A quantum deep convolutional neural network for image recognition

To cite this article before publication: YaoChong Li *et al* 2020 *Quantum Sci. Technol.* in press https://doi.org/10.1088/2058-9565/ab9f93

A quantum deep convolutional neural network for image recognition

YaoChong Li[1,2], Ri-Gui Zhou[1,2]‡, RuQing Xu[1,2], Jia Luo[1,2] and WenWen Hu[1,2]

[1] College of Information Engineering, Shanghai Maritime University, Shanghai 201306, China

[2] Research Center of Intelligent Information Processing and Quantum Intelligent Computing, Shanghai 201306, China

E-mail: rgzhou@shmtu.edu.cn

June 2020

Abstract. Deep learning achieves unprecedented success involves many fields, whereas the high requirement of memory and time efficiency tolerance have been the intractable challenges for a long time. On the other hand, quantum computing shows its superiorities in some computation problems owing to its intrinsic properties of superposition and entanglement, which may provide a new path to settle these issues. In this paper, a quantum deep convolutional neural network (QDCNN) model based on the quantum parameterized circuit for image recognition is investigated. In analogy to the classical deep convolutional neural network (DCNN), the architecture that a sequence of quantum convolutional layers followed by a quantum classified layer is illustrated. Inspired by the variational quantum algorithms, a quantum-classical hybrid training scheme is demonstrated for the parameter updating in the QDCNN. The network complexity analysis indicates the proposed model provides the exponential acceleration comparing with the classical counterpart. Furthermore, the MNIST and GTSRB datasets are employed to numerical simulation and the quantitative experimental results verify the feasibility and validity.

Keywords: quantum information processing, quantum circuit, image recognition, quantum machine learning

1. Introduction

Quantum computing [1] relies on properties of quantum-mechanical phenomena such as superposition and entanglement to perform computation, its massive parallelism provides potential acceleration over classical computing paradigm. Since David Deutsch formalized the quantum Turing machine theory [2] in 1985, a large number of quantum algorithms [3] have been proposed over past decades. On the other hand, as the result of the surge of data and the development of hardware, machine learning has aroused great

‡ Corresponding author

interest in both academia and industry, ranging from face recognition to automated self-driving cars. The great successes of both machine learning and quantum physics inspire researchers to investigate if and how quantum computing can help to improve classical machine learning algorithms. Currently, quantum machine learning (QML) [4] is an emerging interdisciplinary field that is devoted to using the strengths of quantum computing to improve classical machine learning algorithms. Over the past decades, a number of contributions [5, 6, 7, 8, 9, 10, 11, 12, 13] aiming to employ quantum technologies to speed up or enhance the performance in classical machine learning.

Particularly, in the context of Noisy Intermediate Scale Quantum (NISQ) devices [14], several interesting contributions toward quantum deep learning have been proposed using the quantum annealers [15, 16] or variational quantum circuits [17, 18, 19, 20]. However, its interpretability and efficiency compared with classical deep neural networks are worthy of further discussion. As one of the most representative models in deep learning (a subfield of machine learning) [21], the deep convolutional neural network (DCNN) is widely used in many diverse areas including computer vision, speech recognition, natural language processing. However, despite these achievements, its computational cost would be increased dramatically with the width and depth of layers growing, which is deemed as the primary bottleneck in deep learning. Fortunately, recent advances in quantum information processor [22, 23, 24] show that the quantum computer seems to be a potential parallel device for improving the efficiency of classical computing problems. Obviously, it is a highly nontrivial and meaningful task that exploring the ingenious union of quantum computing and deep learning. More specifically, a meaningful quantum deep learning framework should at least satisfy the following four criteria:

- Integrating the nonlinear, dissipative dynamics in neural computing and linear, unitary dynamics in quantum computing [25].
- Abiding by the laws of quantum mechanics rather than a quantum inspired mathematical description [26].
- Constructing an efficient quantum transformation between the layers to obtain the deep semantic feature.
- Taking full advantage of the quantum parallelism in both storage and evolution, which is the original intention for settling the obstructions in classical neural computing.

In this paper, a quantum deep convolutional neural network (QDCNN) model based on the quantum parameterized circuit is proposed. Taking the image recognition as the application background, the input image is first prepared as the quantum state with basis encoding. A sequence of parameter-dependent unitary transformations is then employed to realize the corresponding quantum evolution, and it can be divided into the quantum convolutional layer and quantum classified layer. It should be noticed that the pooling layer is omitted and the subsampling is accomplished by increasing convolutional stride as 2^m, which can remove the intermediate measurement efficiently

and cause the fast dimension reduction in generated feature mapping. Finally, the quantum measurement is performed on the specified quantum bits to obtain the category label. Motivated by the variational quantum model [27, 28, 29, 30], the hybrid quantum-classical training algorithm is applied to optimize parameters in the quantum circuit. The experiment results on the MNIST and GTSRB datasets indicate that satisfactory accuracy is achieved by QDCNN.

Specifically, in the quantum convolutional layer, the quantum random access memory (QRAM) algorithm [31, 32] is employed to the quantum preparation for an input image and related convolutional kernels. It should be mentioned that the convolutional kernel shares the low m quantum bits of the image location register. And then a quantum multiplier and the Hadamard controlled rotation operation are used to realize quantum inner product computation on the kernel-working region in parallel. Next, the conversion between amplitude encoding and basis encoding is accomplished by quantum phase estimation (QPE) and the nonlinear mapping is performed on the computational basis. Finally, the uncomputation operation is employed to obtain the input state in the next layer by disentangling the desired state and intermediate state. Moreover, the case of multiple feature mappings is also discussed, the additional quantum bit resource can be saved with the recycling of the low m quantum bits in location register. In the quantum classified layer, the feature mappings generated from the quantum convolutional layer are evolved further and the corresponding probability distribution of category is obtained by quantum measurement. As compared with classical DCNN, one of the main advantages in the QDCNN model is quantum parallelism that allows simultaneously working with all permissible states where the exponential acceleration is provided.

The remainder of this paper is organized as follows: Section 2 discusses the existing contributions related to the quantum neural networks. Section 3 introduces the involved preliminaries necessary to understand this work. The implementation scheme for QDCNN is detailed in Section 4. To validate the feasibility of the model, Section 5 presents the experimental simulation and analysis. Finally, the conclusion and future work are discussed in Section 6.

2. Related Work

Quantum neural networks (QNNs) are powerful computational models aiming to combine neural computing with the properties of quantum computing, which have been demonstrated to be superior to the classical neural networks. Specifically, those superiorities are mainly focused on the following aspects: processing information in parallel to obtain quantum speedup [33, 34, 35]; extracting the optimal network architecture with low cost [36, 26, 37]; improving the network performance [18, 38, 39] (including accuracy, generalization, and robustness, etc).

Subhash Kak reviewed the limitations of the standard computing paradigm and firstly put forward the concept of quantum neural computing [40]. After that, various

QNN models have been proposed by different researchers but there is no consensus in the components of QNNs to date as an outstanding open problem. There are two systematic reviews of QNNs, refer to [25, 41]. In quantum computation, one state is mapped onto another by linear operators. But in the neural network, non-linearity is introduced by an activation function, which provides a powerful capability for different learning tasks. Therefore, it is essential to introduce non-linearity in the QNN. Several studies attempted to establish the nonlinear mathematical formula according to quantum expression [42, 43, 44, 45, 46]. However, these works are collectively called quantum-inspired neural networks, which can't be defined as real quantum models.

One of the most common approaches to introduce non-linearity in QNNs is the quantum measurement, which is similar to the case of a single perceptron [47, 48]. In the [33, 34], the quantum perceptron is demonstrated with the quantum phase estimation, but the construction of more complex structure (i.e. QNN) based on the above quantum perceptron is not mentioned. Besides, based on the superposition based architecture learning algorithm (SAL), a direct generalization of the classical perceptron, named quantum perceptron over a field (QPF) is proposed in [26], which solves some drawbacks found in previous models of the quantum perceptron. In [49], Chen et al. proposed a quantum probability neural network (QPNN) model, which employs the quantum parallelism property to trace all possible network states. In the recent research [50], base on the basic architecture of [33], the scheme for implementing any nonlinear quantum perceptron and the corresponding example of QNN are presented. However, in the above scheme, the consumption of quantum bit resource is proportional to the input dimension since the input quantum state are stored in the quantum register separately. Rebentrost et al. leveraged the exponential advantage in storage resources to construct the quantum Hopfield neural network with a quantum computational complexity that is logarithmic in the dimension of the data [51]. Furthermore, Tacchino et al. presented a quantum binary-valued perceptron and implement it on the actual IBM quantum processor [52]. In addition, Zhao et al. designed a QNN model based on the quantum swap test algorithm and satisfying results are achieved in the numerical simulation experiment [53].

It should be mentioned that a similar model acronym is found in [54], but is conceptually different from this work. Inspired from quantum physics, Cong et al. defined a new quantum circuit model that can be applied for learning or classifying phase of quantum physical systems, and shares similar aspects with the DCNN. However, they are not reproducing the corresponding operations of DCNN, which is the topic of our work. Besides, the other interesting quantum convolutional publication is presented in the most recent work [39], where a quanvolutional layer is proposed using a random quantum circuit and further replacing the convolutional layer of classical DCNN with it. The effectiveness of quanvolutional layer on image classification is demonstrated by comparing it with the classical counterpart.

By comparing previous related proposals, the major innovations of this work are concluded as follows:

- Comparing with classical DCNN with the same structure, the QDCNN acquires an exponential acceleration in the time complexity and its performance in image recognition is superior to the existing QML algorithms.
- In the structure of the successive layers, the number of degrees of freedom is reduced by disentanglement. Hence, the necessity of intermediate measurement or partial trace operations is eliminated in the QDCNN model.
- QDCNN makes full use of quantum superposition in the input state preparation, convolutional kernel storage, and feature mapping representation, which reduces the consumption of quantum bit resources effectively.
- With the Taylor expansion and basic quantum arithmetic modules (quantum adder and quantum multiplier), the universal nonlinear activation function can be applied in the QDCNN model.

3. Preliminaries

3.1. Quantum Computing Background

In analogy to the bit in the classical information, the basic unit of quantum information is defined as the qubit, which is denoted by Dirac notation [55]:

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

It can be photonic polarization, spin states, electronic states of an ion, or charge states of superconducting systems. Different from the classical bit, a qubit can be existed in both $|0\rangle$ and $|1\rangle$ simultaneously, called a superposition state:

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

where α, β are complex numbers called probability amplitudes and $|\alpha|^2+|\beta|^2 = 1$. When performing the quantum measurement, it will collapse to the state $|0\rangle$ with probability $|\alpha|^2$, or to the state $|0\rangle$ with probability $|\beta|^2$. More generally, the above principle can be extended to the n qubits system, then 2^n computational basis are obtained. It can be expressed as $|\psi\rangle_n = \sum_{i=0}^{2^n-1} a_i|i_b\rangle$, where the $|\psi\rangle_n$ is superposed by n qubits, i_b means the binary representation of number i, and the a_i is the probability amplitude satisfying $\sum_{i=0}^{2^n-1} |a_i|^2 = 1$. The other unique property of a quantum state is the entanglement, an intuitive example is the Bell state $(|00\rangle + |11\rangle)/\sqrt{2}$, which can't be represented as a tensor product of two independent qubits.

In the conventional digital circuit, the manipulations on bits are performed by the classical logic gates. The equivalent building blocks in the quantum circuit are called quantum gates. It should be mentioned that a quantum gate $\mathbf{U}$ must be a unitary operation ($\mathbf{U}\mathbf{U}^\dagger = \mathbf{I}$) and implemented reversibly, which is different from the classical situation. The involved quantum gates is introduced in the Figure. 1. Besides, the

quantum circuit is usually employed to model the behavior of quantum computers. In a quantum circuit, a collection of input quantum state is first prepared to store information, and then passed through a series of unitary operations (quantum gates), the output states are obtained by quantum measurement finally.

