

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/344757629>

QuGAN: A Generative Adversarial Network Through Quantum States

Preprint · October 2020

CITATIONS

0

READS

1,042

8 authors, including:



[Samuel Stein](#)

Fordham University

9 PUBLICATIONS 81 CITATIONS

[SEE PROFILE](#)



[Betis Baheri](#)

Kent State University

21 PUBLICATIONS 79 CITATIONS

[SEE PROFILE](#)



[Ying Mao](#)

University of Massachusetts Boston

72 PUBLICATIONS 1,014 CITATIONS

[SEE PROFILE](#)

QuGAN: A Quantum State Fidelity based Generative Adversarial Network

Samuel A. Stein,^{1,4} Betis Baheri,² Daniel Chen,³

Ying Mao,⁴ Qiang Guan,² Ang Li,¹ Bo Fang,¹ and Shuai Xu³

¹Pacific Northwest National Laboratory (PNNL), {samuel.stein, ang.li, bo.fang}@pnnl.gov

²Department of Computer Science, Kent State University, {bbaheri, qguan}@kent.edu

³Computer and Data Sciences Department, Case Western Reserve University, {txc461, sxx214}@case.edu

⁴ Computer and Information Science Department, Fordham University, {sstein17, ymao41}@fordham.edu

Abstract—Tremendous progress has been witnessed in artificial intelligence where neural network backed deep learning systems have been used, with applications in almost every domain. As a representative deep learning framework, Generative Adversarial Network (GAN) has been widely used for generating artificial images, text-to-image or image augmentation across areas of science, arts and video games. However, GANs are computationally expensive, sometimes computationally prohibitive. Furthermore, training GANs may suffer from convergence failure and modal collapse. Aiming at the acceleration of use cases for practical quantum computers, we propose QuGAN, a quantum GAN architecture that provides stable convergence, quantum-states based gradients and significantly reduced parameter sets. The QuGAN architecture runs both the discriminator and the generator purely on quantum state fidelity and utilizes the swap test on qubits to calculate the values of quantum-based loss functions. Built on quantum layers, QuGAN achieves similar performance with a 94.98% reduction on the parameter set when compared to classical GANs. With the same number of parameters, additionally, QuGAN outperforms state-of-the-art quantum based GANs in the literature providing a 48.33% improvement in system performance compared to others attaining less than 0.5% in terms of similarity between generated distributions and original data sets. QuGAN code is released at <https://github.com/yingmao/Quantum-Generative-Adversarial-Network>

Index Terms—Quantum State Fidelity, Quantum Generative Adversarial Network, IBM-Q

I. INTRODUCTION

The topic of Generative Adversarial Networks (GANs) has been of significant interest since its discovery [1]. GANs have proven to be an exceptionally successful framework for generative modelling, with GANs being able to generate completely unique synthesized samples based off of a real data set. Among the countless applications of GANs [2]–[4], one domain where significant achievement has been witnessed is within computer vision. A few common examples within Computer Vision GANs include generating similar images to 18th century artist’s paintings [5] to synthesizing images of human faces [6]. The application of GANs is not only restricted to the generation of never-before-seen synthesized samples, with applications extending even to domains such as image enhancement techniques, generating higher resolution images from originally low-resolution samples [7]. GANs

have been a tremendously interesting but challenging topic within Machine Learning [8]. It pushes the boundary of computational creativity, with the demonstrations becoming more jaw-dropping by the year [9].

Although GANs have been both tremendously powerful and interesting, their performance is bounded by multiple factors: training a GAN commonly suffers from a stable convergence problem [10], ever-increasing computational complexity in relation to deep neural networks [11] and computer vision tasks, vanishing gradients, and mode collapse [3]. As contributions from both industry and academia continue to attempt to tackle aforementioned problems, studies on developing Quantum Deep Learning models [12]–[17] have started to draw attention in recent years. This can be attributed most likely due to the allure of quantum supremacy, recently demonstrated by Google [18], as well as hopes to find a possible quantum advantage within the machine learning domain. Currently, most Quantum GAN’s proposed discuss potential uses such as data loading or attempt to attain an advantage over their classical counterparts [19]–[25]. Quantum Deep learning continues to draw attention within Quantum Machine Learning, and has shown signs of significant promise with accuracy’s super-seeding similarly designed classical networks [26]. This line of Quantum-enabled research is motivated by the aforementioned allure of a potential quantum advantage which could partially alleviate the computational complexity issue, as well as searching for a solution to the mode collapse, growing computational costs of classical neural networks and vanishing gradients problem. An example of an established Quantum GAN is demonstrated in [21] which makes the use of $O(\text{poly}(n))$ quantum bits to train a GAN to learn random distributions, generating similar patterns.

In this paper, we present the design and implementation of QuGAN, a Quantum GAN architecture. The key feature of the QuGAN architecture is that the QuGAN model is among the first efforts that run the Discriminator and the Generator on quantum-state based loss functions. With quantum fidelity measurements, these loss functions are enabled by the swap test, a procedure in quantum computation that measures how much two quantum states differ that often comes up in Quan-

tum Machine Learning applications [27]. As a result, compared to the quantum discriminators and quantum generators proposed in the prior work [21], [28], our architecture proves to be more stable and performant. When evaluated with MNIST dataset [29], QuGAN leads to a significant improvement in the Hellinger distance, which represents the similarity between original dataset and generated probability distributions. For a comprehensive evaluation, we conduct both simulations on local servers and experiments on real quantum computers. The key contributions are summarized below.

- Based on quantum fidelity measurements, we propose quantum-state based loss functions with quantum gradients for both the Discriminator and the Generator for the GAN models.
- We evaluate QuGAN on the MNIST [29] dataset with PCA [30] reduced dimensions. This extends the application of the prior GAN models on broader domains.
- Comparing to Tensorflow based classical GANs, QuGAN achieves nearly identical performance with 94.98% fewer parameters. Moreover, QuGAN provides a more stable trend to convergence when compared with the state-of-the-art quantum based GANs [21], [28]: we observe a 48.33% reduction of the variance versus the 0.5% reduction for TFQ-GAN and no clear convergent trend for Qi-GAN over the increasing number of the trained epochs.

II. BACKGROUND

A Generative Adversarial Network (GAN) broadly refers to an architecture comprised of two models which compete against one and other, playing an adversarial game. Each model is tasked with attempting to fool the other in a game of cat and mouse [1]. The common example of the generator being a fraudster producing fake money (generator), wanting to fool a detective (discriminator) who is trained in discerning between fake and real money. The GAN architecture typically consists of a real data set (x), a Discriminator (D) and a Generator (G). The discriminator's objective is to be able to correctly classify between real and fake data synthesized by G. The generator's objective is to synthesize samples that the discriminator will classify as real.

