Scalable Quantum Neural Networks for Classification

Jindi Wu
Department of Computer Science
William & Mary
Williamsburg, VA, USA
jwu21@wm.edu

Zeyi Tao
Department of Computer Science
William & Mary
Williamsburg, VA, USA
ztao@cs.wm.edu

Qun Li
Department of Computer Science
William & Mary
Williamsburg, VA, USA
liqun@cs.wm.edu

***Abstract*—Many recent machine learning tasks resort to quantum computing to improve classification accuracy and training efficiency by taking advantage of quantum mechanics, known as quantum machine learning (QML). The variational quantum circuit (VQC) is frequently utilized to build a quantum neural network (QNN), which is a counterpart to the conventional neural network. Due to hardware limitations, however, current quantum devices only allow one to use few qubits to represent data and perform simple quantum computations. The limited quantum resource on a single quantum device degrades the data usage and limits the scale of the quantum circuits, preventing quantum advantage to some extent. To alleviate this constraint, we propose an approach to implementing a scalable quantum neural network (SQNN) by utilizing the quantum resource of multiple small-size quantum devices cooperatively. In an SQNN system, several quantum devices are used as *quantum feature extractors*, extracting local features from an input instance in parallel, and a quantum device works as a *quantum predictor*, performing prediction over the local features collected through classical communication channels. The quantum feature extractors in the SQNN system are independent of each other, so one can flexibly use quantum devices of varying sizes, with larger quantum devices extracting more local features. Especially, the SQNN can be performed on a single quantum device in a modular fashion. Our work is exploratory and carried out on a quantum system simulator using the TensorFlow Quantum library. The evaluation conducts a binary classification on the MNIST dataset. It shows that the SQNN model achieves a comparable classification accuracy to a regular QNN model of the same scale. Furthermore, it demonstrates that the SQNN model with more quantum resources can significantly improve classification accuracy.**

***Index Terms*—Quantum Machine Learning, Quantum Neural Networks, Variational Quantum Circuits, Distributed Quantum computing**

I. Introduction

Quantum machine learning (QML) is a revolutionary approach combining machine learning (ML) and quantum computing [1]–[3]. The explosion of data volume and memory consumption makes the ML algorithms run on classical computers unsupportable, although the massive data is desired for ML algorithms to train models. On the contrary, quantum computing based on the law of quantum mechanics (e.g., superposition, entanglement and teleportation) excels at efficiently processing information and outperforms classical computing [4], [5]. For example, Shor's algorithm is sub-exponentially faster in factoring [6] and Grover's algorithm is quadratically faster in searching [7]. Therefore, the combination of ML and quantum computing is a natural trend.

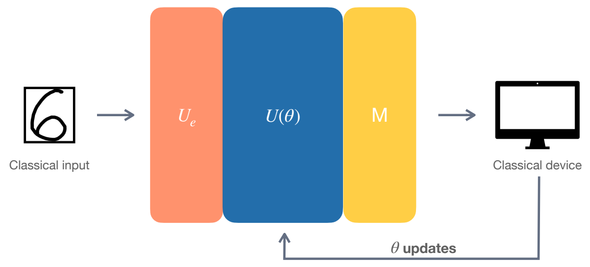


Fig. 1. General QNN in hybrid quantum-classical architecture. The encoding unit U_e prepares quantum states for classical input data. The VQC $U(\theta)$ process the prepared quantum states with variational parameters θ, which will be updated by a classical device according to the circuit output measured by M.

Numerous applications of classical neural networks (NNs) have achieved huge success, which motivates many research into quantum neural networks (QNNs) [8]–[13]. Considering the current Noisy Intermediate-Scale Quantum (NISQ) devices are prone to many noises, the variational quantum circuit (VQC) is the general approach to building QNNs on such devices [14], [15]. VQC, which consists of a set of quantum gates with trainable parameters, is a counterpart of classical NN made up of neurons. The QML algorithm implemented with VQC is the hybrid quantum-classical method that jointly employs quantum and classical devices, as shown in Fig. 1. This method trains a QNN model on a classical dataset by running its quantum circuit on a quantum device, which prepares qubits and modifies their quantum states using quantum gates (unitary transformations). And based on the measurement results of the qubits, calculating the updates to VQC's parameters on a classical device using a classical optimizer, e.g., stochastic gradient descent (SGD) [16], [17]. More details of QNN will be provided in Sec. II.

A barrier to achieving quantum supremacy is the limited number of qubits and connectivities on NISQ devices. With such limitations, only small-scale QNNs can be constructed, and they are unable to load the high-dimensional classical data. Hence, many efforts have been made to overcome this limitation, including quantum encoding and data dimension reduction methods. The amplitude encoding method can load 2^n-dimensional classical data on n qubits by taking the advantage of quantum entanglement and superposition, but its hardware implementation is inefficient and may induce additional errors. Angle encoding is an alternative method that has easier hardware implementation, but it can only load n-dimensional data on n qubits. The encoding methods involve a trade-off between representation capability and hardware implementation. Therefore, quantum encoding is still an open problem. Moreover, Chen et al. use a pre-trained classical NN to extract features from original data in order to fit the data dimension to the number of available qubits on quantum devices [18]. And Steni et al. use principal component analysis (PCA) [19] to downscale original data [20]. Nevertheless, These methods that require additional computation undermine the advantages of quantum acceleration. This unsolved problem motivates us to design a QNN system that is not limited by the size of current quantum devices and is easy to scale up for high-dimensional classical data. We name it as Scalable Quantum Neural Network (SQNN).

An SQNN system combines the quantum resources of multiple small-size quantum devices to simulate a large quantum device with sufficient resources and trains a large-scale QNN for high-dimensional data. Suppose there are five available small-size quantum devices in a quantum system for an image (high-dimensional data) classification task. Four of them work as *quantum feature extractors* and the remaining one acts as a *quantum predictor*. The SQNN system partitions a training instance into four segments that can fit the capacity of the quantum feature extractors and assigns them to quantum feature extractors. Each quantum feature extractor encodes the received classical data to quantum data, uses a VQC to reduce the original data size and represent the information as abstract features, then obtains the extracted features by measuring the readout qubits of the circuit. The quantum predictor collects extracted local quantum features from quantum feature extractors via classical communication channels, learns from them, and makes prediction on them with a VQC. So far, the SQNN system could obtain the prediction result by measuring the readout qubits of the quantum predictor's VQC. Then like the optimization of the classical NNs, a classical device calculates the updates of the parameters in VQCs according to the prediction results and the pre-defined objective function. The updates are further used to set the VQCs' parameters for the next training step.

The SQNN system enables the QNN models to learn from high-dimensional data by scaling QNNs up. Furthermore, the SQNN system supports a flexible data partition strategy to fully utilize the quantum system's resources. With the strategies, SQNN assigns the high-dimensional data segments of different sizes to quantum feature extractors according to their various amount of quantum resources. Fig. 3 illustrates the possible input partition strategies. Especially, a quantum system with fewer or even a single quantum device can also train an SQNN. After partitioning the training instance into several segments, one can repeatedly use the available quantum devices as quantum feature extractors to act on data segments with recording the intermediate result, then as the quantum predictor.