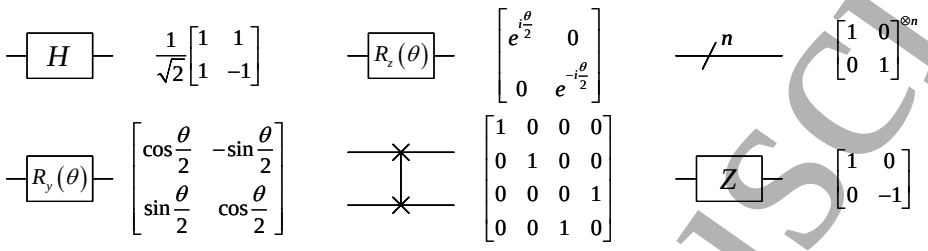


Figure 1. Notations for the elementary gates involved in this paper and their corresponding matrices.

3.2. Convolutional Neural Network

It has been proven that the DCNN model is a very effective network architecture due to the impressive performance on the machine learning tasks [56, 57, 58]. The main advantage of DCNN compared to its predecessors is that it automatically detects the high-level features by various filters in a successive layer structure without any human supervision. A common DCNN model is composed of three elementary modules: convolutional layer, pooling layer, and fully connected layer. A visualization of the basic frame of a DCNN is presented in Figure. 2 (a).

The major function of the convolutional layer is to extract feature mapping from the input, which is realized by a number of the small rectangular matrix, namely convolutional kernel or filter. Given an input image, the convolutional layer performs dot product calculation when the kernel slide on it. An activation function is then applied for introducing non-linearity into the model. Various feature mappings can be obtained by different kernels applied in the input image. To ensure integrity, the basic concept of the pooling layer is also presented here though the corresponding quantum operation is not involved in the QDCNN model. The pooling layer is generally to be inserted between successive convolutional layers in a DCNN architecture. Its role is to reduce the spatial size of the representation while retaining important features, by which it can decrease the number of parameters, computation in the network and also control the over-fitting. There are various types of units that can be performed in the pooling layer and the most used extensively is max pooling. The last module is a fully connected layer, the same as ordinary feed-forward neural networks, which has full connections to all activations in the previous layer. It is designed as a classifier to process the features extracted from the convolutional layer and the pooling layer. An illustrative example of convolutional and pooling operation is shown in Figure. 2 (b).

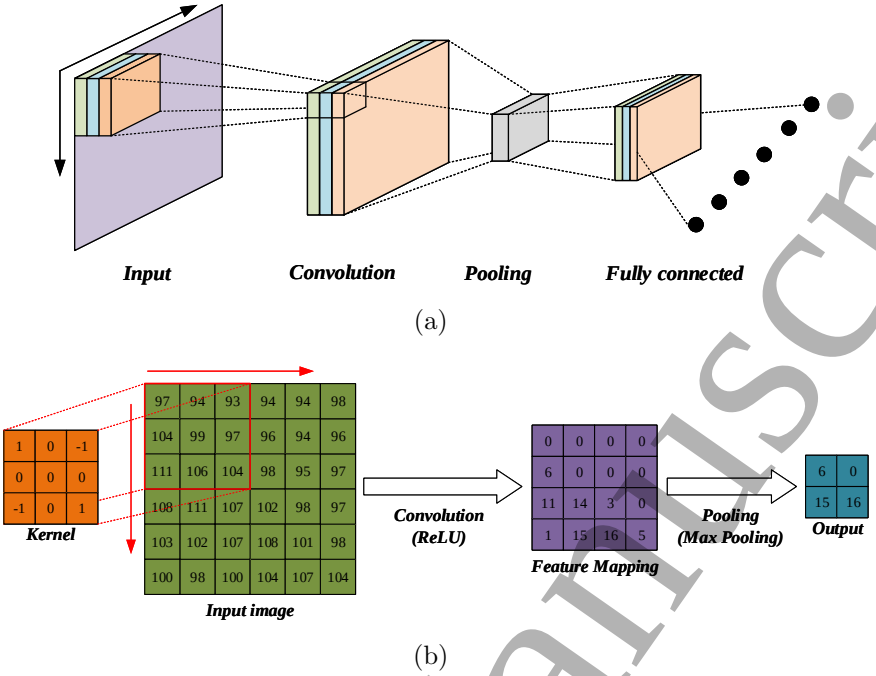


Figure 2. (a) The structure sketch of DCNN with a convolutional layer containing 3 filters, followed by a pooling layer and a fully connected output layer with 6 neurons. (b) An illustrative example of specific convolutional and pooling operation.

4. Research Methodology

To facilitate the understanding, two foundational theorems and related notation description are provided before constructing the detailed implementation of QDCNN.

Theorem 1: The QRAM algorithm [31, 32] is employed for quantum basis encoding, which is defined by:

$$U_{\mathrm{QRAM}}\left(\{d_j\}\right)|j\rangle|0\rangle \rightarrow |j\rangle|d_j\rangle \tag{3}$$

Where the U_{QRAM} is a unitary transformation and the $|j\rangle$ is the address register while the $|d_j\rangle$ is the data register. The whole preparation require only $O(\mathrm{poly}(\log N))$ steps for N classical data.

Theorem 2: The addition and multiplication operation can be computed efficiently between two basis-encoding quantum states [59], which is given as:

$$\begin{aligned} |a\rangle|b\rangle &\rightarrow |a\rangle|a+b\rangle \\ |a\rangle|b\rangle &\rightarrow |a\rangle|ab\rangle \end{aligned} \tag{4}$$

In fact, for the basic functions f has a convergent Taylor series, the transformation $|a\rangle|0\rangle \rightarrow |a\rangle|f(a)\rangle$ can also be realized by Taylor series expansion [60]. According to the theory of quantum arithmetic, its complexity is given by $O(\mathrm{poly}(\log(1/\varepsilon)))$ while the accuracy defined as ε.

Notation: Let bold letters $\boldsymbol{a} = \{\boldsymbol{a}_0, \boldsymbol{a}_1, \cdots, \boldsymbol{a}_{N-1}\}$ be a set of real number with N elements and the value of $\boldsymbol{a}_j$ is in $[0, 1]$, which can be represented by $\boldsymbol{a}_j = \sum_{i=1}^{d} a_j^{(i)} 2^{-i}$ with binary variables $a_j^{(i)} \in \{0, 1\}$. Let a_j denote the binary string $a_j^{(1)} \cdots a_j^{(d)}$.

4.1. Quantum Preparation of Classical Image

The first step of QDCNN is to prepare the classical input image into a quantum state. Given a $2^n \times 2^n$ classical gray input image I, it can be described as a matrix:

$$I = \begin{bmatrix} c_{0,0} & c_{0,1} & \cdots & c_{0,N-1} \\ c_{1,0} & c_{1,1} & \cdots & c_{1,N-1} \\ \vdots & \vdots & \vdots & \vdots \\ c_{N-1,0} & c_{N-1,1} & \cdots & c_{N-1,N-1} \end{bmatrix} \tag{5}$$

Where $N = 2^n$ and the element $c_{x,y} \in [0, 255]$ represents the corresponding pixel value in the position $(x, y) \in [0, N-1]^{\otimes 2}$. Normalization is then applied to the I according to the following formula:

$$\tilde{\boldsymbol{c}}_{x,y} = \frac{c_{x,y} - min(I)}{max(I) - min(I)} \tag{6}$$

In which, $max(I)$ and $min(I)$ denote the maximum element and minimum element in the matrix I respectively. Obviously, the value of $\tilde{\boldsymbol{c}}_{x,y}$ is squashed to range between 0 and 1. Hence, it can be represented by d binary fractions and the numerical error is no more than 2^{-d}:

$$\tilde{\boldsymbol{c}}_{x,y} = \frac{\tilde{c}^{(1)}_{x,y}}{2} + \frac{\tilde{c}^{(2)}_{x,y}}{4} + \cdots + \frac{\tilde{c}^{(d)}_{x,y}}{2^d} \tag{7}$$

According to the standard expression of the QRAM, the original input image I can be prepared into a quantum state $|I\rangle$, which is denoted as:

$$\frac{1}{N}\sum_{l=0}^{N^2-1} |l\rangle_{regL}|0\rangle^{\otimes d}_{regC} \to \frac{1}{N}\sum_{l=0}^{N^2-1} |l\rangle_{regL}|\tilde{c}_l\rangle_{regC} = |I\rangle \tag{8}$$

In the above equation, $|\tilde{c}_l\rangle_{regC}$ represents the color information at the corresponding position $|l\rangle_{regL}$. To be more precise, the quantum registers $|l\rangle_{regL}$ and $|\tilde{c}_l\rangle_{regC}$ can be written as follows:

$$\begin{aligned} |l\rangle_{regL} &= |x, y\rangle_{regL} = |x_{n-1} \cdots x_1 x_0, y_{n-1} \cdots y_1 y_0\rangle_{regL} \\ |\tilde{c}_l\rangle_{regC} &= |\tilde{c}_{x,y}\rangle_{regC} = |\tilde{c}^{(1)}_{x,y}\tilde{c}^{(2)}_{x,y} \cdots \tilde{c}^{(d)}_{x,y}\rangle_{regC} \end{aligned} \tag{9}$$

The bit string $x_{n-1}x_{n-2}\cdots x_0$ and $y_{n-1}y_{n-2}\cdots y_0$ are the n-digit binary representation of the location information x and y, respectively.

4.2. QDCNN Model

Analogous to the classical situation, the QDCNN architecture consists of several successive quantum convolutional layers and a quantum classifier layer. It should be noticed that there are two preconditions compared with classical DCNN:

Precondition 1: As explained previously, the pooling operation is omitted in the QDCNN architecture. The further subsampling (reducing the dimension of the feature

mapping) is realized by preserving the high qubits in location register and disentangling the low qubit(s).

Precondition 2: To reduce the cost of quantum bit resources, the reuse of the location register is employed. The convolutional stride is setting as 2^m with the size of the convolutional kernel is $2^m \times 2^m$, which is different from the ordinary stride in classical DCNN (the effectiveness of even-sized kernel has been demonstrated in the recent research [61]). To compensate the information loss in the 2^m-stride convolutional operation, m is generally set as 1 for the low-resolution images. For high-resolution cases, like the medical image, remote sensing image, the value of m could be increasing properly.

4.2.1. Quantum Convolutional Layer There are several problems should be considered before constructing quantum convolutional layer:

- The inner product calculation between input variables and weights in quantum computation.
- The strategy for reducing dimension of feature mapping in the image convolution operation.
- The intermediate feature mapping should preserve the entanglement between its intensity and location.

Based on the discussions above, the quantum circuit of the convolutional layer in QDCNN is shown in Figure. 3. The detailed transformation in the circuit is given by the following steps:

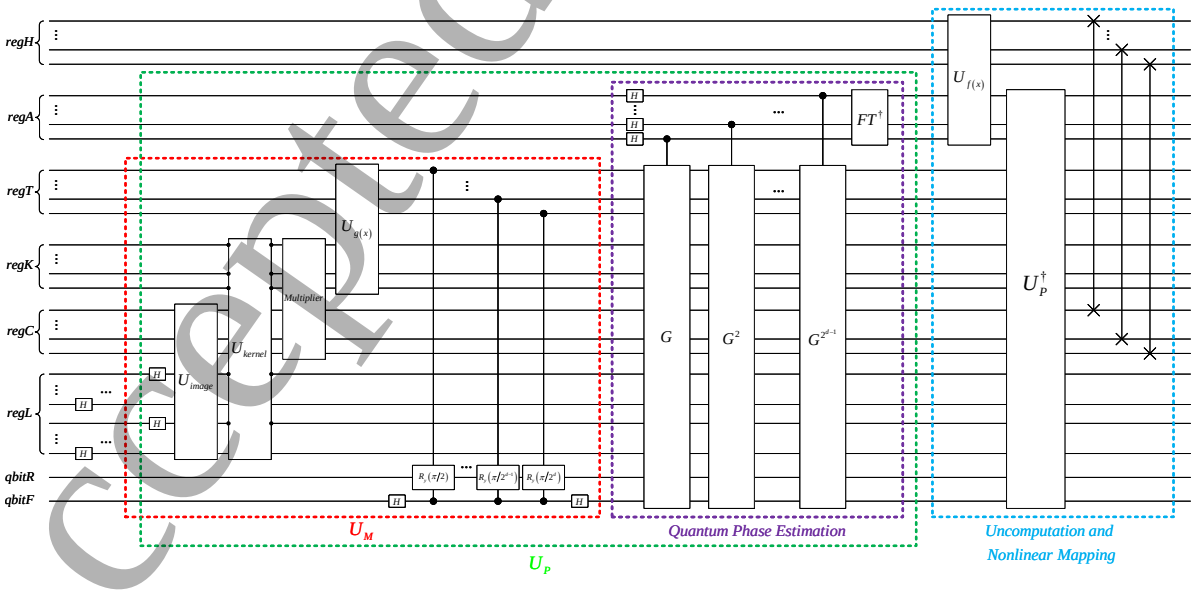


Figure 3. The quantum circuit for the convolutional layer in QDCNN model. (The black points marked in the unitary transformation represent the interaction site.)