With this architecture being rather generic, GANs are agnostic of the type of data they are fed and have applications in an extensive set of domains. However, the performance within each respective application is strongly dictated by the network architecture [2]. GANs have shown exceptional results in, but not limited to, computer vision [2] and time-based domains such as music composition [4]. In traditional generative networks, G samples from a specific distribution p_z to synthesize a fake sample which is to be fed to D. D is fed with both real data x and data synthesized by G. The discriminator's output is a perception on the data that is fed, and its goal is to maximize $D(x)$ and to minimize $D(G(p_z))$. While the generator's goal is to maximize $D(G(p_z))$. Where probabilistic models aim to label real data as 1 and fake data as

0, the primal min-max optimization formulation is described in Equation 1.

$$\min_G \max_D V(D, G) = \mathbf{E}_{x \sim p_{data(x)}} [\log(D(x))] + \mathbf{E}_{z \sim p_z(z)} [\log(1 - D(G(z)))] \quad (1)$$

The one direction in relation to the GAN-related research is to explore the potential of quantum GANs [19], [20], [31]. The conceptual translation between a classical GAN and a quantum GAN is relatively intuitive, as the generator network's task is to learn the underlying probability distribution of a real data set and quantum computers produce probability distributions instead of discrete values. Prior studies propose the use of quantum GANs ranging from attempting to attain a quantum speed up, or for data loading onto quantum computers [21]–[24]. For example, Zoufal et al. [21] present Qi-GAN to learn random distributions for financial applications. Their work, however, is only extended to one dimensional data learning, and displays significant instability within their models evaluation. Furthermore, they make use of a classical discriminator, and hence making their Qi-GAN only a partial quantum GAN that might not be fully accelerated by quantum systems. Another example of a Quantum GAN [25] makes the use of variational circuits. The authors present a Quantum GAN architecture. However, their learning is rather sporadic and inconsistent, with a non-convergent and unclear trend.

Based on the quantum state fidelity based loss functions for Generator and Discriminator, we propose QuGAN that provides a more stable and scalable architecture of quantum GANs.

A. Quantum Fidelity Measurement

One crucial component regarding the proposed Quantum GAN (QuGAN) architecture is the SWAP test, a quantum state fidelity algorithm. Two states, $|\phi\rangle$ and $|\psi\rangle$, are passed to a system where the measured output is their fidelity. One ancilla qubit in state $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ is passed to the system along side the aforementioned states. The quantum state is described in Equation 2.

$$State = \frac{1}{\sqrt{2}}|0, \phi, \psi\rangle + \frac{1}{\sqrt{2}}|1, \phi, \psi\rangle \quad (2)$$

Following this, the algorithm makes use of a control swap gate, where if the control qubit measures 1 the states are swapped, otherwise nothing happens. The state of $|\psi\rangle$ and $|\phi\rangle$ are control swapped with the ancilla qubit as the control qubit, resulting in the state described in Equation 3.

$$State = \frac{1}{\sqrt{2}}|0, \phi, \psi\rangle + \frac{1}{\sqrt{2}}|1, \psi, \phi\rangle \quad (3)$$

The ancilla qubit is passed through a Hadamard gate referenced in Equation 4, placing the system in the state described in Equation 5.

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \quad (4)$$

$$State = \frac{1}{2}(|0, \phi, \psi\rangle + |1, \phi, \psi\rangle + |0, \psi, \phi\rangle - |1, \psi, \phi\rangle) \quad (5)$$

To attain the probability of measuring the ancilla qubit as 0, we square sum the states where $|0\rangle$ is measured.

$$State = \frac{1}{2}|0\rangle(|\phi\rangle|\psi\rangle + |\psi\rangle|\phi\rangle) + \frac{1}{2}(|\phi\rangle|\psi\rangle - |\psi\rangle|\phi\rangle) \quad (6)$$

To attain Equation 7, we isolate the coefficients to the $|0\rangle$ state and square them to attain a $P(\text{Measurement}=0)$, as seen in Equation 6. As described in Equation 7, the fidelity of states $|\phi\rangle$ and $|\psi\rangle$ is bound to $[0.5, 1]$, and measures the fidelity of two states. In the case of the states being orthogonal, the dot product between the matrices will equal 0 and the fidelity measures 0.5, and if the matrices are identical the dot product measures 1 and the fidelity measures 1.

$$P(\text{Measurement} = 0) = \frac{1}{2} + \frac{1}{2}|\langle\psi, \phi\rangle|^2 \quad (7)$$

III. QUGAN SYSTEM DESIGN

In this section, we introduce the QuGAN system architecture design in details. As shown in Figure 1, the system operates through iterative handovers between a classical and quantum computer. It operates broadly by the iterated upon operation of a classical computer passing a parameterized quantum circuit to a quantum computer, which then passes back a measured quantum state fidelity. QuGAN architecture begins with being fed classical data, which is normalized and transformed into quantum data. This data-preprocessing step is done once per data set. A Generator/Discriminator circuit or Real Data/Discriminator circuit is passed to the quantum circuit, where the circuit passed is chosen based on the stage of the training algorithm. The quantum circuits of induced state fidelity's are transferred back to the classical computer. In the case of the system being optimized, the fidelity is used in the calculation of each gates gradient with respect to the objective function, which is used to update the Generator and Discriminator parameters. If the system is being used to generate samples, instead of passing back a state fidelity the quantum computer is sampled from and a data point generated. QuGAN system is iterated upon a specific number of times, or until a certain goal has been reached.

In existing Quantum generative research, models have looked at taking a traditional Discriminator but changing the Generator to a quantum Generator [24], or even implementing a Quantum Wasserstein GAN [32]. The problem with the Quantum/Classical approach however is that a classical generator fails tackle quantum data efficiently, and requires the quantum data to be translated back to classical data. Furthermore, in the case of the Quantum Wasserstein GAN, this makes use of Earth Movers distance over a density matrix. This approach, although valid, is not as applicable to QuGAN architecture as our quantum state fidelity is measured through a single ancilla qubits expectation value that captures the similarity between two quantum states. One of

Wasserstein Distance's advantage over classical log likelihood loss functions is that its range is not bound between 0 and 1. However, through measuring the expectation of a single qubit, network outputs remain bound even when using Wasserstein distance. Therefore, we deem it appropriate to make use of the log-likelihood approach for optimization. We make use of a Quantum Generator as well as a Quantum Discriminator (details in Equation 12 and 11), and accomplish communication between the models through the SWAP test. The quantum circuits are simulated on a classical computer, with a proof of concept illustration by running the circuits on actual quantum computers.

1) *Quantum Deep Learning Layers*: The modelling of deep learning on a quantum system is commonly represented by collections of quantum gates which are grouped together such that they accomplish specific quantum data manipulation [24]. Each parameterized gate accomplishes a specific type of quantum state manipulation [33]. We adopt this variational quantum circuit approach and expand it to create 3 key types of gate combinations, a single qubit unitary, dual qubit unitary, and an entanglement unitary visualized in Figure 2. A single qubit unitary, as seen in Figure 2(a), performs an R_Y gate on a single qubit, parameterized by θ . This gate accomplishes a manipulation of a single qubits superposition. The dual qubit unitary, drawn in Figure 2(b), performs a rotation on 2 qubits around the Y-axis, parameterized by one θ . This gate accomplishes a manipulation on two qubits superposition. Accomplishing entanglement, QuGAN makes use of an entanglement unitary using a $CR_Y(\theta)$ gate parameterized by one θ , as plotted in Figure 2(c). The $CR_Y(\theta)$ gate rotates a qubit by θ provided the control qubit measures 1, entangling the qubits. These grouped gate operations all accomplish specific quantum data manipulation. The gates unitary matrices are outlined in Equations 8, 9 and 10.

