Article

Not peer-reviewed version

Leveraging Data Locality in Quantum Convolutional Classifiers

Mingyoung Jeng , Alvir Nobel , Vinayak Jha , David Levy , Dylan Kneidel , Manu Chaudhary , Ishraq Islam , Audrey Facer , Manish Singh , Evan Baumgartner , Eade Vanderhoof , Abina Arshad , Esam El-Araby *

Posted Date: 18 March 2024

doi: 10.20944/preprints202403.1060.v1

Keywords: quantum computing; convolutional neural networks; quantum machine learning; variational quantum algorithms

Article

Leveraging Data Locality in Quantum Convolutional Classifiers

Mingyoung Jeng, Alvir Nobel, Vinayak Jha, David Levy, Dylan Kneidel, Manu Chaudhary, Ishraq Islam, Audrey Facer, Manish Singh, Evan Baumgartner, Eade Vanderhoof, Abina Arshad, and Esam El-Araby *

Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, KS 66045, USA; mingyoungjeng@ku.edu (M.J.); islam.alvir@ku.edu (A.N.); vinayakjha@ku.edu (V.J.); david.levy@ku.edu (D.L.); dckneidel@ku.edu (D.K.); manu.chaudhary@ku.edu (M.C.); ishraq@ku.edu (I.I.); audfacer@ku.edu (A.F.); singhmanish5208@ku.edu (M.S.); evan.baum@ku.edu (E.B.); eade.vanderhoof@ku.edu (E.V.); abina.arshad@ku.edu (A.A.)

* Correspondence: esam@ku.edu

Abstract: Quantum computing (QC) has opened the door to advancements in machine learning (ML) tasks that are currently implemented in the classical domain. Convolutional neural networks (CNNs) are classical ML architectures that exploit data locality and possess a simpler structure than fully-connected multi-layer perceptrons (MLPs) without compromising the accuracy of classification. However, the concept of preserving data locality is usually overlooked in the existing quantum counterparts of CNNs, particularly for extracting multi-features in multidimensional data. In this paper, we present a multidimensional quantum convolutional classifier (MQCC) that performs multidimensional and multi-feature quantum convolution with average and Euclidean pooling and thus adapting the CNN structure to a variational quantum algorithm (VQA). The experimental work was conducted using multidimensional data to validate the correctness and demonstrate the scalability of the proposed method utilizing both noisy and noise-free quantum simulations. We evaluated the MQCC model with reference to reported work on state-of-the-art quantum simulators from IBM Quantum and Xanadu using a variety of standard ML datasets. The experimental results showed favorable characteristics of our proposed techniques compared to existing work with respect to a number of quantitative metrics such as the number of training parameters, cross-entropy loss, classification accuracy, circuit depth, and quantum gate count.

Keywords: convolutional neural networks; quantum computing; variational quantum algorithms; quantum machine learning

---

1. Introduction

The choice of an appropriate machine learning model for specific applications requires consideration of the size of the model since it is linked to the performance [1]. Considering the aforementioned factor, convolutional neural networks (CNNs) are preferable to multi-layer perceptrons (MLPs) because of their smaller size and reduced training time while maintaining a high accuracy [2,3]. Preserving the spatio-temporal locality of data allows CNNs to reduce unnecessary data connections and therefore reduces their memory requirements [2,3]. This phenomenon reduces the number of required training parameters and thus incurs less training time [2,3].

In the context of quantum computing, great emphasis has been given to quantum-based machine learning, and, in recent years, various techniques have been devised to develop this field [4]. The contemporary quantum machine learning (QML) techniques can be considered as hybrid quantum-classical variational algorithms [5–9]. Generally, variational quantum algorithms (VQAs) utilize parameterized rotation gates in fixed quantum circuit structures usually called ansatz and are optimized using classical techniques like gradient descent [5–9]. However, like MLPs, preserving data locality is challenging for QML algorithms. For instance, the multidimensionality of input datasets is ignored in contemporary QML algorithms, and are flattened into one-dimensional arrays [5–9]. Furthermore, the absence of a generalizable technique for quantum convolution limits the capability of QML algorithms to directly adapt CNN structures.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 March 2024 doi:10.20944/preprints202403.1060.v1

In this work, we present a multidimensional quantum convolutional classifier (MQCC) to address the shortcomings of the existing CNN implementations in reconciling the locality of multidimensional input data. The proposed VQA technique leverages quantum computing to reduce the number of training parameters and time complexity compared to classical CNN models. Similar to the CNN structures, MQCC contains a sequence of convolution and pooling layers for multi-feature extraction from multidimensional input data and a fully-connected layer for classification.

The subsequent sections of this paper are organized in the following structure. Section 2 discusses fundamental background information regarding different basic and complex quantum operations. Section 3 highlights existing works that are related to the proposed techniques. The proposed methodology is introduced in Section 4 with details given to the constituent parts. The experimental results and the explanation of the used verification metrics are presented in 5. Further discussion about the obtained results are provided in Section 6. Finally, Section 7 concludes this work with potential future directions.

2. Background

In this section, fundamental information pertaining to quantum computing and quantum machine learning is provided. Here we present the quantum gates and operations that are utilized in the proposed multidimensional quantum convolutional classifier (MQCC).

2.1. Quantum Bits and States

A single quantum bit, or qubit, can be represented by a normalized vector $|\psi\rangle$ with $N = 2^1$ elements, called a statevector, see (1). The state of a qubit is represented as a point on the surface of the Bloch sphere [10], as shown in Figure 1. To preserve the normalization property of a statevector, a qubit's state can only be expressed as a point on the sphere's surface [11].

$$|\psi\rangle = c_0|0\rangle + c_1|1\rangle = \begin{bmatrix} c_0 \\ c_1 \end{bmatrix}, \text{ where } c_0^2 + c_1^2 = 1 \quad (1)$$

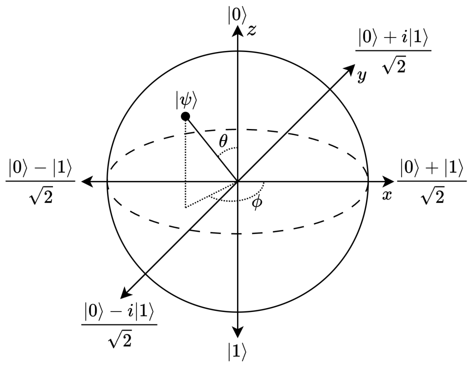


Figure 1. The Bloch Sphere.

For n qubits, the statevector $|\psi\rangle$ has $N = 2^n$ elements . As shown in (2), each element $c_j \in \mathbb{C}$ of $|\psi\rangle$ represents the amplitude/coefficient of the j^{th} entry in $|\psi\rangle$, or the basis state $|j\rangle$ [11].

 doi:10.20944/preprints202403.1060.v1

$$|\psi\rangle = \sum_{j=0}^{N-1} c_j \cdot |j\rangle = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_j \\ \vdots \\ c_{N-2} \\ c_{N-1} \end{bmatrix}, \text{where } \sum_{j=0}^{N-1} |c_j|^2 = 1 \text{ , and } 0 \leq j < (N = 2^n) \tag{2}$$

2.2. *Quantum Gates*

Quantum gates are reversible operations on qubits and can be mathematically represented by unitary matrices [11]. Serial operations of quantum gates can be represented using matrix multiplication, while parallel operations can be represented using the tensor product [11]. In this section, we briefly discuss single- and multi-qubit quantum gates relevant to the proposed MQCC.

2.2.1. Rotation Gates

The rotation gates, $R_x(\theta)$, $R_y(\theta)$, and $R_z(\phi)$, are parameterized versions of the Pauli single-qubit gates, shown in (3), (4), and (5), respectively [12]. In QML, rotation gate parameters are trained to produce better models [5,7].

$$\mathbf{R}_x(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{3}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{4}$$

$$\mathbf{R}_z(\phi) = \begin{bmatrix} e^{-\frac{i\phi}{2}} & 0 \\ 0 & e^{\frac{i\phi}{2}} \end{bmatrix} \tag{5}$$

2.2.2. Hadamard Gate

The Hadamard gate is a fundamental quantum gate that puts the qubit into a superposition of two basis states, the $|0\rangle$ and $|1\rangle$ states, see (6) and Figure 2 [11]. Similarly, by applying Hadamard gates on n qubits independently, an n-qubit superposition can be created that comprises of 2^n basis states, as shown in (7) and Figure 3.

$$\mathbf{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{6}$$

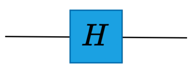


Figure 2. Hadamard gate diagram

$$H^{\otimes n} \cdot |0\rangle^{\otimes n} = \frac{1}{\sqrt{2^n}} \sum_{j=0}^{2^n-1} |j\rangle \tag{7}$$

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 March 2024 doi:10.20944/preprints202403.1060.v1

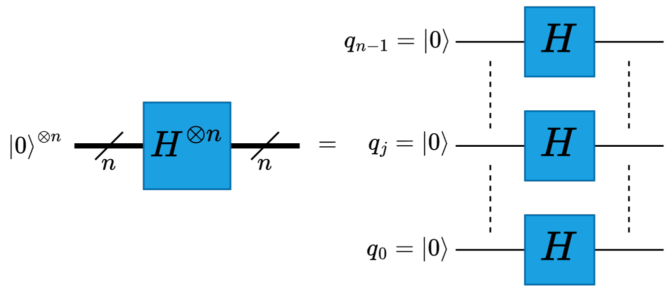


Figure 3. Example parallel operation notation.

2.2.3. Controlled-NOT (CNOT) Gate

CNOT gates can form a fundamental set of basis gates in conjunction with single-qubit rotation gates [12], see (8) and Figure 4. CNOT gates can be used to entangle two qubits, where a measurement of one qubit can provide information about the second qubit [12].

$$\mathbf{CNOT} = \mathbf{cX} = (|0\rangle\langle 0| \otimes I) + (|1\rangle\langle 1| \otimes X) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{8}$$

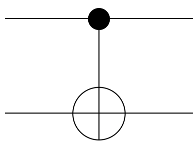


Figure 4. Controlled-NOT gate diagram.

All gates, single- or multi-qubit, can be extended to have control qubit(s), shown for a general U in (9) and Figure 5 [12].

$$\mathbf{cU} = (|0\rangle\langle 0| \otimes I) + (|1\rangle\langle 1| \otimes \mathrm{U}) = \begin{bmatrix} I & 0 \\ 0 & \mathrm{U} \end{bmatrix} \tag{9}$$

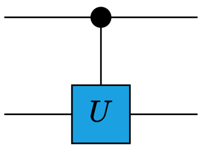


Figure 5. Controlled-U gate diagram.

