Qauntum Neural Networks on Photonic Quantum Computers

Classical quantum hybrid models

Sophie Choe[1]

[1] *Electrical and Computer Engineersing, Portland State University, Portland, OR*

Abstract

The current landscape of quantum computing is defined as Near Intermediate-Scale Quantum or near-term quantum, where functioning quantum processing units (QPUs) are available on cloud, but not yet fully fault-tolerant. Nonetheless, the availability of working QPUs offers the opportunity to run and test quantum algorithms, which were only theoretical previously. The two types of QPUs on cloud are superconducting chips based on the discrete variable model and linear optics (photonics) chips based on the continuous variable (CV) model. Along with quantum chemistry, Gaussian boson sampling, and graph optimization, quantum machine learning is an active area of research. Quantum machine learning models can be developed using variational circuits, wherein parameterized quantum gates execute machine learning algorithms and classical circuits evaluate the computational results for parameter optimization.

Specifically, classical neural networks can be directly converted into quantum on photonic quantum computers based on the CV model, due to linear and nonlinear quantum gates available in the model. Here, we examine the classical and quantum hybrid quantum neural networks proposed in "Continuous variable quantum neural networks" and expand on its binary classifier model to a multi-classifier model on MNIST dataset. The model achieves 97.23% training accuracy on 600 samples.

Keywords: quantum computing, quantum machine learning, quantum neural networks, continuous variable quantum computing, photonic quantum computing, classical quantum hybrid model, quantum MNIST classification

Dated: January 2022

1. INTRODUCTION

We are in an era where quantum computing has emerged from the domain of pure research into a viable commercial paradigm of computing. There are several quantum processing units (QPUs) available on cloud offered by companies like IBM, Google, and Xanadu. Unlike the initial assumption that quantum computers would replace classical computers, they are being integrated into the current classical computing infrastructure, performing task-specific computations and processing. In this way, QPUs can be viewed as special purpose processing units, much like graphics processing units (GPUs).

The basic idea of quantum computing is an embedding of classical data into a higher dimensional quantum information space - a projective Hilbert space – wherein information processing is performed by inducing change of quantum states [24]. This idea of "extended information space", along with some of the properties inherent in quantum mechanical systems, not available in classical computing, is where the promise of quantum supremacy lies.

From the physical implementation perspective, quantum computers are divided into two types: superconducting quantum computers and linear optics (photonics). Superconducting quantum computers utilize the particle-like properties of quantum mechanics and use individual particles as information carriers [46]. Since addition of information carrier particles for more computing power is, according to the mathematical formalism of quantum mechanics, discrete this model is theoretically based on the discrete variable quantum computing model [42]. The discrete variable model is interchangeably called the quantum bit (qubit) model as it is a quantized version of the classical bit model. IBM and Google QPUs are implementations of the superconducting model. Linear optical quantum computers utilize the wave-like properties of quantum mechanics and use channels of photons, called quantum modes (qumodes), as information carriers. Increasing the number of qumodes is adding additional continuous waves, hence this model is theoretically based on the continuous variable (CV) quantum computing model [46]. Xanadu's QPU is an implementation of the linear optics model.

Currently available QPUs are not completely fault-tolerant however, therefore are limited compared to fully fault-tolerant universal quantum computers that can be realized. The current state of quantum computing is called Near Intermediate-Scale Quantum (NISQ) and the QPUs, near-term devices. While it may be a while until we reach the fully fault-tolerant universal quantum computing era, the availability of near-term devices allows researchers to test quantum algorithms on quantum computers and develop more quantum computer specific algorithms. Quantum machine learning is one of the fast-growing areas of research among other areas such as quantum chemistry, Gaussian boson sampling, and graph optimization. Testing classical machine learning algorithms on QPUs or developing new algorithms unique to quantum computing are two main aspects of quantum machine learning.

This paper specifically focuses on quantum neural networks proposed in "Continuous variable quantum neural networks" [16]. Application of these CV-based neural networks on photonic quantum computers produce great results, which cannot be reproduced in superconducting quantum computing. The components of classical neural networks are affine transformations, induced by linear transformation weight matrices and addition of biases, and non-linear transformation activation functions. Quantum neural networks are quantum versions of the same process in a higher dimensional Hilbert space, which inherently offers a higher degree of freedom for encoding and processing information. The quantum gates available in the CV model offer the ability to directly implement the bias addition and non-linear activation components.

Quantum machine learning models can be viewed as the kernel method in which a feature map takes data into a higher dimensional feature space [29]. In this view, subsequent quantum information processing can be interpreted as computations taking place in the feature space. Then the key, which makes a quantum model unique, is the encoding scheme of classical data into quantum states. The CV model, implemented on photonic quantum computers, offers a richer array of quantum gates for data encoding and building circuits using quantum gates, not available in the qubit model [16].

In this paper, I examine the difference between discrete and CV quantum computing models, outline the current landscape of quantum computers, and explore the difference between superconducting chips and linear optical chips. Then I show the importance of classical data encoding into quantum by exploring the kernel method in supervised vector machines. After analysing Google and IBM's different approaches for MNIST classification qubit-based quantum circuits, I outline the classical and quantum hybrid neural networks that I implemented as proposed in "Continuous variable quantum neural networks" [16]. Expanding on the classical-quantum hybrid binary classifier, I created a hybrid multi-classifier distinguishing 10 classes of the MNIST dataset. The trained model achieves 97.23% training accuracy on a dataset of 600 samples.

2. QUANTUM COMPUTING

Quantum computing is an information processing paradigm in which properties of quantum mechanics, not available in classical mechanics, are utilized as components of computation [24]. These properties include superposition, entanglement, and interference. Superposition is a projective complex linear combination of the computational basis elements, which allows for parallel processing. Entanglement is a phenomenon in which the state of one quantum computational channel affects the states of others. Interference is a phenomenon in which the waves of multiple computational channels affect the overall wave of the system by interacting either constructively or destructively. These properties allow computational tasks not possible with classical computers, or conduct computations much faster.

In classical computing, a collection of binary bits, each in state of either 0 or 1, is processed sequentially. In quantum computing, the quantum state of an information carrier is in superposition of computational basis states $|0\rangle$ and $|1\rangle$ simultaneously allowing for parallel processing. The resulting quantum state of information processing is accessed classically, via an operation called measurement [24]. The measurement operation produces only one state of the computational basis as an observable, collapsing the quantum state into a classical state. In order to extract the computational results as accurately as possible, multiple shots of the same computation have to be performed.

Under the mathematical formalism of quantum mechanics, any possible state of a quantum mechanical system is represented by a unit vector in a projective Hilbert space [24]. A Hilbert space is a vector space equipped with the inner product operation. To represent a quantum state vector in a Hilbert space, it is customary to use Bra-ket notation, also known as Dirac notation. Column vectors are denoted by kets $|x\rangle$ and the corresponding complex conjugate transpose row vectors by bras $\langle x|$. Suppose $|x\rangle$ is an $n-$dimensional column vector, $|x\rangle = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix} \in \mathbb{C}^n$. Its conjugate transpose row vector is $\langle x| = |x\rangle^\dagger = \left[x_0^*, x_1^*, \ldots x_{n-1}^*\right]$. The inner prod-

uct of two vectors $|x\rangle, |y\rangle \in \mathbb{C}^n$ is then expressed as

$$\langle x|y\rangle = |x\rangle^\dagger |y\rangle = \begin{bmatrix} x_0^*, x_1^*, \dots x_{n-1}^* \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-1} \end{bmatrix} = \sum_{k=0}^{n-1} x_k^* y_k.$$

The inner product of the vector $|x\rangle$ with itself is the sum of the complex products of its components.

$$\langle x|x\rangle = \sum_{k=0}^{n-1} x_k^* x_k = \sum_{k=0}^{n-1} \| x_k \|^2 = \| |x\rangle \|^2$$

Notice it is the norm squared of the original vector $|x\rangle$. Now we can define the length of $|x\rangle$ to be

$$\| |x\rangle \| = \sqrt{\| |x\rangle \|^2} = \sqrt{\sum_{k=0}^{n-1} \| x_k \|^2} = \sqrt{\langle x|x\rangle}.$$

By the same token, we can view the inner product of a vector with itself to be its norm squared. With the inner product property of a Hilbert space, we can define the distance between two vectors as the length of the difference vector of the two.

$$\begin{aligned} dist(|x\rangle, |y\rangle) &= \| |x - y\rangle \|^2 \\ &= \sqrt{\sum_{k=0}^{n-1} \| x_k - y_k \|^2} \\ &= \sqrt{\langle x - y | x - y \rangle} \end{aligned}$$

We can also define the angle between two vectors, using the formula

$$\begin{aligned} &\langle x|y\rangle = \| |x\rangle \| \| |y\rangle \| \cos\theta \\ &\implies \cos\theta = \frac{\langle x|y\rangle}{\| |x\rangle \| \| |y\rangle \|} \\ &\implies \theta = \arccos\left(\frac{\langle x|y\rangle}{\| |x\rangle \| \| |y\rangle \|} \right) \end{aligned}$$

Clear definition of the distance and angle between two vectors allows generalization of linear algebra and calculus within a quantum state space.

To actualize quantum computing, a physical system needs to be created whose state can be described quantum mechanically. Controlled change of quantum states of the system and readout of the resulting state is regarded as quantum computation [46]. This can be achieved either by implementing a physical system whose quantum mechanical space is described discretely or by a system whose space is continuous [42]. The discrete variable model is realized using the superconducting model and the CV model is realized using the photonics model. The notion of discrete variable comes from the discreteness of the underlying quantum computational state space. The dimension of the computational state space starts from 2, and increases discretely by the factor of 2. The quantum state space of discrete variable quantum computing is finite and increases discretely as computational basis states are added, hence is called discrete. The quantum state space of CV quantum computing is infinite, representing infinite degrees of freedom, hence is called "continuous" variable [16].

2.1 Discrete Variable Model

The discrete variable model works with particle-like properties of nature. This model can be viewed as a quantized version of classical bit computing. The quantum state of computational space is called quantum bit (qubit). The classical states 0 and 1 are embedded into a two-dimensional projective Hilbert space as the computational basis states. Using Dirac notation, express 0 as the quantum state $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and 1 as $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. Any quantum state in this quantum state space is expressed as a projective complex linear combination of $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle = \alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}$$

where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$. This linear combination is called superposition. Here α and β are called probability amplitudes.

