frontiers | Frontiers in Physics

TYPE Original Research
PUBLISHED 03 March 2025
DOI 10.3389/fphy.2025.1529188

OPEN ACCESS

EDITED BY
Jaewoo Joo,
University of Portsmouth, United Kingdom

REVIEWED BY
De-Sheng Li,
Hunan Institute of Engineering, China
Robson Christie,
University of Portsmouth, United Kingdom

*CORRESPONDENCE
Daniel K. Park,
dkd.park@yonsei.ac.kr

†These authors have contributed equally to this work

RECEIVED 16 November 2024
ACCEPTED 03 February 2025
PUBLISHED 03 March 2025

CITATION
Lee C, Araujo IF, Kim D, Lee J, Park S, Ryu J-Y and Park DK (2025) Optimizing quantum convolutional neural network architectures for arbitrary data dimension.
Front. Phys. 13:1529188.
doi: 10.3389/fphy.2025.1529188

Optimizing quantum convolutional neural network architectures for arbitrary data dimension

Changwon Lee[1], Israel F. Araujo[1], Dongha Kim[2†], Junghan Lee[3†], Siheon Park[4†], Ju-Young Ryu[2,5†] and Daniel K. Park[1,6*]

[1]Department of Statistics and Data Science, Yonsei University, Seoul, Republic of Korea, [2]School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea, [3]Department of Physics, Yonsei University, Seoul, Republic of Korea, [4]Department of Physics and Astronomy, Seoul National University, Seoul, Republic of Korea, [5]Quantum AI Team, Norma Inc., Seoul, Republic of Korea, [6]Department of Applied Statistics, Yonsei University, Seoul, Republic of Korea

Quantum convolutional neural networks (QCNNs) represent a promising approach in quantum machine learning, paving new directions for both quantum and classical data analysis. This approach is particularly attractive due to the absence of the barren plateau problem, a fundamental challenge in training quantum neural networks (QNNs), and its feasibility. However, a limitation arises when applying QCNNs to classical data. The network architecture is most natural when the number of input qubits is a power of two, as this number is reduced by a factor of two in each pooling layer. The number of input qubits determines the dimensions (i.e., the number of features) of the input data that can be processed, restricting the applicability of QCNN algorithms to real-world data. To address this issue, we propose a QCNN architecture capable of handling arbitrary input data dimensions while optimizing the allocation of quantum resources such as ancillary qubits and quantum gates. This optimization is not only important for minimizing computational resources, but also essential in noisy intermediate-scale quantum (NISQ) computing, as the size of the quantum circuits that can be executed reliably is limited. Through numerical simulations, we benchmarked the classification performance of various QCNN architectures across multiple datasets with arbitrary input data dimensions, including MNIST, Landsat satellite, Fashion-MNIST, and Ionosphere. The results validate that the proposed QCNN architecture achieves excellent classification performance while utilizing a minimal resource overhead, providing an optimal solution when reliable quantum computation is constrained by noise and imperfections.

KEYWORDS

quantum computing, quantum machine learning, machine learning, quantum circuit, quantum algorithm

1 Introduction

The advent of deep neural networks (DNNs) has transformed machine learning, drawing considerable research attention owing to the efficacy and broad applicability of DNNs [1, 2]. Among the DNNs, convolutional neural networks (CNNs) have emerged to pivotally contribute toward image processing and vision tasks [3, 4]. By leveraging filtering

techniques, the CNN architecture effectively detects and extracts spatial features from input data. CNNs exhibit exceptional performance in diverse domains—including image classification, object detection, face recognition, and medical image processing—and have attracted interest from both researchers and industry [5–9].

Although DNNs have proven successful in various data analytics tasks, the increasing volume and complexity of datasets present a challenge to the current classical computing paradigm, prompting the exploration of alternative solutions. Quantum machine learning (QML) has emerged as a promising approach to address the fundamental limitations of classical machine learning. By leveraging the advantages of quantum computing techniques and algorithms, QML aims to overcome the inherent constraints of its classical counterparts [10–13]. However, a challenge in contemporary quantum computing lies in the difficulty of constructing quantum hardware. This challenge is characterized by noisy intermediate-scale quantum (NISQ) computing [14, 15], as the number of quantum processors that can be controlled reliably is limited owing to noise. Quantum-classical hybrid approaches based on parameterized quantum circuits (PQCs) have been developed to enhance the utility of NISQ devices [16–18]. These strategies have contributed to advancements in quantum computing and machine learning, facilitating improved performance and applicability in various domains. In particular, PQC-based QML models have demonstrated a potential to outperform classical models in terms of sample complexity, generalization, and trainability [19–25]. However, PQCs encounter a critical challenge in addressing real-world problems, particularly in relation to scalability, which is attributed to a phenomenon known as barren plateaus (BP) [26, 27]. This phenomenon is characterized by an intrinsic tradeoff between the expressibility and trainability of PQCs [28], causing the gradient of the cost function to vanish exponentially with the number of qubits under certain conditions. An effective strategy for avoiding BPs is to adopt a hierarchical quantum circuit structure, wherein the number of qubits decreases exponentially with the depth of the quantum circuit [29, 30]. Quantum convolutional neural networks (QCNNs) notably employ this strategy, as highlighted in recent studies [31–37]. Inspired by the CNN architecture, the QCNN is composed of a sequence of quantum convolutional and pooling layers. Each pooling layer typically reduces the number of qubits by a factor of two, thereby increasing the quantum circuit depth to $O(\log(n))$ for n input qubits. This logarithmic depth enables the implementation of extremely compact quantum machine learning models, with the number of parameters growing logarithmically with n [32, 33, 35]. Furthermore, QCNNs exhibit strong generalization capabilities [38] and are closely connected to tensor networks [29], making them an important architecture in QML.

The logarithmic circuit depth is one of the features that renders the QCNN an attractive architecture for NISQ devices, implying that the most natural design approach is to set the number of input qubits to a power of two. However, the number of input qubits required is determined by the input data dimension, i.e., the number of features in the data. If the input data require a number of qubits that is not a power of two, some layers will inevitably have odd numbers of qubits. This can occur either in the initial number of input qubits or during the pooling operation, representing a deviation from the optimal design and requiring appropriate adjustments. In particular, having an odd number of qubits in a quantum convolutional layer results in an increase in the circuit depth if all nearest-neighbor qubits interact with each other. Consequently, the run time increases and noise can negatively impact the overall performance and reliability of the QCNN. Moreover, it is unclear how breaking translational invariance in the pooling layer, a key property of the QCNN, affects overall performance. Because these considerations constrain the applicability of the QCNN algorithm, our goal is to optimize the QCNN architecture, developing an effective QML algorithm capable of handling arbitrary data dimensions.

In this study, we propose an efficient QCNN architecture capable of handling arbitrary data dimensions. Two naive approaches served as baselines to benchmark the proposed architectures: the classical data padding method, which increases the input data dimension through zero padding or periodic padding to encode it as a power of two, and the skip pooling method, which directly passes one qubit from each layer containing an odd number of qubits to the next layer without pooling. The first method requires additional ancillary qubits without increasing the circuit depth, whereas the second method does not require ancillary qubits but results in an increased circuit depth to preserve the translational invariance in the convolutional layers. By contrast, our proposed method effectively optimizes the QCNN architecture by applying a qubit padding technique that leverages ancillary qubits. By introducing an ancillary qubit into layers with an odd number of qubits, we can effectively construct convolutional layers without an additional increase in circuit depth. This enables a reduction in the total number of qubits by up to $\log(n)$ compared with the classical data padding method. Moreover, the reuse of a single ancillary qubit across multiple layers further reduces the required number of ancillary qubits. This strategy of qubit reuse efficiently optimizes the number of ancillary qubits along with the circuit depth. In addition, recycling the ancilla qubit facilitates uniform operations across layers, systematically enhancing the stability and efficiency of the QCNN architecture. To validate our approach, we benchmarked our proposed method against naive methods on various datasets: MNIST, Landsat satellite, Fashion-MNIST and Ionosphere datasets. Numerical simulation results show that our proposed method achieves a high classification accuracy comparable to that of naive methods. Notably, our method significantly reduces the number of qubits used compared with classical data padding methods, providing substantial advantages in terms of resource efficiency. We also conducted noise simulations using information from an IBM quantum device that mimics the operations and characteristics of real quantum hardware. The noise simulation results demonstrate that the proposed method exhibits less performance degradation and lower variability under realistic noise conditions than the skip pooling method. This is a consequence of the skip pooling method requiring a larger circuit depth. Because the proposed method not only improves the runtime but also enhances robustness against noise, it serves as a fundamental building block for the effective applicability of QCNNs to real-world data with an arbitrary number of features.

