IEEE Transactions on Quantum Engineering

Received 9 November 2023; revised 18 January 2024; accepted 12 February 2024; date of publication 19 February 2024; date of current version 3 April 2024.

Digital Object Identifier 10.1109/TQE.2024.3367234

A Quantum-Classical Collaborative Training Architecture Based on Quantum State Fidelity

RYAN L'ABBATE[1], ANTHONY D'ONOFRIO JR.[1], SAMUEL STEIN[2], SAMUEL YEN-CHI CHEN[3], ANG LI[2], PIN-YU CHEN[4] (Senior Member, IEEE), JUNTAO CHEN[1] (Member, IEEE), AND YING MAO[1]

[1]Computer and Information Science Department, Fordham University, Bronx, NY 10458 USA
[2]Pacific Northwest National Laboratory, Richland, WA 99354 USA
[3]Brookhaven National Laboratory, Upton, NY 11973 USA
[4]IBM Research, Yorktown Heights, NY 10598 USA

Corresponding author: Ying Mao (e-mail: ymao41@fordham.edu).

This work was supported in part by the National Science Foundation under Grant 2329020, Grant 2335788, and Grant 2301884 and in part by the U.S. Department of Energy (DOE) through the Office of Advanced Scientific Computing Research's "Advanced Memory to Support Artificial Intelligence for Science." PNNL is operated by Battelle for the DOE under Contract DE-AC05-76RL01830.

ABSTRACT Recent advancements have highlighted the limitations of current quantum systems, particularly the restricted number of qubits available on near-term quantum devices. This constraint greatly inhibits the range of applications that can leverage quantum computers. Moreover, as the available qubits increase, the computational complexity grows exponentially, posing additional challenges. Consequently, there is an urgent need to use qubits efficiently and mitigate both present limitations and future complexities. To address this, existing quantum applications attempt to integrate classical and quantum systems in a hybrid framework. In this article, we concentrate on quantum deep learning and introduce a collaborative classical-quantum architecture called co-TenQu. The classical component employs a tensor network for compression and feature extraction, enabling higher dimensional data to be encoded onto logical quantum circuits with limited qubits. On the quantum side, we propose a quantum-state-fidelity-based evaluation function to iteratively train the network through a feedback loop between the two sides. co-TenQu has been implemented and evaluated with both simulators and the IBM-Q platform. Compared to state-of-the-art approaches, co-TenQu enhances a classical deep neural network by up to 41.72% in a fair setting. In addition, it outperforms other quantum-based methods by up to 1.9 times and achieves similar accuracy while utilizing 70.59% fewer qubits.

INDEX TERMS Collaborative training, quantum deep learning, quantum-classical hybrid systems.

I. INTRODUCTION

Recent years have witnessed significant progress in machine learning and deep learning. Groundbreaking models and algorithms have significantly enhanced our capabilities to identify patterns and process data in areas, such as computer vision, natural language processing, and finance. However, this accelerated development has led to an exponential increase in the computational power needed to execute increasingly sophisticated deep learning tasks. As the era of Moore's Law comes to a close, however, the acceleration of computational demand is starting to surpass the growth in available computing power [1]. Consequently, this trend fuels the search for alternative computing approaches capable of managing the ever-growing computational needs. Quantum computing provides considerable potential in delivering the increased computational power essential to meet the expanding demands of deep learning. Classical computers employ binary bits, representing either 0 or 1, which constitute the current computing standard. In contrast, quantum computers use quantum bits (or qubits), which are probabilistic combinations of 0 and 1, achieved through quantum *superposition* and *entanglement*. As a result, the expected value of a qubit measurement can represent any number between 0 and 1. Therefore, a specific number of qubits can exhibit

substantially greater representational power compared to an equivalent number of classical bits. In 1998, the first quantum computer capable of executing computations was developed [2]. The IBM-Q Experience was introduced in 2016, granting developers access to state-of-the-art quantum resources [3]. In 2020, Google AI demonstrated that a 53-qubit quantum computer could complete a task in 200 s that would require a classical computer more than 10 000 years. This advantage of quantum computing over classical computing is frequently referred to as "quantum supremacy" [4].

Researchers inspired by the concept of quantum supremacy are actively exploring methods to convert classical algorithms into their quantum versions, aiming to achieve significant reductions in time complexity compared to classical counterparts. Quantum speed-ups have already been demonstrated for Shor's algorithm [5], which addresses prime factorization and discrete logarithms, and Grover's algorithm, which tackles database searches [6]. Quantum computing can be applied to machine learning tasks by employing variational quantum circuits (VCQ)–quantum circuits with trainable parameters. Specific areas within classical learning, such as deep learning and support vector machines, could potentially benefit from quantum computing [7], [8]. Quantum speed-ups have been achieved for several algorithms, including expectation maximization solving [9] (where the algorithm's speed has been increased to sublinear time [10]), support vector machines [11], and natural language processing [12].

However, in the noisy intermediate-scale quantum (NISQ) era, the qubits are both limited in number and subject to noise. For instance, IBM-Q provides only 5–7 qubit machines to the public. Furthermore, as the qubit count increases, the computational complexity of the system grows exponentially [13], which leads to a higher overall noise level in a quantum machine. In the context of deep learning, an increased number of qubits may employ a greater number of gates, potentially augmenting circuit depth and noise interference. Consequently, it is crucial to efficiently and reliably utilize the representational power of qubits through effective encoding, making quantum algorithms more feasible on both current and NISQ quantum computers, while mitigating the surge in computational complexity as the number of qubits increases. A potential solution to data encoding challenges involves performing classical preprocessing of the data for compression and/or feature extraction. One prevalent method for dimension reduction is principal component analysis (PCA), as demonstrated in prior works [14], [15], [16], [17], [18]. However, PCA may not possess the representational power necessary to compress data accurately. More sophisticated methods, such as employing neural network layers, demand substantial pretraining and significantly increase the number of parameters requiring tuning. Therefore, there is a pressing need for efficient data compression techniques tailored to quantum machine learning.

In this work, we introduce a novel classical-quantum collaborative training architecture, that incorporates a classical tensor network (TN) into the feature extraction stage to facilitate dimensionality reduction. Specifically, the TN serves as a trainable module designed to capture high-level abstractions of the input data, the output of which is subsequently fed into a VQC for classification purposes. Furthermore, we employ a quantum-state-fidelity-based cost function to train the model directly on qubits' states. Our proposed solution presents significant advantages over existing techniques, such as PCA, which lacks trainability, and conventional neural networks that require a considerable number of parameters to be optimized or pretrained. The integration of our hybrid system enables more efficient data encoding, thereby enhancing the overall performance of the quantum machine learning pipeline. The main contributions are summarized as follows.

1) We propose co-TenQu, a quantum-classical collaborative training architecture. On the classical part, it employs TN layers for data preprocessing and preparation. In the quantum part, it utilizes a preprocessed dataset to build circuits with fewer qubits to reduce the overall qubit requirement and noise interference.
2) We introduce a quantum state fidelity-based cost function. Instead of converting back to classical states, co-TenQu train the model directly on quantum states aiming at accelerating the training process and improving performance.
3) We implement co-TenQu with popular quantum toolkits, e.g., Qiskit and PennyLane, and compare it with state-of-the-art solutions in literature, by up to 1.9× and 70.59% less quantum resources. In addition, we conduct proof-of-concept experiments on 14 different IBM-Q quantum machines.

II. RELATED WORK

Recent developments [19], [20], [21], [22], [23] in quantum computing show great potential to enhance current learning algorithms through utilization of the qubit, the unit of quantum information. In this field, quantum neural networks (QNN) have emerged as a promising research area in quantum machine learning [8], [24]. Due to the limited quantum resources available, most of the existing works focused on numerical analysis or datasets with lower dimensionalities [17], [25], [26], such as MNIST [27].

Farhi et al. [28] introduced a QNN for binary classification, which utilizes quantum entanglement to enhance the model's computational power. In addition, quantum circuit learning [29], [30] developed a quantum-classical hybrid algorithm. They employed an iterative optimization of the parameters to circumvent the high-depth circuit. Moreover, Stokes et al. [31] presented a novel method for gradient descent using quantum circuits, enabling the optimization of VQCs in a manner analogous to classical neural networks. However, these solutions focused on theoritical analysis and only numerical experiments were provided.

In the NISQ era, QCNN [32] suggests a design for a quantum convolutional neural network that uses $O(\log(N))$ trainable parameters for N-dimensional inputs and can be realized on near-term quantum computers. In addition, QuCNN [33] employed an entanglement-based backpropagation for NISQ machines. Jiang et al. [34] proposed a codesign framework named QuantumFlow, which features quantum-friendly neural networks, a mapping tool to generate quantum circuits, and an execution engine. However, QuantumFlow requires local training of the network prior to mapping to quantum circuits, which leads to sensitivity to noise when implemented on real quantum computers as opposed to simulations.