The concept of "projective" is based on the notion of global phase in quantum mechanics, where two quantum states are considered to be equivalent when they differ by a non-zero complex factor.

$$|\psi\rangle \equiv \alpha |0\rangle + \beta |1\rangle = \gamma(\alpha |0\rangle + \beta |1\rangle), \gamma \in \mathbb{C}^*.$$

Then any non-unit vector is equivalent to its normalized version, i.e., of length 1. The state space of a qubit, then, is a 2-dimensional complex Hilbert space minus the origin modded out by the space of all possible global phase values. We take out the origin $(0,0)$ from the space $\mathbb{C}^2$ since the $|\alpha|^2 + |\beta|^2 = 1$ condition prohibits both α and β to be zero at the same time. The resulting space is $\mathbb{C}^2 - (0,0)$, modded out by $\mathbb{C}^*$, which is the space of all possible global phase values, obtaining

$$(\mathbb{C}^2 - \{(0,0)\}) / \mathbb{C}^* \cong (S^3 \times \mathbb{R}^+) / (U(1) \times \mathbb{R}^+) \cong S^3 / U(1) = \mathbb{C}P^1.$$

Geometrically the resulting projective space is of the spherical shape, equivalent to S^2. It is called the Bloch sphere with $|0\rangle$ as the north pole and $|1\rangle$ as the south pole.

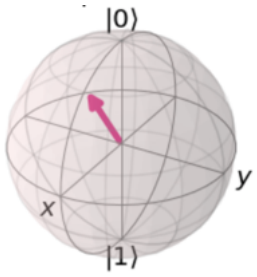


Figure 1. Quantum state space of a qubit

Any state of one qubit would lie on the Bloch sphere as a point, a representative of the entire global phase space $\mathbb{C}^*$.

Compare this qubit space of the Bloch sphere to the classical bit space of $\{0,1\}$. This expanded space used for quantum information processing is one of the areas believed to be giving more power to quantum computing than classical.

For quantum computation, we apply physical operations on the quantum state of a qubit inducing a change of state. A state-changing operator is called a quantum gate, and it is represented by a unitary matrix [24]. Unitary matrices are length-preserving linear transformation matrices, which represent rotations on the Bloch sphere. The mathematical definition of a unitary matrix is $UU^\dagger = U^\dagger U = I$. The inverse operation of a unitary matrix is be the reverse rotation, done via the complex conjugate matrix of the original matrix. Notice that the action of a unitary matrix is reversible, hence a quantum gate on a qubit is a reversible gate. We can apply multiple quantum gates to a quantum state and a collection of quantum gates is called a quantum circuit.

In multiple-qubit systems, the state space is the tensor product space of the individual qubit spaces and the computational basis, the set of all possible combinations of individual qubit computational basis, i.e., the tensor product of the basis states. For example, in a two-qubit system the set of all possible combinations of the computational basis states would be $\{|0\rangle\otimes|0\rangle, |0\rangle\otimes|1\rangle, |1\rangle\otimes|0\rangle, |1\rangle\otimes|1\rangle\}$ where $\otimes$ denotes a tensor product. In vector representation,

$$|0\rangle\otimes|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}\otimes\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\begin{bmatrix}1\\0\end{bmatrix}\\0\begin{bmatrix}1\\0\end{bmatrix}\end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}$$

Notice the size of the computational basis and the length of the basis vectors are both $2^2 = 4$. The state space of n qubits is a 2^n-dimensional Hilbert space with its computational basis being the set of all combinations of the state each qubit can be found in. Geometrically,

$$\mathbb{C}P^n = (\mathbb{C}^{n+1} - \{(0,0)\})/\mathbb{C}^* \cong (S^{2n+1}\times\mathbb{R}^+)/(U(1)\times\mathbb{R}^+) \cong S^{2n+1}/U(1).$$

The corresponding unitary gates acting on the entire system are represented by $2^n \times 2^n$ matrices as tensor products of n 2×2 unitary matrices.

The last stage of a quantum circuit is measurement, a read-out of computational results classically. It is a projection of the quantum computational resulting state onto one of the classical computational basis. Let

$$|\psi\rangle = c_0|00\ldots0\rangle + c_1|00\ldots1\rangle + \ldots + c_{2^n-1}|11\ldots1\rangle$$

be the final state of computation in an $n-$qubit system. The probability of getting the k^{th} computational basis state is given by "Born Rule" [24] as

$$prob(k) = \frac{|c_k|^2}{\sum_j |c_j|^2} = \langle\phi|k\rangle\langle k|\phi\rangle = |\langle k|\phi\rangle|^2$$

In each instance of measurement, we get one of the computational basis elements once. To extract the probability value of the k^{th} element, which is a projection of $|\psi\rangle$ onto $|k\rangle$, we perform multiple shots of the circuit operation.

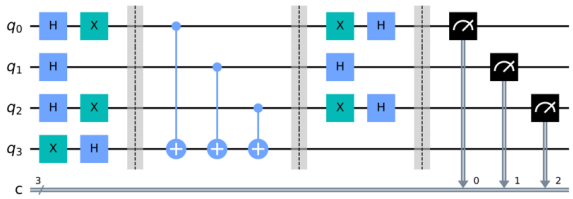


Figure 2. Quantum circuit diagram with measurement

In the circuit diagram above, each line represents the evolution of state of one qubit in time, boxes quantum gates, and the black arrow boxes measurement. Each qubit is connected to a classical wire for read-out.

2.2 Continuous Variable

The CV model works with wave-like properties of nature. It was first proposed by Lloyd and Braustein in 1998 [20] to use amplitudes of the electromagnetic field. They observed that many quantum variables such as position and momentum or amplitudes of electromagnetic fields are continuous, not discrete. In this model, information is carried in the quantum state of bosonic modes (qumodes), i.e. particle channels [42].

Information can be encoded either in wave-function representation or phase space representation [16]. A wave function is a mathematical way of describing the quantum state of a system, as probability amplitudes. The wave function of the position or the momentum of a particle is a complex-valued function of a real valued variable $\Psi : \mathbb{R} \to \mathbb{C} : x \mapsto \alpha$.

Let $\Psi(x)$ be the wave function of the position x of a particle. By interpreting the wave function as probability amplitude, we can get its probability density by multiplying it by its complex conjugate: $\|\Psi(x)\|^2 = \Psi(x)^*\Psi(x) = \rho(x)$. We can have either the wave function of the position observable or that of the momentum observable separately, but not both. To describe the quantum state of a system as a function of position and momentum simultaneously, we use phase space representation instead.

The phase space formalism is a way to describe the quantum state of a system as a function of the observable variables, i.e. position and momentum. The Wigner function is used for this purpose [10]. The Wigner function of the position x and the momentum p is given by

$$W(x,p) = \frac{p}{h} = \frac{1}{h}\int_{-\infty}^{\infty} e^{-\frac{ipy}{\hbar}} \Psi\left(x+\frac{y}{2}\right)\Psi^*\left(x-\frac{y}{2}\right)dy$$

where $h = 6.62607015 \times 10^{-34}$ is the Plank constant and $\hbar = 6.582119569 \times 10^{-16}$ the reduced Plank constant. Notice that $p,x,y \in \mathbb{R}$ and $\Psi(\cdot) \in \mathbb{C}$.

Consider the lowest energy state of a qumode where there is no particle present in the system, i.e., zero particle state. Its wave function is given by

$$\Psi_0(x) = \frac{1}{\sqrt[4]{\pi}\sqrt{a}} e^{-\frac{x^2}{2a^2}}$$

Using this wave function, we can derive the Wigner function of the lowest energy state.

$$\begin{aligned} W_0(x,p) &= \frac{1}{h}\int \frac{1}{a\sqrt{\pi}} e^{-\frac{ipy}{\hbar}} e^{-\frac{(x+\frac{y}{2})^2}{2a^2}} \left(e^{-(x-\frac{y}{2})^2/2a^2}\right)^* dy \\ &= \frac{2}{h} e^{-\left(\left(\frac{ap}{\hbar}\right)^2 + \left(\frac{x}{2}\right)^2\right)} \end{aligned}$$

where $a = \sqrt{\frac{\hbar}{m\omega}}$, m = angular momentum, and ω = angular frequency. It can be simplified by setting $a,h = 1$ as $W_0(x,p) = 2e^{(-(\frac{p}{\hbar})^2 - x^2)}$. The projection of $W_0(x,p)$ onto the position ($x-$ axis) and momentum ($p-$ axis) plane gives a Gaussian distribution on the position-momentum plane as shown in the following plot.

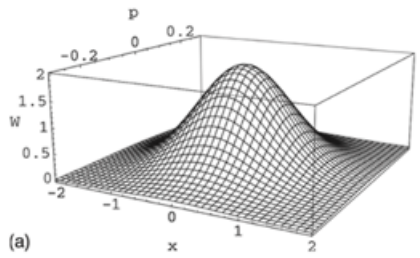


Figure 3. The Wigner function $W_0(x,p)$ onto the position-momentum plane [10]

In the same manner, we can derive the Wigner functions of the different energy states based on the number of photons $n = 1,2,\ldots,n,\ldots$ present in a system, using their wave functions. The projection of the resulting Wigner functions $W(x,p)$ onto the $x-$axis gives the probability distribution of the position x and onto the $p-$axis gives that of the momentum p. The resulting plot of each energy state is depicted in figure 4.

The collection of the projection of $W(x,p)$ onto the $xp-$ plane for each energy state is called Fock basis or number basis: $\{|0\rangle, |1\rangle, \ldots, |n\rangle, \ldots\}$ [16]. The quantum state of the system of a qumode is expressed as an infinite sum (superposition) of the elements of Fock basis.

$$|\psi\rangle = c_0|0\rangle + c_1|1\rangle + \ldots + c_n|n\rangle + \ldots \text{where} \sum_{k=0}^{\infty} |c_k|^2 = 1$$

This means a single qumode, realized with a harmonic oscillator system, can be in zero particle state $|0\rangle$, 1 particle state $|1\rangle$, 2 particle state $|2\rangle$, so on, and an infinite number of particle states. The coefficients c_k represent probability amplitudes, which collectively give the probability distribution of the system at measurement. The resulting space is called Fock space $\mathscr{F} = \bigoplus_{n=0}^{\infty} H^{\otimes n}$ where $H^{\otimes n}$ represents an infinite dimensional Hilbert space corresponding the Fock basis state $|n\rangle$ [6].