The operations visualized in Figure 2 are grouped together to form a layer of operations on the quantum state parameterized by θ_d . Within the Figure 3, we illustrate each of these layer types in order as well as illustrate how a circuit can be used to generate classical numerical outputs. Which qubits are measured and how this output relates to the optimization function is up to the practitioner. In the case of a Generator, the output sample would be a synthesized data point.

$$R_Y(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix} \quad (8)$$

$$R_{YY}(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & 0 & 0 & i\sin\frac{\theta}{2} \\ 0 & \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} & 0 \\ 0 & -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} & 0 \\ i\sin\frac{\theta}{2} & 0 & 0 & \cos\frac{\theta}{2} \end{bmatrix} \quad (9)$$

$$CR_Y(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ 0 & 0 & \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \quad (10)$$

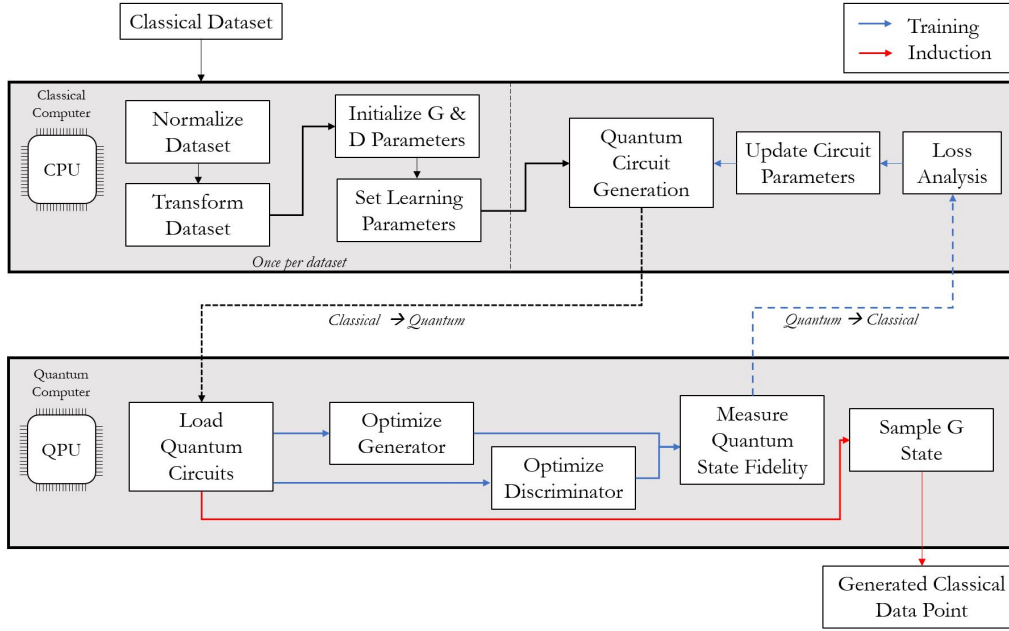
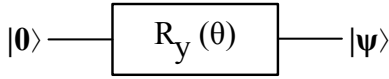
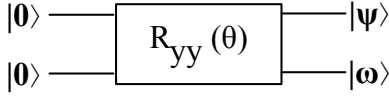


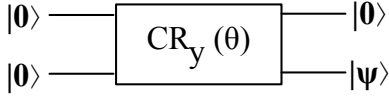
Fig. 1: QuGAN System Architecture



(a) Single-Qubit Unitary



(b) Dual-Qubit Unitary



(c) Entanglement Unitary

Fig. 2: Three Types of Parameterized Quantum Layers

For QuGAN, the architecture is comprised of a Discriminator (a Quantum based Neural Network), the Generator (another Quantum based Neural Network), real data and an ancilla qubit to measure Quantum fidelity. We design both Discriminator and Generator equally to allow for equal ability in learning of the ideal quantum state. The general architecture of Our QuGAN on the actual quantum computer, described in the “Quantum Computer” block of Figure 1, is outlined in Figure 4.

In Figure 4, we illustrate the Quantum Computer’s role through the two possible styles of circuits being prepared. In the case of $|G\rangle$, the circuit prepares the two respective states

using the layered approach. If there is data being prepared, only the Discriminator makes use of a layered approach, whilst the data is encoded using $R_Y(\theta)$ gates, which is fed onto a circuit along side the Discriminator.

The generated quantum circuit operates by whether the system is measuring the Discriminator ($D/|\delta\rangle$) loss relative to real data or fake data, which is illustrated by the “Generator” ($G/|\gamma\rangle$) and “Real Data” ($X/|\xi\rangle$) blocks, based on whether we the system is measuring D/G or D/X loss. The discriminators loss in terms of quantum states is described in Equation 11 and generators in Equation 12. The terms within the log relate to the quantum fidelity measurement being normalized to a $[0,1]$ scale, hence why there is no $\frac{1}{2}$. To generate a sample, the quantum state representing the Generator is induced and sampled from n times, and is visualized in Figure 1. The average measurement per each qubit represents a specific dimensions output.

$$D_{loss} = \mathbf{E}[\log(D(|\langle\xi, \delta\rangle|^2) + \mathbf{E}[\log(1 - D(|\langle\gamma, \delta\rangle|^2)] \quad (11)$$

$$G_{loss} = \mathbf{E}[\log(D(|\langle\gamma, \delta\rangle|^2)] \quad (12)$$

The gradient of the loss function is used to update the generator and discriminators states respectively to improve their performance in the adversarial game.

2) *Quantum based QuGAN Optimization:* Optimization of a Quantum GAN can be characterized by the loss function described in Equation 11 and 12. The design of our QuGAN is comprised of 4 key quantum circuit components, namely the Generator Circuit, Discriminator Circuit, Data Loading Circuit and the Ancilla qubit. The Generator circuit represents the quantum deep learning (QDL) model tasked with generating

