Available online at www.sciencedirect.com

ScienceDirect

Journal of the Franklin Institute 360 (2023) 7438–7456

www.elsevier.com/locate/jfranklin

Efficient quantum feature extraction for CNN-based learning

Tong Dou[a], Guofeng Zhang[b,c], Wei Cui[a,d,*]

[a] *School of Automation Science and Engineering, South China University of Technology, Guangzhou 510640, China*
[b] *Department of Applied Mathematics, The Hong Kong Polytechnic University, Hongkong, China*
[c] *The Hong Kong Polytechnic University Shenzhen Research Institute, Shenzhen, 518057, China*
[d] *Pazhou Laboratory, Guangzhou, 510330, China*

Received 17 February 2022; received in revised form 10 April 2023; accepted 2 June 2023
Available online 8 June 2023

Abstract

Hybrid quantum-classical algorithms provide a promising way to harness the power of current quantum devices. In this framework, parametrized quantum circuits (PQCs) which consist of layers of parametrized unitaries can be considered as a kind of quantum neural networks. Recent works have begun to explore the potential of PQCs as general function approximators. In this work, we propose a quantum-classical deep network structure to enhance model discriminability of convolutional neural networks (CNNs). In CNNs, the convolutional layer uses linear filters to scan the input data followed by a nonlinear operation. Instead, we build PQCs, which are more potent function approximators, with more complex structures to capture the features within the receptive field. The feature maps are obtained by sliding the PQCs over the input in a similar way as CNN. We also give a training algorithm for the proposed model. Through numerical simulation, the proposed hybrid models demonstrate reasonable classification performance on MNIST and Fashion-MNIST (4-classes). In addition, we compare the performance of models in different settings. The results demonstrate that the model with high-expressibility ansaetze achieves lower cost and higher accuracy, but exhibits a “saturation” phenomenon.

* Corresponding author.
E-mail address: aucuiwei@scut.edu.cn (W. Cui).

https://doi.org/10.1016/j.jfranklin.2023.06.003

1. Introduction

Quantum Computers use the principles of quantum mechanics for computing, which are considered more powerful than classical computers in some computing problems. Due to advances in quantum computing hardware [1], we are entering into the so-called noisy intermediate scale quantum (NISQ) era [2]. NISQ devices will be the only quantum devices that can be available in the near term, where we can only use a limited number of qubits with little error correction. Thus, developing useful computational algorithms using NISQ devices is an urgent task at present.

Quantum machine learning (QML) [3] is gaining attention in the hope of speeding up machine learning tasks with quantum computers. Combing quantum computing and machine learning, QML attempts to utilize the power of quantum computers to achieve computational speedups or better performance for some machine learning tasks. Many quantum machine learning algorithms, such as qSVM [4], qPCA [5] and quantum Boltzmann machine [6], have been developed. Recently, several NISQ algorithms, such as quantum transfer learning [7] and quantum kernel methods [8,9], have been proposed. On the other hand, parametrized quantum circuits (PQCs) [10] provide an another path for QML toward quantum advantages in NISQ era. Compared to conventional quantum algorithms, such as Shor's algorithm [11], Grover's algorithm [12], and HHL algorithm [13], PQCs-based algorithms are inherently robust to noise and could benefit from both the high dimensional Hilbert space of quantum systems and the classical optimization scheme. Several popular related algorithms have been proposed, including variational quantum eigensolvers (VQE) [14–16] and quantum approximate optimization algorithm (QAOA) [17,18,56].

In addition, quantum neural network models [19–21] which are based on PQCs have been proposed. Some of them utilized the thoughts of convolutional neural networks (CNNs), which is a popular machine learning model on the classical computer. Cong et al. [22] designed a quantum circuit model with a similar hierarchy to classical CNNs, which dealt with quantum data and could be used to recognize phases of quantum states and to devise a quantum error correction scheme. The convolutional and pooling layers were approximated by quantum gates with adjustable parameters. Henderson et al. [23] proposed a new quanvolutional filter, in which a random quantum circuit was deployed to transform input data locally. Quanvolutional filters were embedded into classical CNNs, forming a quantum-classical hybrid structure. In addition, more complex quantum convolutional neural network designs were presented in recent works [24–28], where delicate quantum circuits were employed to accomplish quantum inner product computations and approximate nonlinear mappings of activation functions.

In this work, we propose a type of hybrid quantum-classical neural network based on PQCs that in analogy with the CNN. As a popular scheme in the machine learning field, CNN replaces the fully connected layer with a convolutional layer. The convolutional layer only connects each neuron of the output to a small region of the input which is referred to as a feature map, thus greatly reducing the number of parameters. However, the filter in classical CNN model is a generalized linear model. It is difficult for linear filters to extract the concepts that are generally highly nonlinear functions of the data patch. In Network-in-Network (NiN) [29], the mlpconv layer is proposed, in which linear filters are replaced with a multilayer perceptron that is a general function approximator. As a "micro network", multilayer perceptron can improve the abstraction ability of the model. In the field of QML, PQCs are considered as the "quantum network" structures. Recent works [30–32] showed that there exist PQCs which are universal function approximators with a proper data encoding strategy. Combining these ideas, we replace the linear filter with a PQC. The resulting structure is called the quantum feature extraction (QFE) layer.

The key idea of our hybrid neural network is to implement the feature map in the convolutional layer with quantum parameterized circuits, and correspondingly, the output of this feature map is a correlational measurement on the output quantum state of the parameterized circuits. Different from Henderson et al. [23], which uses parameters fixed circuits, we can iteratively update the parameters of circuits to get better performance. We also give the training algorithm of this hybrid quantum-classical neural network which are similar to back-propagation algorithm, meaning that our model is trained as a whole. Hence it can be trained efficiently with NISQ devices. Therefore, our scheme could utilize all the features of classical CNN, and is able to utilize the power of current NISQ processors. Notice that our methods of data encoding and decoding are different from Henderson et al. [23]. The key contributions of our work are as follows.

1. We propose a novel deep network based on quantum-classical hybrid architecture. The new structure introduces QFE layers which use PQCs to convolve the input while retaining the advantages of classical modules. As PQCs are universal function approximators with more complex structures, this model is capable of extracting higher-level abstraction of the input.
2. Combining the chain rule and the parameter-shift rule, we provide a training algorithm for the proposed model so that it can be trained as a whole. The derivation of the equations that describe the forward and backward pass during the training process of QFE layers is provided.
3. Through numerical simulation, it is shown that in the case of a toy model, our method can achieve competitive performance on a real-world dataset. Furthermore, we demonstrated that proposed models can implement effective representation learning without a classical "classifier".

This paper is organized as follows. Section 2 is the preliminary, in which we first review the classical convolution neural network and the framework of PQCs. Then, the proposed quantum-classical hybrid model is described in detail in Section 3. In Section 4, we present numerical simulation where we apply the hybrid model to image recognition and analyze the result. Conclusion and discussion are given in Section 5.

2. Preliminaries

2.1. Supervised learning: construct a classifier

In a supervised classification task, we are given a training set T and a test set S on a label set C. Assuming there exists a map $m : T \cup S \to C$ unknown to the algorithm. Our task is to construct a model to infer an approximate map on the test set $\tilde{m} : S \to C$ only receiving the labels of the training set such that the difference between $m(s)$ and $\tilde{m}(s)$ is small on the test set $s \in S$. For such a learning task to be meaningful it is assumed that there is a correlation between the labels given for the training set and the true map. A popular approach to this problem is to use neural networks, where the training data goes through the linear layer and the activation function repeatedly to map the true label. After constructing a neural network model that is accurate enough, we can make a prediction on the test set.