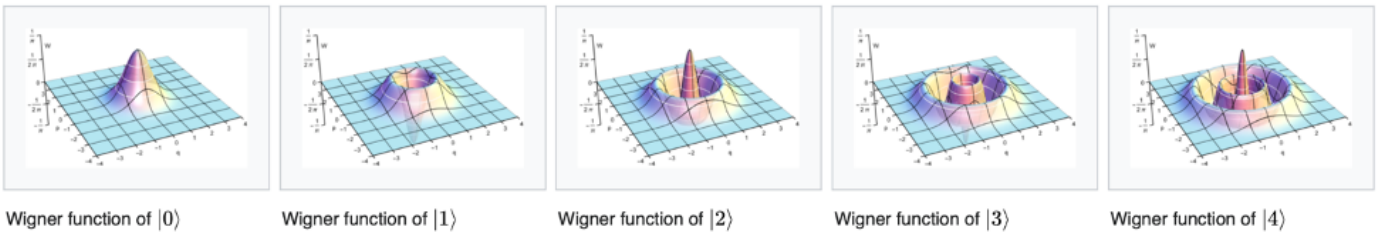


Figure 4. Fock basis image source

In a multiple qumode system, the quantum state space of the system is represented by the tensor product space of the individual qumode state spaces, which is yet another infinite dimensional Hilbert space. Let the quantum states of two qumodes be represented each as $|\phi\rangle$ and $|\psi\rangle$.

$$|\phi\rangle = b_0|0\rangle + b_1|1\rangle + \ldots + b_n|n\rangle + \ldots$$
$$|\psi\rangle = c_0|0\rangle + c_1|1\rangle + \ldots + c_n|n\rangle + \ldots.$$

Then the tensor product space of the entire system is

$$\begin{aligned}|\phi\rangle \otimes |\psi\rangle &= (b_0|0\rangle + b_1|1\rangle + \ldots + b_n|n\rangle + \ldots) \\ &\otimes (c_0|0\rangle + c_1|1\rangle + \ldots + c_n|n\rangle + \ldots) \\ &= b_0c_0|00\rangle + b_0c_1|01\rangle + \ldots + b_0c_n|0n\rangle + \ldots \\ &+ b_1c_0|10\rangle + b_1c_1|11\rangle + \ldots + b_1c_n|1n\rangle + \ldots \\ &+ \ldots \\ &+ b_nc_0|n0\rangle + b_nc_1|n1\rangle + \ldots + b_nc_n|nn\rangle + \ldots \\ &+ \ldots\end{aligned}$$

For the practical purposes of computing, we know that we are not going to have an infinite number of photons in a qumode. Hence, we approximate this infinite dimensional Hilbert space with a finite number of Fock basis elements by cutting off higher number Fock basis elements. The size of Fock basis we want to use to represent the system is called "cutoff dimension".

Suppose cutoff dim $= 3$ on a 2 qumode-system. Then for each qumode, the state of the quantum system is represented by a superposition of 3 Fock basis states: $|\phi\rangle = b_0|0\rangle + b_1|1\rangle + b_2|2\rangle$ and $|\psi\rangle = c_0|0\rangle + c_1|1\rangle + c_2|2\rangle$. The quantum state of the entire system is the tensor product of both qumodes:

$$\begin{aligned}|\phi\rangle \otimes |\psi\rangle &= b_0c_0|00\rangle + b_0c_1|01\rangle + b_0c_2|02\rangle \\ &+ b_1c_0|10\rangle + b_1c_1|11\rangle + b_1c_2|12\rangle \\ &+ b_2c_0|20\rangle + b_2c_1|21\rangle + b_2c_2|22\rangle\end{aligned}$$

Notice the Fock basis

$$\{|00\rangle, |01\rangle, |02\rangle, |10\rangle, |11\rangle, |12\rangle, |20\rangle, |21\rangle, |22\rangle\}$$

has $9 = 3^2$ elements, creating a $\left(3^2 - 1\right)-$dimensional projective Hilbert space. This can be generalized as the computational space of an $m-$qumode system with cutoff dimension$= n$ to be an $(n^m - 1)-$dimensional projective Hilbert space.

With phase space representation of quantum space in the CV model, the standard quantum gates are represented by matrix exponentials of various forms [42]. Matrix exponential of a matrix A is given by

$$e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!} = I + A + \frac{A^2}{2!} + \ldots + \frac{A^n}{n!} + \ldots$$

Specifically, quantum gates are Hamiltonian exponentials of the form $U = e^{-1^{H}/_{2}}$ where H is the Hamiltonian of the system, describing its energy state. Let cutoff dimension be n. Then we are approximating the infinite dimensional projective Hilbert space of the one qumode system with an $(n-1)-$dimensional projective Hilbert space. We use constructor $\hat{a}^\dagger$ and annihilator $\hat{a}$ as basis of Gaussian gates where

$$\hat{a}^\dagger = \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 & 0 \\ \sqrt{1} & 0 & 0 & \ldots & 0 & 0 \\ 0 & \sqrt{2} & 0 & \ldots & 0 & 0 \\ 0 & 0 & \sqrt{3} & \ldots & 0 & 0 \\ \vdots & & \ldots & \ddots & & \vdots \\ 0 & 0 & 0 & \ldots & \sqrt{n-1} & 0 \end{bmatrix} \text{ and}$$

$$\hat{a} = \begin{bmatrix} 0 & \sqrt{1} & 0 & 0 & \ldots & 0 \\ 0 & 0 & \sqrt{2} & 0 & \ldots & 0 \\ 0 & 0 & 0 & \sqrt{3} & \ldots & 0 \\ \vdots & & \ldots & \ddots & & \vdots \\ 0 & 0 & 0 & 0 & \ldots & \sqrt{n-1} \\ 0 & 0 & 0 & 0 & \ldots & 0 \end{bmatrix}.$$

Then standard Gaussian gates which take Gaussian states to Gaussian states are

Squeezer with parameter $z : S(z) = exp\left(\frac{z^*\hat{a}^2 + z\hat{a}^{\dagger 2}}{2}\right)$

Rotation with parameter $\phi : R(\phi) = exp\left(i\phi\hat{a}^\dagger\hat{a}\right)$

Displacement with parameter $\alpha : D(\alpha) = exp\left(\alpha\hat{a}^\dagger - \alpha^*\hat{a}\right)$

Beamsplitter with parameters θ and ϕ:

$$B(\theta, \phi) = exp\left(\theta\left(e^{i\phi}\hat{a}\hat{b}^\dagger + e^{-i\phi}\hat{a}^\dagger\hat{b}\right)\right)$$

where $\hat{b}^\dagger$ and $\hat{b}$ are constructor and annihilator of the 2nd qumode respectively [16].

After a series of quantum state changes are performed by quantum logic gates in a circuit, we need to extract the result of the computation. The CV model is implemented by linear optics (photonics), using photons in qumodes for information processing [3]. In photonic quantum computing, measurement is performed by counting the number of photons per qumode [4]. The resulting photon count is a projection of the quantum state onto one of the Fock basis states, i.e., number count of photons present in a qumode. For an $m-$qumode computation with the cutoff dimension n, the quantum state

$$|\psi_0\rangle \otimes |\psi_1\rangle \otimes \ldots \otimes |\psi_{m-1}\rangle =$$
$$d_0|00\ldots0\rangle + d_1|00\ldots1\rangle + c_{n^m-1}|n-1, n-1, \ldots, n-1\rangle$$

of the system collapses to one single state $|k_0 k_1 \ldots k_{n-1}\rangle$, where $0 \leq k_i \leq n-1$ and k_i represents the number of photons found in each i^{th} qumode. Like in the qubit model, multiple shots are performed to extract the probability of getting each number basis per qumode. The finite dimensional discrete (qubit) model can be embedded in the infinite dimensional CV model [42].

3. TYPES OF QUANTUM COMPUTERS

The models of physical quantum computers that are available can be divided into two groups: the adiabatic model implemented by D-wave and the gate model implemented by IBM, Google, Xanadu, and other companies.

3.1 Adiabatic quantum computing

In adiabatic quantum computing, the overall energy of a physical system is used for information processing. D-wave made headlines with its adiabatic quantum computer in which the Hamiltonian of the given system is used to find the global minimum solution for an objective function. Adiabatic quantum computing uses a gradual process of evolving the energy of a quantum mechanical system from the initial state to the state describing the solution to a given problem. It is well suited for optimization and sampling problems.

The total energy of a quantum mechanical system, both kinetic and potential, can be mathematically described by a function called Hamiltonian. It describes the energy of a system, as a function of the position and momentum of a particle, by mapping eigenstates to energies. Quantum annealing is the process of evolving the initial energy state to the global minimum solution state of an objective function. In physical systems, this ideal process is realized via an adiabatic process, which is a slow and gradual annealing process without interference from outside energy sources.

Manipulating the Hamiltonian of the overall system does not have readily direct correlations to the known classical algorithms. Therefore a lot of quantized versions of classical algorithms are better implemented in gate model quantum computing.

3.2 Gate model quantum computing

In gate model quantum computing, the quantum states of individual information carriers are used as units of information. The state is mathematically expressed as a vector, which is a complex linear combination of the quantized classical computational basis, in a complex projective Hilbert space. Controlled change of states of the information carriers is translated as a quantum logic gate, mathematically represented by a matrix [24]. Quantum gates, unlike classical logic gates, are reversible. A series of quantum gates is called a quantum circuit. The computational results of gate operations are accessed classically via an operation called measurement.

From the physical implementation perspective, quantum gates are realized by controlling either electrical or photonic waveforms [46]. Physical implementation of the gate model can be done via superconducting based on the discrete variable model, linear optics (photonics) based on the CV model, and ion trap, among other means that are being pursued. Although Honeywell and IonQ are working hard on ion trap, their chips have not been made available to the public. IBM and Google have cloud services of their superconducting chips and Xanadu also has of their photonic chips.