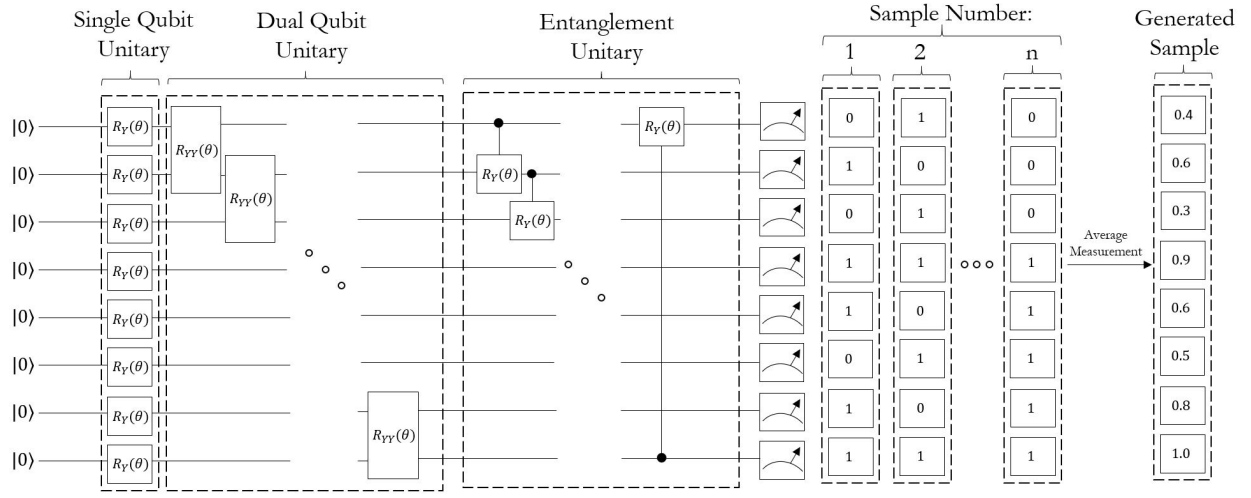


Fig. 3: Layered Quantum Network Design

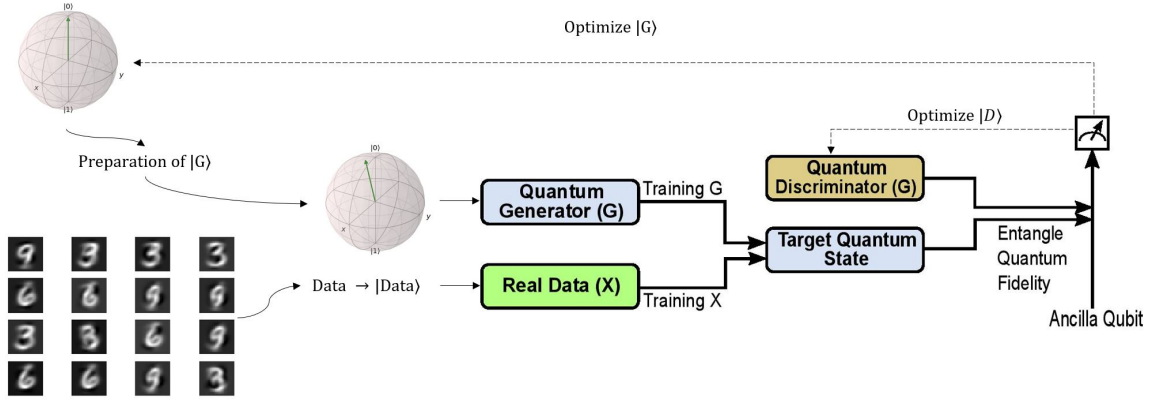


Fig. 4: Inter-model Communication Architecture

samples; the discriminator represents the QDL model that is tasked with discerning between fake and real samples; the data loading circuit is responsible for loading quantum data onto a quantum state; the ancilla qubit is a single qubit with which we measure quantum state fidelity and through which we accomplish our inter-circuit communication. This single qubit being used to measure fidelity has its benefits within the quantum setting, as near term devices can be rather noisy [34], [35] and hence using lower qubit counts requiring measurement can reduce the number of samples needed to attain a confident reading from the quantum computer. Therefore, we find this to be one of the advantages to our model such that we are only measuring one qubit to encapsulate our entire systems performance. A range of $[0.5, 1]$ would not necessarily work as well within a log optimization problem, therefore we normalize our SWAP test value to be between $[0, 1]$, which is seen in the final equation forms in 11 and 12. This possible range of measurements is further motivation to use the original GAN formulation of a log loss function instead of other approaches such as Wasserstein distance. In comparison and

further justifying our use of a probabilistic loss, Wasserstein distance makes use of a linear output neuron in classical GANs, attaining values in the range of $[-\infty, \infty]$, one of the large advantages of Wasserstein GANs, where quantum state observables are bound by $[0, 1]$.

These quantum state fidelity's are used to calculate both the Discriminator and Generator loss. We can update the parameters of the networks similarly to a classical GAN, as visualized in Figure 1, through the use of gradient descent and quantum gate differentiation. In differentiating a quantum gate, we attain the gradient of a parameter in the gates layers weight vector. The system proceeds by measuring the loss of the Discriminator with respect to both the Generator and the Real Data, then updates the quantum deep learning model to improve the performance at a rate of α (learning rate). Following this the generator analyzes how well it performs against the Discriminator and updates its quantum deep learning model to improve performance respectively. To perform gradient descent of each gate, we employ a parameterized differential equation from [36] outlined in Equation 13.

$$\frac{\delta f}{\delta \theta} = \frac{1}{2}(f(\theta + \frac{\pi}{2}) - f(\theta - \frac{\pi}{2})) \quad (13)$$

Where in Equation 13, $\delta \theta$ indicates a specific θ gradient, and f is the cost function for the network, which in our case is described in Equation 11 and 12. This approach allows for analytical differentiation of the Quantum Neural Network with the downfall being that this approach is computationally expensive having to induce the network twice per gate to attain the gradient.

3) *Data Qubitization and QuGAN Algorithm*: Data encoding on quantum computers continues to be an area of research, however is not a primary focus of this paper. Hence we propose our method, but acknowledge alternate methods exist and could be tested on our architecture. Taking a classical numerical data set of values x_1, x_2, \dots, x_n with dimensions d , stored through the use of classical bits, using data types such as floats or integers. We note the limitation of qubit counts renders this method of data encoding infeasible, and additionally this method does not make use of the potential of the exponential Hilbert space representing quantum states. In designing our data encoding method, we operate on the premise of reversibility, which allows for the transformation of classical data into quantum data, and then returned back into its classical version. The E value of a qubit can be used to encode classical data, however the expected value range is between 0 and 1. Therefore, we normalize each dimension d_i range to be between 0 and 1. Following this, we encode a data point of value x to a rotation $\theta = 2 \sin^{-1}(\sqrt{x})$. This results in the E value of a qubit equating to the classical data point.

The Algorithm 1 illustrates the process we go through to implement our design. In Line 1 we initialize all of our initial parameters for the Generator and the Discriminator. The network weights line indicates the initialization of the parameter vector for both models. Each entry is initialized as a randomly sampled value from a uniform distribution of bounds $[0, \pi]$. We set our epoch count, load our data set, and indicate the ratio of which we will train the discriminator against training the generator. Proceeding this, from Line 2, we begin training our model. We train the discriminator on the real data set once in Lines 4 through 6, followed by training the discriminator on fake samples from the generator in Lines 7 through 10, and finally train the generator on the discriminator in a total of R times (Line 11 - 14).

IV. QUGAN EVALUATION

In this section, we discuss the implementation of QuGAN and experiments that we conducted to evaluate it.

QuGAN is implemented based on Python 3.8 with IBM Qiskit, an open-source framework for quantum computing. The generated quantum circuits are trained on GPU-enabled servers on the Google Cloud Platform. In addition, we conduct experiments on IBM-Q Experience to validate QuGAN with real quantum computers.

To produce classical results from the QuGAN, we sample the Generator circuit n times and take the mean of those

Algorithm 1 Quantum States based Learning

```

1: Parameter Initialization:
   Learning Rate :  $\alpha = 0.01$ 
   Network Weights:  $\theta_d = \text{Random}(0, 1) \times \pi$ 
   epochs:  $\epsilon = 25$ 
   Dataset:  $X$ 
   D-to-G Train Count:  $I$ 
   G-to-D Train Count:  $R$ 
   Qubit Channels :  $Q = 1 + (n_{X_{dim}} \times 2)$ 
2: for  $\zeta \in \epsilon$  do
3:   Train The Discriminator On Real Data
4:   for  $x_k \in X$  do
5:      $Grad = \frac{dCost_{x_k}}{d\theta_D}$ 
6:     Update  $\theta_D \leftarrow \theta_D - \alpha Grad$ 
7:   end for
8:   Train the discriminator on the generator
9:   for  $i \in I$  do
10:     $Grad = \frac{dCost_D}{d\theta_D}$ 
11:    Update  $\theta_D \leftarrow \theta_D - \alpha Grad$ 
12:   end for
13:   Train the Generator on Discriminator
14:   for  $j \in R$  do
15:     $Grad = \frac{dCost_G}{d\theta_G}$ 
16:    Update  $\theta_G \leftarrow \theta_G - \alpha Grad$ 
17:   end for
18: end for

```

measurements per qubit. In the evaluation, we set $n = 30$ to attain a data point. For example, we sample on a Qubit $|0\rangle$ 14 times, and $|1\rangle$ 16 times, therefore giving us the measured value of $\frac{(14(0)+16(1))}{30} = 0.533$ on said qubit.

