QuClassi: A Hybrid Deep Neural Network Architecture based on Quantum State Fidelity

Samuel A. Stein [1][2] **Betis Baheri** [3] **Daniel Chen** [4]
Ying Mao [1] **Qiang Guan** [3] **Shuai Xu** [4] **Caiwen Ding** [5] **Ang Li** [2]

ABSTRACT

In the past decade, remarkable progress has been achieved in deep learning related systems and applications. In the post Moore's Law era, however, the limit of semiconductor fabrication technology along with the increasing data size has slowed down the development of learning algorithms. In parallel, the rapid development of quantum computing has pushed it into a new era. Google illustrated quantum supremacy by completing a specific task (random sampling problem), in 200 seconds, which continues to be impracticable for the largest classical computers. Due to the exponential potential of quantum computing, quantum based learning is an area of interest, in hopes that certain systems might offer a quantum speedup. In this work, we propose a novel architecture QuClassi, a quantum neural network for both binary and multi-class classification. Powered by a quantum differentiation function along with a hybrid quantum-classic design, QuClassi encodes the data with a reduced number of qubits and generates the quantum circuit, pushing it to the quantum platform for the best states, iteratively. We conduct intensive experiments on both quantum simulators, IBM-Q's quantum platform as well as evaluate performance on IonQ which was accessed through Microsoft's Azure Quantum Platform. The evaluation results demonstrate that QuClassi is able to outperform the state-of-the-art quantum-based solutions, Tensorflow-Quantum and QuantumFlow by up to 53.75% and 203.00% for binary and multi-class classifications. When comparing to traditional deep neural networks, QuClassi achieves a comparable performance with 97.37% fewer parameters.

1 INTRODUCTION

Deep learning (DL) has drawn tremendous interest in the past decade from industry and academia (Goodfellow et al., 2016; Devlin et al., 2019; LeCun et al., 2015; He et al., 2016). Novel DL algorithms, growing computational power and modern architectural designs have enabled a wide spectrum of applications ranging from scientific data processing(Mustafa et al., 2019; Wozniak et al., 2018) to image and speech recognition (Krizhevsky et al., 2012; Hinton et al., 2012; Amodei et al., 2016). Despite the widespread practical success, the proximity to the physical bound of semiconductor fabrication in post Moore's Law era along with the increasing size of data sets raises the discussion on the future of DL and its limitations (Waldrop, 2016).

In parallel with the breakthrough of DL in the past years, remarkable progress has been achieved in the field of quantum computing. In 2019, Google demonstrated Quantum Supremacy using a 53-qubit quantum computer, where it spent 200 seconds to complete a random sampling task that would cost 10,000 years on the largest classical computer (sup). During this time, quantum computing has become increasingly available to the public. IBM Q Experience, launched in 2016, offers quantum developers to experience the state-of-the-art superconducting quantum computers (ibm). Early 2020, Amazon Braket (aws) provides the general access to a development environment to help customers explore and design quantum algorithms. Microsoft Quantum Development Kit (QDK) (Microsoft) bridges the gap between quantum computing and classic computing, and provides private access to trapped-ion machines by IonQ and Honeywell.

Quantum computing is a computational paradigm that harnesses quantum phenomena and the computational advantages that quantum computers offer (qua). A traditional computer with von Neumann architecture completes tasks

---

[1]Department of Computer and Information Science, Fordham University, New York, USA [2]Pacific Northwest National Laboratory, Richland, WA, USA [3]Department of Computer Science, Kent State University, Kent, OH, USA [4]Department of Computer and Data Sciences, Case Western Reserve University, Cleveland, OH, USA [5]Department of Computer Science & Engineering, University of Connecticut, Storrs, CT, USA. Correspondence to: Samuel A. Stein <samuel.stein@pnnl.gov>, Ying Mao <ymao41@fordham.edu>.

by manipulating bits that can be set to 0 or 1. Everything from tweets and emails to electronic tax returns and healthcare records are essentially long strings of these binary digits. A quantum computer, however, utilizes quantum bits, or qubits, to encode information. Qubits exhibit quantum properties, which means a connected group of them have exponentially more expressive and processing power than the same number of binary bits. Two fundamental properties that enable this phenomena are superposition and entanglement.

A qubit, similar to bits, has two basis states $|0\rangle$ and $|1\rangle$. However, the value of quantum computing comes from the fact that a qubit can be in a superposition of $|0\rangle$ and $|1\rangle$ at the same time. As for entanglement, entangled qubits can be generated by specific interactions within a quantum system. Qubits that are entangled can not be described independently and only as a single quantum state. For a specific pair, changing the state of either qubit will affect the state of the other qubit predictably. This property in quantum physics is known as entanglement.

Due to the great potential of processing complex problems beyond current abilities at a fraction of the time, quantum based learning systems have received great attention recently. The authors in (Schuld et al., 2015) provide a systematic overview of the emerging field of quantum machine learning. Many researchers try to find quantum algorithms that can take the place of classical machine learning algorithms, and show an improvement in terms of a computational complexity reduction (Wiebe et al., 2012; Zhao et al., 2019). Widely used classic machine learning algorithms such as nearest neighbor, the kernel method and other clustering algorithms, which are comprised of expensive distance calculations have been proposed to be accelerated by the design of a quantum counterpart (Wiebe et al., 2015). Recent innovations (Aïmeur et al., 2007; Rebentrost et al., 2014; Casaña-Eslava et al., 2020; Chen et al., 2020b; Xue et al., 2021; Chen et al., 2021; Li et al., 2021; Stein et al., 2021b;a; Chen et al., 2020a) in the field of quantum machine learning for classification mainly focus on solving binary classification problems. With deep quantum neural network architectures and data pooling, many solutions proposed act and perform very similarly to their classical counterparts. Although useful, the binary classification setting significantly limits the scope of the solutions. Furthermore, many of the proposed approaches are still suffering from relatively low accuracy, missing a comprehensive quantum architecture that performs multi-class classification, or requiring an infeasible number of qubits for data encoding and high parameter counts.

In this work, we propose QuClassi, an architecture for high-count multi-class classification with a limited number of qubits. Making use of a quantum state fidelity cost func-

tion, QuClassi utilises a data encoding architecture encoding two dimensions of data per one qubit. QuClassi provides leading classification accuracies within the quantum DL field on the MNIST (mni) and Iris (iri) datasets. Moreover, QuClassi has been evaluated with binary and multi-class experiments on both local clusters, IBM-Q (ibm) (Superconducting Quantum Processor (QPU)) and IonQ (Trapped Ion QPU). We summarize the key contributions below.