The remainder of this paper is organized as follows. We introduce the foundational concepts of QML in Section 2, focusing on principles underlying quantum neural networks (QNNs) and QCNNs. Section 3 presents the detailed design of a QCNN

architecture capable of handling arbitrary data dimensions, including a comparative analysis between naive methods and our proposed methods. Simulation results are presented in Section 4 along with a comparative performance analysis of the naive and proposed methods under both noiseless and noisy conditions. Section 5 explores possible extensions of multi-qubit quantum convolutional operations. Finally, concluding remarks are presented in Section 6.

2 Background

2.1 Quantum neural network

A DNN is a machine learning model constructed by deeply stacking layers of neurons [39]. Using nonlinear activation functions—such as the sigmoid, ReLU, and hyperbolic tangent functions—the DNN can learn patterns in complex data to solve various problems with high performance. Although the mathematical foundation for the success of DNNs remains an active area of research [40], several studies, as well as the universal approximation theorem, have demonstrated that neural networks can approximate complex functions with arbitrary accuracy [41]. On the other hand, a QNN is a quantum machine learning model, where the data is propagated through a PQC in the form of a quantum state. The data can be either intrinsically quantum, if the data source is a quantum system, or classical. In the latter case, which is the primary focus of this work, the classical data first has to be mapped to a quantum state. Note that nonlinear transformation of the input data can occur during this data mapping step. Since the parameters of the PQC are real-valued and its output is differentiable with respect to the parameters, they are typically trained through classical optimizers, similar to how DNNs are trained. In this sense, the QNN-based ML is also known to be a quantum-classical hybrid approach. Quantum-classical hybrid approaches using PQCs are effective at shallow circuit depths [18], which significantly enhances their applicability to NISQ devices with limited numbers of qubits. In addition, the PQC can approximate a broad family of functions with arbitrary accuracy, making it a good machine learning model [16, 21].

A QNN consists of three primary components: (1) Data Embedding, (2) Data Processing, and (3)Measurements. These models transform classical data into quantum states, to be processed using a sequence of parameterized quantum gates. The training process connects the measurement results to the loss function, which is used to tune and train the parameters. Figure 1 depicts the overall training process of a QNN. Consider a dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{M}$ with $\boldsymbol{x}_i \in \mathbb{R}^N$ and $y_i \in \mathbb{R}$, and PQC $U_i(\theta_i)$, where $\theta_i = (\theta^1, \ldots, \theta^m)$ represents a set of tunable parameters. Typically, the dataset is embedded into a quantum Hilbert space by a unitary transformation applied to n qubits initially prepared in $|0\rangle^{\otimes n}$. Denoting the data embedded state as $|\psi_{\text{in}}\rangle$, the final state of the QNN can be expressed as follows:

$$|\psi_{\text{out}}\rangle = U_l(\theta_l)\, U_{l-1}(\theta_{l-1}) \ldots U_1(\theta_1)|\psi_{\text{in}}\rangle.$$

The output function of the QNN is $f(\boldsymbol{\theta}, \boldsymbol{x}_i) = \langle\psi_{\text{out}}|\mathcal{O}|\psi_{\text{out}}\rangle$, where $\mathcal{O}$ is an observable of the quantum circuit. The parameters are optimized using classical methods such as the gradient descent algorithm [42], which minimizes the following loss function:

$$L(\boldsymbol{\theta}) = \frac{1}{M}\sum_{i=1}^{M} |y_i - f(\boldsymbol{\theta}, \boldsymbol{x}_i)|^2.$$

Furthermore, gradients in quantum computing can be computed directly using methods such as parameter-shift rules, wherein derivatives are approximated by shifting the parameters at fixed intervals and then measuring the difference in the output of the quantum circuit as a result of that change [43, 44]. However, as the number of qubits in the training PQC increases, the parameter space of the quantum circuit also increases, leading to the BP phenomenon [26]. Although this phenomenon represents a significant performance limitation of QML, it can be mitigated by applying hierarchical structures in the quantum circuits [29], such as the QCNN [30].

2.2 Quantum convolutional neural network

The QCNN is a type of PQC inspired by the concept of CNNs. QCNNs exhibit the property of translational invariance, with quantum circuits sharing the same parameters within the convolutional layer, and reduce dimensionality by tracing out some qubits during the pooling operation. A primary distinction between a QCNN and a CNN is that data in a QCNN are defined in a Hilbert space that grows exponentially with the number of qubits. Consequently, whereas classical convolution operations typically transform vectors into scalars, quantum convolution operations perform more complex linear mapping, transforming vectors into vectors through a unitary transformation of the state vector. Thus, quantum convolutional operations are distinct from classical convolutional operations. Problems defined in the exponentially large Hilbert space are intractable in a classical setting; however, QCNNs offer the possibility of effectively overcoming these challenges by utilizing qubits in a quantum setting. QCNNs have also demonstrated the capability to classify images in a manner similar to their classical counterparts [32]. In the convolutional layer, local features are extracted through unitary single-qubit rotations and entanglements between adjacent qubits, and the features dimensions are reduced in a pooling layer. Typically, the pooling layer includes parameterized two-qubit controlled unitary gates, and the control qubit is traced out after the gate operation to halve it. In binary classification tasks (i.e., $y_i \in \{+1, -1\}$), a QCNN repeats the convolutional and pooling layers until only one qubit remains, and performs classification by measuring the last qubit. These architectures maintain a shallow circuit depth by effectively reducing the number of qubits through a hierarchical structure, which is crucial for improving model performance and avoiding BP. In addition, by turning off the translational invariance property, which involves sharing the same parameters within the convolutional or pooling layer, more parameters can be introduced to the QML model while preserving the absence of BP. The structure between layers ensures a circuit depth of $O(\log(n))$ for n input qubits. In particular, the shallow depth of the QCNN contributes to its high performance in NISQ devices. In addition, the simple structure of repetitive circuits allows for the applicability of QCNNs to a wide range of tasks including classical and quantum data classification [31, 32, 45], error

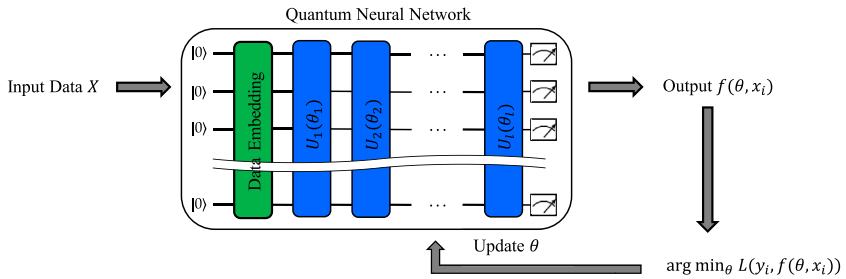


FIGURE 1
A schematic of the training process of a QNN. This figure outlines the sequential steps involved in training a QNN, starting with data preparation and qubit initialization, followed by the application of quantum gates to embed the data, and using a PQC for training. The output is obtained from measurements of the quantum state. Parameters are updated by minimizing the loss function.

correction [31], and classical to quantum transfer learning [33, 46]. Moreover, the QCNN architecture can be easily integrated into other QNN models and tasks, such as quantum recurrent neural networks [47], quantum generative adversarial networks [48, 49], quantum graph convolutional neural networks [50, 51], quantum one-class classifiers [52], and quantum self-supervised learning [53], bringing the aforementioned advantages to these areas as well.

It is also worth noting that mid-circuit measurements are not strictly required for QCNNs to achieve their key features—translational invariance and dimensionality reduction. These characteristics can be effectively realized through specific arrangements of two-qubit gates, along with local measurements that commute with partial trace operations.

3 QCNN architectures for arbitrary data dimensions

Data encoding is a crucial process in quantum computing that transforms classical data into quantum-state. During this process, the input data dimensions determine the number of qubits required to represent the quantum state. For example, amplitude encoding allows the input data $\boldsymbol{x} = (x_1, \ldots, x_N)^T \in \mathbb{R}^N$ with dimensions $N = 2^n$ to be represented as amplitudes of an n-qubit quantum state. Since a pooling layer in QCNN discards half of the qubits, n also has to be a power of two. The condition requiring the number of input qubits to be a power of two plays a critical role in the efficient applicability of QCNN algorithms. However, the dimensions of classical data do not always conform to this condition. In this section, we describe our proposed method that enables QCNN architectures to handle arbitrary data dimensions, along with naive baseline methods. To demonstrate the efficiency of the proposed method, we analyzed the number of ancillary qubits, parameters, and circuit depths when implementing the QCNN algorithm.

