 doi:10.20944/preprints202103.0583.v1

Quantum Computing Aided Machine Learning Through Quantum State Fidelity

Samuel A. Stein

Masters of Science
in Data Science Thesis
Fordham University

Chapters

 doi:10.20944/preprints202103.0583.v1

Abstract

Tremendous progress has been witnessed in artificial intelligence within the domain of neural network backed deep learning systems and its applications. As we approach the post Moore's Law era, the limit of semiconductor fabrication technology along with a rapid increase in data generation rates have lead to an impending growing challenge of tackling newer and more modern machine learning problems. In parallel, quantum computing has exhibited rapid development in recent years. Due to the potential of a quantum speedup, quantum based learning applications have become an area of significant interest, in hopes that we can leverage quantum systems to solve classical problems. In this work, we propose a quantum deep learning architecture; we demonstrate our quantum neural network architecture on tasks ranging from binary and multi-class classification to generative modelling. Powered by a modified quantum differentiation function along with a hybrid quantum-classic design, our architecture encodes the data with a reduced number of qubits and generates a quantum circuit, loading it onto a quantum platform where the model learns the optimal states iteratively. We conduct intensive experiments on both the local computing environment and IBM-Q quantum platform. The evaluation results demonstrate that our architecture is able to outperform *Tensorflow-Quantum* by up to 12.51% and 11.71% for a comparable classic deep neural network on the task of classification trained with the same network settings. Furthermore, our GAN architecture runs the discriminator and the generator purely on quantum hardware and utilizes the swap test on qubits to calculate the values of loss functions. In comparing our quantum GAN, we note our architecture is able to achieve similar performance with 98.5% reduction on the parameter set when compared to classical GANs. With the same number of parameters, additionally, QuGAN outperforms other quantum based GANs in the literature for up to 125.0% in terms of similarity between generated distributions and original data sets.

Keywords— Quantum Machine Learning; Quantum Deep Learning; Quantum Generative Adversarial Networks; Quantum State Fidelity

1 Introduction

In recent years Machine Learning has seen astronomical growth, with large strides being made in algorithms and computational power, ubiquitous Machine Learning use cases and an explosion in rate of global data generation. The learning system has been deployed in various applications, such as edge computing [28, 16, 1, 2, 41], cloud computing [40, 51, 39, 38]. Our daily interactions with machine learning have become so wide spread that we barely pay attention to it anymore, with auto-edits on our photos to music recommendations generated a rate that exceeds what we could ever consume. With this said, the motivation for the search of improvements in the field of Machine Learning is very easily motivated. Machine learning continues to provide prodigious utility to an ever growing multitude of industries, from life saving applications of detecting lung diseases [7] to autonomous driving [42].
One branch of Machine Learning that has seen phenomenal success is Deep Learning [26, 34, 56, 55, 24], a neurological inspired machine learning architecture. Deep learning has drawn tremendous interest in recent decades from industry and academia, with the architecture being extremely successful in a multitude of fields from scientific data processing [43, 53], to image and speech recognition [33, 29]. With every machine learning architecture however, there comes a plethora of cons to its pros. With deep learning, these lie mostly with the computational complexity of the models as well as the inability to interpret models efficiently. As we stride towards more complex deep learning and improved deep learning models, we quickly find that the computational complexity of proposed models becomes infeasible. This is not necessarily a Deep Learning problem, but a common machine learning model problem, where the algorithms are plagued by this computational complexity issue, with training times growing out of reach as data sets and parameter counts grow. Therefore, there is significant motivation to find solutions to improve the efficiency of deep learning models and machine learning models in general.

Many approaches look at improving the classical algorithms, however one field of machine learning that is taking a fundamentally different approach to solving machine learning algorithms, and has gained significant interest over the recent years is Quantum Machine Learning. This sub field of machine learning is relatively unexplored as it proves to be a very challenging discipline, requiring the marriage of Quantum Physics, Statistics, Mathematics and Computer Science. Research within this field is challenging, hence leading to this research field being relatively new and niche. Although it is a niche sub field, there is still significant interest in what progress can be made in Machine Learning through the assistance of Quantum Computers. Quantum Computing needs little introduction to its eye-catching name, and Sci-Fi associations, however most readers most likely will not actually know how a quantum computer works and why it is not in fact science fiction. Quantum computers can complete specific tasks that might be deemed infeasible on classical computers. This was completed for the first time recently by Google, in a landmark paper demonstrating Quantum Supremacy [5] where a computational task took 200 seconds on a quantum computer, but by equivalent benchmarks would take 10,000 years on a classical computer. Previous to this, there was a large amount of anxiety in the Quantum Computing community around if a quantum speed up could actually be attained. However, with this demonstration, we know that certain algorithms can be sped up on quantum computers, thus highly motivating the Quantum Computing community and its associated research. This allure of quantum supremacy is a significant driver behind the motivation to search for Quantum Machine Learning algorithms. Quantum Machine Learning's broad aim is to use Quantum Computers to drastically improve classical machine learning algorithms, or to find machine learning algorithms specific to quantum computers, through the translation of classical algorithms mathematics to a quantum applicable algorithm. These translations can be shifts of a classical algorithm to a quantum algorithm, or a completely novel algorithm that only runs on quantum computers.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

2 Background

2.1 Quantum Computing Basics and Concepts

In the following section, we aim to give a basic introduction to quantum computing and its fundamentals. Quantum computers operate on a fundamentally different architecture than classical computers. Classical computers operate through binary digits (bits), representing data through combinations of 1's and 0's. Everything we see, consume or interact with in the classical computing world is some combination of 1's and 0's, representing the most complex of data structures, playing audio, changing pixel colors and every task a computer accomplishes. Quantum computers, however, operate through a fundamentally different style of data - namely quantum bits (qubits). Qubits can be in a state of either 1 or 0, similar to bits in classical computers, but can also be placed in a probabilistic state of both 1 and 0 at the same time, namely superposition. Superposition is one of the core fundamental ideas to quantum computing, and is one of the significant differentiates between classical mechanics and quantum physics. An extension on the Schrodinger's cat experiment can be a good preamble to understanding superposition.

Schrödingers cat says that if you put a cat in a box with food and poison, then seal the box up, you do not know if the cat is alive. The only way to determine if the cat is alive is by opening the box. We might be able to gain some intuition to the chance of the cat being dead or alive by the amounts of food and poison we put in the box. If we put in 1% poison and 99% food, we could make an educated guess saying that the cat is probably alive. However, if we did the experiment 10,000 times, we might observe that sometimes the cat is indeed dead. More often than not though, we would find the cat to be alive as it would have most likely eaten the non-poisonous food. What we can do though to change our cats chances is shift the ratio of cat food to poison. As we increase the amount of poison in the box, the chance of the cat being alive shrinks.If we have a system of equal parts poison and cat food, we would expect to observe that half of the time the cat would be alive, and the other half dead. If we added more poison to say 99% poison and 1% food, we would expect the cat to be dead more often than not. One notable observation of our experiment is that if we took a system of our cat, observed it was dead, then closed the box and waited an hour, and re-observed it, it would remain dead. As morbid of a thought experiment Schrödingers cat is, it serves as a great analogy to qubits. Quantum systems are the exact same. We prepare them in these probabilistic state of between "dead" (1) and "alive" (0).

With the slight detour of Schrödingers cat described, we hope that the analogy might make the following section marginally easier to understand. Although a qubit can be both in a simultaneous state of 1 and 0, we can never directly observe this. When a qubit is measured, the qubit will fall out of superposition and collapse into one of two possible states (i.e 1 or 0). Although we cannot directly measure and describe a qubit's state with one observational collapse, if we keep repeating this process, the qubit will measure 1 or 0 in a proportion correlated to the superposition it was prepared in. To indicate quantum mechanical states, a specific notation is used, namely the $\langle bra|$ and $|ket\rangle$ notation, where $\langle bra|$ indicates a horizontal state vector ($1 \times n$ vector) and $|ket\rangle$ a vertical state vector ($n \times 1$ vector). The chance of collapsing into 1 or 0 is directly related to how said qubit was placed in superposition. The measuring of the states 1 or 0 is by choice, and measurements are in fact up to how one interprets measuring a qubit. This is best understood through the Bloch sphere visualization, visualized in Figure 1. As visualized in Figure 1, the quantum state of $|\phi\rangle$ is some quantum state in between $|0\rangle$ and $|1\rangle$. However, this state is also in between the x-axis, and the y-axis. When measuring a qubit, one can measure in any direction. The most common being the Z-axis, i.e. the $|0\rangle$ and $|1\rangle$ states, however one can also measure across the x-axis, y-axis, or any axis parameterized by a horizontal angle and a vertical angle.

We briefly relate this thought process to how the Schrödingers cat thought experiment, where we are asking the question "is the cat dead or alive". However, there are many questions that can be asked, such as "is the poison gone or not", or "is the food eaten". These are all correlated, however they all imply different results. This is mentioned as it highlights how quantum systems give answers based on the "question" they are being asked. In the case of qubits, this "question" is the direction we are measuring in; be it left, right or any other direction. The sole constraint one must abide by is that the resulting measurement will always be one of two orthogonal unit eigen vectors on the axis we are observing on. For example, in the case of the x-axis, one could measure either the x-axis pointing inward or outward. For this paper, we only consider the case

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

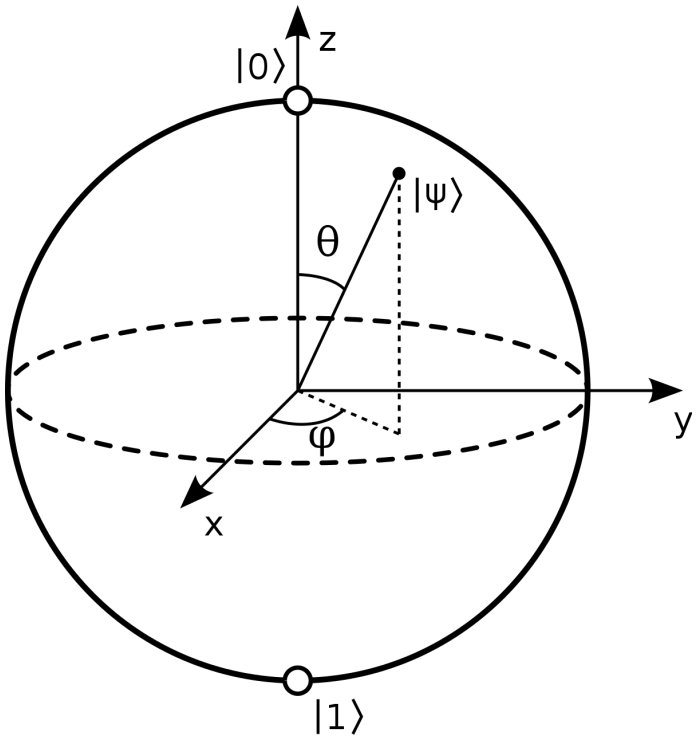


Figure 1: Bloch Sphere Representation

where possible measurement results are 1 or 0, however understanding how qubits are measured is included for completeness and better quantum computing understanding. As mentioned prior, the quantum state of $|\phi\rangle$ is some arbitrary qubits superposition. The vector, namely the state vector, describes the probability distribution of measuring certain outcomes from the quantum state it describes relative to the axis we are measuring. Using this notation, we introduce a common representation describing a single qubit state relative to the states $|0\rangle$ and $|1\rangle$ using $|ket\rangle$ notation, as shown in Equation 1.

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1}$$

The $|\Psi\rangle$ indicates the state vector describing a single qubits superposition. The α and β indicate the square root probabilities of the qubit measuring in said coefficient's state. The probability of measuring $|0\rangle$ is equal to α^2, and $|1\rangle$ is equal to β^2. Therefore, when performing operations on these quantum states, we can describe how much it will affect our observation probabilities by looking at how the operations affect the overall state vector. Examples of the state vector representation of quantum states are presented in Equation 2.

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, |\Psi\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{2}$$

It is of little use to only be able to describe one qubit. Most systems comprise of more than 1 qubit, and with quantum entanglement,which we cover later, there is a requirement of holistic state views instead of single qubit state vector descriptions. Therefore we introduce how one describes the state of two qubits, we take the tensor product ($\otimes$) between two quantum states of a qubit.

$$|\phi\rangle = \gamma|0\rangle + \omega|1\rangle \tag{3}$$

Taking the tensor product of single qubit states, described in Equation 2 and 3, we can describe a quantum state vector of multiple qubits. The quantum system can be represented by a probability distribution of attaining 00,01,10, and 11, shown in Equation 4

$$|\Psi\phi\rangle = |\Psi\rangle \otimes |\phi\rangle = \gamma\alpha|00\rangle + \omega\alpha|01\rangle + \gamma\beta|10\rangle + \omega\beta|11\rangle \tag{4}$$

This tensor product procedure is the same for all quantum states of varying qubit counts, however as is noted this space is exponential. The number of possible measurable is equal to 2^n, where n is the number

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

of qubits. This exponential state space is part of the power of quantum computers, however simulating the states rapidly becomes computationally challenging. This is easily understood when realizing that the number of possible outcomes of n qubits, where each qubit has 2 outcomes, is similar to that of flipping a coin. n coin flips has 2^n possible outcomes, similar to a quantum system of n qubits. As we grow our parameter count space, we still must abide by the laws of probability. Therefore, all quantum state vector coefficients squared-sum must equal to 1, due to the laws of probabilities.