2.2. Convolutional neural network

CNN dates back decades [33,34] which is a specialized kind of neural network for processing data that has a known grid-like topology. Deep CNNs have recently shown an explosive popularity partially due to its success in image classification [35]. It also has been successfully applied to other fields, such as objection detection [36], semantic segmentation [37] and pedestrian detection [38], even for natural language processing [39].

Though there are several architectures of CNN models such as LeNet [34], AlexNet [35], GoogLeNet [40], VGG [41], ResNet [42] and DenseNet [43], they use three main types of layers to build CNN models: convolutional layer, pooling layer and fully-connected layer. Broadly, a CNN model can be divided into two parts: feature extraction module and classifier module. The former includes convolutional layer and pooling layer while the classifier module consists of fully-connected layers.

Formally, a convolutional layer is expressed as an operation f_1:

$$f_1(\mathbf{X}) = g(\mathbf{W}_1 * \mathbf{X} + \mathbf{B}_1), \tag{1}$$

where $\mathbf{W}_1$ and $\mathbf{B}_1$ represent the filters weights and biases respectively, and '$*$' denotes the convolution operation. Concretely, $\mathbf{W}_1$ corresponds to n_c filters of support $f \times f \times c$, where c is the number of channels in the input $\mathbf{X}$, f is the spatial size of a filter. Intuitively, $\mathbf{W}_1$ applies n_c convolutions on the input, and each convolution has a kernel size $f \times f \times c$. The output is composed of n_c feature maps. $\mathbf{B}_1$ is an n_c-dimensional vector, each of whose element is associated with a filter. $g(\cdot)$ is an activation function which element-wisely applies a nonlinear transformation, such as ReLU or Tanh, on the filter responses.

A pooling layer is generally added after the convolutional layer. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Two commonly used pooling operations are max pooling and average pooling. The former calculates the maximum value for each patch of the feature map while the latter calculates the average value for each patch on the feature map. And the units in a fully-connected layer have full connections to all activations in the previous layer, as seen in regular neural networks. The activations are computed with a matrix multiplication followed by a bias offset, as shown in the operation f_2:

$$f_2(\mathbf{x}) = h(\mathbf{W}_2\mathbf{x} + \mathbf{B}_2), \tag{2}$$

where $\mathbf{x}$, $\mathbf{W}_2$, $\mathbf{B}_2$ are an input vector, a weight matrix and a bias vector, respectively. $h(\cdot)$ is an activation function like $g(\cdot)$ above mentioned.

2.3. General framework of PQCs

PQCs, also known as "variational circuits", are a kind of quantum circuits that have trainable parameters subject to iterative optimizations. Previous results show that such circuits are robust against quantum noise inherently and therefore suitable for NISQ devices. Recently, there have been many developments in PQCs-based algorithms which combine both quantum and classical computers to accomplish a particular task. In addition to VQE and QAOA, the PQCs framework has been extended for applications in generative modeling [44], quantum data compression [45], quantum circuit compiling [46] and so on.

Algorithms involving PQCs usually work in the following way. First, prepare an initial state $|\varphi_0\rangle$ by encoding the input into the quantum device. Second, we need to choose an appropriate ansatz, that is, designing the circuit structure of a PQC $U(\boldsymbol{\theta})$, and apply $U(\boldsymbol{\theta})$ to $|\varphi_0\rangle$, where $\boldsymbol{\theta}$ are parameters of the ansatz. After that measure the circuit repeatedly on a specific observable $\hat{O}$ to estimate the expectation value $\langle\hat{O}(\boldsymbol{\theta})\rangle$. Finally, based on the $\langle\hat{O}(\boldsymbol{\theta})\rangle$ which is fed into a classical optimizer, we minimize a cost function $L(\langle\hat{O}(\boldsymbol{\theta})\rangle)$ by iteratively updating $\boldsymbol{\theta}$.

Several factors are of central importance in PQCs-based algorithms. First, it is necessary to find a suitable $\hat{O}$ for the given problem. That is, we require the minimum of $L(\langle\hat{O}(\boldsymbol{\theta})\rangle)$ to correspond to the solution of the task. How easily $\hat{O}$ can be measured on a quantum computer and the locality of $\hat{O}$(i.e. the number of qubits it acts non-trivially on) are also relevant.

Another aspect that determines the success of a PQCs-based algorithm is the choice of ansatz for $U(\boldsymbol{\theta})$. While discrete parameterizations are possible, $\boldsymbol{\theta}$ usually are continuous parameters, such as gate rotation angles. In general, a PQC is of form

$$U(\boldsymbol{\theta}) = \prod_{j=1}^{N} U_j(\theta_j)W_j, \tag{3}$$

where $\boldsymbol{\theta} = (\theta_1, \dots, \theta_N)$ are tunable parameters, and $\{W_j\}_{j=1}^N$ is a set of unitaries without parameters while $U_j = e^{-i\theta_j V_j}$ is a rotation gate of angle θ_j generated by a Hermitian operator V_j such that $V_j^2 = I$. Common choices for V_j are Pauli operators X, Y and Z. The rotation angles $\{\theta_j\}$ are typically considered to be independent.

3. Convolutional network with QFE layers

In this section, we will first describe the quantum-classical hybrid model with QFE layers and give a training algorithm. Then the possible experimental setup for its realization is discussed.

3.1. QFE layer

To learn a distribution of data, classical CNNs use linear filters to extract latent features for further tasks. However, it is desirable to utilize universal function approximators for representation learning, as it can approximate more complex representations. To improve the extraction capability of filters, PQCs are used in this work, composing QFE layers. Firstly, a PQC, which is regarded as a quantum neural network model to some extent, is a universal function approximator with proper data embedding strategies. Secondly, because of the parameter-shift rule, we are able to get gradient of input by treating them as angles. Therefore, PQCs can integrate into the structure of CNNs and be trained using backpropagation. A QFE layer is a transformational layer using quantum circuits with learnable parameters. It is similar to a convolutional layer in classical CNNs. Combining pooling layers, fully-connected layers even with convolutional layers, we can construct a hybrid quantum-classical neural network.

Suppose a QFE layer learns a mapping F, in which each step conceptually consists of three parts, namely encoding, variational evolution, decoding.

1. *Encoding*. This operation encodes patches from the input data $\boldsymbol{x}$(or the output of previous layer) into a quantum circuit by applying $U_{en}(\boldsymbol{x})$ to the initialized states $|0\rangle^n$. Hereafter, we abbreviate $|0\rangle^n$ as $|0\rangle$ if there is no confusion. Note that the output state $|\boldsymbol{x}\rangle$ are characterized by input $\boldsymbol{x}$.

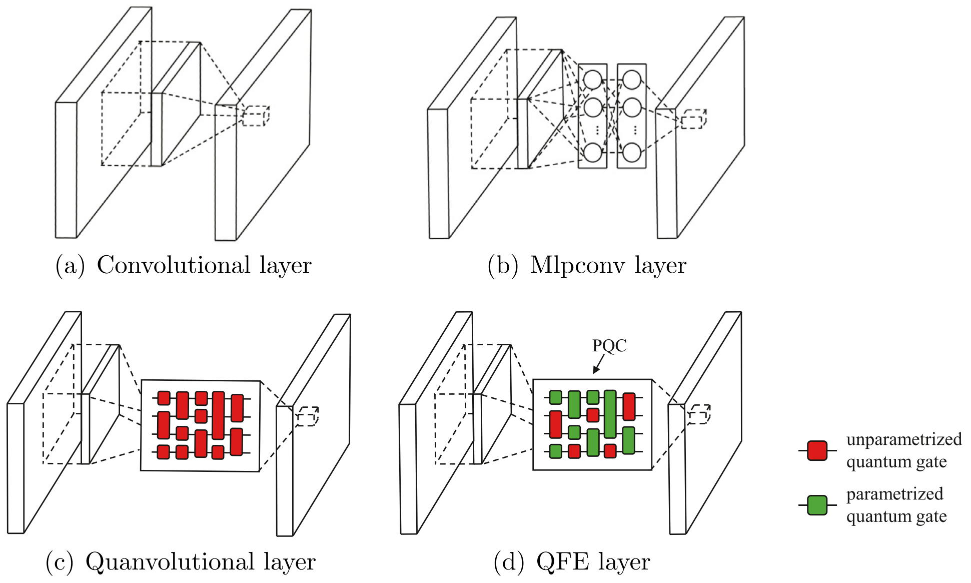


Fig. 1. Comparison of convolutional layer, mlpconv layer, quanvolutional layer and QFE layer. The convolutional layer includes a linear filter. The mlpconv layer includes a multilayer perceptron. The quanvolutional layer includes a random quantum circuit without trainable parameters. The QFE layer includes a "micro quantum network" (we choose the PQC in this paper).