3.2.1 Superconducting model

In the superconducting model, the quantum information carrying agent is an individual particle [46]. Superconductor is a unique class of materials that exhibit no electrical resistance at zero frequency when cooled to below a critical temperature. From a collection of multiple quantized energy states of the superconductor, the two lowest energy states can be selectively accessed to realize the qubit: the ground state as the state $|0\rangle$ and the first excited state as $|1\rangle$ [46].

The quantum state of the particle is manipulated by microwave for computation [43]. A series of operations is performed on the particle for information processing. Precise control of its state for computation requires a supercondensing low temperature to avoid unwanted thermal excitation of the excited state. For Google's quantum chip, for example, it is cooled below 20 mK, using a dilution refrigerator. This sensitivity of a quantum particle to its environment is called a decoherence problem. The decoherence problem is something that needs to be addressed in superconducting quantum computers on the way to fault-tolerant universal quantum computing.

For read-out, each particle needs to connected to classical computers via a wire. Hence as the number of computational particles is increased, the size of the accompanying hardware grows. The interaction between the classical information from room temperature and the quantum data plane connected via wires inside a dilution refrigerator may cause unwanted thermal noise [46].

Since the quantum state of the overall system is based on a cluster of individual particles, each of which state is represented by a qubit, this is a natural implementation of the qubit model of quantum computing.

3.2.2 Linear optics model

Linear optics quantum computing model uses the electromagnetic field of bosonic modes as quantum information carriers. The quantum state of the overall system containing multiple qumodes is defined by the interference of photonic electromagnetic field of the qumodes. As the mathematical description of the electromagnetic field is continuous, linear optics naturally implements the CV model of quantum computing: optical instruments to carry out quantum computations and photon detectors to measure and store the computational results [45].

Available quantum information channels (qumodes) are initialized with no photons present, which are called vacuum states $|0\rangle$. The Wigner function $W_0(x,p)$ of each qumode is a Gaussian distribution depicting the lowest energy state. Laser beam light is applied to the qumodes and the classical light is converted to quantum light through a process called squeezing. It puts the deterministic classical light in superposition of Fock basis by squeezing the momentum observable of the light wave, thus spreading out the position observable. Application of a squeezer parameterized with the squeezing parameter z on the vacuum state $|0\rangle$ is given by [42]

$$S(z)|0\rangle = \frac{1}{\sqrt{\cosh z}} \sum_{n=0}^{\infty} \frac{\sqrt{(2n)!}}{2^n n!} \tanh^n z |2n\rangle .$$

Notice the squeezing operation converted classical light into an infinite sum of Fock basis, i.e., quantum light.

$$|0\rangle \rightarrow \sqrt{\frac{2}{e^z + e^{-z}}} \left(c_0|0\rangle + c_2|2\rangle + \ldots c_{2n}|2n\rangle + \ldots\right)$$

where $c_{2k} = \frac{\sqrt{(2k)!}}{2^k k!} \left(\frac{e^z - e^{-z}}{e^z + e^{-z}}\right)^k$ for $k = 0, 1, 2, \ldots$. The process can be visualized in the figure below.

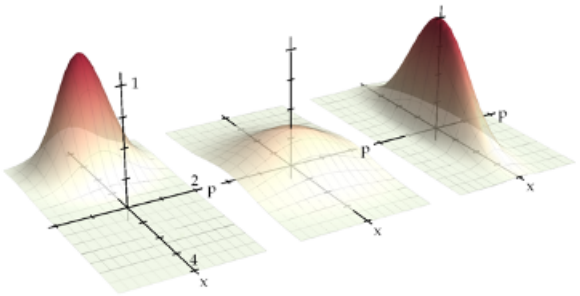


Figure 5. Squeezing: quantizing classical light [23]

Quantum gates are realized by controlling the ambient electromagnetic field of the qumodes using optical instruments. After computation, the resulting quantum states are accessed via photon detectors, by counting the number of photons in each qumode. The optical instruments function at room-temperature and are not susceptible to decoherence problems. Without the need for a huge dilution refrigerator, it can work on any regular size computer, hence can be a regular stand-alone server. It is easily scalable, since it can be embedded into the current fiber optics infrastructure [12].

The table below outlines the comparison between the superconducting model and the linear optics model.

	Superconducting	Photonic
properties	particle like	wave like
units of computation	quantum bits	quantum modes
single mode computational basis	$\|0\rangle, \|1\rangle$ quantized digital binaries	$\|0\rangle, \|1\rangle, \ldots, \|n\rangle, \ldots$ Fock (number) basis
basis size	2	infinite
multi mode computational basis	$\{\|0\rangle, \|1\rangle\}^{\otimes^m}$ tensor product	$\{\|0\rangle, \|1\rangle, \ldots, \|n\rangle, \ldots\}^{\otimes^m}$ tensor product
basis size	2^m	infinite
state space	Hilbert space	Fock space Direct sum of Hilbert spaces
physical implementation	electron	photons
temperature	below 20 mK	room temperature
server	dilution refrigerator	regular server

4. QUANTUM PROCESSING UNITS

Under the gate model of quantum computing, we examined the difference between the superconducting model and the linear optics model. IBM's Honeycomb and Google's Sycamore are superconducting QPUs and Xanadu's X8 is an 8-qumode linear optical QPU. IBM just unveiled a 127 qubit QPU called Eagle, allowing for up to a $2^{1}2710^{3}8-$dimensional Hilbert space for quantum computation. Google's Sycamore offers 63 qubits, allowing for up to a $2^{6}310^{1}9-$dimensional Hilbert space. Xanadu's X8 has 8 qumodes, each of which can be set at a certain cutoff dimension. For building quantum circuits, Google offers a software package called Cirq, IBM Qiskit, and Xanadu Strawberry Fields. To better understand the mechanics of superconducting and photonic QPUs, we have a closer look at Google's Sycamore and Xanadu's X8.

4.1 Sycamore

Google's 63 qubit QPU sycamore uses aluminum for metalization and a thin layer of non-superconducting indium for bump-bonds between two silicon wafers. Conducting electrons on the wafers are condensed to macroscopic quantum state, such that currents and voltage behave quantum mechanically. In order to achieve that, the chips are cooled to below 20 mK in a dilution refrigerator.

The architecture of the chip is of lattice structure where each

node represents a qubit. It is composed of nonlinear resonators at 5 - 7 Ghz. As controls, a microwave drive is used to excite the qubit and a magnetic flux control to tune the frequency.

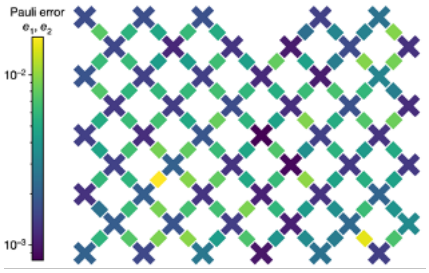


Figure 6. Lattice structure of Sycamore chip [image source]

For readout, a linear resonator connected to each qubit simultaneous readout using frequency- multiplexing technique. The software package offered for quantum circuits is Cirq, based on Python.

4.2 X8

Xanadu's 8-qumode photonic QPU is made of silicon nitride for photon conductance. It is housed in a conventional server at room temperature and is accessible to the user via Python-based Strawberry Fields from a personal computer. A master controller, connected to a personal computer, controls the number of qumodes to be used, quantum circuit building, information processing, and read-out of the measurement results back to the user. For computation, classical light is pumped into the qumodes on the chip, converted into quantum light, goes through a series of quantum state changes based on a user-defined quantum circuit, filtered, and measured. The functional components of X8 are Pump I/O, pump distribution, squeezing, filtering, interferometer and programmable quantum gates. Pump I/O is a custom modulated pump laser source that produces a regular pulse train. The pump distributor directs the photons in classical states to appropriate qumodes.

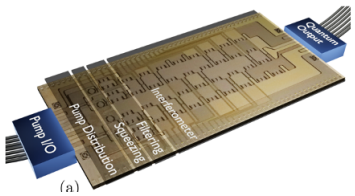


Figure 7. Xanadu's 8-qumode chip [image source]

The process of squeezing converts the classical light into quantum squeezed state. The interferometer portion of the chip performs quantum circuit operations. The output from the chip is run through a filter to suppress unwanted light, passing only wavelengths close to the signal. Then the photon counter reads the number of filtered photons in each qumode and sends the result of the computation back to the user via master controller.

Photonic quantum computing has several advantages over superconducting quantum computing. The most important is that it can be operated at room temperature. It is also compatible with existing optical infra structure, hence networking and multiplexing would be a natural extension of the existing infrastructure. It is robust to decoherence, hence offers near perfect phase stability. The dimension of the entangled state grows exponentially with the number of photons and the number of modes.

5. QUANTUM MACHINE LEARNING

Machine learning is a method of extracting hidden patterns from data that can be applied to new data samples for predictions. A machine learning algorithm is a mathematical expression, whose parameters can be trained to best describe the hidden patterns of data. An algorithm with optimized parameters through training that minimize the given cost function is called a machine learning model.

In quantum machine learning, a quantum circuit implements an algorithm on a QPU and returns results via measurement operation. The cost of the results, gradients, and parameter updates are calculated on a CPU. The whole process is done through a combination of classical and quantum circuits called a variational circuit. The updated parameters, calculated classically, are then used to modify the the quantum circuit and the process is iterated to develop a model.

The first step of building a quantum machine learning model is data encoding, and it is called quantum data preparation. The prepared data are processed through a quantum circuit, which is multiple layers of quantum logic gates. The measurement results are then sent to the CPU of the variational circuit for parameter optimization. The main component of this circuit is the data encoding scheme, the way to encode classical data into quantum states. To understand the importance of the encoding scheme, we explore the parallel between the classical kernel method and quantum state preparation method.

Google and Xanadu offer Python based software packages specifically for quantum machine learning: Tensorflow Quantum (Google) and Pennylane (Xanadu) [7].

5.1 Quantum machine learning models as Kernel methods

Quantum machine learning can be viewed as a way of extracting patterns from data in a higher dimensional Hilbert space [29]. This is similar to the kernel method in a classical support vector machine, where a higher dimensional feature map space is used for extracting data patterns. If we view quantum machine learning models as the kernel method carried out on quantum circuits, what sets an algorithm apart is the encoding scheme of converting classical data into quantum states.