When a general multi-qubit gate is extended to have more control qubits, it becomes an n-qubit multiple-controlled U (MCU) gate. In the context of CNOT gates, the operation becomes a multiple-controlled-not (MCX) gate [13]. For an n-qubit MCX gate with $n-1$ control qubits and 1 target qubit, see (10), the depth of an MCX gate with the addition of a single extra qubit can be found in (11) [13].

$$\mathbf{MCX}(n) = \sum_{i=0}^{2^{n-1}-2} (|i\rangle\langle i| \otimes I) + (|2^{n-1}-1\rangle\langle 2^{n-1}-1| \otimes X) \tag{10}$$

$$\Delta_{\mathrm{MCX}}(n) \leq 48n - 196 = \mathcal{O}(n) \text{ for large } n \tag{11}$$

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 March 2024 doi:10.20944/preprints202403.1060.v1

The most general controlled gate is the *multiplexer*, see (12), which defines a quantum operation to be applied on n_{target} qubits for each state permutation of some control qubit(s), $n_{control}$ [12]. In quantum circuits, the *square box* notation is used to denote a multiplexer operation, as shown in Figure 6 [12].

$$\mathbf{U}_{mux} = \sum_{i=0}^{2^{n_c}-1} |i\rangle\langle i| \otimes \mathrm{U}_i = \begin{bmatrix} \mathrm{U}_0 & & \\ & \ddots & \\ & & \mathrm{U}_{2^{n_{control}}-1} \end{bmatrix} \begin{matrix} \updownarrow 2^{n_{target}} \\ \vdots \\ \updownarrow 2^{n_{target}} \end{matrix} \Bigg] 2^{n_{target}+n_{control}} \quad (12)$$

Here, U_i is a matrix of size $2^{n_{target}} \times 2^{n_{target}}$ defining the unitary operations/gates applied on each data qubit for the corresponding values i, where, $0 < i < 2^{n_{control}-1}$.

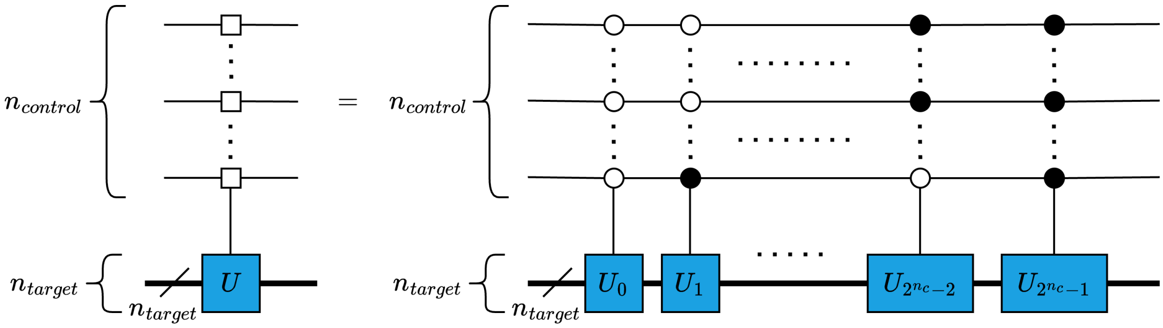


Figure 6. The *square box* notation for a quantum multiplexer. [12]

2.2.4. SWAP Gate

In a quantum circuit, SWAP gates are applied to interchange the position of two qubits, as shown in (13) and Figure 7. The SWAP gate can be decomposed into three CNOT gates [11], see Figure 7.

$$\mathbf{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (13)$$

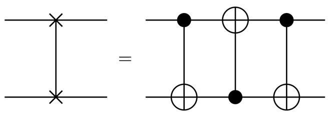
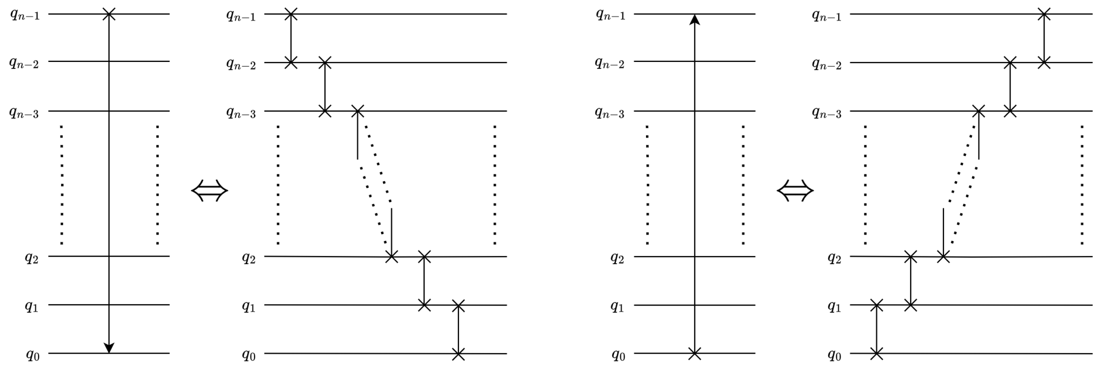


Figure 7. Swap gate diagram and decomposition.

2.2.5. Quantum Perfect-Shuffle Permutation (PSP)

The quantum perfect-shuffle permutations (PSPs) are operations that perform a cyclical rotation of given input qubits and can be implemented using SWAP gates. Figure 8 shows the rotate-left (RoL) and rotate-right (RoR) PSP operations [14] with $(n-1)$ SWAP operations, or $3(n-1)$ equivalent CNOT operations.



(a) RoL : $|q_{n-1}q_{n-2}\cdots q_0\rangle \rightarrow |q_{n-2}\cdots q_0 q_{n-1}\rangle$ **(b)** RoR : $|q_{n-1}\cdots q_1 q_0\rangle \rightarrow |q_0 q_{n-1}\cdots q_1\rangle$

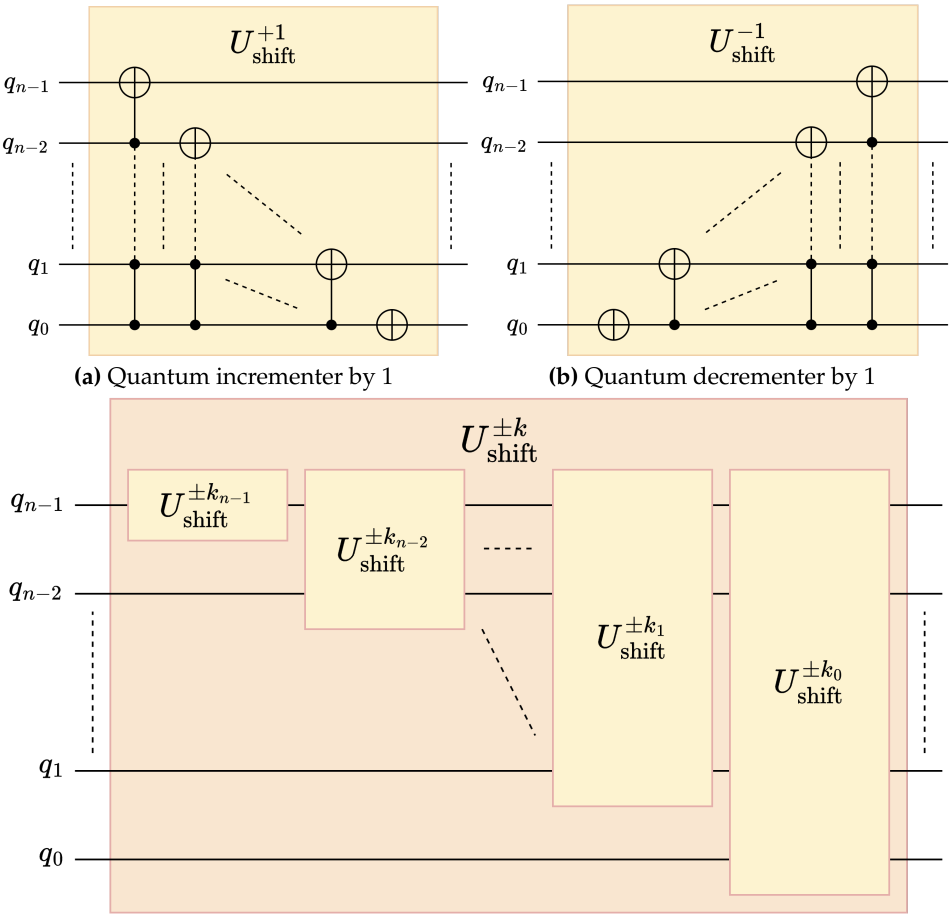
Figure 8. Quantum Perfect-Shuffle Permutations. [14]

2.2.6. Quantum Shift Operation

The quantum shift operation also performs a cyclic rotation of the *basis states* [15] similar to the quantum perfect-shuffle operations. Here, the basis states of the n-qubit state $|\psi\rangle$ are shifted by $+k$ or $-k$ positions by applying a shift operation $\mathrm{U}_{\text{shift}}^{\pm k}$, see (14). Here the value of k determines the type of operation such that when $k > 0$, the corresponding shift operation is denoted as a *quantum incrementer*, shown in Figure 9a [16,17] and when $k < 0$, the corresponding shift operation is denoted as a *quantum decrementer* [16,17], shown in Figure 9b. The quantum shift operation plays a vital role in convolution operation by striding the filter operation over data windows [18].

$$\begin{aligned} \mathbf{U}_{\text{shift}}^{\pm k}|\psi\rangle &= \sum_{i=0}^{2^n-1} c_i|j\rangle, \text{where } j = (i \pm k) \bmod 2^n \\ &= \left(\prod_{i=0}^{k-1} \mathrm{U}_{\text{shift}}^{\pm 1}\right)|\psi\rangle = \left(U_{\text{shift}}^{\pm 1}\right)^k|\psi\rangle \end{aligned} \tag{14}$$

Each quantum shift operation $\mathrm{U}_{\text{shift}}^{\pm 1}$ acting on an n-qubit state can be decomposed into a pyramidal structure of $n \times$ MCX gates in a series pattern [15], as shown in Figures 9a and 9b. In terms of fundamental single-qubit and CNOT gates, the depth of a quantum shift operation with ± 1 can be demonstrated as quadratic [13], see (15). Generalized quantum shift operations $\mathrm{U}_{\text{shift}}^{\pm k}$ can be derived using $k \times \mathrm{U}_{\text{shift}}^{\pm 1}$ operations, from the expression in (14) and the corresponding circuit depth can be derived as shown in (16).