Having established how quantum systems work and how data flows, we now bring light to quantum data manipulation. Similar to classical systems, we make use of quantum gates. To manipulate a quantum state, quantum computers make use of quantum gates. These quantum gates are operations that manipulate the probability amplitudes over the quantum state space. Quantum gates can operate on 1 or multiple qubits. One of the most important and commonly recognized single-qubit gates is the Hadamard gate – mapping a qubit from the basis state $|0\rangle$ or $|1\rangle$ to an equal superposition of $|0\rangle$ and $|1\rangle$. Gates are described by a $2^n \times 2^n$ matrix, where n is the qubit count involved over the gate. Arguably one of the most important and iconic gates in Quantum Computing is the Hadamard Gate. The Hadamard Gate takes in a qubit in the state of $|0\rangle$ or $|1\rangle$ and will transform the state into an equal superposition of both $|0\rangle$ and $|1\rangle$; In this state relating to Equation 2, $\alpha = \beta = \frac{1}{\sqrt{2}}$.The matrix representation of a Hadamard gate is described in Equation 5

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \tag{5}$$

To describe the state change of a qubit after being passed through a quantum gate, the transformation of $|\Phi_{new}\rangle = GATE|\Phi_{old}\rangle$ is performed, with $|Phi_{new}$ being the qubits state leaving the gate. This state change through quantum gate operations is illustrated in Equations 6 through 8, where a qubit in the state of $|0\rangle$ is passed through a Hadamard gate.

$$|\phi_{in}\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{6}$$

$$|\phi_{out}\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{7}$$

$$|\phi_{out}\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{8}$$

As seen in Equations 6 through 8, the transformation of quantum states through gates is described through unit unitary linear matrix operations. Equations 9 through 15 introduce the specific quantum gates that are pertinent to this paper.. As an aside for the avid reader, all quantum gates must be unitary hermitian matrices:

$$R_Y(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{9}$$

$$R_Z(\theta) = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{-i\frac{\theta}{2}} \end{bmatrix} \tag{10}$$

$$CRY(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ 0 & 0 & \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{11}$$

$$CRZ(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{-i\frac{\theta}{2}} & 0 \\ 0 & 0 & 0 & e^{-i\frac{\theta}{2}} \end{bmatrix} \tag{12}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

$$RYY(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & 0 & 0 & i\sin\frac{\theta}{2} \\ 0 & \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} & 0 \\ 0 & -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} & 0 \\ i\sin\frac{\theta}{2} & 0 & 0 & \cos\frac{\theta}{2} \end{bmatrix} \tag{13}$$

$$RZZ(\theta) = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 & 0 & 0 \\ 0 & e^{-i\frac{\theta}{2}} & 0 & 0 \\ 0 & 0 & e^{-i\frac{\theta}{2}} & 0 \\ 0 & 0 & 0 & e^{-i\frac{\theta}{2}} \end{bmatrix} \tag{14}$$

$$CSWAP = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

The gates described in equations 9 through 14 allow us to completely manipulate a quantum state, both when measuring for a $|1\rangle$ or $0\rangle$ and when measuring against some arbitrary axis described by a horizontal and vertical angle. The gate described in 15 will swap two quantum states, $|\phi\rangle$ and $|\psi\rangle$, if a qubit it is controlled on measures as a $1\rangle$. This is tied to quantum entanglement, which is covered in the proceeding section. These gates form the foundation of quantum data processing within our work: taking in some quantum data and performing parameterized operations to produce new manipulated quantum data. Combining these gates with a set of qubits into a sequence of quantum gates operating on qubits at different time steps all in some state forms the foundation of a quantum circuit. Designing quantum circuits with quantum gates and qubits allows for quantum state manipulation, however to attain a classical result from a quantum circuit one needs to add classical bit streams in parallel. In order to observe a result, a measure gate is introduced to map a qubit onto a classical bit, thereby measuring the aforementioned collapse of a 1 or 0. Measuring one quantum state would be of little use, as a classical computer could simply do the calculation instead. Rather, quantum computing's utility comes from the fact that we can sample a distribution of the qubits measuring 1 and 0 over a large number of iterations, generating a probability distribution.

2.2 Quantum Entanglement and SWAP Test

Another fundamental topic of quantum computing is entanglement. Quantum entanglement is a phenomena where by measuring one qubit, one immediately knows information about the state another entangled qubit will measure as. Quantum Entanglement can broadly be thought of as the premise that when qubits become entangled, their states can no longer be described independently of each other and they must be described in one holistic state, and their dependence and interaction with each other persists no matter the distance between them. Two qubits that are entangled and that are on opposite ends of the universe will exhibit their entanglement properties faster than the speed of light could travel between them. This is another key distinguishing topics of quantum systems over classical mechanics. Entanglement can get exceptionally confusing, and is best understood through the math behind Bell's experiment, where the measurement of one qubit tells you what another qubit will measure as. A saying that comes up continually in quantum physics is to not obsess over finer details and just follow the math, for it will all make sense.

Bell's experiment consists of two qubits, namely qubits $|\phi\rangle$ and $|\psi\rangle$. Both of these qubits are initialized in the state of $|0\rangle$. We pass the $|\phi\rangle$ qubit through a Hadamard gate, resulting in the following system transition:

$$|\phi\rangle = H|0\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \quad |\psi\rangle = |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{16}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

We now can describe our entire system by taking the tensor product between the qubits $|\psi\rangle$ and $|\phi\rangle$:

$$|\phi\rangle \otimes |\psi\rangle = |\phi\rangle = \begin{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} \tag{17}$$

$$|\phi\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + 0|01\rangle + \frac{1}{\sqrt{2}}|10\rangle + 0|11\rangle \tag{18}$$

What we observe in Equation 18 is that the possible states that could be observed is either a $|00\rangle$ or $|10\rangle$. Therefore, the second qubit will measure 0 no matter what, but the first qubit will measure 1 or 0 with equal probabilities. We proceed by applying a CNOT gate to this quantum state:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{19}$$

$$|\phi\psi\rangle_{out} = CNOT|\phi\psi\rangle_{in} \tag{20}$$

$$|\phi\psi\rangle_{out} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} \tag{21}$$

$$|\phi\psi\rangle_{out} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} \tag{22}$$

$$|\phi\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + 0|01\rangle + 0|10\rangle + \frac{1}{\sqrt{2}}|11\rangle \tag{23}$$

Qubits can become entangled by being operated on by certain gates, such as the CSWAP gate described in 11. This phenomenon is very powerful, as in the case of machine learning, we can create entangled states where information from a certain dimension affects the information in other dimensions. The parameterized gate used in this paper which leads to entanglement, known as the $CRy(\theta)$ and $CRz(\theta)$gate, is described in Equations 11 and 12. These gates is how we accomplish trainable entanglement within our paper.

One extremely useful use of entanglement is the SWAP test. Measuring quantum state fidelity can be done through the use of the SWAP test. The SWAP test is a quantum algorithm that measures the fidelity between two quantum states through the use of a CSWAP gate, and two Hadamard gates and two quantum states one wants to measure the fidelity of. The SWAP test algorithm requires two quantum states, $|\psi\rangle$ and $|\phi\rangle$, whose similarity is to be measured, and a qubit in state $|\xi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. The two states of $|\psi\rangle$ and $|\phi\rangle$ are Control-SWAPped with the control qubit being $|\xi\rangle$, where the states are swapped if $|\xi\rangle$ qubit measures $|1\rangle$. This leads to the state of $|\xi\rangle$ being entangled with the other states, and having its probability of measuring $|1\rangle$ being dependent on the fidelity between $|\phi\rangle$ and $|\psi\rangle$. If $|\phi\rangle$ and $|\psi\rangle$ are orthogonal (i.e as different as can be), $|\Phi\rangle$ will measure 1 with a probability of 0.5. As the states become more similar, the probability increases towards 1, where 1 indicates the states are the same. This is visualized in Figure 2, where the anicilla qubit starting in state $|0\rangle$ is the aforementioned $|\xi\rangle$ qubit.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

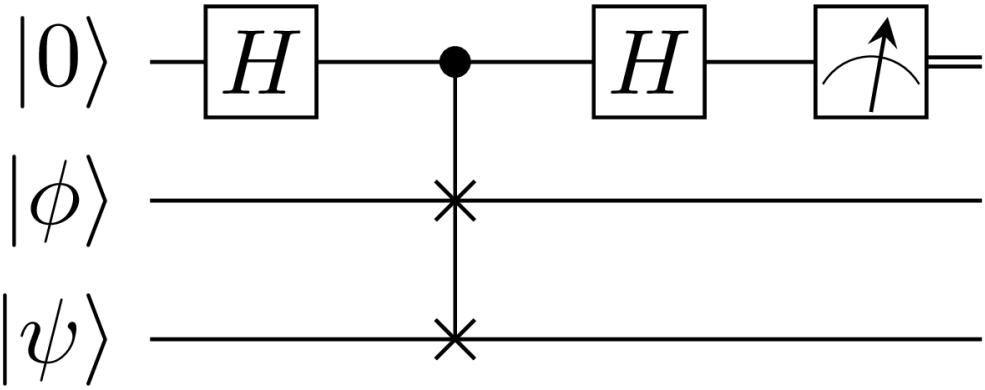


Figure 2: SWAP Test

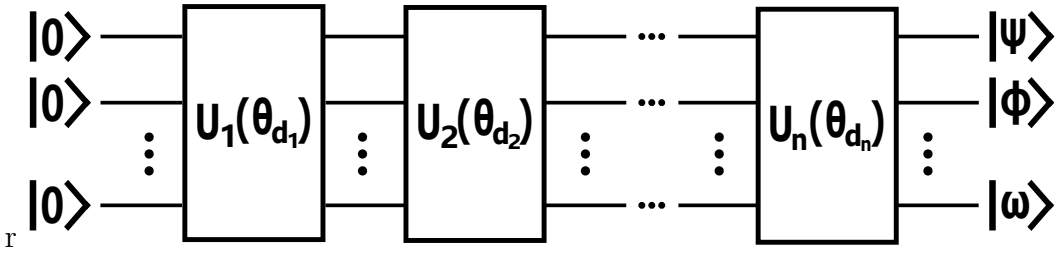


Figure 3: Quantum Deep Learning Layers

2.3 Quantum Deep Learning

Quantum Deep Learning is a relatively new approach to Quantum Machine Learning that takes quantum circuits and applies similar training techniques and learning methods of how classical neural networks work [15, 25, 6]. In traditional deep learning layers are often used, where a layer is some large transformation function that takes in a set of inputs, and outputs a set of outputs, where the number of inputs does not necessarily equal the output. These functions are connected in series, sometimes in parallel, and typically trained through the use of back propagation [26, 15]. Deep learning can be characterized as some function taking in variables $[x_1, x_2, ..., x_n]$ parameterized by $[\theta_1, \theta_2, ..., \theta_n]$, applying a transformation, outputting some variables $[y_1, y_2, ..., y_n]$. This can broadly be thought as $Y = f(X, \theta)$ This data flow through layers to some output is similar to how quantum circuits operate. Similar to how classical deep learning models are designed, the way this data flows through time is up to the practitioner, who chooses and designs their network according to their needs.

 doi:10.20944/preprints202103.0583.v1

3 Related Work

3.1 Quantum Deep Learning

With the recent advances in this field, quantum computing introduces exciting possibilities to enhance the existing learning algorithms and architectures through qubits, the basic unit in quantum information theory. Great efforts have been made to develop a quantum based learning algorithms. In [25], authors conduct a comparative study of classic deep learning architectures with various Quantum-based learning architecture from a different perspective. The first challenge researchers encountered is how to represent classical data (binary states) with quantum states. Different methods have been proposed to address it [18, 57, 36]. Cortese et al. [18] discusses a set quantum circuits and associated techniques that are capable to efficiently transfer binary bits from the classical domain into the quantum world. However, the proposed methods require $Log_2(\mathcal{N})$ qubits and $O(Log(\mathcal{N}))$ depth to load $\mathcal{N}$ bits classical data. Aiming to reduce the complexity, qGANs [57] utilizes quantum Generative Adversarial Networks to facilitate efficient learning and loading of generic probability distributions, which implicitly given by data samples, into quantum computers. With the data loaded into quantum states, authors in [3, 45, 8, 10, 46, 14, 13] constructed quantum versions of classical machine learning algorithms, such as K-means and support-vector machine, to address classical learning problems, e.g. pattern recognition. Many of these works utilize various quantum states and operations are studied to measure the relative distance between classical data points. For instance, with a basic quantum interference circuit that consists of a Hadamard gate and two single-qubit measurements, [46] is able to evaluate the distance measure of two data points. As quantum computers quickly evolve, quantum neural networks rapidly gained attention in the field. A neural network architecture for quantum data has been proposed in [23], which utilizes the predefined threshold to build a binary classifier for bit strings. With quantum deep learning however, a method of performing gradient descent needs to be developed. Focusing on optimizing variational quantum circuits, Stokes et al. [49] introduces a quantum generalization of natural gradients for general unitary and noise-free quantum circuits. Iterations on these base quantum deep learning models continue to be proposed, such as a design of a quantum convolutional neural network proposed in [17] with guaranteed $O(Nlog(N))$ variational parameters for input sizes of N qubits that enables efficient training and implementation on realistic, incoming quantum computers. Based on a general unitary operator that acts on the corresponding input and output qubits, authors in [6] propose a training algorithm for this quantum neural network architecture that is efficient in the sense that it only depends on the width of the individual layers and not on the depth of the network. More recently, QuantumFlow [31] proposes a co-design framework that consists of a quantum-friendly neural network and an automatic tool to generate the quantum circuit. The existing works primarily investigate binary classification, which is an important problem. This problem setting, however, significantly limits the scope of the designs. Furthermore, many solutions suffer from having low accuracy's, missing a comprehensive quantum architecture, and requiring an infeasible number of qubits. We introduce a quantum architecture that employs a quantum state based loss function and a quantum-classic hybrid architecture that addresses the current limitations in classification.

3.2 Quantum Generative Adversarial Networks

GANs are a deep learning architectures where two networks compete against one and other, with each network trying to fool the other [27]. The model is comprised 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 data and fake data synthesized by G.The generator's objective is to synthesize samples that the discriminator will classify as real. Due to this architecture, GANs are agnostic of the type of data they are fed, however performance is strongly dictated by the network architecture [52]. In traditional generative networks, some distribution p_z is sampled from where G uses this sample to synthesize a sample fed to D. D is fed with both real data x and data synthesized by G. The discriminator's output is a probability, and its goal is to maximize $D(x)$ and to minimize $D(G(p_z))$. The generator's goal is to maximize $D(G(p_z)$. If we label the real data as 1 and

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

fake data as 0, this min-max optimization problem described in Equation 24.

$$min_G max_D V(D,G) = \mathbf{E}_{x\ p_{data(x)}}[log(D(x)]+ \\ \mathbf{E}_{z\ p_z(z)}[log(1-D(G(z))] \tag{24}$$

GANs show exceptional results in computer vision [52] and even in time-based domains such as music composition [21]. These results only get better, with significant strides being made on classical GANs every year [4, 52]. One direction within GAN-related research is the potential of quantum GANs. 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. Current works propose the use quantum GANs for uses ranging from attempting to attain a quantum speed up, or for data loading onto quantum computers [36, 30, 57, 50, 6, 15]. For example, Zoufal et al. [57] presents qGAN to learn random distributions for financial applications. Their work, however, is only extended to one dimensional data learning, and displays instability within their models evaluation. Furthermore, they make use of a classical discriminator, and hence making their qGAN only a partial quantum GAN that might not be fully accelerated by quantum systems (we use Qi-GAN to refer to the results of qGAN in the evaluation section to better distinguish between our proposed QuGAN and qGAN). Another example is [19]. In this project, authors present quantum GAN architecture. However, their learning is sporadic and inconsistent, with random convergence and no clear trend. Based on the quantum state based loss functions with modified quantum state based loss functions, the proposed QuGAN provides a more stable version of quantum GANs.