The kernel method is used in classical support vector machines, when an optimal separating hyperplane is not easily defined in the data space [19]. A way of expanding the dimension of data space where a separating hyperplane is better defined is to map data into a higher dimensional feature space using a feature map. In the example on the right, we find a easily defined separating hyperplane in the 3-dimensional feature space using a feature map $\Phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3 : x \mapsto \Phi(x)$ on the $2-$dimensional data space.

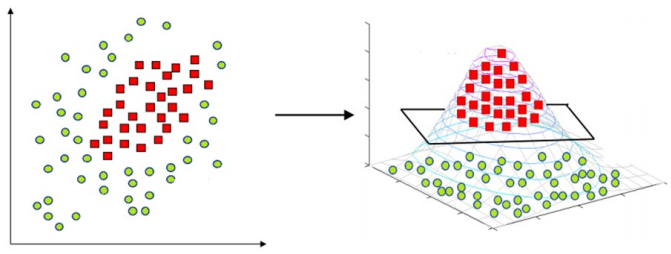


Figure 8. Extended feature space where a separating hyperplane is better defined image source

The method of finding a separating hyperplane is to calculate a similarity measure of the image of data points in the feature space using the inner product of two vectors as similarity measure. The kernel function k of two data vectors x, z is defined as $k(x,z) = \langle \Phi(x), \Phi(z) \rangle$ where $\Phi(\cdot)$ is a feature map [29].

The kernel of the data samples which is the inner product of the corresponding samples in the feature space allows us to glean the angle between them.

$$\langle \Phi(x), \Phi(z) \rangle = \cos\theta \|\Phi(x)\| \|\Phi(z)\|$$
$$\Rightarrow \cos\theta = \frac{\langle \Phi(x), \Phi(z) \rangle}{\|\Phi(x)\| \|\Phi(z)\|}$$

Notice $-1 \leq \cos\theta \leq 1$ where $\cos 0 = 1$. Hence the higher the inner product $\langle \Phi(x), \Phi(z) \rangle$, the smaller the angle between them which leads to greater similarity between the two images of the feature map. The actual feature map and the image of the map are hidden. To find an optimal separating plane, all we need is the kernel of all pairs of data vectors, bypassing the need to know the actual feature map or the image of the map.

Similarly in quantum computing, we get the inner product $\langle \psi_k | \psi_k \rangle$ for each $k^t h$ element of the computational result state $|\psi\rangle$ from measurement, while information processing is done in a higher dimension not directly accessible to the user [29].

The classical input data are mapped into a higher dimensional complex Hilbert space of quantum information and the quantum circuit is applied to the corresponding quantum states. Thus we get results in the form of real valued vectors according to Born Rule: $prob(k) = |\langle k|\psi\rangle|^2 = \langle \psi_k | \psi_k \rangle$ where ψ_k is the k^{th} element of $|\psi\rangle$. Suppose $\psi_k = a + bi$. Then

$$\langle \psi_k | \psi_k \rangle = (a+bi)^*(a+bi) = (a-bi)(a+bi) = a^2 + b^2.$$

Via measurement, we can get a single real value $a^2 + b^2$ but there is no way of knowing the exact values of a and b.

If we consider the feature map as encoding classical data into quantum states and the kernel as the inner product of quantum states, then a trained quantum model can be viewed as a classical kernel method whose kernel is computed by a quantum computer [29].

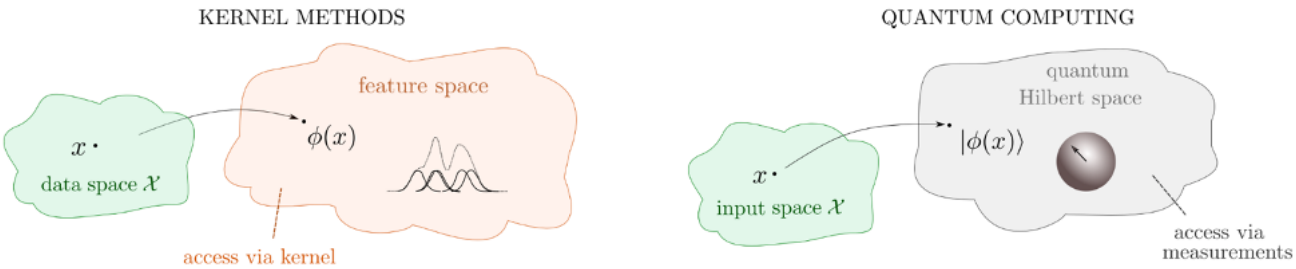


Figure 9. Extending data space [29]

5.2 Data Encoding

Suppose we are encoding the classical data vector into a quantum state using the squeezer, obtaining quantum encoded data vector x. If we consider the quantum encoded state of classical data sample as x, the parameterized quantum circuit as quantum information processing and the measurement operator as the image of a feature map, then the feature map is dependent on the original quantum encoded state and the parameter. Interchangeably, we can consider the parameterized circuit and the act of measurement as a parameterized measurement operator. The feature map is solely dependent on the quantum encoded state. Then the only factor defining the image of the quantum circuit is the value of the encoded quantum state vector. The quality of a quantum model is defined by the quantum embedding strategy of the classical data. One way of encoding is using the entries of each vector as parameters of available quantum gates. Photonic quantum computing provides a rich way to encode data with its linear and nonlinear gates.

6. QUANTUM NEURAL NETWORKS

In classical neural networks, each layer of a neural network is expressed mathematically as $L(x) = \phi(Wx+b)$ where W is a weight matrix, b a bias, and $\phi(\cdot)$ a nonlinear activation function. The quantum version of a neural network would be encoding given feature vector x into a quantum state and performing a series of quantum operations equivalent to the classical transformation $L(x)$. Then the goal of quantum neural networks would be implementing the notion of $L(|x\rangle) = |\phi(Wx+b)\rangle$, using available quantum gates [16].

In the qubit model, the unitary gates are essentially all linear in nature while the nonlinear component can be actualized via measurement. To implement a non-linear activation function $\phi(\cdot)$, one has to perform a measurement. Hence each quantum neural network can implement only one layer of quantum neural network. Additionally, there is no direct way to perform affine transformations representing biases. For quantum data encoding in MNIST binary classification models, Google uses the Pauli-X gate and IBM uses the Hadamard followed by a parameterized R_y gate. The Google approach is to prepare qubits with the Pauli$-X$ gate for the values above a certain threshold and applying 2-qubit rotation gates pairing with the read-out qubit. The IBM approach is to use a classical convolutional network with $1-$dimensional output and using it as the parameter for an R_y gate on the uniform superposition of $|0\rangle$ and $|1\rangle$. Thereafter, measurement is performed without any additional transformation, hence the model can be viewed as a quantum data encoding model, not necessarily a neural network.

In Xanadu's CV model, $L(|x\rangle) = |\phi(Wx+b)\rangle$ can be implemented almost exactly via available CV gates [16]. The composition layer of the gates $L = \phi \circ D \circ U_2 \circ S \circ U_1$ is an exact translation of a classical neural network into quantum, where U_k is an interferometer, S a collection of squeezers, D a collection of displacement gates, and ϕ a collection of Kerr gates. An interferometer is composed of beamsplitters on each adjacent pair of qumodes and rotation gates on all available qumodes.

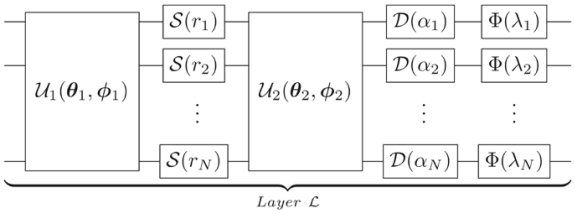


Figure 10. CV model implementation of QNN [16]

The affine transformation $Wx+b$ is realized with $\circ D \circ U_2 \circ S \circ U_1$. The composition of the gates $U_2 \circ S \circ U_1$ implements a quantum version of the linear transformation matrix W and a collection of displacement gates, the bias b.

Any matrix M can be factorized using singular value decomposition (SVD) as $M = U\Sigma V^*$, where U and V are orthogonal and Σ is diagonal [47]. An interferometer is composed of beamsplitters and rotation matrices, where beamsplitters act on a pair of two qumodes. Hence one-qumode interferometer is just one rotation matrix and a multi-qumode interferometer has $m-1$ beamsplitters and m rotation matrices. The action of a phaseless interferometer U_i on the quantum state $|x\rangle = \otimes_{i=1}^{m} |x_k\rangle$ has an effect of an orthogonal matrix acting on $|x\rangle$. Orthogonal matrices are just unitary matrices with real entries, inducing length-preserving rotations. Then the transpose of an orthogonal matrix represents the reverse rotation of the original matrix, thus orthogonal.

The parameterized squeezer $S(r_k)$ acts on the quantum state $|x_k\rangle$ of each k^{th} qumode as $S(r_k)|x_k\rangle = \sqrt{e^{-r_k}}|e^{-r_k}x_k\rangle$. Collectively they have an effect of a diagonal matrix S acting on $|x\rangle = \otimes_{i=1}^{m} |x_k\rangle$.

Then the composition $U_2 \circ S \circ U_1$ can be considered as the composition $O_2 \circ \Sigma \circ \left(O_1^T\right)^T$, where O_2 and O_1^T are orthogonal and Σ diagonal. Together, the interferometers and squeezers implement an SVD form of some linear transformation W.