The major contributions of this paper as following:

- We present the first, to the best of our knowledge, scalable quantum neural network (SQNN) that can learn from high-dimensional data without being constrained by the quantum hardware limitation on qubit amount.
- We propose a scalable quantum ML approach that collaboratively uses the quantum resources of multiple small-size quantum devices.
- We implement a SQNN for a binary classification task and conduct extensive evaluations to show the effectiveness of the proposed design.

In the rest of this paper, we will recap some background knowledge about quantum gates, quantum encoding methods, and QNN in Sec. II. We will introduce the proposed approach in Sec. III, and show the evaluations in Sec. IV. Some related works will be reviewed in Sec. V. At last, we will conclude our work in Sec. VI.

II. Preliminary

A. *Quantum gates*

Qubit is the basic unit of quantum computation. Compared with a classical bit that can only represent state 0 or 1, a qubit can simultaneously represent state 0 and 1 with a certain probability distribution. The state of a qubit is denoted as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \text{ with } \alpha, \beta \in \mathbb{C}$$

where $|\alpha|^2$ and $|\beta|^2$ respectively express the probability of the qubit is measured as state 0 and 1, i.e., $|\alpha|^2 + |\beta|^2 = 1$. The qubit state is also written as a vector $|\psi\rangle = [\alpha, \beta]^\top$.

The quantum computation is conducted by changing the state of the qubits with quantum gates U (unitary matrices that satisfy $U^\dagger U = UU^\dagger = I$). The Pauli gates $\{X, Y, Z\}$ and the Hadamard gate H is the fundamental single-qubit gate. Their matrices are shown as

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$
$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

The Pauli gates X, Y and Z rotate the qubit by π radians around the X, Y and Z-axis of the Bloch sphere (a visual representation of quantum state). And the Hadamard gate H creates an equal superposition state for the computational basis states: $|0\rangle \rightarrow (|0\rangle + |1\rangle)/\sqrt{2}$ and $|1\rangle \rightarrow (|0\rangle - |1\rangle)/\sqrt{2}$. It is a $\pi/2$ rotation around the Y-axis followed by a π rotation around the X-axis.

Based on the Pauli gates, the single-qubit rotation gates are generated that allow the qubit to be rotated arbitrary radians in the Bloch sphere. The $R_x(\theta)$, for example, rotates the qubit by θ radians around the X-axis in the Bloch sphere as

$$R_x(\theta) = \exp\left(-iX\frac{\theta}{2}\right) = \begin{bmatrix} \cos(\frac{\theta}{2}) & -i\sin(\frac{\theta}{2}) \\ -i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix}$$

The same for $R_y(\theta)$ and $R_z(\theta)$. Ising coupling gates are two-qubit gates expended from single-qubit rotation gates, e.g.,

$$R_{xx}(\theta) = \exp\left(-iX\otimes X\frac{\theta}{2}\right) = \begin{bmatrix} \cos(\frac{\theta}{2}) & 0 & 0 & -i\sin(\frac{\theta}{2}) \\ 0 & \cos(\frac{\theta}{2}) & -i\sin(\frac{\theta}{2}) & 0 \\ 0 & -i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) & 0 \\ -i\sin(\frac{\theta}{2}) & 0 & 0 & \cos(\frac{\theta}{2}) \end{bmatrix}$$

It performs the same rotation on two qubits simultaneously. And the same for $R_{yy}(\theta)$ and $R_{zz}(\theta)$. In QML, Ising coupling gates are frequently used to construct VQC [21].

B. *Quantum encoding*

To be processed on quantum devices, classical data must be encoded into quantum states in Hilbert space. Because current quantum devices have limitations on the number of qubits and the depth of circuits, the encoding methods impact the efficiency of hardware implementation and the design of the following quantum information processing circuits. This subsection will introduce three mainly used quantum encoding schemes [22].

1) Basis encoding: The basis encoding method is the most straightforward quantum encoding method for arithmetic operation. Basis encoding first represents classical data in binary form and directly maps them onto quantum computational bases. For example, a numerical data point [0.3, 0.6, 0.2, 0.8] will be converted into a binary data point [0, 1, 0, 1] based on a threshold 0.5, and mapped to the 4-qubit quantum computational basis $|0101\rangle$. For the implementation of the encoding circuit, four qubits are prepared with initialized state $|0\rangle$, and an Pauli-X gate is appended after the second and fourth qubits to flip their states to $|1\rangle$. In general, the basis encoding method takes n qubits to load the binary representation of a classical data point $x = (x_1, x_2, \cdots, x_n)$ and encodes it as

$$|\Psi_x\rangle = \bigotimes_{i}^{n} |x_i\rangle$$

where $\bigotimes$ is tensor product operator.

2) Angle encoding: The angle encoding method loads the classical data as the radians of rotation gates acting on qubits. n qubits and n quantum rotation gates $R \in \{R_x, R_y, R_z\}$ are required to embed the classical data in size n. For a classical data instance $x = (x_1, x_2, \cdots, x_n)$, the angle encoding method prepares it as

$$|\Psi_x\rangle = \bigotimes_{i}^{n} R(x_i)|0\rangle$$

3) Amplitude encoding: The amplitude encoding method embeds classical data into the amplitudes of a quantum state. An n-dimensional normalized classical vector $x = (x_1, x_2, \cdots, x_n)$ is encoded to the amplitudes of $\log n$-qubit quantum state by

$$|\Psi_x\rangle = \sum_{i=1}^{n} x_i|i\rangle$$

where $|i\rangle$ is the i-th computational basis state and $\sum |x_i|^2 = 1$.

C. *Quantum neural network*

The current widely used QNN that can run on NISQ devices for classification tasks was proposed in 2018 [10], which is an analogue of the classical NN in the quantum computing field. The QNN is a quantum circuit within a sequence of parameter-dependent quantum gates (unitary operators) which act on quantum input data. Generally, a QNN can be shown as

$$U(\theta) = \prod_{l=1}^{N} V_l U_l(\theta_l) \tag{1}$$

which is a product of N quantum layers [21]. The l-th quantum layer consists of product of non-parametric quantum gates V_l and parametric quantum gates $U_l(\theta_l)$ where θ_l are variational parameters. We can further represent the parametric quantum gates $U_l(\theta_l)$ in l-th layer as the production of S parametric quantum gates,

$$U_l(\theta_l) \equiv \bigotimes_{j=1}^{S} U_{l,j}(\theta_{l,j}) \tag{2}$$

In which each parametric quantum gate $U_{l,j}(\theta_{l,j})$ can be transformed with Euler's formula as

$$\exp(-i\theta_{l,j}P) = I\cos(\theta_{l,j}) - i\sin(\theta_{l,j})P \tag{3}$$

where i is the imaginary number, I is a 2×2 identity matrix, and P is a Pauli operator from the set $\{X, Y, Z\}$ that acts on qubits.

The output of the QNN is the measurement result on a computational basis of the readout qubits. Since the measurement result of a qubit is probabilistic, the expectation value E of the measurement results is used as the QNN output,

$$E = \langle\Psi_x|U^\dagger(\theta)MU(\theta)|\Psi_x\rangle \tag{4}$$

where $|\Psi_x\rangle$ is the input quantum state of the QNN and M is a linear combination of Pauli operators that serve as observables for readout qubits.