3.1 Naive methods

3.1.1 Classical data padding

In a CNN, the direct application of kernels to input feature maps during convolution operations can reduce the output feature map size compared to that of the input, leading to the potential loss of important information. Padding techniques that artificially enlarge the input feature map by adding specific values (typically zeros) around the input data are employed to prevent information loss and enhance model training. Inspired by CNN, similar padding strategies can be applied in QCNNs by increasing the data dimensions until the data can be encoded into a number of input qubits with a power of two, by either adding a constant value (zero) or periodically repeating the input data. These methods are referred to as 'zero-data padding' and 'periodic-data padding', respectively. Figure 2 illustrates an example of the classical data padding method, with a handwritten digit image reduced to 30 dimensions. This 30-dimensional input can be encoded into five input qubits using amplitude encoding. Classical data padding methods can be applied to expand the data dimension, aligning the number of input qubits to a power of two. By using either zero- or periodic-data padding, the data dimensions can be expanded to 2^8. Through these padding procedures, three additional ancillary qubits are introduced to the original system of five input qubits, resulting in an eight-qubit QCNN structure. Classical data padding approaches not only increase the number of qubits but may also result in poor classification performance because the number of dummy features that are added can often be significantly larger than the number of data features.

We denote the initial number of input qubits as K, and let $\lceil \log_2(K) \rceil = m$. Classical data padding typically employs $2^m - K$ ancillary qubits. Without loss of generality, we assume that both the convolutional and pooling gates of the QCNN have one parameter and a depth of 1. The quantum circuit depth is $\sum_{i=1}^{m}(2l_i + 1) - l_m$, where l_i denotes the number of complete sets of two-qubit gates connecting the nearest-neighboring qubits as well as the top and bottom qubits in the ith convolutional layer as depicted in Figure 2. If the parameters are shared, then the total number of parameters is $\sum_{i=1}^{m}(l_i + 1)$. However, if the parameters are not shared, then the total number of parameters is $\sum_{i=1}^{m-1}(\lceil \frac{2^m}{2^{i-1}} \rceil (l_i + \frac{1}{2})) + l_m + 1$.

3.1.2 Skip pooling

An alternate method enables implementation of the QCNN algorithm without the use of additional ancillary qubits. This method adopts a strategy where, within the QCNN structure, a qubit from each layer containing an odd number of qubits is passed directly to the next layer without performing a pooling operation. Although this approach minimizes the use of qubits, it inherently

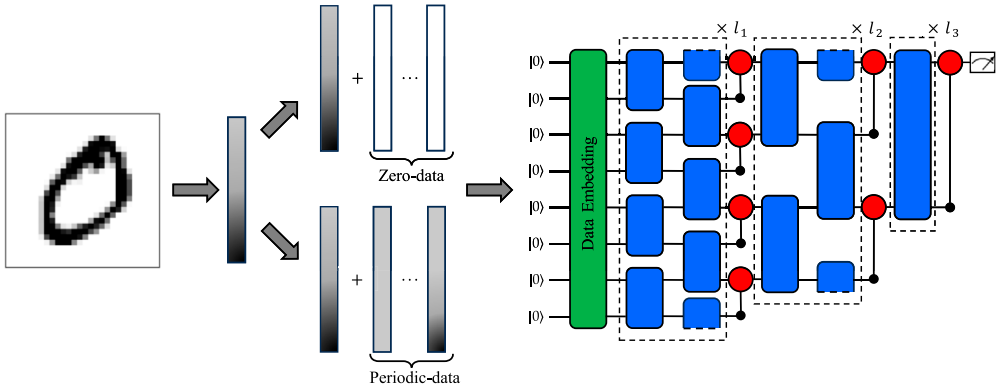


FIGURE 2
Schematic of the QCNN algorithm with eight input qubits using the classical data padding method. The quantum circuit consists of three components: data embedding (green squares), convolutional gate (blue squares), and pooling gate (red circles). The data embedding component is further divided into two methods: top embedding, which pads the zero-data, and bottom embedding, which pads the input data repeatedly. The convolutional and pooling gate use a PQC. Throughout the hierarchy, the convolutional gate consistently applies the same two-qubit ansatz to the nearest-neighboring qubit in each layer. In the ith convolutional layer, the set of gates that completes a loop connecting all nearest-neighboring qubits and the qubits at the boundaries can be repeated l_i times. The pooling gate uses the same approach, and can be represented as a controlled unitary transformation that is activated when the control qubit is 1.

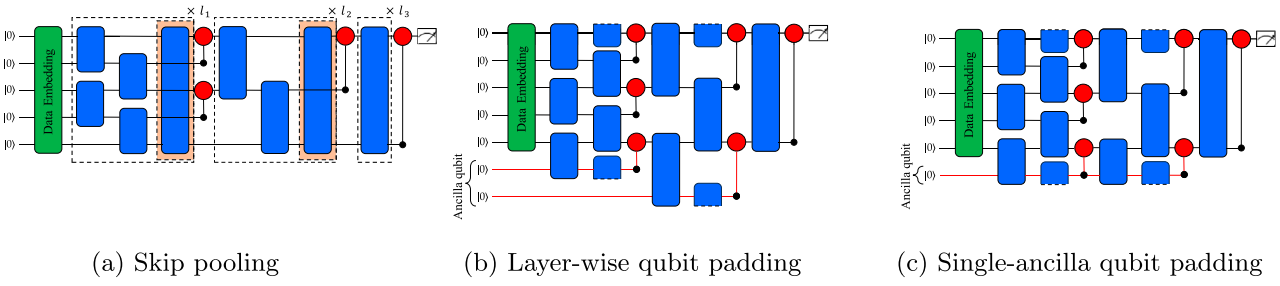


FIGURE 3
Schematic of a QCNN algorithm with five initial input qubits in a circuit with and without ancillary qubits. **(A)** Uses a method called skip pooling to perform convolution and pooling operations between each qubit without ancillary qubits. **(B)** Uses two ancillary qubits to construct the QCNN in a method called layer-wise qubit padding. The first layer has five qubits. Because this is an odd number of layers, one ancillary qubit is used to perform convolution and pooling operations. The second layer has three qubits, and another ancillary qubit is used. **(C)** Uses only one ancillary qubit to construct the QCNN using a method called single-ancilla qubit padding. Unlike **(B)**, the single ancillary qubit performs the convolution and pooling operations sequentially.

increases the circuit depth during convolutional operations in layers with odd numbers of qubits. This can affect the overall efficiency and execution speed of the quantum circuits. We refer to this method as 'skip pooling'. Figure 3A depicts an example of skip pooling, with a 30-dimensional input encoded into five input qubits using amplitude encoding. In the first layer of the QCNN, the convolutional operation between neighboring qubits introduces one more gate over classical data padding. Then, the 5th qubit passes directly to the next layer without any pooling operations. This procedure is repeated for the second layer. When applied to layers that contain an odd number of qubits, skip pooling increases circuit depth during the convolutional operation. The increased circuit depth can be affected by noise, which may potentially propagate through each layer, reducing the accuracy and reliability of the information. This directly affects the efficiency and performance of the QCNN, necessitating an effective optimization strategy.

Unlike classical data padding, no ancillary qubits are used; however, an additional circuit depth of $\sum_{i=1}^{m-1} Y_i l_i$ is incurred, where $Y_i \coloneqq \lceil \frac{K}{2^{i-1}} \rceil \mathrm{mod} 2$ is an odd number of qubits in the ith layer divided by the power of two. If the parameters are shared, then the total number of parameters is $\sum_{i=1}^{m}(l_i+1)$. In contrast, if the parameters are not shared, then the total number of parameters is $\sum_{i=1}^{m-1}(\lceil \frac{K}{2^{i-1}} \rceil (l_i + \frac{1}{2}) - \frac{1}{2} Y_i) + l_m + 1$.

3.2 Proposed methods

3.2.1 Layer-wise qubit padding

As an alternative to the aforementioned naive methods, we introduce a qubit padding method that leverages ancillary qubits in the QCNN algorithm. Whereas classical data padding requires additional ancillary qubits by increasing the size of the input data, qubit padding directly leverages ancillary qubits in the convolutional

and pooling operations of the QCNN. Using ancillary qubits for layers containing odd numbers of qubits, we optimized the QCNN algorithm and designed an architecture capable of handling arbitrary data dimensions. We refer to this method as 'layer-wise qubit padding'. Figure 3B depicts an example of layer-wise qubit padding. As in the skip pooling example, a 30-dimensional input was encoded into five input qubits using amplitude encoding. In the first layer of the QCNN, one ancillary qubit is added to perform convolutional and pooling operations with neighboring qubits. This ensures pairwise matching between all qubits in two steps, avoiding additional circuit depth that may arise in skip pooling. This procedure is repeated for the second layer. Consequently, in a five-qubit QCNN with layer-wise qubit padding and two layers containing an odd number of qubits, two ancillary qubits are used to optimize the architecture.