- We introduce a quantum state fidelity based cost function that enables the stable training of deep quantum circuits for classification.

- We propose QuClassi, a quantum-classical architecture with three different layer designs. The QuClassi works for both binary and multi-class classification problems.

- We evaluate QuClassi with well-known datasets, performing a complete both binary and multi-class classification. QuClassi is evaluated with experiments on both the simulator as well as a real quantum platform, IBM-Q. When comparing with Tensorflow, Tensorflow-Quantum and the state-of-the-art Quantum-Flow (Jiang et al., 2020), we achieve accuracy improvements by up to 53.75% on binary classification and 203.00% on multi-class classification. Comparing with classical deep neural networks with similar performance, QuClassi is able to reduce 97.37% of the parameters.

The rest of this paper is organized as follows. In Section 3, we introduce the background of quantum computing and related techniques that are utilized in this work. We present the system design of QuClassi in Section 4 and, in Section 5, we discuss the results from intensive experiments on local quantum simulator environment and IBM-Q platform. We conclude the paper discussion with Section 6. Acknowledgements (See Sec 7) and Appendix follow.

2 RELATED WORK

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 (Garg & Ramakrishnan, 2020), authors conduct a comparative study of classic DL 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 (Cortese & Braje, 2018; Zoufal et al., 2019; Lloyd &

Weedbrook, 2018). Cortese et al. (Cortese & Braje, 2018) discusses 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 (Zoufal et al., 2019) utilizes quantum Generative Adversarial Networks to facilitate efficient learning and loading of generic probability distributions, which are implicitly are given by data samples, into quantum computers.

As quantum computers quickly evolve, quantum neural networks quickly gained attention in the field. A neural network architecture for quantum data has been proposed in (Farhi & Neven, 2018), which utilizes the predefined threshold to build a binary classifier for bit strings. Focusing on optimizing variational quantum circuits, Stokes et al. (Stokes et al., 2020) introduce a quantum generalization of natural gradient for general unitary and noise-free quantum circuits. A design of quantum convolutional neural network is proposed in (Cong et al., 2019) 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 (Beer et al., 2020) 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 (Jiang et al., 2020) proposes a co-design framework that consists of a quantum-friendly neural network and an automatic tool to generate the quantum circuit. However, it utilizes quantum computers by training the network locally and then mapped to quantum circuits. This design is easy to implement, but shows significant sensitivity to real quantum computer noise. In addition, it is still based on a loss function based on classical data.

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 accuracies, missing a comprehensive quantum architecture, and requiring an infeasible number of qubits. We propose `QuClassi` that employs a quantum state fidelity based loss function and a quantum-classic hybrid architecture to address the current limitations.

3 QUANTUM COMPUTING BASICS

3.1 Quantum Bits - Qubits

Traditional computers are built on the premise of information being represented as either a 0 or a 1. In contrast,

quantum computers represent data through the use of a 1, 0, or both 1 and 0 simultaneously. This ability to represent a 1 and 0 simultaneously is a quantum mechanical phenomena called superposition, that is at the core of quantum computers computational potential. These quantum bits, namely qubits, are the fundamental building block of quantum computers. The qubits are represented in the form of linear vector combinations of both ground states $|0\rangle$ and $1\rangle$. Quantum systems are identified through the $\langle bra||ket\rangle$ notation, where $\langle bra|$ is a horizontal state vector and $|ket\rangle$ is a vertical state vector. A qubit can be mathematically represented with Equation 1:

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

where $|\psi\rangle$ is the qubit itself, and $|0\rangle$ and $|1\rangle$ represent the "0" and "1" translation to quantum systems that the qubit is a linear combination of. The $|0\rangle$ and $|1\rangle$ represent two orthonormal eigen vectors, of which $|\psi\rangle$ is a probabilistic combination. These ground state vectors are described in Equation 2:

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

With superposition, there are multiple new possibilities for data representation on quantum system, where instead of classical binary data encoding, data could be represented in the form of a qubits probability distribution, or expectation. Most importantly, the data representation architecture of quantum computers opens up a wide range of possible data encoding techniques. Taking the tensor product between the qubits described in Equations 3 and 1 in 4 describes the state of two qubits.

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

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

The coefficients to each respective quantum state described in Equation 4, and all quantum state equations in general, describe the probability of measuring each respective outcome. Therefore, a constraint is introduced that the squared sum of the coefficients is equal to 1. A qubit, when measured, will measure as either of the $|0\rangle$ or $|1\rangle$. These probabilities indicate the likelihood of measuring a $|0\rangle$ or $|1\rangle$. The measurement of $|0\rangle$ and $|1\rangle$ is arbitrary, and is simply one "direction" of measurement. The only constraint on direction of measurement is that the outcome of the measurement is one of two orthogonal eigen vectors. The bloch sphere is a common way of representing a qubit, which can be thought of as a unit sphere, and the quantum state is a point on this

sphere's surface. The $|0\rangle$ and $|1\rangle$ vectors indicate the outer sphere points on the Z axis. However, one could measure a qubit to be on the X axis, or the Y axis, or any axis described by two rotations, touching two opposite ended points on the unit sphere. For the purposes of this paper, we will only be measuring against the Z-axis resulting in measurements of $|0\rangle$ and $|1\rangle$. Quantum states are responsible for encoding data, and to perform operations on quantum states quantum gates are used. Quantum gates apply a transformation over a quantum state into some new quantum state.

3.2 Quantum Gates and Operations

3.2.1 Single qubit operations

A quantum computers controlling apparatus allows the application of a single-qubit rotation $R(\theta, \phi)$ to any single qubit by unitary operator described by Equation 5:

$$R(\theta, \phi) = \begin{bmatrix} \cos\frac{\theta}{2} & -ie^{-i\phi}\sin\frac{\theta}{2} \\ -ie^{i\phi}\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \quad (5)$$

Both θ and ϕ can be controlled by changing the duration and phase of the R that drives the Rabi oscillation of the qubit(Debnath et al., 2016).

