a trade-off between the variance in the dataset and the number of remaining features. It would be ideal to have a small number of features, but this could result in a dataset with too little variance for the classifier to distinguish between the classes. This needs to be adjusted while building the model.

Normalization

Depending on the data encoding technique that will be used, the values of the dataset's features must also be normalized to a specific range. One of the most popular methods in the QML community is angle encoding, which treats the feature value as the rotation angle for a rotation gate. Normalization is not subject to rigid regulations. The value was normalized in some studies to [-pi, pi], [-1, 1], or [0, 1]. My experience has shown that neither of them is bad nor has little impact on performance.

One thing is certain: since it is a rotation angle, the normalization range must not go beyond [-pi, pi]. Making the length of each sample's feature vector equal to 1 is another way to normalize the value. This is particularly important for amplitude encoding. When working with high-dimensional datasets, amplitude encoding is very efficient. $\log_2(N)$ qubits, where N is the total number of features, are all that is required. The total number of features shouldn't be excessive given that PCA will be used, and amplitude encoding might not be required. Finding the normalization that will perform best is not simple because it depends on the dataset and the rest of the architecture. Additionally, this needs to be adjusted while building the model.

3.1.2 Circuit Architecture

Two parts of the circuit will be trained in this project: the Quantum Generative Adversarial Network part and the classification part. I would be majorly using the reference[3].

Quantum Generative Adversarial Part

The paper describes a circuit ansatz used in mainstream Quantum Neural Networks (QNNs) called the VQC Circuit Ansatz, $G(\vartheta g)$. This circuit ansatz is constructed by parameterized single-qubit rotation gates followed by nearest-neighbor coupling of qubits using fixed two-qubit CNOT gates. The circuit is illustrated in Figure 1 of the paper.

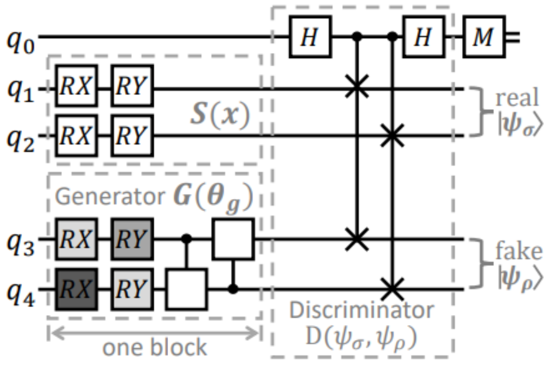

Fig. 1. A standard QGAN (*shaded gates are trainable*).

The purpose behind this design is that while two-qubit CNOT gates offer maximum entanglement between the two target qubits, single-qubit rotations offer a way to parameterize circuits. In several applications, this circuit ansatz has proven to have the superior expressive capability. Classical data is converted into quantum by using Angle Encoding S(x) as this is a widely used method in QNNs [10, 11, 12, 13], and was also used in task IV.

$D(|\psi\sigma\rangle,|\psi\rho\rangle)$ is the discriminator, and uses a SWAP test which is a standard quantum module used for quantum fidelity measurement. The SWAP test circuit consists of one ancillary qubit q0, two Hadamard gates (i.e., H gate), and several controlled SWAP gates (i.e., CSWAP gate) that interchange the two quantum states under test only if q0 is in state $|1\rangle$. The ancillary qubit q0 is finally measured on the z-basis, and the probability it yields a measured output $|0\rangle$ is defined as quantum fidelity $P0 = 1 + \langle\psi\sigma|\psi\rho\rangle/2^\wedge 2$.

The parameter matrix ϑg is trained to minimize a fidelity-based loss shown in equation 1[14].

$$\min_{\theta_g} L(\theta_g) = \min_{\theta_g}[1 - \langle\psi_\sigma|\psi_\rho(\theta_g)\rangle^2] \qquad (1)$$

In MNIST dataset classification, IQGANs performed extremely well in comparison to QuGAN[10] and EQGAN[14]. The following images are generated by these models.