Step.1: All qubits in the quantum register are initialized as $|0\rangle$ state.

Step.2: A group of Hadamard gates are performed on the quantum register $regL$, consisted of $2n$ qubits, to produce the location space of an image. And then employ the QRAM algorithm on the quantum registers $regC$ and $regK$ to prepare the input image and convolutional kernel respectively:

$$\frac{1}{N}\sum_{l=0}^{N^2-1}|l\rangle_{regL}|\tilde{c}_l\rangle_{regC}|k_l\rangle_{regK} \tag{10}$$

According to the Precondition 2, the weights are equivalent if the high $n-m$ qubits in the $|x\rangle$ and $|y\rangle$ are equivalent. Let the $m=1$, the variable $k_{x,y}$ is defined as:

$$k_{x,y}=\begin{cases} w_{00} & (x_0,y_0)=(0,0) \\ w_{01} & (x_0,y_0)=(0,1) \\ w_{10} & (x_0,y_0)=(1,0) \\ w_{11} & (x_0,y_0)=(1,1) \end{cases} \tag{11}$$

Hence, as presented in Figure. 3, the unitary transformation U_{kernel} (preparation of the kernel) is only performed on the least significant digit of the x, y and register $regK$.
Step.3: Based on the Theorem 2, the quantum multiplier is then performed on the register $regC$ and $regK$, thus the quantum state can be written as:

$$\frac{1}{N}\sum_{l=0}^{N^2-1}|l\rangle_{regL}|\tilde{c}_l\rangle_{regC}|\tilde{c}_l * k_l\rangle_{regK} \tag{12}$$

As mentioned in the Step.2, the convolutional kernel shares the location information with the image. Hence, the above multiplication operation can be realized by just one multiplier owing to the quantum parallelism. After that, introducing the quantum register $regT$ and computing $g(x)=\frac{2}{\pi}\arccos(x)$ by quantum arithmetic:

$$\frac{1}{N}\sum_{l=0}^{N^2-1}|l\rangle_{regL}|\tilde{c}_l\rangle_{regC}|\tilde{c}_l * k_l\rangle_{regK}|t_l\rangle_{regT} \tag{13}$$

Where the $t_l=g(\tilde{c}_l * k_l)$.
Step.4: Adding a fork qubit $qbitF$ and a rotation qubit $qbitR$. Performing the quantum Hadamard test coupling with a group of controlled rotation operations. Specifically, it can be divided into the following sub-steps:
−Step. 4.1: A Hadamard gate is firstly applied on the qubit $qbitF$ to produce the quantum fork:

$$\frac{1}{N}\sum_{l=0}^{N^2-1}|l\rangle_{regL}|\tilde{c}_l\rangle_{regC}|\tilde{c}_l * k_l\rangle_{regK}|t_l\rangle_{regT}\otimes\left(\frac{1}{\sqrt{2}}|0\rangle_{qbitF}+\frac{1}{\sqrt{2}}|1\rangle_{qbitF}\right)|0\rangle_{qbitR} \tag{14}$$

−Step. 4.2: Applying a series of controlled y-axis rotation gates on the rotation qubit $qbitR$. The π is divided into a series of sub-modules with fractional form denoted as

$\left\{\frac{\pi}{2}, \frac{\pi}{4}, \cdots, \frac{\pi}{2^d}\right\}$, which are employed as the angles in rotation gates. The control bits for each controlled rotation operation are the corresponding qubit in the register $regT$ and the fork qubit $qbitF$. Consequently, the transformed quantum state is given as:

$$\frac{1}{N}\sum_{l=0}^{N^2-1}|l\rangle_{regL}|\tilde{c}_l\rangle_{regC}|\tilde{c}_l * k_l\rangle_{regK}|t_l\rangle_{regT}\otimes\left(\frac{1}{\sqrt{2}}|0\rangle_{qbitF}|0\rangle_{qbitR}+\frac{1}{\sqrt{2}}|1\rangle_{qbitF}|\boldsymbol{r_l}\rangle_{qbitR}\right) \quad (15)$$

Where $|\boldsymbol{r_l}\rangle_{qbitR} = (\tilde{\boldsymbol{c}}_{\boldsymbol{l}} * \boldsymbol{k_l})\,|0\rangle_{qbitR} + \sqrt{1-(\tilde{\boldsymbol{c}}_{\boldsymbol{l}} * \boldsymbol{k_l})^2}|1\rangle_{qbitR}$.

−Step. 4.3: Finally, applying the other Hadamard gate on the $qbitF$. At this point, the quantum state is described as:

$$\frac{1}{N}\sum_{l=0}^{N^2-1}|l\rangle_{regL}|\tilde{c}_l\rangle_{regC}|\tilde{c}_l * k_l\rangle_{regK}|t_l\rangle_{regT}\otimes\left(\frac{1}{\sqrt{2}}|+\rangle_{qbitF}|0\rangle_{qbitR}+\frac{1}{\sqrt{2}}|-\rangle_{qbitF}|\boldsymbol{r_l}\rangle_{qbitR}\right) \quad (16)$$

Where $|+\rangle_{qbitF} = \frac{1}{\sqrt{2}}\left(|0\rangle_{qbitF} + |1\rangle_{qbitF}\right)$ and $|-\rangle_{qbitF} = \frac{1}{\sqrt{2}}\left(|0\rangle_{qbitF} - |1\rangle_{qbitF}\right)$.

It is found that when a $2^m * 2^m$ filter applied to the $2^n * 2^n$ input quantum image with stride 2^m, the high $n-m$ qubits of $|x\rangle$ and $|y\rangle$ can be employed to represent the location information of feature mapping. Hence, extracting the high $n-1$ qubits of $|x\rangle$ and $|y\rangle$ as $|x'\rangle$ and $|y'\rangle$ respectively, the quantum state in (16) can be reorganized as:

$$\frac{2}{N}\sum_{x'=0}^{\frac{N}{2}-1}\sum_{y'=0}^{\frac{N}{2}-1}|x',y'\rangle_{regL'}\left[\left(|u_{x',y'}\rangle+|v_{x',y'}\rangle\right)\frac{1}{2}|0\rangle_{qbitF}+\left(|u_{x',y'}\rangle-|v_{x',y'}\rangle\right)\frac{1}{2}|1\rangle_{qbitF}\right] \quad (17)$$

In which,

$$\begin{aligned}|u_{x',y'}\rangle &= \tfrac{1}{2}\textstyle\sum_{x'',y''=0}^{1}|0\rangle_{qbitR}|x'',y''\rangle_{regL''}|\tilde{c}_l\rangle|\tilde{c}_l * k_l\rangle|t_l\rangle \\ |v_{x',y'}\rangle &= \tfrac{1}{2}\textstyle\sum_{x'',y''=0}^{1}|\boldsymbol{r_l}\rangle_{qbitR}|x'',y''\rangle_{regL''}|\tilde{c}_l\rangle|\tilde{c}_l * k_l\rangle|t_l\rangle\end{aligned} \quad (18)$$

The register $regL$ is divided into $regL'$ (including the high $n-1$ qubits of $|x\rangle$ and $|y\rangle$) and $regL''$ (consisting of the lowest qubit in $|x\rangle$ and $|y\rangle$). Noticed that the computational basis of $|u_{x',y'}\rangle$ and $|v_{x',y'}\rangle$ in the Hilbert space are corresponding consistent. Therefore, the inner product of them can be concluded as the addition of a kernel working region. To make a more explicit description for its process, the schematic of the coordinate transformation is demonstrated in Figure. 4.

In fact, for each specified $|x',y'\rangle$, the amplitude of $|0\rangle_{qbitF}$ is $\sqrt{1+\langle u_{x',y'}|v_{x',y'}\rangle}/\sqrt{2}$ while the amplitude of $|1\rangle_{qbitF}$ is $\sqrt{1-\langle u_{x',y'}|v_{x',y'}\rangle}/\sqrt{2}$. Let $|\phi_{x',y'}\rangle$ and $|\varphi_{x',y'}\rangle$ denote the normalized vector $|u_{x',y'}\rangle + |v_{x',y'}\rangle$ and $|u_{x',y'}\rangle - |v_{x',y'}\rangle$ respectively. There exists a real number $\theta_{x',y'} \in [0, \pi/2]$ defined as:

$$\theta_{x',y'} = \arccos\left(\sqrt{1-\langle u_{x',y'}|v_{x',y'}\rangle}/\sqrt{2}\right) \quad (19)$$

Recalling the (18), it's not hard to see that $\langle u_{x',y'}|v_{x',y'}\rangle$ is equivalent to the quarter of the inner product in a kernel working region:

$$\langle u_{x',y'}|v_{x',y'}\rangle = -\cos 2\theta_{x',y'} = \frac{\boldsymbol{h_{x',y'}}}{4} \quad (20)$$

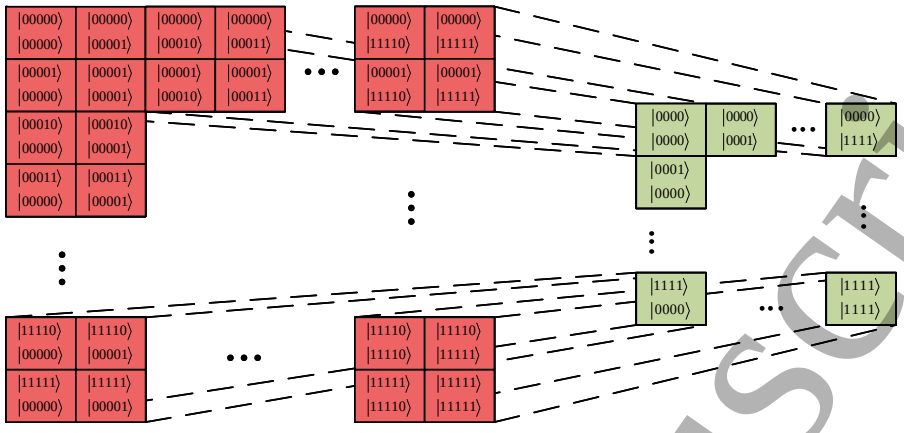


Figure 4. Schematic for coordinate transformation of quantum convolutional operation exampled with employing a $2 * 2$ filter to an $32 * 32$ input quantum image and the stride 2.