2. *Variational evolution*. In this part, we apply a parameterized unitary circuit $U_{var}(\boldsymbol{\theta})$ to the quantum state $|\boldsymbol{x}\rangle$. $U_{var}(\boldsymbol{\theta})$ maps the quantum state $|\boldsymbol{x}\rangle$ onto another quantum state $|\boldsymbol{x};\boldsymbol{\theta}\rangle$ using a serial of quantum gates. This operation can introduce quantum properties such as superposition, entanglement and quantum parallelism.
3. *Decoding.*. This operation include two parts: measurement and classical post-processing. Firstly, at the end of the circuit, measurements are performed on an observable $\hat{O}$ to get an expectation value. Then after the classical post-processing, we get an output of one step in QFE layer. Similar to a classical convolution layer, each output is mapped to a different channel of a single pixel value.

The structure of a QFE layer is compared with convolutional, mlpconv and quanvolutional layer in Fig. 1. Next we detail our definition of each operation.

3.1.1. Encoding

There are two popular strategies to encode classical data into a quantum circuit, including the basis encoding and the amplitude encoding. The basis encoding method treats two basic states of a qubit, $|0\rangle$ and $|1\rangle$, as binary values of a classical bit, 0 and 1. While the amplitude encoding method uses the probability amplitudes of a quantum state to store classical data. As described below, to update the parameters effectively, we need to calculate the partial derivative of input. However, the two methods mentioned above are difficult to achieve it. To satisfy the condition, we treat a single input value as the angle of a parametrized rotation gate. Combining with parameter-shift rule [47], we can calculate the analytical gradients of

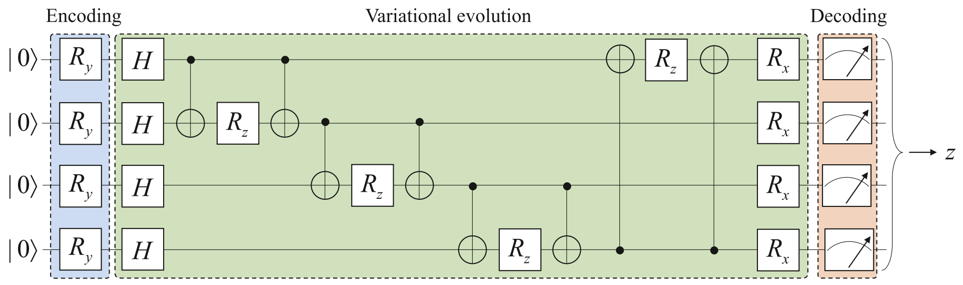


Fig. 2. Single step of a QFE layer with a QAOA-heuristic ansatz.

input on quantum circuits. Formally, the encoding part is expressed as an operation F_1:

$$F_1(\boldsymbol{x}) = U_{en}(\boldsymbol{x})|0\rangle = |\boldsymbol{x}\rangle. \tag{4}$$

Here $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^{\boldsymbol{T}}$ is the input patch of QFE layer where n is the dimension of $\boldsymbol{x}$(usually in a 3×3 or 5×5 size). $U_{en}(\boldsymbol{x})$ is a unitary to embed $\boldsymbol{x}$ into a quantum circuit. In our experiments, $U_{en}(\boldsymbol{x})$ is expressed as:

$$U_{en}(\boldsymbol{x}) = \bigotimes_{j=1}^{n} R_y(x_j) = \bigotimes_{j=1}^{n} e^{-ix_jY/2}, \tag{5}$$

where R_y is the rotation operator about the $\hat{y}$ axis of the Bloch sphere and Y is the Pauli-Y operator.

3.1.2. Variational evolution

The first part encodes the each of input patch $\boldsymbol{x}$ as a quantum state $|\boldsymbol{x}\rangle$. In the second part, applying a unitary $U_{var}(\boldsymbol{\theta})$, we map the state $|\boldsymbol{x}\rangle$ into another state $|\boldsymbol{x}; \boldsymbol{\theta}\rangle$. Utilizing the power of quantum mechanics, we hope that it can improve the usefulness of features. The operation of the second part is:

$$F_2(\boldsymbol{x}) = U_{var}(\boldsymbol{\theta})|\boldsymbol{x}\rangle = |\boldsymbol{x}; \boldsymbol{\theta}\rangle, \tag{6}$$

where $U_{var}(\boldsymbol{\theta})$ is an ansatz characterized by the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\theta}$ is a vector whose dimension depends on the type of ansatz. It is important to find an expressive ansatz that can achieve a broad range of quantum states. More generally, it is possible to repeat $U_{var}(\boldsymbol{\theta})$ to reach more quantum states. But this can increase the complexity of the model, and thus demands more training time.

In order to implement the ansatz in a real quantum device, we select $\{H, R_x, R_y, R_z, CNOT\}$ to compose the ansatz. In this case, motivated by the universality of QAOA [48,49], we use a QAOA-heuristic circuit as an ansatz. The QAOA-heuristic circuit is as shown in Fig. 2.

3.1.3. Decoding

To access the information of the quantum states, we need to “decode” the quantum state $|\boldsymbol{x}; \boldsymbol{\theta}\rangle$ into classical data. This step consists of measurement and classical post-processing. At the measurement part, we measure a expectation value of a observable $\hat{O}$. Then, in classical post-processing part, we transform this expectation value to get the output by adding a bias b. The output is expressed as:

$$F_3(\boldsymbol{x}) = \sigma(\mathrm{tr}(|\boldsymbol{x};\boldsymbol{\theta}\rangle\langle\boldsymbol{x};\boldsymbol{\theta}|\hat{O}) + b), \tag{7}$$

where $|\boldsymbol{x};\boldsymbol{\theta}\rangle$ is the output state from the variational evolution part and b is the bias. $\hat{O}$ is an observable composed of $\{\hat{H}_k\}_{k=1}^{l}$ where H_k is a fixed Hamiltonian. $\mathrm{tr}(\cdot)$ is the trace operator. $\sigma(\cdot)$ is an activation function. Its function is to manipulate the outputs into a range of $[0, 2\pi]$ or $[-\pi, \pi]$.

Thus, one step in the forward pass of a QFE layer is expressed roughly through the following equation:

$$F(\boldsymbol{x}) = \sigma\Big(\mathrm{tr}\Big(U_{var}(\boldsymbol{\theta})U_{en}(\boldsymbol{x})|0\rangle\langle 0|U_{en}(\boldsymbol{x})^{\dagger}U_{var}(\boldsymbol{\theta})^{\dagger}\hat{O}\Big) + b\Big). \tag{8}$$

Here a variant of the Tanh activation function is used in our experiments, which is shown as follows:

$$\sigma(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \times \pi. \tag{9}$$

With a factor of π, we can manipulate the output rangle of QFE layers from $[-1, 1]$ to $[-\pi, \pi]$.