(a) Quantum incrementer by 1 **(b)** Quantum decrementer by 1

(c) Quantum shift operation by $\pm k$, where $k = \sum_0^{j=n-1} k_j 2^j$

Figure 9. Quantum incrementer, decrementer, and construction of quantum shift operation.

$$\begin{aligned}
\Delta_{\mathrm{U}^{-1}_{\text{shift}}}(n) &= \sum_{i=0}^{n-1} \Delta_{\text{MCX}}(i) \\
&= \sum_{i=0}^{n-1} 48i - 196 \\
&\leq 24n^2 - 220n = \mathcal{O}\left(n^2\right) \text{ for large } n
\end{aligned} \tag{15}$$

$$\Delta_{\mathrm{U}^{\pm k}_{\text{shift}}}(n,k) = k \cdot \Delta_{\mathrm{U}^{\pm 1}_{\text{shift}}}(n) = \mathcal{O}\left(kn^2\right) \tag{16}$$

Now, $\mathrm{U}^{\pm k}_{\text{shift}}$ can be expressed as a sequence of controlled shift operations, as shown in (17a). Such operations can be denoted as $\mathrm{U}^{\pm k_j 2^j}_{\text{shift}}(n)$, where $0 \leq j < n$ and (n) indicates that the shift operation is applied to an n-qubit state, as shown in (17b). Instead of applying sequential $\mathrm{U}^{\pm 1}_{\text{shift}}(n)$ operations, each $\mathrm{U}^{\pm 2^j}_{\text{shift}}(n)$ operation can be performed using a single $\mathrm{U}^{\pm 1}_{\text{shift}}(n-j)$ operation, see (17b). Therefore, a more depth-efficient decomposition is shown in Figure 9 and the corresponding depth is provided in (17c).

$$k = \left(k_{n-1}k_{n-2}\cdots k_j \cdots k_1 k_0\right)_2 = \sum_{j=0}^{n-1} k_j 2^j : k_j \in \{0,1\} \tag{17a}$$

$$\begin{aligned} \mathrm{U}_{\mathrm{shift}}^{\pm k}(n) &= \mathrm{U}_{\mathrm{shift}}^{\pm\sum_{j=n-1}^{0} k_j 2^j}(n) = \prod_{j=n-1}^{0} \mathrm{U}_{\mathrm{shift}}^{\pm k_j 2^j}(n) \\ &= \prod_{j=n-1}^{0} \left(\mathrm{U}_{\mathrm{shift}}^{\pm k_j}(n-j) \otimes I^{\otimes j}\right) \end{aligned} \tag{17b}$$

$$\begin{aligned} \Delta^{\mathrm{opt}}_{\mathrm{U}_{\mathrm{shift}}^{\pm k}}(n) &\leq \sum_{j=0}^{n-1} \Delta_{\mathrm{U}_{\mathrm{shift}}^{-1}}(n-j) = 24\sum_{j=0}^{n-1} j^2 - 220\sum_{j=0}^{n-1} j \\ &\leq 8n^3 - 122n^2 + 114n = \mathcal{O}(n^3) \text{ for large } n \end{aligned} \tag{17c}$$

2.3. Quantum Measurement and Reset

The quantum measurement operation of a qubit, is usually and informally referred to as a measurement "gate". The measurement gate is a non-unitary operation that can project the quantum state of a qubit $|\psi\rangle$ to the $|0\rangle$ or $|1\rangle$ basis states [11]. The likelihood of measuring any basis state can be obtained by taking the squared magnitude of their corresponding basis state coefficient. For a n-qubit register $|\psi\rangle$ with 2^n possible basis states, the probability of measuring each qubit in any particular basis state $|j\rangle$, where $0 \leq j < 2^n$, is given by $|c_j|^2$ [14]. The classical output of an n-qubit amplitude-encoded [19] data can be decoded as $\psi^{\mathrm{classical}}_{\mathrm{decoded\text{-}data}}$. This classical output vector can be reconstructed by the square root of the probability distribution $\sqrt{P(|\psi\rangle)}$, as shown in (18), (19) and Figure 10. When amplitude-encoding [19] is used for encoding positive real classical data, the coefficients of the corresponding quantum pure state [11] $|\psi\rangle$ are also positive real, i.e., $c_j \in \mathbb{R}^+$, where $0 \leq j < 2^n$. Thus, the amplitudes of $|\psi\rangle$ are numerically equal in values to the coefficients of $\psi^{\mathrm{classical}}_{\mathrm{decoded\text{-}data}}$, i.e., $|\psi\rangle = \psi^{\mathrm{classical}}_{\mathrm{decoded\text{-}data}}$. Therefore, the quantum state $|\psi\rangle$ can be completely determined from the measurement probability distribution such that $|\psi\rangle = \sqrt{P(|\psi\rangle)}$ only when the amplitudes of the quantum state are all of positive real values. Moreover, the probability distribution $P(|\psi\rangle)$ can be reconstructed by repeatedly measuring (sampling) the quantum state $|\psi\rangle$. In general, an order of 2^n measurements is required to accurately reconstruct the probability distribution. In order to reduce the effects of quantum statistical noise, it is recommended to gather as many circuit samples (shots) [20] as possible.

$$P(|\psi\rangle) = \begin{bmatrix} p_0 \\ p_1 \\ \vdots \\ p_j \\ \vdots \\ p_{N-2} \\ p_{N-1} \end{bmatrix} = \begin{bmatrix} |c_0|^2 \\ |c_1|^2 \\ \vdots \\ |c_j|^2 \\ \vdots \\ |c_{N-2}|^2 \\ |c_{N-1}|^2 \end{bmatrix}, \text{ where } p_j = |c_j|^2, \text{ and } 0 \leq j < N \tag{18}$$

$$\psi^{\mathrm{classical}}_{\mathrm{decoded\text{-}data}} = \sqrt{P(|\psi\rangle)} = \begin{bmatrix} |c_0| \\ |c_1| \\ \vdots \\ |c_j| \\ \vdots \\ |c_{N-2}| \\ |c_{N-1}| \end{bmatrix} \tag{19}$$

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 March 2024 doi:10.20944/preprints202403.1060.v1

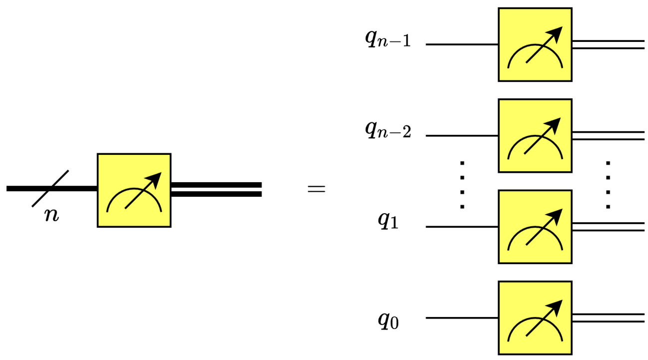


Figure 10. Full quantum state measurement diagram.

The reset operation sets the state of qubits to $|0\rangle$. This operation consists of a mid-circuit measurement gate followed by a conditional X gate [21,22] such that the bit-flip operation is applied when the measured qubit is in state $|1\rangle$. The reset gate and its equivalent circuit are both shown in Figure 11 [22].

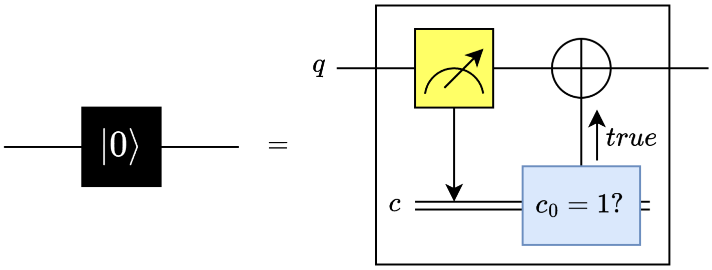


Figure 11. Reset gate and equivalent circuit.

2.4. Classical-to-Quantum (C2Q)

There are a number of quantum data encoding techniques [19,23], each of which uses different methods to initialize quantum circuits from the ground state. Among the many methods, this work leverages the classical-to-quantum (C2Q) arbitrary state synthesis [19,23] operation to perform amplitude encoding and initialize an n-qubit state $|\psi_0\rangle$, see Figure 12. The C2Q operation employs a pyramidal structure of multiplexed R_y and R_z gates. It should be noted that the R_z gates are only required for positive real data. Thus, for positive real data, the circuit depth is $2 \cdot 2^n - n - 2$, while for complex data, the circuit depth is $3 \cdot 2^n - n - 4$ [19].

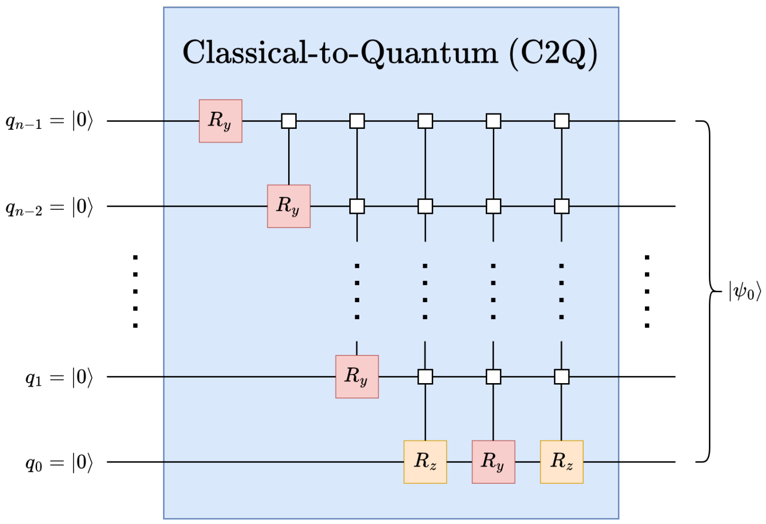


Figure 12. The quantum circuit for classical-to-quantum (C2Q) arbitrary state synthesis. [23]

2.5. Quantum Machine Learning with Variational Algorithms

Variational quantum algorithms are a type of quantum-classical techniques that facilitate implementations of machine learning on noisy-intermediate-scale-quantum (NISQ) machines [5,7]. Current quantum devices are not able to maintain coherent states for sufficient periods, preventing current algorithms from performing meaningful optimization on the machine learning model. Thus, VQAs combine classical optimization algorithms with parameterized quantum circuits, or *ansatz*. Here, the ansatz takes on the role of the model [5]. One specific type of VQA is the variational quantum classifier (VQC), which is used for classification problems. Existing VQCs [6,8,9] have been shown to be effective for classifying datasets with high accuracy and few training parameters in both simulation and current quantum processors.

3. Related Work

In this section, we discuss the existing related works with an emphasis on quantum machine learning. Our discussion focuses on commonly used **Data Encoding** techniques, existing implementations of the **Convolution** and **Quantum Convolution** algorithms, and related **Quantum Machine Learning** algorithms. Moreover, we also discuss existing quantum convolutional classification algorithms that leverage data locality.

3.1. Data Encoding

For encoding classical image data into the quantum domain, the commonly used methods are Flexible Representation of Quantum Images (FRQI) [24] and the Novel Enhanced Quantum Representation (NEQR) [25]. In FRQI, positional and color information are encoded as amplitude encoding and angle encoding respectively. In NEQR, positions of the pixels are encoded using amplitude encoding but color information is encoded using basis encoding, where q represents the number of bits/qubits allocated for color data. For $N = 2^n$ data points, in terms of circuit width and depth, FRQI incurs $n + 1$ and $\mathcal{O}(4^n)$, respectively, while NEQR incurs $n + q$ and $\mathcal{O}(qn2^n)$, respectively [15]. Although these techniques are employed in the existing quantum convolution techniques, their disadvantages are discussed below.

3.2. Convolution

We now discuss existing implementations of convolution and discuss their associated time complexity. These implementations consist of various classical and quantum techniques. In addition, we consider the shortcomings of existing quantum convolution methods.

Classical Convolution

Classical implementations of convolution are usually implemented directly, through general matrix multiplication (GEMM), or through the Fast Fourier transform (FFT). For a data size N, running the direct implementation on CPUs has complexity $\mathcal{O}(N^2)$ [26], while the complexity of an FFT-based implementation is $\mathcal{O}(N \log N)$ [26]. On GPUs, FFT-based convolution incurs a similar $\mathcal{O}(N \log N)$ complexity [27], while the direct approach requires $\mathcal{O}(N_F N)$ FLOPS [28,29], where N_F is the filter size.