Layer-wise qubit padding generally requires $\sum_{i=1}^{m-1} Y_i$ ancillary qubits. The quantum circuit depth of $\sum_{i=1}^{m}(2l_i+1)-l_m$ is identical to that used in classical data padding. If the parameters are shared, then the total number of parameters is $\sum_{i=1}^{m}(l_i+1)$. On the other hand, if the parameters are not shared, then the total number of parameters is $\sum_{i=1}^{m-1}((\lceil \frac{K}{2^{i-1}} \rceil + Y_i)(l_i+\frac{1}{2}))+l_m+1$, which is $\sum_{i=1}^{m-1}(Y_i)(l_i+1)$ more than that for skip pooling.

3.2.2 Single-ancilla qubit padding

Finally, we propose 'single-ancilla qubit padding,' a QCNN architecture designed to handle arbitrary input data dimensions using only one ancillary qubit. By reusing the ancillary qubit throughout the QCNN architecture, we significantly reduced the number of total qubits required for optimization. Figure 3C illustrates an example of single-ancilla qubit padding. Unlike layer-wise qubit padding, this method reuses the same ancillary qubit for every layer with an odd number of qubits. Preserving the information of the ancillary qubit without resetting it when it is passed to the next layer plays a crucial role in enhancing the stability and performance of model training.

Although single-ancilla qubit padding uses only one ancillary qubit, the quantum circuit depth and number of parameters remain the same as those in layer-wise qubit padding. Figure 4A illustrates the circuit depths of the skip pooling and qubit padding methods. As the number of input qubits increases, the circuit depth of the qubit padding method is logarithmically less than that of the skip pooling method. Figure 4B illustrates the number of parameters in the case of parameter-sharing off for the classical data padding, skip pooling, and layer-wise and single-ancilla qubit padding methods. In the case of parameter-sharing on, the number of parameters is the same across all methods, hence we do not track how the number of parameters changes. We only compared the number of parameters in the case of parameter-sharing off. Without loss of generality, we assumed the convolutional layer l_i to be equal to 1. When the number of input qubits is not a power of two, classical data padding uses the largest number of the parameters, whereas skip pooling and qubit padding use relatively fewer parameters. By introducing additional qubits into certain layers, the qubit padding method uses slightly more parameters than skip pooling. Therefore, single-ancilla qubit padding enables the design of efficient QCNN architectures with optimal allocation of quantum resources such as ancillary qubits and quantum gates.

4 Results

The previous section provided an overview of various QCNN padding methods. In this section, we benchmark and evaluate our proposed padding methods in comparison with naive methods using a variety of classical datasets. To address the characteristics of NISQ devices, we added noise to the quantum circuits for benchmarking.

4.1 Methods and setup

4.1.1 Datasets

Our experiments were conducted using a variety of datasets. The MNIST dataset consists of handwritten digits, each represented as a 28×28 pixel image in grayscale [54]. The dataset consists of a total of 60,000 training and 10,000 test images, each labeled with a numerical value ranging from 0 to 9. In our benchmarking, we focused on binary classification tasks by selecting two distinct pairs of labels: 0 & 1 and 5 & 6. In the noiseless scenario, we split the dataset into 10,000 training, 1,000 validation, and 1,000 test sets. In the noisy scenario, we split the dataset into 2,000 training, 200 validation, and 200 test sets. Furthermore, 28×28 features were relatively high-dimensional for current quantum hardware; therefore, we used principal component analysis as a dimensionality reduction technique to reduce the dataset to 30 features.

In addition, we conducted experiments with the Landsat Satellite, Fashion-MNIST and Ionosphere datasets in the noisy scenario. The Landsat Satellite dataset classifies multi-spectral values of pixels from satellite images [55]. The dataset contains a total of 6,435 instances with 36 features, where each instance is labeled into one of six land cover classes. In our benchmarking, we focused on binary classification tasks by selecting two distinct pairs of labels: 1 (Red Soil) and 2 (Cotton Crop). We split the dataset into 1,000 training, 124 validation, and 200 test instances, then reduced the 36 features to 30 features using principal component analysis.

The Fashion-MNIST dataset consists of 28×28 grayscale images of clothing [56]. The dataset consists of a total of 60,000 training and 10,000 test images, each labeled with one of 10 clothing classes. In our benchmarking, we focused on binary classification tasks by selecting two distinct pairs of labels: 0 & 1 and 0 & 2. We split the dataset into 2,000 training, 200 validation, and 200 test sets, then reduced the 28×28 features to 30 features using principal component analysis.

The Ionosphere dataset classifies radar returns from the ionosphere [57]. The dataset contains a total of 351 instances with 34 features labeled as 'good' or 'bad', where 'good' indicates radar returns that show evidence of structure in the ionosphere, and 'bad' indicates signals that pass through without detecting any structure. We divided the data between 251 training and 51 test sets.

4.1.2 Ansatz

We tested two different structures of the parameterized quantum circuit, also referred to as the ansatz, for the convolutional operations. The first one consists of two parameterized single-qubit rotations and a CNOT gate, as shown in Figure 5A [29]. This represents the simplest two-qubit ansatz. The second one is designed to express an arbitrary two-qubit unitary transformation. In general, any two-qubit unitary gate in the SU(4) group can be decomposed using

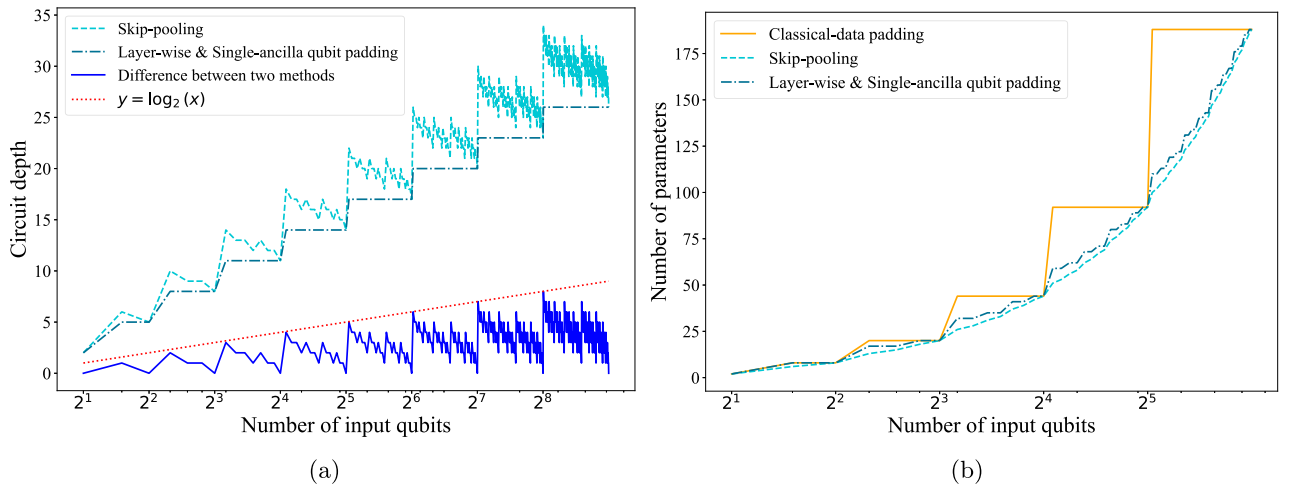


FIGURE 4
(A) Semi-log plot illustrating the difference in circuit depth between the skip pooling method and layer-wise and single-ancilla qubit padding method. The dashed line represents circuit depth in the skip pooling method, the dash-dot line denotes circuit depth in the layer-wise and single-ancilla qubit padding method, the solid line represents the difference between the two methods, and the dotted line corresponds to $\log_2 x$, provided as a guide to the eye. **(B)** Semi-log plot illustrating the number of parameters in the case of parameter sharing off for the classical data padding, skip pooling, and layer-wise and single-ancilla qubit padding methods. The solid line denotes the number of parameters for classical data padding, the dashed line represents the number of parameters for the skip pooling method, and the dash-dot line corresponds to the number of parameters for the layer-wise and single-ancilla qubit padding.

at most three CNOT gates and 15 elementary single-qubit gates [45, 58]. The quantum circuits shown in Figure 5C represent the parameterization of an arbitrary SU(4) gate. Figure 5B shows the pooling circuit, where two controlled rotations, $R_y(\theta_1)$ and $R_y(\theta_2)$, are applied, with each activated when the control qubit is 1 (filled circle) or 0 (open circle). Supplementary Appendix SA presents definitions of the quantum gates used in this study. We constructed two ansatz sets using a different combination of the convolutional and pooling circuits. Ansatz set 1 uses convolutional circuit 1 and a pooling circuit as shown in the figure, whereas ansatz set 2 uses convolution circuit 2 and pooling without a parameterized circuit. In the latter case, the pooling performs the partial trace operations without parameterized gates because the convolution circuit one is expressive enough to implement any two-qubit unitary operation.

