Wei *et al. AAPPS Bulletin* (2022) 32:2
https://doi.org/10.1007/s43673-021-00030-3

AAPPS Bulletin

ORIGINAL ARTICLE Open Access

A quantum convolutional neural network on NISQ devices

ShiJie Wei[1,2], YanHu Chen[3], ZengRong Zhou[1,2] and GuiLu Long[1,2,4,5*]

Abstract

Quantum machine learning is one of the most promising applications of quantum computing in the noisy intermediate-scale quantum (NISQ) era. We propose a quantum convolutional neural network(QCNN) inspired by convolutional neural networks (CNN), which greatly reduces the computing complexity compared with its classical counterparts, with $O((log_2 M)^6)$ basic gates and $O(m^2 + e)$ variational parameters, where M is the input data size, m is the filter mask size, and e is the number of parameters in a Hamiltonian. Our model is robust to certain noise for image recognition tasks and the parameters are independent on the input sizes, making it friendly to near-term quantum devices. We demonstrate QCNN with two explicit examples. First, QCNN is applied to image processing, and numerical simulation of three types of spatial filtering, image smoothing, sharpening, and edge detection is performed. Secondly, we demonstrate QCNN in recognizing image, namely, the recognition of handwritten numbers. Compared with previous work, this machine learning model can provide implementable quantum circuits that accurately corresponds to a specific classical convolutional kernel. It provides an efficient avenue to transform CNN to QCNN directly and opens up the prospect of exploiting quantum power to process information in the era of big data.

Keywords: Quantum computing, Quantum algorithm, Quantum machine learning

1 Introduction

Machine learning has fundamentally transformed the way people think and behave. Convolutional neural network (CNN) is an important machine learning model which has the advantage of utilizing the correlation information of data, with many interesting applications ranging from image recognition to precision medicine.

Quantum information processing (QIP) [1, 2], which exploits quantum-mechanical phenomena such as quantum superpositions and quantum entanglement, allows one to overcome the limitations of classical computation and reaches higher computational speed for certain problems [3–5]. Quantum machine learning, as an interdisciplinary study between machine learning and quantum information, has undergone a flurry of developments in recent years [6–15]. Machine learning algorithm consists of three components: representation, evaluation and optimization, and the quantum version [16–20] usually concentrates on realizing the evaluation part, the fundamental construct in deep learning [21].

A CNN generally consists of three layers, convolution layers, pooling layers, and fully connected layers. The convolution layer calculates new pixel values $x_{ij}^{(\ell)}$ from a linear combination of the neighborhood pixels in the preceding map with the specific weights, $x_{i,j}^{(\ell)} = \sum_{a,b=1}^{m} w_{a,b} x_{i+a-2,j+b-2}^{(\ell-1)}$, where the weights $w_{a,b}$ form a $m \times m$ matrix named as a convolution kernel or a filter mask. Pooling layer reduces feature map size, e.g., by taking the average value from four contiguous pixels, and is often followed by application of a nonlinear (activation) function. The fully connected layer computes the final output by a linear combination of all remaining pixels with specific weights determined by parameters in a

*Correspondence: gllong@tsinghua.edu.cn
[1]Beijing Academy of Quantum Information Sciences, Beijing 100193, China
[2]State Key Laboratory of Low-Dimensional Quantum Physics and Department of Physics, Tsinghua University, Beijing 100084, China
Full list of author information is available at the end of the article

fully connected layer. The weights in the filter mask and fully connected layer are optimized by training on large datasets.

In this article, we demonstrate the basic framework of a quantum convolutional neural network (QCNN) by sequentially realizing convolution layers, pooling layers, and fully connected layers. Firstly, we implement convolution layers based on linear combination of unitary operators (LCU) [22–24]. Secondly, we abandon some qubits in the quantum circuit to simulate the effect of the classical pooling layer. Finally, the fully connected layer is realized by measuring the expectation value of a parametrized Hamiltonian and then a nonlinear (activation) function to post-process the expectation value. We perform numerical demonstrations with two examples to show the validity of our algorithm. Finally, the computing complexity and trainability of our QCNN model are discussed followed by a summary.

2 Results

2.1 Framework of quantum neural networks

2.1.1 Quantum convolution layer

The first step for performing quantum convolution layer is to encode the image data into a quantum system. In this work, we encode the pixel positions in the computational basis states and the pixel values in the probability amplitudes, forming a pure quantum state (Fig. 1). Given a 2D image $F = (F_{i,j})_{M\times L}$, where $F_{i,j}$ represents the pixel value at position (i,j) with $i = 1,\ldots,M$ and $j = 1,\ldots,L$. F is transformed as a vector $\vec{f}$ with ML elements by putting the first column of F into the first M elements of $\vec{f}$, the second column the next M elements , etc. That is,

$$\vec{f} = (F_{1,1}, F_{2,1}, \ldots, F_{M,1}, F_{1,2}, \ldots, F_{i,j}, \ldots, F_{M,L})^T. \quad (1)$$

Accordingly, the image data $\vec{f}$ can be mapped onto a pure quantum state $|f\rangle = \sum_{k=0}^{2^n-1} c_k|k\rangle$ with $n =$