Quantum Convolution

The existing quantum convolution techniques [18,30–33] rely on fixed filter sizes and support only specific filters at a time, e.g., edge-detection. They do not contain methods for implementing a general filter. Additionally, these techniques have a quadratic circuit depth, i.e., $\mathcal{O}(n^2)$, where $n = \lceil \log_2 N \rceil$ is the number of qubits and N is the size of the input data. While these methods appear to show quantum advantage, these results do not include overhead incurred from data encoding. The related methods employ the FRQI and NEQR data encoding methods, leading to inferior performance compared to classical methods once the additional overhead is factored in. The authors in [16] propose an edge-detection technique based on quantum wavelet transform QWT and amplitude encoding, named

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 March 2024 doi:10.20944/preprints202403.1060.v1

quantum Hadamard edge detection (QHED) which is not generalizable for multiple convolution kernels or multidimensional data. Thus, their algorithm loses parallelism, increases circuit depth, and is difficult to generalize beyond capturing 1-D features. In [15], the authors have developed a quantum convolution algorithm that supports single feature/kernel and multidimensional data. In this work, we leverage the convolution method from [15] and generalize it to support multiple features/kernels in our proposed MQCC framework.

3.3. Quantum Machine Learning

There exist two primary techniques for quantum convolutional classification that may leverage data locality through convolution: quantum convolutional neural network (QCNN) [34] and quanvolutional neural networks [35]. QCNN is inspired by classical convolutional neural networks, employs quantum circuits to perform convolutions, while quanvolutional neural networks replace classical convolutional layers with quantum convolutional (or *quanvolutional*) layers.

3.3.1. Quantum Convolutional Neural Networks

The QCNN [34] and our proposed multidimensional quantum convolutional classifier (MQCC) are both VQAs with structures inspired by CNNs. However, QCNN borrows the superficial structure of CNNs without considering the underlying purpose. Specifically, the QCNN quantum ansatz is designed so that its "convolution" and "pooling" operations exploit the locality of qubits in the circuit (rather than the locality of data). However, unlike data locality, qubit locality does not have a practical purpose for machine learning in terms of isolating relevant input features. Moreover, by considering input data as 1-D, QCNNs do not leverage the dimensionality of datasets, which constitute a primary advantage of CNNs. MQCC, on the other hand, faithfully implements CNN operations in quantum circuits, offering performance improvements in computational speed and classification accuracy even over contemporary implementations on classical computers.

3.3.2. Quanvolutional Neural Networks

Quanvolutional neural networks [35] are a hybrid quantum-classical algorithm named eponymously after the *quanvolutional* layer added to a conventional CNN. These quanvolutional layers serve to decimate a 2-D image, which is then sequentially fed into a quantum device. In this manner, the quanvolutional layer effectively exploits data locality. Yet, the model's dependency on classical operations, specifically the decimation of input data and the repeated serial data I/O transfer, vastly increases compute time. In contrast, the required convolution operation is incorporated into our proposed MQCC, reducing classical-quantum data transfer. Moreover, MQCC takes advantage of parallelism inherent to quantum computers, while quanvolutional neural networks do not. Together, this allows the MQCC to apply convolutional filters to window data in parallel.

4. Materials and Methods

In this section, we describe the materials and methods associated with the proposed multidimensional quantum convolutional classifier (MQCC). The proposed method mainly uses generalized quantum convolution, quantum pooling based on the quantum Haar transform (QHT) and partial measurement [14], and a quantum fully-connected layer that will be illustrated in this section. To the best of our knowledge, this work is the first to:

- Develop a generalizable quantum convolution algorithm for a quantum convolution-based classifier that supports multiple features/kernels.
- Design a scalable MQCC that uses multidimensional quantum convolution and pooling based on the QHT. This technique reduces training parameters and time complexity compared to other classical and quantum implementations.
- Evaluate the MQCC model in a state-of-the-art QML simulator from Xanadu using a variety of datasets.

 doi:10.20944/preprints202403.1060.v1

4.1. Quantum Fully-Connected Layer

A fully-connected neural network constitutes a collection of layers that each perform a linear transformation on N_{in} input features $\mathbf{x} \in \mathbb{R}^{N_{\text{in}}}$ to generate N_{out}-feature output $\mathbf{y} \in \mathbb{R}^{N_{\text{out}}}$ [2,3]. Each layer can be represented in terms of a multiply-and-accumulate (MAC) operation and an addition operation as shown in (20), where $\mathbf{W} \in \mathbb{R}^{N_{\text{out}} \times N_{\text{in}}}$ and $\mathbf{b} \in \mathbb{R}^{N_{\text{out}}}$ represent the trainable weight and bias parameters, respectively.

$$\mathbf{y} = \mathbf{W}^T \mathbf{x} + \mathbf{b} \tag{20}$$

The particular weights and biases that generate the j^{th} feature of the output, y_j, can be isolated by taking the j^{th} column of $\mathbf{W}$, $\mathbf{w}_j$, and the j^{th} term of $\mathbf{b}$, b_j, as shown in (21), which can be directly implemented using quantum circuits. Section 4.1.1 discusses the quantum circuits for a single-feature output, and Section 4.1.2 generalizes the proposed technique for an arbitrary amount of features in the output.

$$y_j = (\mathbf{w}_j \cdot \mathbf{x}) + b_j, \text{ where } 0 \le j < N_{\text{out}} \tag{21}$$

4.1.1. Single-Feature Output

For a single-feature output neural network, the weight parameters can be represented as a vector $\mathbf{w} \in \mathbb{R}^{N_{\text{in}}}$. Here $\mathbf{w}$ can be expressed as a quantum state $|w\rangle$ as shown in (22) similar to the process of C2Q data encoding, see Section 2.4.

$$|w\rangle = \frac{1}{\|\mathbf{w}\|} \begin{bmatrix} \mathbf{w} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \updownarrow N_{\text{in}} \\ \\ \\ \\ \end{matrix} \Bigg\updownarrow 2^{n_{\text{in}}}, \text{ where } n_{\text{in}} = \lceil \log_2 N_{\text{in}} \rceil \tag{22}$$

Similarly, for a single-feature output, Dirac notation of the MAC operation follows from (21) as shown in (23), where $|\psi\rangle$ corresponds to the input data.

$$y - b = \langle w|\psi\rangle \tag{23}$$

However, it is necessary to obtain a quantum operator to perform a parameterized unitary linear transformation from the weights vector $|w\rangle$ on the input data $|\psi\rangle$ using an inverse C2Q operation, as shown in (24).

$$\begin{aligned} \mathrm{U}_{\text{MAC}}(|w\rangle) &= \begin{bmatrix} \langle w| \\ \langle \times | \\ \vdots \\ \langle \times | \end{bmatrix} \Bigg\updownarrow 2^{n_{\text{in}}} \\ &= \mathrm{U}^{\dagger}_{\text{C2Q}}(|w\rangle) \end{aligned} \tag{24}$$

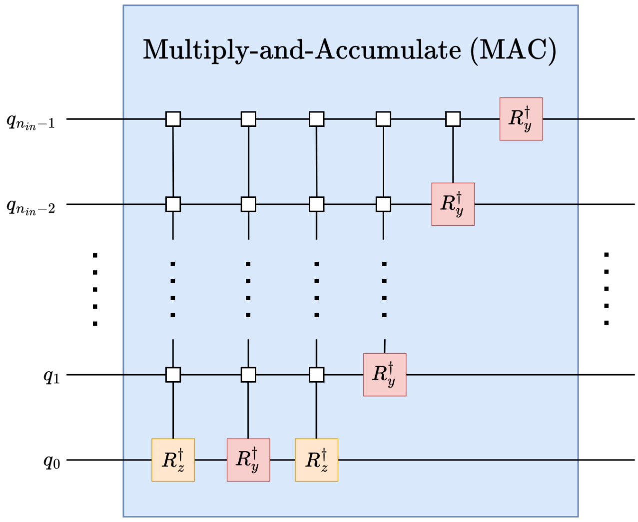


Figure 13. Quantum multiply-and-accumulate (MAC) operation using inverse arbitrary state synthesis.

4.1.2. Multi-Feature Output

A Multi-Feature Output can be implemented in a naive approach using Single-Feature Output (24) for an N_{out}-feature output, where $N_{\text{out}} \leq N_{\text{in}}$, which can be obtained by encoding each weight vector $\mathbf{w}_j : 0 \leq j < N_{\text{out}}$ as a normalized row in U_{MAC}. However, the result might yield a non-unitary operator as the weight vectors can be arbitrary. U_{MAC} is unitary when each row is orthogonal to all other rows in the matrix, mathematically, $\langle W_i | W_j \rangle = \delta_{ij} : \forall i, j \in [0, N_{\text{out}})$. As described in Section 2.2.3, independently-defined weights can be supported for each feature of the output by *multiplexing* U_{MAC}. Now, the generic fully-connected operation, U_{FC}, can be generated as shown in (25), where $n_{\text{out}} = \lceil \log_2 N_{\text{out}} \rceil$.

$$\text{U}_{\text{FC}} = \begin{bmatrix} \text{U}_{\text{MAC}}(|w_0\rangle) & & \\ & \ddots & \\ & & \text{U}_{\text{MAC}}(|w_{2^{n_{\text{out}}}-1}\rangle) \end{bmatrix} \tag{25}$$

By generating N_{out} replicas of the initial state, $|\psi_0\rangle$, the operation can be parallelized, i.e., $\text{U}_{\text{MAC}}(|w_0\rangle) \ldots \text{U}_{\text{MAC}}(|w_{2^{n_{\text{out}}}-1}\rangle)$ transformations from (25).

Replication:

To replicate the initial state, $|\psi_0\rangle$ n_{out} qubits, which extend the statevector to a total size of $2^{n_{\text{in}}+n_{\text{out}}}$, see (26) and Figure 14.

$$|\psi_1\rangle = |0\rangle^{\otimes n_{\text{out}}} \otimes |\psi_0\rangle = \begin{bmatrix} |\psi_0\rangle \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} \updownarrow 2^{n_{\text{in}}} \\ \\ \\ \\ \end{matrix} \; \Big\updownarrow 2^{n_{\text{in}}+n_{\text{out}}} \tag{26}$$

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 March 2024 doi:10.20944/preprints202403.1060.v1

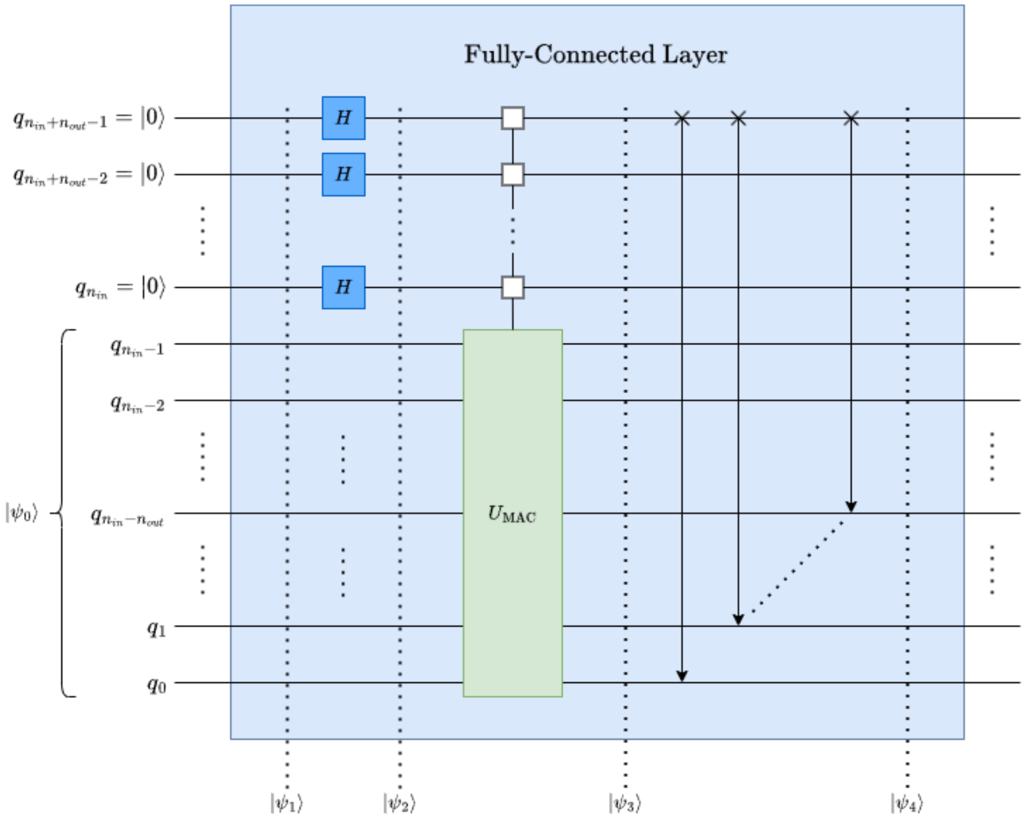


Figure 14. Quantum fully-connected layer with an N_{out}-feature output.