3.2.2 RX, RY, RZ rotation

The single-qubit rotations around the basis axes must be expressed in terms of general R rotation defined in Equation 5. We can separate each rotation in respect to rotations around the X, Y and Z axis. **RX** can be defined by setting $\phi = 0$ in $R(\theta, \phi)$ the rotation about the X axis by the angle θ, as described in Equation 6:

$$R_X(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} = R(\theta, 0) \quad (6)$$

Similarly **RY** is defined by setting $\phi = \frac{\pi}{2}$ in $R(\theta, \phi)$ obtaining the rotation about Y axis by the angle θ described in Equation 7:

$$R_Y(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} = R(\theta, \frac{\pi}{2}) \quad (7)$$

Finally, **RZ** rotation is defined in Equation 8:

$$R_Z(\theta) = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix} \quad (8)$$

3.2.3 Two-qubit gates and Controlled Operation

Quantum gates can operate on more than one qubit simultaneously. The basic Rotation operation forming a single qubit rotation can be expanded to R_{XX}, R_{YY} and R_{ZZ} rotations. $R_{XX}(\theta)$ are defined in Equations 9 - 11:

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

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

where $R_{ZZ}(\theta)$ is defined such that

$$R_{ZZ}(\theta) = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 & 0 & 0 \\ 0 & e^{-i\frac{\theta}{2}} & 0 & 0 \\ 0 & 0 & e^{-i\frac{\theta}{2}} & 0 \\ 0 & 0 & 0 & e^{-i\frac{\theta}{2}} \end{bmatrix} \quad (11)$$

Another key operation in data manipulation is the Controlled-SWAP (CSWAP) gate, allowing for Quantum entanglement, which can be defined as:

$$CSWAP(q_0, q_1, q_2) = |0\rangle\langle 0| \otimes I \otimes I + |1\rangle\langle 1| \otimes \text{SWAP}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

where q_0, q_1, q_2 are qubits entangled to each other. We further make use of Controlled rotations, which are extensions on the RY, RZ and RX gates however with a control operation appended to them.

3.3 Quantum Entanglement

Along with superposition, quantum entanglement is another phenomenon of quantum computing that exposes significant computing potential. Quantum entanglement is described as when two quantum states interact in a certain way, following the interaction the quantum states can not be described independently of each other. This generally implies that the outcome of one qubits measurement implies information about the other. A common experiment is Bell's experiment, which is when two qubits that have been entangled through a CNOT gate, allows for the measurement of one qubit to, with absolute certainty, predict the measurement of the second qubit. As a concrete idea, if qubit $|\phi\rangle$, entangled with qubit $|\psi\rangle$, measures as a $|0\rangle$, $\psi\rangle$ will measure as $|0\rangle$, and vice versa. This principle of quantum entanglement occurs when a controlled gate is used, as described in Equation

3.2.3, and with gates such as controlled rotations, where a control qubit dictates an entangled change in state on the other qubits.

A key use case of entanglement is the SWAP test, a quantum state fidelity measurement algorithm that operates off of quantum entanglement. The SWAP test is a probabilistic distance metric that measures the difference between two quantum states, supposedly $|\phi\rangle$ and $|\omega\rangle$. The SWAP test takes in the two states, performs a CSWAP over the states controlled by an anicilla qubit. This qubit, prior to the SWAP test, is placed in superposition by a Hadamard gate from state $|0\rangle$. After the CSWAP, we apply another Hadamard gate to the anicilla qubit, and measure the qubit. The probability of the qubit measuring 1 ranges from 0.5 to 1. If the states are orthogonal, the qubit will measure 1 approximately 50% of the time, and as the quantum state fidelity increases as does the probability of measuring a $|1\rangle$ increases, eventually at 100%.

4 QUCLASSI: QUANTUM-BASED CLASSIFICATION

4.1 System Design

Our architecture operates through a feedback loop between a classical computer and a quantum computer, illustrated in Fig. 1. Data is initially cleaned, removing any significant outliers or any other necessary data cleaning steps. Data is parsed from classical data into quantum data through a quantum data encoding method outlined in 4.2, and is now in the form of a quantum data set represented by quantum state preparation parameters. For each predictable class in the data set, a quantum state is initialized with the same qubit count as the number of qubits in the classical quantum data set, due to the constraints of the SWAP test. The quantum states along with quantum classical data are sent to a quantum computer.

This initialization of state is the core architecture to QuClassi. In this, a quantum circuit of a certain number of layers representing a quantum deep neural network is prepared with randomly initialized parameters containing a certain number of qubits. The produced quantum state of this circuit is to be SWAP tested against the quantum data point, which is fed back to the classical computer, forming the overall quantum DL architecture of QuClassi.

The quantum computer calculates the quantum fidelity from one anicilla qubit which is used to calculate model loss, and sends this metric back to the classical computer. The classical computer uses this information to update the learnable parameters in attempts to minimize the cost function. This procedure of loading quantum states, measuring state fidelity, updating states to minimize cost is iterated upon until the desired convergence or sufficient epochs have been
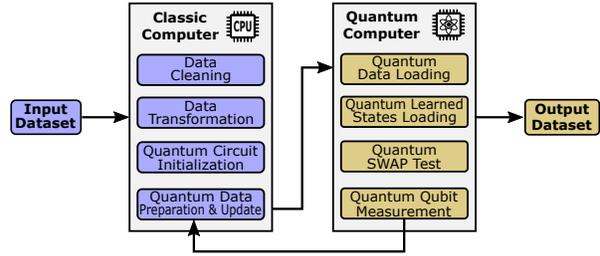
completed.

Figure 1: Quantum-Classic Deep Learning.

4.2 Data Qubitization

When evaluating quantum machine learning architectures on classical datasets, we need a way to translate classical data into quantum states. An important question is how one might represent a classical data set in the quantum setting. Our architecture makes use of translating some traditional numerical data point into the expectation of a qubit. To accomplish this, data $x_1, x_2, ..., x_n$ of dimension d can be translated onto a quantum setting by normalizing each dimension d_i to be bound between 0 and 1 due to the range of a qubits expectation. Encoding a single dimension data point only requires one qubit, unlike classical computing which requires a string of bits to represent the same number. To translate the traditional value x_i into some quantum state, we perform a rotation around the Y axis parameterized by the following equation: $RY(\theta_{x_i}) = 2sin^{-1}(\sqrt{x_i})$. This $RY(\theta_{x_i})$ operation results in the expectation of a qubit being measured against the Z axis equating to the x_i value from the classical data the qubit encodes. An extension on this idea is that we can encode our second dimension of data across the X-Y plane. 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 the required rotations across $\frac{d}{2}$ qubits, each rotation 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 possible method of alleviating high qubit counts and evaluate the performance if we encode each dimension of data into one respective qubit solely through a RY Gate. This approach is validated by the fact that we never measure any of our qubits, and only their quantum fidelity through the SWAP test, thereby bypassing the superposition-collapsing issue of this approach. We encode the second dimension of data on the same qubit through the following rotation:

$$RZ(\theta_{x_{i+1}}) = 2sin^{-1}(\sqrt{x_i}) \tag{12}$$

When number of qubits is a primary concern, methods that

will reduce this number are extremely valuable. Classical data encoding in quantum states has no method that is tried and tested unlike classical computers with formats such as integers and floats, therefore our approach can certainly be critiqued; however the approach was tested and proved to be a viable approach to the problem. Furthermore, knowing both the expectation of a qubit across the Y and Z domain allows for the data classical data to be reconstructed. Alternative methods exist for classical-to-quantum data encoding, where on one extreme, one can encode 2^n classical data points across n qubits using state-vector encoding. However, this method is extremely susceptible to noise. On the other end of the spectra, one can encode classical data into a binary representation on quantum states, by translating a vector of binary values onto qubits. This encodes n data points onto n qubits, and loses a lot of information in the process. However, is not as susceptible to noise and exponential-sampling problems. Exponential data-encoding methods exist however, and can be fitted with `QuClassi` due to `QuClassi` never directly performing quantum state tomography, hence the data encoding section is scalable.

4.3 Quantum Layers

The design of our quantum neural network can be broken down into a sequential layer architecture. Each layer proposed in this paper is designed with the intent of each layer having a specific type of quantum state manipulation available to it. We introduce three layer styles, namely single qubit unitaries, dual qubit unitaries, and controlled qubit unitary. Single qubit unitaries involve rotating a single qubit by a set of parameters θ_d. A qubit enters a single qubit
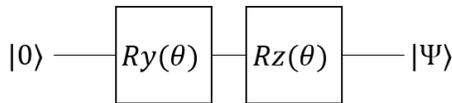
Figure 2: Single Qubit Unitary.

unitary, drawn in Fig. 2, in a certain state and undergoes a rotation around the Y and Z axis. A rotation around the Z and Y axis allows for complete manipulation of a single qubits quantum state.

As for layers comprised of gates operating on more than one qubit simultaneously, we make use of the Dual Qubit unitary and Entanglement unitary. With a dual qubit unitary, drawn in Fig.3, two qubits enter in their each respective gate, followed by an equal Y and Z rotation on both qubits. The rotation performed on both qubits are equal. Entanglement layers, drawn in Fig.4, involve two qubits being entangled through two controlled operations per qubit pair. We introduce the use of $CRY(\theta)$ and $CRZ(\theta)$ gates within this layer, thereby allowing for a learn-able level of entan-

glement between qubits. In the above layer design, two qubits are passed in with one qubit being a control qubit and the other the target qubit. With respect to a CRY gate, 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.
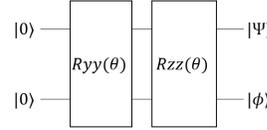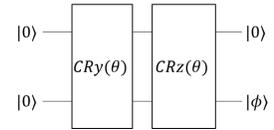
Figure 3: Dual Qubit.

Figure 4: Entanglement.

Each of the aforementioned layers contains rotations parameterized by a group of parameters, which are the trainable parameters in our neural network. We can combine a group of each of the unitaries on a quantum state to form a quantum layer, parameterized by each independent θ in the layer, where the layer is set as a group of these operations, as visualized in Figure 5.
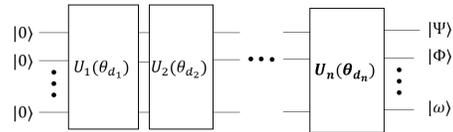
Figure 5: Layered Quantum Deep Learning Architecture.

The use of these layers, drawn in Fig. 5, allows for the abstraction to the design that is similar to the traditional DL. 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.

4.4 State fidelity based Cost Function

When training a neural network to accomplish a task, an explicit description of system improvement goal needs to be established - i.e the cost function. The quantum machine learning cost function landscape can be slightly ambiguous compared to classical machine learning, as we could be manipulating the expected values of each qubit in some way, however even this is ambiguous - the direction being measured in heavily affects the expectation value and or what our iteration count would be for measuring expectation, with lower iterations leading to increasingly noisy outputs. Within our system, we make use of the SWAP test to parse quantum state fidelity to an appropriate cost function. One of the benefits of the SWAP test is that we only need to measure one anicilla qubit.In the case of binary classification, each data point is represented in a quantum

state represented by $|\phi\rangle$, which is used to train the quantum state prepared by our DL model $|\omega\rangle$ such that the state of $|\omega\rangle$ minimizes some cost function. The classical cross-entropy cost function outlined in Equation 14 is an appropriate measure for state fidelity, as we want the fidelity returned to be maximized in the case of Class=1, and minimized otherwise.

$$min(Cost(\theta_d, X) = \frac{1}{n}\sum_{i=1}^{n} SWAP(|\phi_{X(i)}\rangle, |\omega\rangle) \quad (13)$$

$$Cost = -ylog(p) - (1-y)log(1-p) \quad (14)$$

Where θ_d is a collection of parameters defining a circuit, x is the data set, $\phi_{x(i)}$ is the quantum state representation of data point i, and ω is the state being trained to minimize the function in Equation 13 and 14.

Optimization of the parameters θ_d requires us to perform gradient descent on our cost function. We make use of the following modified parameterized quantum gate differentiation formula outlined in Equation 15.

$$\frac{\delta Cost}{\delta\theta_i} = \frac{1}{2}(f(\theta_i + \frac{\pi}{2\sqrt{\epsilon}}) - f(\theta_i - \frac{\pi}{2\sqrt{\epsilon}})) \quad (15)$$

Where in Equation 15 θ_i is a parameter, Cost is the cost function, and ϵ is the epoch number of training the circuit. Our addition of the ϵ is targeted at allowing for a change in search-breadth of the cost landscape, shrinking constantly ensuring a local-minima is found.

4.5 Training and Inducing QuClassi

We implement the Algorithm outlined as Algorithm 1 for the training of our Quantum DL architecture.

Algorithm 1 describes how `QuClassi` works. Firstly, we load the dataset as presented in 4.2 (Line 1). Line 2-3 introduce certain terms which are chosen by the practitioner. The learning rate is a variable chosen at run time that determines at what rate we want to update our learned weights at each iteration. Qubit Channels indicate the number of qubits that will be involved in the quantum system. Epochs indicate how many times we will train our quantum network on the dataset. Within the nested loops, Lines 7-23, 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}). These lines 7-23 accomplish the specific parameter tuning of our network, where $f(\theta_d)$ is the overall cost function of the network. At induction time, the quantum network is induced across all trained classes and the fidelity is softmaxed. The highest probability returned is the classified class.