The loss L of a training sample in hybrid quantum-classical model is calculated in conventional manner on a classical device. For the given training sample, the loss L is calculated with an objective function $\ell(.)$ of the task on the expected output y and the actual output E

$$L = \ell(E, y) \tag{5}$$

During the model optimization phase, like in the classical NN, back-propagation and gradient descent will be performed to update variational parameters in the QNN. The gradient of

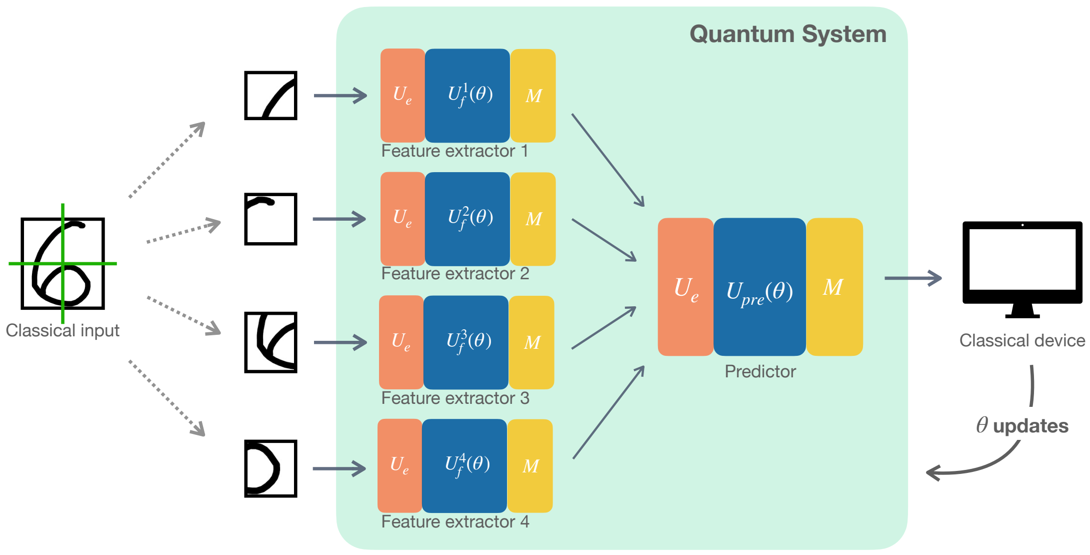


Fig. 2. An SQNN in hybrid quantum-classical architecture. The quantum system contains five small-size quantum devices, four of which work as quantum feature extractors to extract features from the segments of input data, and the remaining one as a predictor to make predictions using the extracted features collected from feature extractors. Each quantum device has a quantum circuit that consists of an encoding unit U_e, a VQC $U(\theta)$, and a measurement unit M. The classical device will assist them in updating their variational parameters during training.

a variational parameter θ_k in k-th quantum layer with respect to with respect to loss L can be calculated by

$$\frac{\partial L}{\partial \theta_k} = \frac{\partial L}{\partial E}\frac{\partial E}{\partial \theta_k} \tag{6}$$

It is easy to obtain $\partial L/\partial E$ according to the objective function $\ell(.)$. $\partial E/\partial \theta_k$ could be calculated by

$$\frac{\partial E}{\partial \theta_k} = i\langle \Psi_x|U_-^\dagger[P_k, U_+^\dagger H U_+]U_-|\Psi_x\rangle \tag{7}$$

where

$$U_+ = \prod_{l=k+1}^{N} V_l U(\theta_l) \text{ and } U_- = \prod_{l=1}^{l=k} V_l U(\theta_l). \tag{8}$$

With the gradients of parameters, the classical device sets updated parameters for QNN using an optimization algorithm, such as SGD. An alternative gradient calculation for a quantum model is parameter-shift, which obtains the gradients by running the same VQCs with shifted parameters and calculating the difference in their outputs [23], [24].

III. Scalable Quantum Neural Network

The limited quantum resources on NISQ devices constrain the scalability of quantum circuits and computational power of quantum computing, particularly for QML tasks that demand extensive computation. In this section, we present a solution, SQNN, to circumvent the hardware constraints of quantum devices. The concept behind SQNN is to construct and train a large-scale VQC by cooperatively using the quantum resource on multiple small-size quantum devices. The architecture overview of the proposed design is shown in Fig. 2. The components of SQNN are detailed in the following subsections. The adopted encoding method is discussed in III-A. The variational quantum layers of SQNN are introduced in III-B. The optimization of SQNN is described in III-C. III-D shows how to build SQNN in small quantum systems. And the important notations are listed in Table I.

Considering a quantum system with $p+1$ small-size quantum devices, we use p of them as *quantum feature extractors* for quantum local feature extraction, and the remaining one as a *quantum predictor* for prediction (classification). Given a classical training dataset in which the instances x are too large to be loaded by a quantum device in the system, we partition x into p small segments of size n using the first partition strategy

TABLE I
Notation List

Notation	**Description**
x	A classical training instance
$\|\Psi_x\rangle$	The quantum state representation of x
f_i	The i-th quantum feature of a training instance
y	The correct label of the training instance
y'	The predicted label of the training instance
θ	The variational parameters of VQC
$R(\theta)$	A rotation gate with θ radians
$U_f^i(\theta)$	The VQC on i-th quantum feature extractor
$U_{pre}(\theta)$	The VQC on quantum predictor
$U_e(x)$	The quantum circuit of the encoding unit
M	The qubit measurement unit

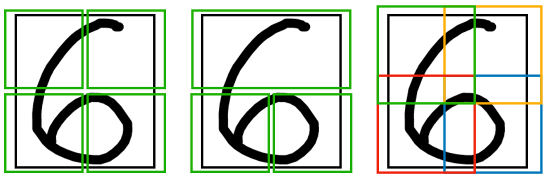


Fig. 3. Input partition strategies. Depending on the size of quantum feature extractors, a high-dimensional input image could be evenly partitioned without overlap (left), unevenly partitioned without overlap (middle), or evenly partitioned with overlap (right).

shown in Fig. 3, where n is less than the number of available qubits on the quantum devices. We denote the instance in segments as $x = \{x^1, x^2, \cdots, x^p\}$ and the i-th segment of instance x as $x^i = \{x^i_1, x^i_2, \cdots, x^i_n\}$.

A. Encoding unit

Both the quantum feature extractors and the quantum predictor in the SQNN system need the encoding unit to transform classical data into quantum data for further quantum operations. We use the angle encoding method in the encoding unit because of its simple hardware implementation and ability to support the gradient calculation for input quantum data, which is necessary for SQNN to perform back-propagation via chain rule.

There are some reasons that the other commonly used encoding methods are not adopted in our work. As mentioned in the Sec. II-B, the basis encoding method embeds numerical data into binary data and is implemented by selectively appending Pauli-X gate behind the qubits with initial state $|0\rangle$. Although the basis encoding method is straightforward, it is not adopted for several reasons. First, it processes the original data with a discontinuous function that does not support the gradient calculation for the input of VQC. Second, the binary representation of an instance will lose much useful information, which is not desirable in ML tasks. And third, multiple numerical instances may map to the exact binary representation, which reduces the amount of training data. The amplitude encoding method is also not used in this work because its hardware implementation is complex and does not offer a simple way to calculate the gradients of quantum circuit input. The details of the calculation and usage of input gradients of quantum circuits in SQNN will be discussed in the Sec. III-C.