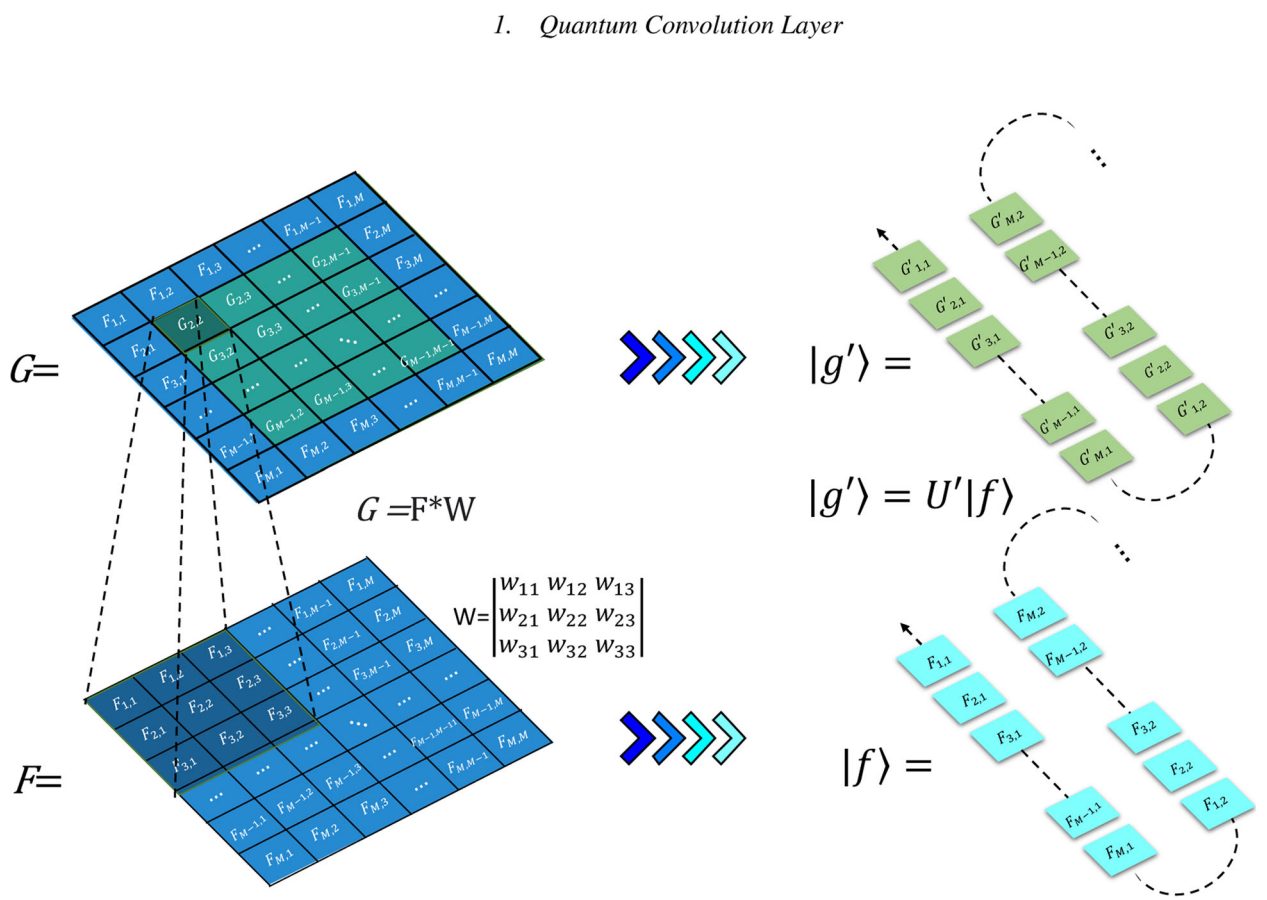

Fig. 1 Comparison of classical convolution processing and quantum convolution processing. F and G are the input and output image data, respectively. On the classical computer, a $M \times M$ image can be represented as a matrix and encoded with at least 2^n bits [$n = \lceil \log_2(M^2) \rceil$]. The classical image transformation through the convolution layer is performed by matrix computation $F * W$, which leads to $x_{ij}^{(\ell)} = \sum_{a,b=1}^{m} w_{a,b} x_{i+a-2,j+b-2}^{(\ell-1)}$. The same image can be represented as a quantum state and encoded in at least n qubits on a quantum computer. The quantum image transformation is realized by the unitary evolution U on a specific quantum state

$\lceil \log_2(ML) \rceil$ qubits, where the computational basis $|k\rangle$ encodes the position (i, j) of each pixel, and the coefficient c_k encodes the pixel value, i.e., $c_k = F_{i,j} / \left(\sum F_{i,j}^2 \right)^{1/2}$ for $k < ML$ and $c_k = 0$ for $k \geq ML$. Here, $\left(\sum F_{i,j}^2 \right)^{1/2}$ is a constant factor to normalizing the quantum state.

Without loss of generality, we focus on the input image with $M = L = 2^n$ pixels. The convolution layer transforms an input image $F = (F_{i,j})_{M\times M}$ into an output image $G = (G_{i,j})_{M\times M}$ by a specific filter mask W. In the quantum context, this linear transformation, corresponding to a specific spatial filter operation, can be represented as $|g\rangle = U|f\rangle$ with the input image state $|f\rangle$ and the output image state $|g\rangle$. For simplicity, we take a 3×3 filter mask as an example

$$W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix}. \tag{2}$$

The generalization to arbitrary $m \times m$ filter mask is straightforward. Convolution operation will transform the input image $F = (F_{i,j})_{M\times M}$ into the output image as $G = (G_{i,j})_{M\times M}$ with the pixel $G_{i,j} = \sum_{u,v=1}^{3} w_{uv} F_{i+u-2,j+v-2}$ $(2 \leq i, j \leq M-1)$. The corresponding quantum evolution $U|f\rangle$ can be performed as follows. We represent input image $F = (F_{i,j})_{M\times M}$ as an initial state

$$|f\rangle = \sum_{k=0}^{M^2-1} c_k |k\rangle, \tag{3}$$

where $c_k = F_{i,j} / \left(\sum F_{i,j}^2 \right)^{1/2}$. The $M^2 \times M^2$ linear filtering operator U can be defined as [25]:

$$U = \begin{bmatrix} E & & & & \\ V_1 & V_2 & V_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots \\ & & & V_1 & V_2 & V_3 \\ & & & & & E \end{bmatrix}, \tag{4}$$

where E is an M dimensional identity matrix, and V_1, V_2, V_3 are $M \times M$ matrices defined by

$$V_1 = \begin{pmatrix} 0 & & & \\ w_{11} & w_{21} & w_{31} & \\ & \ddots & \ddots & \ddots \\ & & w_{11} & w_{21} & w_{31} \\ & & & & 0 \end{pmatrix}_{M\times M}$$

$$V_2 = \begin{pmatrix} 1 & & & \\ w_{12} & w_{22} & w_{32} & \\ & \ddots & \ddots & \ddots \\ & & w_{12} & w_{22} & w_{32} \\ & & & & 1 \end{pmatrix}_{M\times M}$$

$$V_3 = \begin{pmatrix} 0 & & & \\ w_{13} & w_{23} & w_{33} & \\ & \ddots & \ddots & \ddots \\ & & w_{13} & w_{23} & w_{33} \\ & & & & 0 \end{pmatrix}_{M\times M}. \tag{5}$$

Generally speaking, the linear filtering operator U is non-unitary that can not be performed directly. Actually, we can embed U in a bigger system with an ancillary system and decompose it into a linear combination of four unitary operators [26]. $U = U_1 + U_2 + U_3 + U_4$, where $U_1 = (U + U^\dagger)/2 + i\sqrt{I - (U + U^\dagger)^2/4}$, $U_2 = (U + U^\dagger)/2 - i\sqrt{I - (U + U^\dagger)^2/4}$, $U_3 = (U - U^\dagger)/2i + i\sqrt{I + (U - U^\dagger)^2/4}$ and $U_4 = (U - U^\dagger)/2i - i\sqrt{I + (U - U^\dagger)^2/4}$. However, the basic gates consumed to perform U_i scale exponentially in the dimensions of quantum systems, making the quantum advantage diminishing. In [25], the efficient decomposition or the gate complexity of U is an open question. However, the gate complexity is the fundamental standard for measuring algorithm efficiency. Therefore, we present a new approach to construct the filter operator to reduce the gate complexity. For convenience, we change the elements of the first row, the last row, the first column, and the last column in the matrix V_1, V_2, and V_3, which is allowable in imagining processing, to the following form

$$V_1' = \begin{pmatrix} w_{21} & w_{31} & & w_{11} \\ w_{11} & w_{21} & w_{31} & \\ & \ddots & \ddots & \ddots \\ & & w_{11} & w_{21} & w_{31} \\ w_{31} & & & w_{11} & w_{21} \end{pmatrix}_{M\times M}$$

$$V_2' = \begin{pmatrix} w_{22} & w_{32} & & w_{12} \\ w_{12} & w_{22} & w_{32} & \\ & \ddots & \ddots & \ddots \\ & & w_{12} & w_{22} & w_{32} \\ w_{32} & & & w_{12} & w_{22} \end{pmatrix}_{M\times M}$$

$$V_3' = \begin{pmatrix} w_{23} & w_{33} & & w_{13} \\ w_{13} & w_{23} & w_{33} & \\ & \ddots & \ddots & \ddots \\ & & w_{13} & w_{23} & w_{33} \\ w_{33} & & & w_{13} & w_{23} \end{pmatrix}_{M\times M}. \tag{6}$$

Defining the adjusted linear filtering operator U' as

$$U' = \begin{bmatrix} V'_2 & V'_3 & & & & & V'_1 \\ V'_1 & V'_2 & V'_3 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & V'_1 & V'_2 & V'_3 & \\ V'_3 & & & & V'_1 & V'_2 & \end{bmatrix}, \tag{7}$$