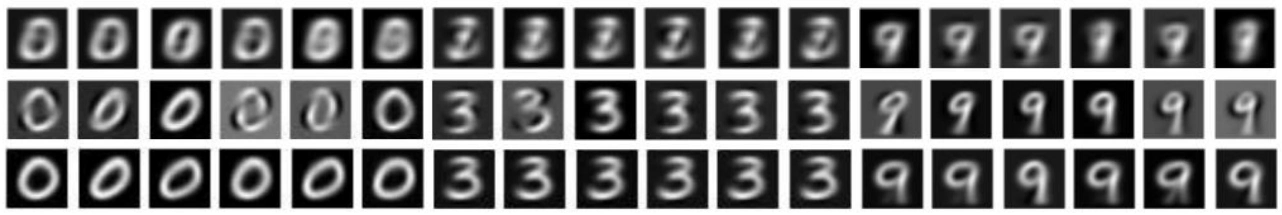

Fig. 2. MNIST images generated by QuGAN 2021 [10] (1st row), EQ-GAN [14] (2nd row), and IQGAN (bottom row)

IQGANs can produce synthetic jet images that are similar to real jet images but have a higher proportion of signal events, which can be used to separate signals from background events in jet images. A more balanced dataset can be created by combining these artificial images with the original dataset, which will improve the training of machine learning algorithms. Additionally, IQGANs can be used to create images with various signal-to-background ratios, giving researchers a way to examine the algorithms' performance in various scenarios. I would be using Pennylane[15] framework or TensorFlow Quantum to implement this, the original paper used Pennylane.

Classification Part

Once we have trained our IQGAN, you can use it to generate new samples that resemble the rare signal we are interested in identifying, the method proposed is primarily inspired by reference [16]. I have divided this into several steps listed below:

1. **Generate Samples using IQGAN** – The usage of a QGAN in the first place is to overcome the lack of data on rare signals. Using the model above we can generate new samples that resemble original rare signals.
2. **Calculate the probability density function (PDF)** – We can estimate the PDF from the generated samples of rare signals. We can use either kernel density estimation or histogramming.
3. **Calculate the PDF of the background** – We also estimate the PDF of the background from the original data. This will represent the distribution of background without any signal.
4. **Likelihood ratio** – Compare the PDFs of both background and signal to get a likelihood ratio, defined as

   $$LR = PDF_{signal(x)}/PDF_{background(x)}$$

   Where x is the sample, you want to classify. If LR is greater than 1, then x is more likely to be a signal. If LR is less than 1, then x is more likely to be a background.
5. **Set a Threshold** – Choosing the right threshold for the determination of the signal is the only thing left. We can either choose it based on the threshold based on the false positive rate (FPR) and false negative rate (FNR) we are willing to accept. Or we can set the threshold to maximize the F1 score, which is the harmonic mean of the precision and recall.

The above method does not utilize the quantum advantage in any way, but the proposal is more inclined toward the GAN part rather than classification.

Optimizer

A QML model cannot exist without an optimizer. Barren plateau is a well-known problem in QML, and selecting the best optimizer might result in a more effective model. For instance, it has been discovered that when optimizing VQEs, Quantum Natural Gradient converges faster and is more stable than regular gradient descent or Adam [17, 18, 19]. Despite how intriguing it is, I think there may not be enough time to fully explore the optimizer in the time allotted for this project. Hyperparameter tuning and experimenting with various ansatzes already take a lot of time.

The papers discussed used an efficient standard optimizer like Adam [20]. Therefore, using Adam or RMSprop for this project is acceptable in my opinion. The optimizer exploration for QML may become an interesting topic for the next cohort of GSoC.

3.2 Recommendations

Considering all things mentioned in section 3.1, I suggest trying both these approaches for this project:

1. First approach: Check to see if PCA works well for the dataset we've chosen. If yes, we proceed with IQGAN to generate signals. We train it on quantum data using the Pennylane framework or Tensorflow Quantum, as described in the paper [9]. The model is then tested against the MNIST dataset to ensure

that no mode collapse occurs. If we successfully trained the model, we could then use it to train the jet images. These generated images can be used for PDA-based classification. The results can be compared to existing classical models. If time allows, I would also like to propose that a classical model inspired by reference [21] be trained. Learning to generate very sparse images with deep generative models has been a difficult problem due to issues such as mode collapse and sparse gradient signals, so using the model from the paper may improve the results of classical GANs as well.

2. Second approach: If the above approach is not producing desired results, I would like to take another approach from the reference [22]. The paper discusses Hamiltonian Quantum Generative Adversarial Networks (HQuGANs), a new algorithm that generates unknown input quantum states using quantum optimal controls. The algorithm's game-theoretic framework is inspired by classical generative adversarial networks, and the quantum optimal control approach adapts the algorithm to experimental constraints and potentially improves convergence. According to the paper, after two rounds of interactions between two players, the generator will output a quantum state that is highly like the target state. The input in the model was an unknown quantum state generated from a black box. We could use our classical data by converting it to a quantum state by either amplitude encoding or angle encoding. Then later use the same classifier to identify new signals.