For a comprehensive evaluation, we compare QuGAN with the following state-of-the-art solutions in the literature.

- **C-GAN**: a classical GAN, which is implemented and trained with different number of parameters as well as epochs under TensorFlow framework with methods similar to that outlined in [37].
- **Qi-GAN**: a quantum GAN [21] that is designed to learn random distributions. It is implemented with Qiskit.
- **TFQ-GAN**: a quantum GAN [28] that utilizes the TensorFlow quantum architecture proposed in [25].

To evaluate the models, we use *Hellinger Distance* [38] as our key metric to compare the generated distribution and original data sets. Hellinger distance is a popular tool to quantify the similarity between two probability distributions and encompasses the difference in two distributions.

Although modern classical GAN's are evaluated on qualitative measures, such as Inception Score (IS) [39], this evaluation is not fit for our dataset due to the minimum requirements on dimensions (e.g. limited number of qubits). Furthermore, the use of Hellinger Distance is a well-suited measurement for a quantum space generator, as the output of the Generator is naturally a distribution, which will have a certain distance to the true data. This difference in distributions encapsulated in

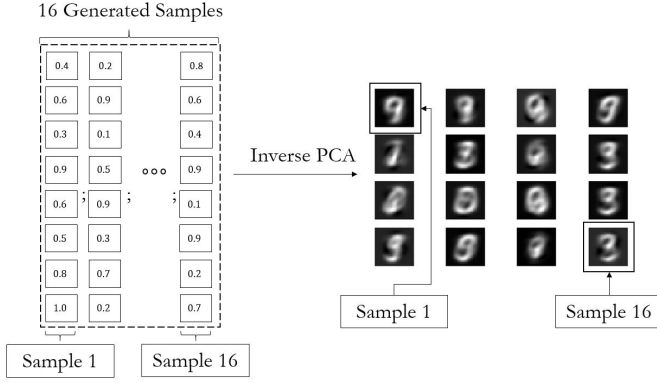


Fig. 5: Image Generation Process: To represent our output, a sampling average of each qubit over the Generator is passed through an Inverse-PCA algorithm. This allows us to intuitively visualize and critique our results.

one number is ideal for evaluating our GAN’s performance. If our output space probability distribution stops moving (i.e. gate gradients have vanished), we will observe no change in the Hellinger Distance. Hence, Hellinger Distance we believe is a good metric to observe for quantum state stability and network convergence.

A. QuGAN: MNIST Learning

To demonstrate the learning ability of our architecture, we evaluate our Quantum GAN over the MNIST data set. The MNIST data set contains 60,000 images of hand-drawn digits, labeled with their respective class (i.e. an image of a hand-drawn 4 is labelled as a 4).

The resolution of the images is 28×28 , which is a high dimensionality for current quantum simulators as well as near-term quantum computers. Therefore, we make use of Principal Component Analysis (PCA) algorithm to downscale [40] the dimensionality from 784 to 4 with an explained variance of 28%. To generate images we sample the QuGAN circuit 30 times and take the mean measurement of each qubit, to which we transform the data points generated from the Quantum GAN back through the PCA transformer into 784 dimensions. From here, we draw the images and attain our QuGAN generated images. The use of PCA as a data compression tool is motivated by previous works highlighting the possibility of a quantum PCA algorithm [41]. To track the progress of our algorithm, we measure the Hellinger Distance between the Generator and the true data set. To generate a sample image, we extend the sample generation described Section III into Figure 5, whereby 16 samples are generated by our quantum circuit to which we apply Inverse PCA, which visualizes our Generator output. The architecture of our Generator/Discriminator is visualized in Figure 6, representing 3 of the 4 key components - namely the Discriminator, Ancilla Qubit, and the Generator. For a Real Sample, instead of Q_5 through Q_8 having trainable rotations, a data point is loaded instead.

Architecture	Parameters	Hellinger Distance
QuGAN	10	0.1951
C-GAN	20	0.4448
C-GAN	45	0.3234
C-GAN	96	0.2515
C-GAN	199	0.1945
C-GAN	1299	0.1389

TABLE I: Different GAN Architectures with 50 Epochs

1) *Comparing to C-GAN*: In comparing our architecture to a classical GAN, we run a TensorFlow-based classical GAN with k parameters over 100 epochs, and measure the average Hellinger Distance using 100,000 samples generated from the generator, trained on classes 3/6/9. As seen, the classical GAN with a parameter count of 20 in the Generator and 20 in the Discriminator performs significantly worse than our QuGAN, with our QuGAN attaining a 56.14% improvement with respect to Hellinger distance. We increase the parameter count up until 199, where the Hellinger distance begins to approach similar values to our QuGAN, with distances less than 0.3% difference. These results are outlined in Table I. We illustrate that to attain similar performance in a classical GAN, we need to use 199 parameters versus the 10 in our QuGAN - a 94.98% reduction in parameters for a similar performance.

For a comprehensive evaluation, we further compare a Classical GAN to generating 2D MNIST distributions, as visualized in Figure 7. In this experiment, we illustrate our architecture on the values combinations $\{3, 8\}$, $\{9\}$ down scaled to 2-dimensions for the purpose of learning visualization, where we illustrate the output distribution of our Generator (Blue distribution) against the actual data points (Red values). This is visualized in Figure 10. Initially, a Hellinger Distance of 0.7561 is obtained for $\{9\}$ and 0.8157 for $\{3, 8\}$. After 100 epochs, the Generator has converged to a Hellinger distance of 0.08171 for 3, 8 and 0.0875 for 9. We note that the ideal outputs too are relatively noisy, as with many encoding to decoding (i.e. PCA) processes suffer from image blurriness post decoding. Therefore, our generators success should be compared to the “Real Images” in Figure 9.

2) *Comparing to Qi-GAN and TFQ-GAN*: Next, we compare our QuGAN architecture with state-of-the-art quantum-based GANs, Qi-GAN and TFQ-GAN, on the PCA-MNIST dataset $\{3, 9\}$ trained over 25 epochs. As visualized in Figure 8 we illustrate QuGAN’s stable learning ability by comparing to alternative quantum learning architectures. In this figure, we see no signs of consistent learning over Qi-GAN, nor with the TFQ-GAN in terms of Hellinger Distance. Comparing to our architecture, we see QuGAN has a stable consistent convergence. As for Qi-GAN, however, the training is extremely sporadic. This sporadic nature is found in our evaluation of the distance. With our architecture, over 25 epochs, a 48.33% reduction in Hellinger Distance was attained, however for TFQ-GAN a less than 0.5% change occurred, and Qi-GAN had no consistent trend and hence any changes attributed to noise.