Algorithm 1 QuClassi Algorithm

1: Data set Loading
    Dataset : $(X|Class = mixed)$
2: Distribute Dataset X By Class
3: Parameter Initialization:
    Learning Rate : $\alpha = 0.01$
    Network Weights : $\theta_d = $ [Rand Num btwn 0 - $1\times\pi$]
    epochs : $\epsilon = 25$
    Dataset : $(X|Class = \omega)$
    Qubit Channels : $Q = n_{X_{dim}} \times 2$
4: **for** $\zeta \in \epsilon$ **do**
5:     **for** $x_k \in X$ **do**
6:        **for** $\theta \in \theta_d$ **do**
7:           Perform Hadamard Gate on Q_0
8:           Load $x_k \xrightarrow[Data\ Encoding]{Quantum} Q_{Q_1 \to \frac{Q_{Count}}{2}+1}$
9:           Load $\theta_d \xrightarrow[Learned\ State]{Quantum} Q_{\frac{Q_{Count}}{2}+1 \to Q_{Count}}$
10:         Add $\frac{\pi}{2\sqrt{\epsilon}} \to \theta$
11:         $\Delta_{fwd} = (\mathbf{E}_{Q_0}f(\theta_d)$
12:         CSWAP(Control Qubit = Q_0, Learned State Qubit, Data Qubit)
13:         Measure Q_0
14:         Reset Q_0 to $|0\rangle$
15:         Perform Hadamard Gate on Q_0
16:         Subtract $\frac{\pi}{2\sqrt{\epsilon}} \to \theta$
17:         CSWAP(Control Qubit = Q_0, Learned State Qubit, Data Qubit)
18:         Measure Q_0
19:         $\Delta_{bck} = (\mathbf{E}_{Q_0}f(\theta_d)$
20:         $\theta = \theta - (-log(\frac{1}{2}\Delta_{fwd} - \Delta_{bck})) \times \alpha$
21:        **end for**
22:     **end for**
23: **end for**

---

5 EVALUATION

5.1 System implementation and Experiment settings

We implement `QuClassi` with Python 3.8 and IBM Qiskit Quantum Computing simulator package. The circuits were trained on a single-GPU machine (RTX 2070). 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. On the figures, we use `QC-S`, `QC-D`, `QC-E`, `QC-SDE` to represent them.

- `QC-S`: A layer that performs one single qubit unitary per qubit is namely a Single Qubit Unitary Layer.

- `QC-D`: A layer that performs a dual Qubit unitary per qubit combination is a Dual Qubit Unitary Layer.
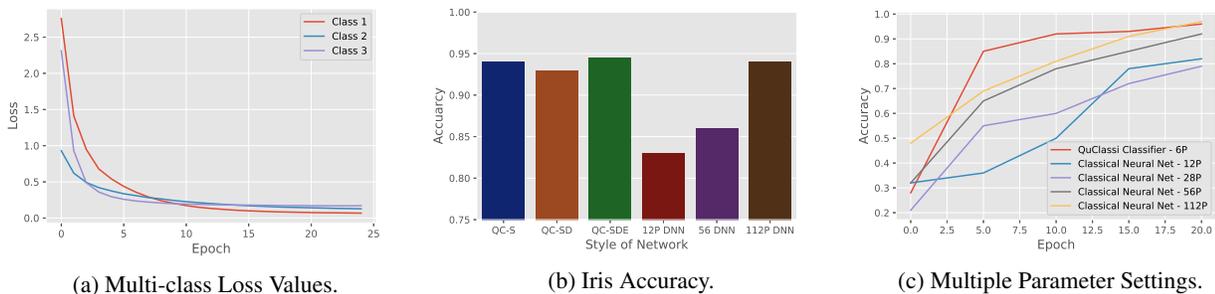
(a) Multi-class Loss Values.

(b) Iris Accuracy.

(c) Multiple Parameter Settings.

Figure 6: Iris Dataset

- `QC-E`: A layer that entangles qubits through controlled rotations is an Entanglement Layer.

- `QC-SD`, `QC-SDE`: A `QuClassi` that consists of multiple layers.

In the evaluation, we utilize Tensorflow, a neural network program framework by Google and Tensorflow Quantum, a quantum programming framework of traditional Tensorflow.

When comparing to traditional neural network methods, our primary interest is to design comparable networks with 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, DNN-12 means a deep neural network with 12 parameters. The network is trained using the same learning rate as `QuClassi`, is designed to have one hidden layer, with an output SoftMax layer. The optimizer utilized is a Stochastic Gradient Descent optimizer, the same that is used for `QuClassi`. Finally, the dataset fed to the classical neural network is the same dataset fed to the quantum neural networks, i.e. normalized data points between 0 and 1, post data-processing techniques.

Furthermore, we compare `QuClassi` with two quantum-based solutions: **Tensorflow-Quantum (TFQ)** and **QuantumFlow** (Jiang et al., 2020). Tensorflow Quatum provides its classification implementation in (Google), however, it only works for binary classifications. QuantumFlow, a state-of-the-art solution, works for both binary and multi-class classifications. In the comparison, we use the results from **QF-pNET**, the version that obtains the best results of QuantumFlow.

5.2 Quantum Classification: Iris Dataset

A common data set that machine learning approaches are tested on for a proof of concept is the Iris dataset (iri). This dataset comprises of 3 classes, namely Setosa, Versicolour and Virginica , and a mix of 150 data points, each contain-ing 4 numeric data points about one of the flowers. We implement QuClassi on this data set as it provides the proof of concept for a quantum multi-class classification architecture. For this dataset to be encoded in quantum states, we perform the quantum data encoding described in 4.2. The data is encoded in the quantum state by performing an RY gate followed by a RZ gate, encoding 2 dimensions on one qubit.
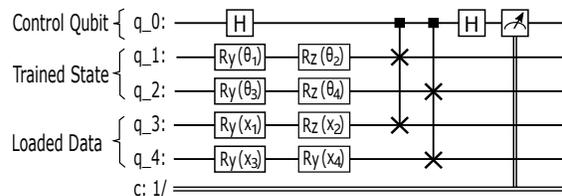

Figure 7: Sample Circuit.

To classify all three classes, there are three target qubit states $|Setosa\rangle$ (Class_1), $|Versicolour\rangle$ (Class_2) and $|Virginica\rangle$ (Class_3). The dataset is separated into its respective classes, and used to train its respective quantum state. We use the methodology outlined in 4.5 for training these 3 quantum states. 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. 7, where the learned 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 Class_1, and has to be run with Class_2 discriminator state loaded into qubits 1 and 2, and then Class_3 with the same methodology. These probabilities are softmaxed which is then used to make a decision. 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. 6a.