The bias addition is realized with displacement gates D. The displacement gate has an effect $D(\alpha_k)|\psi_k\rangle = |\psi_k + \sqrt{2}\alpha_k\rangle$

for each k^th qumode. Then $D(\alpha)|\psi\rangle = |\psi + \sqrt{2}\alpha\rangle$ collectively for $\alpha^T = [\alpha_1, \alpha_2, \ldots, \alpha_m]$. For some desired bias b, let $\alpha = \frac{b}{\sqrt{2}}$, then the collection of displacement gates implements the bias addition. The composition $D \circ U_2 \circ S \circ U_1$ acting on the quantum state $|x\rangle$ gives us the affine transformation

$$D \circ U_2 \circ S \circ U_1 |x\rangle = |O_2 \Sigma O_1 x + b\rangle = |Wx + b\rangle .$$

The non-linear activation function $\phi(\cdot)$ is realized with Kerr gates. The Kerr gate, parameterized by the parameter κ, is a non-linear transformation gate. Let n be the cutoff dimension and m the number of qumodes. For the quantum state $|\psi\rangle$ of one qumode, which is a superposition of n Fock basis states, the Kerr gate with parameter κ has an effect

$$K(\kappa)|\psi\rangle = \begin{bmatrix} e^{i\kappa 0^2} & 0 & \ldots & 0 \\ 0 & e^{i\kappa 1^2} & \ldots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \ldots & e^{i\kappa(n-1)^2\kappa} \end{bmatrix} \begin{bmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{n-1} \end{bmatrix}$$
$$= \begin{bmatrix} \psi_0 \\ e^{i\kappa 1^2}\psi_1 \\ \vdots \\ e^{i\kappa(n-1)^2}\psi_{n-1} \end{bmatrix},$$

which is non-linear.

Together, the circuit $L = \Phi \circ D \circ U_2 \circ S \circ U_1$ gives us a quantum version $L(|x\rangle) = |\Phi(Wx + b)\rangle$ of a classical neural network $L(x) = \Phi(Wx + b)$.

6.1 Qubit-based Implementation of MNIST Classification

This section is an analysis of qubit-based quantum neural network MNIST classifiers. There are two models, one by Google using Tensorflow Quantum and the other by IBM using Qiskit and PyTorch. Both models perform measurements on only one qubit whose computational basis is of size 2. Hence both models are binary classifiers. The Google version is a pure quantum network and the IBM version is a classical-quantum hybrid network with the classical network being a convolutional neural network.

6.1.1 Tensorflow Quantum

QNN implementation of MNIST classification on Tensorflow Quantum website is a binary classifier, composed purely of a quantum circuit. It uses a dataset of 10,338 samples of the classes 3 and 6 with all the pixel values $0 \sim 255$ normalized to $0 \sim 1$. The 28×28 image matrices are reduced to 4×4 and each pixel value is converted either to 0 or 1 depending on the threshold value 0.5. The quantum circuit is composed of $4 \times 4 = 16$ grid qubits mirroring the image matrices. An ancillary qubit is added for read-out, which is the only qubit that gets measured. A simplified version on the circuit is shown in the figure.

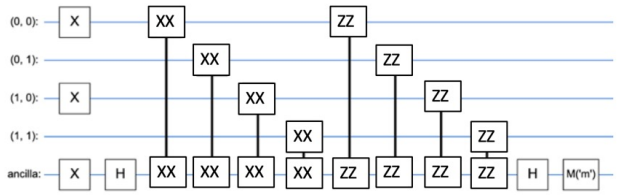


Figure 11. Simplified circuit of MNIST binary classifier

The quantum gates used are Pauli-X gates, Hadamard gates, $XX-$gates, and $ZZ-$gates. The unitary matrix representation of the parameterized $XX-$gate is

$$(X \otimes X)^t = \begin{bmatrix} \cos\left(\frac{\pi t}{2}\right) & 0 & 0 & -i\sin\left(\frac{\pi t}{4}\right) \\ 0 & \cos\left(\frac{\pi t}{2}\right) & -i\sin\left(\frac{\pi t}{2}\right) & 0 \\ 0 & -i\sin\left(\frac{\pi t}{2}\right) & \cos\left(\frac{\pi t}{2}\right) & 0 \\ -i\sin\left(\frac{\pi t}{2}\right) & 0 & 0 & \cos\left(\frac{\pi t}{2}\right) \end{bmatrix},$$

where t is the parameter.

The matrix representation of parameterized $ZZ-$gate with parameter s is given by

$$(Z \otimes Z)^s = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{i\pi s} & 0 & 0 \\ 0 & 0 & e^{i\pi s} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The data flow of the circuit is as follows:

- Initialize all the qubits to the state $|0\rangle$.
- Data encoding: Apply the Pauli-X gate to the qubits whose corresponding pixel value is 1.
- Read-out qubit preparation: Obtain the state $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$ by applying the Pauli-X gate and the Hadamard gate.
- Initialize 32 parameters for 16 $XX-$gates and 16 $ZZ-$gates.
- Apply parameterized $XX-$ gate operations to (read-out qubit, $k^{th}qubit$) pairs.
- Apply parameterized $ZZ-$ gate operations to (read-out qubit, $k^{th}qubit$) pairs.

- Apply the Hadamard gate to the read-out qubit.
- Measure the read-out qubit, obtaining the expectation value $\langle\psi|Z|\psi\rangle$ where $|\psi\rangle$ is the final computational result state of the read-out qubit.
- Calculate the hinge loss function on a CPU.
- Optimize and update the parameters.

After the application of $XX-$gates and $ZZ-$gates, the state of the read-out qubit becomes $\begin{bmatrix} 1 \\ -e^{i\pi\Sigma_0^{15} s_j} \end{bmatrix}$. When the Hadamard gate is applied, we get the state $\begin{bmatrix} 1-e^{i\pi\Sigma_0^{15} s_j} \\ 1+e^{i\pi\Sigma_0^{15} s_j} \end{bmatrix}$. The expectation value $\langle\psi|Z|\psi\rangle$ gives us

$$\begin{aligned}
&\langle\psi|Z|\psi\rangle \\
&= \begin{bmatrix} \left(1-e^{i\pi\Sigma_0^{15} s_j}\right)^* & \left(1+e^{i\pi\Sigma_0^{15} s_j}\right)^* \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1-e^{i\pi\Sigma_0^{15} s_j} \\ 1+e^{i\pi\Sigma_0^{15} s_j} \end{bmatrix} \\
&= \left(1-e^{i\pi\Sigma_0^{15} s_j}\right)^* \left(1-e^{i\pi\Sigma_0^{15} s_j}\right) - \left(1+e^{i\pi\Sigma_0^{15} s_j}\right)^* \left(1+e^{i\pi\Sigma_0^{15} s_j}\right) \\
&= \|1-e^{i\pi\Sigma_0^{15} s_j}\|^2 - \|1+e^{i\pi\Sigma_0^{15} s_j}\|^2 \in \mathbb{R}
\end{aligned}$$

As we identify lower values to class 3 and higher values to class 6, we want $\|1-e^{i\pi\Sigma_0^{15} s_j}\|^2 < \|1+e^{i\pi\Sigma_0^{15} s_j}\|^2$ for class 3 and $\|1-e^{i\pi\Sigma_0^{15} s_j}\|^2 > \|1+e^{i\pi\Sigma_0^{15} s_j}\|^2$ for class 6.

After 3 epochs, the model achieves hinge loss of 0.3673 and training accuracy of 87.69%. It achieves 90.42% evaluation accuracy on the test set of 1,968 samples.

Online code on Tensorflow Quantum

6.1.2 Qiskit PyTorch

The Qiskit approach in a strict sense is not necessarily a quantum network, but rather a quantum data encoding scheme. The network is composed of classical convolutional layers with output of length one, which is encoded as quantum data and then measurement. The convolutional layers are coded using PyTorch and quantum encoding and measurement using Qiskit. The classical convolutional network is composed of two convolutional layers each with kernel size (5,5), max pool, and activation function Rectified Linear Unit (ReLU). The resulting image matrices are flattened, and then one hidden layer of feed-forward neural network is applied with activation function ReLU. The output of the classical network is a vector of length one for each image and it is encoded into quantum state, by being used as the parameter for an R_y gate. Then without any quantum neural network circuit, measurement is performed directly on the quantum encoded data.

The reduced dataset from the original MNIST dataset contains classes 0 and 1.

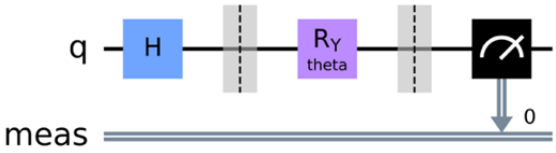


Figure 12. Data encoding/ measurement circuit

The composition operation of the Hadamard gate and a parameterized R_y gate is

$$\begin{aligned}
R_y(\theta)\circ H|0\rangle &= \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\
&= \frac{1}{\sqrt{2}} \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right) \\ \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right) \end{bmatrix}
\end{aligned}$$

To get classification $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, we want

$$\begin{aligned}
&\cos\left(\frac{\theta}{2}\right) - \sin\left(\frac{\theta}{2}\right) > \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right) \\
&\Rightarrow -\sin\left(\frac{\theta}{2}\right) > \sin\left(\frac{\theta}{2}\right)
\end{aligned}$$

Then we want $\frac{\theta}{2}$ to lie in the third or fourth quadrant of the unit circle:

$$\pi < \frac{\theta}{2} < 2\pi \Rightarrow 2\pi < \theta < 4\pi.$$

By the same token, we want $0 < \theta < 2\pi$ to get the label 1. The target range for the parameter θ as output from the classical network is very clear. Hence, most of heavy weight lifting is done by the classical network not by the quantum circuit.

For the computation of the gradient, the network employs the analytic differentiation method, given by

$$\frac{dy}{d\theta} = meas\left(R_y(\theta+\delta)\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}\right) - meas\left(R_y(\theta-\delta)\frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}\right)$$

for some small $\delta \in \mathbb{R}$. Hence measurements of the same circuit are performed twice with slight variation of the parameter value θ by δ and the difference is considered as the gradient of the circuit parameterized by θ. Optimization is carried out by the PyTorch in-built Adam optimizer with learning rate 0.001. The model achieves 100% training accuracy on 100 samples.

Online code on Qiskit

6.2 CV implementation of QNN

This is an examination of the quantum curve fitting neural network proposed in the paper [16] implemented by

Xanadu, using Pennylane. Curve fitting is linear regression using data x and target $f(x)$. The function used in this example is $f(x) = \sin x$. We want the network to learn an optimal set of parameters for the gates so that the output is as close to $\sin(x)$ for input data x.

The architecture of the quantum neural network proposed in the paper is composed of the displacement gate for quantum data encoding, 25 layers of quantum neural network on a single qumode and measurement result interpreted as $f(x)$.