4.2 Simulation without noise

In this section, we present numerical experimental results that evaluate the performance of QCNNs with various padding methods for binary classification tasks in a noiseless environment. Tables 1, 2 summarize the number of ancillary qubits, circuit depth, and number of parameters required for the naive and proposed methods. Classical data padding and skip pooling methods exhibit a significant difference in terms of the utilization of ancillary qubits. Specifically, classical data padding maximally uses ancillary qubits to apply a natural QCNN algorithm, whereas skip pooling does not use any ancillary qubits. However, skip pooling poses a potential drawback in the form of a potential increase in circuit depth, which affects computational power and runtime. Additionally, the use of fewer qubits results in the use of fewer parameters when they are not shared by each layer. However, the qubit padding method can apply an efficient QCNN algorithm with fewer qubits. Because the single-ancilla qubit padding method uses only one ancillary qubit, it does not incur additional circuit depth and offers the advantage of utilizing a slightly larger number of parameters than the skip-pooling method.

The simulation results were based on experiments using two different ansatz sets, denoted as ansatz set 1 and ansatz set 2. Table 3 lists the number of ancillary qubits, circuit depth, and parameters when applying different ansatz sets to the QCNN. All convolutional layers were used only once, and without loss of generality, the circuit depth was obtained by setting the convolution and pooling gate depths to 1. We obtained results from 10 repeated experiments with randomly initialized parameters for each ansatz set. The performance of the QCNN model was evaluated using the mean squared error (MSE) loss function. Model parameters were updated using the Adam optimizer [59]. The learning rate and batch size were set to 0.01 and 25, respectively. Training was performed with 10 epochs on the MNIST datasets.

The average accuracies and standard deviations of the binary classification task on the MNIST (labels 0 & 1 and 5 & 6) datasets obtained using ansatz set 1 are shown in Figure 6. Both single-ancilla qubit padding and skip pooling achieved superior accuracy across the two datasets. For example, for labels 0 & 1 in the MNIST dataset, single-ancilla qubit padding with shared parameters achieved an average accuracy of 91.75 (±2.57), whereas skip pooling showed an accuracy of 91.76 (±2.98). Such a trend was consistently observed in all other test cases, indicating the effectiveness of these methods in handling classification tasks with greater precision. However, the other padding methods, although still effective, did not attain the same levels of accuracy as single-ancilla qubit padding and skip pooling. The result obtained using ansatz set 2 is shown in Figure 7. Single-ancilla qubit padding

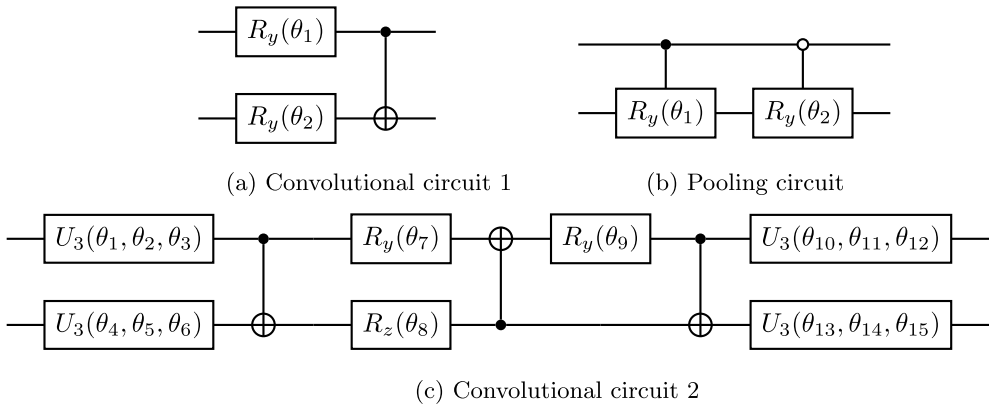


FIGURE 5
PQC for convolution and pooling operations. **(A)** and **(B)** are the convolutional and pooling circuits, respectively, that compose ansatz set 1. **(C)** Is the convolutional circuit that composes ansatz set 2. $R_i(\theta)$ is the rotation by θ around the i-axis of the Bloch sphere, and $U_3(\theta,\phi,\lambda)$ is an arbitrary single-qubit gate, which can be expressed as $U_3(\theta,\phi,\lambda) = R_z(\phi)R_x(-\pi/2)R_z(\theta)R_x(\pi/2)R_z(\lambda)$.

TABLE 1 Comparison of ancillary qubits, circuit depth, and the total number of parameters for classical data padding and skip pooling. $Y_i \coloneqq \lceil \frac{K}{2^{i-1}} \rceil \bmod 2$ is an odd number of qubits in the ith layer when divided by a power of two.

Padding methods		Classical data padding	Skip pooling
Ancillary qubits		$2^m - K$	0
Circuit depth		$\sum_{i=1}^{m}(2l_i+1) - l_m$	$\sum_{i=1}^{m}(2l_i+1) - l_m + \sum_{i=1}^{m-1} Y_i l_i$
Parameters	p-s on	$\sum_{i=1}^{m}(l_i+1)$	$\sum_{i=1}^{m}(l_i+1)$
	p-s off	$\sum_{i=1}^{m-1}(\lceil \frac{2^m}{2^{i-1}} \rceil (l_i+\frac{1}{2})) + l_m + 1$	$\sum_{i=1}^{m-1}(\lceil \frac{K}{2^{i-1}} \rceil (l_i+\frac{1}{2}) - \frac{1}{2}Y_i) + l_m + 1$

The notation "p-s on" and "p-s off" indicates whether parameter-sharing is enabled or disabled, respectively.

TABLE 2 Comparison of ancillary qubits, circuit depth, and the total number of parameters for layer-wise and single-ancilla qubit padding. $Y_i \coloneqq \lceil \frac{K}{2^{i-1}} \rceil \bmod 2$ is an odd number of qubits in the ith layer when divided by a power of two.

Padding methods		Layer-wise qubit padding	Single-ancilla qubit padding
Ancillary qubits		$\sum_{i=1}^{m-1} Y_i$	1
Circuit depth		$\sum_{i=1}^{m}(2l_i+1) - l_m$	$\sum_{i=1}^{m}(2l_i+1) - l_m$
Parameters	p-s on	$\sum_{i=1}^{m}(l_i+1)$	$\sum_{i=1}^{m}(l_i+1)$
	p-s off	$\sum_{i=1}^{m-1}((\lceil \frac{K}{2^{i-1}} \rceil + Y_i)(l_i+\frac{1}{2})) + l_m + 1$	$\sum_{i=1}^{m-1}((\lceil \frac{K}{2^{i-1}} \rceil + Y_i)(l_i+\frac{1}{2})) + l_m + 1$

The notation "p-s on" and "p-s off" indicates whether parameter-sharing is enabled or disabled, respectively.

and skip pooling consistently achieved higher performance. For example, for labels 5 & 6 in the MNIST dataset, single-ancilla qubit padding achieved an average accuracy of 93.07 (±1.14) and skip pooling achieved 94.59 (±0.62), showing better results with fewer qubits. Although skip pooling is efficient in terms of qubit usage and performance, it results in a deeper circuit that may be more susceptible to noise, particularly in existing noisy quantum devices. In contrast, single-ancilla qubit padding is more robust to noise, potentially making it more suitable for implementation on quantum devices. This will be demonstrated in the following section.