Moreover, the quantum state in (17) can be rewritten as

$$\frac{2}{N}\sum_{x'=0}^{\frac{N}{2}-1}\sum_{y'=0}^{\frac{N}{2}-1}|x',y'\rangle_{regL'}\otimes(\sin\theta_{x',y'}|0\rangle_{qbitF}|\phi_{x',y'}\rangle+\cos\theta_{x',y'}|1\rangle_{qbitF}|\varphi_{x',y'}\rangle) \tag{21}$$

Applying the Schmidt decomposition, it can be further represented as:

$$\frac{2}{N}\sum_{x'=0}^{\frac{N}{2}-1}\sum_{y'=0}^{\frac{N}{2}-1}|x',y'\rangle_{regL'}\otimes\frac{-i}{\sqrt{2}}\left(e^{i\theta_{x'y'}}|S^{+}_{x',y'}\rangle-e^{-i\theta_{x'y'}}|S^{-}_{x',y'}\rangle\right) \tag{22}$$

Where the state $|S^{\pm}_{x',y'}\rangle$ is defined as:

$$|S^{\pm}_{x',y'}\rangle=\frac{1}{\sqrt{2}}(|0\rangle_{qbitF}|\phi_{x',y'}\rangle\pm i|1\rangle_{qbitF}|\varphi_{x',y'}\rangle) \tag{23}$$

Step.5: Introducing $regA$ as the first register and running the QPE algorithm of the oracle G constructed in Figure. 5. Herein, the register $regA$ is employed to estimate the value of $\theta_{x',y'}$. Essentially this transformation can be regarded as the quantum amplitude estimation (QAE) [62]. Therefore, the output quantum state is an approximation of:

$$\frac{2}{N}\sum_{x',y'=0}^{\frac{N}{2}-1}|x',y'\rangle_{regL'}\left[\frac{-i}{\sqrt{2}}\left(e^{i\theta_{x',y'}}|\lambda_{x',y'}\rangle_{regA}|S^{+}_{x',y'}\rangle-e^{-i\theta_{x',y'}}|1-\lambda_{x',y'}\rangle_{regA}|S^{-}_{x',y'}\rangle\right)\right] \tag{24}$$

Let $\boldsymbol{\lambda_{x',y'}}$ denotes the decimal number of binary strings $\lambda_{x',y'}$, so we have $\boldsymbol{\lambda_{x',y'}}=\theta_{x',y'}/\pi$. According to (19) and (20), the relationship between the $\boldsymbol{h_{x',y'}}$ and the $\boldsymbol{\lambda_{x',y'}}$ can be described as:

$$\boldsymbol{h_{x',y'}}=8\sin^2(\pi\boldsymbol{\lambda_{x',y'}})-4 \tag{25}$$

To facilitate the understanding, a toy example of the quantum convolutional operation is given in Appendix A.

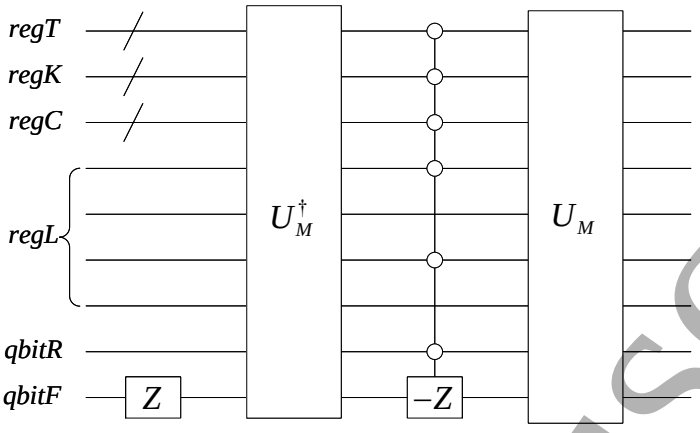


Figure 5. The quantum circuit to implement the oracle G.

Step.6: To make the linear quantum theory compatible with the nonlinear DCNN model, an additional quantum register $regH$ is introduced and a nonlinear module $U_{f(x)}$ is then employed. Since the $regA$ is encoded with basis state, $U_{f(x)}$ can be implemented based on the Theorem 2. Moreover, the composite function $f(x)$ is defined as:

$$f(x) = \sigma\left(8\sin^2 \pi x - 4\right) \tag{26}$$

Where $\sigma(\cdot)$ is an activation function. Note that the $\sin\left(\pi\left(\boldsymbol{\lambda}_{\boldsymbol{x'},\boldsymbol{y'}}\right)\right) = \sin\left(\pi\left(1 - \boldsymbol{\lambda}_{\boldsymbol{x'},\boldsymbol{y'}}\right)\right)$, thus the output quantum state after $U_{f(x)}$ is described as:

$$\frac{2}{N}\sum_{x',y'=0}^{\frac{N}{2}-1} |x',y'\rangle_{regL'}|\psi_{x',y'}\rangle|f(\lambda_{x',y'})\rangle_{regH} \tag{27}$$

In which, $|\psi_{x',y'}\rangle = \frac{-i}{\sqrt{2}}\left(e^{i\theta_{x',y'}}|\lambda_{x',y'}\rangle_{regA}|S^+_{x',y'}\rangle - e^{-i\theta_{x',y'}}|1-\lambda_{x',y'}\rangle_{regA}|S^-_{x',y'}\rangle\right)$.

To reuse the intermediate registers in successive layers, the U_M, U_R, and QPE are defined as a unitary transformation U_P. Its conjugate transpose matrix $U_P^\dagger$ is then applied in the circuit to realize uncomputation. The quantum state after $U_P^\dagger$ operator is written as follows:

$$\frac{2}{N}\sum_{x'=0}^{\frac{N}{2}-1}\sum_{y'=0}^{\frac{N}{2}-1} |x',y'\rangle_{regL'}|\mathbf{0}\rangle|f(\lambda_{x',y'})\rangle_{regH} \tag{28}$$

Where the bold notation $|\mathbf{0}\rangle$ represents the state in other registers. Finally, a ground of the swap gates is employed to exchange the state in $regH$ and $regC$:

$$\frac{2}{N}\sum_{x'=0}^{\frac{N}{2}-1}\sum_{y'=0}^{\frac{N}{2}-1} |x',y'\rangle_{regL'}|f(\lambda_{x',y'})\rangle_{regC}|\mathbf{0}\rangle \tag{29}$$

The state in (29) can be regarded as the input of the next quantum convolutional layer. Hence, the above procedure can be repeated to perform the successive multiple convolutional layers.

4.2.2. QDCNN with Multiple Feature Mappings In the classical CNN model, the volumes of multiple feature mappings are generally connected to subsequent volumes for extracting rich features information. Given an initial number of feature mappings in the first convolutional layer, the number of feature mappings is always doubled in the next layer. The corresponding quantum generalization is also demonstrated in the proposed model. Considering an example initializing with $P = 2^p$ feature mappings, an additional register $regM$ is introduced and the quantum circuit is shown in Figure. 6.

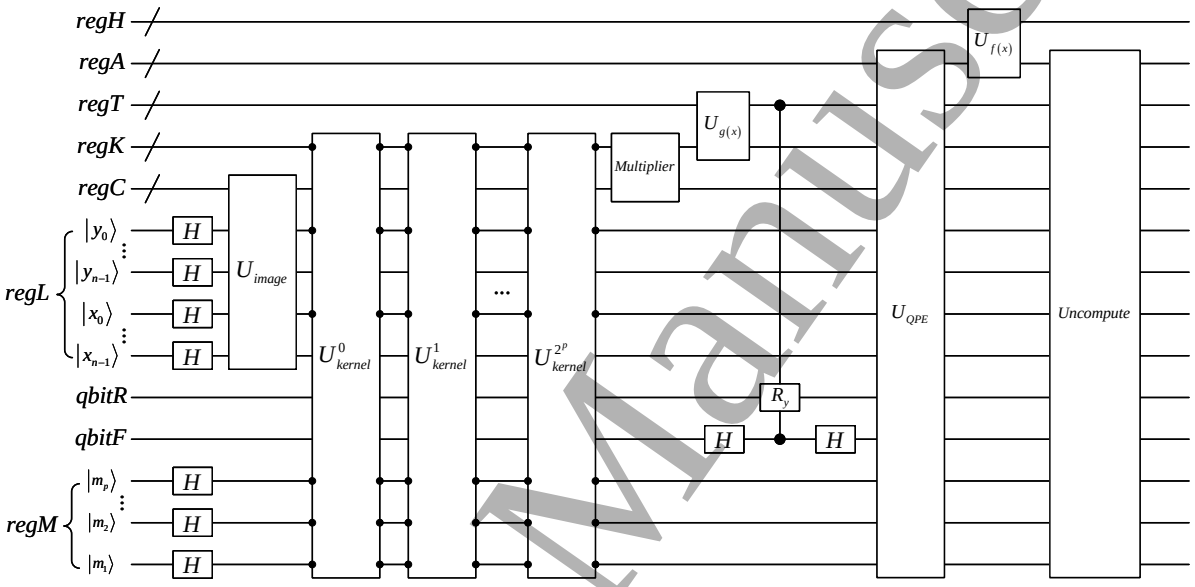


Figure 6. The quantum convolutional layer with multiple features. The notation of R_y is the simplified representation of rotation operation group in the Figure. 3.

Let demonstrate it with $p = 2$ and the value of m is also set as 1. Firstly, an extra control register $regM$ with two qubits is introduced and initialed as $|0\rangle^{\otimes 2}$. A Hadamard gate is then performed on the separate quantum line:

$$\frac{1}{2}\sum_{i=0}^{3}|i\rangle_{regM} = \frac{1}{2}\left(|00\rangle + |01\rangle + |10\rangle + |11\rangle\right) \tag{30}$$

Through the above operation, four copies of the input image are produced in the computational basis of $regM$ owing to the tensor format. In which, the i is regarded as the index of feature mappings and the multiple kernels can be organized by cooperating with the lowest qubit in the $|x\rangle_{regL}$ and $|y\rangle_{regL}$. Hence, the QRAM algorithm can be used to prepare the multiple convolutional kernels (four kernels in this case). Based on the basic model presented in Section 4.2.1, the multiple feature mappings can be computed in parallel. Besides, it should be noticed that the lowest qubits in location $|x\rangle_{regL}$ and $|y\rangle_{regL}$ are disentangled with the feature mappings. Therefore, these two qubits can be reused as the control qubits in the next convolutional layer without extra qubit resources. By this procedure, the multiple feature mappings can be computed in parallel and the number of the feature mappings in the subsequent convolution layers are given by $4, 16, 64, \cdots$.

4.2.3. Quantum Classified Layer The quantum classified layer corresponds to the fully connected layer in the classical DCNN framework. Once the size of feature mappings generated from the quantum convolutional layers becomes sufficiently small, a quantum classified layer is applied for classification.

In fact, based on the above feature mappings, many quantum classifiers mentioned previously can be employed to perform the classification tasks. But to ensure completeness, a simple linear quantum fully connected neural network (corresponds to the fully connected layer in the classical DCNN model) is discussed herein, which is based on the module presented in Section 4.2.1. For simplification, we considered it as a binary classifier and not hard to generalize it to multiple classification situation. The corresponding quantum circuit diagram is shown in Figure. 7.

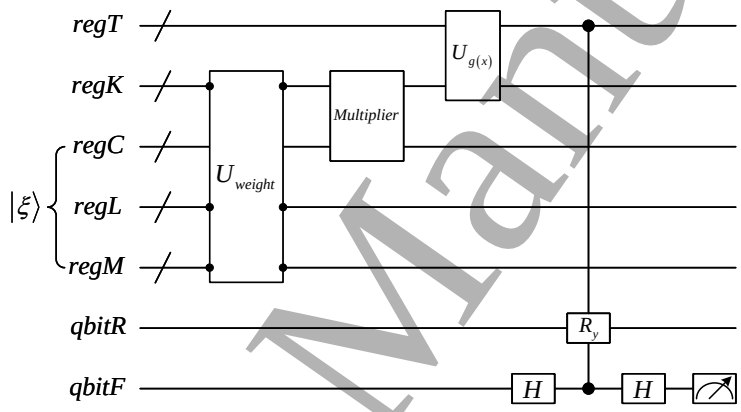


Figure 7. The design of quantum classified layer with one output.

Firstly, the input of the quantum classifier is denoted as $|\boldsymbol{\xi}\rangle$, which is consisted of $regM$, $regL$ and $regC$. It's obvious that $|\boldsymbol{\xi}\rangle$ can be interpreted as a quantum state with QRAM structure, where $regM$ and $regL$ serve as the address register and $regC$ is used to store the corresponding data. Similar to process in the Section 4.2.1, a unitary transformation U_{weight} is applied to prepared the weight quantum state $|\boldsymbol{\omega}\rangle$. Owing to the address of $|\boldsymbol{\omega}\rangle$ is shared with the $|\boldsymbol{\xi}\rangle$, the element-wise product between them can be evaluated efficiently by a quantum multiplier. Next, the unitary operation $U_{g(x)}$ is employed to realize the transformation in (13). After that, performing the quantum Hadamard test coupling with a group of controlled rotation operations to sum the weighted inputs. Finally, we employ the quantum projection measurement on the $qbitF$ and the probability distribution of the output is obtained. The measurement result of the $qbitF$ can be derived as:

$$\begin{cases} P_{qbitF}(0) = \frac{1}{2} + \frac{1}{2}\sum \xi * \omega \\ \\ P_{qbitF}(1) = \frac{1}{2} - \frac{1}{2}\sum \xi * \omega \end{cases} \tag{31}$$

Where the $\xi(\omega)$ is the single data on the quantum state $|\boldsymbol{\xi}\rangle$ ($|\boldsymbol{\omega}\rangle$). If a threshold function is applied to the output, then the network can be easily interpreted as a binary classifier.