Next, we decompose $V'_\mu(\mu = 1, 2, 3)$ into three unitary matrices without normalization, $V'_\mu = V'_{1\mu} + V'_{2\mu} + V'_{3\mu}$, where

$$\begin{aligned} V'_{1\mu} &= \begin{pmatrix} & & & & & & w_{1\mu} \\ w_{1\mu} & & & & & & \\ \ddots & \ddots & \ddots & & & & \\ & & w_{1\mu} & & & & \\ & & & w_{1\mu} & & & \end{pmatrix}_{M\times M} \\ V'_{2\mu} &= \begin{pmatrix} w_{2\mu} & & & & & & \\ & w_{2\mu} & & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & & w_{2\mu} & & & \\ & & & & w_{2\mu} & & \end{pmatrix}_{M\times M} \\ V'_{3\mu} &= \begin{pmatrix} & w_{3\mu} & & & & & \\ & & w_{3\mu} & & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & & & w_{3\mu} & \\ w_{3\mu} & & & & & & \end{pmatrix}_{M\times M} . \end{aligned} \tag{8}$$

Thus, the linear filtering operator U' can be expressed as

$$U' = \sum_{\mu=1}^{3}\sum_{\nu=1}^{3} \left(V'_{\mu\mu}/w_{\mu\mu}\right) \otimes V'_{\nu\mu}. \tag{9}$$

which can be simplified to

$$U' = \sum_{k=1}^{9} \beta_k Q_k, \tag{10}$$

where $Q_k = \left(V'_{\mu\mu}/w_{\mu\mu}\right) \otimes V'_{\nu\mu}/w_{\nu\mu}$ is unitary, and β_k is a relabelling of the indices.

Now, we can perform U' through the linear combination of unitary operators Q_k. The number of unitary operators is equal to the size of filter mask. The quantum circuit to realize U' is shown in Fig. 2. The work register $|f\rangle$ and four ancillary qubits $|0000\rangle_a$ are entangled together to form a bigger system.

Firstly, we prepare the initial state $|f\rangle$ using amplitude encoding method or quantum random access memory (qRAM). Then, performing unitary matrix S on the ancillary registers to transform $|0000\rangle_a$ into a specific superposition state $|\psi\rangle_a$

$$S|0000\rangle_a = |\psi\rangle_a = \sum_{\nu=1}^{9} \beta_k/N|k\rangle \tag{11}$$

where $N_c = \sqrt{\sum_{k=1}^{9} \beta_k^2}$ and S satisfies

$$S_{k,1} = \begin{cases} \beta_k/N_c & \text{if } k \le 9 \\ 0 & \text{if } k > 9. \end{cases} \tag{12}$$

S is a parameter matrix corresponding to a specific filter mask that realizes a specific task.

Then, we implement a series of ancillary system controlled operations $Q_k \otimes |k\rangle\langle k|$ on the work system $|f\rangle$ to realize LCU. Nextly, Hadamard gates $H^T = H^{\otimes 4}$ are acted to uncompute the ancillary registers $|\psi\rangle_a$. The state is transformed to

$$|g'\rangle = \sum_{i=1}^{16} \frac{1}{N_c}|i\rangle \sum_{k=1}^{9} H^T_{(ik)} S_{(k1)} Q_k |f\rangle, \tag{13}$$

where $H^T_{(ik)}$ is the ith row and kth column in matrix H^T and $S_{(k1)}$ is kth row and the first column in matrix S. The first term equals to

$$\frac{1}{N_c}|0\rangle \sum_{k=1}^{9} \beta_k Q_k |f\rangle, \tag{14}$$

which corresponds to the filter mask W. The ith term equals to filter mask $W^i (i = 2, 3, \ldots, 16)$, where

$$W^i = \begin{bmatrix} H^T_{i1} w_{11} & H^T_{i4} w_{12} & H^T_{i7} w_{13} \\ H^T_{i2} w_{21} & H^T_{i5} w_{22} & H^T_{i8} w_{23} \\ H^T_{i3} w_{31} & H^T_{i6} w_{32} & H^T_{i9} w_{33} \end{bmatrix}. \tag{15}$$

Totally, 16 filter masks are realized, corresponding to ancilla qubits in 16 different state $|i\rangle (i = 1, 2, \ldots, 16)$. Therefore, the whole effect of evolution on state $|f\rangle$ without considering the ancilla qubits, is the linear combination of the effects of 16 filter masks.

If we only need one filter mask W, measuring the ancillary register and conditioned on seeing $|0000\rangle$. We have the state $\frac{1}{N_c}|0000\rangle U'|f\rangle$, which is proportional to our expected result state $|g\rangle$. The probability of detecting the ancillary state $|0000\rangle$ is $P_s = \| \sum_{k=1}^{9} \beta_k Q_k |f\rangle \|^2 / N_c^2$.

After obtaining the final result $\frac{1}{N_c} U'|f\rangle$, we can multiply the constant factor N_c to compute $|g'\rangle = U'|f\rangle$. In conclusion, the filter operator U' can be decomposed into a linear combination of nine unitary operators in the case that the general filter mask is W. Only four qubits or a nine energy level ancillary system is consumed to realize the general filter operator U', which is independent on the dimensions of image size.