Extending our work on quantum deep learning for multi-class classification, we theoretically study the quantum inspired algorithms [12, 11] and employ our architecture on a Quantum GAN [48] and illustrate significantly better learning stability and an extension on current literature's Quantum GAN data set domain.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

4 Design: Quantum Deep Learning Architecture

4.1 System Design for Classification

Our proprietary quantum deep learning architecture operates through a feedback loop between the Classical Computer and the Quantum Computer, illustrated in Fig. 4. Data is initially parsed through cleaning and normalization, removing any significant outliers. We normalize our data as the expected value of a qubit can only range from 0 to 1, as those are the measurable states. Data is parsed and qubitized through a quantum data encoding algorithm transforming classical data into quantum data, and is now in the form of a classical quantum data set. The data set is now comprised of classical parameters to prepare a quantum state representing each data point. For each class in the dataset, a quantum state is initialized with the same qubit count as the number of qubits in the classical quantum data set. The quantum states along with quantum classical data are sent to a quantum computer, where the quantum circuit is created using the parameters and gates that come from the classical computer. The quantum computer calculates a quantum-metric of loss between the data, and sends the loss metric back to the classical computer. The classical computer uses this information to update the learned state parameters to minimize the loss. This is iterated until the desired convergence or sufficient iterations have been completed. This iterative quantum machine learning architecture is similar to that in [47, 48].

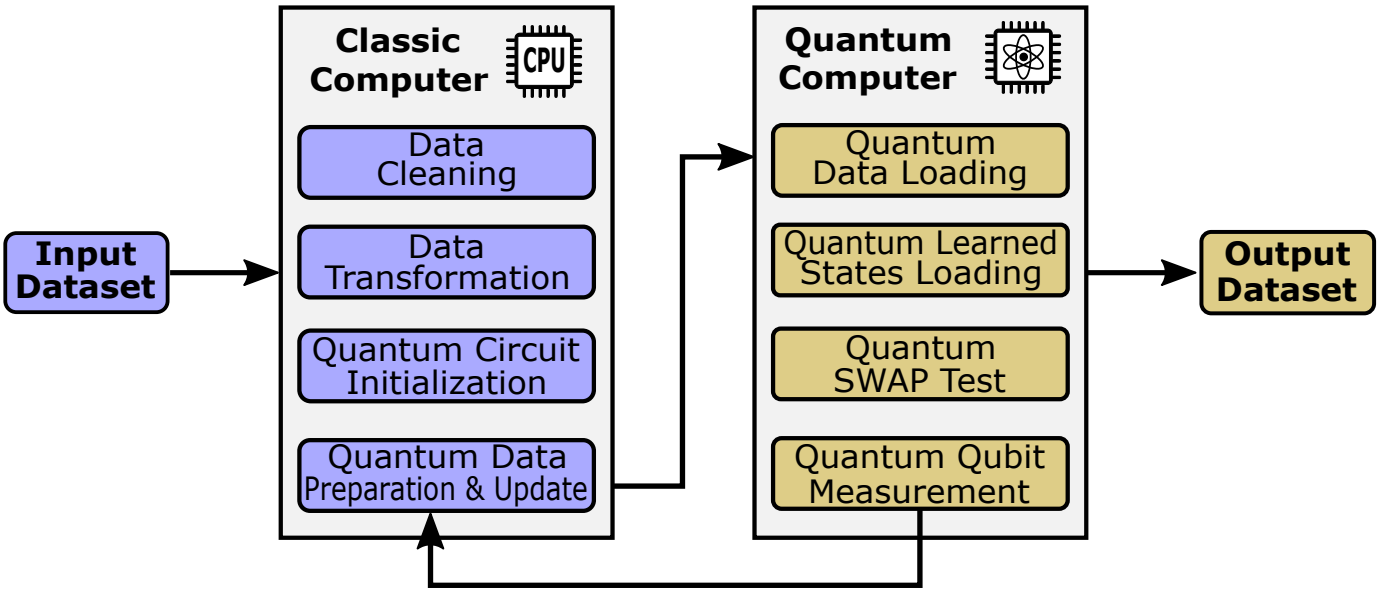


Figure 4: Quantum-Classic Deep Learning

4.2 Quantum Layers

The design of a quantum neural network can be separated into layer architecture and layer count. The abstraction of quantum layers is not new, broadly referenced to as a Variational Quantum Circuit [15], however our direct layer architecture is proprietary to this research. Layer architecture involves designing a layer based on the desired style of manipulation of the quantum state. These are broken down into single qubit unitaries, dual qubit unitaries, and controlled qubit unitaries. Single qubit unitaries involve rotating a single qubit by some parameters θ_d.

A qubit enters a single qubit unitary, drawn in Fig. 5, in a certain state, undergoes a rotation around the Y and Z axis, leaving it in the final state. A rotation around the Z and Y axis allows for complete manipulation of a single qubit.

With a dual qubit unitary, drawn in Fig.6, two qubits enter in their each respective gate, followed by an equal Y and Z rotation on both qubits. The rotary vector performed on both qubits are equal.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

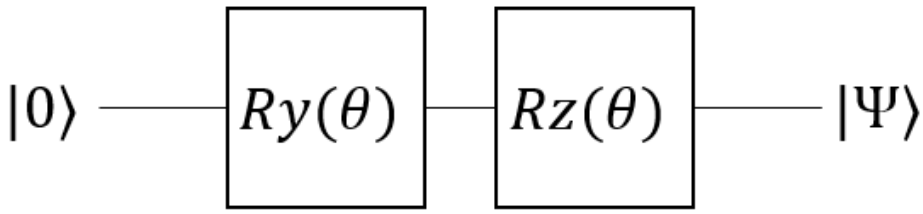


Figure 5: Single Qubit Unitary

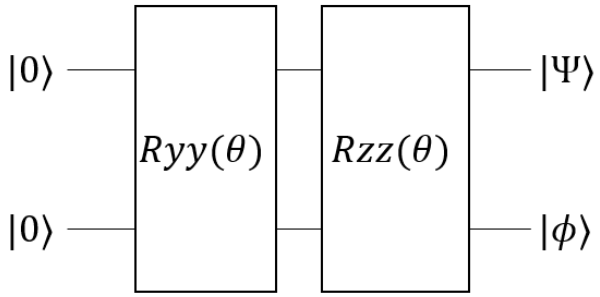


Figure 6: Dual Qubit Unitary

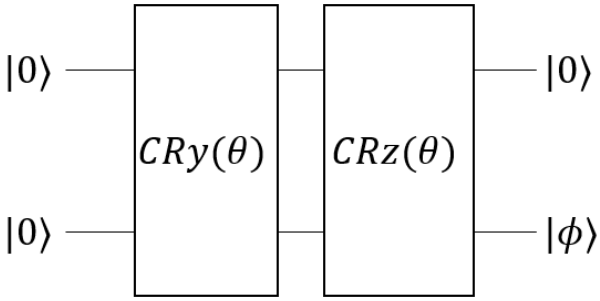


Figure 7: Entanglement Unitary

The final style of qubit operations we mention are entanglement operations, drawn in Fig.7. Entanglement operations lead to ties between qubits, having other qubits be affected based on the measured result of prior qubits. This layer exploits a powerful aspect of quantum computing, allowing for qubits to be manipulated based on what the outcome of other qubits end up in. In the above layer design, two qubits are passed in with one qubit being designated a control qubit. If the control qubit measures $|1\rangle$, the operation of $RY(\theta)$ will have happened on the other qubit, similarly with the $CRz(\theta)$ operation.
We can combine these unitaries to form a quantum layer, parameterized by each independent θ in the layer, where the layer is seen as a group of these operations, however the layered approach allows for a higher level design.

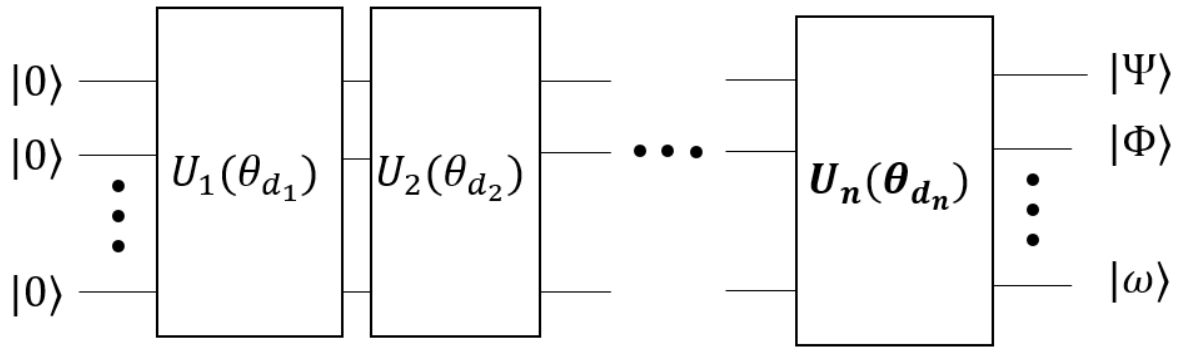


Figure 8: Layered Quantum Deep Learning Architecture

The use of these, drawn in Fig. 8, layers allows for the abstraction to a layered design, similar to that of traditional deep learning. Each layer is similarly parameterized by some set of values θ_d. These parameters can be trained such that the final output of the quantum circuit minimizes some cost function. However, in a quantum setting, minimizing some cost function is more abstract as quantum data is not necessarily measurable, and to what extent we want to measure it. Therefore, this requires explicit and competent system design targeted to accomplish the task at hand. In our case of multi class classification we make use

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

of multi-class cross-entropy along with the SWAP test to train our network. For our GAN architecture, we make use of a similar SWAP test architecture however with a Generator/Discriminator loss function instead and an extra quantum deep learning model.

4.3 Data Qubitization

An often overlooked discussion in quantum machine learning papers is how to represent classical data in the quantum setting. We propose the following approach, whilst acknowledging that there may be better ways to accomplish data encoding on quantum systems. Some data $x_1, x_2, ..., x_n$ of dimension d can be translated into a quantum setting by normalizing each dimension d_i to be bound between 0 and 1. This information is now in the same domain as the possible expectation measurement of a qubit. Encoding a single dimension data point only requires one qubit, unlike classical computing which requires a long string of bits to represent it. To translate traditional value x_i into some quantum state, we perform a RY gate parameterized by the following function: $RY(\theta_{x_i}) = 2sin^{-1}(\sqrt{x_i})$. This results in the expectation of a qubit being measured against the Z axis equating to the x value from the traditional data. We make use of two parameterized rotation on one qubit initialized in state $|0\rangle$ to prepare classical data in the quantum setting. To encode a data point, we prepare this rotation across $\frac{d}{2}$ qubits, each parameterized by each dimension normalized value of that data point. It is worth noting that the 2-dimensional encoding onto one qubit can be problematic in extreme values of x, however we explore the dual dimensional encoding as a possibility and evaluate the performance if we encode each dimension of data into one respective qubit solely through a RY Gate. Furthermore, as we are never actually going to measure the data qubits due to the use of the SWAP test, this is less of a problem. Therefore, we encode the second dimension of data on the same qubit through the following rotation:

$$RZ(\theta_{x_{i+1}}) = 2sin^{-1}(\sqrt{x_i}) \tag{25}$$

The use of a second dimension on one qubit when taking measurements would usually be problematic, as well as the differentiation being used in this paper states that it is only viable for qubits being measured along two orthogonal eigen vectors. However, we never actually measure our quantum states as we use the SWAP test and hence the use of 2 dimensions per qubits with this differentiation formulation is acceptable. When number of qubits is a concern or qubit counts are infeasible, methods that will reduce this number are extremely valuable. Classical data encoding in quantum states has no method that is tried and tested, therefore our approach can certainly be easily critiqued. However, when tested the approach seems to be a viable approach to the problem.

4.4 Implementation of The Swap Test

When training a network to accomplish a task, an explicit description of a system improvement goal needs to be established. In the quantum setting, we could be manipulating the expected values of each qubit in some way, but even this is ambiguous - we haven't defined which direction we would measure in or what our iteration count would be for measuring expectation with lower iteration counts leading to more noise. Fortunately for our system, due to the SWAP test we do not need to measure more than one qubit. The measurement of the anicilla qubit, outlined in Fig.2, measures the quantum state fidelity between two quantum states. This fidelity can be translated to the deep learning setting as an activation with a target. In the case of binary classification, each data point is represented in a quantum state represented by $|\phi\rangle$, which can be used to train $|\omega\rangle$ such that the SWAP test distance between the states is minimized. The approach is modelled as:

$$min_d(Cost(\theta_d, X) = \frac{1}{n}\sum_{i=1}^{n} SWAP(|\phi_{X(i)}\rangle, |\omega\rangle) \tag{26}$$

Where θ_d is a collection of parameters defining a circuit, x is the dataset, $\phi_{x(i)}$ is the quantum state representation of data point i, and ω is the state being trained to minimize the function.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

4.5 Optimization of the Quantum Circuit

Optimization of parameters in iterative machine learning algorithms can be performed through gradient descent of the cost function. For the case of classification, we want to maximize the similarity - quantum fidelity - of a quantum state and labeled data of label λ_i, and minimize the similarity between the quantum state and data that is not λ_i. This probabilistic distance between states seems to naturally lead us to believe that the binary cross entropy function will be well suited to the problem.

$$Loss = -ylog(p) - (1-y)log(1-p) \tag{27}$$

Where y is 1 if it is the targeted label λ, and 0 otherwise, and p is the value the SWAP test returned. In the case of a GAN, the original formulation indicated under Equation 24 would be what we would optimizing for. Therefore, we are maximizing fidelity of real data to a quantum state and reducing fidelity of fake data to a quantum state. This training happens whilst we also try to allow a quantum state to generate samples similar to that of the discriminator quantum state. However, when handling quantum fidelity, the swap test returns probabilities between 0.5 and 1, therefore we scale these probabilities to be between 0 and 1, as well as ensure that they are always positive to avoid invalid logs. Due to noise, similarities of less than 0.5 can occur, hence the requirement of the absolute value function.

$$p = abs(\frac{\frac{n_1}{n_0+n_1} - 0.5}{0.5}) \tag{28}$$

The use of the above loss function gives us an indication of the current "quality" of our circuit, to which we want to minimize the loss function. To differentiate our quantum circuit we use a modified version of the parameterized difference; where our modification is to include a $\sqrt{epsilon}$ to encourage smaller parameterized differences at later epochs of training:

$$\frac{\delta Cost}{\delta \theta_i} = \frac{1}{2}(f(\theta_i + \frac{\pi}{2\sqrt{\epsilon}}) - f(\theta_i - \frac{\pi}{2\sqrt{\epsilon}})) \tag{29}$$

Where θ_i is one theta value in the parameter matrix, Cost is how sensitive the overall cost of the circuit is to that parameter, and ϵ is the current epoch training the circuit. Our differentiation method improves the vanishing gradients problem. The introduction of ϵ in the parameter leads to initial large steps being taken as the direction is clear as to which direction leads to better performance, however as the scope of the function is π, a rotation of that magnitude can skip over the minima, or even lead to a gradient of 0 and hence stall.

