# Quantum Generative Adversarial Networks (Course Project Report)

Anuj Nayak
Instructor: Lav Varshney
Department of Electrical and Computer Engineering
University of Illinois, Urbana-Champaign

***Abstract*—In this report, we present a quantum analog of Generative Adversarial Network (qGAN) as discussed in [1]. The advantage of quantum systems in generative models is that they are capable of representing exponentially large state spaces, which are usually either extremely difficult or intractable with the classical approach. We discuss a quantum-enhanced version of GAN (termed as qGAN, proposed in [1]) containing a quantum generator and a classical discriminator. The key difference between the quantum generator and the classical generator is that the former learns the probability distribution of data explicitly (in other words, the output of the quantum generator itself is a probability mass function (PMF)), rather than learning a function to generate samples, which are distributed similar to the data. Also, the generative process boils down to repeated measurements - once control parameters are trained, the samples can be generated by performing repeated measurements. Further, we perform numerical simulations of qGANs using Qiskit Software Development Kit (SDK) [2], and also explore the effect of noise on the performance of qGANs with different circuit depths. Finally, we also comment on challenges in implementation, and limitations of the current qGAN models.**

***Index Terms*—Quantum machine learning, Generative Adversarial Networks, Quantum-enhanced machine learning, Random data loading.**

## I. Introduction

Today, we are in the Noisy-Intermediate Scale Quantum (NISQ) era, where researchers are exploring ways to improve computationally expensive classical problems using quantum computing, chasing the possibility of quantum advantage in various problems. Due to the current technological bottleneck in quantum computing and as natural transition towards quantum computation, there is an increased interest in quantum-enhanced classical computing, in other words solving classical problems with their sub-problems solved using quantum accelerators. In this regard, there is an increased interest in exploring whether machine learning problems can be solved faster using quantum computing engines. However, as of today, there is no known significant quantum advantage in solving machine learning problems, except for some special cases [3].

Recently, a quantum-enhanced Generative Adversarial Network is proposed in [4], where the classical generative part of GAN is replaced by a variational quantum circuit. In classical GANs, the generator learns a transformation to generate samples, whose distribution is similar to the data. In contrast, the quantum generator learns the underlying probability distribution (PMF) directly; for example, the generator takes uniform PMF as input to generate an output PMF that approximates the data distribution. The learnt quantum generator enables efficient data loading with $O(\text{poly}(n))$ space complexity compared to other previous methods which have a complexity of $O(2^n)$, where $n$ is the number of qubits [4]. Moreover, once the quantum generator is trained, generation of samples boils down to taking repeated measurements of the generator.Our report will be based on the qGAN paper [4] and thesis [1].

Computations performed using a quantum computing hardware are very sensitive (and susceptible) to noise resulting in erroneous outputs. Therefore, in current quantum computers, qubits operate in a strictly controlled conditions, isolated from the external environment. Even though there are efforts to mitigate this error, there is a need to analyse the performance of quantum algorithms under noise. In paper [5], the impact of measurement noise and two-qubit noise on the performance of qGANs. In this report, study the effect of Pauli error (bit-flip noise on single qubit gates) on performance of qGAN through simulations.

The rest of the report is organized as follows: The building blocks of qGANs are introduced in II, the qGAN architecture is described in Section III, where the generator, discriminator and loss functions are defined, Section IV describes noise in quantum circuits and possible ways to address it in the context of qGANs, Section V covers results of qGAN simulation in Qiskit, and finally, Section VI presents our conclusions and some of the open problems.

## II. Building blocks of qGANs

In this section, we introduce quantum data encoding, gradient computation and variational quantum circuits,

which are the building blocks of qGANs.

### A. Quantum Data Encoding

In quantum classification, basis encoding is not preferred since spatial information about the data samples could be lost. However, in variational models like qGANs, since our objective is to learn the probability distribution of the data, basis encoding is favourable, since quantum systems have the ability to represent exponentially large state spaces, which can serve as a support of the probability distribution.