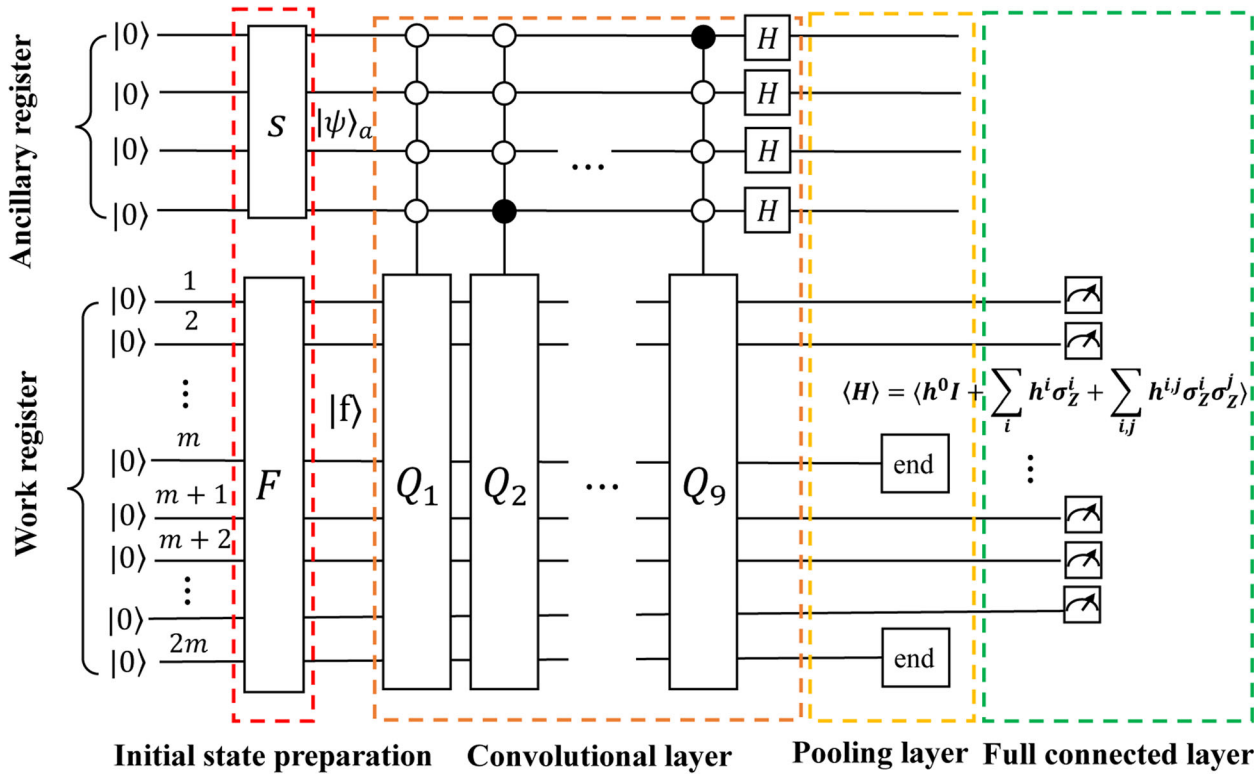

Fig. 2 Quantum circuit for realizing the QCNN. $|f\rangle$ denotes the initial state of work system after encoded the image data, and the ancillary system is a four qubits system in the $|0000\rangle_a$ state. The squares represent unitary operations and the circles represent the state of the controlling system. Unitary operations $Q_1, Q_2, \cdots, Q_9$, are activated only when the auxiliary system is in state $|0000\rangle, |0001\rangle \cdots, |1000\rangle$ respectively

The final stage of our method is to extract useful information from the processed results $|g'\rangle$. Clearly, the image state $|g\rangle$ is different from $|g'\rangle$. However, not all elements in $|f\rangle$ are evaluated, the elements corresponding to the four edges of original image remain unchanged. One is only interested in the pixel values which are evaluated by W in $|f\rangle$. These pixel values in $|g'\rangle$ are as same as that in $|g\rangle$ (see details in Appendix C). So, we can obtain the information of $G = (G_{i,j})_{M\times M}$ $(2 \le i,j \le M-1)$ by evaluating the $|f\rangle$ under operator U' instead of U.

2.1.2 Quantum pooling layer

The function of pooling layer after the convolutional layer is to reduce the spatial size of the representation so as to reduce the amount of parameters. We adopt average pooling which calculates the average value for each patch on the feature map as pooling layers in our model. Consider a $2*2$ pixel pooling operation applied with a stride of 2 pixels. It can be directly realized by ignoring the last qubit and the mth qubit in quantum context. The input image $|g'\rangle = (g_1, g_2, g_3, g_4, \ldots, \ldots, g_{M^2})^T$ after this operation can be expressed as the output image

$$|p\rangle = \left(\sqrt{g_1^2+g_2^2+g_{M+1}^2+g_{M+2}^2}, \sqrt{g_3^2+g_4^2+g_{M+3}^2+g_{M+4}^2}, \ldots, \sqrt{g_{M^2-M-1}^2+g_{M^2-M}^2+g_{M^2-1}^2+g_M^2}\right)^T. \tag{16}$$

2.1.3 Quantum fully connected layer

Fully connected layers compile the data extracted by previous layers to form the final output; it usually appears at the end of the convolutional neural networks. We define a parametrized Hamiltonian up to a seconder order correlation as the quantum fully connected layer. This Hamiltonian consists of identity operators I and Pauli operators σ_z,

$$\mathcal{H} = h^0 I + \sum_i h^i \sigma_z^i + \sum_{i,j} h^{ij}\sigma_z^i\sigma_z^j \tag{17}$$

where h^0, h^i, h^{ij} are the parameters, and Roman indices i,j denote the qubit on which the operator acts, i.e., σ_z^i means Pauli matrix σ_z acting on a qubit at site i. We measure the expectation value of the parametrized Hamiltonian $f(p) = \langle p|\mathcal{H}|p\rangle$. As shown in [27], the local cost function $f(p)$ is

more trainable than global cost function. $f(p)$ is the final output of the whole quantum neural network. Then, we add an active function to nonlinearly map $f(p)$ to $R(f(p))$.

The parameters in Hamiltonian matrix $\mathcal{H}$ are updated by gradient descent method, i.e., are calculated by $\frac{\partial f(p)}{\partial h^i} = \langle p|\sigma_z^i|p\rangle$ and $\frac{\partial f(p)}{\partial h^{ij}} = \langle p|\sigma_z^i\sigma_z^j|p\rangle$. We rewrite the cost function as

$$\begin{aligned} f(p) &= Tr\left(\frac{1}{N_c^2}\sum_{i=1}^{16}|i\rangle\sum_{k=1}^{9}H_{(ik)}^T S_{(k1)}Q_k|f\rangle\langle f|\sum_{i'=1}^{16}\langle i'|\sum_{k'=1}^{9}H_{(i'k')}^T S_{(k'1)}Q_{k'}^\dagger\mathcal{H}\right) \\ &= Tr\left(\frac{1}{N_c^2}\sum_{i=1}^{16}\sum_{k=1}^{9}H_{(ik)}^T S_{(k1)}Q_k\rho_i\sum_{k'=1}^{9}H_{(ik')}^T S_{(k'1)}Q_{k'}^\dagger\mathcal{H}\right), \end{aligned} \tag{18}$$

here $\rho_i = |f\rangle|i\rangle\langle i|\langle f|$. From Eq.(18), the cost function partial derivative with respect to w_k is

$$\frac{\partial f(p)}{\partial w_k} = \frac{1}{N_c^2}Tr\left(\sum_{i=1}^{16}\sum_{k'=1}^{9}H_{(ik)}^T H_{(ik')}^T S_{(k'1)}\left(Q_{k'}^\dagger\mathcal{H}Q_k + Q_k^\dagger\mathcal{H}Q_{k'}\right)\rho_i\right).$$

Therefore, the parameters can be updated by measuring the expectation values of specific operators.

Now, we have constructed the framework of quantum neural networks. We demonstrate the performance of our method in image processing and handwritten number recognition in the next section.

2.2 Numerical simulations

2.2.1 *Image processing: edge detection, image smoothing, and sharpening*

In addition to constructing QCNN, the quantum convolutional layer can also be used to spatial filtering which is a technique for image processing [25, 28–30], such as image smoothing, sharpening, edge detection, and edge enhancement. To show the quantum convolutional layer can handle various image processing tasks, we demonstrate three types of image processing, edge detection, image smoothing, and sharpening with fixed filter mask W_{de}, W_{sm} and W_{sh} respectively

$$\begin{aligned} W_{de} &= \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}, W_{sm} = \frac{1}{13}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 5 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \\ W_{sh} &= \frac{1}{16}\begin{pmatrix} -2 & -2 & -2 \\ -2 & 32 & -2 \\ -2 & -2 & -2 \end{pmatrix}. \end{aligned} \tag{19}$$

In a spatial image processing task, we only need one specific filter mask. Therefore, after performing the above quantum convolutional layer mentioned, we measure the ancillary register. If we obtain $|0\rangle$, our algorithm succeeds and the spatial filtering task is completed. The numerical simulation proves that the output images transformed by a classical and quantum convolutional layer are exactly the same, as shown in Fig. 3.