Expanding upon the use of quantum operations to perform distance measurements, Stein et al. [14], [15] proposed the QuClassi system: a hybrid quantum-classical system with a quantum-state-fidelity based loss function. QuClassi was able to provide improvements in accuracy compared to other contemporary quantum-based solutions, such as TensorFlow Quantum [35] and QuantumFlow. The QuClassi system demonstrated success in both binary and multiclass classification. It used PCA to compress dataset classically. However, PCA fails to fully utilize the classical resources by providing trainable layers. TN-VQC [36] proposed the use of TNs for feature extraction and data compression to achieve higher classification accuracy for VCQs. Tensor networks do provide the advantage of having fewer parameters compared to neural networks while still providing some trainability unlike PCA. TN-VQC employed a circuit architecture involving CNOT gates rather than CSWAP gates like QuClassi.

This article proposes co-TenQu, a hybrid quantum-classical architecture for deep neural networks. Comparing with existing literature, it utilizes a quantum-state fidelity based cost function to train the quantum section directly on qubits' states. In addition, TNs are employed to fully exploit classical resources to compensate for the limitations (e.g., low qubit count and noises) of quantum resources. Through a collaborative training process, co-TenQu is able to outperform state of the arts.

III. BACKGROUND

In this section, we present the background that is necessary for designing our solution.

A. QUANTUM COMPUTING BASICS

1) QUBIT AND ITS SUPERPOSITION

Classical computing uses bits that are binary in nature and measure either 0 or 1. Quantum computing uses quantum bits or qubits. Qubits, unlike classical bits, are a probabilistic mixture of 0 and 1. This mixture of 0 and 1 is known as a superposition. Upon measurement, the qubit in superposition will collapse to either a value of 0 or 1. Quantum circuits are often run many times, using the results to get a probability distribution for the circuit results. Calculations are performed by manipulating the probability distributions of qubits. 0 and 1 can be represented in vector notation, as seen in (1).

Quantum systems are often described using ⟨bra| |ket⟩ notation, where ⟨bra| and |ket⟩ represent horizontal and vertical quantum state vectors, respectively. Because a qubit is a mixture of 0 and 1, qubit states are described mathematically as a linear combination of $|0\rangle$ and $|1\rangle$, as in

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

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle. \tag{2}$$

This linear combination of qubit states is referred to as a qubit's statevector. $|0\rangle$ and $|1\rangle$ are orthonormal vectors in an eigenspace. In (2), $|\Psi\rangle$ represents the qubit state, a probabilistic combination of $|0\rangle$ and $|1\rangle$.

The tensor product of qubit states can be used to describe the quantum states of multiple qubits. The tensor product between the qubits shown in (2) and (3) can be described using (4).

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

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

$|0\rangle$ and $|1\rangle$ represent opposite points of the sphere on the z-axis. Measurements of qubit states can be taken with respect to any basis, but convention typically dictates that measurements are taken against the z-axis. However, the x-axis, y-axis, or any pair of opposite points on the sphere could potentially be used as a basis of measurement. 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.

B. QUANTUM GATES

Similar to classical data which are manipulated and encoded using gates, quantum data are manipulated and encoded using quantum gates. Quantum gates can either perform a rotation about an axis or perform an operation on a qubit based on the value of another qubit. These are referred to as rotation gates and controlled gates, respectively.

1) SINGLE-QUBIT GATES

A common type of single-qubit operations are the rotation gates. These gates perform qubit rotations by parameterized amounts. The generalized single-rotation gate R is shown in matrix form in

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

Three commonly used special cases of this gate are the R_X, R_Y, and R_Z gates. These gates represent rotations in the x, y, and z plane and are expressed in (6), (7), and (8), respectively. R_X and R_Y can be thought of as special cases of the R gate in which $\phi = 0$ and $\phi = \frac{\pi}{2}$, respectively. Therefore, $R_X(\theta)$ is a rotation about the x-axis by angle θ and $R_Y(\theta)$ is a rotation about the y-axis by angle θ. The derivation of R_Z from the general rotation gate is less straightforward, and thus, is not

included here.

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

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

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

2) HADAMARD GATE

A fundamental gate of quantum computation is the Hadamard gate. It is a single-qubit gate that puts a qubit into superposition, as described in Section III-A1. It can be expressed in the matrix shown in (9). The $1/\sqrt{2}$ coefficient is due to the fact that the sum of the squares of the state amplitudes must add to 1, so each state has a probability of $1/2$ and an amplitude of $1/\sqrt{2}$

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{9}$$

3) TWO-QUBIT GATES

There are also operations that function as two-qubit rotations which perform an equal rotation on two qubits. These gates are described in (10), (11), and (12). Note that these gates are expressed as 4×4 matrices while the single-qubit gates were 2×2 matrices. This is because for a two-qubit gate, each individual qubit has two possible measurements, yielding four possible results ($|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$) rather than two as seen previously for the single-qubit gates.

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

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

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

4) CONTROLLED GATES

There are also two-qubit gates which utilize a control qubit and a target qubit. These gates, known as controlled gates, perform an operation on a target qubit depending on the value of the control qubit.

CNOT Gate: The CNOT gate is an example of a two-qubit gate used in quantum computing. The CNOT gate flips the value of the target qubit if the control qubit is measured as 1 and does nothing otherwise. The CNOT gate can be seen

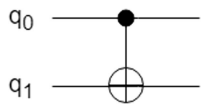


FIG. 1. CNOT gate circuit notation.

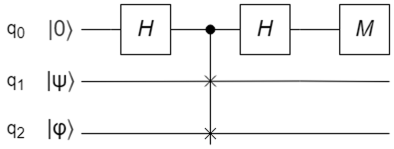


FIG. 2. SWAP test quantum circuit.

represented in matrix form as follows:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{13}$$

Fig. 1 depicts the circuit notation for the CNOT gate. q_0 is the control qubit and q_1 is the target qubit.

Controlled Rotation Gates: Equations (14)–(16) are controlled rotation gates in matrix notation. Controlled rotation gates are similar to the CNOT gate but apply a rotation when the control qubit measures 1 instead of flipping the state. This allows for variable levels of entanglement between qubits.

$$\text{CR}_X(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ 0 & 0 & -\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{14}$$

$$\text{CR}_Y(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ 0 & 0 & \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix} \tag{15}$$

$$\text{CR}_Z(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{\frac{i\theta}{2}} & 0 \\ 0 & 0 & 0 & e^{\frac{i\theta}{2}} \end{bmatrix} \tag{16}$$

C. CONTROLLED SWAP GATE

Another type of controlled gate is the controlled SWAP gate. The SWAP gate measures the difference between two quantum states and outputs the result to an ancilla qubit. Therefore, this gate is a three-qubit gate. The SWAP test output values range from 0.5 to 1. Maximally different (orthogonal) states will measure 1 with 50% probability while identical states will measure 1 with 100% probability. The SWAP test gate can be used to measure quantum state fidelity. The controlled SWAP gate is described in

$$\text{CSWAP}(q_0, q_1, q_2) = |0\rangle\langle 0| \otimes I \otimes I + |1\rangle\langle 1| \otimes \text{SWAP} \tag{17}$$

IEEE Transactions on Quantum Engineering

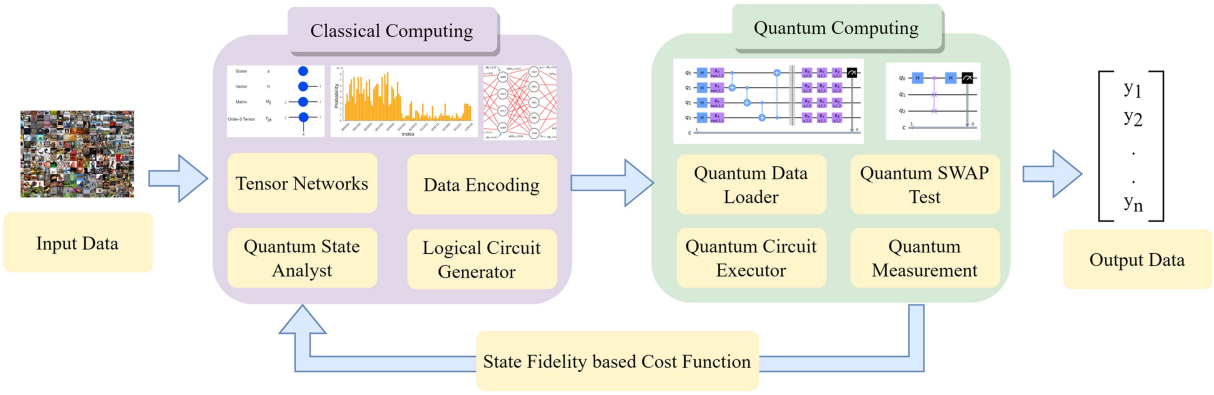


FIG. 3. co-TenQu: A quantum-classical collaborative training architecture.

$$\text{CSWAP}(q_0, q_1, q_2) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (18)$$

Fig. 2 depicts a SWAP test being performed. The ancilla qubit q_0 is placed is superposition using a Hadamard gate. Then, a SWAP test is performed between q_1 and q_2 and measured onto q_0. Another Hadamard gate is performed on the ancilla qubit. Finally, the ancilla qubit is then measured onto a classical bit to obtain the result.