4.6 Binary and Multi Class Classification using Quantum Deep Learning

The p value returned from the measured and processed SWAP test gives a measure of similarity between a data point and a state trained to discriminate between a class data point and non-class data points. In the case of binary classification, if we were to take the traditional approach of taking the boundary line of $p < 0.5$ means it is most likely not the class, and $p > 0.5$ means it is most likely the class, we run into problematic results. This is due to the distance between Class 0 data and Class 1 data possibly being less than the SWAP tests equivalent distance of 0.5. Therefore, in the case for binary classification, a decision boundary needs to be solved for such as to maximize the correct classifications. As a solid example, if Class 0 SWAPped with state $|\phi\rangle$ - which tries to correctly classify data points as 1 or not-1's - averages a similarity of 60%, and Class 1 SWAPped with similarity of 90%, we would want our decision boundary to most likely lie between 60 and 90%, with the actual number being calculated such as to maximize the evaluation metric of the system.
Using a quantum discriminator state to discern between classes 0 and 1 is useful, however more often than not data sets have significantly more than 2 classes. We propose the solution to this that is similar to that of traditional neural networks. Instead of training one state, we train multiple states against different classes, and soft-max the probabilities over the sum of all state's probabilities. The largest probability is then the classified state. This would lead to their being state $|\phi\rangle_i$ where i is the class, and therefore the number of states being k where k is the number of classes.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

Algorithm 1 Quantum Algorithms

Data set Loading
  Dataset : $(X|Class = mixed)$
Distribute Dataset X By Class
Parameter Initialization:
  Learning Rate : $\alpha = 0.01$
  Network Weights : θ_d = [Rand Num btwn 0 - 1 $\times \pi$]
  epochs : $\epsilon = 25$
  Dataset : $(X|Class = \omega)$
  Qubit Channels : $Q = n_{X_{dim}} \times 2$
for $\zeta \in \epsilon$ **do**
  for $x_k \in X$ **do**
    for $\theta \in \theta_d$ **do**
      Perform Hadamard Gate on Q_0
      Load $x_k \xrightarrow[Data\ Encoding]{Quantum} Q_{Q_1 \rightarrow \frac{Q_{Count}}{2}+1}$
      Load $\theta_d \xrightarrow[Learned\ State]{Quantum} Q_{\frac{Q_{Count}}{2}+1 \rightarrow Q_{Count}}$
      Add $\frac{\pi}{2\sqrt{\zeta}} \rightarrow \theta$
      $\Delta_{fwd} = (\mathbf{E}_{Q_0} f(\theta_d)$
      CSWAP(Control Qubit = Q_0, Learned State Qubit, Data Qubit)
      Measure Q_0
      Reset Q_0 to $|0\rangle$
      Perform Hadamard Gate on Q_0
      Subtract $\frac{\pi}{\sqrt{\zeta}} \rightarrow \theta$
      CSWAP(Control Qubit = Q_0, Learned State Qubit, Data Qubit)
      Measure Q_0
      $\Delta_{bck} = (\mathbf{E}_{Q_0} f(\theta_d)$
      $\theta = \theta - (-log(\frac{1}{2}\Delta_{fwd} - \Delta_{bck})) \times \alpha$
    end for
  end for
end for

One natural critique of this architecture in the current quantum computing era is that it uses many qubits. Classifying a data point of n dimensions would require $k(2n+1)$ qubits. This quickly becomes infeasible. However, we propose the solution being distributed quantum computing. The states do not need to be trained simultaneously, and do not know about each other's current state whatsoever. Therefore, we can separate the training of k states, and train them one at a time or in parallel. This is an ideal task for distributed computing, as each state can be trained independently of each other. Once the states are trained on their respective classes, to classify a data point all we need to do is measure the similarity between each state and a data point. This can be done using a distributed model, or we can measure probabilities of the class one at a time. Thereby allowing us to only use $2n+1$ qubits at run time. Distributed models are preferred for larger qubit counts, as mentioned prior the exponential memory usage of quantum states can lead to larger wait times for classification.

4.7 Training our Quantum Network

We implement the Algorithm outlined as Algorithm 1 for the training of our Quantum Deep Learning architecture. Our algorithm is implemented using the Qiskit python library. We also implement a proof-

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

of-concept training on a real quantum computer in which we train a quantum neural network on a real IBM Quantum Computer. Algorithm 1 introduces certain terms which are chosen by the practitioner. The learning rate is a variable chosen at run time that determines by how much we want to update our learned weights at each iteration. A larger learning rate can lead to taking too large steps that the algorithm will not converge, however too small of a learning rate will lead to longer run times. Qubit Channels indicate then number of qubits that will be involved in the quantum simulation. Epochs indicate how many times we will train our quantum network on the data set. Within the nested for loops in Algorithm 1, lines 7-20, we load our trained state alongside our data point with a forward difference applied to the respective θ (Δ_{fwd}), perform the SWAP test, reset and perform the backward difference to aforementioned θ (Δ_{bck}).

4.7.1 An Example

Taking a data point, $[x]$ with the label "A", with one dimension that is bound by 0-1, and said data point is 0.5. Prior to beginning the algorithm we preprocess the data to a Quantum Setting. We translate the value of 0.5 into a quantum rotation by parsing the following equation:

$$\theta = 2sin^{-}1(\sqrt{0.5}) \tag{30}$$

$$\theta = 1.57rads \tag{31}$$

Stepping through the algorithm above, we start at step 3 as the data is only of class "A". We begin by loading in our default parameters for the algorithm. We load in $\alpha = 0.01$,$\theta_d \leftarrow$ [Random Initialization between 0 and Pi of Lea $\epsilon = 25$, $Qubits = [Q_0, Q_1, Q_2]$. Arbitrarily we will say that the θ_d begins at $[2rad]$. From here, we move into our algorithm where we will repeat the following code ϵ times over data points in $[x]$, over the 1 gate. Beginning the loops in steps 4 through 6. Over each epoch, we perform over each data point, over each gate - looped in that order - we perform the following steps. We begin the process by preparing the Control Qubit in superposition by passing it through the Hadamard gate.

$$|cqbit\rangle = H|0\rangle \tag{32}$$

We load our data into the quantum state by rotating a $|0\rangle$ qubit transforming the qubit into a representation of our data.

$$|A_{data}\rangle = RY(1.57rads)|0\rangle \tag{33}$$

We now load our learned class identifying quantum state and make use of the parameterized gate differentiation to optimize this state based on the data loaded previously.

$$|A_{learned-fwd}\rangle = RY(2.00rads + \frac{\pi}{2\sqrt{\epsilon}})|0\rangle \tag{34}$$

We CSWAP the $|A_{data}\rangle$ quantum state with $|A_{learned-fwd}\rangle$ controlled on the $|cqbit\rangle$. We then perform a Hadamard gate on the $|cqbit\rangle$, followed by measuring the $\mathbf{E_{cqbit}}$. This encompasses the forward difference, which we repeat by testing the system again with the backward difference.

$$|A_{learned-bck}\rangle = RY(2.00rads - \frac{\pi}{2\sqrt{\epsilon}})|0\rangle \tag{35}$$

Repeating the $\mathbf{E_{cqbit}}$ measurement, we calculate the loss based using each $mathbfE$

$$loss = 0.5(-log(\mathbf{E}_{fwd}) - (-log(\mathbf{E}p_{bck}))) \tag{36}$$

The parameter of $|A_{learned}\rangle$, currently 2.0, is updated by $-\alpha loss$. We repeat this over each gate in the learning state, $\epsilon times$. This is repeated over every data point in the class of "A", ϵ times.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

4.8 Quantum GANs using Quantum Deep Learning Architecture

Extending our Quantum deep learning architecture into the generative domain, we propose a modified architecture and use for a Quantum GAN. In this section we introduce a modified system architecture for generating fake samples from a classical dataset over a quantum circuit. One of the benefits of generating samples from a quantum circuit is that all we need to do is generate some quantum state similar to the data set, which then can be sampled from and generate new data points that have never been seen before. This is the natural transition from a classical generator, producing samples in a probability space, to a quantum generator producing the probability space.

In designing a GAN, the complexity of the Generator and Discriminator should be comparable. A primitive discriminator pitted against a complex generator will accomplish very little, as the gradients for the generator are passed from the discriminator. We introduce our adapted system architecture for QuGANs in Figure 9. The system begins by being fed classical data, which is converted into quantum data. This quantum data alongside the Quantum Circuit is loaded onto a quantum computer. The quantum circuit is run, and quantum state fidelity's from the SWAP test are passed back to the classical computer. These fidelity's are used in the calculation of each gates gradient, which we use to update the Generator and Discriminator parameters. This architecture is repeated n times, or until a certain goal has been reached.

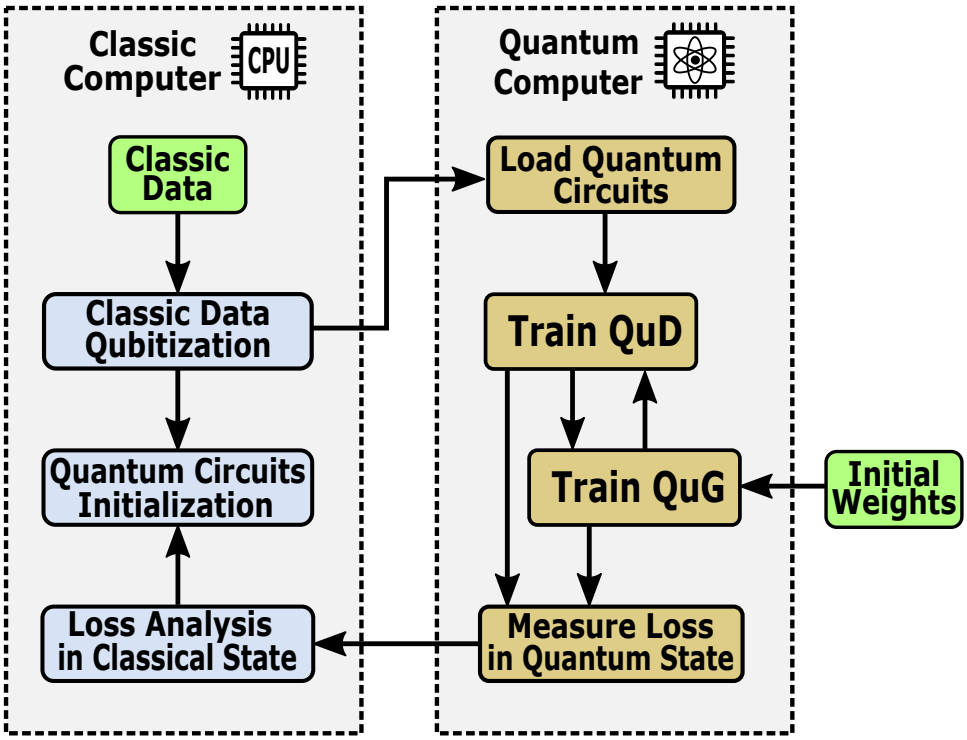


Figure 9: System Architecture

4.8.1 Generator & Discriminator Architecture

In past Quantum generative research, models have looked at taking a traditional Discriminator but changing the Generator to a quantum Generator [15], or even implementing a Quantum Wasserstein GAN. The problem with the Quantum/Classical approach however is that a classical generator cannot 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 EM distance over a density matrix, however in our case measuring the EM distance of a single qubit is relatively equivalent to the original log loss function. We make use of a Quantum Generator as well as a Quantum Discriminator, and accomplish communication between the two 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.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

Algorithm 2 Quantum Algorithms

1: Parameter Initialization:
   Learning Rate : $\alpha = 0.01$
   Network Weights : $\theta_d = [\text{Rand Num btwn 0 - 1} \times \pi]$
   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: $Grad = \frac{dCost_G}{d\theta_D}$
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: $Grad = \frac{dCost_D}{d\theta_G}$
16: Update $\theta_G \leftarrow \theta_D - \alpha Grad$
17: **end for**
18: **end for**

The Algorithm 2 illustrates the process we go through to implement our design. In Line 1 we initialize all of our initial parameters. The network weights line indicates the initialization of the parameter vector. Each entry is initialized as a random number between 0 and π. We set our epoch count, load our data set, and indicate the number of times 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. To produce classical results from a QuGAN, we sample from the Generator circuit n times and take the mean of those measurements per qubit. In the following evaluation section, we set $n = 30$ to attain a data point (For example, we might sample on a 1 Qubit GAN $|0\rangle$ 14 times, and $|1\rangle$ 16 times, therefore giving us the measured value of $\frac{(14(0)+16(1))}{30} = 0.533$.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

5 Quantum Deep Learning Results

5.1 Multi-class Classification

5.1.1 System implementation and Experiment settings

We implement our Quantum Deep Learning architecture with Python 3.8 and IBM Qiskit, a GPU-accelerated Quantum Computing simulator package. The circuits were trained on a single-GPU machine (RTX 2070). Furthermore, to compare our architecture to traditional network results we utilize Tensorflow – a neural network programming package by Google. Additionally, we compare our architecture with Tensorflow Quantum, a quantum framework of traditional Tensorflow based off of the quantum simulator Cirq. In the following results, QuClassi and QuGAN refer to our architecture.
When analyzing our results, certain important architectural terms are used. We describe our quantum neural networks by which qubit-manipulation layer, or what combination of layers were used. Discussed under Section 4 with the three types of qubit manipulation, these types are a type of layer and have a respective name that lists below.

- QuClassi-`S`: A layer that performs one single qubit unitary per qubit is namely a Single Qubit Unitary Layer.
- QuClassi-`D`: A layer that performs a dual Qubit unitary per qubit combination is a Dual Qubit Unitary Layer.
- QuClassi-`E`: A layer that entangles qubits through controlled rotations is an Entanglement Layer.
- QuClassi-`DE`: A QuClassi model that consists of a Dual Layer and an Entanglement layer

When comparing to traditional neural network methods, our primary interest when designing comparable networks is a similar number of parameters. Therefore when referencing a traditional neural network we use the term Deep Neural Network, or DNN. Furthermore, this is preceded by kP, where k is a number indicating the total parameter count of the network. For example, 12P DNN means a deep neural network with 12 parameters.