3. The model will be trained to classify images of particle jets. The dataset classes will be limited to two (binary classification) to shorten the training time.

4 Timeline and Research Plan

The timeline written here is based on the general timeline by GSoC.

4.1 Application Review Period (4 April - 4 May)

Try to become acquainted with the Pennylane framework, Tensorflow Quantum, and Cirq. Learn more about the current state of Quantum Machine Learning and Quantum Advantage while also gaining hands-on experience by reviewing and implementing previous work.

4.2 Community Bonding (4 May - 28 May)

The lined-up tasks can be boiled down into 3 major sections. The timeline may overlap between categories as the works between categories are closely related.

Week 1-2: Planning (4 May – 11 May)

Obtain feedback on the proposal from mentors. Discuss with mentors the high-level decisions in the proposal that need to be changed (goals, priority, deliverables, meeting schedule) and modify them accordingly. Following that, discuss with mentors the more low-level decisions (dataset and pre-processing, the entire pipeline including optimizers, ansatzes, and coding training procedures) down to the smallest details, and modify the timeline and research plan accordingly.

Week 2-3: Early Works (11 May – 20 May)

Prepare the working environment and repository. Obtain and clean the dataset. Do all necessary pre-processing (normalization and PCA) in Jupyter Notebook.

Checking to see if the dataset is appropriate for the task; if not, I will have to discuss other potential datasets with the mentor and complete the necessary pre-processing.

Week 3: Bonding (20 May – 28 May)

I'd also like to learn more about the mentors' and organizations' work outside of GSoC. My ambition is to become a researcher in Machine Learning, particularly generative AI, and this is a fantastic opportunity to learn from experts in the field.

4.3 Coding (29 May - 21 August)

Week 1 (29 May – 1 June)

Code all convolution circuit ansatzes that will be used in Jupyter Notebook or as a separate Python module that can be imported.

Week 2 (1 June – 8 June)

The classical machine learning techniques listed in section 3.2 can be programmed and trained in Jupyter Notebook. Train several times and save the performance statistics and models. When developing the IQGAN model, these performance statistics will be used as a benchmark. I would try to implement the model Sparse Image Generation with Decoupled Generative Model [22] based on time constraints.

Make any changes to the first approach report and notebook as per feedback.

Week 3 – 8 (8 June – 20 July) & Evaluations (10 July – 14 July)

This is the central and integral part of the project, developing the IQGAN model. I would spend about a month coding and developing good-performing models for my undergraduate thesis. Considering that the training pipeline code is already done in week 2, I believe a little more than a month should also be enough to train and fine-tune the model.

The strategy is to focus on one type of VQC circuit ansatz and try other possibilities based on the performance. The model from fig 1. Would be our major reference. After spending the first few weeks getting the right circuit I would spend time fine-tuning, which includes hyperparameters (number of epochs, batch size, learning rate), normalization, optimizer, etc.

If the model shows good performance with the potential to be better at the end of the week, we will spend another week fine-tuning it. If it does not perform well, we will move on to another VQC circuit ansatz the following week. The decision is made at the weekly mentoring meeting.

If the model is doing well, it will be tested with the MNIST dataset to confirm that no mode collapse happens. Since jets images are sparse matrices, it is extremely important to make sure that no mode collapse happens. I would be starting with training the model with the jet images [3] as soon as a model feels appropriate, this should probably take a good few weeks.

We will adhere to the strategy and training procedures discussed during Community Bonding, but it is difficult to plan for each step in the model development because there are many moving parts and every decision is dependent on the numerical results of the model.

Note that after every training, the configurations, the trained model, and that model's statistical performance will be saved. These results will be evaluated by the mentors during 10 July – 14 July.

Week 9 (20 July – 27 July)

Finish up the training part of the jet images and by the end of the week, we should successfully be able to generate images of desired signals.

Week 10 (27 July – 3 August)

Train a classifier based on the PDA of both the backgrounds from the original dataset and generated signals by our model and find the right threshold to identify new signals in the background. This will be further discussed with the mentor based on the use case and application.

Week 11 (3 August – 10 August)

After we have determined the best configuration of the IQGAN model, we will concentrate solely on that model and compare its performance to that of the classical models. 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.

Visualize the comparison, such as the training and testing loss curves, training and testing accuracy curves, and the ROC curve and AUC score of the final models. Calculate the best model's average accuracy and AUC score using both QCNN and traditional methods.

Create a performance comparison analysis based on the visualizations and consider how the IQGAN model could have been improved. This analysis will serve as the basis for the final report.