The circuit of the quantum angle encoding unit is illustrated in Fig 4. When a quantum device receives the i-th segment of the classical instance x, it flats the segment as a vector $x^i = [x^i_1, x^i_2, \cdots, x^i_n]$ and maps the entries into $[0, 2\pi)$, and prepares n qubits with initial sate $|0\rangle$. The encoding unit sets the state of qubits independently to represent the data using rotation gate R. To be more specific, j-th qubit will pass a rotation

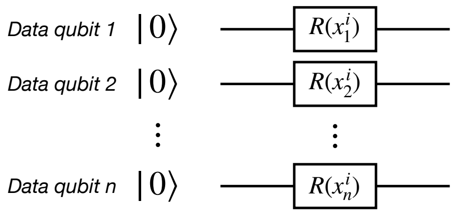


Fig. 4. Angle encoding unit. The classical data vector $x^i = [x^i_1, x^i_2, \cdots, x^i_n]$ is loaded on n data qubits using rotation gate $R \in \{R_x, R_y, R_z\}$.

gate $R(x^i_j)$ to load x^i_j in its state as a certain superposition

$$|\Psi_{x^i_j}\rangle = R(x^i_j)|0\rangle = \cos\left(\frac{x^i_j}{2}\right)|0\rangle - i\sin\left(\frac{x^i_j}{2}\right)|1\rangle \quad (9)$$

Hence, the prepared quantum state for x^i is shown as a tensor product of each qubit state

$$|\Psi_{x^i}\rangle = \bigotimes_j^n |\Psi_{x^i_j}\rangle. \quad (10)$$

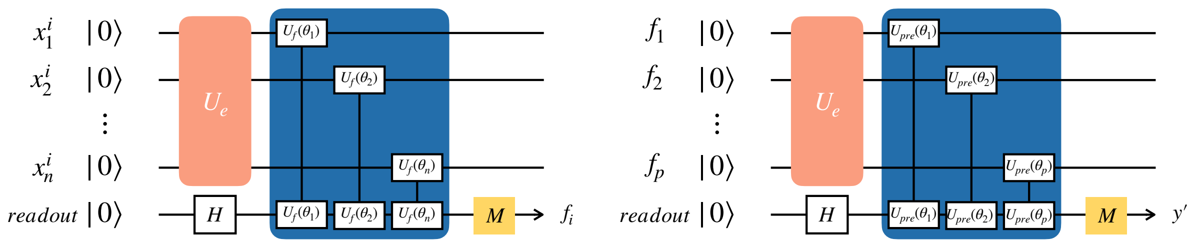
B. Variational quantum layers

A SQNN consists of two types of variational quantum layers: quantum feature extraction layer and quantum prediction layer. They are deployed on quantum feature extractors and a quantum predictor, respectively.

1) Quantum feature extractor: A quantum feature extractor that receives i-th piece of the classical instance x^i of size n will use $n+1$ qubits in its quantum circuits, as shown in Fig. 5(a). The first n qubits are *data qubits* that pass the encoding unit to load the classical data as quantum state $|\Psi_{x^i}\rangle$. And the last one serves as *readout qubit* that will be measured to obtain the results of the quantum data processing. In general, a quantum circuit could have multiple readout qubits, but we only show one out of simplification.

The quantum feature extraction layer is a VQC that maps the quantum features from the data qubits to the readout qubit using two-qubit parameterized quantum gates that create entanglement between them. In this work, we adopt the Ising coupling gates to entangle qubits and follow the same design of VQC introduced in [25]. We define the n gates that consecutively act on n pairs of data and readout qubits as a **block**. Similar to the classical NN, the variational gates works as neurons and a block as a layer of the QNN. The block could be repeated to build a complex QNN with more parameters. We denote the quantum feature extraction layer deployed on i-th quantum feature extractor as $U^i_f(\theta^i_f)$ where θ^i_f is the trainable parameters. With the entanglement with all data qubits, the state of the readout is the extracted quantum features from the segment x^i and will be obtained as a real number f_i by measurement with Eq. 4

$$f_i = \langle\Psi_{x^i}|{U^i_f}^{\dagger}(\theta^i_f) M U^i_f(\theta^i_f)|\Psi_{x^i}\rangle \quad (11)$$



(a) Quantum feature extraction layer (b) Quantum prediction layer

Fig. 5. Two types of quantum layers in SQNN

2) Quantum predictor: The quantum predictor collects extracted local features that represented as real numbers $f = \{f_1, f_2, \cdots, f_p\}$ from p quantum feature extractors via conventional communication channels. The circuit of the predictor uses p data qubits to load extracted local features using the encoding unit and one readout qubit for outputting prediction result, as shown in Fig. 5(b). Generally, the number of readout qubits depends on the classification task, i.e., $\log k$ readout qubits are needed for a k-class classification task. Assuming we are performing a binary classification task, only one readout qubit is used.

The quantum prediction layer is a VQC that jointly learns local features and makes a prediction on them. The design of the VQC is the same as that of the quantum feature extraction layer. We represent the quantum prediction layer as $U_{pre}(\theta_{pre})$ and θ_{pre} stands for the trainable parameters. The prediction result of the instance x is

$$y' = \langle \Psi_f | U_{pre}{}^{\dagger}(\theta_{pre}) M U_{pre}(\theta_{pre}) | \Psi_f \rangle \tag{12}$$

where $|\Psi_f\rangle$ is the quantum state of the extracted features f and y' is a real number in the range of [-1, 1]. For the binary classification with labels -1 and 1, we map the result y' to label -1 if it is negative; otherwise, to label 1.

C. Optimization

After the forward propagation, a classical device receives the prediction result y' of x from the quantum predictor. The classical device will calculate the loss of the current training instance x by

$$L = \ell(y, y') \tag{13}$$

where $\ell(\cdot)$ is a pre-defined loss function, e.g., Mean squared error (MSE) loss, and y is the actual label of x. The classical device with the knowledge of VQCs on quantum devices will optimize the parameters of SQNN. In the following, we will demonstrate the gradient derivation of the quantum prediction layer and quantum feature extraction layers according to back-propagation and the optimization with the gradient descent method.

The gradient of the variational parameters θ_{pre} in the quantum prediction layer with respect to the loss L is obtained by

$$\frac{\partial L}{\partial \theta_{pre}} = \frac{\partial L}{\partial y'} \frac{\partial y'}{\partial \theta_{pre}} \tag{14}$$

where the $\partial y'/\partial \theta_{pre}$ can be calculated with Eq. 7 and $\partial L/\partial y'$ is easily calculated. Then the parameters θ_{pre} will be updated with the learning rate r by

$$\theta_{pre} = \theta_{pre} - r \frac{\partial L}{\partial \theta_{pre}} \tag{15}$$

The gradients of the variational parameters θ_f^i in the i-th quantum feature extraction layer with respect to the loss L can be obtained as follow

$$\frac{\partial L}{\partial \theta_f^i} = \frac{\partial L}{\partial y'} \frac{\partial y'}{\partial f_i} \frac{\partial f_i}{\partial \theta_f^i} \tag{16}$$

The $\partial f_i/\partial \theta_f^i$ can be calculated by Eq. 7 as well. The $\partial y'/\partial f_i$ is the partial derivative of the input local feature f_i with respect to the output of the quantum predictor y'. With the encoding method mentioned above, the input f_i of the quantum prediction layer is encoded in the i-th qubit by using $R(f_i)$. Hence, we have

$$\frac{\partial y'}{\partial f_i} = i\langle 0 | [R, U_{pre}{}^{\dagger}(\theta_{pre}) M U_{pre}(\theta_{pre})] | 0 \rangle. \tag{17}$$

To optimize i-th quantum feature extraction layer, the classical device updates θ_f^i with learning rate r by

$$\theta_f^i = \theta_f^i - r \frac{\partial L}{\partial \theta_f^i}. \tag{18}$$

So far, we have gone through the optimization process of SQNN on a single training instance. To train the SQNN in the mini-batch style, one just needs to record the gradients of parameters for each sample in a mini-batch of data, and make the classical device update the variational parameters using the average of gradients.