Fig. 6b and 6c plot the results of our experiments in different settings. As can be seen, the quantum neural network architectures converge extremely quickly, and with a rela-

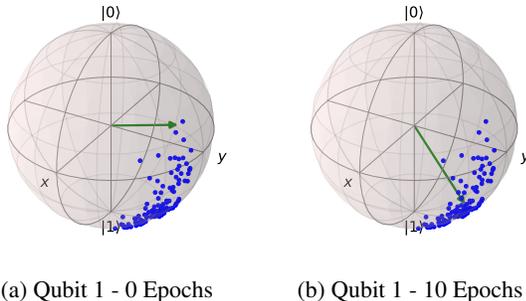(a) Qubit 1 - 0 Epochs   (b) Qubit 1 - 10 Epochs

Figure 8: Identify 0 (Epoch 1 vs 10).

tively high accuracy. Similarly parameterized classical deep neural networks perform below that of their quantum counterparts. We analyze and compare our network to classical deep 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 Fig. 6c, where multiple classical deep neural networks of varying parameter counts were compared with the our architecture. 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.

5.3 Quantum Classification: MNIST dataset

Although the Iris dataset was able to provide a proof of concept, and the potency of this architecture, a more challenging task is classifying the MNIST dataset (mni). Furthermore, in comparing our architecture to others, MNIST is a common benchmark presented in literature. The MNIST is a dataset of hand written digits of resolution 28×28, 784 dimensions. Unfortunately, utilizing the evaluation data-encoding technique, it is impossible to conduct experiments on near-term quantum computers and simulators due to lacking of qubits and complexity of computation. Hence, we need to scale the dimensionality down. We make use of Principal Component Analysis (PCA (Lloyd et al., 2014)), as this is an efficient down scaling algorithm that has the potential to work on a quantum computer. We downscale from our 784 dimensions to 16 dimensions for quantum simulation. As for IBM-Q experiments, we make use of 4 dimensions due to the qubit-count limit of publicly available quantum computers. We make a note of that this PCA-ed data is the same data being fed to the classical neural networks. Furthermore, due to the cost-per-run model of IonQ, we do the majority of our architecture's evaluation on IBM-Q.

QuClassi allows us to perform 10-class classification on the MNIST dataset, unlike few other works that tackle

binary classification problems such as (3,6), or extend into low-class count multi-class classification such as (0,3,6). To understand how our learning state, we visualize the training process on learning to identify 0 against a 6 by looking at the final state that is to be passed to the SWAP test. As visualized in Fig. 8, an initial random quantum state is visualized to learn to classify a 0 against a 6. It is of note that the state visualization does not encapsulate possible learned entanglements, but serves as a visual aid to the learning process. As shown in Fig. 8, we see the evolution of the identifying state through epochs. Green arrows indicate the deep learning final state, and blue points indicate training points. We see initial identifying states to be random, but rotates and moves towards the data, such that its cost is minimized.
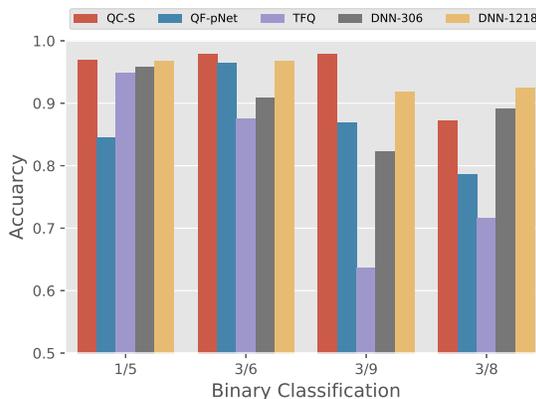

Figure 9: Binary Classification Comparison.

5.3.1 Binary Classification

To evaluate our network architecture and its efficiency, we compare our system performance with other leading quantum neural network implementations. We compare our architecture to two leading quantum deep learning models, namely Tensorflow Quantum (**TFQ**) (Broughton et al., 2020) and QuantumFlow (**QF-pNet**) (Jiang et al., 2020) and run classical deep neural networks that attain similar accuracy based on similar learning settings. Within this section, our quantum neural network is comprised of 17 qubits, with a total of 32 trainable parameters in the `QuClassi-S`, the single layer setting.

The comparison of binary classification results is visualized in Fig.9. Clearly, `QuClassi` consistently outperforms Tensorflow-Quantum, with the largest improvement of 53.75% being seen in our (3,9) classification, with an accuracy of 97.80%. When comparing with QF-pNet, `QuClassi` also surpasses it, with the largest margin being attained in the (1,5) classification, where we observe a 14.69% improvement over QF-pNet. `QuClassi` achieves 96.98% and QF-pNet 84.56%. Therefore, we consistently

perform above the binary classification tasks that shown in their works.

As for comparing with a classical deep neural network, to attain similar accuracy's, 1218 parameters were used on similarly parameterized networks. This is in contrast to our 32 parameters per `QuClassi-S` network. This is a substantial parameter count reduction of 97.37%.

5.3.2 Multi-class Classification

One significant contribution of our quantum neural network architecture is its multi-class adaptability. This can be relatively ambiguous in existing solutions. `QuClassi` provides substantially better multi-class classification accuracies.
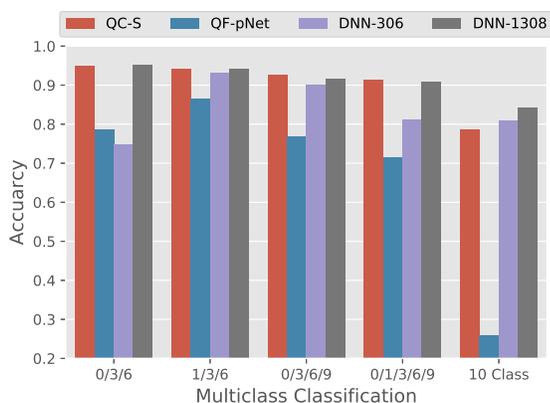
Figure 10: Multi-class Classification Comparison.

With the multi-class classification results visualized in 10, we observe that `QuClassi` consistent outperforming of QF-pNet in 3 class classification. For example, `QuClassi` achieves 94.91% and 94.18% for (0,3,6) and (1,3,6), comparing with 78.70% and 86.50% obtained by QF-pNet, which leads to accuracy increases of 20.60% and 8.88%. In 4 class classification, `QuClassi` gains 20.46% (92.49% vs 76.78%). As the number of classes increase, `QuClassi` outperforms QF-pNet more with 27.72% (91.40% vs 71.56%) in 5-class and 203.00% (78.69% vs 25.97%) 10-class classification. In QuantumFlow (QF-pNet), most of the training is done on the classical computer, where the traditional loss function is in use. With `QuClassi`, however, we employ a quantum-state based cost function that can fully utilize the qubits.