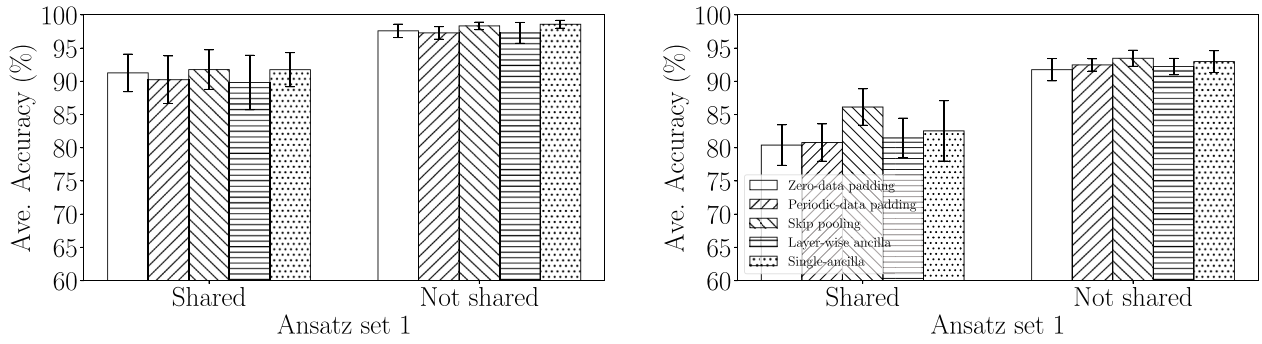
4.3 Simulation with noise

We conducted an additional experiment to evaluate the impact of noise in a quantum computing environment on the performance of QCNN algorithms. In particular, we considered the influence of circuit depth on error accumulation in quantum computation by comparing performance between the single-ancilla qubit padding and skip pooling methods. The noise simulations focused on types of noise that closely relate to circuit depth, and state preparation and measurement errors (SPAM) were excluded, as they were beyond the scope of our interest in this study. We considered various types

10.3389/fphy.2025.1529188

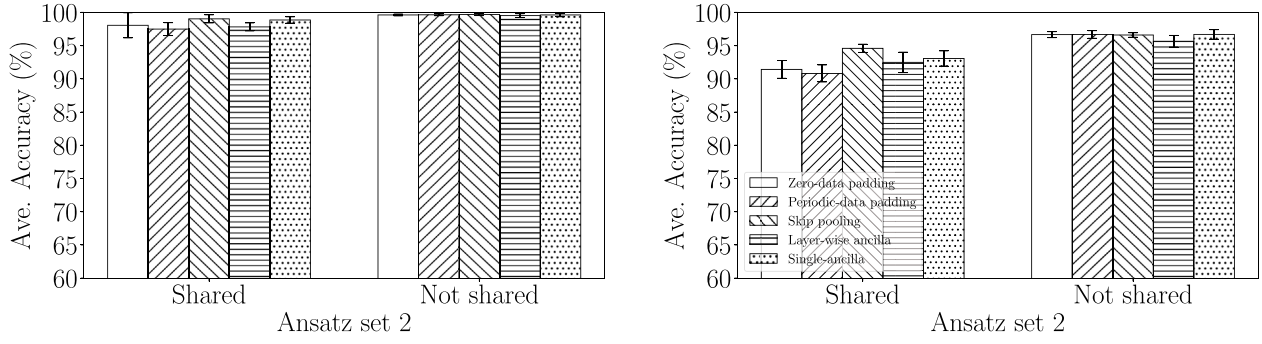
TABLE 3 Comparison of the number of ancillary qubits, circuit depth, and total number of parameters for each padding method with different ansatz sets based on five initial input qubits. The notation 'p-s on' and 'p-s off' indicates whether parameter-sharing is enabled or disabled, respectively.

Padding methods	Ancillary qubits	Circuit depth	Parameters (p-s on)		Parameters (p-s off)	
			ansatz set 1	ansatz set 2	ansatz set 1	ansatz set 2
Classical data padding	3	8	12	45	40	195
Skip pooling	0	10	12	45	26	135
Layer-wise qubit padding	2	8	12	45	34	165
Single-ancilla qubit padding	1	8	12	45	34	165



(a) MNIST 0 & 1 (b) MNIST 5 & 6

FIGURE 6
QCNN model performance with various padding methods constructed using ansatz set 1. The bar chart shows the average accuracy and standard deviation for **(A)** the MNIST 0 & 1 dataset, **(B)** and the MNIST 5 & 6 dataset. The x-axis differentiates between the case of parameter-sharing on and parameter-sharing off. Unfilled bars represent zero-data padding, forward slash bars represents periodic-data padding, backslash bars represents skip Pooling, horizontal dash bars represents layer-wise ancilla, and dots bars represent single-ancilla.



(a) MNIST 0 & 1 (b) MNIST 5 & 6

FIGURE 7
QCNN model performance with various padding methods constructed using ansatz set 2. The bar chart shows the average accuracy and standard deviation for **(A)** the MNIST 0 & 1 dataset, and **(B)** the MNIST 5 & 6 dataset. The x-axis differentiates between the case of parameter-sharing on and parameter-sharing off. Unfilled bars represent zero-data padding, forward slash bars represents periodic-data padding, backslash bars represents skip pooling, horizontal dash bars represents layer-wise ancilla, and dots bars represent single-ancilla.

TABLE 4 Average error rates for the IBM quantum device, *ibmq_jakarta*, utilized in the noisy simulation.

1-Qubit depolarizing	2-Qubit depolarizing	1-Qubit gate length	2-Qubit gate length	T_1	T_2
0.0004	0.0126	35.56 (ns)	327.11 (ns)	128.43 (us)	33.85 (us)

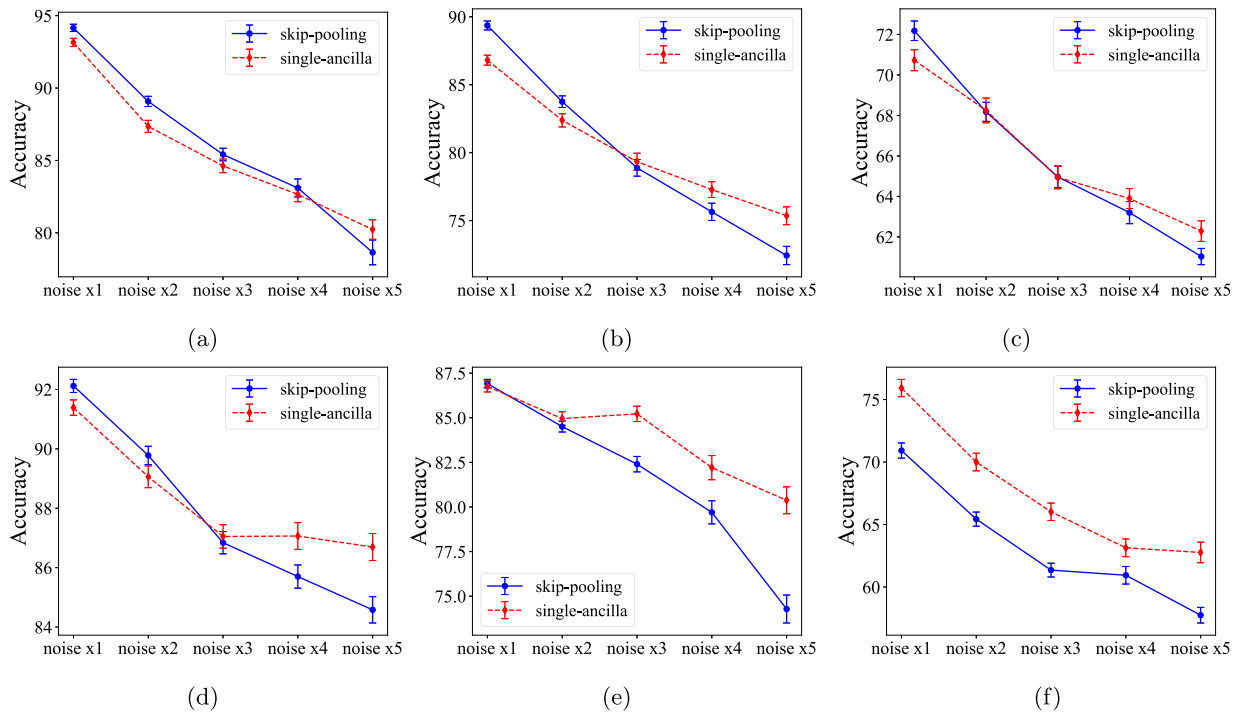


FIGURE 8
Test results of QCNN model with skip pooling and single-ancilla qubit padding constructed with ansatz set 2. The depolarizing errors and gate lengths were increased from their original values (x1) up to a maximum of (x5). The solid blue line represents skip pooling, whereas the dashed red line denotes single-ancilla qubit padding. The mean and standard error were obtained from 100 repeated experiments with parameters initialized randomly. **(A)** Shows the average accuracy and standard error of classification for 0 & 1 in the MNIST dataset, **(B)** shows the average accuracy and standard error of classification for 5 & 6 in the MNIST dataset, and **(C)** shows the average accuracy and standard error of classification for 1 & 2 in the Landsat satellite dataset. **(D)** Shows the average accuracy and standard error of classification for 0 & 1 in the Fashion-MNIST dataset, **(E)** shows the average accuracy and standard error of classification for 0 & 2 in the Fashion-MNIST dataset. **(F)** Shows the average accuracy and standard error of classification for the Ionosphere dataset.

of noise in quantum devices, such as depolarization errors, gate lengths, and thermal relaxation, but not the physical connectivity of qubits. Supplementary Appendix SC provides details of the noise circuits used in this experiment.