5.1.2 Quantum Classification of The Iris Dataset

A common data set that machine learning approaches are tested on for proof of concept is the Iris Data set. This data set comprises of 3 classes, namely Setosa, Versicolour and Virginica , and a mix of 150 data points, each containing 4 numeric data points about one of the flowers. For this data set to be encoded in quantum states, we perform the quantum data encoding algorithm described above. This data is encoded in the quantum state by performing an RY gate followed by a RZ gate, encoding 2 dimensions on one qubit. To classify all three classes, we have three target qubit states $|Setosa\rangle$ (Class 1),$|Versicolour\rangle$ (Class 2) and $|Virginica\rangle$ (Class 3). The data set is separated into its respective classes, and used to train its respective quantum state.Taking the data values as x_1, x_2, x_3 and x_4 and trained parameters as $\theta_1, \theta_2, \theta_3$ and θ_4 we program our quantum circuit as shown in Fig. 10.

Where the discriminator state is loaded into the qubits 1 and 2, and the data loaded into qubit 3 and 4. This circuit is a discriminator circuit of $Class1$, and has to be run with $Class2$ discriminator state loaded into qubits 1 and 2, and then $Class3$ with the same methodology. These probabilities are softmaxed which is then used to make a classification. We train our circuit iteratively with a learning rate of $\alpha = 0.01$ and over 25 epochs. We illustrate our gradients, loss function and accuracy as a measure of epoch, and illustrate the significant improvement that we have attained in stability of quantum machine learning, in Fig. 11.

This initial Iris dataset that was used as a proof of concept and is not very complex, and hence having complex layers can be unnecessary for the problem. This leads to no actual gain from deepening our circuit. Fig. 12 covers the notable results of our experiments with a learning rate of $\alpha = 0.01$ and epoch count of 25.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

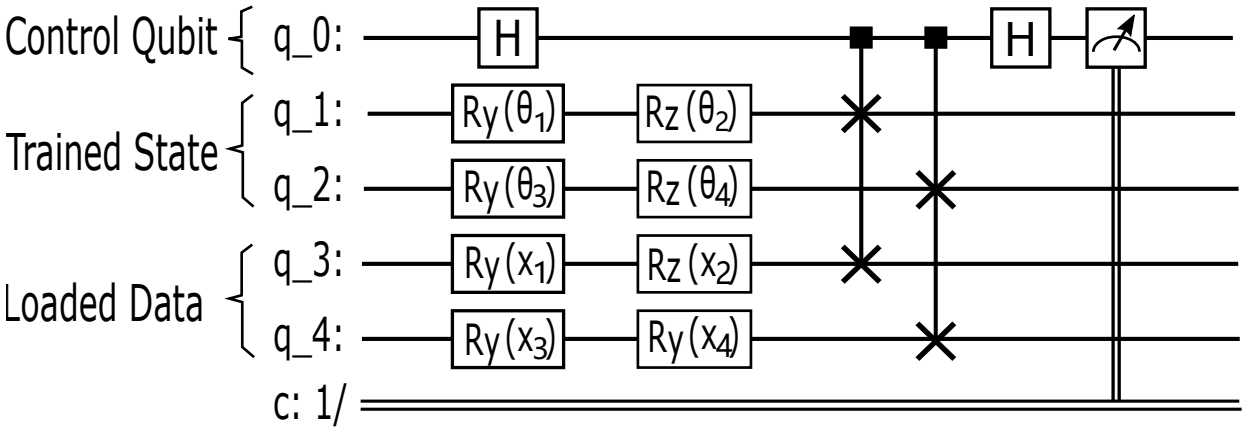


Figure 10: Sample Circuit

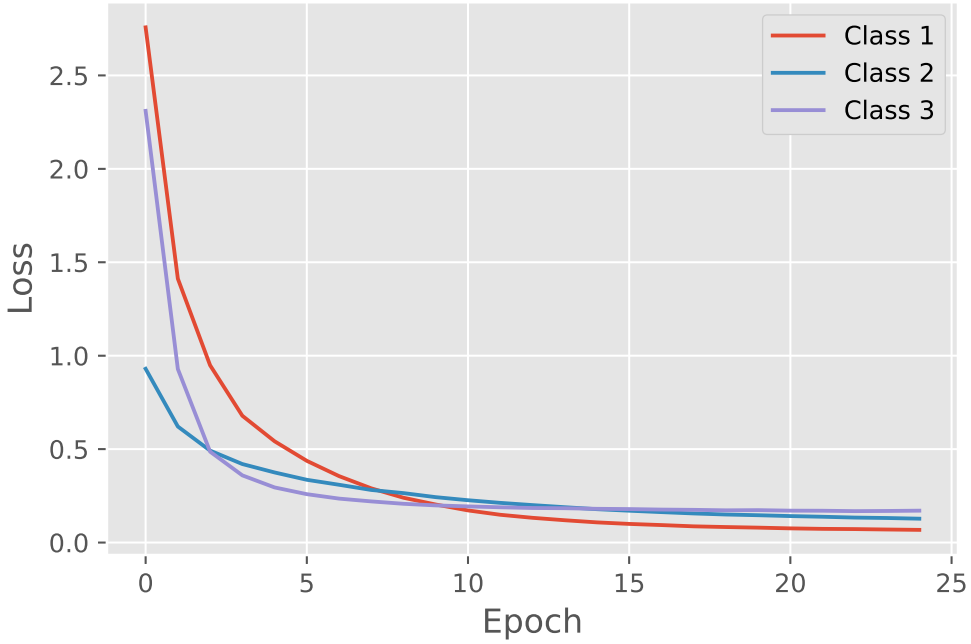


Figure 11: Multi-Class Loss Plot

As can be seen in Figure 12, the quantum neural network architectures converge extremely quickly, and with a relatively high accuracy of 94%. Similarly, parameterized classical neural networks perform below that of their quantum counterparts. We analysed and compared our network to classical neural network structures and show that the quantum neural network learns at a significantly faster rate than classical networks. We test and compare our network design of 12 parameters compared to a varying range of classical neural networks parameter counts, between 12 and 112. This is illustrated in Figure 13, where multiple classical neural networks of varying parameter counts were compared with the our architecture. In Fig. 13, kP indicates the number of parameters in said network.
Our network was able to learn to classify the three classes much quicker than any of the other neural network designs, and attained higher accuracy's at at almost all epoch iteration.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

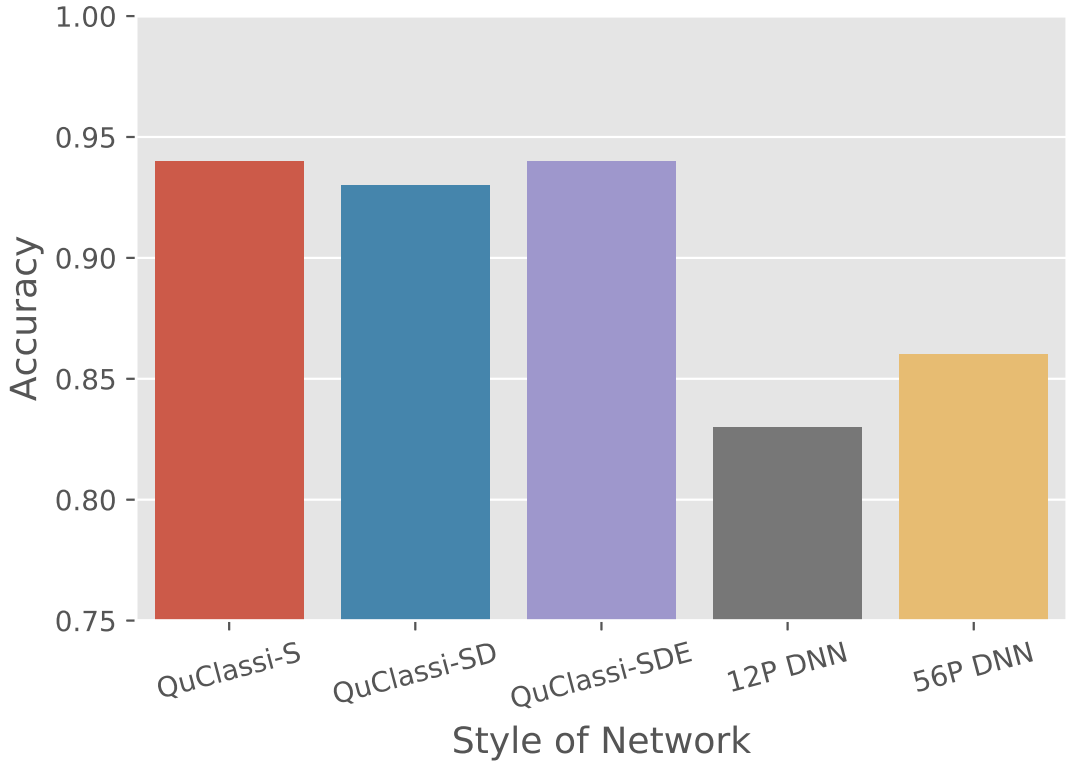


Figure 12: Iris Accuracy (3 classes)

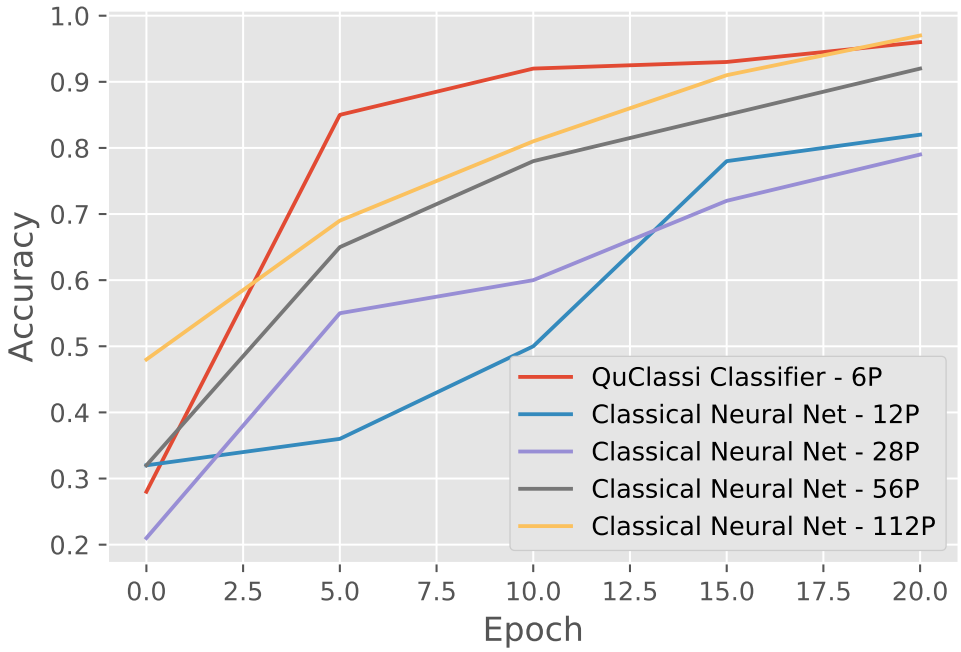
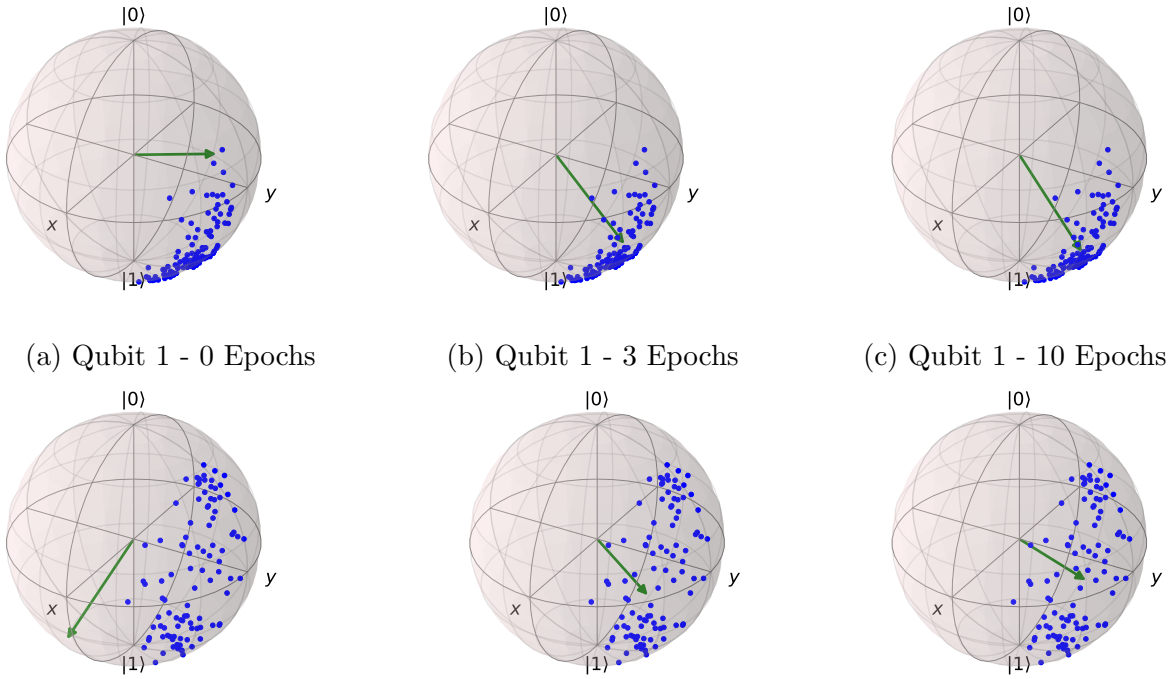


Figure 13: Iris Accuracy of Multiple Parameter Settings

5.1.3 Quantum Classification of the MNIST data set

Although the Iris data set was able to provide proof of concept, and the potency of this architecture, a more challenging task is classifying the MNIST dataset. The MNIST dataset is a large data set of hand written digits of resolution 28×28 - 784 dimensions. Unfortunately, it is impossible to simulate this number of

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1



(a) Qubit 1 - 0 Epochs (b) Qubit 1 - 3 Epochs (c) Qubit 1 - 10 Epochs

(d) Qubit 2 - 0 Epochs (e) Qubit 2 - 3 Epochs (f) Qubit 2 - 10 Epochs

Figure 14: Learning To Identify 0

qubits, and almost all images are too complex to simulate on quantum computers, and hence we need to scale this dimensionality down. We make use of PCA, as this is an efficient down scaling algorithm that has the potential to work on a quantum computer [35]. We downscale to 4 dimensions, leaving us with an explained variance of 24%. It is worth noting that in our comparisons to other current leading quantum classification methods, they might not use PCA, which could be thought to give us an advantage.However, with only an explained variance of 24% being used, we are not attaining an overwhelming amount of information from PCA that would make the process seem unfair. Our design allowed us to perform 10 class classification on the MNIST data set, unlike many other papers which are binary classification problems such as "Is it a 3 or a 6", or "3 or 8". We can visualize the training process on learning to identify 0 by looking at our qubit system, an initial qubit state with an unlearned state and how the state evolves through the epochs. As shown in Fig. 14, we see the evolution of the identifying state through epochs. Green arrows indicate the learned state, and blue points indicate data points. We see initial identifying states to be random, but learn very quickly. The largest difference is in the first few epochs of learning, and as epoch numbers increase the differentiation will assist in finding the absolute best position for the identifying state. In classifying a data point, we would take all possible trained identifying states and attain the probability that said data point is of each class. These probabilities are SoftMaxed and we classify the data point as the most probable identifying state.

