Quantum Machine Intelligence (2022) 4: 3
https://doi.org/10.1007/s42484-021-00061-x

RESEARCH ARTICLE

# Quantum convolutional neural network for classical data classification

**Tak Hur**[1] · **Leeseok Kim**[2] · **Daniel K. Park**[3]

Received: 2 August 2021 / Accepted: 30 November 2021 / Published online: 10 February 2022


**Abstract**

With the rapid advance of quantum machine learning, several proposals for the quantum-analogue of convolutional neural network (CNN) have emerged. In this work, we benchmark fully parameterized quantum convolutional neural networks (QCNNs) for classical data classification. In particular, we propose a quantum neural network model inspired by CNN that only uses two-qubit interactions throughout the entire algorithm. We investigate the performance of various QCNN models differentiated by structures of parameterized quantum circuits, quantum data encoding methods, classical data pre-processing methods, cost functions and optimizers on MNIST and Fashion MNIST datasets. In most instances, QCNN achieved excellent classification accuracy despite having a small number of free parameters. The QCNN models performed noticeably better than CNN models under the similar training conditions. Since the QCNN algorithm presented in this work utilizes fully parameterized and shallow-depth quantum circuits, it is suitable for Noisy Intermediate-Scale Quantum (NISQ) devices.

**Keywords** Quantum machine learning · Convolutional neural network · Deep learning

## 1 Introduction

Machine learning techniques with artificial neural networks are ubiquitous in modern society as the ability to make reliable predictions from the vast amount of data is essential in various domains of science and technology. A convolutional neural network (CNN) is one such example, especially for data with a large number of features. It effectively captures spatial correlation within data and learns important features (Lecun et al. 1998), which is shown to be useful for many pattern recognition problems, such as image classification, signal processing, and natural language processing (LeCun et al. 2015). It has also opened the path to Generative Adversarial Networks (GANs) (Goodfellow et al. 2014). CNNs are also rising as a useful tool for scientific research, such as in high-energy physics (Aurisano et al. 2016; Acciarri et al. 2017), gravitational wave detection (George and Huerta 2018) and statistical physics (Tanaka and Tomiya 2017). By all means, the computational power required for the success of machine learning algorithms increases with the volume of data, which is increasing at an overwhelming rate. With the potential of quantum computers to outperform any foreseeable classical computers for solving certain computational tasks, quantum machine learning (QML) has emerged as the potential solution to address the challenge of handling an ever-increasing amount of data. For example, several innovative quantum machine learning algorithms have been proposed to offer speedups over their classical counterparts (Lloyd et al. 2013; 2014; Wiebe et al. 2015; Rebentrost et al. 2014; Kerenidis et al. 2019; Blank et al. 2020). Motivated by the benefits of CNN and the potential power of QML, quantum convolutional neural network (QCNN) algorithms have been developed (Cong et al. 2019; Kerenidis et al. 2019; Liu et al. 2021; Henderson et al. 2020; Chen et al. 2020; Li et al. 2020; MacCormack et al. 2020; Wei et al.

Tak Hur and Leeseok Kim contributed equally to this work and are listed in alphabetical order.

✉ Daniel K. Park
dkp.quantum@gmail.com

[1] Department of Physics, Imperial College London, London, UK

[2] Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM, USA

[3] Sungkyunkwan University Advanced Institute of Nanotechnology, Suwon, South Korea

2021; Mangini et al. 2021) (see Appendix A for a brief summary and comparison of other approaches to QCNN). Previous constructions of QCNN have reported success in developing efficient quantum arithmetic operations that exactly implement the basic functionalities of classical CNN or in developing parameterized quantum circuits inspired by key characteristics of CNN. While the former likely requires fault-tolerant quantum devices, the latter has been focused on quantum data classification. In particular, Cong et al. proposed a fully parameterized quantum circuit (PQC) architecture inspired by CNN and demonstrated its success for some quantum many-body problems (Cong et al. 2019). However, the study of fully parameterized QCNN for performing pattern recognition, such as classification, on classical data is missing.

In this work, we present a fully parameterized quantum circuit model for QCNN that solves supervised classification problems on classical data. In a similar vein to Cong et al. (2019), our model only uses two-qubit interactions throughout the entire algorithm in a systematic way. The PQC models—also known as variational quantum circuits (Cerezo et al. 2021)—are attractive since they are expected to be suitable for Noisy Intermediate-Scale Quantum (NISQ) hardware (Preskill 2018; Bharti et al. 2021). Another advantage of QCNN models for NISQ computing is their intrinsically shallow circuit depth. Furthermore, QCNN models studied in this work exploit entanglement, which is a global property, and hence have the potential to transcend classical CNN that is only able to capture local correlations. We benchmark the performance of the parameterized QCNN with respect to several variables, such as quantum data encoding methods, structures of parameterized quantum circuits, cost functions, and optimizers using two standard datasets, namely MNIST and Fashion MNIST, on Pennylane (Bergholm et al. 2020). The quantum encoding benchmark also examines classical dimensionality reduction methods, which is essential for early quantum computers with a limited number of logical qubits. The various QCNN models tested in this work employs a small number of free parameters, ranging from 12 to 51. Nevertheless, all QCNN models produced high classification accuracy, with the best case being about 99% for MNIST and about 94% for Fashion MNIST. Moreover, we discuss a QCNN model that only requires nearest neighbor qubit interactions, which is a desirable feature for NISQ computing. Comparing classification performances of QCNN and CNN models shows that QCNN is more favorable than CNN under the similar training conditions for both benchmarking datasets.

The remainder of the paper is organized as follows. Section 2 sets the theoretical framework of this work by describing the classification problem, the QCNN algorithm, and various methods for encoding classical data as a quantum state. Section 3 describes variables of the QCNN model, such as parameterized quantum circuits, cost functions, and classical data pre-processing methods, that are subject to our benchmarking study. Section 4 compares and presents the performance of various designs of QCNN for binary classification of MNIST and Fashion MNIST datasets. Conclusions are drawn and directions for future work are suggested in Section 5.

## 2 Theoretical framework

### 2.1 Classification

Classification is an example of pattern recognition, which is a fundamental problem in data science that can be effectively addressed via machine learning. The goal of $L$-class classification is to infer the class label of an unseen data point $\tilde{x} \in \mathbb{C}^N$, given a labelled dataset

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_M, y_M)\} \subset \mathbb{C}^N \times \{0, 1, \ldots, L-1\}.$$

The classification problem can be solved by training a parameterized quantum circuit. Hereinafter, we refer to fully parameterized quantum circuits trained for machine learning tasks as quantum neural network (QNN). For this supervised classification task, a QNN is trained by optimizing the parameters of quantum gates so as to minimize the cost function

$$C(\boldsymbol{\theta}) = \sum_{i=1}^{M} \alpha_i c(y_i, f(x_i, \boldsymbol{\theta}))$$

where $f(x_i, \boldsymbol{\theta})$ is the machine learning model defined by the set of parameters $\boldsymbol{\theta}$ that predicts the label of $x_i$, $c(a, b)$ quantifies the dissimilarity between $a$ and $b$, and $\alpha_i$ is a weight that satisfies $\sum_{i=1}^{M} \alpha_i = 1$. After the training is finished, the class label for the unseen data point $\tilde{x}$ is determined as $\tilde{y} = f(\tilde{x}, \boldsymbol{\theta}^*)$, where $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} C(\boldsymbol{\theta})$. If the problem is restricted to binary classification (i.e., $L = 2$), the class label can be inferred from a single-qubit von Neumann measurement. For example, the sign of an expectation value of $\sigma_z$ observable can represent the binary label (Park et al. 2021). Hereinafter, we focus on the binary classification, albeit potential future work towards multi-class classification will be discussed in Section 5.

### 2.2 Quantum convolutional neural network

An interesting family of quantum neural networks utilizes tree-like (or hierarchical) structures (Grant et al. 2018) with which the number of qubits from a preceding layer is reduced by a factor of two for the subsequent layer. Such architectures consist of $O(\log(n))$ layers for $n$ input qubits, thereby permitting shallow circuit depth. Moreover,

they can avoid one of the most critical problems in the PQC based algorithms known as "barren plateau", thereby guaranteeing the trainability (Pesah et al. 2021). These structures also make a natural connection to the tensor network, which serves as a useful ground for exploring many-body physics, neural networks, and the interplay between them.

The progressive reduction of the number of qubits is analogous to the pooling operation in CNN. A distinct feature of the QCNN architecture is the translational invariance, which forces the blocks of parameterized quantum gates to be identical within a layer. The quantum state resulting from an $i$th layer of QCNN can be expressed as

$$|\psi_i(\boldsymbol{\theta}_i)\rangle\langle\psi_i(\boldsymbol{\theta}_i)| = \mathrm{Tr}_{B_i}(U_i(\boldsymbol{\theta}_i)\,|\psi_{i-1}\rangle\langle\psi_{i-1}|\,U_i(\boldsymbol{\theta}_i)^\dagger), \quad (1)$$

where $\mathrm{Tr}_{B_i}(\cdot)$ is the partial trace operation over subsystem $B_i \in \mathbb{C}^{2^{n/2^i}}$, $U_i$ is the parameterized unitary gate operation that includes quantum convolution and the gate part of pooling, and $|\psi_0\rangle = |0\rangle^{\otimes n}$. Following the existing nomenclature, we refer to the structure (or template) of the parameterized quantum circuit as *ansatz*. In our QCNN architecture, $U_i$ always consists of two-qubit quantum circuit blocks, and the convolution and pooling part each uses the identical quantum circuit blocks within the given layer. Since a two-qubit gate requires 15 parameters at most (Vatan and Williams 2004), in $i$th layer consisting of $l_i > 0$ independent convolutional filter and one pooling operation the maximum number of parameters subject to optimization is $15(l_i + 1)$. Then, the total number of parameters is at most $15\sum_{i=1}^{\log_2(n)}(l_i + 1)$ if the convolution and pooling operations are iterated until only one qubit remains. One can also consider an interesting hybrid architecture in which the QCNN layers are stacked until $m$ qubits are remaining and then a classical neural network takes over from the $m$ qubit measurement outcomes. In this case, the number of quantum circuit parameters is less than the maximum number given above. Usually, $l_i$ is set to be a constant. Therefore, the number of parameters subject to optimization grows as $O(\log(n))$, which is an exponential reduction compared to the general hierarchical structure discussed in Ref. Grant et al. (2018). This also implies that the number of parameters can be suppressed double-exponentially with the size of classical data if the exponentially large state space is fully utilized for encoding the classical data. An example quantum circuit for a QCNN algorithm with eight qubits for binary classification is depicted in Fig. 1. Generalizing Fig. 1 to larger systems can be done simply by connecting all neighboring qubits with the two-qubit parameterized gates in the translationally invariant way.

The optimization of the gate parameters can be carried out by iteratively updating the parameters based on the gradient of the cost function until some condition for the termination is reached. The cost function gradient can be calculated classically or by using a quantum computer via *parameter-shift rule* (Li et al. 2017; Mitarai et al. 2018; Schuld et al. 2019). When the parameter-shift rule is used, QCNN requires an exponentially smaller number of quantum circuit executions compared to the general hierarchical structures inspired by tensor networks (e.g., tree tensor network) in Ref. Grant et al. (2018). While the latter uses $O(n)$ runs, the former only uses $O(\log(n))$ runs.

## 2.3 Quantum data encoding

Many machine learning techniques transform input data $\mathcal{X}$ into a different space to make it easier to work with. This transformation $\phi : \mathcal{X} \to \mathcal{X}'$ is often called a feature map. In quantum computing, the same analogy can be applied to perform a *quantum feature map*, which acts as $\phi : \mathcal{X} \to \mathcal{H}$ where the vector space $\mathcal{H}$ is a Hilbert space (Schuld and Killoran 2019). In fact, such feature mapping is mandatory when one uses quantum machine learning on classical data, since classical data must be encoded as a quantum state (Rebentrost et al. 2014; Lloyd et al. 2014; Giovannetti et al. 2008; Park et al. 2019; Veras et al. 2020; Araujo et al. 2021). The quantum feature map $x \in \mathcal{X} \to |\phi(x)\rangle \in \mathcal{H}$ is equivalent to applying a unitary transformation $U_\phi(x)$ to the initial state $|0\rangle^{\otimes n}$ to produce $U_\phi(x)|0\rangle^{\otimes n} = |\phi(x)\rangle$, where $n$ is the number of qubits. This refers to the green rectangle in Fig. 1.