D. SQNN in small quantum systems

This section shows how the SQNN can be built and trained in a small quantum system with insufficient quantum devices with the assistance of a classical device. We consider a particular case where only one quantum device is available in the quantum system.

Since the quantum feature extractors work independently, we partition the classical training instance into p segments of the equal size based on the number of qubits on the quantum device, and make the device serve as p feature extractors sequentially. When the quantum device works as the i-th quantum feature extractor, it builds the circuit U_f^i with initialized parameters θ_f^i and performs quantum operations on the data segment x^i. The classical device records the circuit structure, current parameters, and the extracted local feature f_i for it. Once the device has completed the work as quantum feature extractors, it acts as the quantum predictor by initializing the quantum predictor circuit $U_{pre}(\theta_{pre})$ and performing it on the extracted features f. The classical device also records the circuit structure and parameters for it. The classical device then updates the parameters as introduced in III-C. In the following training steps, the quantum device will first reproduce the circuits and reload the latest parameters before repeating the previous step.

In the quantum system that consists of several quantum devices of different sizes, one can divide the classical instance into segments that fit the various sizes of available quantum devices as Fig. 3, to fully use the system's quantum resources.

IV. Experiments and results

The SQNN is evaluated using a binary classification task. We conduct the binary classification using the images labeled "3" and "6" in the MNIST handwritten digits dataset [26]. After removing the images with other labels, we have 12049 training and 1968 test samples. The handwritten digit images in MNIST are pre-processed to position the digits in the center of 28×28 fields. In our experiments, we simulate the circuits of SQNN on a classical device with the TensorFlow Quantum library. And two scenarios of multi-quantum machine systems for SQNN are considered: the SQNNs with quantum feature extractors of the same and different sizes.

Here we define the form of the **basic model** used in our experiments to construct QNN classifiers. For a quantum device with $n+1$ qubits, the basic model is defined as: the first n qubits are data qubits that store quantum data, and the last qubit is the readout that provides the quantum computation result by measurement. In each **block** of the basic model, the entanglement between each data and the readout qubit is created by the same variational Ising coupling gates in $\{R_{xx}(\theta), R_{yy}(\theta), R_{zz}(\theta)\}$ as shown in Fig. 5. The source code used to generate the experiment results is available in github.com/Jindi0/SQNN.

A. Observations

We first show some observations about how the hardware limitations of NISQs impact the performance of regular QNN classifiers. Assuming there are three quantum machines of size $4+1$, $9+1$, and $16+1$ qubits, we implement a QNN classifier with the basic model defined above on each device. The last qubit works as the readout, and the remaining qubits are used to load classical training data with R_x gate in encoding unit described in Sec. III-A. During the encoding phase, the classical handwritten digit images of size 28×28 are downscaled to 2×2, 3×3, and 4×4 sizes and loaded on the three quantum devices, respectively.

Fig. 6(a) demonstrates the performance of the QNN classifiers with three blocks. Each model has been trained 100 epochs, and its best accuracy is indicated by a red dot in the right panel of Fig. 6(a). Among the QNN classifiers with three blocks, *16qb_3blk* model with 16 data qubits has the best accuracy of 92.04% on the validation dataset. The following is the *9qb_3blk* model with 9 data qubits has an accuracy of 88.37%. The *4qb_3blk* model with 4 data qubits has the lowest accuracy of 81.18%. We also evaluate the performance of QNN classifiers with six blocks on these quantum devices and present the results in Fig. 6(b). The best accuracy of the models are 90.99% on *16qb_6blk* model, 89.45% on *9qb_6blk* model and 82.00% on *4qb_6blk* model. We observe that the QNN model that learns from higher dimensional input data can achieve lower training loss and higher accuracy with the same number of blocks based on the comparison among the models with various amounts of data qubits. Additionally, because more parameters need to be trained, the larger model requires more training epochs to reach the point with the best performance.

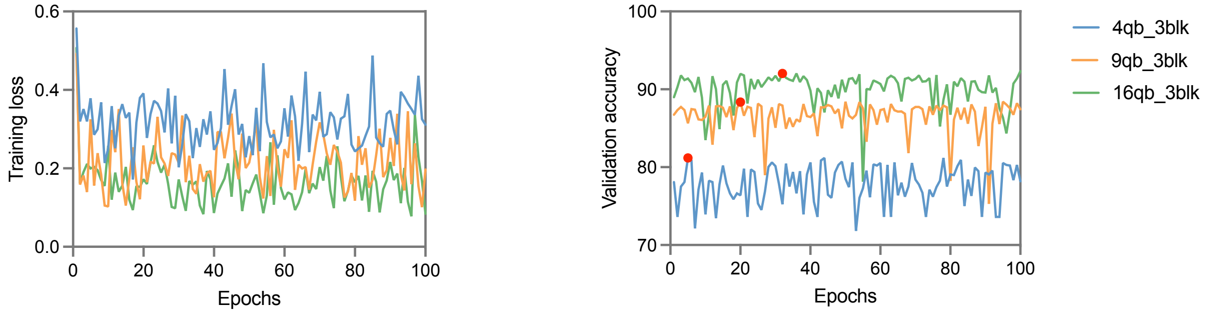
We further compared the classification accuracy of three pairs of QNN classifiers with the same number of data qubits but the different number of blocks. For the QNNs with the same number of data qubits, as shown in Fig. 7, more blocks will not improve the validation accuracy but can improve the stability of the training process. Therefore, we argue that the number of qubits (the dimension of input data) has a more significant impact on the classification accuracy of the QNN classifier than the circuit depth.

B. SQNN with even partitioning

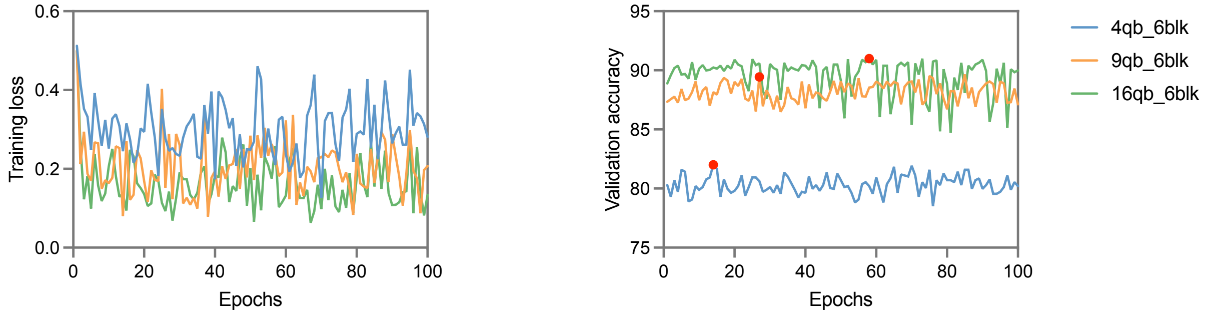
This subsection shows the performance of SQNN-based quantum classifiers. SQNN classifiers are supposed to learn from raw-sized training data. However, very high-dimensional quantum operations cannot be simulated by classical devices. As a result, in our simulator, a VQC made up of basic quantum blocks can only contain at most 17 qubits.