By applying an n_{out}-qubit Hadamard operation to the relevant qubits the replicas can be obtained through superposition, see (27) and Figure 14, which generates the desired replicas scaled by a factor of $\frac{1}{\sqrt{2^{n_{out}}}}$ to maintain normalization $\langle\psi_2|\psi_2\rangle = 1$.

$$|\psi_2\rangle = \left(H^{\otimes n_{out}} \otimes I^{\otimes n_{in}}\right)|\psi_1\rangle = \frac{1}{\sqrt{2^{n_{out}}}} \begin{bmatrix} |\psi_0\rangle \\ \vdots \\ |\psi_0\rangle \end{bmatrix} \begin{matrix} \updownarrow 2^{n_{in}} \\ \vdots \\ \updownarrow 2^{n_{in}} \end{matrix} \Bigg] 2^{n_{in}+n_{out}} \tag{27}$$

Applying the U_{FC} Filter:

U_{FC} can perform the MAC operation for the entire N_{out}-feature output in parallel with the set of replicas of $|\psi_0\rangle$, see, (28) and Figure 14.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 March 2024 doi:10.20944/preprints202403.1060.v1

$$
\begin{aligned}
|\psi_3\rangle = \mathrm{U}_{\mathrm{FC}}|\psi_2\rangle &= \begin{bmatrix} \mathrm{U}_{\mathrm{MAC}}(|w_0\rangle)|\psi_0\rangle \\ \vdots \\ \mathrm{U}_{\mathrm{MAC}}(|w_j\rangle)|\psi_0\rangle \\ \vdots \\ \mathrm{U}_{\mathrm{MAC}}(|w_{2^{n_{\mathrm{out}}}-1}\rangle)|\psi_0\rangle \end{bmatrix} \begin{matrix} \updownarrow 2^{n_{\mathrm{in}}} \\ \vdots \\ \updownarrow 2^{n_{\mathrm{in}}} \\ \vdots \\ \updownarrow 2^{n_{\mathrm{in}}} \end{matrix} \; 2^{n_{\mathrm{in}}+n_{\mathrm{out}}} \\
&= \begin{bmatrix} \langle w_0|\psi_0\rangle \\ \times \\ \vdots \\ \times \\ \vdots \\ \langle w_j|\psi_0\rangle \\ \times \\ \vdots \\ \times \\ \vdots \\ \langle w_{2^{n_{\mathrm{out}}}-1}|\psi_0\rangle \\ \times \\ \vdots \\ \times \end{bmatrix} \begin{matrix} 2^{n_{\mathrm{in}}} \\ \vdots \\ 2^{n_{\mathrm{in}}} \\ \vdots \\ 2^{n_{\mathrm{in}}} \end{matrix} \; 2^{n_{\mathrm{in}}+n_{\mathrm{out}}}
\end{aligned} \tag{28}
$$

Data Rearrangement:

The data rearrangement operation can be performed by applying perfect-shuffle gates, see Section 2.2.5. It simplifies the output-feature extraction by gathering them into N_{out} data points at the top of the statevector, see (29) and Figure 14.

$$
|\psi_4\rangle = \prod_{i=0}^{n_{\mathrm{out}}-1} \left(\mathrm{RoL}(n_{\mathrm{in}} + n_{\mathrm{out}} - i) \otimes I^{\otimes i}\right)|\psi_3\rangle = \begin{bmatrix} \langle w_0|\psi_0\rangle \\ \vdots \\ \langle w_j|\psi_0\rangle \\ \vdots \\ \langle w_{2^{n_{\mathrm{out}}}-1}|\psi_0\rangle \\ \times \\ \vdots \\ \times \end{bmatrix} \begin{matrix} 2^{n_{\mathrm{out}}} \\ \\ \\ \end{matrix} \; 2^{n_{\mathrm{in}}+n_{\mathrm{out}}} \tag{29}
$$

It is worth mentioning that instead of applying the auxiliary qubits at the most-significant position as shown in the decomposed and simplified fully-connected circuit in Figure 15, auxiliary qubits can be applied at the least-significant position to avoid perfect-shuffle permutations.

4.1.3. Circuit Depth of the Quantum Fully-Connected Layer

As discussed in Section 2.4, $\mathrm{U}_{\mathrm{MAC}}$ operation is implemented by applying the C2Q / arbitrary state synthesis operation with a depth of $(3 \cdot 2^{n_{\mathrm{in}}} - n_{\mathrm{in}} - 4)$ fundamental single-qubit and CNOT gates. The depth is expected to increase by a factor of $2^{n_{\mathrm{out}}}$ when multiplexing $\mathrm{U}_{\mathrm{MAC}}$ to implement an N_{out}-feature output [12], see (30).

$$
\begin{aligned}
\Delta_{\mathrm{U_{FC}}}(n_{\mathrm{in}}, n_{\mathrm{out}}) &= 2^{n_{\mathrm{out}}} \cdot \Delta_{\mathrm{U_{MAC}}}(n_{\mathrm{in}}) \\
&= 2^{n_{\mathrm{out}}}(3 \cdot 2^{n_{\mathrm{in}}} - n_{\mathrm{in}} - 4) \\
&= \mathcal{O}(2^{n_{\mathrm{in}}+n_{\mathrm{out}}})
\end{aligned} \tag{30}
$$

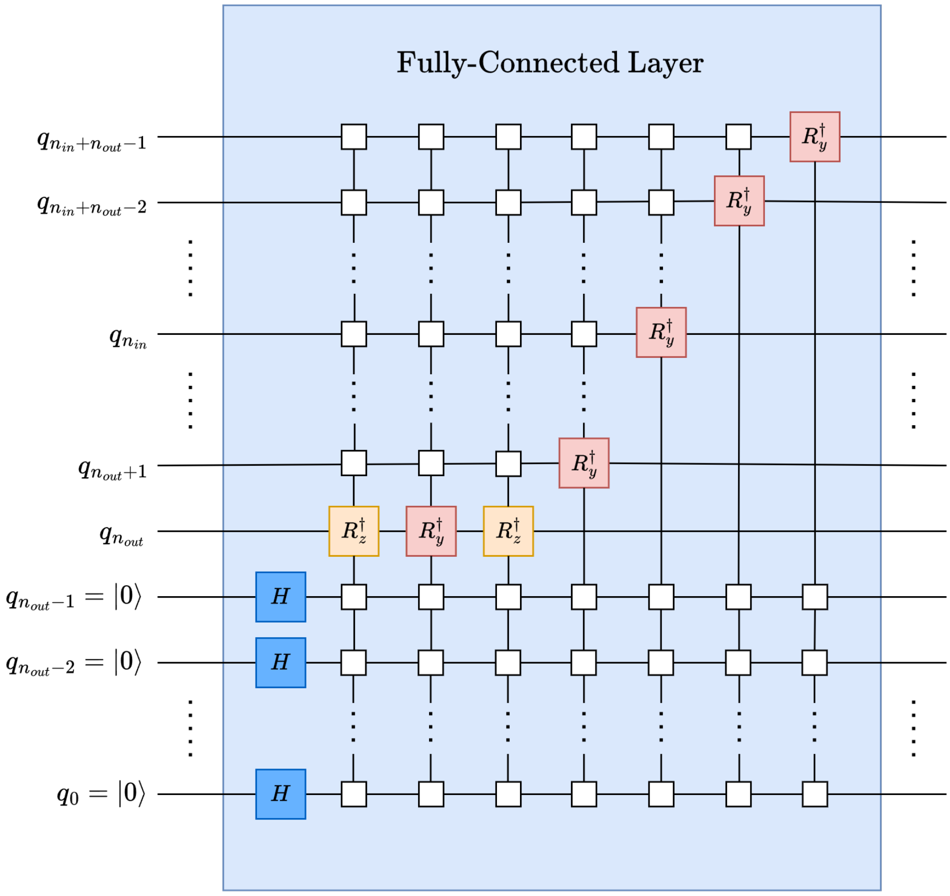


Figure 15. Decomposed and simplified quantum fully-connected layer.

4.2. Generalized Quantum Convolution

The most significant part of the MQCC framework is the generalized quantum convolution operation with support for arbitrary, parameterized filters. Compared to the classical convolution operation, the convolution operation in the quantum domain achieves exponential improvement in time complexity due to its innate parallelism. The convolution operation consists of stride, data rearrangement, and multiply-and-accumulate (MAC) operations.

Stride:

The first step of quantum convolution is generating shifted replicas of the input data. Quantum decrementers controlled by additional qubits, called "filter" qubits are used for this purpose. The $\mathrm{U}_{\mathrm{shift}}^{-1}$ operator as shown in Figure 9b shifts the replica by a single stride.

Multiply-and-Accumulate (MAC):

Kernels are applied to the strided replicas of the input data in parallel using the MAC operation, see Figure 13. In the MAC operations, kernels are applied to the contiguous set of data with the help of the inverse arbitrary state synthesis operation. One benefit achieved from using this MAC operation is the superposition of multiple kernels. The superposition of the kernel can be helpful for the classification of multiple features.

Data Rearrangement:

Data rearrangement is required to coalesce the output pieces of the MAC steps and create one contiguous piece of output. This step is performed using perfect shuffle permutation (PSP) operations described in Section 2.2.5.

4.2.1. One-Dimensional Multi-Feature Quantum Convolution

The one-dimensional quantum convolution operation, with kernel of size N_K terms, requires generating N_K replicas of the input data in a range of possible strides between $0 \leq k < N_K$. Therefore, a total of $N_K N$ terms need to be encoded into a quantum circuit including the $n_k = \lceil \log_2 N_K \rceil$ additional auxiliary qubits, denoted as "kernel" qubits that are allocated the most-significant qubits to maintain data contiguity.