More generally, there are two strategies for quantum multiple classification tasks: (1) For a k-categories classification problem, k binary classifiers are required and each classifier recognizes whether an image belongs to this class or not. The result of a binary classifier always be represented by single qubit, which can be written as $|\psi_{\text{result}}\rangle = \alpha|0\rangle + \beta|1\rangle$. The α^2 and β^2 denote the probability of label "yes" and "no" respectively. The maximal value among the result of k binary classifiers gives which category the image belongs to.
(2) Inspired by the one-hot coding in the classical situation, the above network can be extended into multiple outputs. For the k-categories classification problem, the $qbitF$ is expanded as k qubits and employed to represent the k output neurons. As for the example of the three-categories classification task, three qubits are prepared as follows:

$$|0\rangle^{\otimes 3} \rightarrow \frac{1}{\sqrt{3}}|001\rangle + \frac{1}{\sqrt{3}}|010\rangle + \frac{1}{\sqrt{3}}|100\rangle \tag{32}$$

The above preparation can be implemented according to the strategy proposed in [63]. The multiple U_{weight} transformations are then employed in the circuit by controlled operation respectively. Following the subsequent steps in Figure. 8, the measurement result in the forking qubits can be regarded as the outputs. The elements of the vector represent the probability distribution of categories and the position of the maximum element is considered as the determined result.

4.3. The Learning Process

In the training processing, a quantum-classical hybrid scheme is adopted, where the state evolution and measurement are performed on a quantum computer while the optimization is realized via an optimization algorithm running on a classical computer. Such a hybrid scheme is widely used in the context of quantum circuit learning [27, 28, 29, 30]. In the common variational quantum model, the parameters are always storing in the rotation gate and updating these angles by the classical training algorithm. The only difference is that in the training procedure of the QDCNN model, the model parameters to be optimized are the classical variable inputs of QRAM algorithm. Specifically, the learning process can be concluded as:

Definition:

- A prepared training set of quantum image is defined as $I = \left\{I^{(1)}, \cdots, I^{(m)}\right\}$, where $I^{(i)} \in I$ and the corresponding class label is $y^{(i)}$.
- The one-hot encoding of the ground truth label $y^{(i)}$ is given as $[y_1^{(i)} \quad y_2^{(i)} \quad \ldots \quad y_k^{(i)}]^T$, where k is the number of target classes.

(i) The probability distribution of the output is obtained by performing the quantum measurement on the qubits representing label information (output neurons):

$$f\left(I^{(i)}, \Omega\right) = [\bar{y}_1^{(i)} \quad \bar{y}_2^{(i)} \quad \ldots \quad \bar{y}_k^{(i)}]^T \tag{33}$$

where the Ω is denoting the parameters of QDCNN which is to be optimized. Obviously, the $\bar{y}_j^{(i)}$ is the probability of the j-th quantum basis state in (32). Correspondingly, the loss function is described as:

$$L\left(f\left(I^{(i)},\Omega\right),y^{(i)}\right)=-\sum_{j=0}^{k-1}\left(y_j^{(i)}*\log\overline{y}_j^{(i)}\right) \tag{34}$$

(ii) The updating parameters are calculated on a classical computer according to the Adam algorithm [64].

(iii) Applying the calculated parameter as the classical input of QRAM and repeat (i).

5. Experiments and Discussions

In this section, two image datasets MNIST [65] and the GTSRB [66] are employed to evaluate the QDCNN's ability of image recognition task. The visualization of two datasets is presented in Figure. 8. The experiment is performed with an ideal quantum simulator of Qiskit, which does not have quantum decoherence and gate errors. Therefore, the result from the simulator is considered as perfect without any quantum errors. Although it does have errors and inaccuracies which arise from the algorithm, such as the discretization error in encoding procedure. In the future experiment, the "Qiskit Aer" module (simulating quantum circuits with noise) could be employed for evaluating the effects of noise.

(a) (b)

Figure 8. (a) Examples of MNIST dataset. (b) Random representatives of the 43 traffic sign classes in the GTSRB dataset.

5.1. Datasets Preprocessing and Quantum Preparation

5.1.1. Handwritten Digit Classification The MNIST database of handwritten digits has a training set of 60,000 samples, and the remaining 10,000 samples are used for testing. The size of each image is $28\times28\times1$ and the digits have been size normalized and centered in a fixed size image. To meet the basic requirement of quantum image representation, the size of the original samples in the dataset are padded into $32\times32\times1$ with pixels valued 0.

5.1.2. German Traffic Sign Recognition The GTSRB dataset consists of 51,839 color images of size $32 \times 32 \times 3$. It can be divided into 43 categories (speed limit, no passing, priority road, no vehicles, no entry, double curve, etc) with imbalanced class distribution. Compared with handwritten digit recognition, GTSRB contains a lot of real objects of the actual world. It is not only noisy but also different in proportions and features, which brings great difficulty to the identification. In this experiment, the top ten categories are selected to reduce the negative impact caused by imbalanced class distribution. Besides, the original color images are converted into a gray-scale form for saving the computational resources.

After the classical preprocessing, the input image can be then prepared theoretically based on the method described in Section 4.1. However, there are different preparation circuits required for different input images though the subsequent circuit is the same. Hence, to solve this problem, a general input interface sub-procedure is designed for reading an input image and representing it as the corresponding quantum state automatically. The detailed implementation can be seen in Appendix B.

5.2. Experimental Results and Comparisons

In the experiment, the number of feature mappings is initialized as 4 and the number of quantum convolutional layers is given as 3. Besides, the epoch is set as 100 and the loss function is defined as the cross entropy.

The loss and accuracy curves of two datasets is shown in Figure. 9. In which, the experiment result on MNIST was a training accuracy of 99.16% and a test accuracy of 98.97%. Correspondingly, the training accuracy on the GTSRB is 97.45%, the validation accuracy is 92.56%, and the test accuracy of 91.40%. In addition, the confusion matrix on MNIST and GTSRB is also presented in Figure. 10 (a) and (b), respectively. The entries along main diagonals indicate correct classifications and all the elements in a confusion matrix are displayed by the intensity of background color.

Table 1. The performance comparison between the proposal against others.

Reference	**Dataset**	**Classes**	**Accuracy**
Chen et al. [49]	MNIST	10	97.62%
Dang et al. [67]	Caltech-101	9	78%
Edward et al.[68]	MNIST	2 (Is<4)	79.10±0.90%
		2 (Is even)	84.85±0.20%
		2 (0 or 1)	99.87±0.02%
		2 (2 or 7)	98.86±0.07%
This Paper	MNIST	10	98.97%
	GTSRB	10	91.40%
Classical Counterpart	MNIST	10	96.33±1.90%
	GTSRB	10	91.57±1.13%

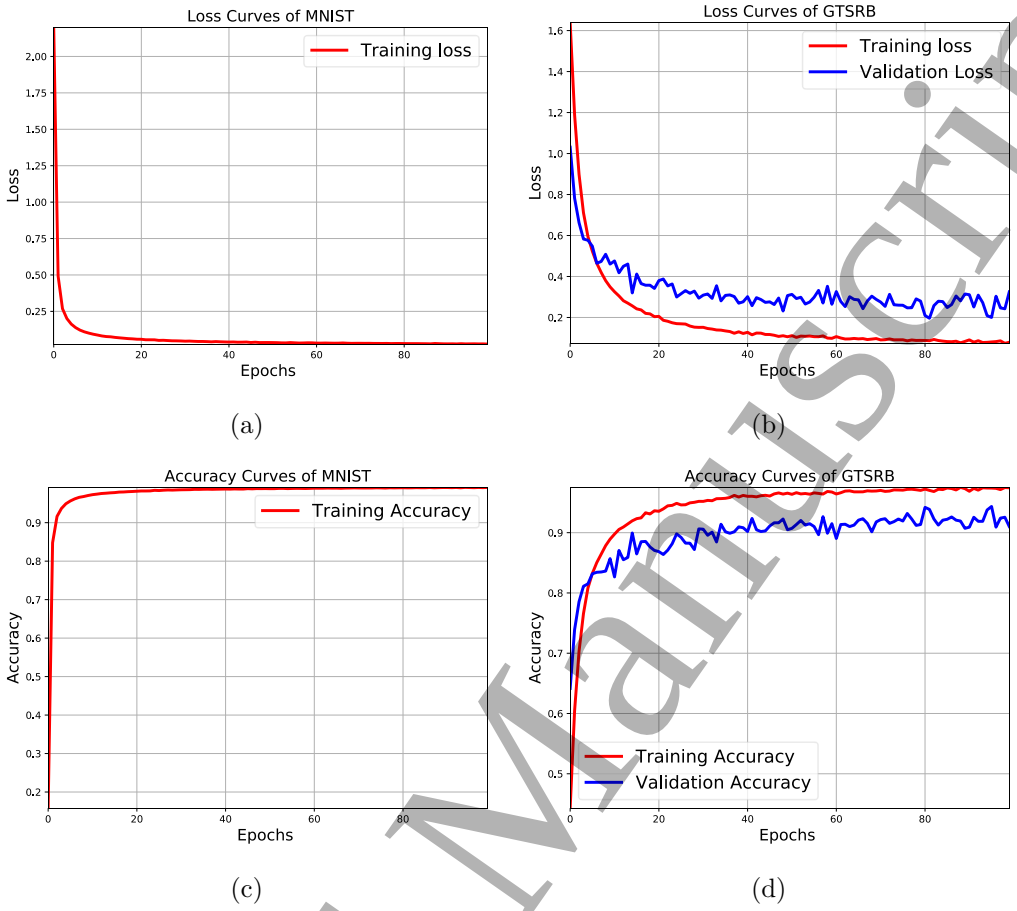
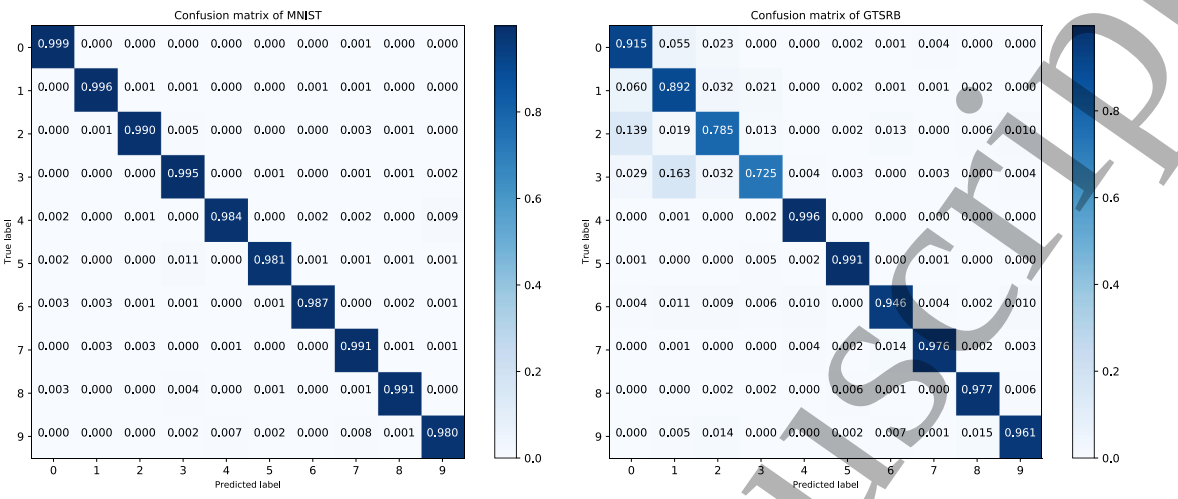


Figure 9. (a) and (c) represent the training loss curves the training accuracy curves on MNIST respectively. (b) and (d) denote the training/validation loss curves and the training/validation accuracy curves on GTSRB.