3.2. Backward pass of QFE layer

In classical neural networks, gradient descent based optimization algorithms are usually applied to update the parameters. A powerful routine in training is the backpropagation algorithm, which combines the chain rule for derivatives and stored values from intermediate layers to efficiently compute gradients. However, this strategy can not be generalized straightforwardly to the QFE layer, because intermediate values from a quantum circuit can only be accessed via measurements, which will cause the collapse of quantum states. Using the parameter-shift rule and the chain rule, we proposed a training algorithm to compute the gradients of QFE layer which can be embedded in the standard backpropagation. Hence we can train the hybrid model as a whole.

Suppose a QFE layer with parameters $\boldsymbol{\theta}$, b, whose input is Z and output is A. The size of Z is $m \times m$. The size of kernel is $f \times f$ and the stride is s, therefore, the size of A is $(\lfloor\frac{m-f}{s}\rfloor + 1) \times (\lfloor\frac{m-f}{s}\rfloor + 1)$. Let $n = \lfloor\frac{m-f}{s}\rfloor + 1$ to make the procedure clearer. Then, Z and A can be expressed as:

$$Z = \begin{pmatrix} z_{11} & z_{12} & \cdots & z_{1m} \\ z_{21} & z_{22} & \cdots & z_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ z_{m1} & z_{m2} & \cdots & z_{mm} \end{pmatrix}, \tag{10}$$

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}. \tag{11}$$

L denotes the loss function. Firstly, consider the condition $s = 1$ which is be commonly used. Assuming that the partial derivative $\frac{\partial L}{\partial A}$ was known. $\frac{\partial a_{ij}}{\partial z_{uv}}$ and $\frac{\partial a_{ij}}{\partial \theta_k}$ can be calculated by the parameter-shift rule, where $i, j = 1, \dots, n,\ u, v = 1, \dots, m$ and $k = 1, \dots, l$. To implement the backpropagation of a QFE layer, we need to get $\frac{\partial L}{\partial \boldsymbol{\theta}}$, $\frac{\partial L}{\partial b}$ and $\frac{\partial L}{\partial Z}$. For convenience, we flatten $\frac{\partial L}{\partial A}$ row by row as follows:

$$\frac{\partial L}{\partial A} = \begin{pmatrix} \frac{\partial L}{\partial a_{11}} & \frac{\partial L}{\partial a_{12}} & \cdots & \frac{\partial L}{\partial a_{1n}} \\ \frac{\partial L}{\partial a_{21}} & \frac{\partial L}{\partial a_{22}} & \cdots & \frac{\partial L}{\partial a_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial L}{\partial a_{n1}} & \frac{\partial L}{\partial a_{n2}} & \cdots & \frac{\partial L}{\partial a_{nn}} \end{pmatrix} \to \frac{\partial L}{\partial \overline{A}} = \begin{pmatrix} \frac{\partial L}{\partial \overline{a}_1} \\ \frac{\partial L}{\partial \overline{a}_2} \\ \vdots \\ \frac{\partial L}{\partial \overline{a}_{n\times n}} \end{pmatrix}. \tag{12}$$

For $\frac{\partial L}{\partial b}$, by the chain rule, we have

$$\frac{\partial L}{\partial b} = \sum_{i,j} \frac{\partial a_{i,j}}{\partial b} \frac{\partial L}{\partial a_{i,j}} = \sum_{i,j} \frac{\partial L}{\partial a_{i,j}} = \sum \frac{\partial L}{\partial A}, \tag{13}$$

where

$$\frac{\partial a_{i,j}}{\partial b} = \frac{\partial \Big(\mathrm{tr}(|\boldsymbol{x}; \boldsymbol{\theta}\rangle\langle \boldsymbol{x}; \boldsymbol{\theta}| \hat{O} + b) \Big)}{\partial b} = \frac{\partial b}{\partial b} = 1 \quad i, j \in \{1, 2, \dots, n\}. \tag{14}$$

For $\frac{\partial L}{\partial \boldsymbol{\theta}}$, by the chain rule, we have

$$\begin{aligned} \frac{\partial L}{\partial \theta_k} &= \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial a_{ij}}{\partial \theta_k} \frac{\partial L}{\partial a_{ij}} = \sum_{i=1}^{n\times n} \frac{\partial \overline{a}_i}{\partial \theta_k} \frac{\partial L}{\partial \overline{a}_i} \\ &= \begin{pmatrix} \frac{\partial \overline{a}_1}{\partial \theta_k} & \frac{\partial \overline{a}_2}{\partial \theta_k} & \cdots & \frac{\partial \overline{a}_{n\times n}}{\partial \theta_k} \end{pmatrix} \begin{pmatrix} \frac{\partial L}{\partial \overline{a}_1} \\ \frac{\partial L}{\partial \overline{a}_2} \\ \vdots \\ \frac{\partial L}{\partial \overline{a}_{n\times n}} \end{pmatrix} = \frac{\partial \overline{A}}{\partial \theta_k}^T \frac{\partial L}{\partial \overline{A}}. \end{aligned} \tag{15}$$

Thus,

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \begin{pmatrix} \frac{\partial L}{\partial \theta_1} \\ \frac{\partial L}{\partial \theta_2} \\ \vdots \\ \frac{\partial L}{\partial \theta_l} \end{pmatrix} = \frac{\partial \overline{A}}{\partial \boldsymbol{\theta}}^T \frac{\partial L}{\partial \overline{A}}, \tag{16}$$

where

$$\frac{\partial \overline{A}^T}{\partial \boldsymbol{\theta}} = \begin{pmatrix} \frac{\partial \overline{a}_1}{\partial \theta_1} & \frac{\partial \overline{a}_2}{\partial \theta_1} & \cdots & \frac{\partial \overline{a}_{n\times n}}{\partial \theta_1} \\ \frac{\partial \overline{a}_1}{\partial \theta_2} & \frac{\partial \overline{a}_2}{\partial \theta_1} & \cdots & \frac{\partial \overline{a}_{n\times n}}{\partial \theta_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \overline{a}_1}{\partial \theta_l} & \frac{\partial \overline{a}_l}{\partial \theta_1} & \cdots & \frac{\partial \overline{a}_{n\times n}}{\partial \theta_l} \end{pmatrix}. \tag{17}$$

To compute the $\frac{\partial L}{\partial Z}$, we define $\frac{\partial A}{\partial Z}$ as

$$\frac{\partial A}{\partial Z} = \begin{pmatrix} \frac{\partial A}{\partial z_{11}} & \frac{\partial A}{\partial z_{12}} & \cdots & \frac{\partial A}{\partial z_{1m}} \\ \frac{\partial A}{\partial z_{21}} & \frac{\partial A}{\partial z_{22}} & \cdots & \frac{\partial A}{\partial z_{2m}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial A}{\partial z_{m1}} & \frac{\partial A}{\partial z_{m2}} & \cdots & \frac{\partial A}{\partial z_{mm}} \end{pmatrix}, \tag{18}$$

where

$$\frac{\partial A}{\partial z_{uv}} = \begin{pmatrix} \frac{\partial a_{11}}{\partial z_{uv}} & \frac{\partial a_{12}}{\partial z_{uv}} & \cdots & \frac{\partial a_{1n}}{\partial z_{uv}} \\ \frac{\partial a_{21}}{\partial z_{uv}} & \frac{\partial a_{22}}{\partial z_{uv}} & \cdots & \frac{\partial a_{2n}}{\partial z_{uv}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial a_{n1}}{\partial z_{uv}} & \frac{\partial a_{n2}}{\partial z_{uv}} & \cdots & \frac{\partial a_{nn}}{\partial z_{uv}} \end{pmatrix} \quad u, v \in \{1, 2, \ldots, m\}. \tag{19}$$