### B. Quantum Gradients

The physics of quantum computation allows computations only in a feedforward manner [1]; hence, one cannot perform backpropagation to manipulate the parameters of the quantum generator. Therefore, one of the ways to compute gradients is by "first principles" approach, i.e., perform forward computation twice, first without perturbing any parameter and secondly, by perturbing (systematically) only the parameter for which we need to compute the gradient. This is performed for each parameter to obtain the gradients for the entire generator. The gradients obtained by this procedure are called *parameter-shift gradients*. Let $\hat{O}$ is the observable and $|g_{\boldsymbol{\theta}}\rangle$ be a pure quantum state parameterized by $\boldsymbol{\theta}$, then the gradient of $|g_{\boldsymbol{\theta}}\rangle$ with respect to the parameter $\theta_i$ is given by

$$\frac{\partial \langle g_{\boldsymbol{\theta}}| \hat{O} |g_{\boldsymbol{\theta}}\rangle}{\partial \theta_i} = \frac{1}{2}\Big(\langle g_{\boldsymbol{\theta}+\pi/2\boldsymbol{e}_i}| \hat{O} |g_{\theta+\pi/2\boldsymbol{e}_i}\rangle - \langle g_{\theta+\pi/2\boldsymbol{e}_i}| \hat{O} |g_{\theta+\pi/2\boldsymbol{e}_i}\rangle\Big), \quad (1)$$

where $\boldsymbol{e}_i$ a canonical basis vector where only $i^{th}$ element is 1 while the rest are zeros.

One can notice that the complexity of gradient computation grows linearly in the number of parameters. Moreover, one should also note that while performing computations in real quantum hardware, multiple measurements are necessary for noise averaging, which adds to the complexity in practice (however, asymptotic complexity is still $O(n_\theta)$, where $n_\theta$ is the number of parameters).

## III. qGAN Model

qGAN model consists of a generator which approximates data distribution through quantum computation, and the discriminator and the optimizer perform computations classically. A schematic diagram of the qGAN is shown in Figure 1.

[1]However, measurement-based feedback is possible, but it is not useful in our context, since gradients must be computed for every input, which is a superposition of basis states.

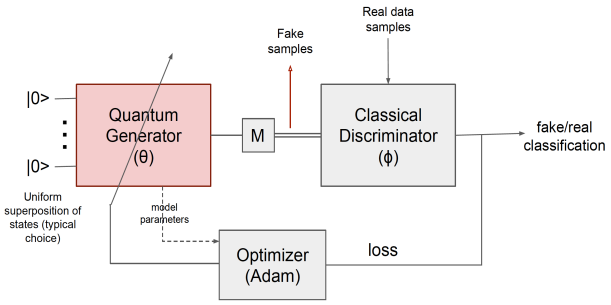


Fig. 1: Schematic diagram of a Quantum Generative Adversarial Network. In the figure, M stands for measurement operator applied to the generator output, which is equivalent to drawing samples from the distribution

### A. Generator

A generator is implemented as a variational quantum circuit, which takes superposition of states with probability amplitudes corresponding to a base distribution. The base distribution is typically uniform, which is generated using all zero qubits as input followed by applying Hadamard gates to each of the qubits. This generates superposition of basis states with constant amplitude, which can be interpreted as a uniform distribution. Let $n$ be the number of qubits in the quantum generator, then the superposition of all the basis states is generated by

$$H^{\otimes n} |0\rangle^{\otimes n} = \frac{1}{2^n} \sum_j |j\rangle, \quad (2)$$

where $H$ is the Hadamard gate. The goal of the generator is to take uniform superposition of states as input to generate a distribution (over a finite sample space) that minimizes Jensen-Shannon divergence with $p_{data}$ by applying a series of transformations (rotations and entanglements). If $G_\theta(.)$ be the quantum generator, then its output state is computed as

$$|g_\theta\rangle = G_\theta(|0\rangle^{\otimes n}) = \sum_{j=0}^{2^n-1} \sqrt{p_\theta^j}\, |j\rangle, \quad (3)$$

where $(|0\rangle)^{\otimes n}$ is the input to the generator, $G_{\boldsymbol{\theta}}$ contains $H^{\otimes n}$ as the first unitary transformation, and $p_\theta^j$ is the probability of the qubit $|g_\theta\rangle$ being in state $|j\rangle$.