Table 1 shows the comparison of accuracy rates with four different quantum recognition schemes for several image datasets. The QPNN model proposed in [49] achieves 97.62% accuracy on the MNIST dataset. Besides, a hierarchical quantum classifier model [68] based on Multi-Scale Entanglement Renormalization Ansatz (MERA) performs a series of binary classification tasks on the same dataset and the corresponding recognition rate is also presented below. For the more challenging dataset, the scheme based on classical feature extraction and quantum K-Nearest Neighbor (QKNN) algorithm [67] achieve 78% and 83.1% accuracy on Caltech-101 and Graz-01 respectively. To the best of our knowledge, the experiment results show that the proposed model is superior to some existing quantum methods in performance. Since the QDCNN is a quantum reproduction of classical DCNN, it inherits the advantages of learning deep features as expected. Besides, an additional control experiment is performed on a classical DCNN model with the same structure and settings. The observations indicated that the QDCNN acquires almost comparable performance to the classical counterpart. Although there is no guarantee that QDCNN is always superior to the classical counterpart, the comparable performance also proves its effectiveness.



(a) (b)

Figure 10. Visualizing the performance of QDCNN on MNIST (a) and GTSRB (b).

In addition to the comparison of accuracy, an additional comparison between the QDCNN and the recent proposal in [39] (the other interesting quantum convolutional publication) is also presented herein. Specifically, it can be concluded as follows:

(i) The quanvolutional layer in [39] is constructed by a random quantum circuit and further replaces the classical counterpart with it for improving performance. The random quantum circuit doesn't reproduce the element-wise matrix multiplication operation in the classical situation though it may be more efficient. On the other hand, the main topic of this work is exploring the quantum generalization of DCNN for quantum speedup.

(ii) The number of transformations in a quanvolutional layer may be considerable though the QRAM requirement is not required. In our work, the quantum superposition is applied in the convolutional kernel storage and feature mapping representation, which shows the advantage of quantum parallelism.

(iii) In [39], the quantum measurement is adopted for decoding the quantum feature information, where the potential "quantum speedup" may be removed. In our work, the necessity of intermediate measurement or partial trace operation is eliminated.

5.3. Analysis of Computational Cost

5.3.1. Requirement of Qubit Counts Qubit resources plays an essential role in quantum computing. The number of qubits used in the proposed model and the corresponding function of all quantum register are concluded in Table 2. In conclusion, the number of required qubits in the proposed model is $2n + 5d + p + 2$. Obviously, the $5d + 2$ is fixed constant when the numerical accuracy $\varepsilon = 1/2^d$ is definite. Besides, the n and p are logarithmically increased with the input image size and the number of initial

feature mappings respectively. Hence, the proposed model fully utilizes the exponential advantages of storage resources in quantum superposition.

Table 2. The function of quantum register and its corresponding qubits requirement.

Register	Function	Number
$regL$	Storing the location information of image	$2n$
$regM$	Storing the initial multiple feature mappings	p
$regC$	Storing the color information of image	d
$regK$	Storing the weight information of filter	d
$regT$	Storing the intensity information of feature mapping	d
$regH$	Performing the nonlinear transformation	d
$regA$	The first register in the quantum phase estimation	d
$qbitF$	Producing quantum forking	1
$qbitR$	Performing the rotation operation	1

5.3.2. Network Complexity Analysis In this paper, the network complexity is measured in terms of the total number of elementary quantum gates (i.e. the standard 1- and 2-bit gate operations) and the constant terms in the complexity expression are omitted for simplification. It should be mentioned that we now consider the complexity of circuit constructions using a noiseless quantum computer (the overhead for fault-tolerant quantum computation is not considered herein, which is a separate issue). To make a comprehensive complexity analysis about the QDCNN, the whole network is divided into several modules: preparation modules based on QRAM (including U_{image}, U_{kernel} and U_{weight}), quantum arithmetic modules (including quantum multiplier, $U_{g(x)}$ and $U_{f(x)}$), controlled rotation module and QPE module. Some related parameters are defined as:

- The input image is single channel gray-scale with the size of $2^n \times 2^n$.
- The numerical accuracy of proposed model is defined as $\varepsilon = 1/2^d$.
- The number of feature mappings in the first convolutional layer is given as $P = 2^p$, and the number of convolutional layers is defined as m, where $m \leq n$.
- The number of output neurons is defined as k.

As for quantum convolutional layer, an $2^n \times 2^n$ image can be prepared in time $O(\text{poly}(n))$ according to the Theorem 1 while exponential operations are required in the classical situation. Besides, the complexity of U_{kernel} is associated with the parameter p and m, which is described as:

$$\begin{cases} O(\text{poly}(p)), & \text{the first convolutional layer} \\ O(1), & \text{the other convolutional layers} \end{cases} \tag{35}$$

Hence, the whole complexity of U_{kernel} is given as $O(\text{poly}(p) + m)$. According to the Theorem 2, the complexity of quantum arithmetic module is given by $O(\text{poly}(\log(1/\varepsilon)))$.

In addition, the basic component of the controlled rotation module is the two-qubit controlled rotation gate and its decomposition circuit is presented in the Figure. 11. Therefore, the circuit complexity of the controlled rotation module is $O(dm)$. Moreover, the QPE algorithm runs using $O(m/\varepsilon)$ controlled-U_G operations. Note that the

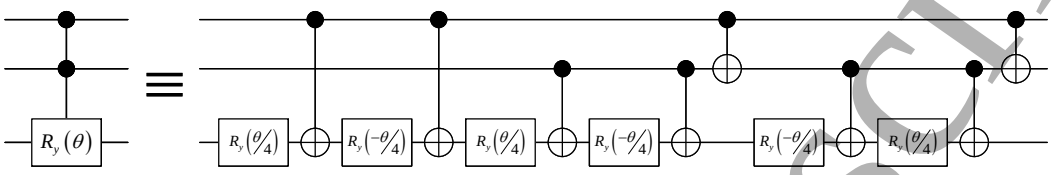


Figure 11. The decomposition of two-qubit controlled rotation gate.

uncomputation operation $U_P^\dagger$ doesn't influence the magnitude of the whole complexity. The group of swap gates in the last of the convolutional layer is given by $O(dm)$.

For quantum classified layer, the main cost is preparation of the connection weight. For k output neurons, the cost of the preparation process is given as $O(\text{poly}(p+n+k))$. Besides, the total number of operations in (32) is $O(k^2)$. And both complexity of the quantum multiplier and the $U_{g(x)}$ module is $O(\text{poly}(\log(1/\varepsilon)))$.

As discussed above, the QDCNN acquires an exponential acceleration in the whole computation procedure comparing with the classical counterpart. Nevertheless, it should be mentioned that an important prerequisite in the above model complexity analysis (as well as many other quantum algorithms) is the efficient implementation of QRAM. Although it is theoretically possible that the classical data can be efficiently prepared in a quantum superposition, its physical implementation may be far from trivial and remains an open question till now. It is necessary for larger quantum computers that may arise in the future. Some discussions and recent developments about QRAM are presented in Appendix C.

6. Conclusion and Future Work

In this paper, an explicit quantum model namely QDCNN is presented, which can be used to efficiently reproduce the classical DCNN with a certain structure. It takes full the parallel advantage of the quantum paradigm in both storage and computation, which provides an exponential acceleration comparing with the classical counterpart. Besides, a competitive result of the numerical experiment indicated the feasibility and validity of the QDCNN model in multi-class image recognition.

Clearly, some limitations need to be resolved in future research. Firstly, the size of the input image is constrained as $2^n \times 2^n$ in the quantum preparation phase, and the additional image-scaling is required for the more general case. Secondly, although the stride of 2^m in convolution operation is feasible, the arbitrary stride and kernel should be further explored.

Acknowledgments

This work is supported by the National Key R&D Plan under Grant No. 2018YFC1200200 and 2018YFC1200205.

References

[1] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, 1982.

[2] D. Deutsch. Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 400(1818):97–117, 1985.

[3] Ashley Montanaro. Quantum algorithms: an overview. *npj Quantum Information*, 2(1):15023, 2016.

[4] Jacob Biamonte, Peter Wittek, Nicola Pancotti, et al. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[5] Nathan Wiebe, Daniel Braun, and Seth Lloyd. Quantum algorithm for data fitting. *Physical Review Letters*, 109(5):050505, 2012.

[6] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631–633, 2014.

[7] Songfeng Lu and Samuel L. Braunstein. Quantum decision tree classifier. *Quantum Information Processing*, 13(3):757–770, 2014.

[8] Guang Hao Low, Theodore J. Yoder, and Isaac L. Chuang. Quantum inference on Bayesian networks. *Physical Review A*, 89(6):062315, 2014.

[9] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical Review Letters*, 113(3):130503, 2014.

[10] Iris Cong and Luming Duan. Quantum discriminant analysis for dimensionality reduction and classification. *New Journal of Physics*, 18(7):073011, 2016.

[11] Guoming Wang. Quantum algorithm for linear regression. *Physical Review A*, 96(1):012335, 2017.

[12] Nathan Wiebe, Ashish Kapoor, and Krysta M Svore. Quantum Nearest-Neighbor Algorithms for Machine Learning. *Quantum Information and Computation*, 15(3-4):318–358, 2018.

[13] YaoChong Li, Ri-Gui Zhou, RuiQing Xu, et al. A quantum mechanics-based framework for EEG signal feature extraction and classification. *IEEE Transactions on Emerging Topics in Computing*, 2020.

[14] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, 2018.

[15] Steven H. Adachi and Maxwell P. Henderson. Application of Quantum Annealing to Training of Deep Neural Networks. *arXiv preprint arXiv:1510.06356*, 2015.

[16] Max Wilson, Thomas Vandal, Tad Hogg, et al. Quantum-assisted associative adversarial network: Applying quantum annealing in deep learning. *arXiv preprint arXiv:1904.10573*, 2019.

[17] Guillaume Verdon, Michael Broughton, and Jacob Biamonte. A quantum algorithm to train neural networks using low-depth circuits. *arXiv preprint arXiv:1712.05304*, 2017.

[18] Nathan Killoran, Thomas R. Bromley, Juan Miguel Arrazola, et al. Continuous-variable quantum neural networks. *Physical Review Research*, 1(3):033063, 2019.

[19] Edward Farhi and Hartmut Neven. Classification with Quantum Neural Networks on Near Term Processors. *arXiv preprint arXiv:1802.06002*, 2018.

[20] Sirui Lu, Lu-Ming Duan, and Dong-Ling Deng. Quantum Adversarial Machine Learning. *arXiv preprint arXiv:2001.00030*, 2019.

[21] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[22] T. D. Ladd, F. Jelezko, R. Laflamme, et al. Quantum computers. *Nature*, 464(7285):45–53, 2010.

[23] C. Monroe and J. Kim. Scaling the Ion Trap Quantum Processor. *Science*, 339(6124):1164–1169, 2013.

[24] Frank Arute, Kunal Arya, Ryan Babbush, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[25] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. The quest for a Quantum Neural Network. *Quantum Information Processing*, 13(11):2567–2586, 2014.

[26] Adenilton José da Silva, Teresa Bernarda Ludermir, and Wilson Rosa de Oliveira. Quantum perceptron over a field and neural network architecture selection in a quantum computer. *Neural Networks*, 76:55–64, 2016.

[27] S. Gammelmark and K. Mølmer. Quantum learning by measurement and feedback. *New Journal of Physics*, 11(3):033017, 2009.

[28] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, et al. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1):4213, 2014.

[29] Jeongho Bang, Junghee Ryu, Seokwon Yoo, et al. A strategy for quantum algorithm design assisted by machine learning. *New Journal of Physics*, 16(7):073017, 2014.

[30] Jarrod R. McClean, Jonathan Romero, Ryan Babbush, et al. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, 2016.

[31] Vittorio Giovannetti, Seth Lloyd, and Lorenzo MacCone. Quantum random access memory. *Physical Review Letters*, 100(16):160501, 2008.

[32] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Architectures for a quantum random access memory. *Physical Review A*, 78(5):052310, 2008.

[33] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. Simulating a perceptron on a quantum computer. *Physics Letters A*, 379(7):660–663, 2015.

[34] Alexandre Y. Yamamoto, Kyle M. Sundqvist, Peng Li, et al. Simulation of a Multidimensional Input Quantum Perceptron. *Quantum Information Processing*, 17(6):128, 2018.

[35] Changpeng Shao. Quantum Speedup of Training Radial Basis Function Networks. *Quantum Information and Computation*, 19(7–8):609–625, 2019.