5.1.4 Multi Class Classification Results

While training our system to classify the 10 class MNIST data set, we attain the accuracy as shown in Fig. 15 when using a learning rate of $\alpha = 0.01$ and $epochs = 25$:
Where QNN indicates Quantum Neural Network, S is a single unitary, D is a dual qubit unitary, E is an entanglement layer, kP indicates parameter count, and DNN is Deep Neural Network.
Through using more complex quantum layers, the quantum network was able to learn slightly better than the previous less complex network. As can be seen by the 5% realized accuracy gain in Figure 15 comparing the Single quantum layer against the entangled single and dual layered quantum network. When we compare our models to classical networks, the classical neural network when trained on the same epochs and data

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

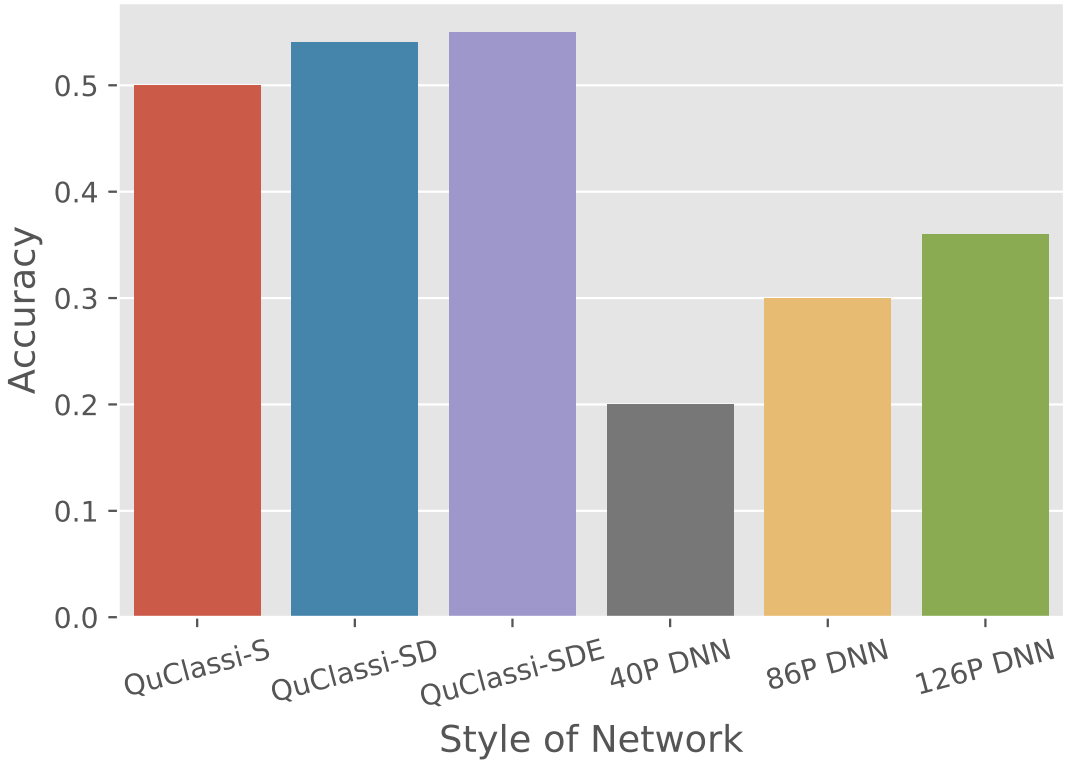


Figure 15: MNIST Accuracy (10 Classes)

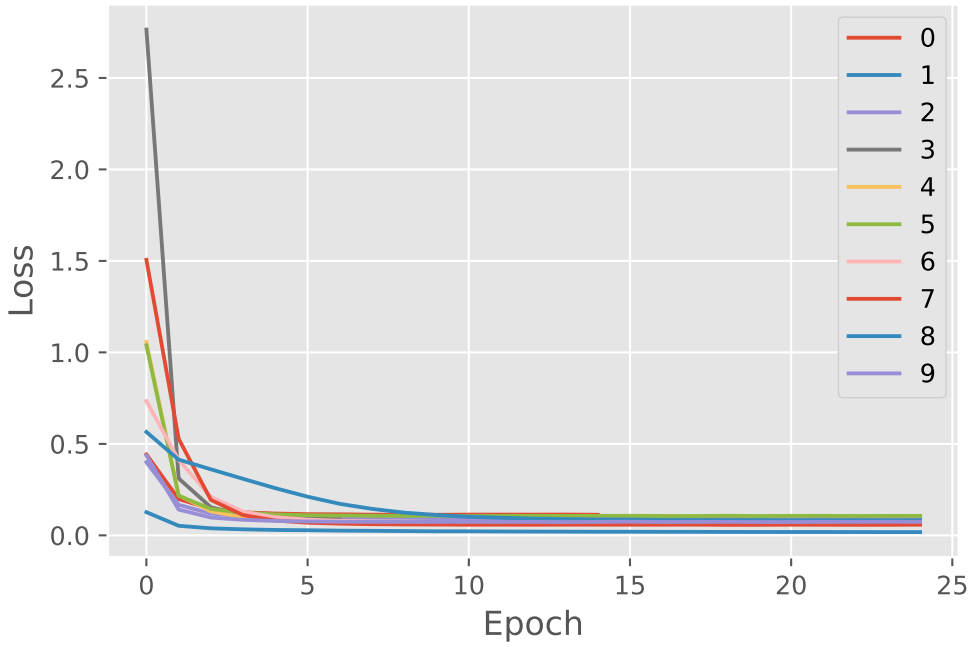


Figure 16: Loss vs Epoch for different classes in MNIST.

set is significantly worse than our quantum neural network. Looking at the loss plot of our quantum neural network we see that certain classes converged less successfully with the loss of the 1 class converging the lowest, versus 0 class converging with the greatest loss.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

5.1.5 Binary Classification of the MNIST Dataset

Many quantum classification papers aim to perform binary or three class classification. Therefore, to evaluate our architecture, it is of value to compare how our system performs compared to other papers. Comparing our results to other leading quantum machine learning as well as classical machine learning models, we need to ensure the problems are comparable. The results discussed tackle 3 and 10 multi-class classification using deep quantum networks. The current state of QML is very under researched, leading to few comparable results that are readily available. The comparable results are different such that making comparisons does not necessarily make complete sense, however for the sake of completeness and a gauge of the quality of our system we include comparisons with other systems with descriptions on how they differ. One paper discusses using kNN and PCA to classify MNIST, and with comparable PCA dimensions attains an accuracy of around 35% [32]. Comparing our systems performance to Tensorflow Quantum (TF-Q), we are able to attain 12.51% better results when distinguishing 0's and 1's, and 11.7% better results when distinguishing between numbers 3 and 8. This is visualized in Fig.17.

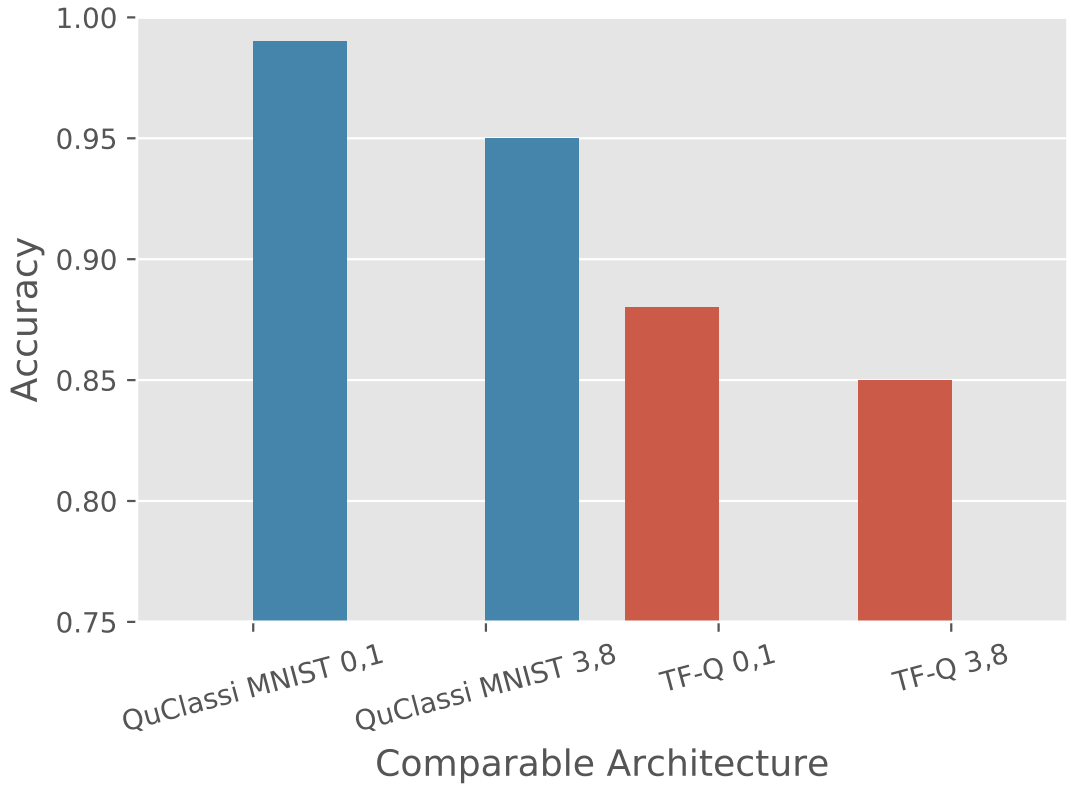


Figure 17: Comparing Results with TF-Q

5.1.6 Comparison of Single vs Dual Dimensional Qubit Encoding

For all results attained, the Quantum Networks had each qubit within the network encoded with 2 dimensions of data. Therefore, in the case where we performed PCA on the MNIST dataset down to 4 dimensions, we only required 2 qubits per MNIST data point. As mentioned prior, it is of significant worth in evaluating to what extent this affects the results. We compare our results by retraining the same network, except this network trained and tested using a 1-to-1 qubit-to-dimension representation. These results are outlined in Table 1 Representing 2 dimensions on a qubit lead to a 32% increase in performance. It is of significant use that the system performed better by using less resources. We hypothesize that this can be attributed to the difference in state space. N qubits can be represented by 2^n probability amplitudes. This state space is what we are trying to learn when we train a quantum state. Therefore, when we halve the number of qubits, we

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

Qubit Design	Qubits	Parameters	Accuracy
Encode in Z	2	4	38%
Encode in Z & Y	4	4	50%

Table 1: 1 vs 2 Qubit Data Encoding

reduce the solution space by $2^{(n/2)}$ values. This method would be a lot more problematic if we were actually measuring our qubits, as each qubits data is dependant on each other, however in our case the qubits surface is populated with data replicating the data distribution.

5.1.7 IBM-Q Evaluation

As a proof of concept, we evaluate our result on actual quantum computers. Our experiments were ran with a total of 5 qubits on different IBM-Q machines. Despite the fact IBM-Q quantum computer's communication channel has an extra overhead and since the users share a public interface channel leading to large backlogs and queues, we attained a result of IBM-Q London site with an accuracy of 96% percent on IRIS data set. The IRIS data set was used for this proof of concept while considering the limited number of physical qubits and communication overhead between end users and the quantum machine. Similar to our experiment on simulator we followed the same principle and architecture design we explained in QuClassi design and we trained each epoch through IRIS data set with 8000 shots (number of repetitions of each circuit) to calculate the loss of the circuit. Based on our observation and previous research in this area [54, 22] the integrity of physical qubits and `T1`, `T2` errors of IBM-Q machines could vary, however, our design managed to attain an optimal solution after few iterations, comparable to the simulator results we attained. Since Machine learning applications, unlike chemistry or other noise-sensitive applications, can tolerate more noise within the system, running experiment on actual quantum machines generated stable results and accuracy similar to the simulators. As seen in Fig 18, the loss of the quantum circuit converges similarly on a real quantum circuit similarly to that of a simulator.

5.2 Quantum GAN (QuGAN) implementation

We implement our QuGAN using the same software packages and simulators as our Deep Learning model, making use of Qiskit as our quantum simulator. For a comprehensive evaluation, we compare QuGAN with the following 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.
- **Qi-GAN**: a quantum GAN [57] that is designed to learn random distributions. It is implemented with Qiskit.
- **TFQ-GAN**: a quantum GAN [9] that utilizes the TensorFlow quantum architecture proposed in [19].

Moreover, to evaluate the models quality, we use Hellinger Distance [44] to compare the generated distribution and original data sets. Hellinger distance is a popular metric to quantify the similarity between two probability distributions.