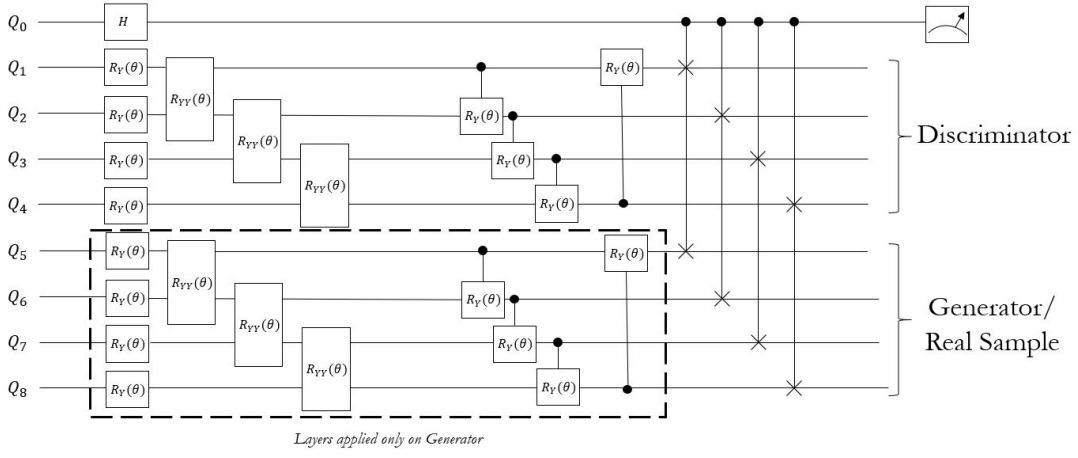


Fig. 6: QuGAN Circuit with PCA MNIST data set: Visualized on the circuit, the Discriminator and Generator have 10 parameters (θ on the figure) each. Qubits 1 through 4 are the Discriminator, and qubits 5 through 8 are reserved for the Generator parameters or data loading. Qubit 0 is the ancilla qubit for the SWAP test. Qubit 0 is measured onto a classical bit which reads either 1 or 0.

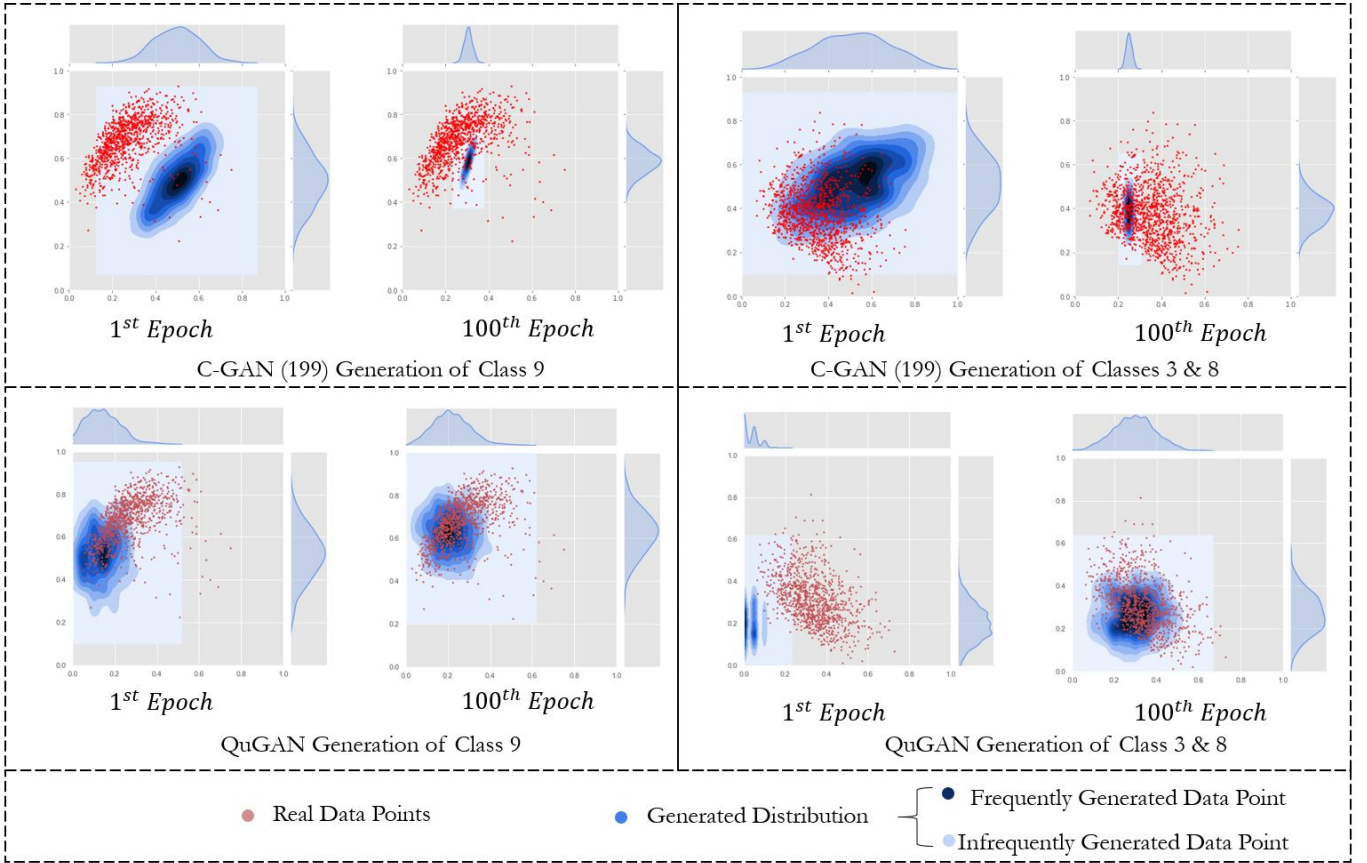


Fig. 7: Generator Learned Distribution for MNIST: In comparing to a C-GAN (199), our generator is able to attain a significantly more diverse output in comparing to the 2D representation, with both classical experiments consistently leading to mode collapse [3], [42]. We show how our architecture represents a more general and diverse output over its classical counterpart.

3) *Learning Stability*: In this experiment, we train the PCA dataset. The evolution of generated images is visualized in Figure 9. As seen in Figure 9, initial generated images

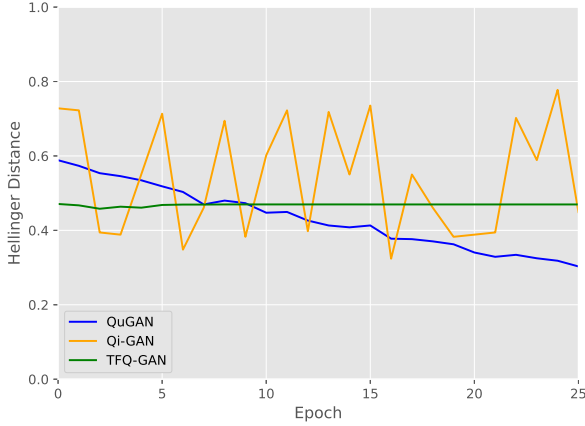


Fig. 8: Comparison with TFQ-GAN and Qi-GAN

are unrecognizable with the same noise being generated each time. This consistent noise can be attributed to PCA's focus on preserving dimensions of highest variance. Hence, corners remain black and without interference however the center where the strokes commonly comprise numbers are highly activated. We notice some learning by the 25th epoch, with improvements by the 50th epoch. This is captured within the Hellinger Distance tracked in Figure 10 that illustrates our learning process through the three stages of a Generator learning. Initially, the discriminator provides no meaningful gradients illustrated through an increasing Hellinger distance (red zone), after which we observe a stable training visualized through a decreasing Hellinger distance (green zone), proceeding this we observe convergence whereby the Discriminator provides no meaningful gradients (black zone). As we see in Figure 10, convergence was reached at the 38th epoch into training with oscillations around the converged value being attributed to the inherent noisiness of quantum computers and our sampling technique.