[36] Massimo Panella and Giuseppe Martinelli. Neural networks with quantum architecture and quantum learning. *International Journal of Circuit Theory and Applications*, 39(1):61–77, 2011.

[37] Fernando M. de Paula Neto, Teresa B Ludermir, Wilson R. de Oliveira, et al. Implementing Any Nonlinear Quantum Neuron. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–6, 2019.

[38] Kerstin Beer, Dmytro Bondarenko, Terry Farrelly, et al. Training deep quantum neural networks. *Nature Communications*, 11(1):808, 2020.

[39] Maxwell Henderson, Samriddhi Shakya, Shashindra Pradhan, et al. Quanvolutional neural networks: powering image recognition with quantum circuits. *Quantum Machine Intelligence*, 2(1):1–9, 2020.

[40] S Kak. On quantum neural computing. *Information Sciences*, 83(3-4):143–160, 1995.

[41] S. K. Jeswal and S. Chakraverty. Recent Developments and Applications in Quantum Neural Network: A Review. *Archives of Computational Methods in Engineering*, 26(4):793–807, 2019.

[42] Noriaki Kouda, Nobuyuki Matsui, Haruhiko Nishimura, et al. Qubit neural network and its learning efficiency. *Neural Computing and Applications*, 14(2):114–121, 2005.

[43] Panchi Li, Hong Xiao, Fuhua Shang, et al. A hybrid quantum-inspired neural networks with sequence inputs. *Neurocomputing*, 117:81–90, 2013.

[44] Panchi Li and Hong Xiao. Sequence Input-Based Quantum-Inspired Neural Networks with Applications. *Neural Processing Letters*, 40(2):143–168, 2013.

[45] Rigui Zhou and Qiulin Ding. Quantum M-P neural network. *International Journal of Theoretical Physics*, 46(12):3209–3215, 2007.

[46] Huaixin Cao, Feilong Cao, and Dianhui Wang. Quantum artificial neural networks with applications. *Information Sciences*, 290:1–6, 2015.

[47] Michail Zak and Colin P. Williams. Quantum neural nets. *International Journal of Theoretical*

Physics, 37(2):651–684, 1998.

[48] Kwok Ho Wan, Oscar Dahlsten, Hlér Kristjánsson, et al. Quantum generalisation of feedforward neural networks. *npj Quantum Information*, 3(1):36, 2017.

[49] Jialin Chen, Lingli Wang, and Edoardo Charbon. A quantum-implementable neural network model. *Quantum Information Processing*, 16(10):245, 2017.

[50] Fernando M. de Paula Neto, Teresa B Ludermir, Wilson R. de Oliveira, et al. Implementing Any Nonlinear Quantum Neuron. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–6, 2019.

[51] Patrick Rebentrost, Thomas R. Bromley, Christian Weedbrook, et al. Quantum Hopfield neural network. *Physical Review A*, 98(4):042308, 2018.

[52] Francesco Tacchino, Chiara Macchiavello, Dario Gerace, et al. An artificial neuron implemented on an actual quantum processor. *npj Quantum Information*, 5(1):26, 2019.

[53] Jian Zhao, Yuan-Hang Zhang, Chang-Peng Shao, et al. Building quantum neural networks based on a swap test. *Physical Review A*, 100(1):012334, 2019.

[54] Iris Cong, Soonwon Choi, and Mikhail D. Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019.

[55] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. 2000.

[56] Tara N. Sainath, Brian Kingsbury, George Saon, et al. Deep Convolutional Neural Networks for Large-scale Speech Tasks. *Neural Networks*, 64:39–48, 2015.

[57] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[58] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[59] Lidia Ruiz-Perez and Juan Carlos Garcia-Escartin. Quantum arithmetic with the quantum Fourier transform. *Quantum Information Processing*, 16(6):152, 2017.

[60] S. S. Zhou, T. Loke, J. A. Izaac, et al. Quantum Fourier transform in computational basis. *Quantum Information Processing*, 16(3):82, 2017.

[61] Shuang Wu, Guanrui Wang, Pei Tang, et al. Convolution with even-sized kernels and symmetric padding. In *Advances in Neural Information Processing Systems 32*, pages 1194–1205. 2019.

[62] Gilles Brassard, Peter Høyer, Michele Mosca, et al. Quantum Amplitude Amplification and Estimation. *arXiv preprint arXiv:quant-ph/0005055*, 2000.

[63] Gui-Lu Long and Yang Sun. Efficient scheme for initializing a quantum register with an arbitrary superposed state. *Physical Review A*, 64(1):014303, 2001.

[64] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[65] Yann Lecun, Leon Bottou, Yoshua Bengio, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[66] J. Stallkamp, M. Schlipsing, J. Salmen, et al. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, 2012.

[67] Yijie Dang, Nan Jiang, Hao Hu, et al. Image classification based on quantum K-Nearest-Neighbor algorithm. *Quantum Information Processing*, 17(9):239, 2018.

[68] Edward Grant, Marcello Benedetti, Shuxiang Cao, et al. Hierarchical quantum classifiers. *npj Quantum Information*, 4(1):65, 2018.

[69] Scott Aaronson. Read the fine print. *Nature Physics*, 11(4):291–293, 2015.

[70] A Prakash. *Quantum Algorithms for Linear Algebra and Machine Learning*. PhD thesis, 2014.

[71] John A. Cortese and Timothy M. Braje. Loading Classical Data into a Quantum Computer. *arXiv preprint arXiv:1803.01958*, 2018.

[72] Guang Hao Low, Vadym Kliuchnikov, and Luke Schaeffer. Trading T-gates for dirty qubits in state preparation and unitary synthesis. *arXiv preprint arXiv:1812.00954*, 2018.

[73] Daniel K. Park, Francesco Petruccione, and June-Koo Kevin Rhee. Circuit-Based Quantum Random Access Memory for Classical Data. *Scientific Reports*, 9(1):3949, 2019.

[74] Alexandru Paler, Oumarou Oumarou, and Robert Basmadjian. Constant Depth Bucket Brigade Quantum RAM Circuits Without Introducing Ancillae. *arXiv preprint arXiv:2002.09340*, 2020.

[75] Connor T. Hann, Chang-Ling Zou, Yaxing Zhang, et al. Hardware-efficient quantum random access memory with hybrid quantum acoustic systems. *Phys. Rev. Lett.*, 123:250501, 2019.

[76] Srinivasan Arunachalam, Vlad Gheorghiu, Tomas Jochym-O'Connor, et al. On the robustness of bucket brigade quantum RAM. *New Journal of Physics*, 17(12):123010, 2015.

Appendix A. A Toy Example of the Quantum Convolutional Operation

To facilitate the understanding, a simple toy example for implementing quantum convolutional operation is designed and presented in this appendix. The related experimental settings are given as follows:

- The input image is binary and its size is given as 4×4.
- The size of convolutional kernel is set as 2×2 while the stride is given as 2.
- The number of encoding bits is defined as $d = 8$.

As shown in the Figure A1, the input image is a 4×4 binary image, where the black block is encoded with 0 and the white block is encoded with 1. And the corresponding convolutional kernel is also defined and encoded with eight-digit binary number.

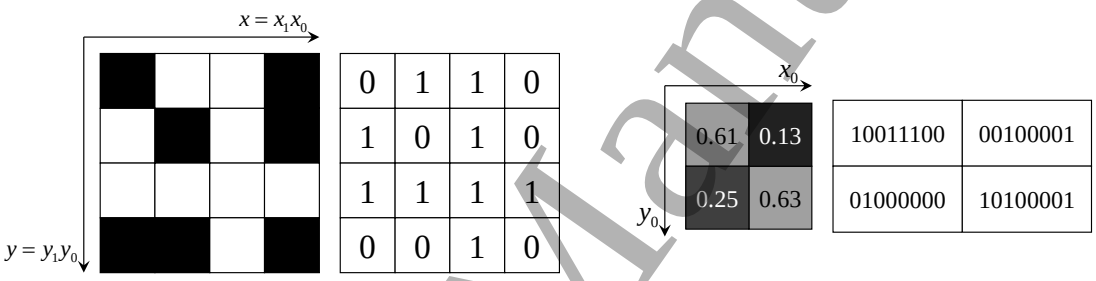


Figure A1. The left sub-figure is the input image while the right sub-figure is the corresponding convolutional filter.

The detailed quantum circuit is presented in Figure A2, which can be divided into six parts. The input image is prepared in the first part, where $|x_1x_0y_1y_0\rangle$ is denoted as the location information while the color information is given by $|c_0\rangle$. In the second part, the convolutional filter is prepared and the element-wise multiplication is implemented by adding an additional control qubit $|c_0\rangle$. Since the input variable is explicit, the transformation $U_{g(x)}$ is designed with simple Boolean function mapping shown in the third part. Specifically, the transformation in third part can be described as follows:

$$\begin{array}{ll} |10011100\rangle & |10010110\rangle \\ |00100001\rangle & |11101011\rangle \\ |01000000\rangle \Rightarrow & |11010111\rangle \\ |10100001\rangle & |10010001\rangle \\ |00000000\rangle & |11111111\rangle \end{array} \tag{A.1}$$

The fourth part is to produce the quantum forking and assign the quantum state to the amplitude, which corresponds to the transformations in (14)-(16). For example, as for the state in $|11101011\rangle$, the $|0\rangle_{qbitR}$ is rotated as $\cos\frac{235\pi}{512}|0\rangle_{qbitR} + \sin\frac{235\pi}{512}|1\rangle_{qbitR}$ if and only if $qbitF$ is in the state $|1\rangle$. In which, the $\frac{235\pi}{512}$ is computed by:

$$\frac{235\pi}{512} = \frac{\pi}{4}\times 1 + \frac{\pi}{8}\times 1 + \frac{\pi}{16}\times 1 + \frac{\pi}{32}\times 0 + \frac{\pi}{64}\times 1 + \frac{\pi}{128}\times 0 + \frac{\pi}{256}\times 1 + \frac{\pi}{512}\times 1 \tag{A.2}$$

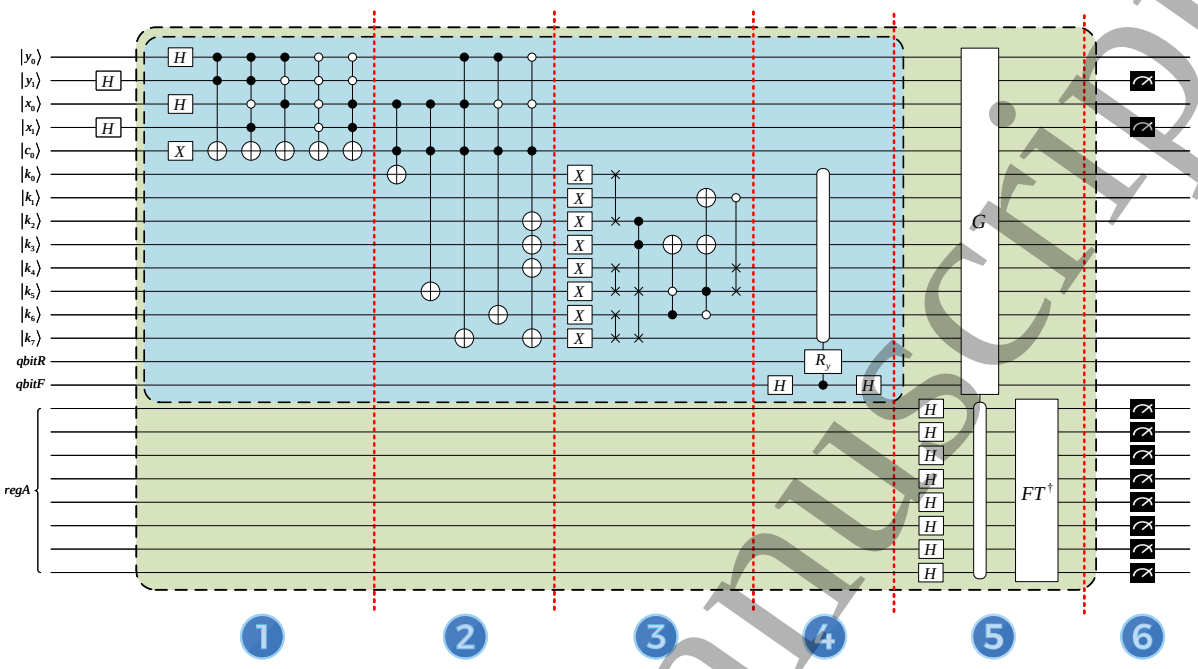


Figure A2. The detailed circuit of toy example for the quantum convolutional layer.