5.2.1

Learning Bivariate Normal Distributions GANs can be thought to attempt to learn the underlying distribution of data sets and produce never-before-seen outputs. As a low-level analysis of our GAN, we implement the learning process on a bivariate normal distribution. We generate 2D data based on Equation 37, with the 1st dimension generated with parameters $\sigma = 0.10$ and $\mu = 0.65$. The 2nd dimension is generated with

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

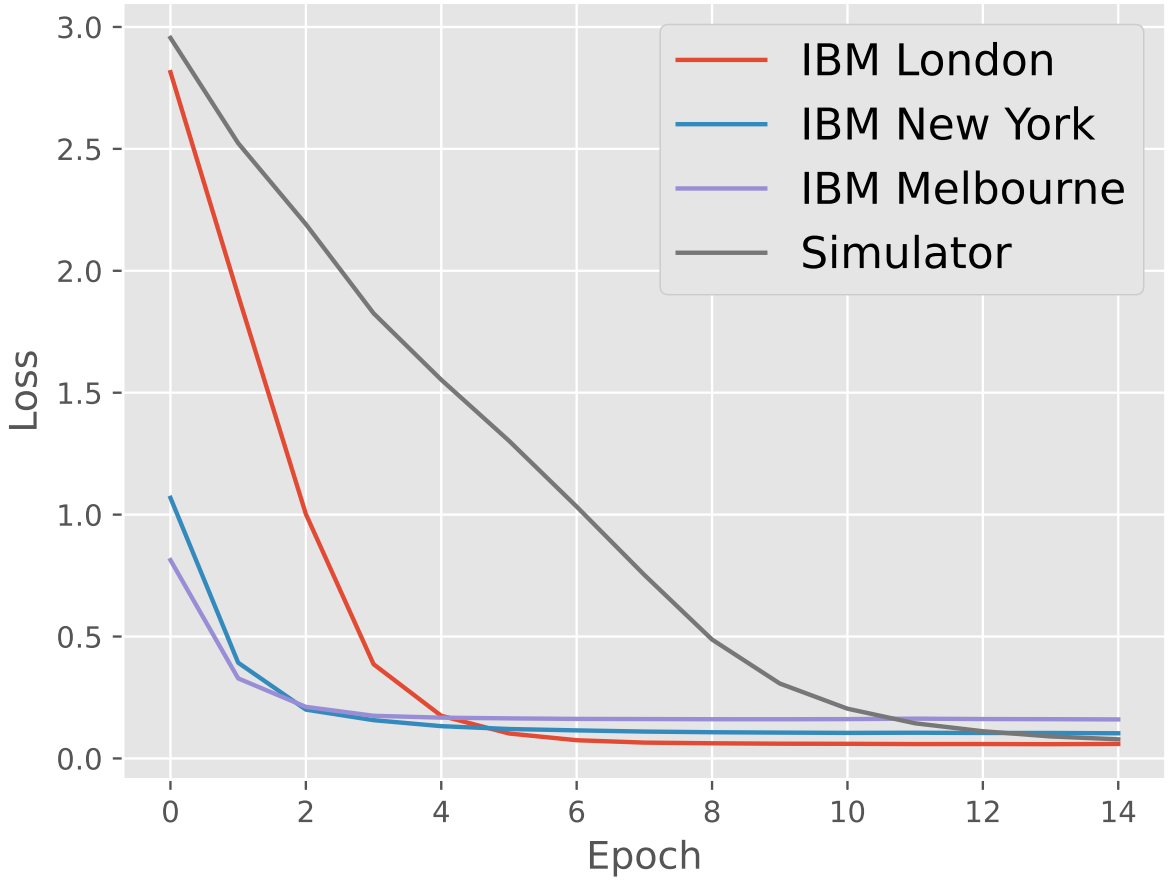


Figure 18: Real Quantum Computer Training

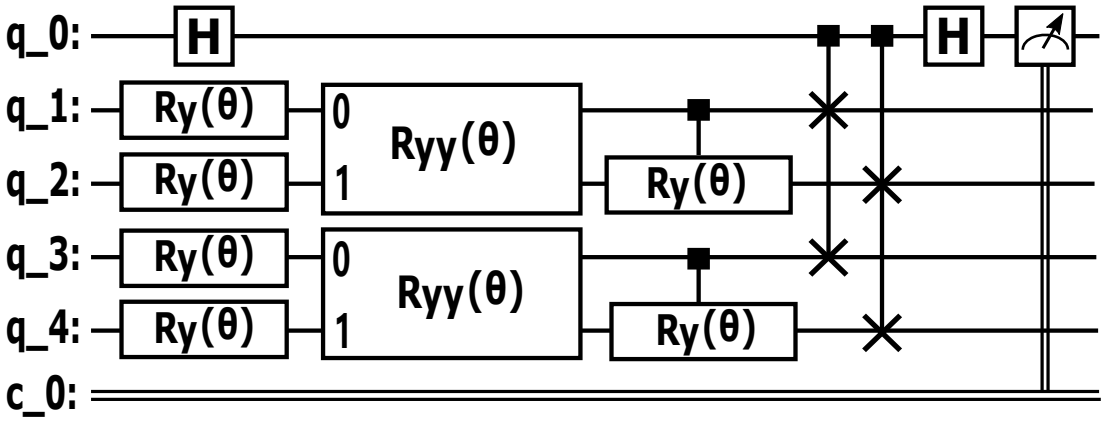


Figure 19: Bivariate QuGAN Circuit

parameters $\sigma = 0.05$ and $\mu = 0.45$. All dimensions are capped at a value of 1, as a qubit's expectation cannot surpass 1.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{(x-\mu)^2}{2\sigma^2}} \tag{37}$$

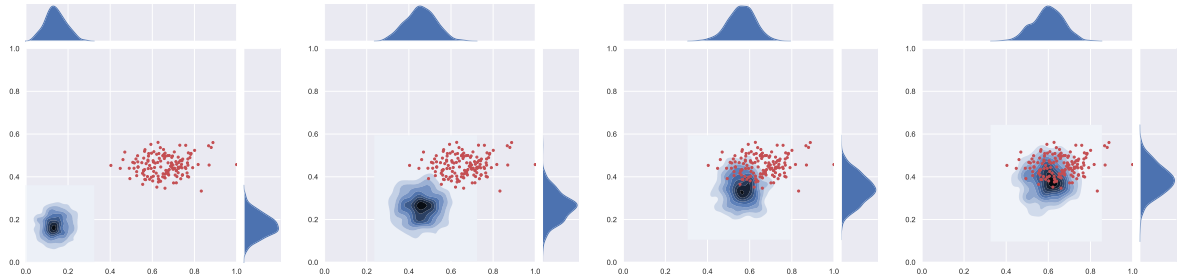
The Generator and Discriminator architecture makes use of one Single Qubit Unitary per qubit (Figure 2-(A)), one Dual Qubit Unitary (Figure 2-(B)) per qubit pair, and one Entanglement Unitary (Figure 2-(C)). The network is illustrated in Figure 4, with the Discriminator and Generator having 4 parameters each.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

Architecture	Parameter Count	Hellinger Distance
QuGAN	4	0.32
C-GAN	4	0.60
C-GAN	32	0.48
C-GAN	64	0.45
C-GAN	268	0.34

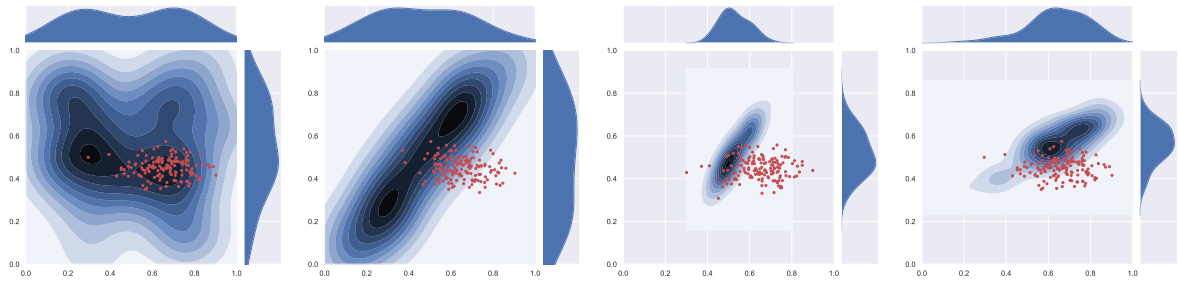
Table 2: Different GAN Architectures with 50 Epochs

Qubits 1 and 2 are the Discriminator, and qubits 3 and 4 are reserved for the Generator parameters or data loading channels. Qubit 0 is the extra qubit for the SWAP test.



(a) 1^{st} epoch (b) 10^{th} epochs (c) 25^{th} epochs (d) 50^{th} epochs

Figure 20: Generator (QuGAN) Learned Bivariate Distribution Progression



(a) 4 Parameters, 50 Epochs (b) 4 Parameters, 500 Epochs (c) 64 Parameters, 50 Epochs (d) 268 Parameters, 50 Epoch

Figure 21: Generators (C-GANs) Bivariate Distribution Learning

In Figure 20 we observe the learning process of our algorithm. The red dots indicate the data being fed to the system, and the blue Gaussian distributions indicate the generators' output. As seen in the progressions, our algorithm is able to accurately learn the bivariate Gaussian distribution that was passed to it. At epochs 1 through 10, the majority of the learning has been completed, with the finer alignment of distributions happening in the later epochs. This learning process can also be seen in Figure 22 where we illustrate the Hellinger distance between our GAN and the true bivariate distribution. This graph illustrates the stability of our GAN, with no large swings in Hellinger distance between epochs, and a stable improvement in distance per epoch. This stability is significantly better than many other current quantum GANs illustrate. On implementation of other quantum deep learning techniques, we observe that our method converges 94.4% better than the Qi-GAN, and 125.0% better than TFQ-GAN. One of the benefits of these quantum generators, is that within the Gaussian area, all possible outputs will be produced as well as new never-before-seen samples. This result is very attractive as a generator, as it tackles a problem discussed in past work of loading large data sets to distributions, as well as mode collapse leading to constantly producing the same

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

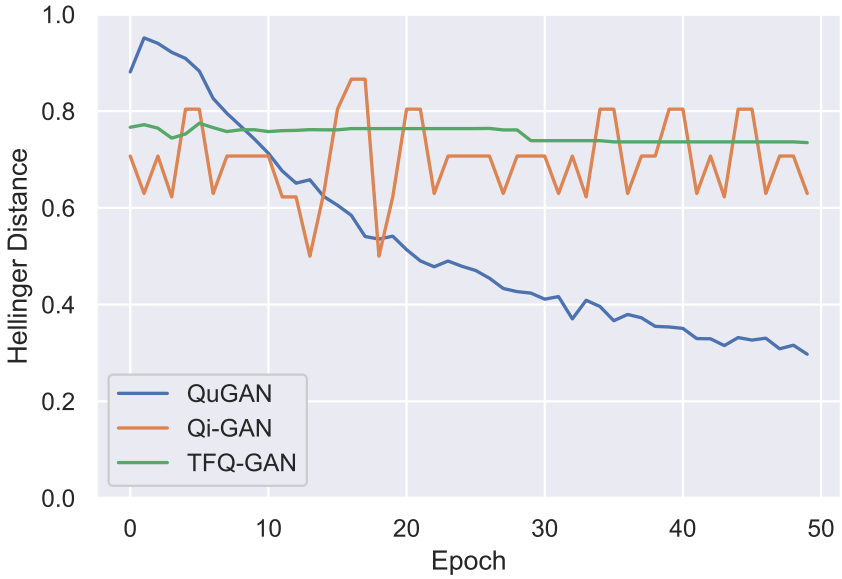


Figure 22: Hellinger distance (Bivariate distribution)

output, and is an improvement in the current quantum GAN architectures. Comparing with other quantum GANs, our architecture demonstrates an improvement in stability and learning ability, as well as extends the applications into newer domains. In comparison to Qi-GAN, we illustrate an extension of their work with higher dimensional data, as well as an improved learning process that requires fewer epochs to learn. Comparing with TFQ-GAN, we show a significant improvement in stability and learning, and the ability to learn distributions substantially faster.

In comparing our architecture to a classical GAN, parameterized by the same parameter counts, we visualize the result in Figure 21. As seen, the classical GAN with a parameter count of 4 in the Generator and 4 in the Discriminator performs significantly worse than our QuGAN, attaining a 87.5% further Hellinger distance when compared with our QuGAN. Furthermore, we confirm this is not due to under-training the network, and illustrate in Figure 21b that even with 500 epochs the network does not learn the distribution well. We increase the parameter count up until 268, where the Hellinger distance begins to approach similar values to our GAN. These results are outlined in Table 2. We illustrate that to attain similar performance in a classical GAN, we need to use 268 parameters versus the 4 in our QuGAN - a 98.5% reduction in parameters.

5.2.2 MNIST Dataset Processed by t-SNE

In addition to experimenting with generated bivariate data above, we implement our architecture on the MNIST data set of handwritten digits. Observing performance on this data demonstrates our GAN's ability and function in real-world applications. In these experiments, we apply the same learning process we use for learning from our generated bivariate normal distributions, defined under Algorithm 1.

Due to the exponential nature of simulating quantum states, it is impossible to fully simulate almost any images on quantum computers. Therefore, we downsample the highly dimensional MNIST data set with the t-SNE (T-distributed Stochastic Neighbor Embedding [37]) technique to project the original 784-dimensional MNIST data set into 2D data, still preserving enough information to examine performance in a real-world context. Figure 23 visualizes this MNIST data after reducing dimensionality with t-SNE. Our QuGAN architecture is able to learn on this data set by testing its performance on a sub-sampled cluster of two of its classes {5,3}, as well as testing it solely on learning class {3}. As we see in Figure 24, our QuGAN learns the majority of the MNIST data cluster distribution by the 25^{th} epoch, and improves on the 50^{th} indicated by the distance between the red plotted data points. The heat map shown in blue again represents the Gaussian distributions of data outputted at the indicated epoch. Within the learning results on the classes {3,8}, visualized in Figure 24-C&D, we can see how the learning covers areas in both the the 3 domain and

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

the 5 domain, thereby producing samples from both sub-spaces. This convergence can also be visualized in Figure 25, where we see the Hellinger distance between the learned distribution and the actual distribution is reducing. Furthermore, in Figure 25, we confirm the strength of our QuGAN against other quantum based GAN solutions. We implement their solutions on the same data set, and observe that we are able to attain Hellinger distances 74.1% better than TFQ-GAN, and 66.2% better distances than Qi-GAN at the 25^{th} epoch.