A trend emerges from these improvements, highlighting how `QuClassi` performs on higher class count. In comparing to classical deep neural networks that can achieve similar accuracies, `QuClassi` attains a 96.33% reduction in parameters on 5-class classification (48 vs 1308), and a 47.71% reduction in parameters on 10-class classification (160 vs 306) in the quantum setting.
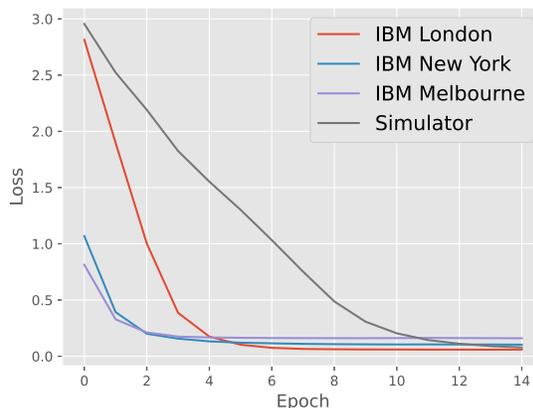
Figure 11: Quantum Computer Training: Iris.

5.4 Experiments on IBM-Q

As a proof of concept, we evaluate our result on actual quantum computers. For Iris dataset (4 dimensions), the `QuClassi-S` utilize 5 qubits. With MNIST dataset, however, the previous simulations were ran with a total of 17 qubits. It is relatively difficult to perform the same experiments on publicly quantum computers due to limited qubits available. Therefore, we downscale our data to 4 dimensions with PCA, such that we can make use of 5-qubit IBM-Quantum Computers. For experiments of Iris dataset, we utilize multiple IBM-Q sites around the world. For MNIST dataset, the experiment is conducted at IBM-Q Rome.

Fig. 11 presents the results of Iris dataset. Despite the fact IBM-Q quantum computer's communication channel introduces an extra overhead and shared public interfaces leading to large backlogs and queues, we attain a result of IBM-Q London site with an accuracy of 96.15% on Iris dataset, which is similar to the value obtained by simulations. We train each epoch through Iris dataset 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 (Zhang et al., 2019; Dueck et al., 2018) the integrity of physical qubits and `T1`, `T2` errors of IBM-Q machines could vary (Tannu & Qureshi, 2019), however, our design managed to attain a solution after few iterations, comparable to the simulator results we attain. Running experiments on actual quantum computers generated stable results and accuracy similar to the simulators. As seen in Fig 11, the loss of the quantum circuit converges similarly on a real quantum circuit, in different IBM-Q sites, to that of a simulator.

Fig. 12 plots the results of `QuClassi-S` with 4-dimension MNIST dataset on IBM-Q Rome site. We compare the real quantum experiments with simulations of TFQ and three ver-

sions of `QuClassi`. With (3,4) and (6,9), the experiments achieve similar results as simulations. For example, the measured difference of the experiment on the IBM-Q quantum computer and simulation of (6,9) is 0.2% (95.98% 96.17%). With (3,4), the accuracies of QC-S are 94.56% and 96.81% for real quantum computer and simulation, respectively. In the experiment of (2,9), however, the measured difference is 6.9%, where the experiment result is 89.09% and simulation, 95.27%. The larger difference is due to noise on quantum processors that depends on the processors performance and topology. We further evaluate `QuClassi`on IonQ's trapped ion quantum processor. We test (3,6) on IonQ and IBM-Q's Cairo machine. An ideal accuracy of 97.80% is attained, with IonQ attaining 80.00% and IBM-Q Cairo 72.00%. This difference can be attributed to the fully-connected nature of trapped IonQ, allowing for 0 SWAP operations, compared to IBM-Q Cairo, requiring 21 CNOT operations due to topological constraints, attributing to this discrepancy in accuracy. In addition, we can see that three versions of `QuClassi` perform similarly and consistently. This is the same trend as we have seen in Section 5.2 that, with low-dimensional data, the improvement of a deeper network is limited.
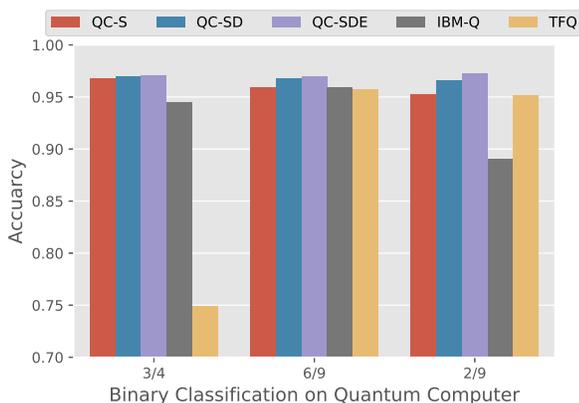

Figure 12: Quantum Computer Accuracies: MNIST.

6 DISCUSSION AND CONCLUSION

In this project, we propose `QuClassi`, a novel quantum-classic architecture for multi-class classification. The current accuracy attained by `QuClassi` performing binary classification on the MNIST data set is competitive, beating out Tensorflow Qauntum by up to 53.75% on binary classification. `QuClassi` also outperforms QuantumFlow by up to 14.69% on binary classification, and up to 203.00% on multi-class classification. Furthermore, to the best of our knowledge, `QuClassi` is the first solution that tackles a 10-class classification problem in the quantum setting with a performant result. Additionally, comparing `QuClassi` with similarly performed classical deep neural networks, `QuClassi` outperforms them by learning significantly faster, and requiring drastically less parameters by up to 97.37% on binary classification and up to 96.33% in multiclass classification.

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 78.69% accuracy, which is relatively poor when compared to its classical counterparts. Although using many more parameters, classical counterparts are able to reach an accuracy of near 100%, `QuClassi` shows the potential that quamtum may bring to us. Our future work will focus on improving the multi-class classification that aims to further improve the accuracy. Moreover, understanding the low-qubit representation of quantum data and its implications within the quantum based learning field should be investigated.

REFERENCES

Microsoft azure quantum. hhttps://azure.microsoft.com/en-us/services/quantum/.

Aws braket. https://aws.amazon.com/braket/.

ibmq. https://quantum-computing.ibm.com/.