One advantage of the CSWAP gate is that it only requires the measurement of the ancilla qubit. When qubits are measured directly, their states collapse and the superposition is lost. The SWAP test allows the superposition of the other qubits to be maintained by measuring the quantum state fidelity through the ancilla qubit instead of measuring the qubits directly. Therefore, minimal information is lost through measurement.

D. QUANTUM ENTANGLEMENT

A key principle of quantum computing is quantum entanglement. A qubit's state is said to be entangled when its measurement is dependent on the measurement of another qubit. This dependence allows information to be transferred between qubits, even if they are not physically close together (a phenomena sometimes referred to as "action at a distance"). When one entangled qubit is measured, the other entangled qubit's state also collapses. For example, if two qubits are entangled using the CNOT gate, after the state of one qubit is measured, the state of the second entangled qubit can be predicted with absolute certainty. Quantum entanglement is a key component of the quantum advantage over classical computing, as it is a property of quantum computing with no classical equivalent.

IV. SYSTEM DESIGN

Our architecture employs a feedback loop between classical computers and quantum computers, as illustrated in Fig. 3. Initially, the data are fed into TNs with a layer of trainable parameters and output into a preconfigured dimension. The data are then converted from classical data into quantum data through a quantum data encoding method, as outlined in Section IV-A. This results in a quantum dataset are represented by quantum state preparation parameters. For each predictable class in the dataset, a quantum state is initialized with the same qubit count as the number of qubits in the classical quantum dataset, due to the constraints of the SWAP test. The quantum states, along with quantum classical data, are then used to generate a logical quantum circuit and sent to a quantum computer for further processing.

This initialization of state is the core architecture to co-TenQu. In this, a quantum circuit of a certain number of layers representing a quantum deep neural network (detailed in Section IV-B) 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 and analyzed with quantum state fidelity-based cost function (described in Section IV-D), forming the overall collaborative quantum-classical deep learning architecture of co-TenQu.

The quantum computer calculates the quantum fidelity from one ancilla 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.

A. DATA ENCODING ON QUBITS

When evaluating quantum machine learning architectures on classical datasets, it is crucial to have a method for translating classical data into quantum states. One question that arises is how to represent a classical dataset in a quantum setting.

Our architecture utilizes the expectation of a qubit to translate traditional numerical data points. To achieve this, data $x_1, x_2, \ldots, x_n$ of dimension d can be mapped onto a quantum setting by normalizing each dimension d_i to fall within the range of 0 to 1. This is because a qubit's expectation can only take on values within this range. In contrast to classical computing, which requires a string of bits to represent the same number, encoding a single dimension data point only requires one qubit. To translate the traditional value x_i into some quantum state, we perform a rotation around the y-axis parameterized by the following equation:

$$\mathrm{RY}(\theta_{x_i}) = 2\sin^{-1}(\sqrt{x_i}). \tag{19}$$

The $\mathrm{RY}(\theta_{x_i})$ operation results in the expectation of a qubit being measured against the z-axis, corresponding to the x_i value from the classical data that the qubit encodes. Building upon this concept, we can encode the second dimension of data across the X–Y plane. To achieve this, we employ two parameterized rotations on one qubit initialized in state $|0\rangle$ to prepare classical data in the quantum setting. To encode a data point, we apply the necessary rotations across $d/2$ qubits, with each rotation parameterized by the normalized value of that data point's corresponding dimension. It is worth noting that the encoding of 2-D data onto a single qubit may pose challenges for extreme values of x. However, we explore the dual-dimensional encoding as a possible method of reducing high qubit counts and evaluate the performance when each dimension of data are encoded into one respective qubit solely through an RY Gate. This approach is validated by the fact that we never measure any of our qubits, but only their quantum fidelity through the SWAP test. As a result, we can bypass the superposition-collapsing issue inherent in this approach. We encode the second dimension of data on the same qubit through the following rotation:

$$\mathrm{RZ}(\theta_{x_{i+1}}) = 2\sin^{-1}(\sqrt{x_i}). \tag{20}$$

When dealing with a limited number of qubits, methods that can reduce the number required are highly valuable. Unlike classical computers, which utilize formats, such as integers and floats, classical data encoding in quantum states does not have a tried and tested method. Therefore, our approach may be subject to criticism. Nevertheless, our approach has been tested and proven to be a viable solution to the problem at hand. In addition, having knowledge of both the qubit's expectation across the Y and Z domains enables the reconstruction of classical data. Various methods for classical-to-quantum data encoding exist, ranging from encoding 2^n classical data points across n qubits using state-vector encoding to encoding classical data into a binary representation on quantum states by translating a vector of binary values onto qubits. The former method is highly susceptible to noise, whereas the latter loses significant information in the process but is less susceptible to noise and exponential-sampling problems. Exponential data-encoding methods also exist and can be integrated into co-TenQu since it does not directly perform quantum state tomography, making the data encoding section scalable.

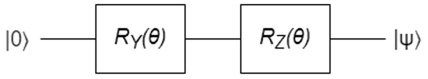


FIG. 4. Single qubit unitary.

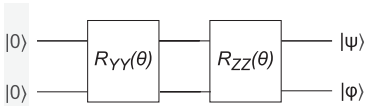


FIG. 5. Dual qubit.

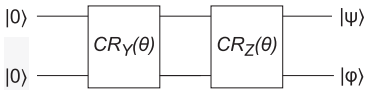


FIG. 6. Entanglement.

The co-TenQu quantum circuits consist of $n+1$ qubits, with n representing the dimension of the input data. The input data are encoded on $n/2$ qubits, while trainable parameters are applied to the remaining $n/2$ qubits. In addition, there is one ancilla qubit used for SWAP test measurements.

B. QUANTUM LAYERS

Similar to classical artificial neural networks, quantum circuits can also be thought of as having layers. For a quantum circuit, these layers would be comprised of quantum gates.

In co-TenQu, we define three quantum layer styles: 1) single-qubit unitary; 2) dual-qubit unitary; and 3) controlled-qubit unitary. Each of these layer styles comprises rotations that serve as the trainable parameters in our quantum machine learning model. Defining these three types of layers enables system design at a higher level than individual gates.

Single-Qubit Unitary: A single-qubit unitary layer involves single-qubit rotations around the y-axis and z-axis (R_Y and R_Z). This allows for total manipulation of a qubit's quantum state. A single-qubit unitary layer is depicted in Fig. 4.

Dual-Qubit Unitary: A dual-qubit untary layer involves dual-qubit rotations around the y- and z-axis (R_{YY} and R_{ZZ}). The same y rotation and z rotation are applied to both qubits involved. A dual-qubit unitary layer is depicted in Fig. 5.

Entanglement-Based Unitary: A controlled-qubit unitary utilizes controlled rotation gates (CR_Y and CR_Z) to entangle qubits. The use of these gates allows the level of entanglement between qubits to be trainable. In Fig. 6, the top row is the control qubit and the bottom row is the target qubit.

The layers can be combined linearly to composite a multilayer mode. For example, as seen in Fig. 7, the circuit features three layer types: 1) single-qubit unitary; 2) dual-qubit; unitary; and 3) controlled-qubit unitary.

C. PARAMETER SHIFT

Backpropagation is a necessary step for training any deep neural network. Gradients for the parameters of quantum

IEEE Transactions on Quantum Engineering

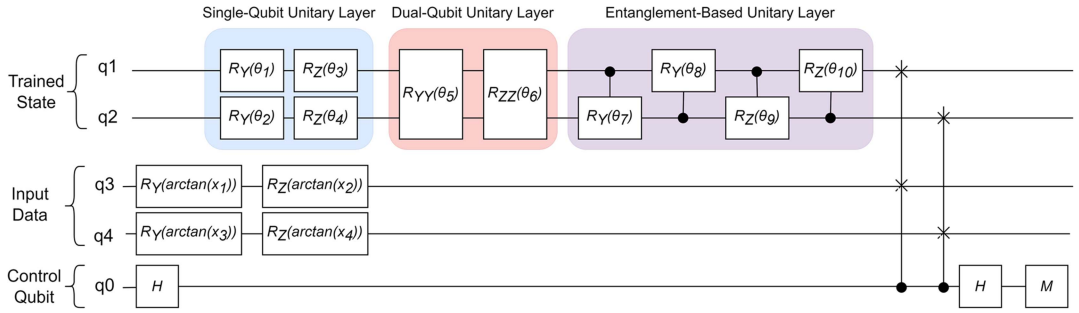


FIG. 7. co-TenQu with 3-layer and 5-qubit setting.

circuits cannot be calculated by the same methods used in classical backpropagation. Therefore, the gradients of the parameters are calculated using parameter shift shown as

$$\nabla_\theta f(\theta) = 0.5 * [f(\theta + s) - f(\theta - s)]. \tag{21}$$

With the parameter shift rule, the quantum circuit can be viewed as a black box and the gradient is calculated by obtaining circuit results when the parameter is increased or decreased by a shift s. The difference in results can be used to obtain a gradient for the parameter.