4) *QuGAN: IBM-Q Evaluation:* To further evaluate our architecture, we run our QuGAN on IBM-Q “Melbourne”, a 15-Qubit, 8-Quantum-Volume Quantum Processor through IBM Quantum Experience. Making use of 20 samples per GAN sample, we measure the Hellinger Distance on both the Quantum Computer and the simulator with the same configurations. The measured distances are depicted Figure 11, where we illustrate the learning process of our architecture on IBM-Q Melbourne and the simulator. Two key aspects are taken away from this figure, being the noise of quantum computers as illustrated by the inability to converge to lower Hellinger Distances, and the initial better performance using the same parameters due to noise hence covering larger areas of the latent space.

As can be seen in Figure 11, initially the simulator attains worse results than the real quantum computer with values 0.678 and 0.548 respectively - a 19.20% difference. However, the simulated Quantum Computer converges closer to the true distribution at a Hellinger Distance of 0.280. This can be

attributed to noise, where the simulator is confidently incorrect initially, but becomes more confidently correct as time goes on and can make accurate samples each circuit sampling. In contrast, the IBM-Q results start out with a lower Hellinger Distance, which can be attributed to it being noisy, therefore covering more of the spannable space. IBM-Q converges at a higher Hellinger Distance of 0.337, with the 20.357% higher distance attributed to the noise of real Quantum Computers.

V. CONCLUSIONS AND DISCUSSION

In this paper, we proposed QuGAN, a Quantum Generative Adversarial Network. QuGAN utilizes quantum states to encode classical data and by using Quantum State Fidelity, we make use of a quantum-based loss functions for the Generator and Discriminator.

The proposed model is evaluated with MNIST dataset. We implement QuGAN with IBM Qiskit and conduct extensive simulations as well as experiments to evaluate QuGAN. It is compared with Tensorflow based classical GANs with different settings, a Qiskit based quantum GAN (Qi-GAN) and Tensorflow-Quantum based GANs in the recent literature (TFQ-GAN). The results demonstrate that QuGAN is able to achieve similar performance at meanwhile, reduces 94.98% of the parameter count compared with classical GANs. Furthermore, it outperforms comparable quantum GANs, Qi-GAN and TFQ-GAN.

Due to the current limits on quantum computers, it is infeasible to evaluate existing quantum based GANs with ImageNet [43] or other large datasets. However, with the rapid evolution of quantum computing, the dimensionality and applicability of QuGANs should be explored as the future work. In addition, further investigation of the effect of deeper quantum neural networks should be considered. QuGAN code is released at <https://github.com/yingmao/Quantum-Generative-Adversarial-Network>

ACKNOWLEDGEMENTS

This material is based upon work partially supported by the U.S. Department of Energy, Office of Science, National Quantum Information Science Research Centers, Co-design Center for Quantum Advantage (C2QA) under contract number DESC0012704. We thank the IBM Quantum Hub to provide the quantum-ready platform for our experiments. Dr. Mao thanks the Faculty Fellowship from Fordham University.

REFERENCES

- [1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.
- [2] Z. Wang, Q. She, and T. E. Ward, “Generative adversarial networks in computer vision: A survey and taxonomy,” *arXiv preprint arXiv:1906.01529*, 2019.
- [3] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein gan,” *arXiv preprint arXiv:1701.07875*, 2017.
- [4] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, “Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

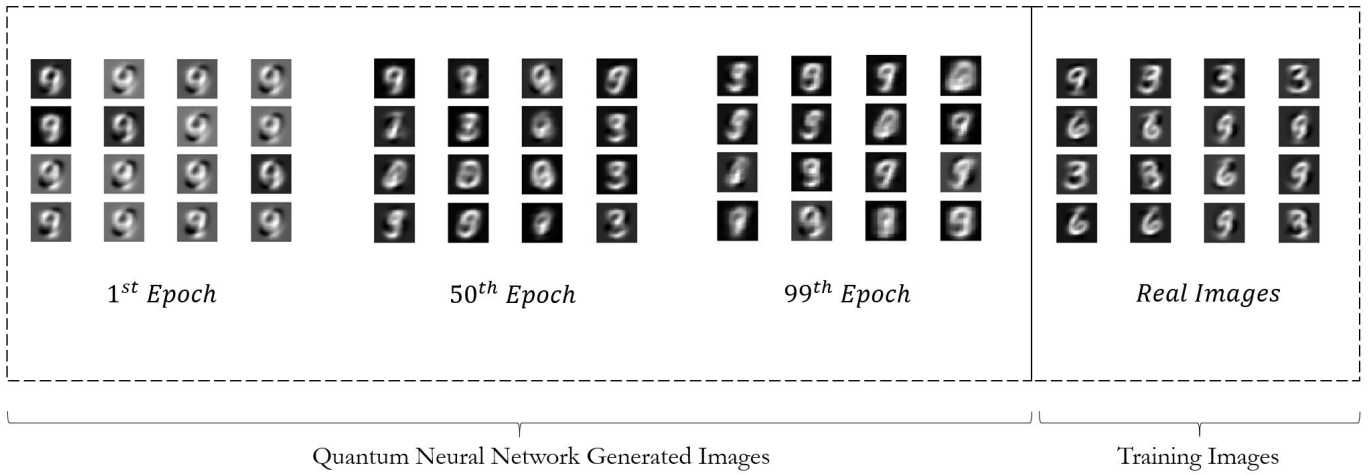


Fig. 9: QuGAN Generated Images (3/6/9): We illustrate our image generation process over 99 epochs. Critiquing the images generated on the 100th epoch, targeting our respective labels we trained the generator on we see strong signs of a 9 being generated in row 2 column 4, evidence of a 6 in row 1 column 4, and a 3 in row 3 column 2