There exist numerous structures of $U_\phi(x)$ to encode the classical input data $x$ into a quantum state. In this work, we benchmark the performance of the QCNN algorithm with several different quantum data encoding techniques. These techniques are explained in detail in this section.

### 2.3.1 Amplitude encoding

One of the most general approaches to encode classical data as a quantum state is to associate normalized input data with probability amplitudes of a quantum state. This encoding scheme is known as the amplitude encoding (AE). The amplitude encoding represents input data of $x = (x_1, ..., x_N)^T$ of dimension $N = 2^n$ as amplitudes of an $n$-qubit quantum state $|\phi(x)\rangle$ as

$$U_\phi(x) : x \in \mathbb{R}^N \to |\phi(x)\rangle = \frac{1}{\|x\|}\sum_{i=1}^{N} x_i|i\rangle, \quad (2)$$

where $|i\rangle$ is the $i$th computational basis state. Clearly, with amplitude encoding, a quantum computer can represent exponentially many classical data. This can be of great advantage in QCNN algorithms. Since the number of parameters subject to optimization scales as $O(\log(n))$

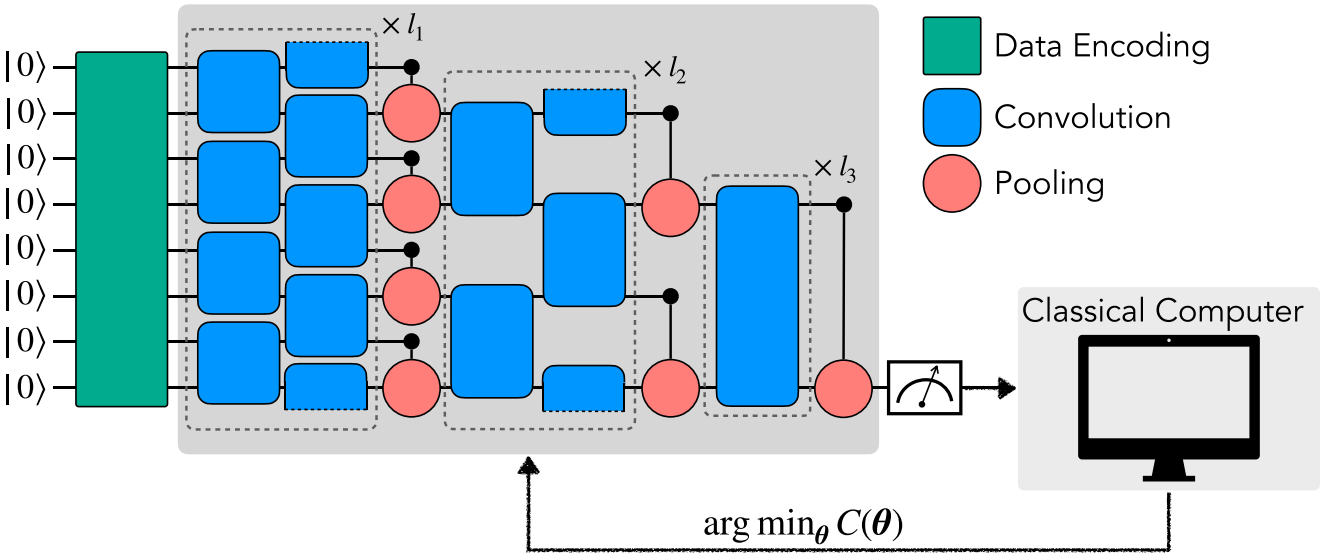



**Fig. 1** A schematic of the QCNN algorithm for an example of 8 input qubits. The quantum circuit consists of three parts: quantum data encoding (green rectangle), convolutional filters (blue rounded rectangle), and pooling (red circle). The quantum data encoding is fixed in a given structure of QCNN, while the convolutional filter and pooling use parameterized quantum gates. There are three layers in this example, and in each layer, multiple convolutional filters can be applied. The number of filters for $i$th layer is denoted by $l_i$. In each layer, the convolutional filter applies the same two-qubit ansatz to nearest neighbor qubits in a translationally invariant way. Similarly, pooling operations within the layer use the same ansatz. In this example, the pooling operation is represented as a controlled unitary transformation, which is activated when the control qubit is 1. However, general controlled operations can also be considered. The measurement outcome of the quantum circuit is used to calculate the user-defined cost function. The classical computer is used to compute the new set of parameters based on the gradient, and the quantum circuit parameters are updated accordingly for the subsequent round

(see Section 2.2), the amplitude encoding reduces the number of parameters doubly-exponentially with the size (i.e., dimension) of the classical data. However, the quantum circuit depth for amplitude encoding usually grows as $O(poly(N))$, although there exists a method to reduce it to $O(\log(N))$ at the cost of increasing the number of qubits to $O(N)$ (Araujo et al. 2021).

#### 2.3.2 Qubit encoding

The computational overhead of amplitude encoding motivates qubit encoding, which uses a constant quantum circuit depth while using $O(N)$ number of qubits. The qubit encoding embeds one classical data point $x_i$, that is rescaled to lie between 0 and $\pi$, into a single qubit as $|\phi(x_i)\rangle = \cos(\frac{x_i}{2})|0\rangle + \sin(\frac{x_i}{2})|1\rangle$ for $i = 1, ..., N$. Hence, the qubit encoding maps input data of $x = (x_1, \ldots, x_N)^T$ to $N$ qubits as

$$\begin{aligned} &U_\phi(x) : x \in \mathbb{R}^N \rightarrow |\phi(x)\rangle \\ &= \bigotimes_{i=1}^{N} (\cos(\frac{x_i}{2})|0\rangle + \sin(\frac{x_i}{2})|1\rangle), \end{aligned} \tag{3}$$

where $x_i \in [0, \pi)$ for all $i$. The encoding circuit can be expressed with a unitary operator $U_\phi(x) = \bigotimes_{j=1}^{N} U_{x_j}$ where

$$U_{x_j} = e^{-i\frac{x_j}{2}\sigma_y} := \begin{bmatrix} \cos(\frac{x_j}{2}) & -\sin(\frac{x_j}{2}) \\ \sin(\frac{x_j}{2}) & \cos(\frac{x_j}{2}) \end{bmatrix}.$$

#### 2.3.3 Dense qubit encoding

In principle, since a quantum state of one qubit can be described with two real-valued parameters, two classical data points can be encoded in one qubit. Thus, the qubit encoding described above can be generalized to encode two classical vectors per qubit by using rotations around two orthogonal axes (LaRose and Coyle 2020). By choosing them to be the $x$ and $y$ axes of the Bloch sphere, this method, which we refer to as dense qubit encoding, encodes $x_j = (x_{j_1}, x_{j_2})$ into a single qubit as

$$|\phi(x_j)\rangle = e^{-i\frac{x_{j_2}}{2}\sigma_y} e^{-i\frac{x_{j_1}}{2}\sigma_x} |0\rangle.$$

Hence, the dense qubit encoding maps an $N$-dimensional input data $x = (x_1, \ldots, x_N)^T$ to $N/2$ qubits as

$$\begin{aligned} &U_\phi(x) : x \in \mathbb{R}^N \rightarrow |\phi(x)\rangle \\ &= \bigotimes_{j=1}^{N/2} \left( e^{-i\frac{x_{N/2+j}}{2}\sigma_y} e^{-i\frac{x_j}{2}\sigma_x} |0\rangle \right). \end{aligned} \tag{4}$$

Note that there is freedom to choose which pair of classical data to be encoded in one qubit. In this work, we chose the pairing as shown in Eq. (4), but one may choose to encode $x_{2j-1}$ and $x_{2j}$ in $j$th qubit.

#### 2.3.4 Hybrid encoding

As shown in previous sections, the amplitude encoding is advantageous when the quantum circuit width (i.e., the number of qubits) is considered while the qubit encoding is advantageous when the quantum circuit depth is considered. These two encoding schemes represent the extreme ends of the quantum circuit complexities for loading classical data into a quantum system. In this section, we introduce simple hybrid encoding methods to compromise the quantum circuit complexity between these two extreme ends. In essence, the hybrid encoding implements the amplitude encoding to a number of independent blocks of qubits in parallel. Let us denote the number of qubits in each independent block that amplitude-encodes classical data by $m$. Then, each block can encode $O(2^m)$ classical data. Let us also denote that there are $b$ such blocks of $m$ qubits by $b$. Then, the quantum system of $b$ blocks contains $b2^m$ classical data. The first hybrid encoding, which we refer to as hybrid direct encoding (HDE), can be expressed as

$$U_\phi(x) : x \in \mathbb{R}^N \rightarrow |\phi(x)\rangle = \bigotimes_{j=1}^{b} \left( \frac{1}{\|x\|_j} \sum_{i=1}^{2^m} x_{ij} |i\rangle_j \right). \quad (5)$$

Note that each block can have a different normalization constant, and hence, the amplitudes may not be a faithful representation of the data unless the normalization constant have similar values. To circumvent this problem, we also introduce hybrid angle encoding (HAE), which can be expressed as

$$|\phi(x)\rangle = \bigotimes_{k=1}^{b} \left( \sum_{i=1}^{2^m} \prod_{j=0}^{m-1} \cos^{1-\mathtt{i}_j}\left(x_{g(j),k}\right) \sin^{\mathtt{i}_j}\left(x_{g(j),k}\right) |i\rangle_k \right), \quad (6)$$

where $\mathtt{i} \in \{0, 1\}^m$ is the binary representation of $i$ with $\mathtt{i}_j$ being the $j+1$th bit of the bit string, $x_{j,k}$ represents the $j$th element of the data assigned to the $k$th block of qubits, and $g(j) = 2^j + \sum_{l=0}^{j-1} \mathtt{i}_l 2^l$. In this case, having $b$ block of $m$ qubits allows $b(2^m - 1)$ classical data to be encoded. The performance of these hybrid encoding methods will be compared in Section 4.

Since the hybrid methods are parallelized, the quantum circuit depth is reduced to $O(2^m)$ where $m < N$, while the number of qubits is $O(mN/2^m)$. Therefore, the hybrid encoding algorithms use fewer number of qubits than the qubit encoding and use shallower quantum circuit depth than the amplitude encoding. Finding the best trade-off between the quantum circuit width and depth (i.e., the choice of $m$) depends on the specific details of given quantum hardware.

## 3 Benchmark variables

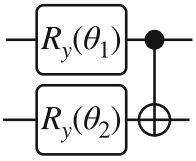
### 3.1 Ansatz

An important step in the construction of a QCNN model is the choice of ansatz. In general, the QCNN structure is flexible to use an arbitrary two-qubit unitary operation at each convolutional filter and each pooling step. However, we constrain our design such that all convolutional filters use the same ansatz, and the same applies to all pooling operations (but differ from convolutional filters). We later show that the QCNN with fixed ansatz provides excellent results for the benchmarking datasets. While using different ansatz for all filters can be an interesting attempt for further improvements, this will increase the number of parameters to be optimized.

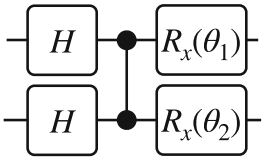
In the following, we introduce a set of convolutional and pooling ansatz (i.e., parameterized quantum circuit templates) used in our QCNN models.