First, we demonstrate the effectiveness of the SQNN classifier by comparing the performance of the SQNN classifier trained on data of the same size with the regular QNN classifier. We implement an SQNN classifier trained on MNIST "3" and "6" images of downscaled size 4 x 4 to compare with the *16qb_3blk* model that achieves the best performance among regular QNN classifiers.

Assuming there are five $(4+1)$-qubit quantum devices in a quantum system, we use four of them as the quantum feature extractors and the remaining one as the quantum predictor. The quantum circuits on quantum feature extractors are structured similarly to the *16qb_3blk* model, and the quantum circuit on the quantum predictor has an encoding unit and a block to incorporate the extracted features. We refer the SQNN model as the *16qb_sqnn* model, which has 16 data qubits in total. We evenly partition the 4x4 downscaled image into four 2x2 segments, as shown in the left panel of Fig. 3. The comparison of the training loss and validation accuracy

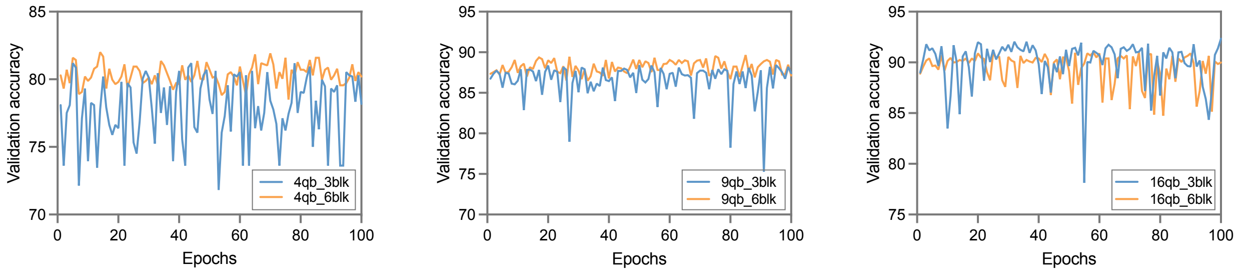


(a) Training loss and validation accuracy of 3-block QNN classifiers



(b) Training loss and validation accuracy of 6-block QNN classifiers

Fig. 6. **The performance of QNN classifiers on a single device.** The single-device QNN models consist of 4, 9, and 16 data qubits (qb). The QNNs in each size are evaluated using three and six variational circuit blocks (blk). Their best accuracy is marked with red dots during 100 epochs of training. These results indicate the QNN model that learns from higher dimensional data can achieve better performance.



(a) Accuracy of 4-qubit QNNs classifiers (b) Accuracy of 9-qubit QNNs classifiers (c) Accuracy of 16-qubit QNNs classifiers

Fig. 7. **The comparison of accuracy between 3-block and 6-block QNN classifiers.** The QNN models with the same amount of data qubits (qb) but the different numbers of variational circuit blocks (blk) are compared. The images demonstrate the models with the same number of qubits achieve similar performance, but the accuracy of the models with six blocks shows less fluctuation.

between the *16qb_3blk* and *16qb_sqnn* model is shown in Fig. 8. *16qb_sqnn* achieves the accuracy of 92.59%, which is comparable with the accuracy achieved by *16qb_3blk*, 92.04%. Thus, the experiment shows the effectiveness of SQNN.

Next, we discuss the performance of SQNN classifiers in quantum systems at different scales. SQNN classifiers are implemented in three scales of quantum systems, and their performance is shown in Fig. 9. Each quantum system considered in this experiment contains a $(4+1)$-qubit quantum predictor and four quantum feature extractors of $(4+1)$-qubit, $(9+1)$-qubit, or $(16+1)$-qubit size, i.e., the data samples are evenly partitioned into four segments as the first panel shown in Fig. 3, then the three SQNN classifiers are trained on 4×4, 6×6, and 8×8 sizes of images, respectively. The SQNN classifier with 64 data qubits (four quantum feature extractors with 16 qubits each) in total is denoted as *64qb_sqnn* model and achieves the best accuracy of 97.47%. The *36qb_sqnn* model with 36 data qubits has accuracy of 95.10% and

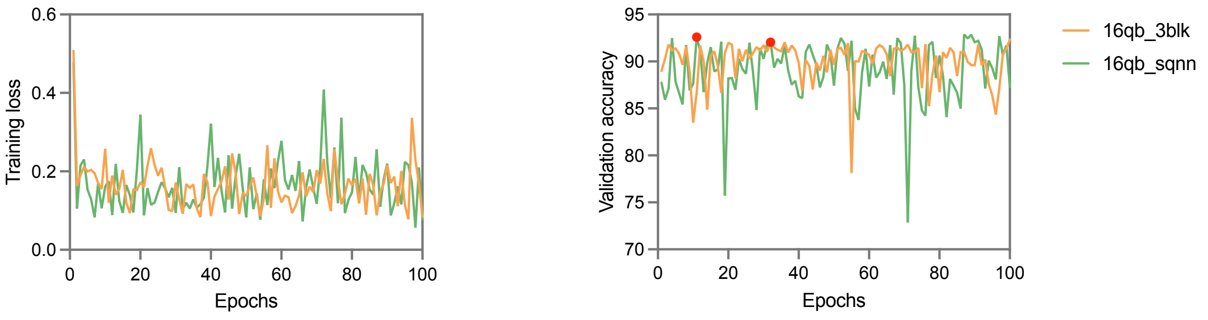


Fig. 8. **The comparison of performance between QNN and SQNN classifiers**. The SQNN and the single-device QNN models both have three variational circuit blocks (blk) and 16 data qubits (qb). Their comparable performance illustrate the effectiveness of the SQNN model.

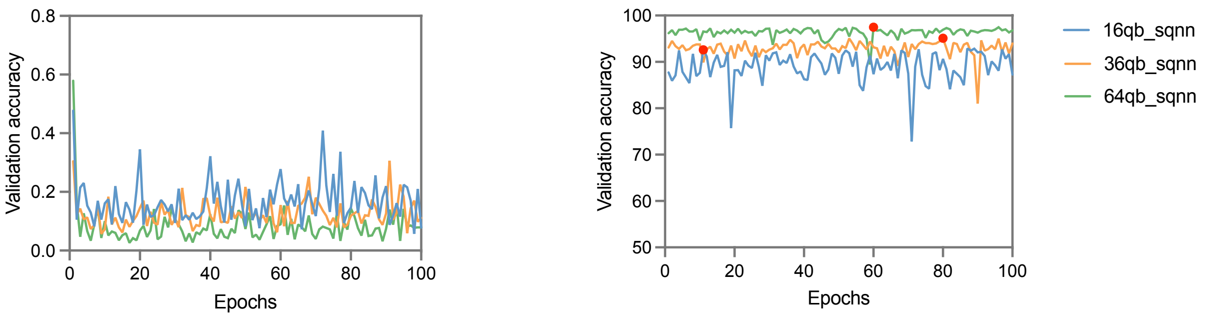


Fig. 9. **The performance of SQNN classifiers in three scales of quantum systems.** Three SQNN models are implemented with 16, 36, and 64 data qubits (qb), and their best accuracy is marked with red dots. The comparison shows that models with more quantum resources can perform better.