*Construction of Generator as a Parameterized Ansatz:* Quantum gates allow only unitary operations. An ansatz for the generator is constructed as multiple layers of Y-rotation gates ($R_Y$ gates) and controlled-Z gates stacked together. The single qubit rotation along the Y-axis of the block sphere is given by

$$R_Y(\theta) = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \quad (4)$$

The reason for choosing $R_Y$ gates to construct ansatz is because only $R_Y$ gate has the ability to change the

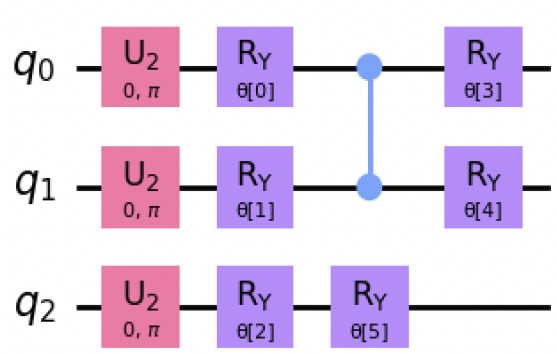


Fig. 2: Quantum generator ansatz with 3 qubits and circuit depth, $k = 1$. For greater circuit depths the rotations $R_Y$ and a cz (controlled-Z gate) entanglement (shown as a connector between $q_0$ and $q_1$) are cascaded together $k$ times, $U_2(0, \pi)$ is the Hadamard gate.

amplitudes (and also phase) of the input state (qubit), on the other hand $R_X$ and $R_Y$ gates (for example) change only the phase of the qubit with respect to basis states. Since, the objective of the generator is to learn the distribution through series of amplitude changes, only $R_Y$ gate seems to be a viable choice.

### B. Discriminator

Discriminator is a classical neural network, which is a binary classifier that learns to distinguish between the data samples and the samples generated by the quantum generator. The optimizers of both discriminator and generator are implemented classically.

### C. Loss Functions

The loss function of the generator is given by

$$L_G(\phi, \theta) = -\frac{1}{m}\sum_{l=1}^{m} \log D_\phi(g^l), \tag{5}$$

where $m$ is the batch size, $x^l$ is the real data sample, and $g^l$ is the fake data sample generated by the quantum circuit. The discriminator optimizes the following loss function:

$$L_D(\phi, \theta) = \frac{1}{m}\sum_{l=1}^{m} \log D_\phi(x^l) + \log(1 - D_\phi(g^l)). \tag{6}$$

## IV. Noise in Quantum Circuits

The near-term quantum computers are prone to noise, resulting in the performance quantum generator to deviate from its expected behavior. Therefore, at least in the near or middle term quantum computers will not be able to harness the power of expressivity similar to the classical deep neural networks. In the presence of the technological bottleneck, careful engineering of physical components and quantum error correction are necessary to undo the effects of noise. In this regard, some of the sources of noise affecting quantum computation are given as follows.

### A. Pauli Error

Pauli error is a cannonical error model used in quantum information theory, which is used to model bit flips and phase flips. It is expressed as a list of tuples of the form $(P_i, p_i)$, where $P_i \in \{I, X, Y, Z\}$ is a Pauli operator and $p_i$ with $\sum_i p_i = 1$ are the corresponding probability. We consider Pauli error in our qGAN simulations in Section V.

### B. Other noise types

Quantum hardware is susceptible to multiple noises such as - coherent noise, incoherent noise, state preparation and measurement Noise, thermal relaxation noise, etc., which makes it challenging to implement deep quantum circuits for machine learning tasks. In this report, our simulations are limited to Pauli error; other types of error can be studied as an extension to this work.

### C. Ways to mitigate noise in qML

To improve the performance of quantum machine learning circuits, a few near-term work-arounds are as follows: Since, deep quantum circuits compound noise, one can explore optimizing shallow quantum ML circuits. Secondly, quantum error-correction is not a completely solved problem yet - there could be room for mitigation of errors through different error control techniques, thereby enabling the implementation of deep quantum neural networks.