#### 3.1.1 Convolution filter

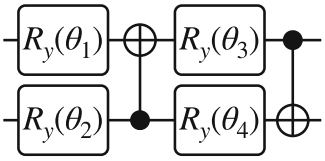
Parameterized quantum circuits for convolutional layers in QCNN are composed of different configurations of single-qubit and two-qubit gate operations. Most circuit diagrams in Fig. 2 are inspired by past studies. For instance, circuit 1 is used as the parameterized quantum circuit for training a tree tensor network (TTN) (Grant et al. 2018). Circuits 2, 3, 4, 5, 7, and 8 are taken from the work by Sim et al. (2019) which includes the analysis on expressibility and entangling capability of four-qubit parameterized quantum circuits. We modified these quantum circuits to two-qubit forms to utilize them as building blocks of the convolutional layer, which always consists of two qubits. Circuits 7 and 8 are reduced versions of circuits that recorded the best expressibility in the study. Circuit 2 is a two-qubit version of the quantum circuit that exhibited the best entangling capability. Circuits 3, 4 and 5 are drawn from circuits that have balanced significance in both expressibility and entangling capability. Circuit 6 is developed as a proper candidate of two-body Variational Quantum Eigensolver (VQE) entangler in Ref. Parrish et al. (2019). This circuit is also known to be able to implement an arbitrary $SO(4)$ gate (Wei and Di 2012). In fact, a total VQE entangler can be constructed by linearly arranging the $SO(4)$ gates throughout input qubits. Since this structure is similar to the structure of convolutional layers in QCNN, the $SO(4)$ gate would be a great candidate to be used in the convolution layer. Circuit 9 represents the parameterization of an arbitrary $SU(4)$ gate (Vatan and Williams 2004; MacCormack et al. 2020).



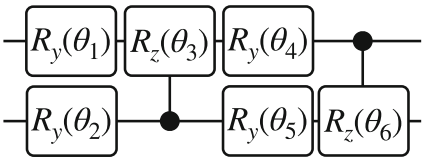

(a) Convolutional circuit 1



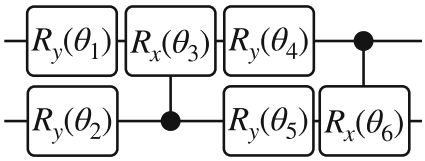

(b) Convolutional circuit 2



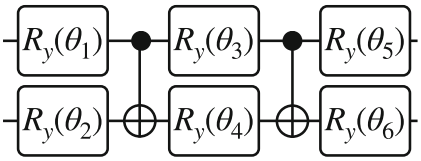

(c) Convolutional circuit 3



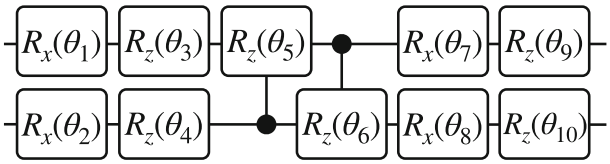

(d) Convolutional circuit 4



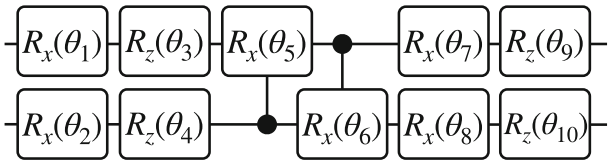

(e) Convolutional circuit 5



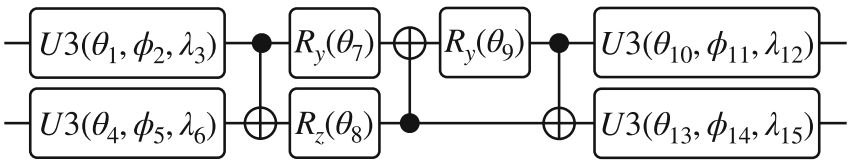

(f) Convolutional circuit 6




(g) Convolutional circuit 7




(h) Convolutional circuit 8




(i) Convolutional circuit 9

**Fig. 2** Parameterized quantum circuits used in the convolutional layer. $R_i(\theta)$ is a rotation around the $i$ axis of the Bloch sphere by an angle of $\theta$, and $H$ is the Hadamard gate. $U3(\theta, \phi, \lambda)$ is an arbitrary single-qubit gate that can be expressed as $U3(\theta, \phi, \lambda) = R_z(\phi)R_x(-\pi/2)R_z(\theta)R_x(\pi/2)R_z(\lambda)$. (a) Convolutional circuit 1. (b) Convolutional circuit 2. (c) Convolutional circuit 3. (d) Convolutional circuit 4. (e) Convolutional circuit 5. (f) Convolutional circuit 6. (g) Convolutional circuit 7. (h) Convolutional circuit 8. (i) Convolutional circuit 9

#### 3.1.2 Pooling

The pooling layer applies parameterized quantum gates to two qubits and traces out one of the qubits to reduce the two-qubit states to one-qubit states. Similar to the choice of ansatz for the convolutional filter, there exists a variety of choices of two-qubit circuits of the pooling layer. In this work, we choose a simple form of a two-qubit circuit consisting of two free parameters for the pooling layer. The circuit is shown in Fig. 3.

Application of the parameterized gates in the pooling step in conjunction with the convolutional circuit 9 might be redundant since it is already an arbitrary $SU(4)$ gate. Thus, for the convolutional circuit 9, we test two QCNN constructions, with and without the parameterized two-qubit circuit in the pooling layer. In the latter, the pooling layer only consists of tracing out one qubit.

### 3.2 Cost function

The variational parameters of the ansatz are updated to minimize the cost function calculated on the training dataset. In this benchmark study, we test the performance of QCNN

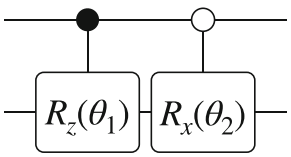



**Fig. 3** Parameterized quantum circuit used in the pooling layer. The pooling layer applies two controlled rotations $R_z(\theta_1)$ and $R_x(\theta_2)$, respectively, each activated when the control qubit is 1 (filled circle) or 0 (open circle). The control (first) qubit is traced out after the gate operations in order to reduce the dimension

models with two different cost functions, namely the mean squared error and the cross-entropy loss.

#### 3.2.1 Mean squared error

Before training QCNN, we map original class labels of $\{0, 1\}$ to $\{1, -1\}$ respectively to associate them with the eigenvalues of the qubit observables. Then, the mean squared error (MSE) between predictions and class labels becomes

$$C(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{M}_z(\psi_i(\boldsymbol{\theta})) - \tilde{y}_i)^2, \tag{7}$$

where $\hat{M}_z(\psi_i) = \langle\psi_i|\sigma_z|\psi_i\rangle$ is the Pauli-Z expectation value of one-qubit state extracted from QCNN for $i$th training data, and $\tilde{y}_i \in \{1, -1\}$ is the label of the corresponding training data (i.e., $\tilde{y}_i = 1 - 2y_i$). Since QCNN performs a single-qubit measurement in the $Z$ basis, the final state can be thought of as a mixed state $a_i|0\rangle\langle 0| + b_i|1\rangle\langle 1|$. Then, minimizing the cost function above with respect to $\boldsymbol{\theta}$ would correspond to forcing $a_i$ to be as larger as possible than $b_i$ if the $i$th training data is labelled 0, and vice versa if it is labelled 1.

#### 3.2.2 Cross-entropy loss

Cross-entropy loss is widely used in training classical neural networks. It measures the performance of a classification model whose output is a probability between 0 and 1. Due to the probabilistic property of quantum mechanics, one could consider the cross-entropy loss by considering probabilities of measuring computational basis states in the single-qubit measurement of QCNN. The cross-entropy loss for the $i$th training data can be expressed as

$$\begin{aligned} C(\boldsymbol{\theta}) = &\sum_{i=1}^{N} y_i \log\left(\Pr[\psi_i(\boldsymbol{\theta}) = 1]\right) \\ &+(1 - y_i)\log\left(\Pr[\psi_i(\boldsymbol{\theta}) = 0]\right), \end{aligned} \tag{8}$$

where $y_i \in \{0, 1\}$ is the class label and $\Pr[\psi_i(\boldsymbol{\theta}) = y_i]$ is the probability of measuring the computational basis state $|y_i\rangle$ from the QCNN circuit.

### 3.3 Classical data pre-processing

The size of quantum circuits that can be reliably executed on NISQ devices is limited due to the noise and technical challenges of building quantum hardware. Thus, the encoding schemes for high-dimensional data usually require the number of qubits that are beyond the current capabilities of quantum devices. Therefore, classical dimensionality reduction techniques will be useful in the near-term application of quantum machine learning techniques. In this work, we pre-process data with three classical dimensionality reduction techniques, namely bilinear interpolation, principal component analysis (PCA) (Jolliffe 2002) and autoencoding (AutoEnc) (Goodfellow et al. 2016). For the simulation presented in the following section, amplitude encoding is used only with bilinear interpolation while all other encoding schemes are tested with PCA and autoencoding. Bilinear interpolation and PCA are carried out by utilizing `tf.image.resize` from TensorFlow and `sklearn.decomposition.PCA` from scikit-learn, respectively. Autoencoders are capable of modelling complex non-linear functions, while PCA is a simple linear transformation with cheaper and faster computation. Since the pre-processing step should not produce too much computational resource overhead or result in overfitting, we train a simple autoencoder with one hidden layer. The data in the latent space (i.e., hidden layer) is then fed to quantum circuits.

## 4 Simulation

### 4.1 QCNN results overview

This section reports the classical simulation results of the QCNN algorithm for binary classification carried out with Pennylane (Bergholm et al. 2020). The test is performed with two standard datasets, namely MNIST and Fashion MNIST, under various conditions as described in the previous section. Note that the MNIST and Fashion MNIST datasets are $28\times28$ image data, each with ten classes. Our benchmark focuses on binary classification, and hence, we select classes 0 and 1 for both datasets.

The variational parameters in the QCNN ansatze are optimized by minimizing the cost function with an optimizer provided in Pennylane (Bergholm et al. 2020). In particular, we tested Adam (Kingma and Ba 2017) and Nesterov moment (Nesterov 1983) optimization algorithms. At each iteration $t$, we create a small batch by randomly selecting data from the training set. Compared to training on the full dataset, training on the mini-batch not only reduces simulation time but also helps the gradients to escape from local minima. For both Adam and Nesterov moment optimizers, the batch size was 25 and the learning rate was 0.01. We also fixed the number of iterations to be 200 to speed up the training process. Note that the training can alternatively be stopped at different conditions, such as when the validation set accuracy does not increase for a predetermined number of consecutive runs (Grant et al. 2018). The numbers of training (test) data are 12665 (2115) and 12000 (2000) for MNIST and fashion MNIST datasets, respectively.

Tables 1 and 2 show the mean classification accuracy and one standard deviation obtained from five instances with random initialization of parameters. The number of random initialization is chosen to be the same as that of Ref. Grant et al. (2018). The results are obtained for various QCNN models of different convolutional and pooling circuits and data encoding strategies. When benchmarking with the hybrid encoding schemes (i.e., HDE and HAE), we used two blocks of four qubits, which results in having 32 and 30 features encoded in 8 qubits, respectively. For all results presented here, training is done with the cross-entropy loss. Similar results are obtained when MSE is used for training, and we present the MSE results in Appendix C. Here we only report the classification results obtained with the Nesterov optimizer, since it consistently provided better convergence. The ansatze in the table are listed in the same order as the list of convolutional circuits shown in Fig. 2. The last row of the table (i.e., Ansatz 9b) contains the results when the QCNN circuit only consists of the convolutional circuit 9 without any unitary gates in the pooling step.