We used the noise parameters observed from IBMQ Jakarta, a real quantum device, to simulate a realistic noise model. Table 4 lists the average error rates observed on IBMQ Jakarta. To evaluate the influence of a range of noise levels, we performed experiments with depolarizing errors and gate lengths ranging from one to five times the original values. This multiplication was applied consistently over 100 repeated experiments with randomly initialized parameters, and the results are shown in Figure 8. For the MNIST, Landsat satellite, and Fashion-MNIST dataset, as noise levels increased, the single-ancilla qubit padding method showed less accuracy degradation compared to skip pooling. In the case of the Ionosphere dataset, the single-ancilla method consistently outperformed skip pooling across all tested noise levels. These results demonstrate that the proposed method provides an optimal solution for constructing an efficient QCNN architecture with minimal resource overhead, maintaining robust performance despite noise and imperfections.

5 Extension to multi-qubit quantum convolutional operations

An arbitrary unitary operation acting on n qubits, which is an element in the $SU(2^n)$ group, can be specified using $4^n - 1$ real parameters. This implies that the number of elementary gates required to implement an arbitrary n-qubit unitary operation increases exponentially with n. Therefore, minimizing the number of qubits involved in a quantum convolutional operation is beneficial in practice. This is a primary motivation for designing quantum convolutional operations that act on only two qubits, as considered in this study. However, quantum convolutional operations can theoretically act on any n number of qubits. In general, the quantum circuit depth of any given convolutional layer, denoted by l, is greater

than or equal to n (i.e., $l \geq n$), and equality is satisfied only when the quantum convolutional layer consists of an qubits where $a \in \mathbb{Z}_+$ is a positive integer (i.e., m is a positive-integer multiple of n). Therefore, if the number of qubits in a quantum convolutional layer, denoted by m, is not an integer multiple of n, the circuit depth of the given convolutional layer can be minimized at the cost of introducing $n' < n$ ancilla qubits such that $m + n' = an$.

For example, consider a quantum convolutional layer consisting of $m = 7$ qubits where each quantum convolutional operation acts on $n = 3$ qubits. By utilizing $n' = 2$ ancilla qubits, the circuit depth of the quantum convolutional layer can be minimized to three.

6 Conclusion

In this study, we designed a QCNN architecture that can handle arbitrary data dimensions. Using qubit padding, we optimized the allocation of quantum resources through the efficient use of ancillary qubits. Our method not only reduces the number of ancillary qubits, but also optimizes the circuit depth to construct an efficient QCNN architecture. This results in an optimal solution that is computationally efficient and robust against noise. We benchmarked the performance of our QCNN using both naive methods and the proposed methods on various datasets for binary classification. In simulations without noise, both skip pooling and our proposed single-ancilla qubit padding method achieved high accuracy in most cases. We also compared performance between single-ancilla qubit padding and skip pooling in a noisy simulation, using the noise model and parameters of an IBM quantum device. Our results demonstrate that as the noise level increases, single-ancilla qubit padding exhibits less performance degradation and lower sensitivity to variation. Therefore, the proposed method serves as a fundamental building block for the effective application of QCNN to real-world data of arbitrary input dimension.

The main focus of our study is on the analysis of classical data using QML, reflecting the prevalence of classical datasets in modern society. Nevertheless, data can also be intrinsically quantum [60]. In such cases, classical dimensionality reduction or data padding to adjust the number of input qubits to a power of two is not feasible. However, the single-ancilla qubit padding method can be easily adapted and remain valuable.

As a final remark, our work aims to guide users in selecting the optimal QCNN circuit design with respect to their specific requirements and system environment. We do not intend to rule out the skip-pooling method; it remains a viable option if increasing the number of qubits is more challenging than increasing the circuit depth. Conversely, if minimizing circuit depth is critical and adding an extra qubit is relatively easy, then the single-ancilla method would be preferable. Additionally, if the task at hand requires a higher model complexity, the single-ancilla method with parameter-sharing off can be used.

Data availability statement

The original contributions presented in the study are included in the article/Supplementary Material, further inquiries can be directed to the corresponding author.

Author contributions

CL: Formal Analysis, Investigation, Methodology, Software, Validation, Writing–original draft, Writing–review and editing. IA: Investigation, Software, Writing–review and editing. DK: Investigation, Methodology, Software, Writing–original draft. JL: Investigation, Software, Writing–original draft. SP: Investigation, Methodology, Software, Writing–original draft, Writing–review and editing. J-YR: Investigation, Methodology, Software, Writing–original draft. DP: Conceptualization, Formal Analysis, Funding acquisition, Methodology, Supervision, Writing–original draft, Writing–review and editing.

Funding

The author(s) declare that financial support was received for the research, authorship, and/or publication of this article. This research was supported by Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (No. 2019-0-00003, Research and Development of Core Technologies for Programming, Running, Implementing and Validating of Fault-Tolerant Quantum Computing System), the Yonsei University Research Fund of 2024 (2024-22-0147), the National Research Foundation of Korea (2022M3E4A1074591, 2023M3K5A1094813), the KIST Institutional Program (2E32941-24-008), and the Ministry of Trade, Industry, and Energy (MOTIE), Korea, under the Industrial Innovation Infrastructure Development Project (Project No. RS-2024-00466693).

Conflict of interest

Author J-YR was employed by Norma Inc.

The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The author(s) declare that no Generative AI was used in the creation of this manuscript.

Publisher's note

Supplementary material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fphy.2025.1529188/full#supplementary-material

References

1. Jain AK, Mao J, Mohiuddin KM. Artificial neural networks: a tutorial. *Computer* (1996) 29(3):31–44. doi:10.1109/2.485891

2. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. *Adv Neural Inf Process Syst* (2017) 30.

3. Yann LC, Bernhard B, John SD, Henderson D, Howard RE, Hubbard W, et al. Backpropagation applied to handwritten zip code recognition. *Neural Comput* (1989) 1(4):541–51. doi:10.1162/neco.1989.1.4.541

4. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. *Proc IEEE* (1998) 86(11):2278–324. doi:10.1109/5.726791

5. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. *Adv Neural Inf Process Syst* (2012) 25.

6. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016). p. 770–8.

7. Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014). p. 580–7.

8. Li H, Lin Z, Shen X, Brandt J, Hua G. A convolutional neural network cascade for face detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015) 5325–34.

9. Tajbakhsh N, Shin JY, Gurudu SR, Todd Hurst R, Kendall CB, Gotway MB, et al. Convolutional neural networks for medical image analysis: full training or fine tuning? *IEEE Trans Med Imaging* (2016) 35(5):1299–312. doi:10.1109/tmi.2016.2535302

10. Jacob B, Wittek P, Pancotti N, Rebentrost P, Wiebe N, Lloyd S. Quantum machine learning. *Nature* (2017) 549(7671):195–202. doi:10.1038/nature23474

11. Schuld M, Killoran N. Quantum machine learning in feature hilbert spaces. *Phys Rev Lett* (2019) 122(4):040504. doi:10.1103/physrevlett.122.040504

12. Schuld M, Sinayskiy I, Petruccione F. An introduction to quantum machine learning. *Contemp Phys* (2015) 56(2):172–85. doi:10.1080/00107514.2014.964942

13. Lloyd S, Mohseni M, Rebentrost P. Quantum algorithms for supervised and unsupervised machine learning. *arXiv preprint arXiv:1307.0411* (2013). doi:10.48550/arXiv.1307.041

14. Preskill J. Quantum computing in the nisq era and beyond. *Quantum* (2018) 2(79):79. doi:10.22331/q-2018-08-06-79

15. Bharti K, Cervera-Lierta A, Kyaw TH, Haug T, Alperin-Lea S, Anand A, et al. Noisy intermediate-scale quantum algorithms. *Rev Mod Phys* (2022) 94(1):015004. doi:10.1103/revmodphys.94.015004

16. Benedetti M, Lloyd E, Sack S, Fiorentini M. Parameterized quantum circuits as machine learning models. *Quan Sci Technology* (2019) 4(4):043001. doi:10.1088/2058-9565/ab4eb5

17. Sim S, Johnson PD, Aspuru-Guzik A. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Adv Quan Tech* (2019) 2(12):1900070. doi:10.1002/qute.201900070