D. 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 ancilla 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 (23) 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.

$$\min(\text{Cost}(\theta_d, X) = \frac{1}{n}\sum_{i=1}^{n} \text{SWAP}(|\phi_{X(i)}\rangle, |\omega\rangle) \tag{22}$$

$$\text{Cost} = -y\log(p) - (1-y)\log(1-p) \tag{23}$$

where θ_d is a collection of parameters defining a circuit, x is the dataset, $\phi_{x(i)}$ is the quantum state representation of data point i, and ω is the state being trained to minimize the function in (22) and (23).

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 as

$$\frac{\delta \text{Cost}}{\delta \theta_i} = \frac{1}{2}\left(f(\theta_i + \frac{\pi}{2\sqrt{\epsilon}}) - f\left(\theta_i - \frac{\pi}{2\sqrt{\epsilon}}\right)\right) \tag{24}$$

where 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.

The gradients of quantum parameters can also be determined using numerical methods. Equation (25) shows a formula to numerically determine the gradients of quantum parameters. However, numerical methods can run into issues due to the noise an error associated with current quantum computers. Therefore, the gradients calculated may be inaccurate and lead to inefficiency in training [37].

$$\nabla_\theta f(\theta) = \frac{f(\theta + s) - f(\theta - s)}{2\tilde{}s} \tag{25}$$

E. HYBRID TENSOR NETWORK AND QUANTUM CIRCUIT DESIGN

A hybrid model with a TN and a quantum circuit is used to classify 28 × 28 MNIST images. The TN functions as a trainable feature extractor to compress the 784-D data into four dimensions for classification by the quantum circuit.

There are several different types of TNs. For this study, the matrix product state (MPS) will be employed. The MPS, also referred to as a tensor train, is the simplest type of TN. In an MPS, tensors are contracted through virtual indexes. The number of these indexes is referred to as a bond dimension, denoted by χ. A greater bond dimension indicates a greater amount of quantum entanglement that can be represented and therefore more representational power in the MPS. An N-dimensional input is mapped into a product state using the mapping shown in (26). This mapping for the MPS input is known as a feature map.

$$x \to |\Psi\rangle = \begin{bmatrix} \cos\left(\frac{\pi}{2}x_1\right) \\ \sin\left(\frac{\pi}{2}x_1\right) \end{bmatrix} \otimes \begin{bmatrix} \cos\left(\frac{\pi}{2}x_2\right) \\ \sin\left(\frac{\pi}{2}x_2\right) \end{bmatrix} \otimes \ldots \otimes \begin{bmatrix} \cos\left(\frac{\pi}{2}x_N\right) \\ \sin\left(\frac{\pi}{2}x_N\right) \end{bmatrix} \tag{26}$$

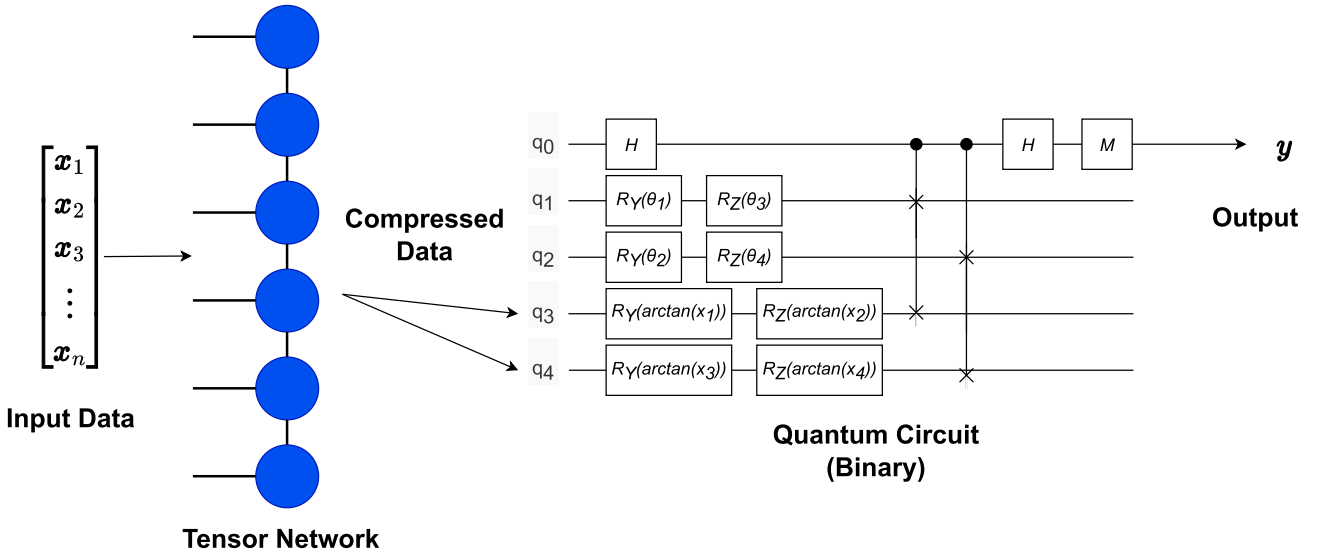


FIG. 8. co-TenQu diagram (binary).

The MPS takes an input of size 784 (28 times 28) and outputs an n-length tensor. The output dimension of the MPS is a hyperparameter of the system that can be adjusted based on the problem at hand. This tensor output from the MPS is then encoded into a quantum circuit. n dimensions are encoded onto $n/2$ qubits using an R_Y and R_Z rotation on each qubit to encode two dimensions per qubit. Because the output of the MPS is not bounded, the arctangent of the input values are encoded for the rotations to keep inputs to the quantum circuit in the range of $[-\frac{\pi}{2}, \frac{\pi}{2}]$. After encoding, the circuit is run to get a quantum state fidelity measurement. This measurement is then mapped from [0.5, 1] to [0, 1] by subtracting 0.5 and multiplying by 2. The swap test may sometimes measure below 0.5 due to statistical error, so an ReLu layer is applied after the quantum circuit to prevent negative outputs. For multiclass classification, the ReLu layer is not used due to the presence of the Softmax layer. If the output is below 0.5, the image is classified as 0; otherwise the image is classified as label 1. The quantum circuit has up to three types of layers: 1) single-qubit unitary, 2) dual-qubit unitary, and 3) controlled-qubit unitary.

For binary classification, a single quantum circuit is run. For n-class classification where $n > 2$, n quantum circuits with the same circuit design, but different parameters are run in parallel. The outputs of these circuits are then softmaxed to get probabilities for each class. The image is classified as the class with the highest probability. System diagrams for the binary and multiclass versions of this system can be seen in Figs. 8 and 9, respectively

This system can be trained all together at once rather than requiring a feature extractor to be pretrained. The entire training algorithm is summarized in Algorithm 1. First, the data are loaded, as shown in (26) (Line 1). Lines 2–3 involve introducing training parameters set by the user at run time. The learning rate α indicates how large the updates to the system parameters should be during training. The network weights are initialized randomly. The number of epochs ϵ indicates how many times the network will be trained on the dataset X. Line 6 represents the input data x being encoded into the TN. Line 7 represents the output of the TN being obtained through tensor contractions. Lines 8–23 represent the process by which each of the quantum parameters θ is updated. The output of the TN and the trainable quantum circuit parameters θ_d are all loaded into the quantum circuit with one of the parameters (θ) either increased by $\frac{\pi}{2}$ (Δ_{fwd}) and the SWAP test is performed. Then, the parameters are reset, θ is decreased by $\frac{\pi}{2}$ (Δ_{bck}), and the SWAP test is performed again. The overall cost function of the network $f(\theta_d)$ is then obtained for the two adjusted parameter values and used to update θ, as seen in Line 22. After all of the quantum parameters have been updated, the parameters of the TN layer are updated, as seen in Line 24. The quantum neural network is induced across all trained classes and the quantum state fidelity outputs are softmaxed. The class with the highest probability is returned as the classification.