Then, the $\frac{\partial L}{\partial Z}$ can be expressed as

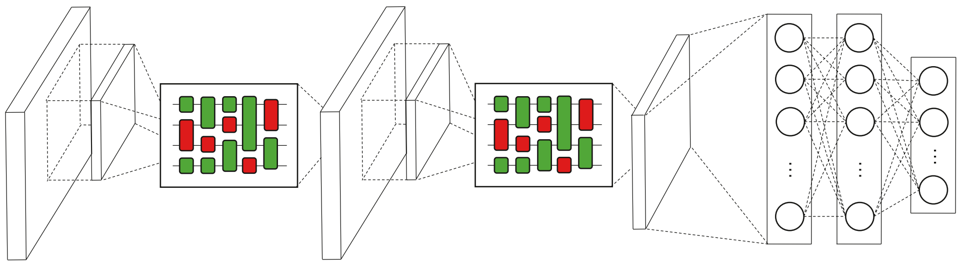
$$\frac{\partial L}{\partial Z} = \frac{\partial A}{\partial Z} * \frac{\partial L}{\partial A}, \tag{20}$$

where '$*$' denotes the convolution operation with the stride n.

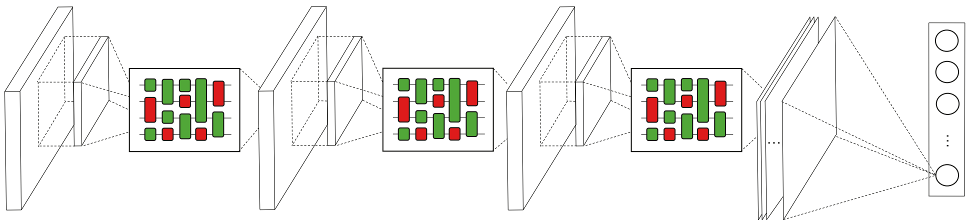
Here is the backpropagation of a QFE layer with only one quantum convolution kernel. The case of multiple quantum convolution kernels and stride $s > 1$ can be obtained by extension easily, so that won't be covered again here.

4. Experiments

In this section, we evaluate the performance of our hybrid models on the MNIST [34] and Fashion-MNIST datasets by numerical simulations. All of the experiments are performed with Yao.jl [50] package in Julia language.



(a) Model 1 (two QFE layers and a classical “classifier”)



(b) Model 2 (three QFE layers and a GAP layer)

Fig. 3. The structures of evaluated models.

4.1. Dataset

The MNIST dataset is composed of 10 classes of handwritten digits 0–9, and the Fashion-MNIST dataset is composed of 10 classes of fashion products. There are 60,000 training images and 10,000 testing images for both datasets. Each image is a grayscale image of size 28×28. To save the training time, the data needs to be preprocessed. For MNIST dataset, the image is cropped into 22×22 without information loss, as the original images are 20×20. For Fashion-MNIST dataset, we perform a classification task for 4 classes (“Trouser”, “Pullover”, “Bag”, “Ankle boot”), and each sample is down-sampled to 22×22 by bilinear interpolation. For each dataset, we use 6000 and 600 samples for training and test sets, respectively. The labels use one-hot encoding.

4.2. Effectiveness

Architecture selection. Here, to verify the feasibility of the scheme, we evaluate models of following architectures. The models all consist of stacked QFE layers instead of linear convolutional layers.

1. *QFE layers with fully-connected layers*. A model, which is similar to the classical CNN, replaces the linear convolutional layers with QFE layers. The structure is as followed: QFE1 - POOL1 - QFE2 - POOL2 - FC1 - FC2 - FC3. In this model, the fully-connected layers work as a “classifier”. We refer to this model as “model 1”, as shown in Fig. 3(a).
2. *QFE layers with global average pooling (GAP)*. A model uses GAP instead of fully-connected layers at the top of the network, with the following structure: QFE1 - POOL1 - QFE2 - POOL2 - QFE3 - GAP. In this case, GAP introduced in Lin et al. [29] is

Table 1
The comparison of model architectures.

Model	Feature extraction module						Classifier module
	First layer	Number of output channels	Second layer	Number of output channels	Third layer	Number of output channels	
Model 1	QFE layer	6	QFE layer	8	\	\	Fully-connected network
Model 2	QFE layer	6	QFE layer	8	QFE layer	10 (MNIST) or 4 (Fashion-MNIST)	GAP layer
Classical model 1	Conv layer	6	Conv layer	8	\	\	Fully-connectednetwork
Classical model 2	Mlpconv layer	6	Mlpconv layer	8	Mlpconv layer	10 (MNIST) or 4 (Fashion-MNIST)	GAP layer

used to remove the “classical factors” of the network. We refer to this model as “model 2”, as shown in Fig. 3(b).

Each QFE layer uses “quantum” filters of size 3×3. Max pooling is used in the pooling layers which downsamples the size of input by a factor of 2.

Models for comparison. To explore potential advantages over classical models, we make comparisons with similar classical models.

1. *Convolutional layers with full-connected layers.* A similar classical CNN to model 1 architecture, except that QFE layers are replaced by convolutional layers. Its structure is as follows: CONV1 - POOL1 - CONV2 - POOL2 - FC1 - FC2 - FC3. We refer to this model as “classical model 1”.
2. *Mlpconv layers with GAP.* A similar classical CNN to model 2 architecture, except instead of using QFE layers, mlpconv layers are applied instead. Its structure is as follows: MLPCONV1 - POOL1- MLPCONV2 - MLPCONV3 - GAP. We refer to this model as “classical model 2”.

Each convolutional layer uses filters of size 3×3 and each mlpconv layer uses a multilayer perceptron with two hidden layers. The comparison of model architectures between hybrid models and classical models is shown in the Table 1.

Training strategy. For fair comparison, we adopt the same dataset for training. To save training time, we also explore a step-by-step training procedure. First, we train the network from scratch with a relatively small dataset. Then when the training is saturated, we add more training data for fine-tuning and repeat this step. With this strategy, the training converges sooner and faster than training with the larger dataset from the beginning. For initialization, the weights of QFE layers are initialized by drawing randomly from a uniform distribution with a range of $[-\pi, \pi]$, while the weights of convolutional layers and fully-connected layers are initialized by a Gaussian distribution with zero mean and standard deviation 0.001. The models are trained using the Adam optimization algorithm with mini-batches of size 50. The initial learning rate is 0.01. The learning rate and the batch size are adjusted manually during training.

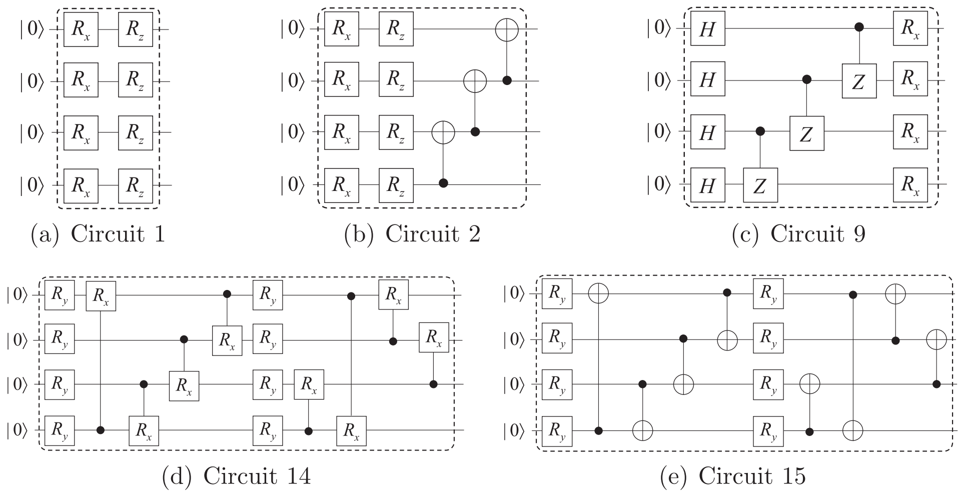


Fig. 4. Circuits 1, 2, 9, 14 and 15 in [51], which are used in controlling experiments.