The simulation results show that all ansatze perform reasonably well, while the ones with more number of free parameters tend to produce higher score. Since all ansatze perform reasonably well, one may choose to use the ansatz with smaller number of free parameters to save the training time. For example, by choosing ansatz 4 and amplitude encoding, one can achieve 97.8% classification accuracy while using only 24 free parameters total instead of achieving 98.4% at the cost of increasing the number of free parameters to 51. It is also important to note that most of the results obtained with the hybrid data encoding and PCA is considerably worse than the others. This is due to the normalization problem discussed in Section 2.3.4, which motivated the development of the hybrid angle encoding. We observed that the normalization problem is negligible in the case with autoencoding. The simulation results clearly demonstrates that the normalization problem is resolved by the hybrid angle encoding as expected, and reasonably good results can be obtained with this method. For MNIST, HAE with PCA provides the best solution among the hybrid

**Table 1** Mean accuracy and one standard deviation of the classification for 0 and 1 in the MNIST dataset when the model is trained with the cross-entropy loss

| | | Classification accuracy | | | | |
|---|---|---|---|---|---|---|
| Ansatz | # of params | Amplitude | Qubit | Dense | HDE | HAE |
| 1 | 12 | $96.8 \pm 5.3$ | $98.0 \pm 0.4$ | $97.6 \pm 1.1$ | $68.7 \pm 5.1$ | $97.9 \pm 0.3$ |
| | | | $91.4 \pm 2.3$ | $88.4 \pm 9.2$ | $88.4 \pm 2.6$ | $77.7 \pm 6.0$ |
| 2 | 12 | $94.5 \pm 3.1$ | $98.2 \pm 4.5$ | $98.2 \pm 0.5$ | $62.2 \pm 3.2$ | $94.7 \pm 2.1$ |
| | | | $98.2 \pm 6.6$ | $85.6 \pm 4.5$ | $93.4 \pm 5.4$ | $80.0 \pm 4.0$ |
| 3 | 18 | $93.8 \pm 4.4$ | $\mathbf{98.5} \pm 0.2$ | $96.9 \pm 1.6$ | $76.4 \pm 2.7$ | $98.1 \pm 0.2$ |
| | | | $93.3 \pm 3.8$ | $95.8 \pm 1.7$ | $95.3 \pm 3.4$ | $84.4 \pm 0.7$ |
| 4 | 24 | $97.8 \pm 2.4$ | $98.2 \pm 0.4$ | $98.2 \pm 0.4$ | $70.2 \pm 1.3$ | $98.0 \pm 0.3$ |
| | | | $\mathbf{98.5} \pm 1.2$ | $\mathbf{97.2} \pm 1.1$ | $96.6 \pm 1.0$ | $\mathbf{90.4} \pm 3.8$ |
| 5 | 24 | $96.7 \pm 2.1$ | $98.3 \pm 0.4$ | $98.1 \pm 0.5$ | $72.6 \pm 5.7$ | $98.0 \pm 0.1$ |
| | | | $94.9 \pm 2.1$ | $96.0 \pm 1.3$ | $93.5 \pm 1.3$ | $86.5 \pm 5.0$ |
| 6 | 24 | $97.2 \pm 2.2$ | $98.1 \pm 0.4$ | $98.1 \pm 0.3$ | $77.4 \pm 1.7$ | $\mathbf{98.3} \pm 0.2$ |
| | | | $97.7 \pm 1.0$ | $93.4 \pm 0.5$ | $97.0 \pm 2.0$ | $86.9 \pm 7.3$ |
| 7 | 36 | $98.3 \pm 2.2$ | $98.2 \pm 0.3$ | $\mathbf{98.7} \pm 2.4$ | $74.6 \pm 3.2$ | $98.2 \pm 0.1$ |
| | | | $93.7 \pm 4.5$ | $95.1 \pm 1.6$ | $97.2 \pm 2.2$ | $90.2 \pm 3.0$ |
| 8 | 36 | $98.1 \pm 0.7$ | $98.3 \pm 0.4$ | $\mathbf{98.7} \pm 0.1$ | $\mathbf{79.7} \pm 1.6$ | $\mathbf{98.3} \pm 0.1$ |
| | | | $96.9 \pm 2.4$ | $95.4 \pm 2.8$ | $96.6 \pm 1.7$ | $89.1 \pm 2.6$ |
| 9a | 51 | $\mathbf{98.4} \pm 0.2$ | $98.4 \pm 0.5$ | $\mathbf{98.7} \pm 0.4$ | $78.1 \pm 2.8$ | $98.2 \pm 0.2$ |
| | | | $96.4 \pm 2.3$ | $96.7 \pm 1.4$ | $97.8 \pm 2.2$ | $87.0 \pm 5.3$ |
| 9b | 45 | $98.3 \pm 0.2$ | $97.7 \pm 0.6$ | $98.3 \pm 0.5$ | $77.4 \pm 2.5$ | $98.1 \pm 0.1$ |
| | | | $96.6 \pm 2.2$ | $96.5 \pm 1.7$ | $\mathbf{98.0} \pm 1.2$ | $88.5 \pm 2.8$ |

The mean and the standard deviation are obtained from five repetitions with random initialization of parameters. The first column shows the ansatz label. The second column shows the total number of parameters that are subject to optimization. For qubit, dense and hybrid encoding, two rows indicate the values obtained with different classical data pre-processing, namely PCA and autoencoding, respectively. The best result under each quantum data encoding method is written in bold

**Table 2** Mean accuracy and one standard deviation of the classification for 0 (t-shirt/top) and 1 (trouser) in the Fashion MNIST dataset when the model is trained with the cross-entropy loss

| | | Classification accuracy | | | | |
|---|---|---|---|---|---|---|
| Ansatz | # of params | Amplitude | Qubit | Dense | HDE | HAE |
| 1 | 12 | $90.9 \pm 2.0$ | $83.1 \pm 3.3$ | $85.0 \pm 3.6$ | $64.4 \pm 3.2$ | $83.9 \pm 1.7$ |
| | | | $87.3 \pm 4.2$ | $84.7 \pm 7.2$ | $90.5 \pm 1.3$ | $82.7 \pm 4.6$ |
| 2 | 12 | $88.2 \pm 3.8$ | $87.2 \pm 4.6$ | $82.2 \pm 2.1$ | $63.1 \pm 1.4$ | $84.3 \pm 4.1$ |
| | | | $86.6 \pm 4.6$ | $86.9 \pm 7.7$ | $86.0 \pm 4.7$ | $82.1 \pm 5.0$ |
| 3 | 18 | $90.1 \pm 2.7$ | $87.7 \pm 3.6$ | $87.2 \pm 3.0$ | $65.5 \pm 1.3$ | $85.5 \pm 1.2$ |
| | | | $88.4 \pm 7.4$ | $88.8 \pm 2.2$ | $91.7 \pm 1.5$ | $88.1 \pm 4.0$ |
| 4 | 24 | $89.1 \pm 2.2$ | $87.2 \pm 2.4$ | $89.7 \pm 1.6$ | $64.7 \pm 1.9$ | $84.7 \pm 1.1$ |
| | | | $91.6 \pm 3.4$ | $91.8 \pm 1.7$ | $91.2 \pm 0.5$ | $88.4 \pm 4.4$ |
| 5 | 24 | $90.7 \pm 1.1$ | $86.3 \pm 2.9$ | $87.6 \pm 2.3$ | $64.1 \pm 2.4$ | $84.5 \pm 1.7$ |
| | | | $91.9 \pm 1.4$ | **93.7** $\pm 1.4$ | $92.4 \pm 1.3$ | $85.9 \pm 3.9$ |
| 6 | 24 | $90.4 \pm 1.7$ | $87.9 \pm 3.9$ | $88.7 \pm 2.4$ | $65.7 \pm 0.9$ | $87.7 \pm 2.6$ |
| | | | $93.6 \pm 1.6$ | $90.2 \pm 1.9$ | $94.3 \pm 1.2$ | $88.5 \pm 2.6$ |
| 7 | 36 | $88.2 \pm 1.4$ | $87.1 \pm 3.8$ | $88.7 \pm 2.9$ | $66.3 \pm 0.9$ | $86.7 \pm 1.8$ |
| | | | $92.2 \pm 0.5$ | $92.7 \pm 2.2$ | $93.8 \pm 0.8$ | $89.7 \pm 2.9$ |
| 8 | 36 | $89.9 \pm 1.9$ | **89.8** $\pm 0.9$ | $88.7 \pm 2.6$ | **66.5** $\pm 0.3$ | $86.2 \pm 1.5$ |
| | | | $91.6 \pm 3.0$ | $92.2 \pm 2.6$ | $93.2 \pm 1.5$ | **91.6** $\pm 3.6$ |
| 9a | 51 | **91.3** $\pm 2.3$ | $88.9 \pm 2.7$ | $89.2 \pm 2.0$ | $65.0 \pm 1.3$ | $88.6 \pm 2.0$ |
| | | | $92.4 \pm 2.6$ | $92.1 \pm 1.3$ | **94.3** $\pm 1.6$ | $88.3 \pm 3.7$ |
| 9b | 45 | $88.8 \pm 1.6$ | $88.0 \pm 2.5$ | **90.2** $\pm 0.5$ | $64.7 \pm 2.8$ | **88.9** $\pm 1.6$ |
| | | | **94.1** $\pm 1.1$ | $92.7 \pm 1.5$ | $92.9 \pm 1.3$ | $87.1 \pm 3.6$ |

The mean and the standard deviation are obtained from five repetitions with random initialization of parameters. The first column shows the ansatz label. The second column shows the total number of parameters that are subject to optimization. For qubit, dense and hybrid encoding, two rows indicate the values obtained with different classical data pre-processing, namely PCA and autoencoding, respectively. The best result under each quantum data encoding method is written in bold

encoding schemes on average. On the other hand, for Fashion MNIST, HDE with autoencoding provides the best solution among the hybrid encoding schemes on average.

In the following, we also compare the two classical dimensionality reduction methods by presenting the overall mean accuracy and standard deviation values obtained by averaging over all ansatze and the random initializations. The average values are presented in Table 3. As discussed before, the HDE with PCA does not perform well for both datasets due to the data normalization issue. Besides this case, interestingly, PCA works better than autoencoding for MNIST data, and vice versa for Fashion MNIST data, thereby suggesting that the choice of the classical pre-processing method should be data-dependent.

Finally, we also examine how classification performance improves as the number of convolutional filters in each layer increases. For simplicity, we set the number of convolutional filters in each layer to be same, i.e., $l_1 = l_2 = l_3 = L$ (see Fig. 1 for the definition of $l_i$). Without loss of generality, we pick two ansatze and five encodings. For ansatze, we choose the one with the smallest number of free parameters and another with arbitrary $SU(4)$ operations. These are circuit 2 and circuit 9b, and they use 12 and 45 parameters total, respectively. For data encoding, we tested amplitude, qubit, and dense encoding. The qubit and dense encoding are further grouped under two different classical dimensionality reduction techniques, PCA and autoencoding. Since the qubit and dense encoding load 8 and 16 features, respectively, we label them as PCA8, AutoEnc8, PCA16, and AutoEnc16 based on the number of features and the dimensionality reduction techniques. The classification accuracies for $L = \{1, 2, 3\}$ are plotted in Fig. 4. The simulation results show that in some cases, the classification accuracy can be improved by increasing the number of convolutional filters. For example, the classification accuracy for MNIST data can be improved from about 86% to 96% when circuit 2 and dense encoding with autoencoding are used. For Fashion MNIST, the classification accuracy is improved from about 88% to 90% when circuit 2 and amplitude encoding are used, and from about 86% to 90% when circuit 2 and qubit encoding with PCA is used. However, we do not observe general trend with respect to the number of convolutional filters. In particular, the relationship between the classification accuracy and $L$ is

**Table 3** Comparison of the classical dimensionality reduction methods for angle encoding, dense encoding, and hybrid encoding