Necessary N_K replicas of the input vector are created by using Hadamard gates, see Figure 16. Convolution kernels can be implemented using multiply-and-accumulate (MAC) operations; as such, it is possible to leverage $\mathrm{U}_{\mathrm{MAC}}$ as defined in Section 4.1 for implementing quantum convolution kernels. Given a kernel $\mathbf{K} \in \mathbb{R}^{N_K}$, the corresponding kernel operation U_K can be constructed from the normalized kernel $|K\rangle$ as shown in (31).

$$\mathrm{U}_K = \mathrm{U}_{\mathrm{MAC}}(|K\rangle), \text{ where } |K\rangle = \hat{\mathbf{K}} = \frac{\mathbf{K}}{||\mathbf{K}||} \tag{31}$$

When applied to the n_k lower qubits of the state vector, U_K applies the kernel $\mathbf{K}$ to all data windows in parallel. However, in CNNs, convolution layers typically must support multiple convolution kernels/ features. Fortunately, one major advantage of the proposed quantum convolution technique is that multiple features can be supported by multiplexing only the MAC operations – the stride and data rearrangement operations do not need to be multiplexed, see Figure 16. Accordingly, for N_F features, $n_f = \lceil \log_2 N_F \rceil$ must be added to the circuit and placed in superposition using Hadamard gates, similar to the process in (26). The depth of the proposed multi-feature 1-D quantum convolution can be obtained as (32).

$$\Delta_{\text{1-D conv}}(n, n_k) = \mathcal{O}\left(n_k n^2 - n_k^2 n + 2^{n_f + n_k}\right), \text{ where } n \gg n_f, n_k \tag{32}$$

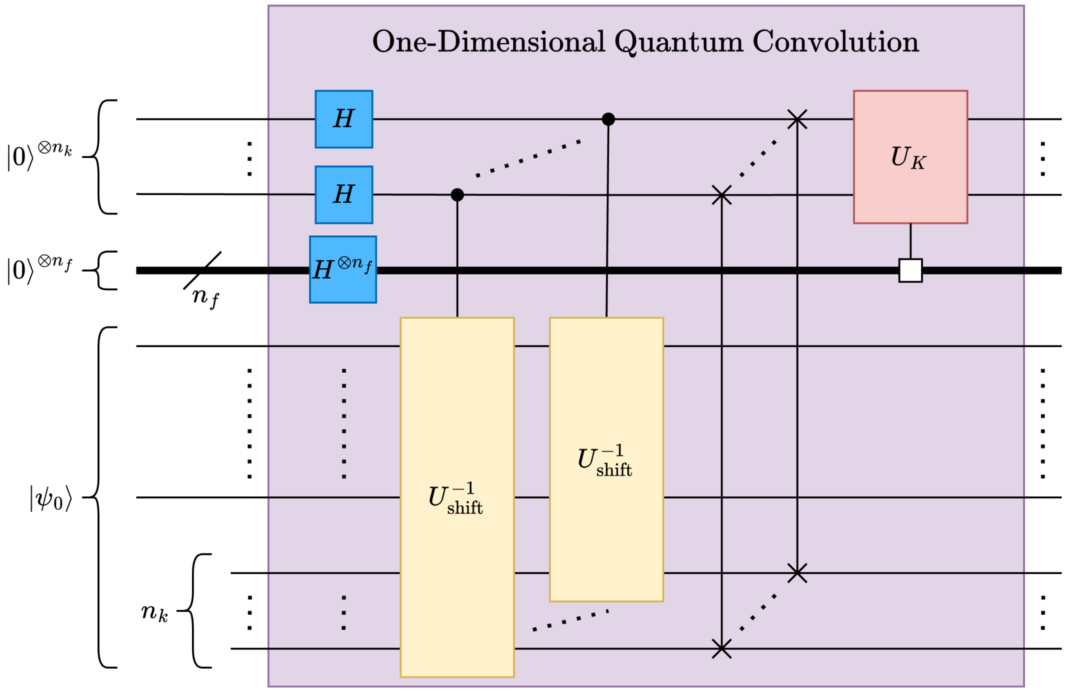


Figure 16. One-dimensional quantum convolution circuit.

4.2.2. Multi-Dimensional Multi-Feature Quantum Convolution

Multi-dimensional quantum convolution can be implemented by stacking multiple one-dimensional quantum circuits as shown in Figure 17. A d-dimensional quantum convolution circuit can be constructed with a stacked kernel of 1-dimensional convolution circuits only when the multidimensional kernels are outer products of d instances of 1-dimensional kernels. The depth of d-D quantum convolution can be obtained as (33).

$$\Delta_{d\text{-D conv}}(n, n_k) = \mathcal{O}\left(n_{k_{\max}} n_{\max}^2 - n_{k_{\max}}^2 n_{\max} + 2^{n_k + n_f}\right), \text{ where}$$
$$n_{\max} = \max_{i=0}^{d-1}(n_i),\ n_{k_{\max}} = \max_{i=0}^{d-1}(n_{k_i}), \text{ and } n_{\max} \gg n_f, n_{k_{\max}} \quad (33)$$

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 March 2024 doi:10.20944/preprints202403.1060.v1

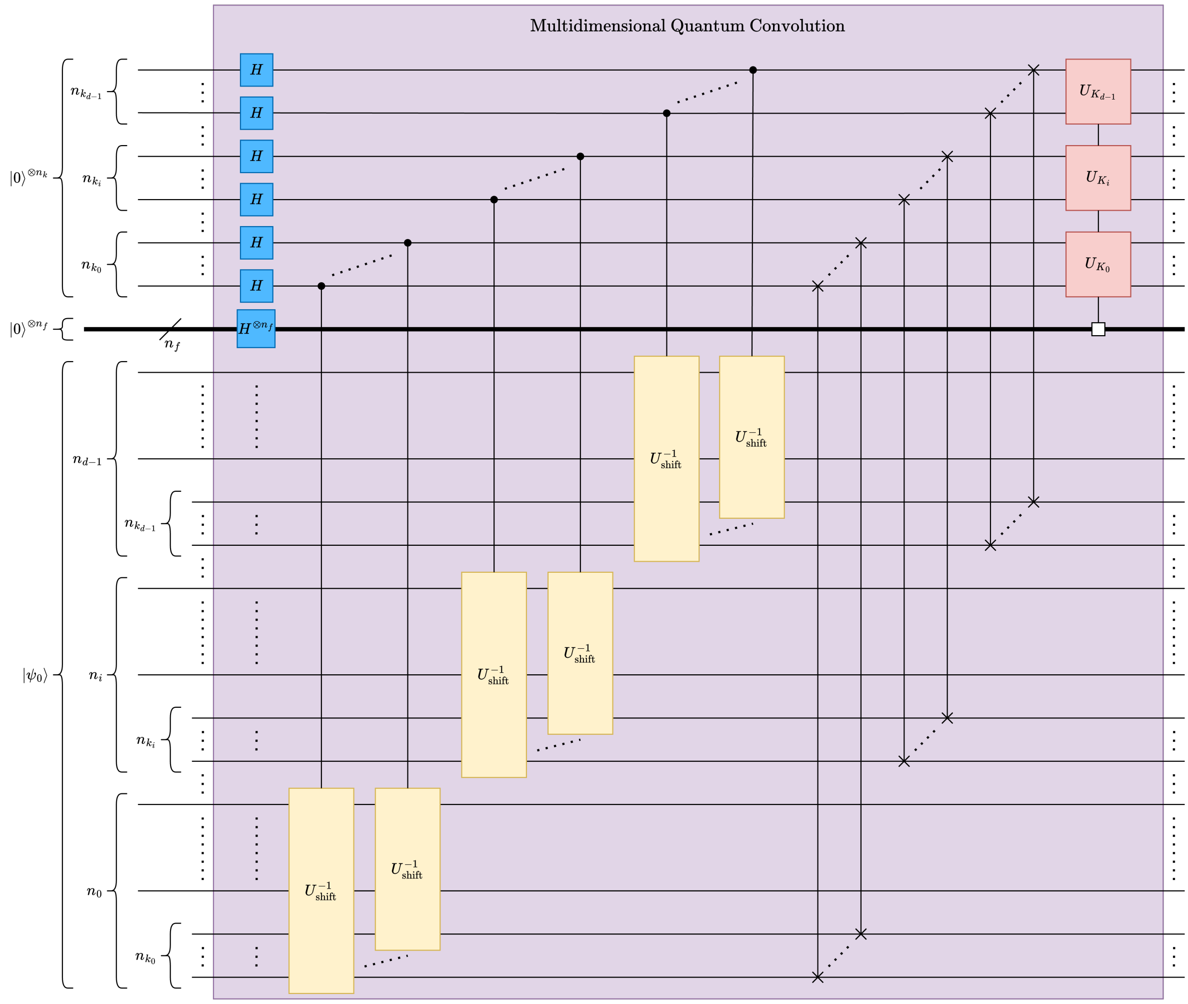


Figure 17. Multi-dimensional quantum convolution circuit

4.3. Quantum Pooling

A critical part of CNNs is the pooling operation or downsampling of the feature maps. One widely-used method is average pooling, where only the average of the adjacent pixels in the feature map is preserved, creating a smoothing effect [36].

4.3.1. Quantum Average Pooling using Quantum Haar Transform

The average pooling operation can be implemented using the quantum wavelet transform (QWT) [14] which has the advantage of preserving data locality using wavelet transform decomposition. It is a commonly used technique for dimension reduction in image processing [14]. In this work, we utilize the simplest and first wavelet transform, quantum Haar transform (QHT) [14] to implement quantum pooling operation. This operation is executed in 2 steps: the Haar wavelet operation and data rearrangement.

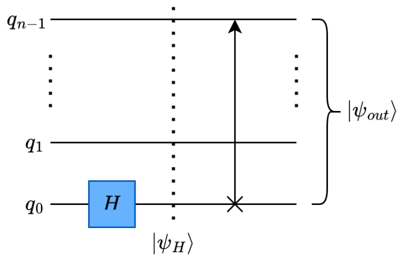
Haar Wavelet Operation:

For separating the high and low-frequency components from the input data, H gates are applied in parallel. The number of H gates applied in QHT is equal to the levels of decomposition.

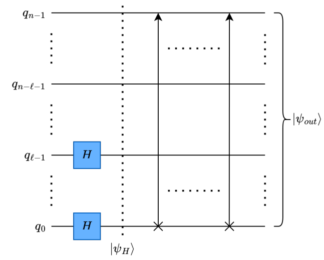
Data Rearrangement:

After separating the high- and low-frequency components, quantum rotate-right (RoR) operations are applied to group them accordingly.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 March 2024 doi:10.20944/preprints202403.1060.v1



(a) Single-level One-dimensional (1-D) QHT circuit.



(b) Multilevel One-dimensional (1-D) QHT circuit.

Figure 18. Quantum Haar Transform Circuits.

As mentioned before, the proposed framework is highly parallelizable regardless of the dimensions of the data as the QHT operation can be applied to multiple dimensions of data in parallel.

As shown in Figure 18a, for a single-level of decomposition, H gates are applied on one qubit (the least significant qubit) per dimension, and for ℓ-level decomposition, shown in Figure 18b, l number of H gates are applied per dimension. Each level of decomposition reduces the size of the corresponding dimension by a factor of 2.