4.3. Investigation of different settings

To investigate the property of the model, we design a set of controlling experiments with different settings of two variables - the type of the ansatz and the number of layers of ansatz, based on the structure of model 2 (i.e. QFE1 - POOL1- QFE2 - QFE3 - GAP).

4.3.1. Type of ansatz

Recent work by Sim et al.[51] studies expressibility the entangling capability of quantum circuits, showing these descriptors are related to the performance of PQCs-based algorithms. Here, we compare the model performance with different ansaetze. Specifically, besides the QAOA-heuristic circuit, we choose circuits 1, 2, 9, 14 and 15 in Sim et al. [51], as shown in Fig. 4.

4.3.2. Number of layers

In general, the performance would improve if we increase the number of layers of ansatz at the cost of running time. Here, we examine the model sensitivity to different number of layers. In previous experiments, we set the number of layers of ansatz $L = 3$ for each “quantum” filter. To be consistent with previous experiments, we keep the kinds of ansaetze but change the number of layers from 1 to 5. All the other settings remain the same with model 2.

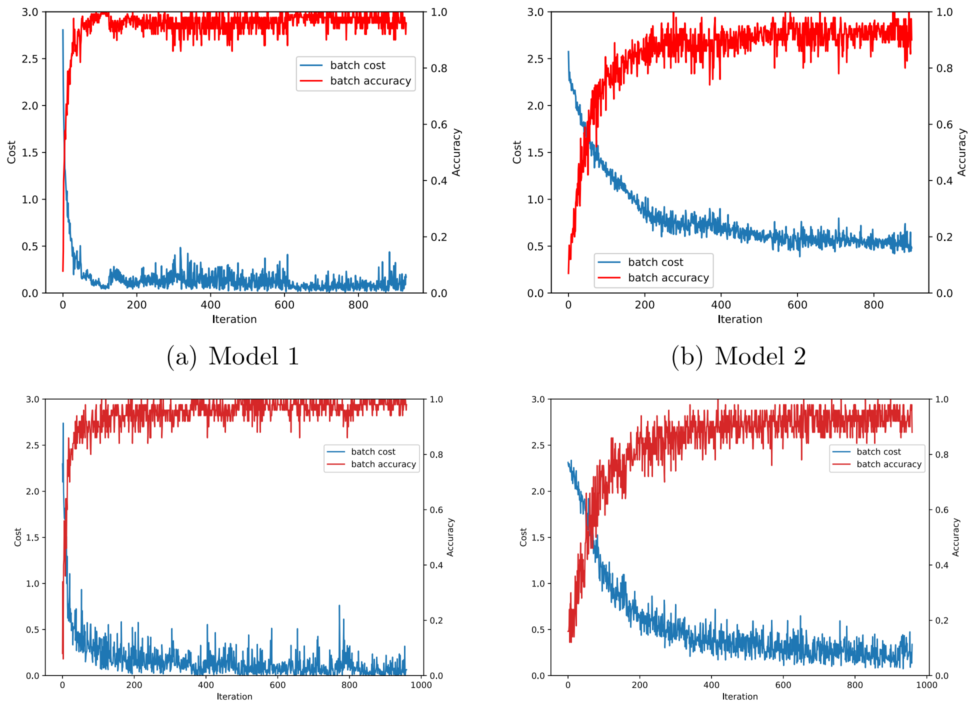
Totally, we examine 30 models for the same training data with 2560 samples for 9 epochs. The experimental setup is shown in the Table 2.

4.4. Results

We first report results on the classification to demonstrate that our method works well with and without fully-connected layers. Then we compare the performance between different settings.

Table 2
Experimental setup to investigate different settings.

Epoch	Learning rate	Batch size
1	0.01	32
2–3	0.005	32
4–6	0.001	32
7–9	0.0005	16

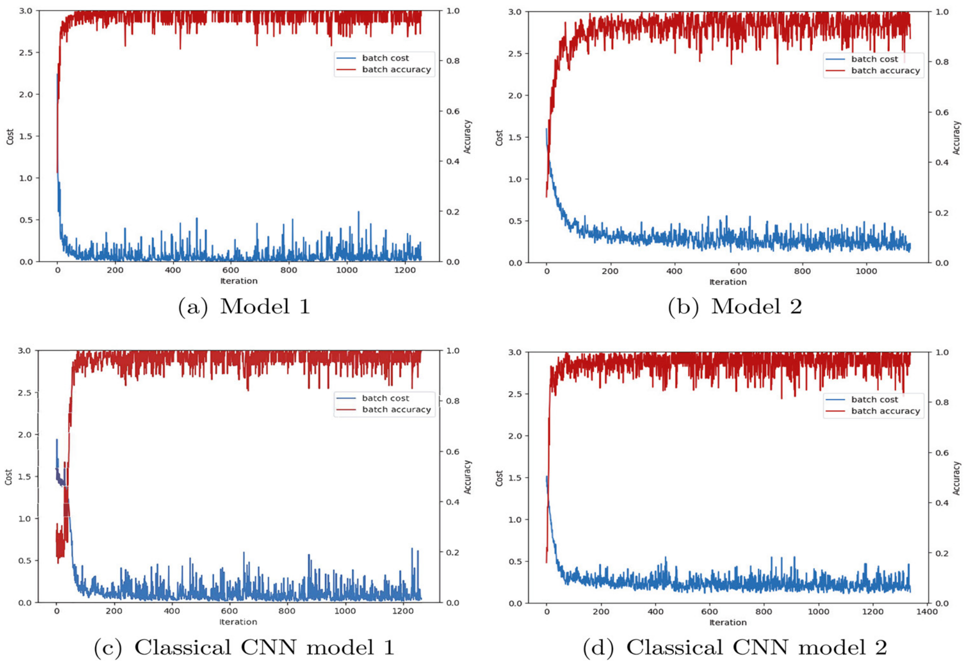


(a) Model 1 (b) Model 2

(c) Classical CNN model 1 (d) Classical CNN model 2

Fig. 5. Training cost and accuracy of models on the MNIST dataset.