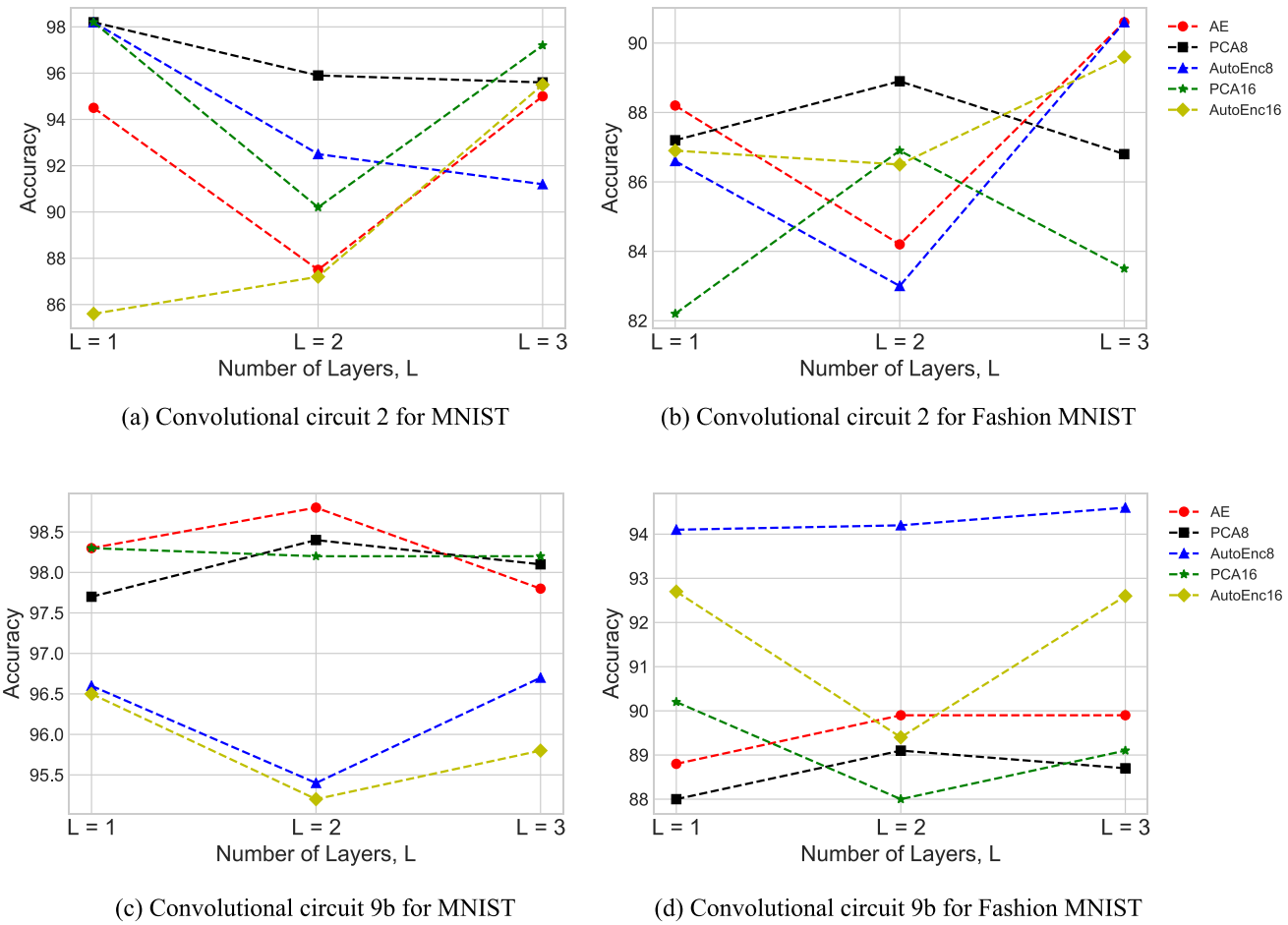
| | | Qubit | Dense | HDE | HAE |
|---|---|---|---|---|---|
| MNIST | PCA | $98.0 \pm 1.5$ | $98.0 \pm 1.0$ | $73.7 \pm 3.0$ | $97.8 \pm 0.4$ |
| | AutoEnc | $95.4 \pm 3.6$ | $94.2 \pm 4.6$ | $95.4 \pm 2.3$ | $86.1 \pm 4.1$ |
| Fashion MNIST | PCA | $87.3 \pm 3.7$ | $87.7 \pm 3.2$ | $65.0 \pm 1.6$ | $86.1 \pm 1.9$ |
| | AutoEnc | $91.0 \pm 4.1$ | $90.6 \pm 4.4$ | $92.0 \pm 1.5$ | $87.2 \pm 3.8$ |

For each encoding scheme, classification results from all instances (i.e., various ansatze and random initialization of parameters) are averaged out to produce the mean and standard deviation

less obvious for circuit 9b. We speculate that this attributes to the fact that circuit 9b implements an arbitrary $SU(4)$, which is an arbitrary two-qubit gate, and hence, repetitive application of an arbitrary $SU(4)$ is redundant.

## 4.2 Boundary conditions of the QCNN circuit

The general structure of QCNN shown in Fig. 1 uses two-qubit gates between the first (top) and last (bottom) qubits, which can be thought of as imposing periodic-boundary condition. One may notice that all-to-all connectivity can be established even without connecting the boundaries. Thus, we tested the classification performance of a QCNN architecture without having the two-qubit gates to close the loop. We refer to this case as the open-boundary QCNN. Without loss of generality, we tested QCNNs with two different ansatz, the convolutional circuit 2 (Ansatz 2 in Tables 1 and 2) which uses the smallest number




(a) Convolutional circuit 2 for MNIST

(b) Convolutional circuit 2 for Fashion MNIST

(c) Convolutional circuit 9b for MNIST

(d) Convolutional circuit 9b for Fashion MNIST

**Fig. 4** Classification accuracy vs the number of convolutional filters $L$ for MNIST and Fashion MNIST datasets. The number of filters in each layer is set to be equal, i.e., $l_1 = l_2 = l_3 = L$ (see Fig. 1 for the definition of $l_i$). The simulation is carried out with two ansatze, circuit 2 and circuit 9b, and five encoding schemes, amplitude encoding (AE), qubit encoding with PCA (PCA8) and with autoencoding (AutoEnc8), and dense encoding with PCA (PCA16) and with autoencoding (AutoEnc16). (a) Convolutional circuit 2 for MNIST. (b) Convolutional circuit 2 for Fashion MNIST. (c) Convolutional circuit 9b for MNIST. (d) Convolutional circuit 9b for Fashion MNIST

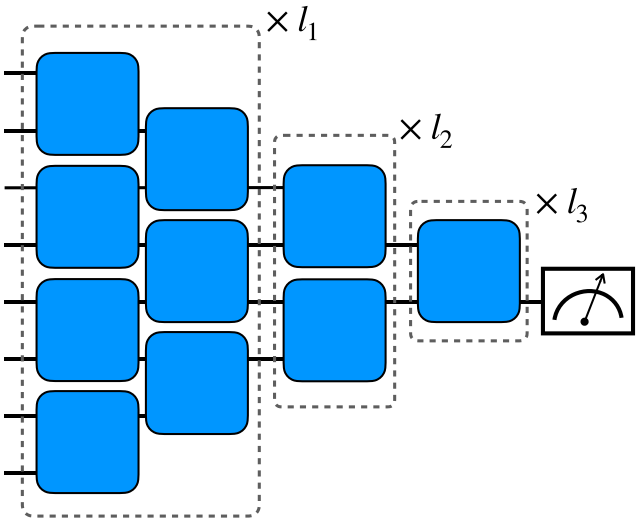



**Fig. 5** A QCNN circuit with the open-boundary condition and no gate operations for pooling. In this case, the QCNN circuit can be constructed with nearest neighbor qubit interactions only

of free parameters and the convolutional circuit 9 which implements an arbitrary $SU(4)$. In case of the latter, pooling was done without parameterized gates, and hence, the ansatz is equivalent to ansatz 9b in Tables 1 and 2. By imposing the open-boundary condition in conjunction with the ansatz 9b, one can modify the qubit arrangement of the QCNN circuit so as to use nearest neighbor qubit interactions only. For an example of 8-qubit QCNN circuit, the modified structure is depicted in Fig. 5. Such design is particularly advantageous for NISQ devices that have limited physical qubit connectivity. For example, if one employs the qubit or the dense encoding method, the QCNN algorithm can be implemented with a 1-dimensional chain of physical qubits.

The simulation results are presented in Table 4 for MNIST and Fashion MNIST datasets. These results are attained with one convolutional filters per layer, i.e., $l_1 = l_2 = l_3 = 1$. The simulation results demonstrate that for the case of two ansatze tested the classification performance between open- and periodic-boundary QCNN circuits are similar. Although the number of free parameters are the same under these conditions, depending on the specification of the quantum hardware such as the qubit connectivity, the open-boundary QCNN circuit can have shallower depth. The open-boundary circuit with ansatz 9b is even more attractive for NISQ devices since the convolutional operations can be done with only nearest neighbor qubit interactions as mentioned above.

### 4.3 Comparison to CNN

We now compare the classification results of QCNN to that of classical CNN. Our goal is to compare the classification accuracy of the two given a similar number of parameters subject to optimization. To make a fair comparison between the two, we fix all hyperparameters of the two methods to be the same, except we used the Adam optimizer for CNN since it performed significantly better than the Nesterov moment optimizer. A detailed description of the classical CNN architecture is provided in Appendix B.

It is important to note that CNN can be trained with such small number of parameters effectively only when the number of nodes in the input layer is small. Therefore, the CNN results are only comparable to that of the qubit and dense encoding cases which requires 8 and 16 classical input nodes, respectively. We designed four different CNN models with the number of free parameters being 26, 34, 44 and 56 to make them comparable to the QCNN models. In these cases, a dimensionality reduction technique must precede. For hybrid and amplitude encoding, which require relatively simpler data pre-processing, the number of nodes in the CNN input layer is too large to be trained with a small number of parameters as in QCNN.

Comparing the values in Table 5 with the QCNN results, one can see that QCNN models perform better than their corresponding CNN models for the MNIST dataset. The same conclusion also holds for the Fashion dataset, except for the CNN models with 44 and 56 parameters that achieve similar performance as their corresponding QCNN models. Another noticeable result is that the QCNN models have considerably smaller standard deviations than the CNN models on average. This implies that the QCNN models not only achieve higher classification accuracy than the CNN models under similar training conditions but also are less sensitive to the random initialization of the free parameters.

**Table 4** Mean classification accuracy and one standard deviation of the classification for 0 and 1 in the benchmarking datasets when the QCNN circuit is constructed under the open-boundary condition

| | Ansatz | AE | PCA8 | PCA16 | AutoEnc8 | AutoEnc16 |
|---|---|---|---|---|---|---|
| MNIST | 2 | $90.4 \pm 2.5$ | $98.0 \pm 0.3$ | $96.3 \pm 3.2$ | $97.7 \pm 0.1$ | $86.4 \pm 4.6$ |
| | 9b | $98.0 \pm 0.4$ | $98.1 \pm 0.2$ | $97.8 \pm 0.3$ | $96.8 \pm 0.7$ | $94.4 \pm 1.9$ |
| Fashion | 2 | $91.1 \pm 1.4$ | $86.2 \pm 0.5$ | $88.4 \pm 3.0$ | $83.3 \pm 4.6$ | $87.2 \pm 5.7$ |
| MNIST | 9b | $90.1 \pm 1.4$ | $87.6 \pm 2.4$ | $89.1 \pm 1.9$ | $91.9 \pm 1.4$ | $92.6 \pm 2.0$ |

Each column represents the results produced under a different encoding scheme with the numbers 8 and 16 indicates the qubit and dense encoding, respectively

**Table 5** Mean classification accuracy and one standard deviation obtained with classical CNN for classifying 0 and 1 in the MNIST and Fashion MNIST datasets

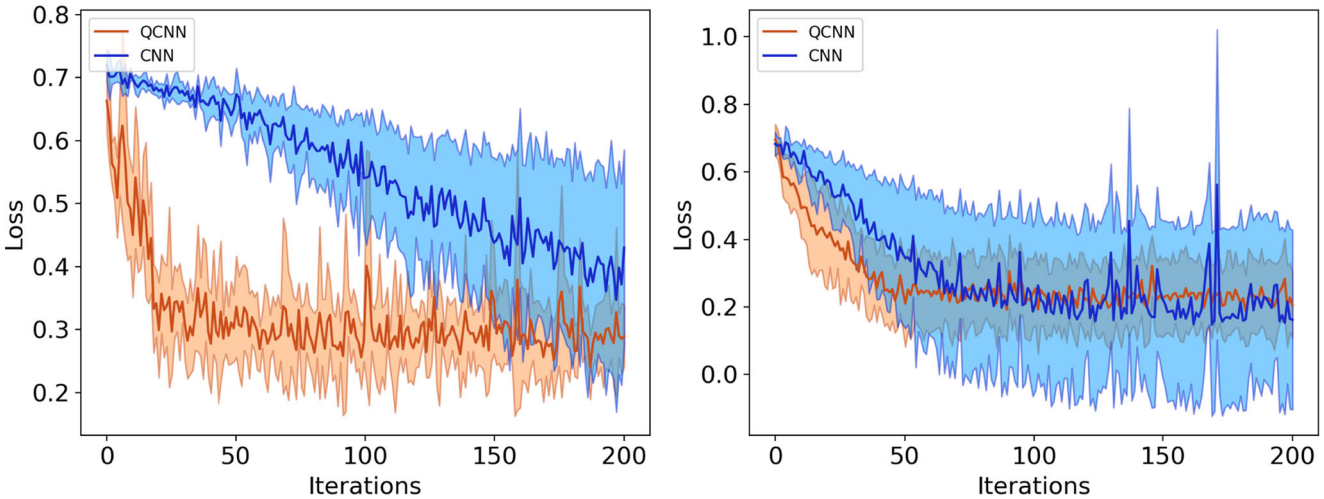
| | # of params | Input size | Classification accuracy | |
|---|---|---|---|---|
| | | | PCA | AutoEnc |
| MNIST | 26 | 8 | $91.0 \pm 12.7$ | $82.7 \pm 15.2$ |
| | 34 | 16 | $97.0 \pm 3.5$ | $83.5 \pm 15.5$ |
| | 44 | 8 | $93.3 \pm 13.2$ | $90.4 \pm 13.4$ |
| | 56 | 16 | $93.0 \pm 13.4$ | $95.5 \pm 2.3$ |
| Fashion MNIST | 26 | 8 | $82.2 \pm 16.6$ | $86.8 \pm 12.7$ |
| | 34 | 16 | $78.8 \pm 19.1$ | $79.0 \pm 19.0$ |
| | 44 | 8 | $89.4 \pm 3.9$ | $92.4 \pm 2.8$ |
| | 56 | 16 | $91.9 \pm 2.0$ | $93.6 \pm 2.2$ |