4.3.2. Quantum Euclidean Pooling using Partial Measurement

In machine learning applications, the average and maximum (max) pooling [36] are the most commonly used pooling schemes for dimension reduction. The two schemes differ in the sharpness of data features. On one hand, max pooling yields a sharper definition of input features which makes it preferable for edge detection and certain classification applications [36]. On the other hand, average pooling offers a smoother dimension reduction that may be preferred in other workloads [36]. Thus, to accompany our implementation of quantum averaging pooling using QHT, see Section 4.3.1, it would be beneficial to have an implementation of quantum max pooling. However, such an operation would be non-unitary, creating difficulty for the implementation of quantum max pooling [37]. Therefore, instead of max pooling, we utilize an alternative pooling technique we denote as quantum Euclidean pooling.

Mathematically, average and Euclidean pooling are special cases of the p-norm [38], where for a vector of size N elements, the p-norm or ℓ^p norm of a vector $\mathbf{x} \in \mathbb{C}^N$ is given by (34) for $p \in \mathbb{Z}$ [38]. The average pooling occurs for the 1-norm ($p = 1$) and Euclidean pooling occurs for the 2-norm ($p = 2$). A notable benefit of the Euclidean pooling technique is its zero-depth circuit implementation by leveraging partial measurement [37].

$$\|\mathbf{x}\|_p = \left(\sum_{i=0}^{N} x_i^p \right)^{\frac{1}{p}} \tag{34}$$

This work leverages the multilevel, d-dimensional quantum Euclidean pooling circuit presented in [37], see Figure 19. Here, for each dimension i, ℓ_i is the number of decomposition levels for dimension where $0 \leq i < d$ [37].

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 March 2024 doi:10.20944/preprints202403.1060.v1

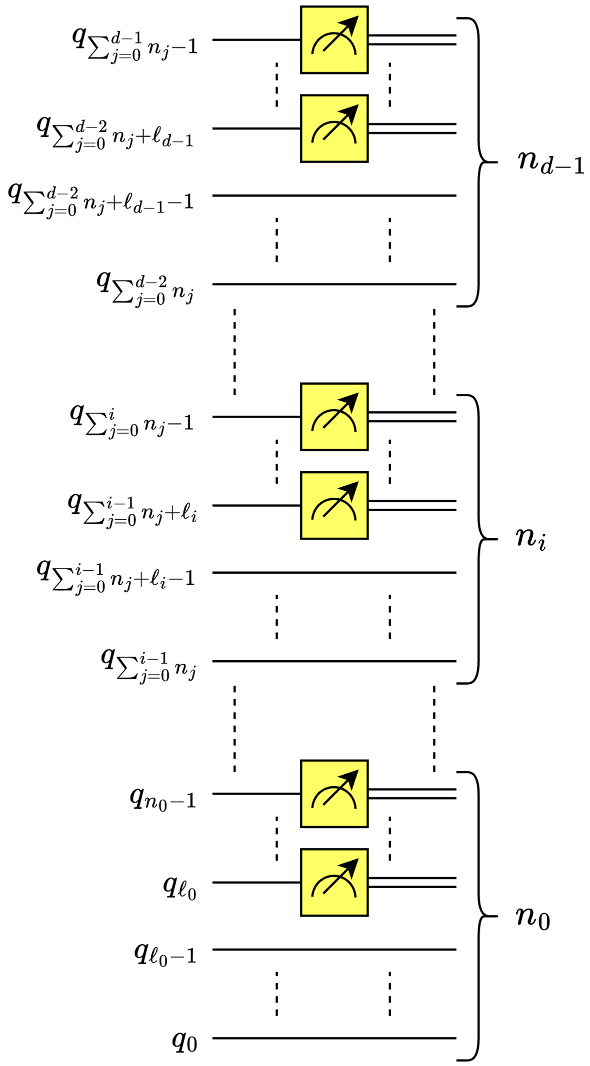


Figure 19. Multilevel, *d*-D Euclidean pooling circuit.

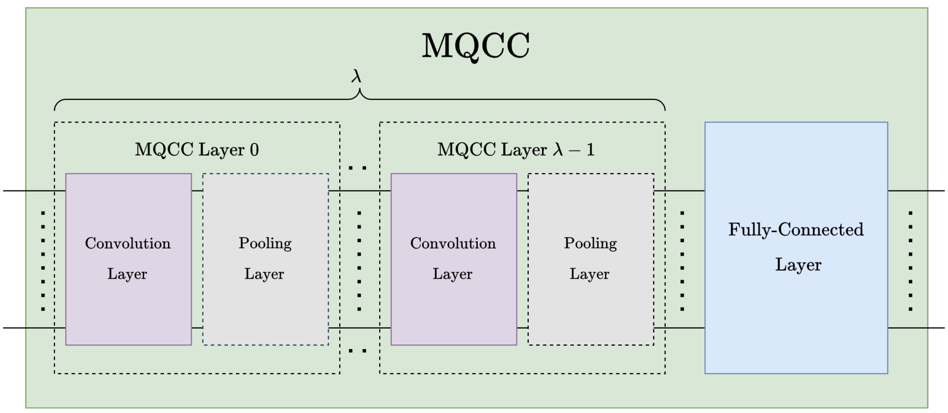


Figure 20. High-level overview of MQCC architecture.

4.4. Multidimensional Quantum Convolutional Classifier

The proposed multidimensional quantum convolution classifier framework, see Figure 20, resembles CNN [2] structures. After a sequence of convolution-pooling (CP) pairs, the model is finally

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 March 2024 doi:10.20944/preprints202403.1060.v1

connected to a fully-connected layer, see Figures 15 and 20. The total number of layers in the proposed model can be expressed in terms of CP pairs as $2\lambda + 1$, where λ is the number of CP pairs.

It is worth mentioning that there is no advantage in changing the number of features among convolution/pooling layers in the MQCC because of the implementation constraints. Therefore, the total number of kernel features can be estimated globally instead of layer-by-layer.

The circuit width of MQCC (35) can be derived from the number of convolution layers, pooling layers, and the fully-connected layer. Input data is encoded using n qubits and each convolution operation adds $n_f = \lceil \log_2 N_F \rceil$ qubits for N_F features and $n_k = \lceil \log_2 N_K \rceil$ qubits per layer for kernels. In addition, the fully-connected operation contributes $n_c = \lceil \log_2 N_C \rceil$ qubits to encode N_C output features/classes. On the other hand, each Euclidean pooling layer frees ℓ qubits, which can then be reused by other layers.

$$\begin{aligned} n_{\text{MQCC}}^{\text{Avg}} &= n + n_f + \lambda n_k + n_c \\ n_{\text{MQCC}}^{\text{Euclidean}} &= n + n_f + n_k + \max(0, (\lambda - 1)n_k - \lambda\ell + n_c) \end{aligned} \tag{35}$$

The MQCC can be further parallelized in terms of circuit depth between the inter-layer convolution/fully-connected layers. This parallelism can be achieved by performing (multiplexed) MAC operations from the quantum convolution and fully-connected layers in parallel with the stride operation from the previous layer(s) of quantum convolution. The circuit depth of MQCC can be derived as shown in (36).

$$\begin{aligned} \Delta_{\text{MQCC}} = &\max_{i=0}^{d-1} \Delta_{\text{U}_{\text{stride}}}(n_i, n_{k_i}) + \lambda(\Delta_H + \Delta_{\text{SWAP}}) \\ &+ \sum_{j=0}^{\lambda-2} \max\left(\max_{i=0}^{d-1} \Delta_{\text{U}_{\text{stride}}}(n_i - j\ell_i, n_{k_i}), \Delta_{\text{U}_K}(n_k, n_f) \right) \\ &+ \max\left(\Delta_{\text{FC}}(n - (\lambda - 1)\ell, n_c), \Delta_{\text{U}_K}(n_k, n_f) \right) \end{aligned} \tag{36}$$

4.5. Optimized MQCC

Figure 21 presents a width-optimized implementation of MQCC, to which we refer as Quantum-Optimized Multidimensional Quantum Convolutional Classifier (MQCC Optimized). To reduce the required number of qubits, the convolution and pooling operations are swapped which allows kernel qubits to be trimmed for each convolution layer, see Section 4.2. To achieve higher processing efficiency, trimmed qubits are reassigned to later layers of dimension reduction and run in parallel. Furthermore, only Euclidean pooling with partial measurements is used because of the inherent circuit depth efficiency. The circuit width of MQCC Optimized is shown in (37), where n is the number of qubits corresponding to the data size, n_f is the number of qubits corresponding to the features, and n_c is the number of qubits corresponding to the classes. If necessary, additional pooling operations can be applied to keep the circuit width at or below the absolute minimum number of qubits n, via excluding qubits dedicated to features and classes. It should be noted that reordering convolution and pooling operations reduces the maximum number of convolution operations by 1.

$$n_{\text{MQCC Optimized}} = n + n_f + n_c \tag{37}$$

Accordingly, the depth of MQCC Optimized can be expressed as shown in (38).

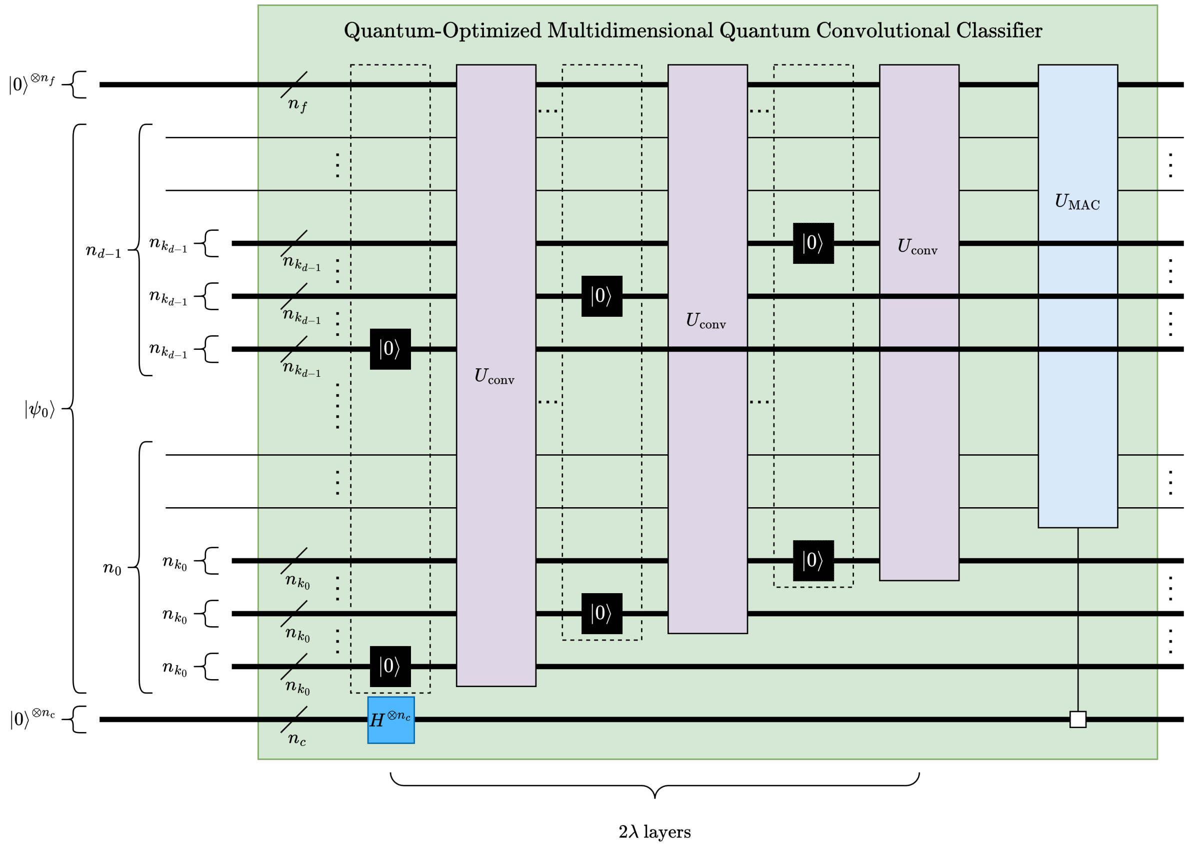


Figure 21. Quantum-Optimized Multidimensional Quantum Convolutional Classifier (MQCC Optimized).