Classification. For the MNIST dataset, the training cost and accuracy of model 1 and model 2 are shown in Fig. 5(a) and (b). For testing error, model 1 and model 2 achieve 3.1% and 6.8% respectively. As for classical model 1 and classical model 2 that are used for comparisons, their training cost and accuracy are shown in Fig. 5(c) and (d), with 5.3% and 7.9% testing error respectively. For the Fashion-MNIST dataset in 4 classes, the training cost and accuracy of model 1, model 2, classical model 1 and classical model 2 are shown in Fig. 6. Their testing errors are 2.16%, 4.83% for model 1 and model 2, and 2.16%, 5% for classical model 1 and classical model 2. The results show that with similar structures, our models outperform classical CNN models at test accuracy in most cases. However, for further



(a) Model 1

(b) Model 2

(c) Classical CNN model 1

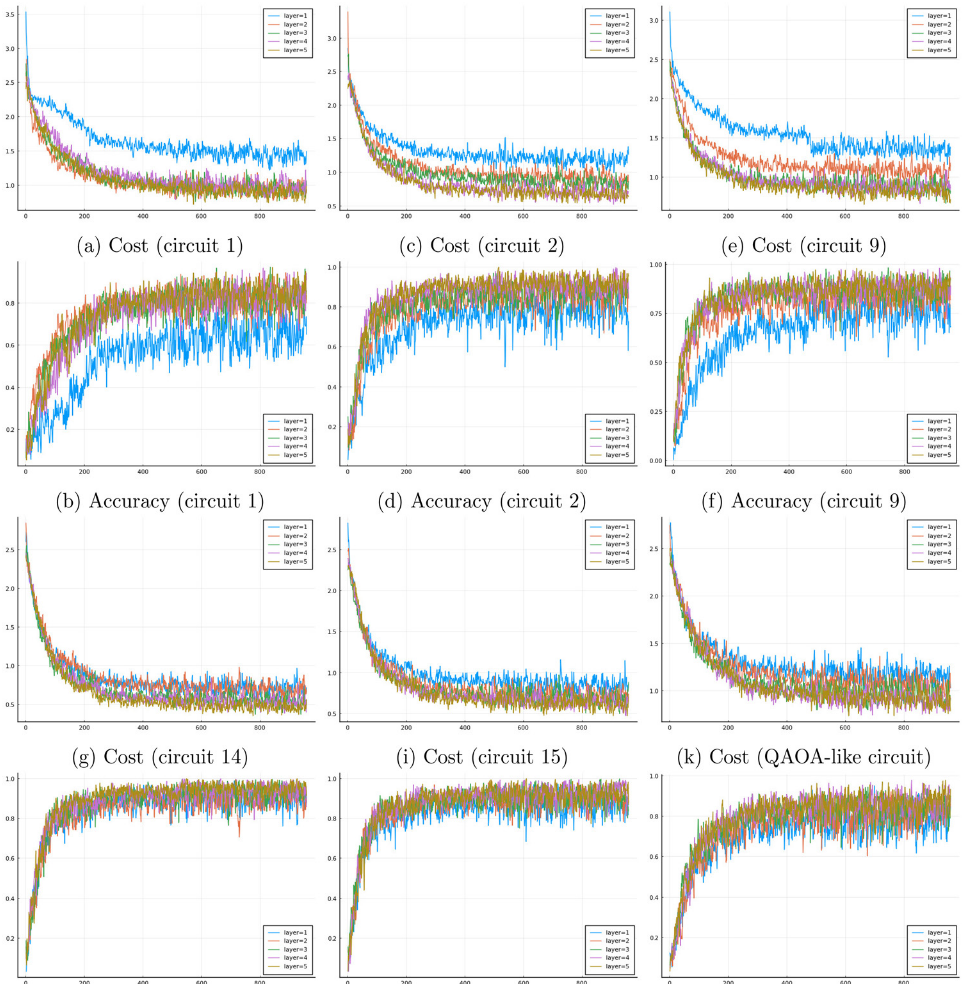
(d) Classical CNN model 2

Fig. 6. Training cost and accuracy of models on the Fashion-MNIST dataset.

comparison, more experiments need to be performed on larger models and more datasets. Note that these results are reasonable for the toy models, and removing classical fully-connected layers, there is a small performance drop of about 2–3%.

Compare using different settings. To further compare our approach with different settings mentioned above, we train the models with the same dataset based on the MNIST dataset and the results are shown in Fig. 7. Note that these results are obtained using different ansaetze with different layers. We can observe that the model converges lower cost and achieves higher accuracy with deeper layers in ansaetze. However, we note that "saturation" phenomenon exists, which means adding addition layers to PQCs, the performance of the model does not always continue to improve. This phenomenon appears more and less in all of the ansaetze. Take circuit 9 as an example, the model with 2-layer PQCs converges on a lower value and higher accuracy than that with 1-layer PQCs, while the curves of cost and accuracy are almost the same respectively, when the models equipped with circuit 9 in 4 layers and 5 layers. The "saturation" phenomenon may relate with expressibility saturation [51] or barren plateaus [52]. To understand how expressibility correlates with the model performance further, future research is needed.

This section aims to demonstrate a proof of concept for the proposed method on a real-world dataset. Because the training is time-consuming, we construct our architectures based on the small model which has only two or three layers. Note that model 2 which has no classical fully-connected layers achieves 6.8% test set error. This shows the effectiveness of QFE layers and their training algorithm.



(a) Cost (circuit 1) (c) Cost (circuit 2) (e) Cost (circuit 9)

(b) Accuracy (circuit 1) (d) Accuracy (circuit 2) (f) Accuracy (circuit 9)

(g) Cost (circuit 14) (i) Cost (circuit 15) (k) Cost (QAOA-like circuit)

(h) Accuracy (circuit 14) (j) Accuracy (circuit 15) (l) Accuracy (QAOA-like circuit)

Fig. 7. Costs and accuracies of models with different settings.

5. Conclusions

This paper proposed a hybrid quantum-classical network with deep architectures. The new structure consists of QFE layers which use PQCs to “convolve” the input. QFE layers generalize convolution to the high dimensional Hilbert space and model the local patches better. It has been shown that our hybrid models can achieve competitive performances based on simple architectures and tend to have higher test accuracies compared to similar classical CNNs. Besides, we compare the performances of models with different ansaetze in different depths, showing that the models with ansaetze in high expressibility performs better. In practice, the network architecture and the initialization method have a significant impact on performance. However, there are few effective methods to avoid the barren plateaus yet. On the other hand, since the modern classical network are so deep and our method provides many possibilities via PQCs, we cannot perform exhaustive tests to find the best sequence of QFE layers and the best combination of circuit ansaetze. Due to the large number of choices of PQCs, we cannot perform a brute force search for all the possibilities, while this opens a new space for the construction of hybrid quantum-classical networks. We expect the introduction of QFE layers in more architectures and extensive hyperparameter searches can improve the performance.

Besides, it is noted that quantum neural tangent kernel theory [53–55] is developed recently, which can be applied to quantum neural networks. It will be interesting to analyze the model with QFE layers based on the quantum neural tangent kernel theory. This open question is left for future research.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Tong Dou: Methodology, Writing – original draft, Writing – review & editing. **Guofeng Zhang:** Writing – review & editing. **Wei Cui:** Conceptualization, Writing – review & editing.

Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant 2022YFB3103100, National Natural Science Foundation of China under Grants 62273154, 62173288, Hong Kong Research Grant Council under Grants nos. 15203619 and 15208418, Shenzhen Fundamental Research Fund under Grant JCYJ20190813165207290 and the CAS AMSS-polyU Joint Laboratory of Applied Mathematics.

References

[1] F. Arute, K. Arya, R. Babbush, D. Bacon, J. Bardin, R. Barends, R. Biswas, S. Boixo, F. Brandao, D. Buell, Quantum supremacy using a programmable superconducting processor, Nature 574 (7779) (2019) 505–510.
[2] J. Preskill, Quantum computing in the NISQ era and beyond, Quantum 2 (2018) 79.
[3] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, S. Lloyd, Quantum machine learning, Nature 549 (7671) (2017) 195–202.
[4] P. Rebentrost, M. Mohseni, S. Lloyd, Quantum support vector machine for big data classification, Phys. Rev. Lett. 113 (13) (2014) 130503.
[5] S. Lloyd, M. Mohseni, P. Rebentrost, Quantum principal component analysis, Nat. Phys. 10 (9) (2014) 631–633.
[6] M. Amin, E. Andriyash, J. Rolfe, B. Kulchytskyy, R. Melko, Quantum Boltzmann machine, Phys. Rev. X 8 (2) (2018) 021050.
[7] A. Mari, T. Bromley, J. Izaac, M. Schuld, N. Killoran, Transfer learning in hybrid classical-quantum neural networks, Quantum 4 (2020) 340.
[8] V. Havlíček, A. Córcoles, K. Temme, A. Harrow, A. Kandala, J. Chow, J. Gambetta, Supervised learning with quantum-enhanced feature spaces, Nature 567 (7747) (2019) 209–212.
[9] M. Schuld, N. Killoran, Quantum machine learning in feature hilbert spaces, Phys. Rev. Lett. 122 (4) (2019) 040504.
[10] M. Benedetti, E. Lloyd, S. Sack, M. Fiorentini, Parameterized quantum circuits as machine learning models, Quantum Sci. Technol. 4 (4) (2019) 043001.