Each column is named with the pre-processing method (PCA or AutoEnc). These results directly compare to the second and third columns of Tables 1 and 2 denoted by Qubit and Dense

In Fig. 6, we present two representative examples of the cross-entropy loss as a function of the number of training iterations. For simplicity, we show such data for two cases in MNIST data classification: circuit 9b and qubit encoding with autoencoding, and circuit 9b and dense encoding with PCA. Considering the number of free parameters, these cases are comparable to the CNN models with 8 inputs with autoencoding and 16 inputs with PCA, respectively. Recall that the mean classification accuracy and one standard deviation in QCNN (CNN) is $96.6 \pm 2.2$ ($90.4 \pm 13.4$) for the first case, and $98.3 \pm 0.5$ ($93.0 \pm 13.4$) for the second case. Figure 6 shows that in both cases, the QCNN models are trained faster than the CNN models, while the advantage manifests more clearly in the first case. Furthermore, the standard deviations in the QCNN models are significantly smaller than that of the CNN models.

## 5 Conclusion

Fully parameterized quantum convolutional neural networks pave promising avenues for near-term applications of quantum machine learning and data science. This work presented an extensive benchmark of QCNN for solving classification problems on classical data, a fundamental task in pattern recognition. The QCNN algorithm can be tailored with many variables such as the structure of parameterized quantum circuits (i.e., ansatz) for convolutional




(a) 8-input CNN with AutoEnc vs QCNN

(b) 16-input CNN with PCA vs QCNN

**Fig. 6** Cross-entropy loss as a function of the number of training iterations. The QCNN models use circuit 9b as the ansatz. (a) The QCNN model with qubit encoding and autoencoding is compared to the CNN model with 8-inputs. (b) The QCNN model with dense encoding and PCA is compared to the CNN model with 16-inputs. (a) 8-input CNN with AutoEnc vs QCNN. (b) 16-input CNN with PCA vs QCNN

filters and pooling operators, quantum data encoding methods, classical data pre-processing methods, cost functions and optimizers. To improve the utility of QCNN for classical data, we also introduced new data encoding schemes, namely hybrid direct encoding and hybrid angle encoding, with which the exchange between quantum circuit depth and width for state preparation can be configured. With diverse combinations of the aforementioned variables, we tested 8-qubit QCNN models for binary classification of MNIST and Fashion MNIST datasets by simulation with Pennylane. The QCNN models tested in this work operated with a small number of free parameters, ranging from 12 to 51. Despite the small number of free parameters, QCNN produced high classification accuracy for all instances, with the best case being close to 99% for MNIST and 94% for Fashion MNIST. We also compared QCNN results to CNN and observed that QCNN performed noticeably better than CNN given the similar training conditions for both benchmarking datasets. The comparison between QCNN and CNN is only valid for qubit and dense encoding cases in which the number of input qubits grows linearly with the dimension of the input data. With amplitude or hybrid encoding, the number of input qubits is substantially smaller than the dimension of the data, and hence, there is no classical analogue. We speculate that the advantage of QCNN lies in the ability to exploit entanglement, which is a global effect, while CNN is only capable of capturing local correlations.

The QCNN architecture proposed in this work can be generalized for $L$-class classification through one-vs-one or one-vs-all strategies. It also remains an interesting future work to examine the construction of a multi-class classifier by leaving $\lceil \log_2(L) \rceil$ qubits for measurement in the output layer. Another interesting future work is to optimize the data encoding via training methods provided in Ref. Lloyd et al. (2020). However, since QCNN itself can be viewed as a feature reduction technique, it is not clear whether introducing another layer of the variational quantum circuit for data encoding would help until a thorough investigation is carried out. Understanding the underlying principle for the quantum advantage demonstrated in this work also remains to be done. One way to study this is by testing QCNN models with a set of data that does not exhibit local correlation but contains some global feature while analyzing the amount of entanglement created in the QCNN circuit. Since the circuit depth grows only logarithmically with the number of input qubits and the gate parameters are learned, the QCNN model is expected to be suitable for NISQ devices. However, the verification through real-world experiments and noisy simulations remains to be done. Furthermore, testing the classification performance as the QCNN models grow bigger remains an interesting future work. Finally, the application of the proposed QCNN algorithms for other real-world datasets such as those relevant to high-energy physics and medical diagnosis is of significant importance.

## Appendix A. Related works

The term *quantum convolutional neural network* (QCNN) appears in several places, but it refers to a number of different frameworks. Several proposals have been made in the past to reproduce classical CNN on a quantum circuit by imitating the basic arithmetic of the convolutional layer for a given filter (Kerenidis et al. 2019; Li et al. 2020; Wei et al. 2021). Although these algorithms have the potential to achieve exponential speedups against the classical counterpart in the asymptotic limit, they require an efficient means to implement quantum random access memory (QRAM), expensive subroutines such as the linear combination of unitaries or quantum phase estimation with extra qubits, and they work only for specific types of quantum data embedding. Another branch of CNN-inspired QML algorithms focuses on implementing the convolutional filter as a parameterized quantum circuit, which can be stacked by inserting a classical pooling layer in between (Liu et al. 2021; Henderson et al. 2020; Chen et al. 2020). Following the nomenclature provided in Henderson et al. (2020), we refer to this approach as *quanvolutioanl* neural network to distinguish it from QCNN. The potential quantum advantage of using quanvolutional layers lies in the fact that quantum computers can access kernel functions in high-dimensional Hilbert spaces much more efficiently than classical computers. In quanvolutional NN, a challenge is to find a good structure for the parametric quantum circuit in which the number of qubits equals the size of the filter. This approach is also limited to qubit encoding since each layer requires a quantum embedding which has a non-negligible cost. Furthermore, stacking quanvolutional layers via pooling requires each parameterized quantum circuit to be measured multiple times for the measurement statistics.

Variational quantum circuits with the hierarchical structure consisting of $O(\log(n))$ layers do not exhibit the problem of "barren plateau" (Pesah et al. 2021). In other words, the precision required in the measurement grows at most polynomially with the system size. This result guarantees the trainability of the fully parameterized QCNN models studied in this work when randomly initializing their parameters. Furthermore, numerical calculations in Ref. Pesah et al. (2021) show that the cost function gradient vanishes at a slower rate (with $n$, the number of initial qubits) when all unitary operators in the same layer are identical as in QCNN (Cong et al. 2019). The hierarchical structure inspired by tensor network, without translational invariance, was first introduced in Ref. Grant et al. (2018).

The hierarchical quantum circuit can be combined with a classical neural network as demonstrated in Ref. Huang et al. (2021).

We note in passing that there exist several works proposing the quantum version of perceptron for binary classification (Tacchino et al. 2020; Mangini et al. 2020; Monteiro et al. 2021). While our QCNN model defers from them as it implements the entire network as a parameterized quantum circuit, interesting future work is to investigate the alternative approach to construct a complex network of quantum artificial neurons developed in the previous works.

## Appendix B. Classical CNN

In order to compare the classification accuracy of CNN and QCNN in fair conditions, we fixed hyperparameters used in the optimization step to be the same, which include iteration numbers, batch size, optimizer type, and its learning rates. In addition, we modified the structure of CNN in ways that its number of parameters subject to optimization is as close to that used in QCNN as possible. For example, since QCNN attains the best results with about 40 to 50 free parameters, we adjust the CNN structure accordingly. This led us to come up with two CNN, one with the input shape of (8, 1, 1) and another with the input shape of (16, 1, 1). In order to occupy the small number of input nodes for MNIST and Fashion MNIST classification, PCA and autoencoding are used for data pre-processing as done in QCNN. The CNNs go through convolutional and pooling stages twice, followed by a fully connected layer. The number of free parameters used in the CNN models is 26 or 44 for the case of 8 input nodes and 34 or 56 for the case of 16 input nodes.

The training also mimics that of QCNN. For every iteration step, 25 data are randomly selected from the training dataset, and trained via Adam optimizer with the learning rate of 0.01. We also fixed the number of iterations to be 200 as done in QCNN. The number of training (test) data are 12665 (2115) and 12000 (2000) for MNIST and fashion MNIST datasets, respectively.

## Appendix C. QCNN simulation results for MSE loss

In Section 4 of the main text, we presented the Pennylane simulation results of QCNN trained with the cross-entropy loss. When MSE is used as the cost function, similar results are obtained. We report classification results for MNIST and Fashion MNIST data attained from QCNN models that are trained with MSE in Tables 6 and 7.

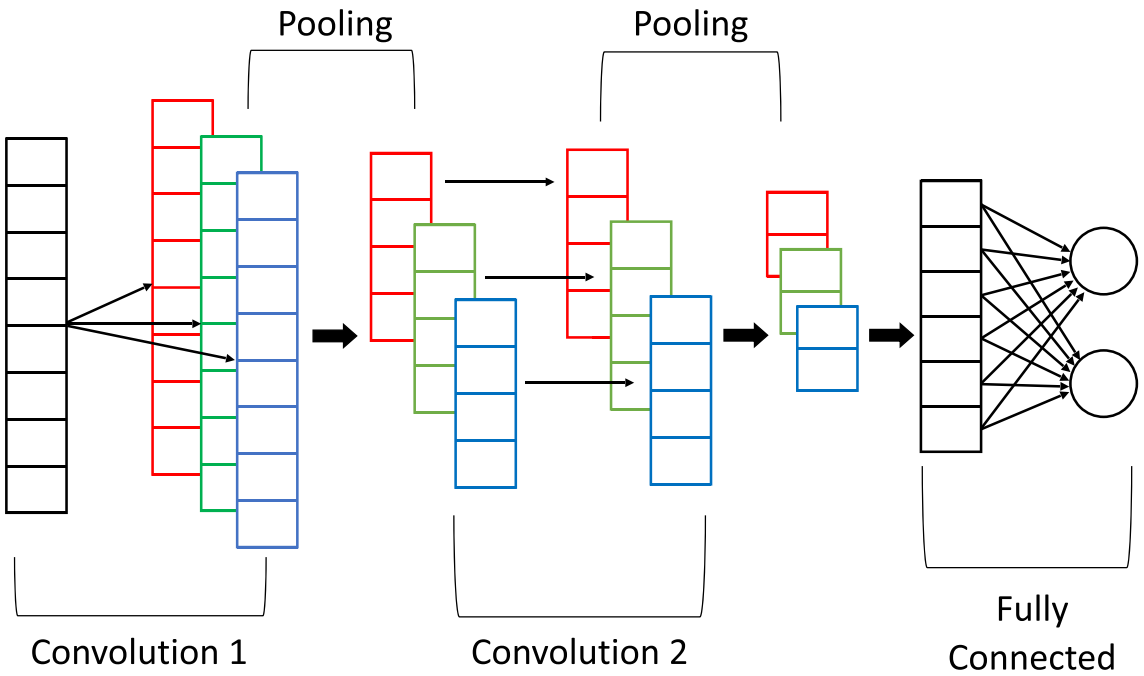



**Fig. 7** A schematic of CNN used in this work for comparing to the classification performance of QCNN. To make the comparison as fair as possible, the number of free parameters are adjusted to be similar to that used in QCNN. This leads to starting with a small number of input nodes. While we used two CNN structures with 8 and 16 input nodes, the figure shows a CNN structure with 8 input nodes as an example