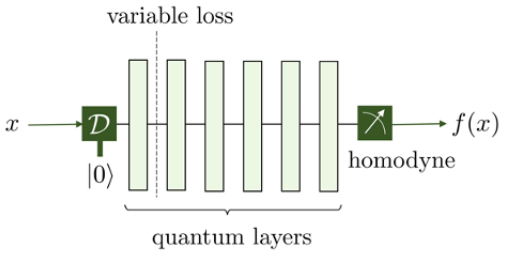


Figure 13. QNN circuit architecture for curve fitting [16]

The data flow of the network is as follows:

- Data encoding: Use the input value as the parameter of the Displacement gate.
- QNN: Apply rotation matrix, squeezer, rotation matrix, displacement gate, and Kerr gate. Repeat 25 layers.
- Measurement:
- Update the parameters using the square loss function and Adam optimizer with learning rate 0.01

The result of the network is marked with red crosses. From the noisy data marked in blue, the model was able to accurately learn and predict $f(x) = \sin(x)$.

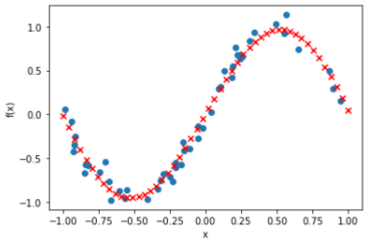


Figure 14. Model output $f(x) = \sin(x)$ in red

Online code on Pennylane

7. EXPERIMENTS

The experiments in this section include Pennylane-Keras implementation of proposed QNNs [16] and a novel hybrid classifier on MNIST.

Quantum neural networks can be used in conjunction with classical feed-forward networks. The quantum circuit can be viewed as a sub-circuit inside an overall classical neural network, whose component is farmed out to a QPU. The calculations needed for optimization - loss function, gradients, updated parameters - are carried out classically. Pennylane's plug-in feature allows for conversion of the quantum circuit into a Keras layer. Then the whole network can be treated as a classical neural network which can be easily trained using Keras' or PyTorch's built-in loss functions and optimizers.

In the following hybrid models, the role of the classical layers is for pre-processing of data into vectors of desired size. This is done to match the number of parameters for data encoding gates. The QNN circuits uniformly include the interferometer $\rightarrow$ squeezer $\rightarrow$ interferometer $\rightarrow$ displacement gate $\rightarrow$ Kerr gate sequence. Here the notion of cutoff dimension plays an important role as a powerful tool to control the length of the output vectors. When used together with the probability method for measurement, the length of the output vectors equals n^m where n =cutoff dimension and m = the number of qumodes.

7.1 Auto-encoder

Auto encoder is a network, composed of an encoder which encodes data in a compact manner and a decoder which retrieves the original data. The proposed classical and quantum hybrid auto-encoder uses a classical neural network as encoder and a one-qumode quantum neural network as decoder [16]. The classical encoder compresses the input vectors of length 3 into vectors of length 2. The quantum decoder takes in the vectors of length 2 and outputs vectors of length 3. A one qumode circuit is used with cutoff dimension= 3. The training process is to find optimal parameters for the classical encoding gates and the decoding quantum circuit so that the output vectors are as close to the original data as possible. The architecture of the proposed hybrid network is shown below.

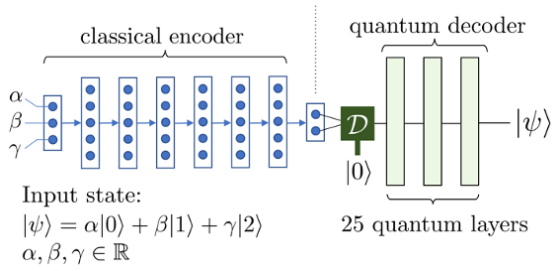
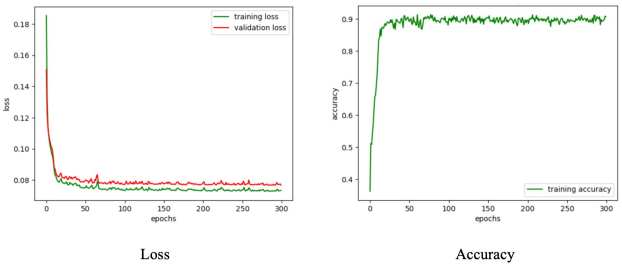


Figure 15. Auto-encoder hybrid circuit architecture [16]

The pipeline of the network is:

- classical encoder: 6 hidden layers, each with 5 neurons using Exponential Linear Unit (ELU) as activation function. Output layer with 2 neurons.
- data encoding: The 2 entries of the output vector from the encoder are used as parameters of the displacement gate to prepare a quantum state.
- quantum decoder: 25 layers of the sequence - rotation matrix 1, squeezer, rotation matrix 2, displacement gate, and Kerr gate.
- measurement: probability method, obtaining output vectors of size equal to the cutoff dimension.

The output vectors from the network are compared to the original vectors. Keras' built-in Mean Squared Error loss function and Adam optimizer with learning rate=0.01 are used for parameter update. After 300 epochs with batch size 50, the model achieves loss= 0.0736 and accuracy= 90.54%.



Loss Accuracy

Figure 16. Auto-encoder experimental results

7.2 Binary Classifier

The paper "Continuous-variable quantum neural networks" proposes a classical quantum hybrid network for supervised binary classification using credit card transaction data [16]. The original data set contains 284,806 genuine and fraudulent credit card transactions with 29 features, out of which only 492 are fraudulent. For the experiment, I selected only 10 features as per the paper and 1,968 samples with 1:3 ratio of fraudulent vs. genuine. The labels are converted to one-hot encoded vectors of size 2.

The proposed classical-quantum hybrid model is composed of three main parts: a classical neural network, a data encoding circuit, and a quantum neural network. The quantum circuit is built on 2 qumodes. The architecture of the hybrid network is

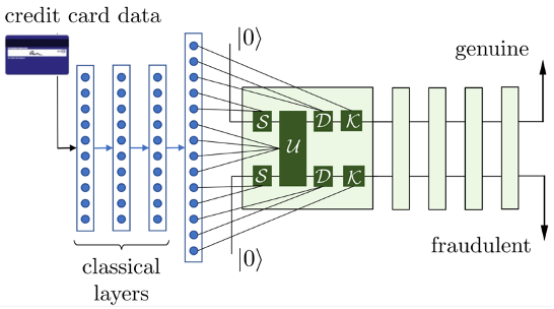
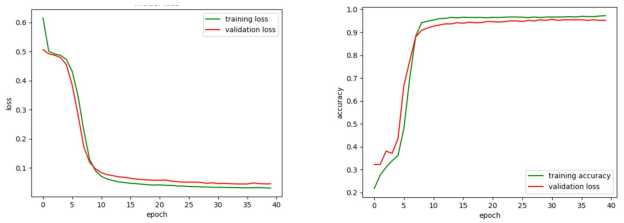


Figure 17. Binary hybrid classifier circuit architecture [16]

The data flow of the network is

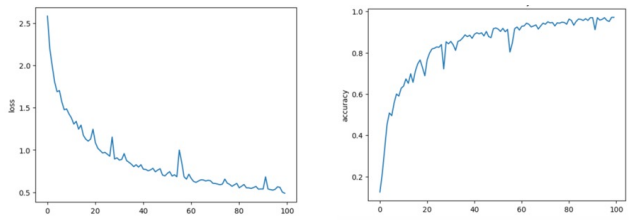
- classical network: 2 hidden layers composed of 10 neurons with ELU activation function and an output layer with 14 neurons.
- data encoding: 14 entries of the output vectors from the classical network are used them as parameters of squeezers, interferometer, displacement gates, and Kerr gates.
- QNN: 4 iterations of the sequence - interferometer 1, squeezers, interferometer 2, displacement gates, and Kerr gates
- measurement: expectation value of the Pauli$-X$ gate $\langle\phi_k|X|\phi_k\rangle$ for each qumode's final state $|\phi_k\rangle$, producing a vector of size 2: $[\langle\phi_0|X|\phi_0\rangle, \langle\phi_1|X|\phi_1\rangle]$.

The output vector $[\langle\phi_0|X|\phi_0\rangle, \langle\phi_1|X|\phi_1\rangle]$ is regarded as prediction vector $[genuine, fraudulent]$ for classification. Keras' built-in Mean Squared Error loss function and Adam optimizer with learning rate 0.01 are used for optimization. After 40 epochs with batch size = 190, the model achieves loss= 0.0321 and accuracy= 97.00%.



Loss Accuracy

Figure 18. Binary classifier experimental results



Loss Accuracy

Figure 19. MNIST multi classifier experimental results

7.3 MNIST Multi Classifier

Expanding on the proposed binary classification hybrid model, I implemented a multi-class hybrid classifier on the MNIST dataset. Comparison with the qubit-based MNIST binary classifiers is shown in the table.

	Tensorflow Q	Qiskit - PyTorch	Pennylane - Keras
Number of classes	2	2	10
Number of samples	10,338	100	600
Training accuracy	89.92%	100%	97.23%

The circuit is composed of 2 qumodes as in the binary classifier. I set cutoff dimension $= 4$ to get the desired output vector size of $4^2 = 16$, which is the nearest possible integer from the network greater than 10. Each qumode, therefore, is in quantum superposition of 4 Fock basis states: $\{|0\rangle, |1\rangle, |2\rangle, |3\rangle\}$. On the 2 qumode network, the quantum state of the entire system is the tensor product of two vectors of size 4, which is of size $4^2 = 16$. The pipeline is consisted of

- Keras sequential layers with output size 14
- data encoding: using the Keras output vector as parameters of squeezers, interferometer, displacement gates, and Kerr gates
- QNN: interferometer 1, squeezers, interferometer 2, displacement gates, and Kerr gates
- measurement: probability method, obtaining output vectors of size $4^2 = 16$

The modification to the binary classifier is the measurement method from expectation to probability. One-hot encoding of the labels and padding them with extra 6 zeros result in label vectors of dimension 16, which are compatible with the network output vectors.

For experiment, I used 600 samples for training, categorical crossentropy as loss function and stochastic gradient descent optimizer with learning rate 0.02. After 100 epochs with batch size, the model achieves loss= 0.324 and accuracy= 97.23%.