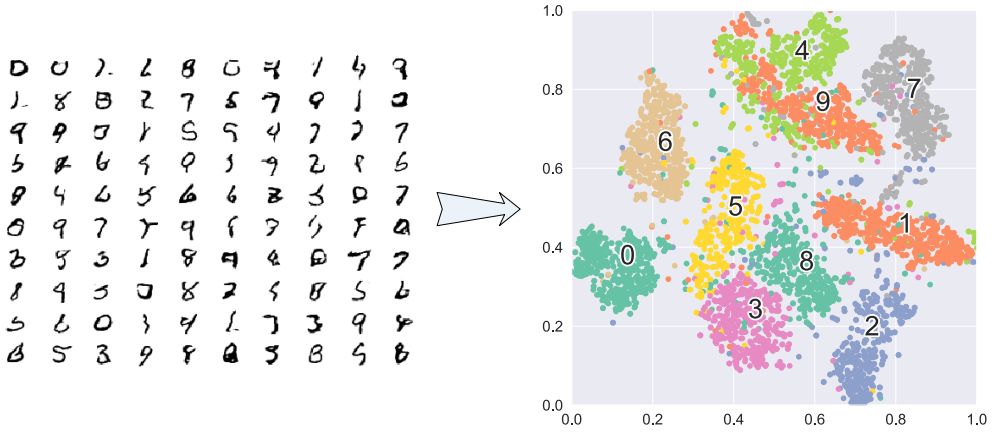
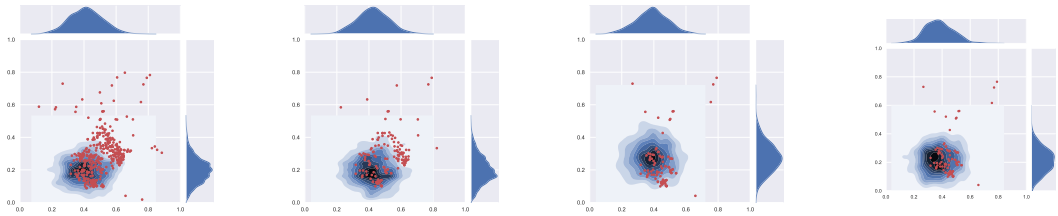


Figure 23: MNIST with t-SNE dimensionality reduction



(a) 25 Epochs Learning 3 & 8 (b) 50 Epochs Learning 3 & 8 (c) 25 Epochs learning 3 (d) 50 Epochs learning 3

Figure 24: Generator Learned Distribution for MNIST

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 March 2021 doi:10.20944/preprints202103.0583.v1

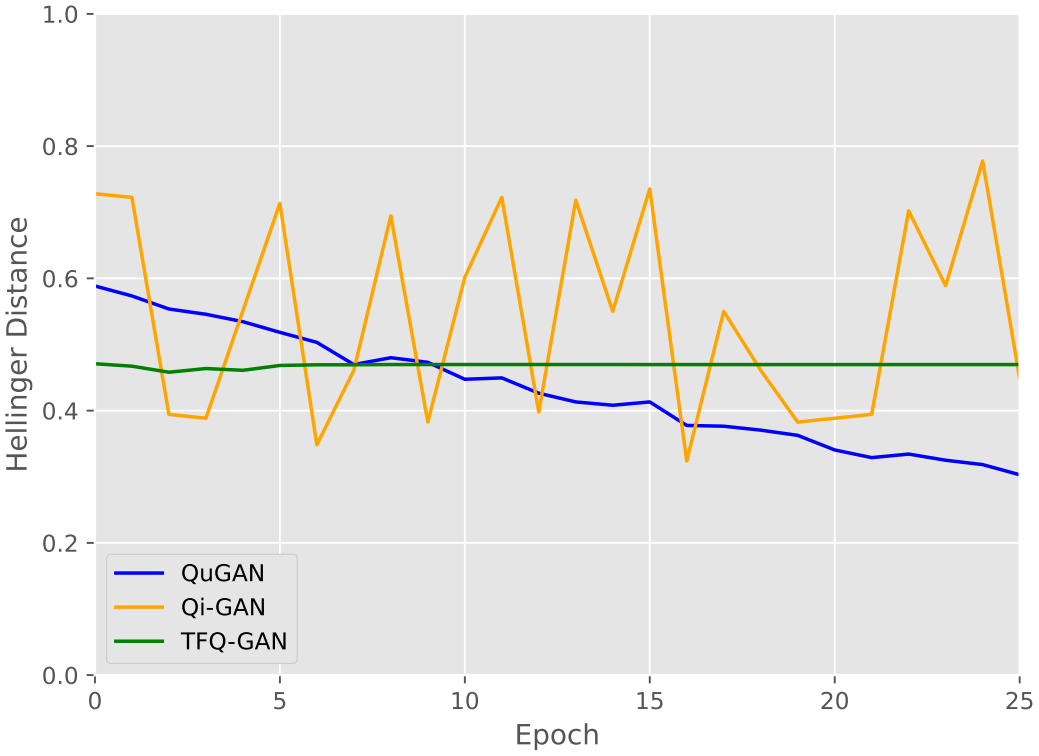


Figure 25: Hellinger distance (MNIST)

 doi:10.20944/preprints202103.0583.v1

6 Conclusions & Future Work

In this research, we propose QuClassi and QuGAN, two novel quantum-classical architectures for multi-class classification and generative modelling.

With respect to QuClassi; the current accuracy attained by QuClassi performing binary classification on the MNIST data set is competitive, beating out Tensorflow Qauntum by up to 13%. Furthermore, to the best of our knowledge, QuClassi is the first solution that tackles a 10-class classification problem in the quantum setting. Additionally, comparing QuClassi with similarly parameterized classical neural networks, QuClassi outperforms them by learning significantly faster, and requiring drastically significantly less parameters by up to 26%. When developing these quantum neural networks for multi classification, we observe a particularly interesting phenomena where encoding our data onto lower qubit counts out performed the higher qubit counterpart. This observation directly tackles one of the limiting factors in quantum computing, that is the number of qubits involved.

Our work provides a general step forward in the quantum deep learning domain. There is, however, still significant progress to be made. Performing the 10-class classification of MNIST lead to a 50% accuracy, which is relatively poor when compared to its classical counterparts. Although using much more parameters, classical counterparts are able to reach an accuracy of near 100%. Therefore, future work focusing on improving the multi-class classification should aim to further improve the accuracy. Moreover, understanding the lower qubit representation of quantum data should be investigated, and its implications within the quantum based learning field. Furthermore, on our proposed QuGAN, a Quantum Generative Adversarial Network. QuGAN utilizes quantum states to encode classic data and with SWAP Test on qubits, it develops quantum based loss functions for the QuG and QuD. The proposed model is evaluated with a bivariate normal distribution as well as the t-SNE downsampled MNIST data set. We conduct extensive experiments to evaluate QuGAN and compare it with Tensorflow based classical GANs with different settings, a Qiskit based quantum GAN and a Tensorflow quantum based GAN in the recent literature. The results demonstrate that QuGAN is able to achieve similar performance at meanwhile, reduces 98.5% of the parameter set compared with classical GANs. Furthermore, it records a performance boost of up to 125.0% when comparing with other recent quantum based solutions. Due to the current limits on quantum computers, it is infeasible to evaluate existing quantum based GANs with ImageNet [20] or other popular 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.

References

[1] A. Acharya, Y. Hou, Y. Mao, M. Xian, and J. Yuan. Workload-aware task placement in edge-assisted human re-identification. In *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9. IEEE, 2019.

[2] A. Acharya, Y. Hou, Y. Mao, and J. Yuan. Edge-assisted image processing with joint optimization of responding and placement strategy. In *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 1241–1248. IEEE, 2019.

[3] E. Aïmeur, G. Brassard, and S. Gambs. Quantum clustering algorithms. In *Proceedings of the 24th international conference on machine learning*, pages 1–8, 2007.

[4] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[5] F. Arute and et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574:505–510, 2019.

[6] K. Beer, D. Bondarenko, T. Farrelly, T. J. Osborne, R. Salzmann, D. Scheiermann, and R. Wolf. Training deep quantum neural networks. *Nature communications*, 11(1):1–6, 2020.

[7] S. Bharati, P. Podder, and M. R. H. Mondal. Hybrid deep learning for detecting lung diseases from x-ray images. *Informatics in Medicine Unlocked*, 20:100391, 2020.

[8] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[9] 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.

[10] R. V. Casaña-Eslava, P. J. Lisboa, S. Ortega-Martorell, I. H. Jarman, and J. D. Martín-Guerrero. Probabilistic quantum clustering. *Knowledge-Based Systems*, page 105567, 2020.

[11] D. Chen, Y. Xu, B. Baheri, C. Bi, Y. Mao, Q. Quan, and S. Xu. Quantum-inspired classical algorithm for principal component regression. *arXiv preprint arXiv:2010.08626*, 2020.

[12] D. Chen, Y. Xu, B. Baheri, S. A. Stein, C. Bi, Y. Mao, Q. Quan, and S. Xu. Quantum-inspired classical algorithm for slow feature analysis. *arXiv preprint arXiv:2012.00824*, 2020.

[13] S. Y.-C. Chen, T.-C. Wei, C. Zhang, H. Yu, and S. Yoo. Quantum convolutional neural networks for high energy physics data analysis. *arXiv preprint arXiv:2012.12177*, 2020.

[14] S. Y.-C. Chen, T.-C. Wei, C. Zhang, H. Yu, and S. Yoo. Hybrid quantum-classical graph convolutional network. *arXiv preprint arXiv:2101.06189*, 2021.

[15] 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.

[16] X. Chen, L. Cheng, C. Liu, Q. Liu, J. Liu, Y. Mao, and J. Murphy. A woa-based optimization approach for task scheduling in cloud computing systems. *IEEE Systems Journal*, 14(3):3117–3128, 2020.

[17] I. Cong, S. Choi, and M. D. Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019.

[18] J. A. Cortese and T. M. Braje. Loading classical data into a quantum computer. *arXiv preprint arXiv:1803.01958*, 2018.

[19] P.-L. Dallaire-Demers and N. Killoran. Quantum generative adversarial networks. *Physical Review A*, 98(1):012324, 2018.

[20] 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*, pages 248–255. Ieee, 2009.

[21] 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.

[22] G. W. Dueck, A. Pathak, M. M. Rahman, A. Shukla, and A. Banerjee. Optimization of circuits for ibm's five-qubit quantum computers. In *2018 21st Euromicro Conference on Digital System Design (DSD)*, pages 680–684, 2018.

[23] E. Farhi and H. Neven. Classification with quantum neural networks on near term processors. *arXiv preprint arXiv:1802.06002*, 2018.

[24] Y. Fu, S. Zhang, J. Terrero, Y. Mao, G. Liu, S. Li, and D. Tao. Progress-based container scheduling for short-lived applications in a kubernetes cluster. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 278–287. IEEE, 2019.

[25] S. Garg and G. Ramakrishnan. Advances in quantum deep learning: An overview. *arXiv preprint arXiv:2005.04316*, 2020.

[26] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[27] 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*, pages 2672–2680, 2014.

[28] H. H. Harvey, Y. Mao, Y. Hou, and B. Sheng. Edos: Edge assisted offloading system for mobile devices. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9. IEEE, 2017.

[29] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[30] 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*, 5(1):eaav2761, 2019.

[31] 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.

[32] I. Kerenidis and A. Luongo. Quantum classification of the mnist dataset via slow feature analysis. *arXiv preprint arXiv:1805.08837*, 2018.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[34] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[35] S. Lloyd, M. Mohseni, and P. Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631–633, 2014.

[36] S. Lloyd and C. Weedbrook. Quantum generative adversarial learning. *Physical review letters*, 121(4):040502, 2018.

[37] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[38] Y. Mao, Y. Fu, S. Gu, S. Vhaduri, L. Cheng, and Q. Liu. Resource management schemes for cloud-native platforms with computing containers of docker and kubernetes. *arXiv preprint arXiv:2010.10350*, 2020.

[39] Y. Mao, Y. Fu, W. Zheng, L. Cheng, Q. Liu, and D. Tao. Speculative container scheduling for deep learning applications in a kubernetes cluster. *arXiv preprint arXiv:2010.11307*, 2020.

[40] Y. Mao, J. Oak, A. Pompili, D. Beer, T. Han, and P. Hu. Draps: Dynamic and resource-aware placement scheme for docker containers in a heterogeneous cluster. In *2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8. IEEE, 2017.

[41] Y. Mao, J. Wang, J. P. Cohen, and B. Sheng. Pasa: Passive broadcast for smartphone ad-hoc networks. In *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–8. IEEE, 2014.

[42] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3061–3070, 2015.

[43] M. Mustafa, D. Bard, Z. Bhimji, R. Al-Rfou, and J. M. Kratochvil. Cosmogan: Creating high-fidelity weak lensing convergence maps using generative adversarial network. *Computational Astrophysics and Cosmology*, 2019.

[44] M. S. Nikulin. Hellinger distance. *Encyclopedia of mathematics*, 78, 2001.

[45] P. Rebentrost, M. Mohseni, and S. Lloyd. Quantum support vector machine for big data classification. *Physical review letters*, 113(13):130503, 2014.

[46] M. Schuld, M. Fingerhuth, and F. Petruccione. Implementing a distance-based classifier with a quantum interference circuit. *EPL (Europhysics Letters)*, 119(6):60002, 2017.

[47] S. A. Stein, B. Baheri, R. M. Tischio, Y. Chen, Y. Mao, Q. Guan, A. Li, and B. Fang. A hybrid system for learning classical data in quantum states. *arXiv preprint arXiv:2012.00256*, 2020.

[48] S. A. Stein, B. Baheri, R. M. Tischio, Y. Mao, Q. Guan, A. Li, B. Fang, and S. Xu. Qugan: A generative adversarial network through quantum states. *arXiv preprint arXiv:2010.09036*, 2020.

[49] J. Stokes, J. Izaac, N. Killoran, and G. Carleo. Quantum natural gradient. *Quantum*, 4:269, 2020.

[50] 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.

[51] J. Wang, Y. Yao, Y. Mao, B. Sheng, and N. Mi. Fresh: Fair and efficient slot configuration and scheduling for hadoop clusters. In *2014 IEEE 7th International Conference on Cloud Computing*, pages 761–768. IEEE, 2014.

[52] Z. Wang, Q. She, and T. E. Ward. Generative adversarial networks in computer vision: A survey and taxonomy. *arXiv preprint arXiv:1906.01529*, 2019.

[53] J. M. Wozniak, R. Jain, P. Balaprakash, J. Ozik, N. T. Collier, J. Bauer, F. Xia, T. Brettin, R. Stevens, J. Mohd-Yusof, C. G. Cardona, B. V. Essen, and M. Baughman. Candle/supervisor: a workflow framework for machine learning applied to cancer research. *BMC Bioinformatics*, 19(18):491, Dec 2018.

 doi:10.20944/preprints202103.0583.v1

[54] Y. Zhang, H. Deng, Q. Li, H. Song, and L. Nie. Optimizing quantum programs against decoherence: Delaying qubits into quantum superposition. In *2019 International Symposium on Theoretical Aspects of Software Engineering (TASE)*, pages 184–191, 2019.

[55] W. Zheng, Y. Song, Z. Guo, Y. Cui, S. Gu, Y. Mao, and L. Cheng. Target-based resource allocation for deep learning applications in a multi-tenancy system. In *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–7. IEEE, 2019.

[56] W. Zheng, M. Tynes, H. Gorelick, Y. Mao, L. Cheng, and Y. Hou. Flowcon: Elastic flow configuration for containerized deep learning applications. In *Proceedings of the 48th International Conference on Parallel Processing*, pages 1–10, 2019.

[57] C. Zoufal, A. Lucchi, and S. Woerner. Quantum generative adversarial networks for learning and loading random distributions. *npj Quantum Information*, 5(1):1–9, 2019.