## V. Implementation, Results and Discussion

In this section, we will discuss the qGAN simulations performed using Qiskit SDK [6]. We perform each of the simulations for a dataset (real data) of 2000 samples distributed as a bimodal Gaussian distribution centered around $\mu_1 = 1.5$ and $\mu_2 = 5.5$, and the standard deviations of both the modes is $\sigma_1 = \sigma_2 = 1$, number of training iterations (epochs) = 200. The generator is a 3 qubit variational circuit - the samples that fall outside the range ($x^l < 0$ and $x^l > 7$) are discarded. In each layer of the generator, only qubit 0 and qubit 1 are entangled using a controlled-Z gate, while the rest of qubit pairs are not entangled. We use Adam optimizer to learn the model parameters, and the learning rates of both generator and discriminator are set to $10^{-3}$.

In Figure 3, evolution of KL divergence with training epochs is depicted for a qGAN for one iteration - as expected, the KL divergence decreases with training epochs. Figure 4 depicts the learning curves (evolution of loss functions) of the generator and discriminator. In Figure 5, the comparison between the data distribution

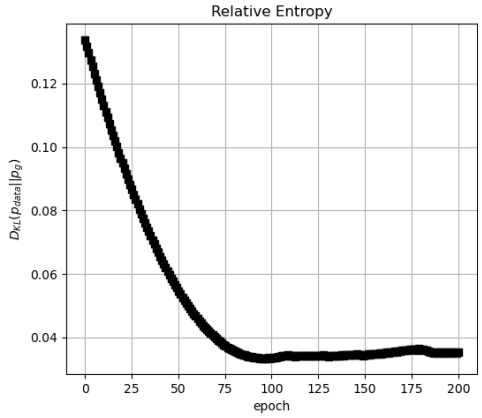


Fig. 3: KL divergence between $p_g$ and $p_{data}$ over training iterations (circuit depth = 2).

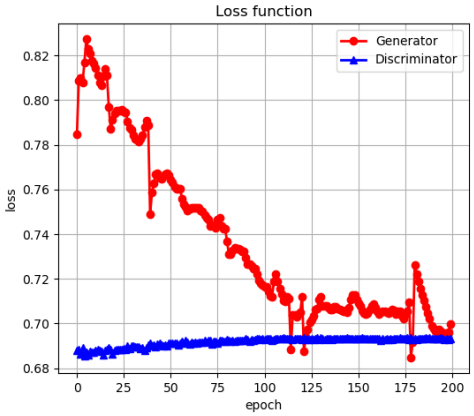


Fig. 4: Loss function of generator and discriminator.

and the distribution of the samples generated from the the quantum generator are shown; the generator approximates the data distribution within a KL divergence of 0.04 (on average) for circuit depth $k = 2$.

### A. *Performance of Ideal Noiseless qGAN*

We investigate the performance of qGANs for multiple circuit depths $k \in [1, 4]$. For each circuit depth, the simulation is averaged over 5 iterations. Figure 6 shows Kullback-Leibler (KL) divergence between the data distribution $p_{data}$ and quantum generator's learnt distribution $p_g$ with increasing circuit depths - we can notice that circuit with higher number of layers approximates the distribution better.

It is important to note that increasing number of layers (depth) does not improve the performance monotonously, because of the well-known problem in quantum variational implementations known as *barren plateau problem*, which is equivalently known as *vanishing gradient* problem in classical machine learning literature.

### B. *Performance of qGAN under Simulated Noise*

We investigated the performance of qGAN with circuit depth for different levels of Pauli error. The noise is applied for every gate and every qubit individually and independently. For a given circuit depth $k$ and a given Pauli error probability $p$, we performed 5 rounds of simulations. In Figure 7, we can observe that for lower error probabilities $p \in \{0, 0.01\}$,d increasing the circuit size improves the performance (the generator approximates the data distribution well). On the other hand, when $p = 0.1$, the KL divergence increases with increase in the number of rotation-entanglement layers. This could be because in the high noise regime, the compounding effect of noise dominates over the expressivity gained by increased circuit depth.