Algorithm 1 presents a hybrid training process that involves both classical and quantum ends, e.g., data loading and TNs on the classical side; quantum layers and measurements on the quantum side. The time and space complexity analysis should consider both quantum and classical resources. Due to the page limit and scope, we omit the theoretical algorithm analysis in this article.

V. EVALUATION

We utilized Python 3.9 and the IBM Qiskit Quantum Computing simulator package to implement co-TenQu. The circuits were trained on NSF Cloudlab M510 nodes at the University of Utah datacenter. In our experiments, co-TenQu is compared with state-of-the-art solutions listed as follows.

1) *PCA-QuClassi [14]:* It is the predecessor of co-TenQu. Instead of a collaborative quantum-classical training

IEEE Transactions on Quantum Engineering

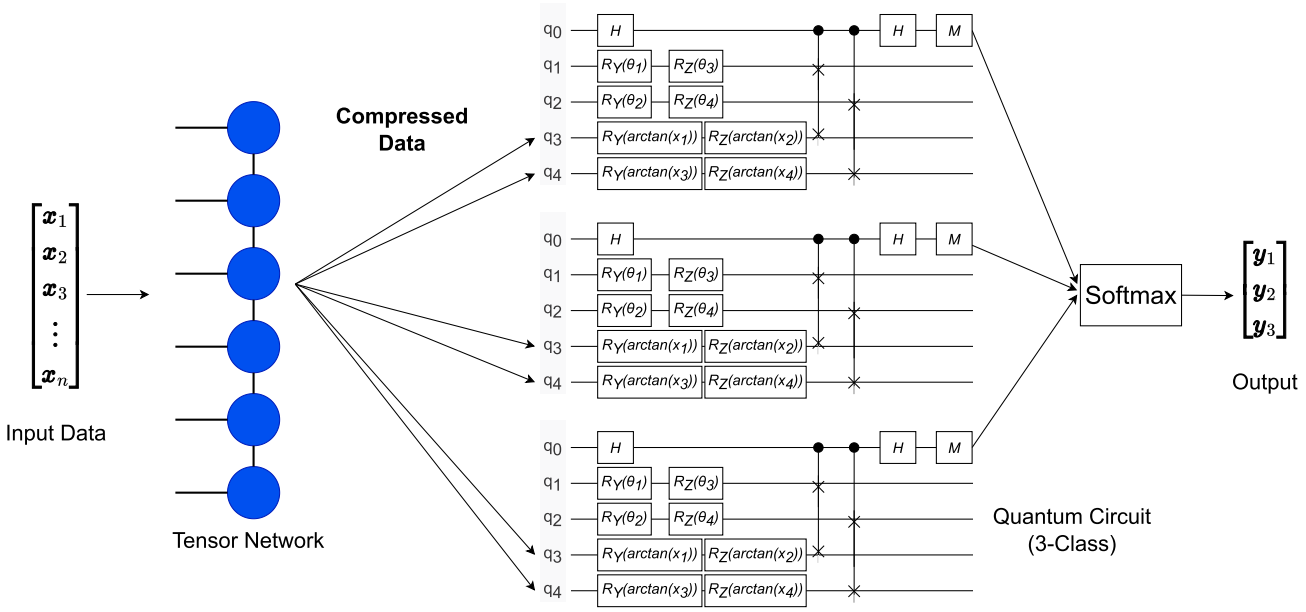


FIG. 9. co-TenQu diagram (3-class).

framework, it utilizes PCA to reduce the dimensions of the dataset. In our evaluations, we use PCA-5, PCA-7, and PCA-17 to denote its 5-qubit, 7-qubit, and 17-qubit settings. In addition, PCA-QuClassi has been compared with its different versions, including the single qubit unitary layer, dual qubit unitary layer, and entanglement layer.

2) *QuntumFlow [34] (QF-pNet):* It employs a codesign framework of QNNs and utilizes downsampling to reduce the dimensions along with the amplitude encoding method.
3) *TensorFlow Quantum [35] (TFQ):* The example codes provided by the Tensorflow Quantum library are based on Cirq circuits and standard layer designs.
4) *DNN-Fair [38]:* A classical deep neural network for MNIST data may contain 1.2 M parameters. For a more fair comparison, we construct a deep neural network with 3145 parameters.

Furthermore, when comparing our co-TenQu architecture to above-mentioned solutions in literature of quantum deep learning, the MNIST dataset is a commonly used benchmark. MNIST comprises hand-written digits of resolution 28 × 28, resulting in 784 dimensions. However, the evaluation data-encoding technique makes it impractical to perform experiments on near-term quantum computers and simulators due to the lack of qubits and computational complexity. As a result, we need to reduce the dimensionality to perform practical experiments. Therefore, it is necessary to reduce the dimensionality of the dataset. In our research, we have reduced the number of dimensions to 4 for binary experiments/simulations and 6 for multiclass evaluations. Besides the original MNIST dataset, our evaluation involves two derived datasets, Fashion MNIST and Extended MNIST. We conducted both binary and multiclass experiments and evaluated them with simulators, as well as IBM-Q quantum machines.

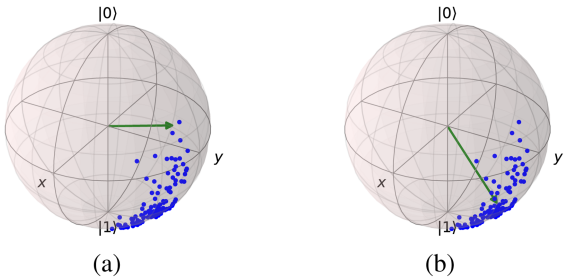


FIG. 10. Identify 0 (Epoch 1 versus 10). (a) Qubit 1–0 epochs. (b) Qubit 1–10 epochs.

A. QUANTUM BINARY CLASSIFICATION

In order to understand how our learning process works, we visualized the training process of identifying a 0 against a 6 by looking at the final state that is passed to the SWAP test. As illustrated in Fig. 10, an initial random quantum state is used to learn to classify 0 against 6. It is important to note that the state visualization does not account for potential learned entanglements, but serves as a visual aid to the learning process. In Fig. 10, we can observe the evolution of the identifying state through epochs. The green arrows indicate the deep learning final state, and the blue points represent the training points. Initially, the identifying states are random, but they rotate and move toward the data, gradually minimizing the cost.

Algorithm 1: co-TenQu Algorithm

Data set Loading Dataset: $(X|Class : Mixed)$
Distribute Dataset X By Class
Parameter Initialization:
Learning Rate : $\alpha = 10^{-4}$
Network Weights : θ_d = [Rand Num between $0 - 1 \times \pi$]
epochs : $\epsilon = 40$
Dataset: $(X|Class = \omega)$
Qubit Channels: $Q = 2n_{X_{\text{dim}}}$
for $\zeta \in \epsilon$ **do**
  for $x_k \in X$ **do**
    Encode in TN $x \rightarrow \begin{bmatrix} \cos(\frac{\pi}{2}x_1) \\ \sin(\frac{\pi}{2}x_1) \end{bmatrix} \otimes \begin{bmatrix} \cos(\frac{\pi}{2}x_2) \\ \sin(\frac{\pi}{2}x_2) \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} \cos(\frac{\pi}{2}x_N) \\ \sin(\frac{\pi}{2}x_N) \end{bmatrix}$
    Perform Tensor contractions to get TN output
    for $\theta \in \theta_d$ **do**
      Perform Hadamard Gate on Q_0
      Load $x_k \xrightarrow[\text{Data Encoding}]{\text{Quantum}} Q_{Q_1} \rightarrow Q_{count}$
      Load $\theta_d \xrightarrow[\text{Data Encoding}]{\text{Quantum}} Q_{\frac{Q_{count}}{2}+1} + 1 \rightarrow \frac{Q_{count}}{2} + 1$
      Add $\frac{\pi}{2} \rightarrow \theta$
      $\Delta_{fwd} = (E_{Q_0} f(\theta_d))$
      CSWAP(Control Qubit $= Q_0$, Learned State Qubit, Data Qubit)
      Measure Q_0
      Reset Q_0 to $|0\rangle$
      Perform Hadamard Gate on Q_0
      Subtract $\frac{\pi}{2} \rightarrow \theta$
      CSWAP(Control Qubit $= Q_0$, Learned State Qubit, Data Qubit)
      Measure Q_0
      $\Delta_{bck} = (E_{Q_0} f(\theta_d))$
      $\theta = \theta - (0.5 * (\Delta_{fwd} - \Delta_{bck})) \times \alpha$
    end for
    Update TN parameters
  end for
end for

For binary classifications, we adopted popular digit combinations from literature, specifically (1, 5), (3, 6), (3, 8), and (3, 9). The binary classification results are compared and visualized in Fig. 11. Clearly, co-TenQu consistently outperforms all other solutions. For example, in the (1, 5) classification with MNIST dataset [as shown on Fig. 11(a)], it achieves the largest improvement of 41.72% compared to classical deep neural networks, DNN-Fair (3145 parameters), with an accuracy of 99.79%. While classical DNN can achieve perfect accuracies on the MNIST dataset, it requires a much larger parameter size. By introducing 5-qubits, co-TenQu is able to achieve better or similar performance with 49.54% less parameters.