Week 11-12 (10 August – 21 August)

Clean up the Jupyter Notebook in preparation for IQGAN training. Make comments and markdowns in the notebook to explain everything you've done. This notebook will serve as documentation and a tutorial for people interested in trying IQGAN, similar to but more detailed than what I did for the evaluation test.

Write the final report/summary document. This report will contain:

1. Brief Introduction (project summary, motivation, related works, dataset). The project proposal's contents can be reused.
2. Explain the best IQGAN model implementation that is found during model development and how it was trained in minute details. Most of the contents are already written in the tutorial-like documentation made in week 11, so this is just a matter of moving and reformatting them.
3. Performance comparison analysis from week 11(no reformatting as it can be directly reused).
4. If the first approach gives good results, then the short report that was made will be integrated here as well.
5. Conclusions and future work recommendations.

4.4 Students Submit Code and Final Evaluations (21 August - 28 August)

Make any changes to the tutorial-like docs made in week 10 and the final report made in weeks 11-12 as per feedback.

Write final evaluation for mentors.

This is a spare week in case of some works in the timeline need more time, emergency events, etc.

5 About Me

I am currently pursuing a Bachelor of Technology in Computer Science and Engineering with a specialization in Artificial Intelligence and Data Science. I am confident in my programming capabilities. I use Python regularly and have experience in C++ and C.

I am a curiosity-driven individual, dedicating all my attention and zeal to things I am passionate about. Machine learning has been my favorite topic since freshman year, and it was the obvious choice for the course I took. In the following section, I briefly explained how I got into these fields and my experience in them. I am a quick learner who can excel in both group and solo settings.

While learning about ML, I also learned about other technologies that may or may not be relevant to this project, but to name a few, I know and understand databases such as SQL, web technologies such as React JS, Node JS, and other languages such as Java, C++, C. some of which were part of the course work and some of which were other external courses.

5.1 Classical Machine Learning

My journey into machine learning began in the middle of my freshman year. I used to enjoy coding and figuring out various data structures and algorithms, but I realized that this was not what the world needed. I was always fascinated by AI in general since high school, but I never really dug into it.

Based on my interest, I initiated my machine learning journey online and at my university. I spent three months on Coursera completing a Machine learning specialization course taught by Andrew NG, one of my favorite online teachers. Later, I began with simple projects involving SVMs, Regressions, Decision Trees, and so on.

I got to work on some real projects by learning and implementing neural networks with TensorFlow. This is when I started learning about CNNs, RNNs, NLP, Transformers, and other AI techniques, and I worked on simple projects like image captioning, image classification, and audio generation with RNNs. The real deal began when OpenAI released its DALL-E model, and I was blown away by the power of generative AI. I then learned more about generative models such as diffusion models, but one of the most intriguing was Generative Adversarial networks, which are responsible for deep fakes.

Then I worked on GANs, beginning with the Fashion MNIST and progressing to more complex image generation, such as faces. Another intriguing release was Google's AudioLM [23], which I'm currently working on. The concept of employing diffusion-based models for audio.

5.2 Quantum Machine Learning

QML was something I discovered later in my machine-learning journey. I decided to contribute to this organization in early February, and the moment I saw this project, I knew it was the one.

Quantum physics piqued my interest in high school, but I was never fully convinced until I learned how quantum computing has evolved so rigorously over the years. It was when I overheard a professor discussing cyber security and the threat it poses to data security, I explored it and understood the potential it held for the upcoming future.

The tasks have helped me understand how quantum machine learning works, and I am very confident that, I will be able to implement everything mentioned above. You can find the tasks over here – link.

6 Other Commitments

- My semester exams will be at the end of August. I'm fairly confident, I'll be able to cater time while devoting sufficient hours to the remaining work. I am in my sophomore year and will have no other significant responsibilities.

References

[1] Cirq Developers. *Cirq*. Version v0.10.0. See full list of authors on Github: https://github.com/quantumlib/ Cirq/graphs/contributors. Mar. 2021. doi: 10.5281/zenodo.4586899. url: https://doi.org/10.5281/ zenodo.4586899.

[2] Michael Broughton et al. *TensorFlow Quantum: A Software Framework for Quantum Machine Learning*. 2020. arXiv: 2003.02989[quant-ph].

[3] IQGAN: Robust Quantum Generative Adversarial Network for Image Synthesis On NISQ Devices. arXiv:2210.16857v2.