8. CONCLUSION

I examined the two fundamentally different models of quantum computing: the discrete variable model realized with superconducting quantum computers and the CV model realized with photonic quantum computers. The CV model offers more flexibility and computational freedom with a richer array of quantum gates for data encoding and information processing. This is due to the fact Fock space, which is the computational state space of the CV model, is infinite dimensional while the qubit-based computational state space is finite.

Especially for building quantum neural network circuits, there is a natural way of directly translating the weight matrix, bias, and non-linear activation functions in the CV model. Viewing quantum computation as a kernel method, we examined how a quantum model is defined by the data encoding scheme, converting classical data into quantum states.

For comparison of different quantum encoding schemes under the qubit model and the CV model, I analyzed the circuits for MNIST binary classification implemented using Tensorflow Quantum and Qiskit. Building a multi-class classification network on the qubit model would require a very complex scheme of using multiple qubits as control. Due to a rich array of CV quantum gates and multiple measurement operators for producing different size output vectors, it is much more natural to build multi-class classification models in photonic quantum computers.

Using Xanadu's Pennylane and its Tensorflow plug-in feature, I implemented the auto-encoder and the binary classifier proposed in "Continuous variable quantum neural networks" [16]. The plug-in function allows for easy conver-

sion of quantum circuits into Keras layers, which then are easily incorporated into Keras' optimization scheme with access to built-in loss functions and optimizers. I extendeded the binary classifier to classify multi-class labels on MNIST, using the notion of "cutoff dimension" and different measurement methods Pennylane offers. The current model achieves 97.23% training accuracy on dataset of 600 samples.

As further work, I can explore the relationship between the cutoff dimension and the number of qumodes in auto-encoders with larger input size or multi-classifiers. I can also explore replacing the classical feed forward network with convolutional network and with quantum convolutional network [16]. Quantum convolutional neural networks can be instrumental in the areas of computer vision and image processing. We are truly in an era of exciting new research into fine-tuning or developing meaningful new quantum algorithms.

CODES

The codes for this paper can be found on my github page:
Auto-encoder
Binary classifier
MNIST multi-classifier

ACKNOWLEDGMENTS

I appreciate guidance from Dr. Marek Perkowski at Portland State University. For discussions about quantum neural networks and photonic quantum computers, I appreciate Maria Schuld and those at Xanadu. Martha Himel helped me with editing and I appreciate her input.

References

[1] Scott Aaronson and Alex Arkhipov, (2011), *The computational complexity of linear optics*, Proceedings of the 43rd annual ACM Symposium on Theory of Computing.

[2] Scott Aaronson and Lijie Chen, (2017), *Complexity-Theoretic Foundations of Quantum Supremacy Experiments*, Proceedings of the 32nd Computational Complexity Conference, No. 22, Pages 1–67.

[3] C. Adami, N. Cerf, (1999), *Quantum Computation with Linear Optics*, Quantum Computing and Quantum Communications. Lecture Notes in Computer Science, vol 1509. Springer.

[4] J. Arrazola et al., (2021), *Quantum circuits with many photons on a programmable nanophotonic chip*, Nature volume 591, pages 54–60.

[5] Frank Arute, Kunal Arya, and John M. Martinis, (2019), *Quantum supremacy using a programmable superconducting processor*, Nature volume 574, pages 505–510.

[6] Stephan Attal, *Lectures in Quantum Noise Theory*, http://math.univ-lyon1.fr/homes-www/attal/chapters.html.

[7] Ville Bergholm et al., (2018), *PennyLane: Automatic differentiation of hybrid quantum-classical computations*, arXiv:1811.04968

[8] Jacob Biamonte, Peter Wittek, Nicola Pancotti, P. Rebentrost, Nathan Wiebe, Seth Lloyd, (2017), *Quantum Machine Learning*, Nature 549.

[9] Samuel Braunstein and Peter van Loock, (2005), *Quantum information with continuous variables*, Reviews of Modern Physics 77, 513.

[10] William Case, (2008), *Wigner functions and Weyl functions for pedestrians*, American Journal of Physics 76, 937.

[11] Siddhartha Das, George Siopsis, and Christian Weedbrook, (2018), *Continuous-variable quantum Gaussian process regression and quantum singular value decomposition of non-sparse low rank matrices*, Physical Review A 97, 022315

[12] Abhinav Deshpande et al., (2021), *Quantum computational supremacy using Gaussian boson sampling*, arXiv:2102.12474.

[13] Edward Farhi and Hartmut Neven, (2018), *Classification with Quantum Neural Networks on Near Term Processors*, arXiv:1802.06002v2.

[14] Alessandro Ferraro, Stefano Olivares, Matteo G. A. Paris, (2005), *Gaussian states in continuous variable quantum information*, Bibliopolis.

[15] Craig Hamilton, Regina Kruse, Linda Sansoni, Sonja Barkhofen, Christine Silberhorn, Igor Jex, (2017), *Gaussian boson sampling*, Physic Review Letters.

[16] Nathan Killoran, Thomas Bromley, Juan Miguel Arrazola, Maria Schuld, Nicolás Quesada, and Seth Lloyd, (2019), *Continuous variable quantum neural networks*, Physical Review Research 1, 033063.

[17] E. Knill, R. Laflamme, and G. J. Milburn, (2001), *A scheme for efficient quantum computation with linear optics*, Nature volume 409, pages 46–52.

[18] P. Kok and B. W. Lovett, (2010), *Introduction to Optical Quantum Information Processing*, Cambridge University Press.

[19] Dave Krebs, (2007), *Introduction to Kernel Methods*, cs.pitt.edu.

[20] Seth Lloyd and Samuel Braunstein, (1999), *Quantum computation over continuous variables*, Physics Review Letters 82, 1784.

[21] Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac, and Nathan Killoran, (2020), *Quantum embeddings for machine learning*, arXiv:2001.03622.

[22] M. Menotti, B. Morrison, K. Tan, Z. Vernon, J.E. Sipe, and M. Liscidini, (2018), *Nonlinear coupling of linearly uncoupled resonators*, Physics Review Letters.

[23] Jonas Neergaard-Nielsen, (2008), *Generation of single photons and Schrodinger kitten states of light*, PhD Thesis, Technical University of Denmark.

[24] Michael Nielsen and Isaac Chuang, (2010), *Quantum computation and quantum information*, Cambridge University Press.

[25] K. Parthasarathy, (2010), *What is a Gaussian State?*, Communications of Statistical Analysis, Vol 4, Num 2.

[26] Anatoli Polkovnikov, (2010), *Phase space representation of quantum dynamics*, Annals of Physics.

[27] J. Preskill, (2018), *Quantum Computing in the NISQ era and beyond*, Quantum, 2:79.

[28] Patrick Rebentrost et al, (2019), *Quantum gradient descent and Newton's method for constrained polynomial optimization*, New Journal of Physics 21 073023.

[29] Maria Schuld, (2021), *Supervised quantum machine learning models are kernel methods*, arXiv:2101.11020v2.

[30] Maria Schuld, Ville Bergholm, Christian Gogolm, Josh Izaac, and Nathan Killoran, (2018), *Evaluating analytic gradients on quantum hardware*, Physical Review A.

[31] Maria Schuld, Alex Bocharou, Krista Svore, and Nathan Wiebe, (2018), *Circuit-centric quantum classifiers*, Physical Review A.

[32] Maria Schuld, Mark Fingerhuth, and Francesco Petruccione, (2017), *Implementing a distance-based classifier with a quantum interference circuit*, Europhysics Letters, Volume 119, Number 6.

[33] Maria Schuld and Nathan Killoran, (2019), *Quantum machine learning in feature Hilbert spaces*, Physical Review Letters 122, 040504.

[34] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione, (2016), *Prediction by linear regression on a quantum computer*, Physical Review A 94, 022342.

[35] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer, (2021), *Effect of data encoding on the expressive power of variational quantum machine learning models*, Physical Review A 103, 032430.

[36] Yichen Shen, Nicholas C. Harris, Scott Skirlo, Mihika Prabhu, Tom Baehr-Jones, Michael Hochberg, Xin Sun, Shijie Zhao, Hugo Larochelle, Dirk Englund, and Marin Soljačić, (2017), *Deep learning with coherent nanophotonic circuits*, Nature Photonics.

[37] Daiqin Su, Ish Dhand, Lukas G. Helt, Zachary Vernon, and Kamil Brádler, (2019), *Hybrid spatiotemporal architectures for universal linear optics*, Physical Review A 99, 062301.

[38] Alexander Tait, Mitchell Nahmias, Bhavin Shastri, and Paul Prucnal, (2014), *Broadcast and weight: an integrated network for scalable photonic spike processing*, Journal of Lightwave Technology.

[39] K.Tan, M. Menotti, Z. Vernon, J. E. Sipe, M. Liscidini, and B. Morrison1, (2019), *Stimulated four-wave mixing in linearly uncoupled resonators*, Optics Letters.

[40] Ilan Tzitrin, Takaya Matsuura, Rafael Alexander, Guillaume Dauphinais, J. Eli Bourassa, Krishna Sabapathy, Nicolas Menicucci, and Ish Dhand, (2021), *Fault-tolerant quantum computation with static linear optics*, PRX Quantum, Physics Review Journal.

[41] V. Vaidya et al., (2019), *Broadband quadrature-squeezed vacuum and nonclassical photon number correlation from a nanophotonic device*, Science Advances.

[42] Christian Weedbrook, Stefano Pirandola, Raúl García-Patrón, Nicolas J. Cerf, Timothy C. Ralph, Jeffrey H. Shapiro, and Seth Lloyd, (2012), *Gaussian Quantum Information*, Reviews of Modern Physics 84, 621.

[43] Peter Wittek, (2014), *Quantum Machine Learning: What Quantum Computing Means to Data Mining*, Elsevier Insights, Academic Press.

[44] Y. Zhang et al., (2021), *Squeezed light from a nanophotonic molecule*, Nature Communications.

[45] Han-Sen Zhong et al., (2020), *Quantum computational advantage using photons*, Science 10.1126.

[46] (2019), *Quantum Computing: Progress and Prospects*, Consensus Study Report, The National Academic Press.

[47] 18.06SC Linear Algebra Fall, (2011), *Singular Value Decomposition*, MIT OpenCourseWare