2.2.2 *Handwritten number recognition*

Here, we demonstrate a type of image recognition task on a real-world dataset, called MNIST, a handwritten character dataset. In this case, we simulate a complete quantum convolutional neural network model, including a convolutional layer, a pooling layer, and a full-connected layer, as shown in Fig. 2. We consider the two-class image recognition task(recognizing handwritten characters ′1′ and ′8′) and ten-class image recognition task(recognizing

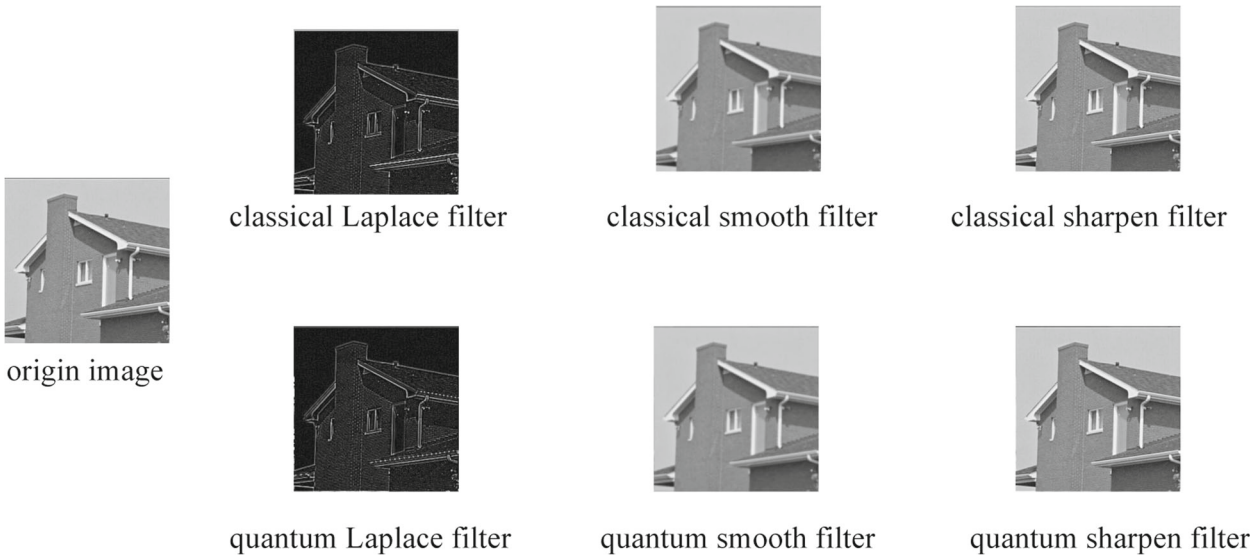

Fig. 3 Three types of image processing, edge detection, image smoothing, and sharpening, are implemented on an image by classical method and quantum method respectively

handwritten characters ′0′-′9′). Meanwhile, considering the noise on NISQ quantum system, we respectively simulate two circumstances that are the quantum gate Q_k is a perfect gate or a gate with certain noise. The noise is simulated by randomly acting a single qubit Pauli gate in $[I, X, Y, Z]$ with a probability of 0.01 on the quantum circuit after an operation implemented. In detail, the handwritten character image of MNIST has 28×28 pixels. For convenience, we expand 0 at the edge of the initial image until 32×32 pixels. Thus, the work register of QCNN consists of 10 qubits, and the ancillary register needs 4 qubits. The convolutional layer is characterized by 9 learnable parameters in matrix W that is the same for QCNN and CNN. In QCNN, by abandoning the 4-th and 9-th qubit of the work register, we perform the pooling layer on quantum circuit. In CNN, we perform average pooling layer directly. Through measuring the expected values of different Hamiltonians on the remaining work qubits, we can obtain the measurement values. After putting them in an activation function, we get the final classification result. In CNN, we perform a two-layer fully connected neural network and an activation function. In the two-classification problem, the QCNN's parametrized Hamiltonian has 37 learnable parameters, and the CNN's fully-connected layer has 256 learnable parameters. The classification result that is close to 0 are classified as handwritten character ′1′, and the result that is close to 1 are classified as handwritten character ′8′. In the ten-classification problem, the parametrized Hamiltonian has 10×37 learnable parameters and the CNN's fully-connected layer has 10×256 learnable parameters. The result is a 10-dimension vector. The classification results are classified as the index of the max element of the vector. Details of parameters, accuracy, and gate complexity are listed in Table 1.

For the 2 class classification problem, the training set and test set have a total of 5000 images and 2100 images, respectively. For the 10 class classification problem, the training set and test set have a total of 60000 images and 10000 images, respectively. Because in a training process, 100 images are randomly chosen in one epoch, and 50 epochs in total, the accuracy of the training set and the test set will fluctuate. So, we repeatedly execute noisy QCNN, noise-free, and CNN 100 times, under the same construction. In this way, we obtain the average accuracy and the field of accuracy, as shown in Fig. 4. We can conclude that from the numerical simulation result, QCNN and CNN provide similar performance. QCNN involves fewer parameters and has a smaller fluctuation range.

3 Algorithm complexity and trainability analysis

We analyze the computing resources in gate complexity and qubit consumption. (1) Gate complexity. At the convolutional layer stage, we could prepare an initial state in $O(poly(log_2(M^2)))$ steps. In the case of preparing a particular input $|f\rangle$, we employ the amplitude encoding method in [31–33]. It was shown that if the amplitude c_k and $P_k = \sum_k |c_k|^2$ can be efficiently calculated by a classical algorithm, constructing the $log_2\left(M^2\right)$-qubit X state takes $O\left(poly\left(log_2\left(M^2\right)\right)\right)$ steps. Alternatively, we can resort to quantum random access memory [34–36]. Quantum random access memory (qRAM) is an efficient method to do state preparation, whose complexity is $O\left(log_2\left(M^2\right)\right)$ after the quantum memory cell established. Moreover, the controlled operations Q_k can be decomposed into $O\left(\left(log_2 M\right)^6\right)$ basic gates (see details in Appendix A). In summary, our algorithm uses $O\left(\left(log_2 M\right)^6\right)$ basic steps to realize the filter progress in the convolutional layer. For CNN, the complexity of

Table 1 The important parameters of models

Models	Problems	Data set	Parameters		
			Learnable parameters	Average accuracy	Gate complexity
Noisy QCNN	′1′ or ′8′	Training set	46	0.948	$O((log_2 M)^6)$
		Test set		0.960	
	′0′ ∼′ 9′	Training set	379	0.742	
		Test set		0.740	
Noise-free QCNN	′1′ or ′8′	Training set	46	0.954	
		Test set		0.963	
	′0′ ∼′ 9′	Training set	379	0.756	
		Test set		0.743	
CNN	′1′ or ′8′	Training set	265	0.962	$O(M^2)$
		Test set		0.972	
	′0′ ∼′ 9′	Training set	2569	0.802	
		Test set		0.804	