[4] Benjamin Nachman, Luke de Oliveira, and Michela Paganini. Pythia Generated Jet Images for Location Aware Generative Adversarial Network Training. Zenodo, Feb. 2017. doi: 10.17632/4r4v785rgx.1. url: https://doi.org/10.17632/4r4v785rgx.1.

[5] Luke de Oliveira, Michela Paganini, and Benjamin Nachman. "Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis". In: Computing and Software for Big Science 1.1 (Sept. 2017). ISSN: 2510-2044. doi 10.1007/s41781-017-0004-6. url: http://dx.doi.org/10.1007/s41781-017-0004-6.

[6] Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks.. Zenodo, May. 2017. doi: 10.5281/zenedo.584155 url: https://zenodo.org/record/584155#.ZCcawXZBw2w

[7] Andrea Skolik et al. "Layerwise learning for quantum neural networks". In: Quantum Machine Intelligence 3.1 (June 2021), p. 5. issn: 2524-4906. doi: 10.1007/s42484-020-00036-4. url:http://link. springer.com/10.1007/s42484-020-00036-4.

[8] Sevak Mardirosian. "Quantum-enhanced Supervised Learning with Variational Quantum Circuits". PhD thesis. Leiden University, 2019.

[9] Yann LeCun, Corinna Cortes, and CJ Burges. "MNIST handwritten digit database". In: ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist 2 (2010).

[10] Samuel A. Stein, Betis Baheri, Daniel Chen, Ying Mao, Qiang Guan, Ang Li, Bo Fang, and Shuai Xu, "QuGAN: A Quantum State Fidelity based Generative Adversarial Network," in IEEE International Conference on Quantum Computing and Engineering (QCE). 2021, pp. 71– 81, IEEE.

[11] Hanrui Wang, Yongshan Ding, Jiaqi Gu, Yujun Lin, David Z Pan, Frederic T Chong, and Song Han, "Quantumnas: Noise-adaptive search for robust quantum circuits," in IEEE International Symposium on HighPerformance Computer Architecture (HPCA). IEEE, 2022, pp. 692–708.

[12] Edward Farhi and Hartmut Neven, "Classification with quantum neural networks on near term processors," arXiv preprint arXiv:1802.06002, 2018.

[13] Ryan LaRose and Brian Coyle, "Robust data encodings for quantum classifiers," Physical Review A, vol. 102, no. 3, pp. 032420, 2020.

[14] Murphy Yuezhen Niu, Alexander Zlokapa, Michael Broughton, Sergio Boixo, Masoud Mohseni, Vadim Smelyanskyi, and Hartmut Neven, "Entangling Quantum Generative Adversarial Networks," arXiv preprint arXiv:2105.00080, 2021.

[15] Pennylane, "PennyLane," url: https://pennylane.ai/.

[16] Systematic Review of Unsupervised Genomic Clustering Algorithms Techniques for High Dimensional Datasets. Diyar Q Zeebaree
url:https://www.academia.edu/43377764/Systematic_Review_of_Unsupervised_Genomic_Clustering _Algorithms_Techniques_for_High_Dimensional_Datasets

[17] James Stokes et al. “Quantum Natural Gradient”. In: Quantum 4 (May 2020), p. 269. issn: 2521327X. doi: 10.22331/Q-2020-05-25-269. arXiv: 1909.02108. url: https://quantum-journal.org/papers/q2020-05-25-269/.

[18] David Wierichs, Christian Gogolin, and Michael Kastoryano. “Avoiding local minima in variational quantum eigensolvers with the natural gradient optimizer”. In: Phys. Rev. Research 2 (4 Nov. 2020), p. 043246. doi: 10.1103/PhysRevResearch.2.043246. url: https://link.aps.org/doi/10.1103/PhysRevResearch.2.043246.

[19] Naoki Yamamoto. On the natural gradient for variational quantum eigensolver. 2019. arXiv: 1909.05074 [quant-ph].

[20] Diederik P Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. Ed. by Yoshua Bengio and Yann LeCun. 2015. url: http://arxiv.org/abs/1412.6980.

[21] Yadong Lu, Julian Collado, Kevin Bauer, Daniel Whiteson, Pierre Baldi “Sparse Image Generation with Decoupled Generative Model”. url: https://ml4physicalsciences.github.io/2019/files/NeurIPS_ML4PS_2019_161.pdf

[22] Leeseok Kim, Seth Lloyd, Milad Marvian “Hamiltonian Quantum Generative Adversarial Networks”. arXiv:2211.02584.

[23] AudioLM: a Language Modeling Approach to Audio Generation. Link : https://ai.googleblog.com/2022/10/audiolm-language-modeling-approach-to.html