18. Cerezo M, Arrasmith A, Babbush R, Benjamin SC, Endo S, Fujii K, et al. Variational quantum algorithms. *Nat Rev Phys* (2021) 3(9):625–44. doi:10.1038/s42254-021-00348-9

19. Huang H-Y, Kueng R, Preskill J. Information-theoretic bounds on quantum advantage in machine learning. *Phys Rev Lett* (2021) 126(19):190505. doi:10.1103/physrevlett.126.190505

20. Aharonov D, Cotler J, Xiao-Liang Q. Quantum algorithmic measurement. *Nat Commun* (2022) 13(1):887. doi:10.1038/s41467-021-27922-0

21. Schuld M, Sweke R, Meyer JJ. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Phys Rev A* (2021) 103(3):032430. doi:10.1103/physreva.103.032430

22. Caro MC, Gil-Fuster E, Meyer JJ, Eisert J, Sweke R. Encoding-dependent generalization bounds for parametrized quantum circuits. *Quantum* (2021) 5:582. doi:10.22331/q-2021-11-17-582

23. Thanasilp S, Wang S, Cerezo M, Holmes Z. Exponential concentration and untrainability in quantum kernel methods,. *arXiv preprint arXiv:2208.11060*(2022). doi:10.1038/s41467-024-49287-w

24. Abbas A, Sutter D, Zoufal C, Lucchi A, Figalli A, Woerner S. The power of quantum neural networks. *Nat Comput Sci* (2021) 1(6):403–9. doi:10.1038/s43588-021-00084-1

25. Caro MC, Huang H-Y, Cerezo M, Sharma K, Sornborger A, Cincio L, et al. Generalization in quantum machine learning from few training data. *Nat Commun* (2022) 13(1):4919. doi:10.1038/s41467-022-32550-3

26. McClean JR, Boixo S, Smelyanskiy VN, Babbush R, Neven H. Barren plateaus in quantum neural network training landscapes. *Nat Commun* (2018) 9(1):4812. doi:10.1038/s41467-018-07090-4

27. Larocca M, Thanasilp S, Wang S, Sharma K, Jacob B, Coles PJ, et al. A review of barren plateaus in variational quantum computing. *arXiv preprint arXiv:2405.00781* (2024). doi:10.48550/arXiv.2405.00781

28. Holmes Z, Sharma K, Cerezo M, Coles PJ. Connecting ansatz expressibility to gradient magnitudes and barren plateaus. *PRX Quan* (2022) 3:010313. doi:10.1103/prxquantum.3.010313

29. Grant E, Benedetti M, Cao S, Hallam A, Lockhart J, Stojevic V, et al. Hierarchical quantum classifiers. *npj Quan Inf* (2018) 4(1):65. doi:10.1038/s41534-018-0116-9

30. Pesah A, Cerezo M, Wang S, Tyler V, Sornborger AT, Coles PJ. Absence of barren plateaus in quantum convolutional neural networks. *Phys Rev X* (2021) 11(4):041011. doi:10.1103/physrevx.11.041011

31. Cong I, Choi S, Lukin MD. Quantum convolutional neural networks. *Nat Phys* (2019) 15(12):1273–8. doi:10.1038/s41567-019-0648-8

32. Hur T, Kim L, Park DK. Quantum convolutional neural network for classical data classification. *Quan Machine Intelligence* (2022) 4(1):3. doi:10.1007/s42484-021-00061-x

33. Kim J, Huh J, Park DK. Classical-to-quantum convolutional neural network transfer learning. *Neurocomputing* (2023) 555:126643. doi:10.1016/j.neucom.2023.126643

34. Lourens M, Sinayskiy I, Park DK, Blank C, Petruccione F. Hierarchical quantum circuit representations for neural architecture search. *npj Quan Inf* (2023) 9(1):79. doi:10.1038/s41534-023-00747-z

35. Oh H, Park DK. Quantum support vector data description for anomaly detection. *Machine Learn Sci Technology* (2024) 5:035052. doi:10.1088/2632-2153/ad6be8

36. Chen G, Chen Q, Long S, Zhu W, Yuan Z, Wu Y. Quantum convolutional neural network for image classification. *Pattern Anal Appl* (2023) 26(2):655–67. doi:10.1007/s10044-022-01113-z

37. Smaldone AM, Kyro GW, Batista VS. Quantum convolutional neural networks for multi-channel supervised learning. *Quan Machine Intelligence* (2023) 5(2):41. doi:10.1007/s42484-023-00130-3

38. Banchi L, Pereira J, Pirandola S. Generalization in quantum machine learning: a quantum information standpoint. *PRX Quan* (2021) 2:040321. doi:10.1103/prxquantum.2.040321

39. Goodfellow I, Bengio Y, Courville A. *Deep learning*. MIT Press (2016). Available from: http://www.deeplearningbook.org.

40. Zhang C, Bengio S, Hardt M, Benjamin R, Vinyals O. Understanding deep learning (still) requires rethinking generalization. *Commun ACM* (2021) 64(3):107–15. doi:10.1145/3446776

41. Lu Y, Lu J. A universal approximation theorem of deep neural networks for expressing probability distributions. *Adv Neural Inf Process Syst* (2020) 33: 3094–105.

42. Ruder S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747* (2016). doi:10.48550/arXiv.1609.04747

43. Mitarai K, Negoro M, Kitagawa M, Fujii K. Quantum circuit learning. *Phys Rev A* (2018) 98(3):032309. doi:10.1103/physreva.98.032309

44. Schuld M, Bergholm V, Gogolin C, Izaac J, Killoran N. Evaluating analytic gradients on quantum hardware. *Phys Rev A* (2019) 99(3):032331. doi:10.1103/physreva.99.032331

45. MacCormack I, Delaney C, Galda A, Aggarwal N, Narang P. Branching quantum convolutional neural networks. *Phys Rev Res* (2022) 4(1):013117. doi:10.1103/physrevresearch.4.013117

46. Mari A, Bromley TR, Izaac J, Schuld M, Killoran N. Transfer learning in hybrid classical-quantum neural networks. *Quantum* (2020) 4:340. doi:10.22331/q-2020-10-09-340

47. Li Y, Wang Z, Han R, Shi S, Li J, Shang R, et al. Quantum recurrent neural networks for sequential learning. *Neural Networks* (2023) 166:148–61. doi:10.1016/j.neunet.2023.07.003

48. Lloyd S, Weedbrook C. Quantum generative adversarial learning. *Phys Rev Lett* (2018) 121:040502. doi:10.1103/physrevlett.121.040502

49. Dallaire-Demers P-L, Killoran N. Quantum generative adversarial networks. *Phys Rev A* (2018) 98:012324. doi:10.1103/physreva.98.012324

50. Yen-Chi Chen S, Wei T-C, Zhang C, Yu H, Yoo S. Hybrid quantum-classical graph convolutional network. *arXiv preprint arXiv:2101.06189* (2021). doi:10.48550/arXiv.2101.06189

51. Tüysüz C, Rieger C, Novotny K, Demirköz B, Dobos D, Potamianos K, et al. Hybrid quantum classical graph neural networks for particle track reconstruction. *Quan Machine Intelligence* (2021) 3(2):29. doi:10.1007/s42484-021-00055-9

52. Park G, Huh J, Park DK. Variational quantum one-class classifier. *Machine Learn Sci Technology* (2023) 4(1):015006. doi:10.1088/2632-2153/acafd5

53. Jaderberg B, Anderson LW, Xie W, Albanie S, Kiffner M, Jaksch D. Quantum self-supervised learning. *Quan Sci Technology* (2022) 7(3):035005. doi:10.1088/2058-9565/ac6825

54. LeCun Y, Cortes C, Burges CJ. Mnist handwritten digit database. *ATT Labs* (2010). Available from: http://yann.lecun.com/exdb/mnist,2.

55. Srinivasan A. Statlog (Landsat satellite). *UCI Machine Learn Repository* (1993). doi:10.24432/C55887

56. Xiao H, Rasul K, Vollgraf R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR* (2017) 07747. abs/1708. doi:10.48550/arXiv.1708.07747

57. Sigillito V, Baker K. Ionosphere. In: *UCI machine learning repository* (1989). doi:10.24432/C5W01B

58. Vatan F, Williams C. Optimal quantum circuits for general two-qubit gates. *Phys Rev A* (2004) 69(3):032315. doi:10.1103/physreva.69.032315

59. Diederik PK, Jimmy B. Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). doi:10.48550/arXiv.1412.6980

60. Cerezo M, Verdon G, Huang H-Y, Cincio L, Coles PJ. Challenges and opportunities in quantum machine learning. *Nat Comput Sci* (2022) 2:567–76. doi:10.1038/s43588-022-00311-3