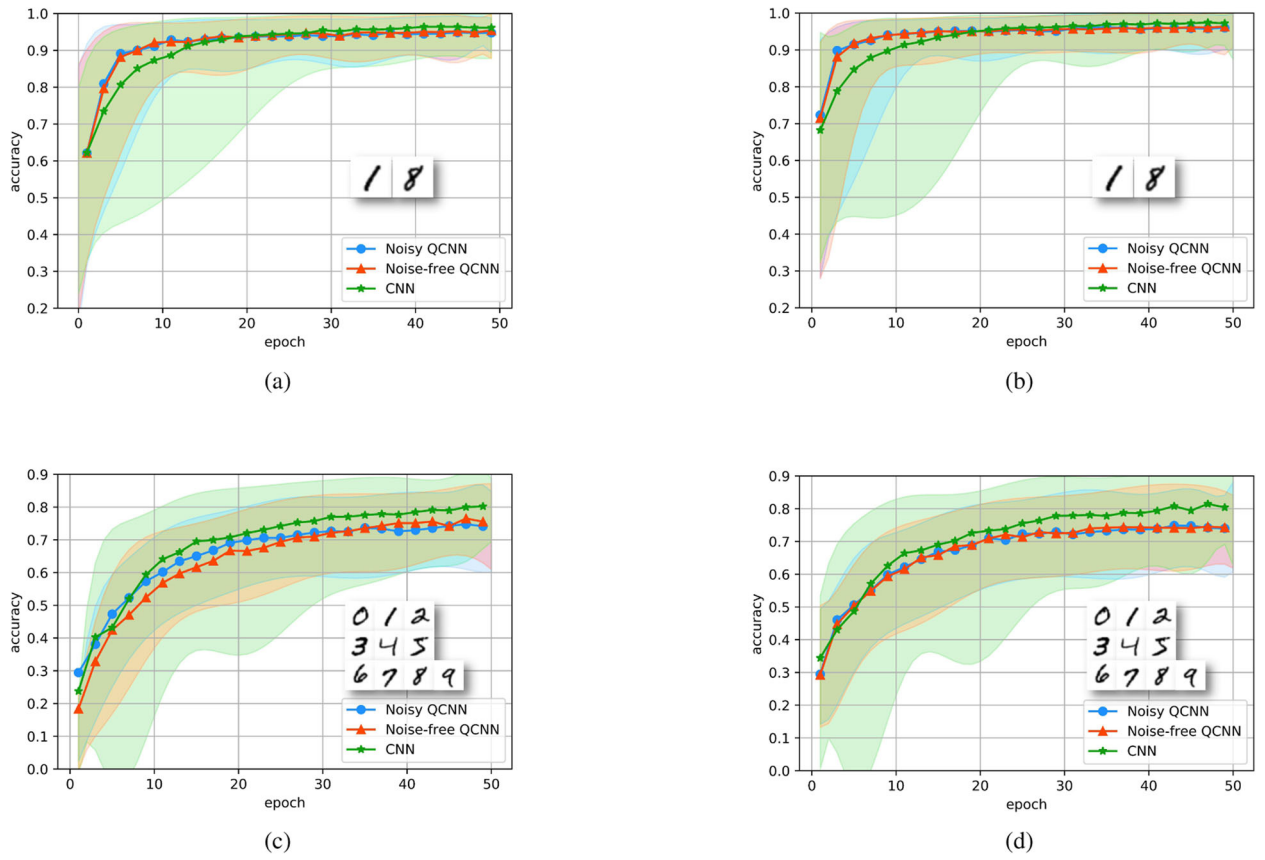

Fig. 4 The performance of QCNN based on MNIST. The blue, red, and green curves denote the average accuracy of the noisy QCNN, noise-free QCNN, and CNN, respectively. The shadow areas of the corresponding color denote the accuracy fluctuation range in the 100 times simulation results. The insets are the typical images from MNIST set. **a**, **b** The curves representing the result from the training set and the test set for the 2 class classification problem respectively. **c**, **d** The result from the training set and the test set for the 10 class classification problem respectively

implementing a classical convolutional layer is $O\left(M^2\right)$. Thus, this algorithm achieves an exponential speedup over classical algorithms in gate complexity. The measurement complexity in fully connected layers is $O(e)$, where e is the number of parameters in the Hamiltonian.

(2) Memory consumption. The ancillary qubits in the whole algorithm are $O\left(log_2\left(m^2\right)\right)$, where m is the dimension of the filter mask, and the work qubits are $O\left(log_2\left(M^2\right)\right)$. Thus, the total qubits resource needed is $O\left(log_2\left(m^2\right) + O\left(log_2\left(M^2\right)\right.\right.$.

According to [27, 37–39], we can analyze the trainability of the parameters in our QCNN model by studying the scaling of the variance

$$Var\left[\frac{\partial f(p)}{\partial w}\right] = \left\langle \left(\frac{\partial f(p)}{\partial w}\right)_S^2 \right\rangle - \left\langle \frac{\partial f(p)}{\partial w} \right\rangle_S^2, \tag{20}$$

where the expectation value $\langle \cdots \rangle$ is taken over the parameters in S [39, 40]. The cost will exhibit a barren plateau in the case the variance is exponentially small, and hence leads to the circuit untrainable. In contrast, large variances (polynomial small) indicate the absence of barren plateaus and that the trainability of the parameters can be guaranteed.

The variance in our model is (see details in Appendix C)

$$\begin{aligned} Var\left[\frac{\partial f(p)}{\partial w}\right] &= \left\langle \left(\frac{\partial f(p)}{\partial w}\right)_S^2 \right\rangle - \left\langle \frac{\partial f(p)}{\partial w} \right\rangle_S^2 \\ &= \frac{1}{N_c^4}\left(\frac{1}{17}\left(2\alpha_0^2 + \sum_i \alpha_i^2 + \sum_{ij} \alpha_{ij}^2\right) - \alpha_0^2\right) \end{aligned} \tag{21}$$

If $\frac{\left(2\alpha_0^2+\sum_i \alpha_i^2+\sum_{ij} \alpha_{ij}^2\right)-\alpha_0^2}{N_c^4} \in O(poly(log(n))$, then $Var\left[\frac{\partial f(p)}{\partial w}\right] \propto O(1/poly(log(n))$. This assumption is reasonable and easy to be satisfied, because parameters N_c^4 in a convolutional kernel which is usually a 3×3 or 5×5 matrix are independent on input image size . This implies that the cost function landscape does not present a barren plateau,and hence that this QCNN architecture is trainable under a convolutional kernel.

4 Discussion

In summary, we designed a quantum neural network which provides exponential speed-ups over their classical counterparts in gate complexity. With fewer parameters, our model achieves similar performance compared with classical algorithm in handwritten number recognition tasks. Therefore, this algorithm has significant advantages over the classical algorithms for large data. We present two interesting and practical applications, image processing and handwritten number recognition, to demonstrate the validity of our method. The mapping relations between a specific classical convolutional kernel to a quantum circuit is given that provides a bridge between QCNN to CNN. We analyze the trainability and the existence of barren plateaus in our QCNN model. It is a general algorithm and can be implemented on any programmable quantum computer, such as superconducting, trapped ions, and photonic quantum computer. In the big data era, this algorithm has great potential to outperform its classical counterpart, and works as an efficient solution.

Appendix A: Adjusted operator U' can provide enough information to remap the output imagine

Proof.- The different elements of image matrix after implementing operator U' compared with U are in the edges of image matrix. We prove that the evolution under operator U' can provide enough information to remap the output image. The different elements between U' and U are included in

$$U'_{k,n} \neq U_{k,n} \begin{cases} (1 \le k \le M; 1 \le n \le 2M, M^2 - M \le n \le M^2) \\ (M^2 - M \le k \le M^2; 1 \le n \le M, M^2 - 3M \le n \le M^2 - M) \\ (k = sM + 1; n = 1 + (s-1)M, 2 + (s-1)M, sM, sM \\ +1, sM + 2, (s+1)M, (s+1)M+1, (s+1)M+2, (s+2)M) \\ (k = (s+1)M; n = 1, sM - 1, sM, sM + 1, (s+1)M - 2, \\ (s+1)M - 1, (s+1)M + 1, (s+2)M - 2, (s+2)M - 1) \end{cases} \tag{22}$$

where $1 \le s \le M - 2$.

After performing U' and U on quantum state $|f\rangle$ respectively, the difference exits in the elements $|g'_k\rangle \neq |g_k\rangle (k = 1, 2, \cdots, M, sM+1, (s+1)M, M^2 - M + 1, \cdots, M^2)$, where $1 \le s \le M - 2$. Since $|g'\rangle$ can be remapped to G', U' will give the output image $G' = (G'_{i,j})_{M \times M}$. The elements in U' which is different from U only affect the pixel $i, j \notin 2, \cdots, M - 1$. Thus, only and if only $i, j \notin 2, \cdots, M - 1$, the matrix elements satisfy $G_{i,j} \neq G'_{i,j}$. Namely, the output imagine $G'_{i,j} = G_{i,j} (2 \le i, j \le M - 1)$.