When compared to quantum-based solutions with MNIST dataset, co-TenQu outperforms others, with the largest margin achieved in the (3, 8) and (3, 9) classification, where we observe improvements of 35.07% and 30.71% over Tensorflow Quantum and QF-pNet. One noticeable thing is that, if we train Tensorflow Quantum with 17 qubits (verus 5 qubits), the accuracies increase substantially. For example, the accuracy boosted to from 71.25% to 90.63%. The primary difference between the designs is that co-TenQu utilizes a quantum-state-based evaluation function that can directly train the network on qubits and provide stable results. co-TenQu also outperformed its predecessor, PCA-QuClassi with MNIST dataset. While both employ a quantum-state-based evaluation function, co-TenQu incorporates a new trainable TN layer, allowing part of the training job to be completed on the classical part of the collaborative architecture.

A similar trend is discovered with both Fashion and Extended MNIST datasets, as illustrated on Fig. 11(b) and (c). We can see that co-TenQu outperforms all other solutions in compared two-digit combinations. Comparing the results across three different datasets, TensorFlow Quantum's performance is not stable. For example, it achieves 62.58%, 84.08%, and 66.25% for (3, 8) classification that is a 21.50% difference between datasets. With co-TenQu, however, the same value is 1.55% with 97.65%, 99.20%, and 98.54% for original, Fashion and Extended MNIST datasets, respectively. co-TenQu also beats PCA-QuClassi with 5-qubit setting (shown as PCA-5 on the figures) in all binary combinations with the largest gain, 26.58%, observes at (3, 6) Fashion MNIST [Fig. 11(b)]. This is due to the fact that co-TenQu utilizes the classical computational resource to partially complete training and preprocess the data for quantum parts.

Furthermore, we find that co-TenQu converges faster than PCA-QuClassi when taking a closer look at the training processes. Fig. 12 presents the accuracy per each epoch of (1,5) classification on Extended MNIST dataset. co-TenQu reaches 93.75% at its their epoch, after which it increases 5.10% to 98.85% at the 40th epoch. Comparing with PCA-QuClassi with the 5-qubit setting, however, it records a 87.95% accuracy at the 18th epoch and climbs up to 93.30% at the end, a 5.35% increase. Given the training process, co-TenQu converges significantly faster than PCA-QuClassi as it leverages trainable layers on the classical part.

B. QUANTUM MULTICLASS CLASSIFICATION

Next, we evaluate our solution with multiclass classifications. In these experiments, co-TenQu utilizes a 7-qubit setting. The results demonstrate that co-TenQu provides substantially better multiclass classification accuracies when comparing with the state of the arts. With the multiclass

IEEE Transactions on Quantum Engineering

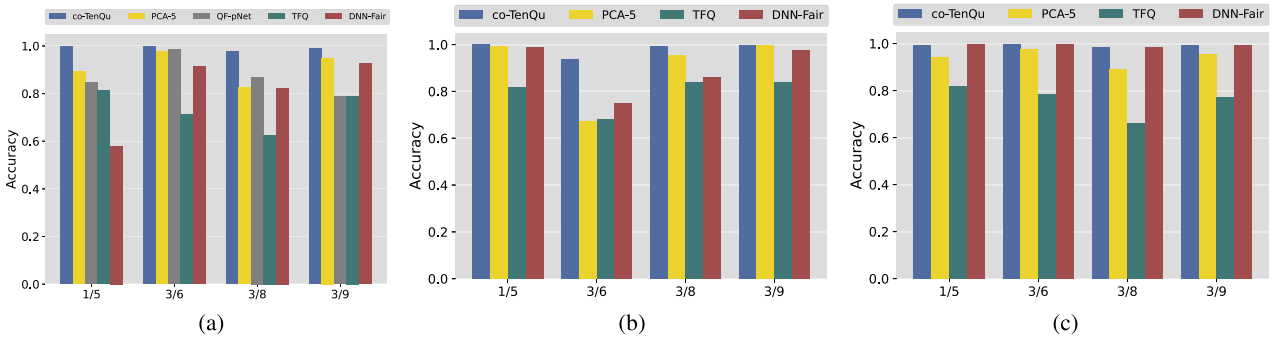


FIG. 11. Binary classifications with 5-qubit circuits for co-TenQu. (a) MNIST. (b) Fashion MNIST. (c) Extended MNIST.

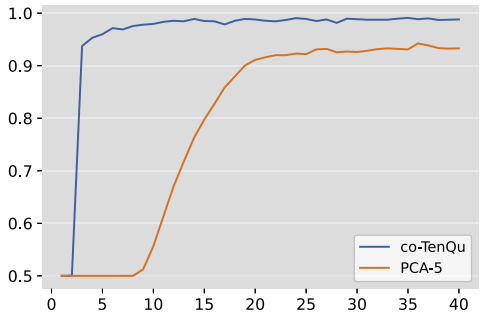


FIG. 12. 1/5 Extended MNIST training.

classification, we select the popular digit combinations, (0, 3, 6), (1, 3, 6), (0, 1, 3, 6, 9), and 10-class, in literature. The results are illustrated in Fig. 13. On the figure, we observe that co-TenQu consistently outperforms other solutions. It achieves 97.39%, 98.94%, and 91.48% for the first three multiclass experiments. PCA-QuClassi with the same 7-qubit setting records 58.55%, 67.68%, and 62.02%. It demonstrates that co-TenQu gains superior performance improvement, up to 66.3%, by introducing the quantum-classical collaborative training architecture. When increase the qubits utilization of PCA-QuClassi to the 17-qubit setting (shown as PCA-17 on the figures), its performance boosts to 94.91%, 94.18%, and 92.49% such that co-TenQu wins the first two experiments, but fails the last one by 1%. It further proves that co-TenQu is able to achieve similar performance with 70.59% less quantum resources (5 versus 17). Considering 10-class experiment, co-TenQu performs significantly better PCA-QuClassi 7-qubit setting (73.21% versus 33.41%), but slightly worse than its 17-qubit version by 5%. The reason lies in the fact that 17-qubit setting contains much more information for the training.

When comparing with QF-pNet, co-TenQu improves the accuracies in all experiments. For example, co-TenQu achieves 97.39% and 98.94% 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 23.75% and 14.38%. In 5-class classification, co-TenQu gains 19.92% (91.48% versus 71.56%). As the number of classes increase, co-TenQu outperforms QF-pNet by more than 181.90% (73.21% versus 25.97%) for 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 co-TenQu, however, we employ a quantum-state-based evaluation function that can fully utilize the qubits and a collaborative training architecture.

We further compare co-TenQu with PCA-QuClassi under the same 7-qubit setting with Fashion and Extended MNIST datasets. The same trend can be found on the Fig. 13(b) and (c), where co-TenQu consistently outperforms its predecessor. It achieves the largest gain on (1, 3, 6) classification with Extended MNIST that is 99.06% comparing with PCA-7's 50.90%. co-TenQu achieves stable performance on all 3-class and 5-class classifications across different datasets. For example, the accuracies for (0, 3, 6), (1, 3, 6), and (0, 1, 3, 6, 9) on Extended MNIST are 98.16%, 99.06%, and 94.88%, respectively. With the 10-class job, the values drop to 73.40% and 63.38% for Fashion and Extended MNIST, respectively. However, co-TenQu utilizes merely 7 qubits and performs much better, up to 1.90×, than PCA-QuClassi.