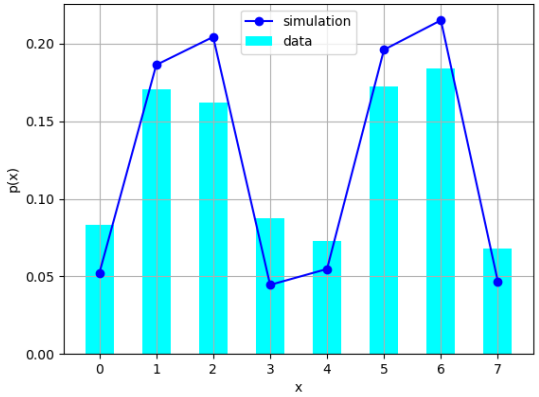


Fig. 5: Plot depicting distribution of the samples from data and quantum generator (simulation) - qGAN approximates the data distribution well.

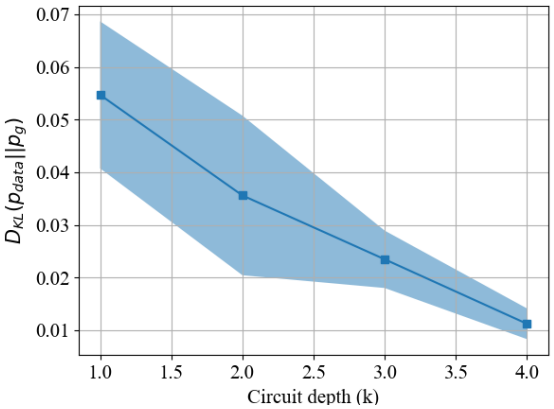


Fig. 6: KL divergence between data and generator distributions for different generator circuit depths. The shaded region covers one standard deviation from the mean KL divergence.

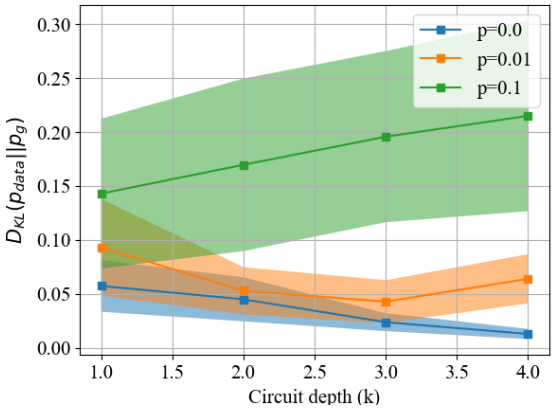


Fig. 7: KL divergence between data and generator distributions for different circuit depths ($k$) and noise levels ($p$).

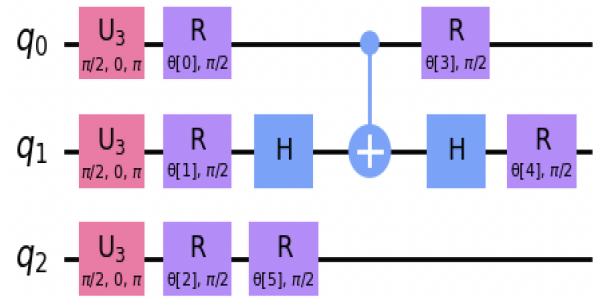


Fig. 8: Noisy quantum generator, each gate is replaced with its noisy version (circuit depth, $k = 1$)

### C. Some Comments on Implementation Challenges

Similar to their classical counterparts qGANs are hard to train. However, the reason for this difficulty in training the quantum generator is quite different - the probability distribution (PMF) that is expressible by the generator is highly dependent and very sensitive to initialization, namely input distribution (superposition of qubits) and also the choice of rotation gates and entanglers. In our simulations, we found an approximate ansatz, which can marginally express the target distribution, by random sampling. This is followed by training qGAN, which navigates through a restricted parameter (based on the chosen ansatz) landscape to find the local optimum.

From the aforementioned exercise, one can ask - what constitutes a good ansatz in terms of expressibility and trainability? This is an open problem even in the noiseless regime. In the classical world, from machine learning theory it is known that neural networks are universal approximators. In this regard, it is important to note that in the quantum generator architecture discussed in this report (and in paper [4]), only rotations are parameterized, but the entanglements are fixed. It is not known whether such an architecture is a universal approximator or not. Therefore, if there are ways to parameterize entanglements, a given ansatz could approximate a richer class of functions than is currently possible, which makes it more closer to a universal approximator.