Appendix B: Decomposing operator Q into basic gates

Considering the nine operators $Q_1, Q_2, \cdots, Q_9$ consist of filter operator U'. Q_k is the tensor product of two of the following three operators

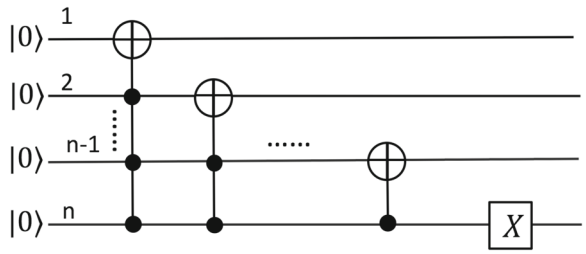

Fig. 5 Decomposition of operator E_1 in the form of basic gates

$$E_1 = \begin{pmatrix} & & & & 1 \\ 1 & & & & \\ \ddots & \ddots & \ddots & & \\ & & 1 & & \\ & & & 1 & \end{pmatrix}_{M \times M} \quad E_2 = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & \ddots & \ddots & \ddots & \\ & & & 1 & \\ & & & & 1 \end{pmatrix}_{M \times M}$$

$$E_3 = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & \ddots & \ddots & \ddots & \\ & & & & 1 \\ 1 & & & & \end{pmatrix}_{M \times M} . \tag{23}$$

E_2 is a $M \times M$ identity matrix not need to be further decomposed. For convenient, consider a n-qubits operator E_1 with dimension $M \times M$, where $n = log_2(M^2)$. It can be expressed by the combination of $O(n^3)$ CNOT gates and Pauli X gates as shown in Fig. 5. Consequently, E_3 can be decomposed into the inverse of combinations of basic gate as shown in Fig. 5, because of the fact $E_3 = E_1^\dagger$. Thus, Q_k can be implemented by no more than $O(n^6)$ basic gates. Totally, the controlled Q_k operation can be implemented by no more than $O(n^6) = O((log_2 M)^6)$(ignoring constant number).

Appendix C: Trainable analysis of the QCNN model

Firstly, we recall the definition of a t-design. Consider a finite set $S = \{S_y\}_{y \in Y}$ contains $|Y|$ number d-dimensional unitaries S_y. And $P_{t,t}(S)$ is a polynomial function with degree at most t in the matrix elements of S and at most of degree t in those of $S^\dagger$. Then, we say that this finite set is a t-design if

$$\frac{1}{|Y|} \sum_{y \in Y} P_{t,t}(S_y) = \int d\mu(S) P_{t,t}(S) , \tag{24}$$

where the integral is over $U(d)$ with respect to the Haar distribution. In our QCNN model, S forms a 2-design and for any function $F(S)$, and for any unitary matrix A

$$\int F(AS) d\mu(S) = \int F(S) d\mu(S). \tag{25}$$

The average of the partial derivative of the cost function is

$$\left\langle \frac{\partial f(p)}{\partial w_k} \right\rangle_S = \frac{1}{N_c^2} Tr\left(\sum_{k'=1}^{9} H^T_{(ik)} H^T_{(ik')} S_{(k'1)} \left(Q^\dagger_{k'} \mathcal{H} Q_k + Q^\dagger_k \mathcal{H} Q_{k'} \right) \right) Tr(|f\rangle\langle f|)$$

and $Tr(|f\rangle\langle f|) = 1$. Consider the fact that $\mathcal{H}$ maintains the property that being constructed by Pauli product matrices under the transformation of Q_k,i.e., $\sum_{k'=1}^{9} H^T_{(ik)} H^T_{(ik')} S_{(k'1)} (Q^\dagger_{k'} \mathcal{H} Q_k + Q^\dagger_k \mathcal{H} Q_k) = \mathcal{H}^{new}$, where $\mathcal{H}^{new} = \alpha_0 I + \sum_i \alpha_i \sigma_z^i + \sum_{i,j} (\alpha_{ij}) \sigma_z^i \sigma_z^j$. Then, we have $Tr(\mathcal{H}^{new}) = \alpha_0$, and

$$\left\langle \frac{\partial f(p)}{\partial w_k} \right\rangle = \frac{\alpha_0}{N_c^4}.$$

The expectation value of the squares of gradients is

$$\begin{aligned}
\left\langle \left(\frac{\partial f(p)}{\partial w} \right)^2 \right\rangle_S &= \int d\mu(S) \frac{1}{N_c^4} Tr\left(\sum_{i=1}^{16} \sum_{k'=1}^{9} H^T_{(ik)} H^T_{(ik')} S_{(k'1)} \left(Q^\dagger_{k'} \mathcal{H} Q_k \right.\right. \\
&\quad \left.\left. + Q^\dagger_k \mathcal{H} Q_{k'} \right) \rho_i \right)^2 \\
&= \frac{1}{N_c^4} \frac{1}{16^2 - 1} \left(Tr(\mathcal{H}^{new}) Tr(|f\rangle\langle f|) Tr(\mathcal{H}^{new}) Tr(|f\rangle\langle f|) \right. \\
&\quad \left. + Tr((\mathcal{H}^{new})^2) Tr(|f\rangle\langle f|) \right) \\
&\quad - \frac{1}{N_c^4} \frac{1}{16(16^2 - 1)} \left(Tr((\mathcal{H}^{new})^2) Tr(|f\rangle\langle f|) Tr(|f\rangle\langle f|) \right. \\
&\quad \left. + Tr(\mathcal{H}^{new}) Tr(\mathcal{H}^{new}) Tr(|f\rangle\langle f|) \right) \\
&= \frac{1}{N_c^4} \left(\frac{1}{16^2 - 1} (\alpha_0^2 + Tr((\mathcal{H}^{new})^2)) - \frac{1}{16(16^2 - 1)} \right. \\
&\quad \left. (\alpha_0^2 + Tr((\mathcal{H}^{new})^2)) \right) \\
&= \frac{1}{N_c^4} \left(\frac{1}{17} (2\alpha_0^2 + \sum_i \alpha_i^2 + \sum_{ij} \alpha_{ij}^2) \right)
\end{aligned}$$

Therefore, the variance is

$$\begin{aligned}
Var[\frac{\partial f(p)}{\partial w}] &= \left\langle \left(\frac{\partial f(p)}{\partial w} \right)^2 \right\rangle_S - \left\langle \frac{\partial f(p)}{\partial w} \right\rangle_S^2 \qquad (26) \\
&= \frac{1}{N_c^4} \left(\frac{1}{17} (2\alpha_0^2 + \sum_i \alpha_i^2 + \sum_{ij} \alpha_{ij}^2) - \alpha_0^2 \right)
\end{aligned}$$

Acknowledgements
We thank X. Yao and X. Peng for inspiration and fruitful discussions.

Authors' contributions
S.W. formulated the theory. Y.C. and Z.Z. performed the calculation. All work was carried out under the supervision of G.L. All authors contributed to writing the manuscript. The authors read and approved the final manuscript.

Funding
This research was supported by National Basic Research Program of China. S.W. acknowledge the China Postdoctoral Science Foundation 2020M670172 and the National Natural Science Foundation of China under Grants No. 12005015. We gratefully acknowledge support from the National Natural Science Foundation of China under Grants No. 11974205 and No. 11774197, The National Key Research and Development Program of China (2017YFA0303700), The Key Research and Development Program of Guangdong province (2018B030325002), and Beijing Advanced Innovation Center for Future Chip (ICFC).