C. EXPERIMENTS ON IBM-Q PLATFORM

As a proof of concept, we evaluate co-TenQu on real quantum computers through the IBM-Q platform. 300 data points of the (1, 5) and (3, 6) MNIST experiments are submitted to 14 of IBM-Q's superconducting quantum computers. Circuits are generated based off of a trained co-TenQu network, whereby 300 circuits are submitted per machine in one job at 8192 shots each. The results are demonstrated in Fig. 14. Eight of the 14 machines generate a 66.67% accuracy, which is the accuracy of the experiment for assuming all 0's (i.e., ground state). Variational parameters from simulation can perform poorly on real machines, with problems, such as temporal drift and machine specific bias causing induction issues [39]. Within tested machines, IBMQ-Lima achieved the best results, at 82.10%. Lima's topology is drawn in Fig. 15, and has a Quantum Volume of 8, one of the lowest of IBM machines. This highlights the complexity that is predicting machine performance of quantum routines,

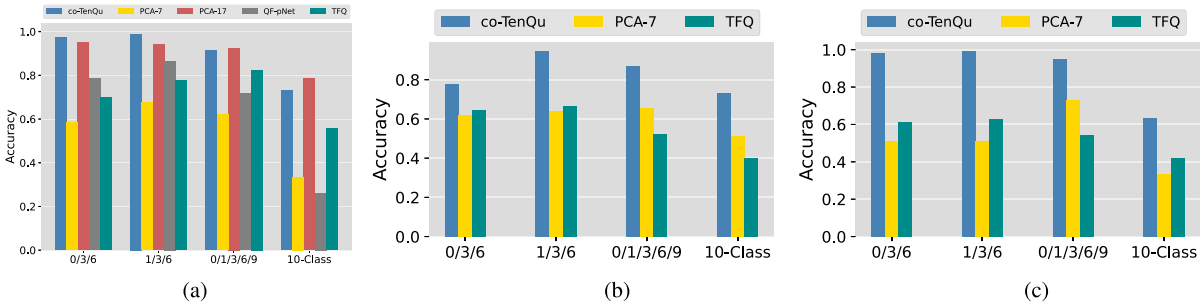


FIG. 13. Multiclass classifications with 7-qubit circuits for co-TenQu . (a) MNIST. (b) Fashion MNIST. (c) Extended MNIST.

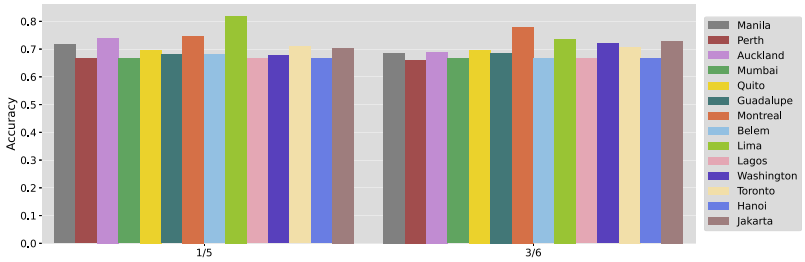


FIG. 14. (1, 5) and (3, 6) MNIST binary classifications on IBM-Q quantum.

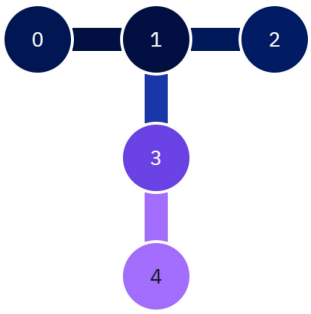


FIG. 15. IBM-Q Lima topology.

and the implications that temporal drift has on learned parameters. Therefore, given sufficient resource, performance can be improved by optimizing the trained network locally, and finalizing training on the processor to learn the machine specific biases.

VI. CONCLUSION

In this work, we propose co-TenQu, a collaborative quantum-classic architecture for QNNs. On the classical side, it utilizes a TN with trainable layers to preprocess the dataset to extract features and reduce the dimensionality. On the quantum part, it employs the quantum-state fidelity-based cost function to train the model. Comparing to classical deep neural networks, co-TenQu achieves 41.72% accuracy improvement with a 49.54% reduction in the parameter count. In addition, it outperforms other quantum-based solutions, up to 1.9 times, in multiclass classification. Furthermore, it records similar performance with 70.59% less quantum resources.

co-TenQu represents a notable advancement in the realm of quantum deep learning. However, there remains considerable room for progress. Due to the limitations of current quantum machines, the existing solutions can only be evaluated on small dataset such as MNIST. In addition, the 10-class classification of MNIST resulted in a 73.21% accuracy, which is relatively modest in comparison to classical counterparts. Although classical methods employ a higher number of parameters, they achieve accuracies approaching 100%, which highlights the potential benefits that quantum computing could offer.

Our future research will concentrate on extending the quantum-state fidelity-based cost function and collaborative quantum-classical architecture to other applications, such as quantum transformers and quantum natural language processing. In addition, exploring the low-qubit representation and its resilience to dynamic noises in the field of quantum-based learning warrants further investigation.

REFERENCES

[1] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The computational limits of deep learning," 2020, *arXiv:2007.05558*, doi: 10.48550/arXiv.2007.05558.
[2] I. L. Chuang, N. Gershenfeld, and M. Kubinec, "Experimental implementation of fast quantum searching," *Phys. Rev. Lett.*, vol. 80, pp. 3408–3411, Apr. 1998, doi: 10.1103/PhysRevLett.80.3408.
[3] [Online]. Available: https://quantum-computing.ibm.com/
[4] F. Arute et al., "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, pp. 505–510, 2019, doi: 10.1038/s41586-019-1666-5.

[5] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997, doi: 10.1137/0097539795293172.

[6] L. K. Grover, "A fast quantum mechanical algorithm for database search," 1996, *arXiv:quant-ph/9605043*, doi: 10.48550/arXiv.quant-ph/9605043.

[7] S. Garg and G. Ramakrishnan, "Advances in quantum deep learning: An overview," 2020, *arXiv:2005.04316*, doi: 10.48550/arXiv.2005.04316.

[8] K. Beer et al., "Training deep quantum neural networks," *Nature Commun.*, vol. 11, no. 1, 2020, Art. no. 808, doi: 10.1038/s41467-020-14454-2.

[9] I. Kerenidis, A. Luongo, and A. Prakash, "Quantum expectation-maximization for Gaussian mixture models," 2019, *arXiv:1908.06657*, doi: 10.48550/arXiv.1908.06657.

[10] T. Li, S. Chakrabarti, and X. Wu, "Sublinear quantum algorithms for training linear and kernel-based classifiers," 2019, *arXiv:1904.02276*, doi: 10.48550/arXiv.1904.02276.

[11] C. Ding, T.-Y. Bao, and H.-L. Huang, "Quantum-inspired support vector machine," 2019, *arXiv:1906.08902*, doi: 10.48550/arXiv.1906.08902.

[12] A. Panahi, S. Saeedi, and T. Arodz, "word2ket: Space-efficient word embeddings inspired by quantum entanglement," 2019, *arXiv:1911.04975*, doi: 10.48550/arXiv.1911.04975.

[13] P. Kaye, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing*. London, U.K.: Oxford Univ. Press, 2007, doi: 10.1093/oso/9780198570004.001.0001.

[14] S. A. Stein et al., "Quclassi: A hybrid deep neural network architecture based on quantum state fidelity," *Proc. Mach. Learn. Syst.*, vol. 4, pp. 251–264, 2022, doi: 10.48550/arXiv.2103.11307.

[15] S. A. Stein et al., "A hybrid system for learning classical data in quantum states," in *Proc. IEEE Int. Perform. Comput. Commun. Conf.*, 2021, pp. 1–7, doi: 10.1109/IPCCC51483.2021.9679430.

[16] S. A. Stein et al., "QuGAN: A quantum state fidelity based generative adversarial network," in *Proc. IEEE Int. Conf. Quantum Comput. Eng.*, 2021, pp. 71–81, doi: 10.1109/IPCCC51483.2021.9679430.

[17] S. Y.-C. Chen, C.-M. Huang, C.-W. Hsing, and Y.-J. Kao, "An end-to-end trainable hybrid classical-quantum classifier," *Mach. Learn. Sci. Technol.*, vol. 2, no. 4, 2021, Art. no. 045021, doi: 10.1088/2632-2153/ac104d.

[18] T. Hur, L. Kim, and D. K. Park, "Quantum convolutional neural network for classical data classification," *Quantum Mach. Intell.*, vol. 4, no. 1, 2022, Art. no. 3, doi: 10.1007/s42484-021-00061-x.

[19] E. H. Houssein, Z. Abohashima, M. Elhoseny, and W. M. Mohamed, "Machine learning in the quantum realm: The state-of-the-art, challenges, and future vision," *Expert Syst. Appl.*, vol. 194, 2022, Art. no. 116512, doi: 10.1016/j.eswa.2022.116512.

[20] F. V. Massoli, L. Vadicamo, G. Amato, and F. Falchi, "A leap among quantum computing and quantum neural networks: A survey," *ACM Comput. Surv.*, vol. 55, no. 5, pp. 1–37, 2022, doi: 10.1145/3529756.

[21] M. Cerezo, G. Verdon, H.-Y. Huang, L. Cincio, and P. J. Coles, "Challenges and opportunities in quantum machine learning," *Nature Comput. Sci.*, vol. 2, no. 9, pp. 567–576, 2022, doi: 10.1038/s43588-022-00311-3.

[22] W.-L. Chang and A. V. Vasilakos, *Fundamentals of Quantum Programming in IBM's Quantum Computers*. Berlin, Germany: Springer, 2021, doi: 10.1007/978-3-030-63583-1.