In our experiments, we also found that qGAN is quite sensitive to the parameters of the generator Ansatz. Randomly initializing the parameters of the rotation blocks and random entanglements made the qubits to be entangled in ways that resulted in the the generator output (PMF) to be trapped in function space with high KL divergence.

## VI. Conclusion

In this report, we presented Quantum Generative Adversarial Networks. We also investigated the performance of qGANs for different circuit depths in the presence of noise. Through simulations we noticed that increasing the circuit complexity is beneficial only if the circuit noise levels are low. Therefore, in the high noise regime, shallow circuits are preferable. We also commented on the some of the limitations and challenges of implementing variational models in quantum setting.

Due to the implementation challenges during the project, several open problems came to the fore. Since, we observed that the performance of the qGAN relied strongly upon the variational ansatz, one can explore ways to search for expressible and trainable ansatz's. In this regard, bounds on expressibility and trainabiliy of a given ansatz are derived in [7]. However, searching for an optimum ansatz is still an open question.

In our simulations, ansatz search the problem was solved by exhaustively searching for different entanglements that generate the desired PMF. This is feasible for small circuits (3 qubits in our case), but for larger circuits an efficient algorithm is need. In general, the generator training can be interpreted as a two-step optimization procedure: expressive ansatz search (which is a combinatorial optimization problem) followed by fine-tuning using gradient descent.

Finally, since the quantum circuits are prone to errors due to different sources of noise, we need to consider robustness of the ansatz in addition to expressivity and trainability. In [8], multiple ansatz's for variational quantum algorithms are discussed in terms of hardware complexity. Even though shallow circuits are preferable in the presence of noise, their relationship with robustness, expressivity and trainability is not clear. Hence, noise resilient ansatz search is also a direction one can consider.

## REFERENCES


[1] C. Zoufal, "Generative quantum machine learning," *arXiv preprint arXiv:2111.12738*, 2021.

[2] A. Cross, "The ibm q experience and qiskit open-source quantum computing software," in *APS March meeting abstracts*, vol. 2018, 2018, pp. L58–003.

[3] Y. Liu, S. Arunachalam, and K. Temme, "A rigorous and robust quantum speed-up in supervised machine learning," *Nature Physics*, vol. 17, no. 9, pp. 1013–1017, 2021.

[4] C. Zoufal, A. Lucchi, and S. Woerner, "Quantum generative adversarial networks for learning and loading random distributions," *npj Quantum Information*, vol. 5, no. 1, pp. 1–9, 2019.

[5] K. Borras, S. Y. Chang, L. Funcke, M. Grossi, T. Hartung, K. Jansen, D. Kruecker, S. Kühn, F. Rehm, C. Tüysüz *et al.*, "Impact of quantum noise on the training of quantum generative adversarial networks," *arXiv preprint arXiv:2203.01007*, 2022.

[6] "Qiskit: An open-source framework for quantum computing," 2021.

[7] Z. Holmes, K. Sharma, M. Cerezo, and P. J. Coles, "Connecting ansatz expressibility to gradient magnitudes and barren plateaus," *PRX Quantum*, vol. 3, no. 1, p. 010313, 2022.

[8] A. Wu, G. Li, Y. Wang, B. Feng, Y. Ding, and Y. Xie, "Towards efficient ansatz architecture for variational quantum algorithms," *arXiv preprint arXiv:2111.13730*, 2021.

[9] G. De Palma, M. Marvian, D. Trevisan, and S. Lloyd, "The quantum wasserstein distance of order 1," *IEEE Transactions on Information Theory*, vol. 67, no. 10, pp. 6627–6643, 2021.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[11] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven, "Barren plateaus in quantum neural network training landscapes," *Nature communications*, vol. 9, no. 1, pp. 1–6, 2018.

[12] P. Selig, N. Murphy, A. Sundareswaran, D. Redmond, and S. Caton, "A case for noisy shallow gate-based circuits in quantum machine learning," in *2021 International Conference on Rebooting Computing (ICRC)*. IEEE, 2021, pp. 24–34.

[13] A. Cross, "The ibm q experience and qiskit open-source quantum computing software," in *APS March meeting abstracts*, vol. 2018, 2018, pp. L58–003.