16qb_sqnn model with 16 data qubits has accuracy of 92.59%.

Based on the results, one can observe that the SQNN classifiers implemented in the larger-scale quantum systems have better performance, which is consistent with the observations in Sec. IV-A. The *36qb_sqnn* model and *64qb_sqnn* that utilize more quantum resources from multiple quantum devices all achieve higher classification accuracy than *16qb_3blk* model, the best model trained on a single quantum device.

C. SQNN with uneven partitioning

We now assume a quantum system containing various size quantum feature extractors. A quantum system, for example, contains four quantum devices: a $(8+1)$-qubit quantum device, two $(4+1)$-qubit quantum devices, and a $(3+1)$-qubit quantum device. Typically, the smallest device is chosen as the quantum predictor. To make full use of the quantum resources on quantum feature extractors, we downscale the data to 4×4 and partition it into a 2×4 segment and two 2×2 segments, as the second panel in Fig. 3. We then respectively assign them to the $(8 + 1)$-qubit quantum device and the two $(4 + 1)$-qubit quantum devices. Fig. 10 compares the performance of uneven partitioning model *16qb_uneven_sqnn* to the other two models, *16qb_3blk* and *16qb_sqnn*, that are trained with the same size input data. Fig. 10 illustrates that the SQNN with an uneven partitioning strategy performs similarly to the other two models. Hence, we argue that SQNN in the quantum system with quantum devices of different sizes is also effective.

V. Related Work

The topic studied in this paper is known as quantum neural network [27]–[29], a case of variational quantum algorithm [14]. There have been some works in this field that perform classification tasks with quantum computing [30]. The approaches include quantum machine learning [31]–[34], quantum-inspired machine learning [35]–[37], and hybrid quantum-classical machine learning [13], [38]–[41].

Among these approaches, the hybrid approach attracts much attention. A quantum multi-class classifier, QMCC, proposed in [39], uses n qubits to encode an N-dimensional data vector and uses k ancilla qubits as readouts to store the predicted label out of k classes. Although the quantum computer is much faster than the classical computer, the speed of hybrid quantum-classical approaches is still constrained by classical phases. Hence, some efforts have been made for acceleration. A “single-shot training” style proposed in [13] uses all the input samples with the same label to train the classifier at the same time to speed up the training procedure. QReliefF accelerates the training process by combining quantum machine learning and edge computing so that some computations can be finished parallelly [42]. Besides the quantum computation phase, the time cost in the quantum encoding phase can

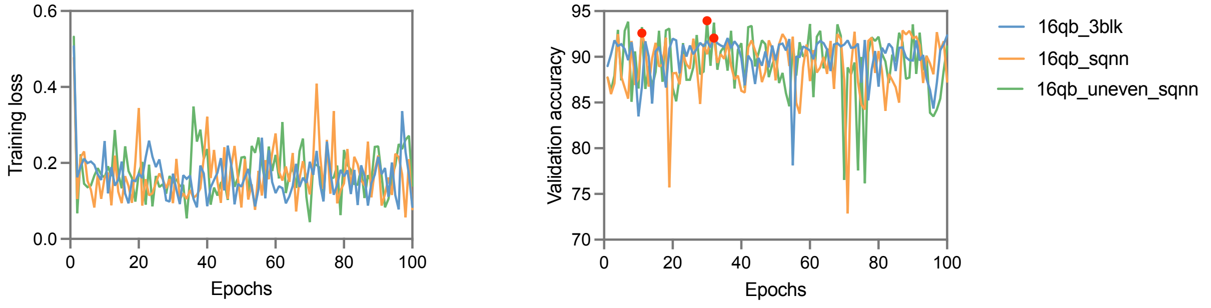


Fig. 10. **The comparison of performance among input partition strategies.** Three 16 data-qubit (qb) models are compared in this experiment: single-device model $16qb\_3blk$, SQNN model $16qb\_sqnn$ and SQNN model with uneven input partition $16qb\_uneven\_sqnn$. The availability of various input partition strategies is proven by their comparable performance.

also be reduced by using a quantum dataset so that the quantum devices do not need to perform encoding for classical data. A quantum dataset NTangled is proposed in [43]. It is composed of quantum states with different amounts and types of entanglements.

Quantum models are limited in their ability to investigate potential patterns in data due to hardware restrictions on quantum devices. The existing approaches must shrink the raw data size to fit the number of available qubits on the quantum device. The intuitive method is to downscale the raw data into a small size, but much helpful information is lost. A solution proposed in [44] is to use a quantum-inspired tensor network (TN), particularly a matrix product state, as a feature extractor to reduce data size and then use the compressed data as the input of VQC to perform supervised learning tasks. However, the quantum-inspired TN runs on classical devices, which slows down the training speed. A large-scale QML is proposed in [45]. It computes quantum kernels using randomized measurements and loads high-dimensional classical data on a quantum device by repeating the encoding layer, which increases the depth of the quantum circuit and is incompatible with the QNN approaches considered in our work.

The distributed systems of QNN are designed to train a quantum model among multiple parties collaboratively. QuantumFed [46] combines QNN and federated learning to allow multiple quantum devices to train a QML model collaboratively on their private dataset. And the authors further consider the security issue of quantum federated learning. Their work presented in [47] protects the QML model trained in the federated learning system from Byzantine attacks, in which adversaries upload malicious information to degrade model performance. The federated QNN through blind quantum computing implemented with differential privacy in order to guarantee clients' privacy [48]. DSQML, distributed secure quantum machine learning, is proposed to implement a secure QNN without leakage of any information about training data [49]. The above approaches are computationally intensive on classical devices and do not overcome the quantum hardware limitation. In comparison to the previous works, our proposed approach performs no additional computation on classical devices other than the update calculation required in the hybrid method.

VI. Conclusion

Considering the difficulty and cost of building a large-scale quantum device for QNN, we propose a scalable quantum neural network (SQNN), which is deployed on multiple small-size quantum devices. In this paper, we consider SQNN models for a binary classification task. The SQNN classifier learns from high-dimensional training data by partitioning it into small segments. And the resource-constrained quantum devices in the quantum system extract local features from the segments in parallel. The extracted features are then merged and predicted using a quantum device acting as a quantum predictor. Moreover, the training sample could be partitioned into segments of different sizes to fully utilize the quantum devices with various amounts of quantum resources. Furthermore, we show that a single small-size quantum device is also able to train a large neural network according to the SQNN approach. Extensive experiments indicate that having more quantum resources improves SQNN classifier accuracy significantly, and our proposed approach outperforms regular QNNs trained with limited quantum resources.

Acknowledgment

The authors would like to thank all the reviewers for their helpful comments. This project was supported in part by US National Science Foundation grant CNS-1816399. This work was also supported in part by the Commonwealth Cyber Initiative, an investment in the advancement of cyber R&D, innovation and workforce development. For more information about CCI, visit cyberinitiative.org.

References

[1] M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," *Contemporary Physics*, vol. 56, no. 2, pp. 172–185, 2015.

[2] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[3] 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 Systems with Applications*, p. 116512, 2022.

[4] M. A. Nielsen and I. Chuang, “Quantum computation and quantum information,” 2002.