[11] P. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM Rev 41 (2) (1999) 303–332.
[12] L. Grover, Quantum mechanics helps in searching for a needle in a haystack, Phys. Rev. Lett. 79 (2) (1997) 325–328.
[13] A. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for linear systems of equations, Phys. Rev. Lett. 103 (15) (2009) 150502.
[14] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. Love, A. Aspuru-Guzik, J. O'Brien, A variational eigenvalue solver on a photonic quantum processor, Nat. Commun. 5 (1) (2014) 1–7.
[15] D. Wecker, M. Hastings, M. Troyer, Progress towards practical quantum variational algorithms, Phys. Rev. A 92 (4) (2015) 042303.
[16] J. McClean, J. Romero, R. Babbush, A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms, New J. Phys. 18 (2) (2016) 023023.
[17] E. Farhi, J. Goldstone, S. Gutmann, A quantum approximate optimization algorithm, (2014), arXiv:1411.4028.
[18] S. Hadfield, Z. Wang, B. O'Gorman, E. Rieffel, D. Venturelli, R. Biswas, From the quantum approximate optimization algorithm to a quantum alternating operator ansatz, Algorithms 12 (2) (2019) 34.
[19] M. Schuld, I. Sinayskiy, F. Petruccione, The quest for a quantum neural network, quantum, Inf. Process. 13 (11) (2014) 2567–2586.
[20] E. Farhi, H. Neven, Classification with quantum neural networks on near term processors, (2018), arXiv:1802.06002.
[21] S. Lloyd, M. Schuld, A. Ijaz, J. Izaac, N. Killoran, Quantum embeddings for machine learning, (2020). arXiv:2001.03622.
[22] I. Cong, S. Choi, M. Lukin, Quantum convolutional neural networks, Nat. Phys. 15 (12) (2019) 1273–1278.
[23] M. Henderson, S. Shakya, S. Pradhan, T. Cook, Quanvolutional neural networks: powering image recognition with quantum circuits, Quantum Mach. Intell. 2 (1) (2020) 1–9.
[24] I. Kerenidis, J. Landman, A. Prakash, Quantum algorithms for deep convolutional neural networks, in: 2020 Proceedings of the International Conference on Learning Representations (ICLR, 2020.
[25] 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) (2020) 044003.
[26] H. Zheng, Z. Li, J. Liu, S. Strelchuk, R. Kondor, Speeding up learning quantum states through group equivariant convolutional quantum ansätze, (2021). arXiv:2112.07611.
[27] D.-L. Deng, Quantum enhanced convolutional neural networks for NISQ computers, Sci. China Phys. Mech. Astron. 64 (10) (2021) 1.
[28] W. Li, D.-L. Deng, Recent advances for quantum classifiers, Sci. China Phys. Mech. Astron. 65 (2) (2022) 1–23.
[29] M. Lin, Q. Chen, S. Yan, Network in network, in: 2014 Proceedings of the International Conference on Learning Representations (ICLR, 2013.
[30] M. Schuld, R. Sweke, J. Meyer, Effect of data encoding on the expressive power of variational quantum machine learning models, Phys. Rev. A 103 (3) (2021) 032430.
[31] T. Goto, Q. Tran, K. Nakajima, Universal approximation property of quantum machine learning models in quantum-enhanced feature spaces, Phys. Rev. Lett. 127 (9) (2021) 090506.
[32] J. Liu, K. Lim, K. Wood, W. Huang, C. Guo, H.-L. Huang, Hybrid quantum-classical convolutional neural networks, Sci. China Phys. Mech. Astron. 64 (9) (2021) 1–8.
[33] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, L. Jackel, Backpropagation applied to handwritten zip code recognition, Neural Comput. 1 (4) (1989) 541–551.
[34] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[35] A. Krizhevsky, I. Sutskever, G. Hinton, ImageNet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems (NeurIPS, 2012, pp. 1097–1105.
[36] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788.
[37] J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR, 2015, pp. 3431–3440.
[38] L. Zhang, L. Lin, X. Liang, K. He, Is faster R-CNN doing well for pedestrian detection? in: European Conference on Computer Vision (ECCV, 2016, pp. 443–457.
[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, I. Ł. Kaiser, Attention is all you need, in: Advances in Neural Information Processing Systems (NeurIPS, 2017, pp. 5998–6008.

[40] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR, 2015, pp. 1–9.
[41] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: Proceedings of the International Conference on Learning Representations (ICLR, 2014.
[42] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR, 2016, pp. 770–778.
[43] G. Huang, Z. Liu, L. Maaten, K. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR, 2017, pp. 2261–2269.
[44] P.-L. Dallaire-Demers, N. Killoran, Quantum generative adversarial networks, Phys. Rev. A 98 (1) (2018) 012324.
[45] J. Romero, J. Olson, A. Aspuru-Guzik, Quantum autoencoders for efficient compression of quantum data, Quantum Sci. Technol. 2 (4) (2017) 045001.
[46] K. Sharma, S. Khatri, M. Cerezo, P. Coles, Noise resilience of variational quantum compiling, New J. Phys. 22 (4) (2020) 043006.
[47] K. Mitarai, M. Negoro, M. Kitagawa, K. Fujii, Quantum circuit learning, Phys. Rev. A 98 (3) (2018) 032309.
[48] S. Lloyd, Quantum approximate optimization is computationally universal, (2018), arXiv:1812.11075.
[49] M. Morales, J. Biamonte, Z. Zimborás, On the universality of the quantum approximate optimization algorithm, Quantum Inf. Process. 19 (9) (2020) 1–26.
[50] X.-Z. Luo, J.-G. Liu, P. Zhang, L. Wang, Yao. jl: extensible, efficient framework for quantum algorithm design, Quantum 4 (2020) 341.
[51] S. Sim, P. Johnson, A. Aspuru-Guzik, Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms, Adv. Quantum Technol. 2 (12) (2019) 1900070.
[52] J. McClean, S. Boixo, V. Smelyanskiy, R. Babbush, H. Neven, Barren plateaus in quantum neural network training landscapes, Nat. Commun. 9 (1) (2018) 1–6.
[53] K. Nakaji, H. Tezuka, N. Yamamoto, Quantum-enhanced neural networks in the neural tangent kernel framework, (2021) arXiv:2109.03786.
[54] N. Shirai, K. Kubo, K. Mitarai, K. Fujii, Quantum tangent kernel, (2021). arXiv:2111.02951.
[55] J. Liu, F. Tacchino, J. Glick, L. Jiang, A. Mezzacapo, Representation learning via quantum neural tangent kernels, PRX Quantum 3 (3) (2021) 030323.
[56] Y. Pan, Y. Tong, S. Xue, G. Zhang, Efficient depth selection for the implementation of noisy quantum approximate optimization algorithm, Journal of the Franklin Institute 359 (18) (2022) 11273–11287.