Noticed that $\cos\frac{235\pi}{512} = 0.1285$, which is is a good approximation to 0.13 in the filter. After that, the quantum phase estimation algorithm with oracle G (the construction is refer to the Figure 5 in main text) is performed in part five. It is easy to check:

$$\begin{aligned}
\langle u_{00}|v_{00}\rangle &= \frac{\boldsymbol{h}_{00}}{4} = -\cos 2\theta_{00} = \frac{1}{4}\cos\frac{255\pi}{512} + \frac{1}{4}\cos\frac{215\pi}{512} + \frac{1}{4}\cos\frac{235\pi}{512} + \frac{1}{4}\cos\frac{255\pi}{512} = 0.097\\
\langle u_{01}|v_{01}\rangle &= \frac{\boldsymbol{h}_{01}}{4} = -\cos 2\theta_{01} = \frac{1}{4}\cos\frac{150\pi}{512} + \frac{1}{4}\cos\frac{255\pi}{512} + \frac{1}{4}\cos\frac{235\pi}{512} + \frac{1}{4}\cos\frac{255\pi}{512} = 0.187\\
\langle u_{10}|v_{10}\rangle &= \frac{\boldsymbol{h}_{10}}{4} = -\cos 2\theta_{10} = \frac{1}{4}\cos\frac{150\pi}{512} + \frac{1}{4}\cos\frac{215\pi}{512} + \frac{1}{4}\cos\frac{255\pi}{512} + \frac{1}{4}\cos\frac{255\pi}{512} = 0.217\\
\langle u_{11}|v_{11}\rangle &= \frac{\boldsymbol{h}_{11}}{4} = -\cos 2\theta_{11} = \frac{1}{4}\cos\frac{150\pi}{512} + \frac{1}{4}\cos\frac{215\pi}{512} + \frac{1}{4}\cos\frac{235\pi}{512} + \frac{1}{4}\cos\frac{255\pi}{512} = 0.247
\end{aligned} \tag{A.3}$$

In theory, the desired output mapping can be described as a matrix:

$$\begin{aligned}
&\begin{bmatrix} 0.61\times 0 + 0.25\times 1 + 0.13\times 1 + 0.63\times 0 & 0.61\times 1 + 0.25\times 1 + 0.13\times 0 + 0.63\times 0 \\ 0.61\times 1 + 0.25\times 0 + 0.13\times 1 + 0.63\times 0 & 0.61\times 1 + 0.25\times 1 + 0.13\times 1 + 0.63\times 0 \end{bmatrix}\\
&= \begin{bmatrix} 0.38 & 0.86 \\ 0.74 & 0.99 \end{bmatrix}
\end{aligned} \tag{A.4}$$

From (A.3), the experiential result of feature mapping is given as:

$$\begin{bmatrix} \boldsymbol{h}_{00} & \boldsymbol{h}_{10} \\ \boldsymbol{h}_{01} & \boldsymbol{h}_{11} \end{bmatrix} = \begin{bmatrix} 0.388 & 0.868 \\ 0.748 & 0.988 \end{bmatrix} \tag{A.5}$$

Obviously, each element in (A.5) is a good approximation of the theoretical result in (A.4). Nevertheless, it should be mentioned that the register $regA$ is only the estimation of $\lambda_{x'y'}$ and $1-\lambda_{x'y'}$ as demonstrated in (24). To verify this, a quantum measurement is performed on $|x_1y_1\rangle$ and $regA$ as presented in the part six. The measurement result of

regA in $|x_1y_1\rangle = |00\rangle$ is presented in the Figure A3 (the size of full measurement results is too large, thus the segment of $|x_1y_1\rangle = |00\rangle$ is given herein). As presented in the Figure A3, the λ_{00} is estimated as $|01000100\rangle$ (i.e. $\boldsymbol{\lambda}_{00} = 0.2656$). Finally, substituting the $\boldsymbol{\lambda}_{00}$ into the (25), the $\boldsymbol{h}_{00} = 0.376$ is obtained. Moreover, the whole measurement result of *regA* and its corresponding $\boldsymbol{h}_{x'y'}$ are presented in Table A1.

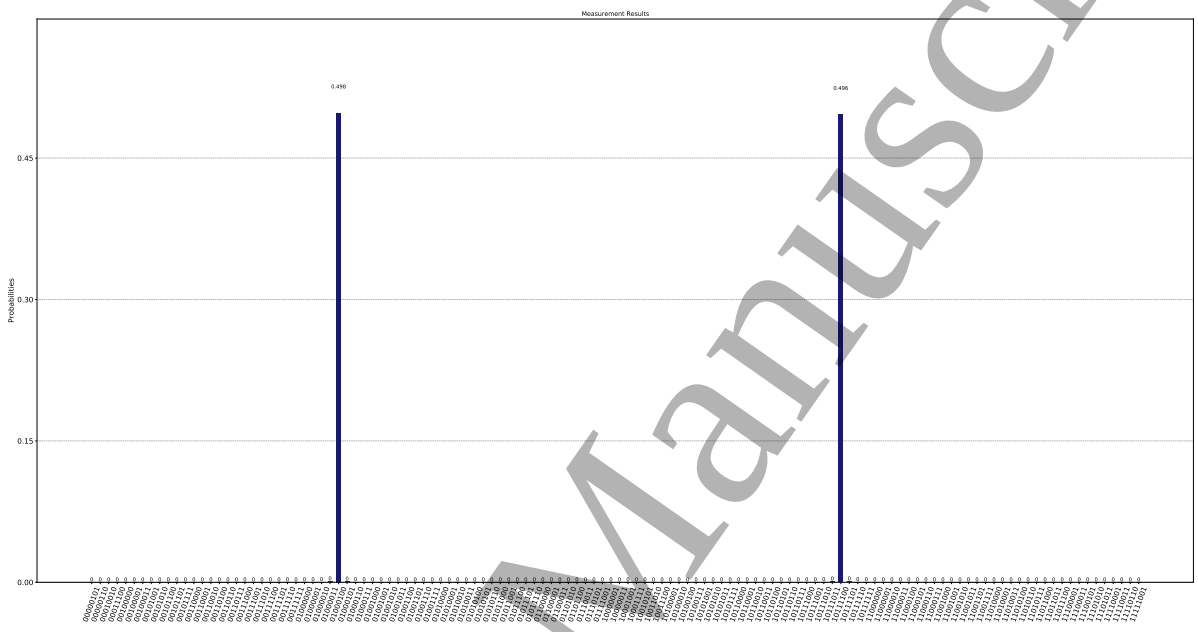


Figure A3. The measurement result of *regA* in $|x_1y_1\rangle = |00\rangle$.

Table A1. The whole measurement results of *regA* and its corresponding $\boldsymbol{h}_{x'y'}$.

$\|x_1y_1\rangle$ ($\|x'y'\rangle$)	$\lambda_{x'y'}$	$\boldsymbol{\lambda}_{x'y'}$	$\boldsymbol{h}_{x'y'}$	Expectation of $\boldsymbol{h}_{x'y'}$
$\|00\rangle$	$\|01000100\rangle$	0.2656	0.3914	0.38
$\|01\rangle$	$\|01000111\rangle$	0.2799	0.7471	0.74
$\|10\rangle$	$\|01001000\rangle$	0.2848	0.8677	0.86
$\|11\rangle$	$\|01001010\rangle$	0.2897	0.9875	0.99

Appendix B. Implementation of Input Interface Sub-Procedure

In order to generate the corresponding preparation oracle for different input images automatically, an input interface sub-procedure is designed and its flow diagram is shown in Figure B1.

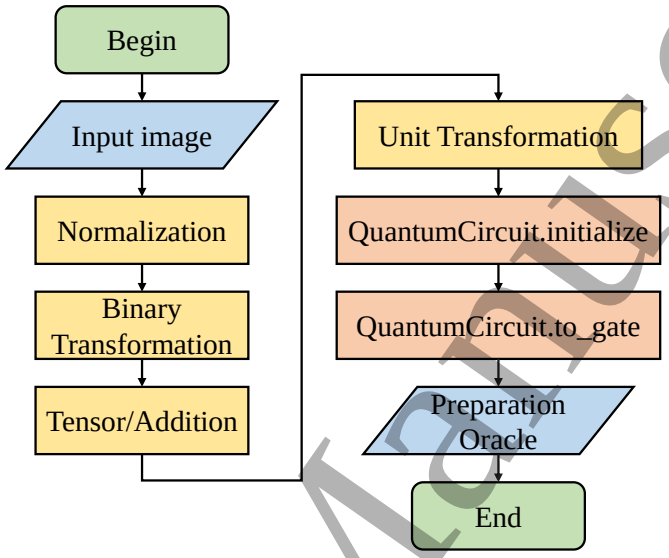


Figure B1. The flow diagram of input interface sub-procedure.

Specifically, this sub-procedure can be described as follows:

(i) *Calculating image vector in Hilbert space (blocks filled with yellow)* Given a preprocessed input image, the normalization is firstly performed on it, thus all pixel values are squashed to range between 0 and 1 (corresponds to the Equation (6) in the main text). Next, each pixel value is converted into the d binary fractions format (corresponds to the Equation (7) in the main text). After that, the binary string that represents location information and color information are connected and represented as a vector in the Hilbert space by performing tensor product. The whole image is represented as a vector by adding up all pixel vectors. Finally, the unit transformation is applied to the image vector.

(ii) *Obtaining preparation oracle of the input image (blocks filled with red)* Taking the unit vector in (i) as input, the corresponding preparation oracle can be obtained by two methods provided in Qiskit. Finally, the preparation oracle can be appended to the beginning of the complete circuit.

Appendix C. Some Discussions and Recent Developments about QRAM

QRAM is a quantum equivalent to classical RAM and it is considered as an essential component of many quantum algorithms. And the speedup of many quantum machine learning algorithms hinge crucially on the low qRAM queries. As pointed in [69], some well-known problems related to QRAM can be concluded as follows:

(i) The physical resource is exponential scaling with respect to the number of qubits.

(ii) Whether all the components require to be error-corrected.

(iii) The requirement of uniformly distributed data over a quantum register.

Notation: In the discussions below, N is the scale of classical data and $n = \log N$

Discussion on the issue 1: By analyzing the existing proposals towards QRAM [70, 71, 72, 73, 74], it is found that there are always one costs $O(N)$ between running time and resources. Although the QRAM oracle is criticized for the requirement of large physical resources, recent work [75] has proven possible the practical implementation of the QRAM oracle. In which, Connor T. Hann et al. presented a hardware efficient implementation of QRAM using the multimode circuit quantum acoustodynamic (cQAD) system. The physical resource requirement can be reduced drastically comparing with original cavity-implementation in [32], which is due to the compactness of multimode cQAD systems.

Discussion on the issue 2: In the QRAM, there are only $O(n)$ routing components (where N is the scale of classical data and $n = \log N$) is the need to be "active" while the others can be considered as "non-active" and "error-free". As presented in the [76], given a certain error model, algorithms that require to query the memory a polynomial number of times (like the quantum linear system algorithm) might not require fault-tolerant components. However, for super polynomial query algorithms (like Grover's search), the QRAM requires error-corrected components. Similar to the quantum linear system algorithm, the fault-tolerant components are not required in our proposed algorithm owing to the polynomial query.

Discussion on the issue 3: Considering the QRAM process as follows:

$$\sum_j \alpha_j |j\rangle|0\rangle \xrightarrow{QRAM} \sum_j \alpha_j |j\rangle|b_j\rangle \tag{C.1}$$

where $\sum_j \alpha_j |j\rangle$ is a superposition of queried address and $|b_j\rangle$ represents the content of the j-th memory location. In our work, the input image is prepared according to the above equation, where the all $\alpha_j = \frac{1}{\sqrt{N}}$. Hence, the requirement of relatively uniform distribution is completely satisfied.