[5] A. Steane, “Quantum computing,” *Reports on Progress in Physics*, vol. 61, no. 2, p. 117, 1998.

[6] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[7] L. K. Grover, “A fast quantum mechanical algorithm for database search,” in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 212–219.

[8] K. Beer, D. Bondarenko, T. Farrelly, T. J. Osborne, R. Salzmann, D. Scheiermann, and R. Wolf, “Training deep quantum neural networks,” *Nature communications*, vol. 11, no. 1, pp. 1–6, 2020.

[9] N. Killoran, T. R. Bromley, J. M. Arrazola, M. Schuld, N. Quesada, and S. Lloyd, “Continuous-variable quantum neural networks,” *Physical Review Research*, vol. 1, no. 3, p. 033063, 2019.

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

[11] R. Zhou and Q. Ding, “Quantum mp neural network,” *International Journal of Theoretical Physics*, vol. 46, no. 12, pp. 3209–3215, 2007.

[12] R. Zhou, “Quantum competitive neural network,” *International Journal of Theoretical Physics*, vol. 49, no. 1, pp. 110–119, 2010.

[13] S. Adhikary, S. Dangwal, and D. Bhowmik, “Supervised learning with a quantum classifier using multi-level systems,” *Quantum Information Processing*, vol. 19, no. 3, pp. 1–12, 2020.

[14] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio *et al.*, “Variational quantum algorithms,” *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.

[15] J. Preskill, “Quantum computing in the nisq era and beyond,” *Quantum*, vol. 2, p. 79, 2018.

[16] L. Bottou, “Stochastic gradient descent tricks,” in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 421–436.

[17] R. Sweke, F. Wilde, J. Meyer, M. Schuld, P. K. Fährmann, B. Meynard-Piganeau, and J. Eisert, “Stochastic gradient descent for hybrid quantum-classical optimization,” *Quantum*, vol. 4, p. 314, 2020.

[18] S. Y.-C. Chen and S. Yoo, “Federated quantum machine learning,” *Entropy*, vol. 23, no. 4, p. 460, 2021.

[19] S. Lloyd, M. Mohseni, and P. Rebentrost, “Quantum principal component analysis,” *Nature Physics*, vol. 10, no. 9, pp. 631–633, 2014.

[20] S. A. Stein, B. Baheri, D. Chen, Y. Mao, Q. Guan, A. Li, S. Xu, and C. Ding, “Quclassi: A hybrid deep neural network architecture based on quantum state fidelity,” *Proceedings of Machine Learning and Systems*, vol. 4, pp. 251–264, 2022.

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

[22] M. Schuld and F. Petruccione, *Supervised learning with quantum computers*. Springer, 2018, vol. 17.

[23] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, “Quantum circuit learning,” *Physical Review A*, vol. 98, no. 3, p. 032309, 2018.

[24] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, “Evaluating analytic gradients on quantum hardware,” *Physical Review A*, vol. 99, no. 3, p. 032331, 2019.

[25] “Tensorflow quantum tutorial,” https://www.tensorflow.org/quantum/tutorials/mnist#21_build_the_model_circuit, 2020.

[26] L. Deng, “The mnist database of handwritten digit images for machine learning research,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[27] M. Altaisky, “Quantum neural network,” *arXiv preprint quant-ph/0107012*, 2001.

[28] S. Jeswal and S. Chakraverty, “Recent developments and applications in quantum neural network: a review,” *Archives of Computational Methods in Engineering*, vol. 26, no. 4, pp. 793–807, 2019.

[29] B. Ricks and D. Ventura, “Training a quantum neural network,” *Advances in neural information processing systems*, vol. 16, 2003.

[30] Z. Abohashima, M. Elhosen, E. H. Houssein, and W. M. Mohamed, “Classification with quantum machine learning: A survey,” *arXiv preprint arXiv:2006.12270*, 2020.

[31] D. Willsch, M. Willsch, H. De Raedt, and K. Michielsen, “Support vector machines on the d-wave quantum annealer,” *Computer physics communications*, vol. 248, p. 107006, 2020.

[32] P. Rebentrost, M. Mohseni, and S. Lloyd, “Quantum support vector machine for big data classification,” *Physical review letters*, vol. 113, no. 13, p. 130503, 2014.

[33] M. Schuld, I. Sinayskiy, and F. Petruccione, “Prediction by linear regression on a quantum computer,” *Physical Review A*, vol. 94, no. 2, p. 022342, 2016.

[34] A. J. da Silva, T. B. Ludermir, and W. R. de Oliveira, “Quantum perceptron over a field and neural network architecture selection in a quantum computer,” *Neural Networks*, vol. 76, pp. 55–64, 2016.

[35] P. Tiwari and M. Melucci, “Towards a quantum-inspired binary classifier,” *IEEE Access*, vol. 7, pp. 42 354–42 372, 2019.

[36] G. Sergioli, R. Giuntini, and H. Freytes, “A new quantum approach to binary classification,” *PloS one*, vol. 14, no. 5, p. e0216224, 2019.

[37] C. Ding, T.-Y. Bao, and H.-L. Huang, “Quantum-inspired support vector machine,” *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[38] M. Schuld and N. Killoran, “Quantum machine learning in feature hilbert spaces,” *Physical review letters*, vol. 122, no. 4, p. 040504, 2019.

[39] A. Chalumuri, R. Kune, and B. Manoj, “A hybrid classical-quantum approach for multi-class classification,” *Quantum Information Processing*, vol. 20, no. 3, pp. 1–19, 2021.

[40] A. Gianelle, P. Koppenburg, D. Lucchesi, D. Nicotra, E. Rodrigues, L. Sestini, J. de Vries, and D. Zuliani, “Quantum machine learning for b-jet identification,” *arXiv preprint arXiv:2202.13943*, 2022.

[41] Z. Tao, J. Wu, Q. Xia, and Q. Li, “Laws: Look around and warm-start natural gradient descent for quantum neural networks,” *arXiv preprint arXiv:2205.02666*, 2022.

[42] W. Liu, J. Chen, Y. Wang, P. Gao, Z. Lei, and X. Ma, “Quantum-based feature selection for multiclassification problem in complex systems with edge computing,” *Complexity*, vol. 2020, 2020.

[43] L. Schatzki, A. Arrasmith, P. J. Coles, and M. Cerezo, “Entangled datasets for quantum machine learning,” *arXiv preprint arXiv:2109.03400*, 2021.

[44] 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,” *arXiv preprint arXiv:2011.14651*, 2020.

[45] T. Haug, C. N. Self, and M. Kim, “Large-scale quantum machine learning,” *arXiv preprint arXiv:2108.01039*, 2021.

[46] Q. Xia and Q. Li, “Quantumfed: A federated learning framework for collaborative quantum training,” in *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 1–6.

[47] Q. Xia, Z. Tao, and Q. Li, “Defending against byzantine attacks in quantum federated learning,” in *2021 17th International Conference on Mobility, Sensing and Networking (MSN)*. IEEE, 2021, pp. 145–152.

[48] W. Li, S. Lu, and D.-L. Deng, “Quantum federated learning through blind quantum computing,” *Science China Physics, Mechanics & Astronomy*, vol. 64, no. 10, pp. 1–8, 2021.

[49] Y.-B. Sheng and L. Zhou, “Distributed secure quantum machine learning,” *Science Bulletin*, vol. 62, no. 14, pp. 1025–1029, 2017.