**Table 6** Mean accuracy and one standard deviation of the classification for 0 and 1 in the MNIST dataset when the QCNN model is trained with MSE

| | | Classification accuracy | | | | |
|---|---|---|---|---|---|---|
| Ansatz | # of params | Amplitude | Qubit | Dense | HDE | HAE |
| 1 | 12 | 92.4 ± 3.1 | 96.1 ± 3.3 | 91.4 ± 8.6 | 60.3 ± 3.0 | 97.8 ± 1.2 |
| | | | 86.1 ± 8.8 | 88.9 ± 7.1 | 83.4 ± 4.0 | 77.9 ± 6.1 |
| 2 | 12 | 88.2 ± 7.1 | 86.4 ± 7.1 | 91.4 ± 0.3 | 54.4 ± 4.6 | 93.0 ± 2.7 |
| | | | 84.2 ± 10.0 | 88.9 ± 1.4 | 87.5 ± 8.6 | 78.7 ± 6.6 |
| 3 | 18 | 94.1 ± 1.7 | 98.0 ± 1.4 | 98.3 ± 0.1 | 69.8 ± 6.2 | 98.3 ± 0.3 |
| | | | 91.6 ± 8.9 | 95.3 ± 3.2 | 94.2 ± 3.8 | 83.5 ± 3.9 |
| 4 | 24 | 90.1 ± 2.0 | 98.2 ± 0.1 | 84.9 ± 2.5 | 63.2 ± 3.7 | 98.0 ± 0.2 |
| | | | 88.8 ± 6.3 | 85.8 ± 6.3 | 94.8 ± 1.1 | 85.5 ± 2.0 |
| 5 | 24 | 91.9 ± 1.7 | 98.1 ± 0.1 | 98.3 ± 0.1 | 65.7 ± 2.3 | 98.0 ± 0.1 |
| | | | 92.7 ± 3.5 | 94.0 ± 2.0 | 91.8 ± 3.5 | 83.0 ± 4.5 |
| 6 | 24 | 96.2 ± 2.0 | 98.1 ± 0.1 | 97.8 ± 0.2 | 74.4 ± 3.9 | 98.3 ± 0.2 |
| | | | 94.4 ± 3.9 | 92.5 ± 1.7 | 94.8 ± 2.8 | **86.6** ± 4.2 |
| 7 | 36 | 93.2 ± 4.3 | 98.4 ± 0.1 | 98.0 ± 0.3 | 68.9 ± 3.8 | 98.1 ± 0.2 |
| | | | 95.2 ± 4.5 | 92.4 ± 3.0 | 94.7 ± 4.1 | 80.2 ± 3.5 |
| 8 | 36 | 95.2 ± 1.4 | **98.5** ± 0.1 | 98.3 ± 0.1 | 68.5 ± 3.7 | 98.2 ± 0.1 |
| | | | **97.0** ± 2.8 | 93.4 ± 2.0 | 94.0 ± 2.8 | 86.1 ± 4.5 |
| 9a | 51 | 97.0 ± 1.0 | 98.2 ± 1.0 | 98.4 ± 0.1 | **77.4** ± 1.8 | **98.5** ± 0.2 |
| | | | 96.2 ± 1.4 | **96.6** ± 1.7 | **97.2** ± 1.8 | 85.0 ± 5.8 |
| 9b | 45 | **98.4** ± 0.1 | **98.5** ± 0.3 | **98.5** ± 0.1 | 75.9 ± 0.9 | 98.3 ± 0.3 |
| | | | **97.0** ± 2.0 | 95.7 ± 1.7 | 96.7 ± 1.7 | 85.9 ± 3.8 |

The mean and the standard deviation are obtained from five repetitions with random initialization of parameters. The first column shows the ansatz label. The second column shows the total number of parameters that are subject to optimization. For qubit, dense and hybrid encoding, two rows indicate the values obtained with different classical data pre-processing, namely PCA and autoencoding, respectively. The best result under each quantum data encoding method is written in bold

**Table 7** Mean accuracy and one standard deviation of the classification for 0 (t-shirt/top) and 1 (trouser) in the Fashion MNIST dataset when the QCNN model is trained with MSE

| | | Classification accuracy | | | | |
|---|---|---|---|---|---|---|
| Ansatz | # of params | Amplitude | Qubit | Dense | HDE | HAE |
| 1 | 12 | 88.1 ± 3.0 | 79.6 ± 10.9 | 81.4 ± 5.5 | 58.6 ± 2.4 | 84.2 ± 1.3 |
| | | | 80.0 ± 2.4 | 77.8 ± 8.4 | 89.7 ± 1.8 | 83.1 ± 4.3 |
| 2 | 12 | 87.8 ± 3.0 | 80.0 ± 8.4 | 78.0 ± 5.2 | 54.1 ± 3.5 | 78.8 ± 4.5 |
| | | | 70.0 ± 10.0 | 78.5 ± 8.3 | 88.7 ± 2.8 | 81.8 ± 5.1 |
| 3 | 18 | 87.0 ± 3.0 | 87.6 ± 3.6 | 92.7 ± 2.4 | 61.2 ± 2.9 | 86.0 ± 2.6 |
| | | | 85.1 ± 5.7 | 84.7 ± 7.2 | 90.1 ± 1.1 | 86.2 ± 3.3 |
| 4 | 24 | 89.7 ± 1.3 | 85.4 ± 2.9 | 90.7 ± 1.5 | 62.5 ± 2.0 | 84.0 ± 1.7 |
| | | | 81.2 ± 2.2 | 84.2 ± 5.6 | 88.4 ± 4.9 | 88.8 ± 3.6 |
| 5 | 24 | 90.7 ± 1.2 | 83.8 ± 4.4 | 88.8 ± 3.0 | 60.4 ± 0.9 | 84.7 ± 3.5 |
| | | | 81.1 ± 1.4 | 86.2 ± 3.3 | 90.7 ± 1.9 | 84.9 ± 4.8 |
| 6 | 24 | 88.8 ± 3.0 | 86.6 ± 2.4 | 89.4 ± 4.4 | 63.9 ± 2.2 | 84.4 ± 1.0 |
| | | | 86.1 ± 3.0 | 84.6 ± 5.3 | 91.7 ± 1.8 | 85.8 ± 4.0 |
| 7 | 36 | 90.0 ± 1.2 | 85.4 ± 3.2 | 92.5 ± 3.6 | 64.4 ± 1.8 | 84.6 ± 0.8 |
| | | | 86.4 ± 2.7 | 87.7 ± 6.1 | 91.9 ± 1.8 | 89.0 ± 3.2 |
| 8 | 36 | 89.7 ± 2.8 | 82.3 ± 2.1 | 90.0 ± 2.3 | 64.9 ± 2.5 | 86.5 ± 1.9 |
| | | | 85.8 ± 2.2 | 90.1 ± 3.6 | **92.9** ± 0.7 | 86.3 ± 7.2 |

**Table 7** (continued)

| | | Classification accuracy | | | | |
|---|---|---|---|---|---|---|
| Ansatz | # of params | Amplitude | Qubit | Dense | HDE | HAE |
| 9a | 51 | 90.8 ± 1.2 | **89.9** ± 1.9 | 88.8 ± 1.9 | **66.8** ± 1.7 | 88.4 ± 1.4 |
| | | | **92.7** ± 0.4 | **93.3** ± 1.1 | 92.8 ± 1.3 | **90.8** ± 2.1 |
| 9b | 45 | **91.0** ± 1.1 | 89.4 ± 2.5 | **93.0** ± 1.1 | 65.6 ± 2.6 | **89.3** ± 1.5 |
| | | | 89.1 ± 2.0 | 90.4 ± 3.1 | **92.9** ± 1.6 | 89.2 ± 4.6 |

The mean and the standard deviation are obtained from five repetitions with random initialization of parameters. The first column shows the ansatz label. The second column shows the total number of parameters that are subject to optimization. For qubit, dense and hybrid encoding, two rows indicate the values obtained with different classical data pre-processing, namely PCA and autoencoding, respectively. The best result under each quantum data encoding method is written in bold

## Appendix D. Classification with hierarchical quantum classifier

The hierarchical structure inspired by tensor network named as hierarchical quantum classifier (HQC) was first introduced in Ref. Grant et al. (2018). The HQC therein does not enforce translational invariance, and hence, the number of free parameters subject to optimization grows as $O(n)$ for a quantum circuit with $n$ input qubits. Although the simulation presented in the main manuscript aims to benchmark the classification performance of the QML model in which the number of parameters grows as $O(\log(n))$, we also report the simulation results of HQC with the tensor tree network (TTN) structure (Grant et al. 2018) in this supplementary section for interested readers. The TTN classifier does not employ parameterized quantum gates for pooling. Thus, for certain ansatz, the number of parameters differs from that of QCNN models. For example, although the convolutional circuit 2 in Fig. 2 has two free parameters, only one of them is effective since one of the qubits is traced out

**Table 8** Mean accuracy and one standard deviation of the classification for 0 and 1 in the MNIST dataset when the HQC model is trained with cross-entropy loss

| | | Classification accuracy | | | | |
|---|---|---|---|---|---|---|
| Ansatz | # of params | Amplitude | Qubit | Dense | HDE | HAE |
| 1 | 14 | 96.5 ± 0.4 | 94.7 ± 2.2 | 50.5 ± 3.5 | 69.9 ± 1.4 | **98.5** ± 0.1 |
| | | | 91.8 ± 1.2 | 82.1 ± 6.5 | 94.2 ± 3.0 | 81.6 ± 2.3 |
| 2 | 7 | 55.6 ± 1.7 | 57.3 ± 3.2 | 52.7 ± 1.2 | 53.8 ± 0.2 | 60.3 ± 0.2 |
| | | | 64.5 ± 15.9 | 74.0 ± 3.4 | 76.4 ± 8.9 | 56.1 ± 6.9 |
| 3 | 28 | 98.7 ± 0.2 | 98.2 ± 0.5 | 97.8 ± 0.2 | 81.2 ± 0.9 | 98.3 ± 0.1 |
| | | | 97.7 ± 1.2 | 95.7 ± 1.8 | 97.8 ± 1.4 | **88.7** ± 4.3 |
| 4 | 42 | 95.6 ± 2.2 | 87.4 ± 15.0 | 84.2 ± 15.5 | 77.0 ± 2.2 | 90.0 ± 10.3 |
| | | | 95.9 ± 1.1 | 95.7 ± 1.4 | 89.5 ± 8.0 | 81.5 ± 3.0 |
| 5 | 42 | 97.8 ± 0.5 | 96.7 ± 2.0 | 97.0 ± 1.1 | 77.1 ± 0.8 | 98.2 ± 0.2 |
| | | | 96.0 ± 3.1 | 96.5 ± 1.0 | 94.1 ± 3.4 | 88.1 ± 2.1 |
| 6 | 35 | 98.6 ± 0.2 | 98.0 ± 0.8 | 97.8 ± 0.1 | **81.6** ± 0.4 | 98.3 ± 0.3 |
| | | | 98.1 ± 1.5 | 95.7 ± 3.1 | 98.2 ± 0.9 | 90.2 ± 3.0 |
| 7 | 56 | 89.9 ± 13.0 | 88.7 ± 14.8 | 63.0 ± 5.1 | 72.4 ± 0.4 | 93.0 ± 10.8 |
| | | | 92.8 ± 1.5 | 94.9 ± 1.6 | 90.2 ± 7.1 | 85.2 ± 1.9 |
| 8 | 56 | 98.6 ± 0.2 | 95.8 ± 2.5 | 97.5 ± 0.6 | 77.8 ± 1.8 | 92.2 ± 12.2 |
| | | | **98.3** ± 1.0 | **97.9** ± 0.9 | 96.7 ± 1.1 | 85.7 ± 7.1 |
| 9 | 84 | **98.9** ± 0.1 | **98.6** ± 0.2 | **98.6** ± 0.2 | 81.0 ± 5.4 | 98.3 ± 0.0 |
| | | | 97.6 ± 0.9 | 96.9 ± 0.6 | **98.6** ± 0.2 | 82.2 ± 5.5 |