Iris dataset. https://en.wikipedia.org/wiki/Iris_flower_data_set.

Mnist. http://yann.lecun.com/exdb/mnist/.

What is quantum computing? https://www.ibm.com/quantum-computing/learn/what-is-quantum-computing/.

Quantum supremacy experiment. https://ai.googleblog.com/2019/10/quantum-supremacy-using-programmable.html.

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

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pp. 173–182, 2016.

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

Broughton, M., Verdon, G., McCourt, T., Martinez, A. J., Yoo, J. H., Isakov, S. V., Massey, P., Niu, M. Y., Halavati, R., Peters, E., et al. Tensorflow quantum: A software framework for quantum machine learning. *arXiv preprint arXiv:2003.02989*, 2020.

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

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

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

Chen, S. Y.-C., Huang, C.-M., Hsing, C.-W., and Kao, Y.-J. An end-to-end trainable hybrid classical-quantum classifier. *Machine Learning: Science and Technology*, 2(4): 045021, 2021.

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

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

Debnath, S., Linke, N. M., Figgatt, C., Landsman, K. A., Wright, K., and Monroe, C. Demonstration of a small programmable quantum computer with atomic qubits. *Nature*, 536(7614):63–66, 2016.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423.

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

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

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

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

Google. Tensorflow quantum. https://www.tensorflow.org/quantum/tutorials/mnist.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. 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.

Jiang, W., Xiong, J., and Shi, Y. A co-design framework of neural networks and quantum circuits towards quantum advantage. *Nature communications*, 15(1):1–15, 2020.

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

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

Li, J., Topaloglu, R. O., and Ghosh, S. Quantum generative models for small molecule drug discovery. *IEEE Transactions on Quantum Engineering*, 2:1–8, 2021.

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

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

Microsoft. Quantum development kit. https://github.com/microsoft/Quantum.

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

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

Schuld, M., Sinayskiy, I., and Petruccione, F. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015.

Stein, S. A., Baheri, B., Chen, D., Mao, Y., Guan, Q., Li, A., Fang, B., and Xu, S. Qugan: A quantum state fidelity based generative adversarial network. In *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pp. 71–81. IEEE, 2021a.

Stein, S. A., L'Abbate, R., Mu, W., Liu, Y., Baheri, B., Mao, Y., Qiang, G., Li, A., and Fang, B. A hybrid system for learning classical data in quantum states. In *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pp. 1–7. IEEE, 2021b.

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

Tannu, S. S. and Qureshi, M. K. Not all qubits are created equal: a case for variability-aware policies for nisq-era quantum computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 987–999, 2019.

Waldrop, M. M. The chips are down for moores law. *Nature News*, 530(7589), 2016.

Wiebe, N., Braun, D., and Lloyd, S. Quantum algorithm for data fitting. *Phys. Rev. Lett.*, 109: 050505, Aug 2012. doi: 10.1103/PhysRevLett.109. 050505. URL https://link.aps.org/doi/10.1103/PhysRevLett.109.050505.

Wiebe, N., Kapoor, A., and Svore, K. M. Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning. *Quantum Info. Comput.*, 15(3–4): 316–356, March 2015. ISSN 1533-7146.

Wozniak, J. M., Jain, R., Balaprakash, P., Ozik, J., Collier, N. T., Bauer, J., Xia, F., Brettin, T., Stevens, R., Mohd-Yusof, J., Cardona, C. G., Essen, B. V., and Baughman, M. Candle/supervisor: a workflow framework for machine learning applied to cancer research. *BMC Bioinformatics*, 19(18):491, Dec 2018. ISSN 1471-2105. doi: 10.1186/s12859-018-2508-4. URL https://doi.org/10.1186/s12859-018-2508-4.

Xue, L., Cheng, L., Li, Y., and Mao, Y. Quantum machine learning for electricity theft detection: an initial investigation. In *2021 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, pp. 204–208. IEEE, 2021.

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

Zhao, Z., Fitzsimons, J. K., and Fitzsimons, J. F. Quantum-assisted gaussian process regression. *Phys. Rev. A*, 99:052331, May 2019. doi: 10.1103/PhysRevA.99. 052331. URL https://link.aps.org/doi/10.1103/PhysRevA.99.052331.

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

A ARTIFACT DESCRIPTION

A.1 Abstract

The following artifact appendix contains information used to reproduce the QuClassi results in the described paper. The evaluation is done on a quantum computing simulator, and can be connected to a IBM-Q device using IBM-Q Backend. Training the network is computationally expensive, hence changing the subsample variable will change the amount of data being fed to the network. On induction, the network will train, the variables will be saved, and the performance of the network will be posted.

A.2 Artifact check-list (meta-information)

- **Data set:** MNIST, Iris

- **Run-time environment:** Python 3.7, Qiskit

- **Hardware:** CPU, GPU for simulations, IBM-Q for real quantum experiments

- **Execution**: Single file execution, main.py

- **Metrics:** Accuracy

- **Output:** Accuracy

- **How much disk space required (approximately)?:** Approx. 10mb

- **How much time is needed to prepare workflow (approximately)?:** Time to install dependencies

- **How much time is needed to complete experiments (approximately)?:** Up to 7 days for full dataset

- **Publicly available?:** Yes

A.3 Description

A.3.1 How delivered

To access the code, see https://github.com/Samuelstein1224/QuClassiExample

A.3.2 Hardware dependencies

Quantum Processor access for real world validation, CPU for simulator simulation

A.3.3 Software dependencies

Python 3.7.0 with packages Qiskit,Tensorflow,Numpy and sklearn

A.3.4 Data sets

MNIST and Iris dataset.

A.4 Installation

Installation of Python, followed by using pip install suffices for all packages (Qiskit, Tensorflow, Numpy, Matplotlib and Sklearn)

A.5 Evaluation and expected result

On induction, the model will train a quantum neural network over the MNIST dataset, with SUBSAMPLE data points. The computational cost of training can be high, therefore changing this to what best suits the system you are running on is best. Accuracies expected are in the high 90's (approx. 96% accuracy), with some very slight variance due to the random nature of quantum state sampling.

A.6 Experiment customization

To change the simulator, the backend must be changed within the code. This can be used to apply noise etc. For the system to be run on a real quantum processor, the backend must be changed from a simulator to a real quantum processor.

A.7 Methodology

Submission, reviewing and badging methodology:

- http://cTuning.org/ae/submission-20190109.html

- http://cTuning.org/ae/reviewing-20190109.html

- https://www.acm.org/publications/policies/artifact-review-badging