Availability of data and materials
The code used to generate the quantum circuit and implement the experiment is available on reasonable request.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Beijing Academy of Quantum Information Sciences, Beijing 100193, China. [2]State Key Laboratory of Low-Dimensional Quantum Physics and Department of Physics, Tsinghua University, Beijing 100084, China. [3]Institute of Information Photonics and Optical Communications, Beijing University of Posts and Telecommunications, Beijing 100876, China. [4]Beijing National Research Center for Information Science and Technology and School of Information Tsinghua University, Beijing 100084, China. [5]Frontier Science Center for Quantum Information, Beijing 100084, China.

Received: 15 October 2021 Accepted: 6 December 2021
Published online: 10 January 2022

References

1. P. Benioff, The computer as a physical system: a microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. J. Stat. Phys. **22**(5), 563–591 (1980)
2. R. P. Feynman, Simulating physics with computers. Int. J. Theor. Phys. **21**(6), 467–488 (1982)
3. P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Rev. **41**(2), 303–332 (1999)
4. L. K. Grover, Quantum mechanics helps in searching for a needle in a haystack. Phys. Rev. Lett. **79**(2), 325 (1997)
5. G. L. Long, Grover algorithm with zero theoretical failure rate. Phys. Rev. A. **022307**, 64 (2001)
6. J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, S. Lloyd, Quantum machine learning. Nature. **549**(7671), 195–202 (2017)
7. V. Dunjko, J. M. Taylor, H. J. Briegel, Quantum-enhanced machine learning. Phys. Rev. Lett. **117**(13), 130501 (2016)
8. N. Killoran, T. R. Bromley, J. M. Arrazola, M. Schuld, N. Quesada, S. Lloyd, Continuous-variable quantum neural networks. Phys. Rev. Res. **1**(3), 033063 (2019)
9. J. Liu, K. H. Lim, K. L. Wood, W. Huang, C. Guo, H.-L. Huang, Hybrid quantum-classical convolutional neural networks. arXiv preprint arXiv:1911.02998 (2019)
10. F. Hu, B.-N. Wang, N. Wang, C. Wang, Quantum machine learning with d-wave quantum computer. Quantum Eng. **1**(2), e12 (2019)
11. E. Farhi, H. Neven, Classification with quantum neural networks on near term processors. Quantum Rev. Lett. **1**(2), 10–37686 (2020)
12. W. Huggins, P. Patil, B. Mitchell, K. B. Whaley, E. M. Stoudenmire, Towards quantum machine learning with tensor networks. Quantum Sci. Technol. **4**(2), 024001 (2019)
13. X. Yuan, J. Sun, J. Liu, Q. Zhao, Y. Zhou, Quantum simulation with hybrid tensor networks. Phys. Rev. Lett. **127**(4), 040501 (2021)
14. Y. Zhang, Q. Ni, Recent advances in quantum machine learning. Quantum Eng. **2**(1), e34 (2020)
15. J.-G. Liu, L. Mao, P. Zhang, L. Wang, Solving quantum statistical mechanics with variational autoregressive networks and quantum circuits. Mach. Learn. Sci. Technol. **2**(2), 025011 (2021)
16. E. Farhi, H. Neven, Classification with quantum neural networks on near term processors. arXiv preprint arXiv:1802.06002 (2018)

17. I. Cong, S. Choi, M. D. Lukin, Quantum convolutional neural networks. Nat. Phys. **15**(12), 1273–1278 (2019)
18. B. C. Britt, Modeling viral diffusion using quantum computational network simulation. Quantum Eng. **2**(1), e29 (2020)
19. M. Schuld, N. Killoran, Quantum machine learning in feature hilbert spaces. Phys. Rev. Lett. **122**(4), 040504 (2019)
20. Y. Li, R.-G. Zhou, R. Xu, J. Luo, W. Hu, A quantum deep convolutional neural network for image recognition. Quantum Sci. Technol. **5**(4), 044003 (2020)
21. I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, *Deep learning, volume 1*. (MIT press, Cambridge, 2016)
22. L. Gui-Lu, General quantum interference principle and duality computer. Commun. Theor. Phys. **45**(5), 825 (2006)
23. S. Gudder, Mathematical theory of duality quantum computers. Quantum Inf. Process. **6**(1), 37–48 (2007)
24. S.-J. Wei, G.-L. Long, Duality quantum computer and the efficient quantum simulations. Quantum Inf. Process. **15**(3), 1189–1212 (2016)
25. X.-W. Yao, H. Wang, Z. Liao, M.-C. Chen, J. Pan, J. Li, K. Zhang, X. Lin, Z. Wang, Z. Luo, et al., Quantum image processing and its application to edge detection: theory and experiment. Phys. Rev. X. **7**(3), 031041 (2017)
26. T. Xin, S. Wei, J. Cui, J. Xiao, I. Arrazola, L. Lamata, X. Kong, D. Lu, E. Solano, G. Long, Quantum algorithm for solving linear differential equations: theory and experiment. Phys. Rev. A. **101**(3), 032307 (2020)
27. M. Cerezo, A. Sone, T. Volkoff, L. Cincio, P. J. Coles, Cost function dependent barren plateaus in shallow parametrized quantum circuits. Nat. Comput. **12**(1), 1–12 (2021)
28. F. Yan, A. M. Iliyasu, S. E. Venegas-Andraca, A survey of quantum image representations. Quantum Inf. Process. **15**(1), 1–35 (2016)
29. S. E. Venegas-Andraca, S. Bose, Storing, processing, and retrieving an image using quantum mechanics. Inf. Comput. (2003)
30. P. Q. Le, F. Dong, K. Hirota, A flexible representation of quantum images for polynomial preparation, image compression, and processing operations. Quantum Inf. Process. **10**(1), 63–84 (2011)
31. G.-L. Long, Y. Sun, Efficient scheme for initializing a quantum register with an arbitrary superposed state. Phys. Rev. A. **64**(1), 014303 (2001)
32. L. Grover, T. Rudolph, Creating superpositions that correspond to efficiently integrable probability distributions. arXiv preprint quant-ph/0208112 (2002)
33. A. N. Soklakov, R. Schack, Efficient state preparation for a register of quantum bits. Phys. Rev. A. **73**(1), 012307 (2006)
34. V. Giovannetti, S. Lloyd, L. Maccone, Quantum random access memory. Phys. Rev. Lett. **100**(16), 160501 (2008)
35. V. Giovannetti, S. Lloyd, L. Maccone, Architectures for a quantum random access memory. Phys. Rev. A. **78**(5), 052310 (2008)
36. S. Arunachalam, V. Gheorghiu, T. Jochym-O'Connor, M. Mosca, P. V. Srinivasan, On the robustness of bucket brigade quantum ram. New J. Phys. **17**(12), 123010 (2015)
37. J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, H. Neven, Barren plateaus in quantum neural network training landscapes. Nat. Commun. **9**(1), 1–6 (2018)
38. K. Sharma, M. Cerezo, L. Cincio, P. J. Coles, Trainability of dissipative perceptron-based quantum neural networks. arXiv preprint arXiv:2005.12458 (2020)
39. A. Pesah, M. Cerezo, S. Wang, T. Volkoff, A. T. Sornborger, P. J. Coles, Absence of barren plateaus in quantum convolutional neural networks. Phys. Rev. X. **11**(4), 041011 (2021)
40. B. Collins, P. Śniady, Integration with respect to the haar measure on unitary, orthogonal and symplectic group. Commun. Math. Phys. **264**(3), 773–795 (2006)

Publisher's Note