[23] A. D'Onofrio et al., "Distributed quantum learning with co-management in a multi-tenant quantum system," in *Proc. IEEE Int. Conf. Big Data*, 2023, pp. 221–228, doi: 10.1109/BigData59044.2023.10386676.

[24] A. Abbas, D. Sutter, C. Zoufal, A. Lucchi, A. Figalli, and S. Woerner, "The power of quantum neural networks," *Nature Comput. Sci.*, vol. 1, no. 6, pp. 403–409, 2021, doi: 10.1038/s43588-021-00084-1.

[25] Z. Liang et al., "Variational quantum pulse learning," in *Proc. IEEE Int. Conf. Quantum Comput. Eng.*, 2022, pp. 556–565, doi: 10.1109/QCE53715.2022.00078.

[26] P. Easom-McCaldin, A. Bouridane, A. Belatreche, R. Jiang, and S. Al-Maadeed, "Efficient quantum image classification using single qubit encoding," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 2, pp. 1472–1486, Feb. 2024, doi: 10.1109/TNNLS.2022.3179354.

[27] MNIST. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[28] E. Farhi and H. Neven, "Classification with quantum neural networks on near term processors," 2018, *arXiv:1802.06002*, doi: 10.48550/arXiv.1802.06002.

[29] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, "Quantum circuit learning," *Phys. Rev. A*, vol. 98, no. 3, 2018, Art. no. 032309, doi: 10.1103/PhysRevA.98.032309.

[30] M. Ostaszewski, L. M. Trenkwalder, W. Masarczyk, E. Scerri, and V. Dunjko, "Reinforcement learning for optimization of variational quantum circuit architectures," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 18182–18194, doi: 10.1038/s43588-021-00084-1.

[31] J. Stokes, J. Izaac, N. Killoran, and G. Carleo, "Quantum natural gradient," *Quantum*, vol. 4, May 2020, Art. no. 269, doi: 10.22331/q-2020-05-25-269.

[32] I. Cong, S. Choi, and M. D. Lukin, "Quantum convolutional neural networks," *Nature Phys.*, vol. 15, no. 12, pp. 1273–1278, Aug. 2019, doi: 10.1038/41567-019-0648-8.

[33] S. Stein, Y. Mao, J. Ang, and A. Li, "QuCNN: A quantum convolutional neural network with entanglement based backpropagation," in *Proc. IEEE/ACM 7th Symp. Edge Comput.*, 2022, pp. 368–374, doi: 10.1109/SEC54971.2022.00054.

[34] W. Jiang, J. Xiong, and Y. Shi, "A co-design framework of neural networks and quantum circuits towards quantum advantage," *Nature Commun.*, vol. 12, no. 1, Jan. 2021, Art. no. 579, doi: 10.1038/41467-020-20729-5.

[35] M. Broughton et al., "TensorFlow quantum: A software framework for quantum machine learning," 2020, *arXiv:2003.02989*, doi: 10.48550/arXiv.2003.02989.

[36] S. Y.-C. Chen, C.-M. Huang, C.-W. Hsing, and Y.-J. Kao, "Hybrid quantum-classical classifier based on tensor network and variational quantum circuit," 2020, *arXiv:2011.14651*, doi: 10.48550/arXiv.2011.14651.

[37] V. Bergholm et al., "PennyLane: Automatic differentiation of hybrid quantum-classical computations," 2018, *arXiv:2011.14651*, doi: 10.48550/arXiv.1811.04968.

[38] Tensorflow quantum fair comparison. Accessed: Mar. 07, 2023. [Online]. Available: https://www.tensorflow.org/quantum/tutorials/mnist

[39] S. Stein et al., "EQC: Ensembled quantum computing for variational quantum algorithms," in *Proc. 49th Annu. Int. Symp. Comput. Architecture*, 2022, pp. 59–71, doi: 10.1145/3470496.3527434.

Ryan L'Abbate received the bachelor's degrees in chemical engineering and mathematics from Manhattan College, Bronx, NY, in 2017, and the master's degree in data science from Fordham University, Bronx, in 2022.

He is currently a Database Developer and Research Chemist with En-Tech Corp, Closter, NJ, USA. He has authored or coauthored data science research in the *Haseltonia* journal. His research interests include quantum computing, quantum data science, and data structures.

Mr. L'Abbate was inducted into the Omega Chi Epsilon honors society for chemical engineering and the Pi Mu Epsilon honors society for mathematics.

Anthony D'Onofrio Jr. received the bachelor's and master's degrees in computer science from Fordham University, Bronx, NY, USA, in 2022 and 2023, respectively.

During his time at Fordham, he was a Fordham-IBM research intern. His research interests include distributed systems, quantum systems, quantum deep learning, and software engineering.

Mr. D'Onofrio Jr. was as an Associate Member of the Chapter of Sigma Xi, the Scientific Research Honor Society, at Fordham University's for his work.

Samuel Stein received the bachelor's degree in chemical engineering from the University of Cape Town, Cape Town, South Africa, in 2018, and the master's degree in data science from Fordham University, Bronx, NY, USA, in 2020.

He has been a Staff Scientist with the High-Performance Computing (HPC) group, Pacific Northwest National Laboratory, Richland, WA, USA, since December 2022. His research interests include quantum machine learning, quantum error mitigation, and distributed quantum computing. More recently, his research has focused on heterogeneous quantum computing designs, and distributed quantum computing architectures.

Samuel Yen-Chi Chen received the B.S. degree in physics, the M.D. degree in medicine and the Ph.D. degree in physics from National Taiwan University, Taipei, Taiwan, in 2016 and 2020, respectively.

He was an Assistant Computational Scientist with Brookhaven National Laboratory, Upton, NY, USA. His research interests include combining quantum computing and machine learning.

Dr. Chen was the recipient of Theoretical High-Energy Physics Fellowship from Chen Cheng Foundation in 2014 and First Prize in the Software Competition (Research Category) from Xanadu Quantum Technologies in 2019.

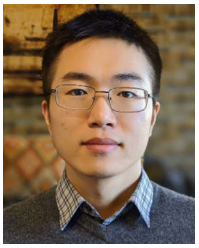
Ang Li received the B.E. degree in computer science from Zhejiang University, Hangzhou, China, and two Ph.D. degrees in electrical and computer engineering from the National University of Singapore, Singapore, and the Eindhoven University of Technology, Eindhoven, The Netherlands, in 2016.

He is currently a Senior Computer Scientist with the Physical and Computational Directorate, Pacific Northwest National Laboratory, Richland, WA, USA, and an affiliated Associate Professor with the University of Washington, Seattle, WA, USA. His research interests include software-hardware codesign for scalable heterogeneous high-performance computing and quantum computing.

Pin-Yu Chen (Senior Member, IEEE) received the Ph.D. degree in electrical engineering and computer science from the University of Michigan, Ann Arbor, MI, USA, in 2016.

He is currently a Principal Research Staff Member with IBM Research, Yorktown Heights, NY, USA, where he is also the Chief Scientist with RPI-IBM AI Research Collaboration and PI with the ongoing MIT-IBM Watson AI Lab projects. He has authored or coauthored more than 50 papers related to trustworthy machine learning at major AI and machine learning conferences. His recent research interests include adversarial machine learning and robustness of neural networks. His long-term research vision is to build trustworthy machine learning systems. His research works contribute to IBM open-source libraries including Adversarial Robustness Toolbox (ART 360) and AI Explainability 360 (AIX 360).

Dr. Chen is an Associate Editor for *Transactions on Machine Learning Research*. He was the recipient of the honor of IBM Master Inventor at IBM Research, and several research accomplishment awards, including an IBM Master Inventor and IBM Corporate Technical Award in 2021.



Juntao Chen (Member, IEEE) received the B.Eng. degree in electrical engineering and automation with honors from Central South University, Changsha, China, in 2014, and the Ph.D. degree in electrical engineering from New York University (NYU), Brooklyn, NY, USA, in 2020.

He is currently an Assistant Professor with the Department of Computer and Information Sciences and an affiliated faculty member with the Fordham Center of Cybersecurity, Fordham University, Bronx, NY, USA. His research interests include cyber-physical security and resilience, quantum artificial intelligence and its security, game and decision theory, network optimization and learning.

Dr. Chen was the recipient of the Ernst Weber Fellowship, the Dante Youla Award, and the Alexander Hessel Award for the Best Ph.D. Dissertation in Electrical Engineering from NYU.

Ying Mao received the Ph.D. degree in computer science from the University of Massachusetts, Boston, MA, USA, in 2016.

He was a Fordham-IBM Research Fellow. He is currently an Associate Professor with the Department of Computer and Information Science, Fordham University, Bronx, NY, USA. His research interests include quantum systems, quantum deep learning, quantum-classical optimizations, quantum system virtualization, cloud resource management, data-intensive platforms, and containerized applications.