The mean and the standard deviation are obtained from five repetitions with random initialization of parameters. The first column shows the ansatz label. The second column shows the total number of parameters that are subject to optimization. For qubit, dense and hybrid encoding, two rows indicate the values obtained with different classical data pre-processing, namely PCA and autoencoding, respectively. The best result under each quantum data encoding method is written in bold

**Table 9** Mean accuracy and one standard deviation of the classification for 0 (t-shirt/top) and 1 (trouser) in the Fashion MNIST dataset when the HQC model is trained with cross-entropy loss

| | | Classification accuracy | | | | |
|---|---|---|---|---|---|---|
| Ansatz | # of params | Amplitude | Qubit | Dense | HDE | HAE |
| 1 | 14 | 90.1 ± 2.27 | 89.7 ± 2.3 | 61.8 ± 12.8 | 87.7 ± 1.9 | 61.2 ± 1.1 |
| | | | 89.0 ± 3.6 | 76.2 ± 12.2 | 85.6 ± 5.5 | 83.4 ± 18.8 |
| 2 | 7 | 77.7 ± 5.7 | 66.5 ± 0.0 | 52.6 ± 13.3 | 60.3 ± 0.0 | 49.5 ± 0.3 |
| | | | 63.5 ± 9.4 | 55.9 ± 6.8 | 64.7 ± 10.1 | 69.0 ± 13.4 |
| 3 | 28 | 89.5 ± 2.0 | 90.4 ± 2.6 | 85.0 ± 2.3 | 89.4 ± 1.7 | 68.7 ± 3.1 |
| | | | 93.3 ± 1.0 | 93.7 ± 1.7 | 91.0 ± 1.8 | **93.8** ± 1.7 |
| 4 | 42 | 91.4 ± 1.2 | 75.0 ± 22.7 | **91.6** ± 0.5 | 88.0 ± 1.3 | 63.8 ± 2.5 |
| | | | 92.2 ± 1.1 | 94.4 ± 0.4 | 82.0 ± 5.0 | 84.2 ± 19.1 |
| 5 | 42 | 90.1 ± 2.0 | 83.7 ± 18.4 | 92.0 ± 0.9 | 89.2 ± 2.5 | 66.3 ± 1.5 |
| | | | 93.0 ± 1.4 | 91.0 ± 2.1 | 86.2 ± 3.4 | 75.9 ± 23.7 |
| 6 | 35 | **92.5** ± 1.1 | 88.8 ± 1.8 | 86.6 ± 1.8 | **89.8** ± 1.9 | 68.9 ± 2.9 |
| | | | **94.2** ± 1.1 | 92.3 ± 1.5 | **91.4** ± 3.4 | 67.4 ± 23.8 |
| 7 | 56 | 90.9 ± 0.6 | 85.3 ± 14.0 | 76.4 ± 2.8 | 81.4 ± 6.9 | 62.9 ± 1.8 |
| | | | 91.6 ± 3.4 | 92.3 ± 2.5 | 81.4 ± 8.7 | 67.1 ± 23.4 |
| 8 | 56 | 90.7 ± 2.4 | **90.8** ± 0.8 | 91.1 ± 0.4 | 89.5 ± 1.7 | 67.6 ± 1.9 |
| | | | 94.1 ± 1.1 | 92.8 ± 1.0 | 88.7 ± 3.2 | 84.9 ± 19.5 |
| 9 | 84 | 89.9 ± 1.9 | 92.0 ± 1.6 | 91.3 ± 2.2 | 89.6 ± 1.0 | **69.0** ± 1.7 |
| | | | 93.8 ± 0.7 | **94.5** ± 0.8 | 91.0 ± 1.8 | 85.4 ± 19.8 |

The mean and the standard deviation are obtained from five repetitions with random initialization of parameters. The first column shows the ansatz label. The second column shows the total number of parameters that are subject to optimization. For qubit, dense and hybrid encoding, two rows indicate the values obtained with different classical data pre-processing, namely PCA and autoencoding, respectively. The best result under each quantum data encoding method is written in bold

as soon as the parameterized gate is applied. For brevity, here we only report the results obtained with the cross-entropy loss but similar results can be obtained with MSE. As can be seen from Table 8 and Table 9, the number of effective parameters (i.e., the second column) grows faster than that of QCNN models. An interesting observation is that there is no clear trend as the number of parameters is increased beyond 42, which is close to the maximum number of parameters used in QCNN. In other words, there is no clear motivation to increase the number of free parameters beyond 42 or so when seeking to improve the classification performance. Studying overfitting under the growth of the number of parameters remains an interesting open problem.

**Acknowledgements** We thank Quantum Open Source Foundation as this work was initiated under the Quantum Computing Mentorship program.

**Funding** This research is supported by the National Research Foundation of Korea (Grant No. 2019R1I1A1A01050161 and 2021M3H3A1038085) and Quantum Computing Development Program (Grant No. 2019M3E4A1080227).

**Data availability** The source code used in this study is available at https://github.com/takh04/QCNN.

## Declarations

**Conflict of interest** The authors declare no competing interests.

## References

Acciarri R et al (2017) J Instrum 12(03):P03011

Araujo IF, Park DK, Petruccione F, da Silva AJ (2021) Scientif Rep 11(1):6329

Aurisano A, Radovic A, Rocco D, Himmel A, Messier M, Niner E, Pawloski G, Psihas F, Sousa A, Vahle P (2016) J Instrum 11(09):P09001

Bergholm V, Izaac J, Schuld M, Gogolin C, Alam MS, Ahmed S, Arrazola JM, Blank C, Delgado A, Jahangiri S, McKiernan K, Meyer JJ, Niu Z, Száva A, Killoran N (2020) Pennylane: Automatic differentiation of hybrid quantum-classical computations. arXiv:1811.04968

Bharti K, Cervera-Lierta A, Kyaw TH, Haug T, Alperin-Lea S, Anand A, Degroote M, Heimonen H, Kottmann JS, Menke T, Mok WK, Sim S, Kwek LC, Aspuru-Guzik A (2021) Noisy intermediate-scale quantum (NISQ) algorithms. arXiv:2101.08448

Blank C, Park DK, Rhee JKK, Petruccione F (2020) npj Quantum Information 6(1):1

Cerezo M, Arrasmith A, Babbush R, Benjamin SC, Endo S, Fujii K, McClean JR, Mitarai K, Yuan X, Cincio L, Coles PJ (2021) Variational quantum algorithms. Nature Reviews Physics 3(9):625–644. https://doi.org/10.1038/s42254-021-00348-9

Chen SYC, Wei TC, Zhang C, Yu H, Yoo S (2020) Quantum convolutional neural networks for high energy physics data analysis. arXiv:2012.12177
Cong I, Choi S, Lukin MD (2019) Nat Phys 15(12):1273. http://www.nature.com/articles/s41567-019-0648-8
George D, Huerta E (2018) Phys Lett B 778:64. https://www.sciencedirect.com/science/article/pii/S0370269317310390
Giovannetti V, Lloyd S, Maccone L (2008) Phys Rev Lett 100:160501
Goodfellow I, Bengio Y, Courville A (2016) Deep Learning. MIT Press, Cambridge. http://www.deeplearningbook.org
Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (MIT Press, Cambridge, MA, USA), NIPS'14, p 2672–2680
Grant E, Benedetti M, Cao S, Hallam A, Lockhart J, Stojevic V, Green AG, Severini S (2018) npj Quantum Information 4(1):65. http://www.nature.com/articles/s41534-018-0116-9
Henderson M, Shakya S, Pradhan S, Cook T (2020) Quantum Mach Intell 2(1):2
Huang R, Tan X, Xu Q (2021) Neurocomputing 452:89. https://www.sciencedirect.com/science/article/pii/S092523122100624X
Jolliffe IT (2002) Principal Component Analysis, 2nd edn Springer Series in Statistics. Springer, New York
Kerenidis I, Landman J, Luongo A, Prakash A (2019) In: Wallach H, Larochelle H, Beygelzimer A, D'Alché-Buc F, Fox E, Garnett R (eds) Advances in Neural Information Processing Systems. vol. 32, vol 32. Curran Associates Inc. https://proceedings.neurips.cc/paper/2019/file/16026d60ff9b54410b3435b403afd226-Paper.pdf
Kerenidis I, Landman J, Prakash A (2019) Quantum algorithms for deep convolutional neural networks. arXiv:1911.01117
Kingma DP, Ba J (2017) Adam: A method for stochastic optimization. arXiv:1412.6980
LaRose R, Coyle B (2020) Phys Rev A 102:032420
LeCun Y, Bengio Y, Hinton G (2015) Nature 521(7553):436
Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Proc IEEE 86(11):2278
Li J, Yang X, Peng X, Sun CP (2017) Phys Rev Lett 118:150503
Li Y, Zhou RG, Xu R, Luo J, Hu W (2020) Quantum Sci Technol 5(4):044003
Liu J, Lim KH, Wood KL, Huang W, Guo C, Huang HL (2021) Hybrid quantum-classical convolutional neural networks. Science China Physics, Mechanics & Astronomy 64(9):290311
Lloyd S, Mohseni M, Rebentrost P (2013) arXiv:1307.0411
Lloyd S, Mohseni M, Rebentrost P (2014) Nat Phys 10(9):631
Lloyd S, Schuld M, Ijaz A, Izaac J, Killoran N (2020) Quantum embeddings for machine learning. arXiv:2001.03622
MacCormack I, Delaney C, Galda A, Aggarwal N, Narang P (2020) [physics, physics:quant-ph]. arXiv:2012.14439
Mangini S, Tacchino F, Gerace D, Bajoni D, Macchiavello C (2021) EPL (Europhysics Letters) 134(1):10002
Mangini S, Tacchino F, Gerace D, Macchiavello C, Bajoni D (2020) Quantum computing model of an artificial neuron with continuously valued input data. Machine Learning: Science and Technology 1(4):045008
Mitarai K, Negoro M, Kitagawa M, Fujii K (2018) Phys Rev A 98:032309
Monteiro CA, Filho GI, Costa MHJ, de Paula Neto FM, de Oliveira WR (2021) Neural Netw 143:698. https://www.sciencedirect.com/science/article/pii/S0893608021003051
Nesterov Y (1983) Proceedings of the USSR Academy of Sciences 269:543
Park DK, Blank C, Petruccione F (2021) In: 2021 International Joint Conference on Neural Networks (IJCNN), pp 1–7
Park DK, Petruccione F, Rhee JKK (2019) Scientif Rep 9(1):3949
Parrish RM, Hohenstein EG, McMahon PL, Martínez TJ (2019) Phys Rev Lett 122:230401
Pesah A, Cerezo M, Wang S, Volkoff T, Sornborger AT, Coles PJ (2021) Phys Rev X 11:041011
Preskill J (2018) Quantum 2:79
Rebentrost P, Mohseni M, Lloyd S (2014) Phys Rev Lett 113:130503
Schuld M, Bergholm V, Gogolin C, Izaac J, Killoran N (2019) Phys Rev A 99:032331
Schuld M, Killoran N (2019) Phys Rev Lett 122:040504
Sim S, Johnson PD, Aspuru-Guzik A (2019) Adv Quantum Technol 2(12):1900070
Tacchino F, Macchiavello C, Gerace D, Bajoni D (2020) An artificial neuron implemented on an actual quantum processor. npj Quantum Information 5(1):26
Tanaka A, Tomiya A (2017) J Phys Soc Japan 86(6):063001
Vatan F, Williams C (2004) Phys Rev A 69:032315
Veras TML, De Araujo ICS, Park KD, Da Silva AJ (2020) IEEE Trans Comput 1–1
Wei S, Chen Y, Zhou Z, Long G (2021) [quant-ph]. arXiv:2104.06918
Wei H, Di Y (2012) Quantum Inf Comput 12(3-4):262
Wiebe N, Kapoor A, Svore KM (2015) Quantum Info Comput 15(3–4):316–356