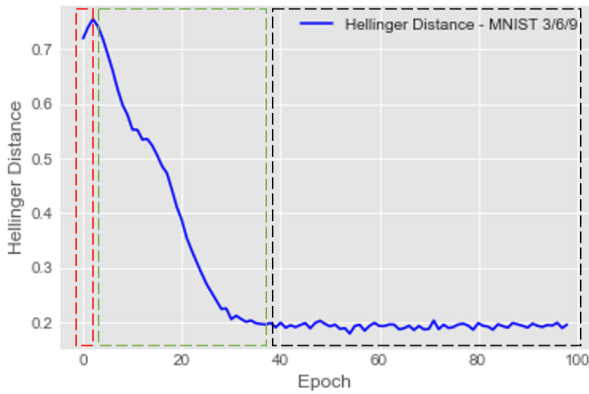


Fig. 10: Hellinger Distance of MNIST 3/6/9

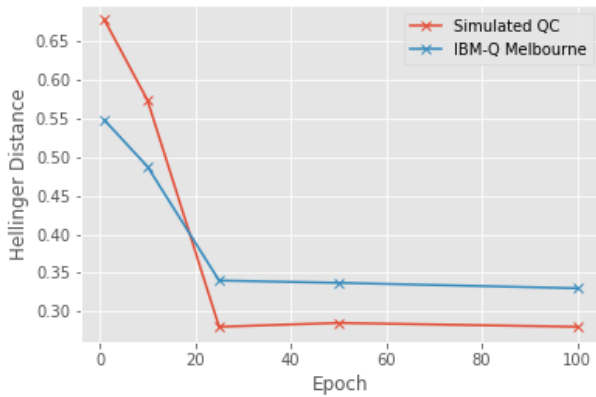


Fig. 11: IBM-Quantum v.s. Simulator

- [5] P. Kravtsov and P. Kuznetsov, "Creative adversarial networks," <https://github.com/mlberkeley/Creative-Adversarial-Networks>, 2017.
- [6] J. Zhao, L. Xiong, P. Karlekar Jayashree, J. Li, F. Zhao, Z. Wang, P. Sugiri Pranata, P. Shengmei Shen, S. Yan, and J. Feng, "Dual-agent gans for photorealistic and identity preserving profile face synthesis," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 66–76.
- [7] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 105–114.
- [8] M. P. Kumar and P. Jayagopal, "Generative adversarial networks: a survey on applications and challenges," *International Journal of Multimedia Information Retrieval*, pp. 1–24, 2020.
- [9] Y. Liao, K. Schwarz, L. Mescheder, and A. Geiger, "Towards unsupervised learning of generative models for 3d controllable image synthesis," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [10] K. Roth, A. Lucchi, S. Nowozin, and T. Hofmann, "Stabilizing training of generative adversarial networks through regularization," in *Advances in neural information processing systems*, 2017, pp. 2018–2028.
- [11] P. Orponen *et al.*, "Computational complexity of neural networks: a survey," *Nordic Journal of Computing*, 1994.
- [12] S. Garg and G. Ramakrishnan, "Advances in quantum deep learning: An overview," *arXiv preprint arXiv:2005.04316*, 2020.
- [13] N. Wiebe, A. Kapoor, and K. M. Svore, "Quantum deep learning," *arXiv preprint arXiv:1412.3489*, 2014.
- [14] S. Lu, L.-M. Duan, and D.-L. Deng, "Quantum adversarial machine learning," *Phys. Rev. Research*, vol. 2, p. 033212, Aug 2020. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevResearch.2.033212>
- [15] L. Hu, S.-H. Wu, W. Cai, Y. Ma, X. Mu, Y. Xu, H. Wang, Y. Song, D.-L. Deng, C.-L. Zou, and L. Sun, "Quantum generative adversarial learning in a superconducting quantum circuit," *Science Advances*, vol. 5, no. 1, 2019. [Online]. Available: <https://advances.sciencemag.org/content/5/1/eaav2761>
- [16] S. A. Stein, B. Baheri, D. Chen, Y. Mao, Q. Guan, A. Li, S. Xu, and C. Ding, "Quclassi: A hybrid deep neural network architecture based on quantum state fidelity," *Proceedings of Machine Learning and Systems*, vol. 4, pp. 251–264, 2022.
- [17] S. A. Stein, R. L'Abbate, W. Mu, Y. Liu, B. Baheri, Y. Mao, G. Qiang, A. Li, and B. Fang, "A hybrid system for learning classical data in quantum states," in *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*. IEEE, 2021, pp. 1–7.

- [18] F. Arute and et al, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, p. 505–510, 2019.
- [19] S. Lloyd and C. Weedbrook, "Quantum generative adversarial learning," *Physical review letters*, vol. 121, no. 4, p. 040502, 2018.
- [20] L. Hu, S.-H. Wu, W. Cai, Y. Ma, X. Mu, Y. Xu, H. Wang, Y. Song, D.-L. Deng, C.-L. Zou *et al.*, "Quantum generative adversarial learning in a superconducting quantum circuit," *Science advances*, vol. 5, no. 1, p. eaav2761, 2019.
- [21] 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.
- [22] G. Verdon, M. Broughton, J. R. McClean, K. J. Sung, R. Babbush, Z. Jiang, H. Neven, and M. Mohseni, "Learning to learn with quantum neural networks via classical neural networks," *arXiv preprint arXiv:1907.05415*, 2019.
- [23] K. Beer, D. Bondarenko, T. Farrelly, T. J. Osborne, R. Salzmann, D. Scheiermann, and R. Wolf, "Training deep quantum neural networks," *Nature communications*, vol. 11, no. 1, pp. 1–6, 2020.
- [24] S. Y.-C. Chen, C.-H. H. Yang, J. Qi, P.-Y. Chen, X. Ma, and H.-S. Goan, "Variational quantum circuits for deep reinforcement learning," *arXiv preprint arXiv:1907.00397*, 2019.
- [25] P.-L. Dallaire-Demers and N. Killoran, "Quantum generative adversarial networks," *Physical Review A*, vol. 98, no. 1, p. 012324, 2018.
- [26] W. Jiang, J. Xiong, and Y. Shi, "Can quantum computers learn like classical computers? a co-design framework for machine learning and quantum circuits," *arXiv preprint arXiv:2006.14815*, 2020.
- [27] E. Aïmeur, G. Brassard, and S. Gambs, "Machine learning in a quantum world," in *Conference of the Canadian Society for Computational Studies of Intelligence*. Springer, 2006, pp. 431–442.
- [28] M. Broughton, G. Verdon, T. McCourt, A. J. Martinez, J. H. Yoo, S. V. Isakov, P. Massey, M. Y. Niu, R. Halavati, E. Peters, M. Leib, A. Skolik, M. Streif, D. V. Dollen, J. R. McClean, S. Boixo, D. Bacon, A. K. Ho, H. Neven, and M. Mohseni, "Tensorflow quantum: A software framework for quantum machine learning," 2020.
- [29] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database. 2010," URL <http://yann.lecun.com/exdb/mnist>, vol. 7, p. 23, 2010.
- [30] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.
- [31] H.-L. Huang, Y. Du, M. Gong, Y. Zhao, Y. Wu, C. Wang, S. Li, F. Liang, J. Lin, Y. Xu *et al.*, "Experimental quantum generative adversarial networks for image generation," *arXiv preprint arXiv:2010.06201*, 2020.
- [32] S. Chakrabarti, H. Yiming, T. Li, S. Feizi, and X. Wu, "Quantum wasserstein generative adversarial networks," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019.
- [33] D. P. DiVincenzo, "Quantum gates and circuits," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1969, pp. 261–276, 1998.
- [34] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.
- [35] C. Xue, Z.-Y. Chen, Y.-C. Wu, and G.-P. Guo, "Effects of quantum noise on quantum approximate optimization algorithm," *Chinese Physics Letters*, vol. 38, no. 3, p. 030302, 2021.
- [36] G. E. Crooks, "Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition," *arXiv preprint arXiv:1905.13311*, 2019.
- [37] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *arXiv preprint arXiv:1701.04862*, 2017.
- [38] M. S. Nikulin, "Hellinger distance," *Encyclopedia of mathematics*, vol. 78, 2001.
- [39] S. Barratt and R. Sharma, "A note on the inception score," *arXiv preprint arXiv:1801.01973*, 2018.
- [40] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.
- [41] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum principal component analysis," *Nature Physics*, vol. 10, no. 9, pp. 631–633, 2014.
- [42] R. Durall, A. Chatzimichailidis, P. Labus, and J. Keuper, "Combating mode collapse in gan training: An empirical analysis using hessian eigenvalues," *arXiv preprint arXiv:2012.09673*, 2020.
- [43] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.