$$\begin{aligned}\Delta_{\text{MQCC}}^{\text{opt}} = &\max_{i=0}^{d-1} \Delta_{\text{U}_{\text{stride}}}(n_i - \ell_i, n_{k_i}) + \lambda(\Delta_H + \Delta_{\text{SWAP}}) \\ &+ \sum_{j=1}^{\lambda-1} \max\left(\max_{i=0}^{d-1} \Delta_{\text{U}_{\text{stride}}}(n_i - j\ell_i, n_{k_i}), \Delta_{\text{U}_K}(n_k, n_f)\right) \\ &+ \max\left(\Delta_{\text{FC}}(n - \lambda\ell, n_c), \Delta_{\text{U}_K}(n_k, n_f)\right)\end{aligned} \quad (38)$$

To further reduce the depth of MQCC Optimized, we investigated replacing inverse-C2Q operations for MAC operations with different parameterized ansatz. More specifically, a common ansatz in QML, namely, `NLocal` operation in `Qiskit` [39] or `BasicEntanglerLayer` in `Pennylane` [40] was utilized, see Figure 22. The depth of this ansatz is linear with respect to the data size, see (39), which is a significant improvement over arbitrary state synthesis, which has a circuit depth of $\mathcal{O}(2^n)$ for an n-qubit state. Although the ansatz could potentially reduce circuit depth, its structure lacks theoretical motivation or guarantees for high fidelity when modeling convolution kernels.

$$\Delta_\alpha(n, \ell) = \ell \cdot n + 1 \quad (39)$$

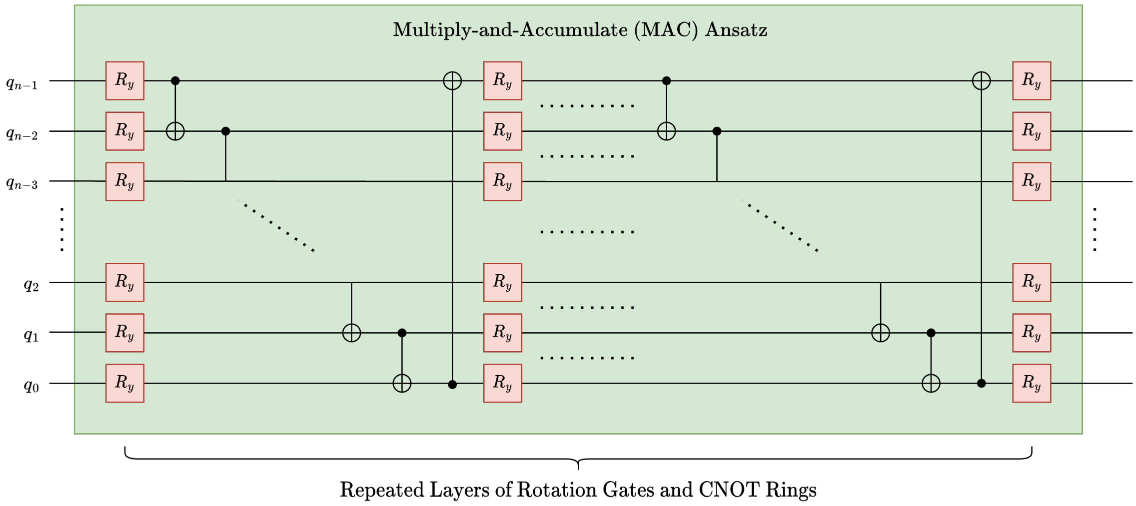


Figure 22. Alternate ansatz option, distinct from U_{MAC}

5. Experimental Work

In this section, we first detail our experimental setup, followed by the results for the proposed MQCC technique. Experiments were conducted using real-world, multidimensional image data to test both the individual and composite components of our techniques.

5.1. Experimental Setup

The MQCC methodology was validated by first evaluating its components, and then evaluating the complete technique. For experiments on 1-D data, we used audio material published by the European Broadcasting Union for sound quality assessment [41]. Using a pre-processing step, the data was converted into a single channel, with the data size varying from 2^8 data points to 2^{20} data points, sampled at a rate of 44.1 kHz. Further tests were conducted using high-resolution, multidimensional, real-world images of various kinds. The 2-D images used were either black-and-white or color *Jayhawk*s [42], as shown in Figure 23, sized from (8×8) pixels to $(512 \times 512 \times 3)$ pixels. For the 3-D image experiments, we used hyperspectral images from the Kennedy Space Center (KSC) dataset [43]. It was pre-processed and resized, with the sizes ranging from $(8 \times 8 \times 8)$ pixels to $(128 \times 128 \times 128)$ pixels. Simulations of quantum convolution operation were run using `Qiskit` SDK (v0.45.0) from IBM Quantum [20] over the given data. To demonstrate the effect of statistical noise on the fidelity (40), both noise-free and noisy (with 1,000,000 circuit samples/shots) simulation environments were evaluated.

$$\text{Fidelity}(\mathbf{X}, \mathbf{Y}) = \frac{\langle \mathbf{X}, \mathbf{Y} \rangle}{\|\mathbf{X}\|_F \|\mathbf{Y}\|_F} \tag{40}$$

(a) 2-D B/W Image [42] **(b)** 3-D RGB Image [42] **(c)** 3-D Hyperspectral Image [43]

Figure 23. High-resolution, multidimensional, real-world input data used in experimental trials.

 doi:10.20944/preprints202403.1060.v1

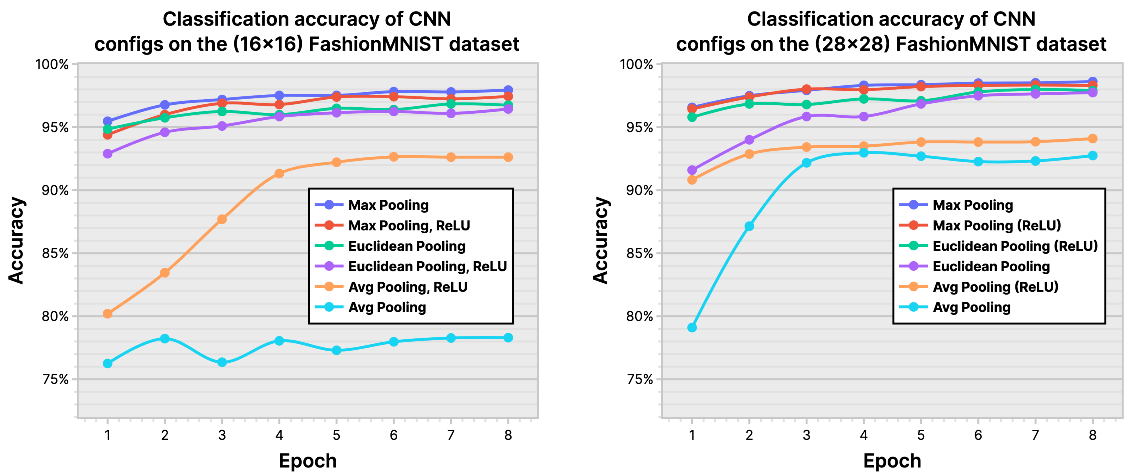
To evaluate the performance of the complete MQCC technique, it was tested against CNNs, QCNNs, and quanvolutional neural networks by its capabilities of binary classification on real-world datasets such as MNIST [45], FashionMNIST [46], and CIFAR10 [47]. The classical components in these trials were run using `PyTorch` (v2.1.0) [48], while the quantum circuits used `Pennylane` (v0.32.0), a Xanadu QML-focused framework [49].

The experiments were performed on a cluster node at the University of Kansas [50]. The node consisted of a 48-Core Intel Xeon Gold 6342 CPU, three NVIDIA A100 80GB GPUs (CUDA version 11.7) with PCIe 4.0 connectivity and 256GB of 3200MHz DDR4 RAM. To account for initial parameter variance in ML or noise in noisy simulations, experiments were repeated for 10 trials with the median being displayed in graphs.

5.2. Configuration of ML models

The different techniques fundamentally being ML models meant that they could share some parameters and metrics during their testing. For example, the log loss and the `Adam` optimizer [51] was shared by all the techniques, and the "feature-count" metric was shared between the CNN and MQCC, which had 4 features per convolution layer. The parameters that were *unique* to each model are discussed next.

Convolutional Neural Networks: In Figure 24, we show the classification accuracy of the CNN model on (16 × 16) and (28 × 28) FashionMNIST datasets, using average pooling, max pooling, and Euclidean pooling. The plots show the obtained accuracy with and without ReLU [52], which is an optional layer that can be appended to each pooling layer in CNN. Based on the results, which show **Max Pooling without ReLU** to be the configuration with the best accuracy, we chose it to be the baseline configuration for CNN in our tests.

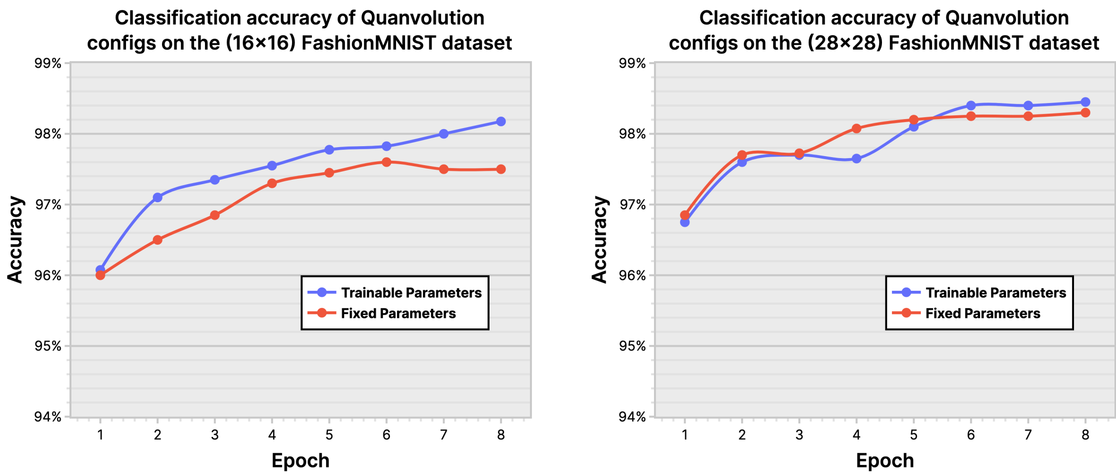


(a) (16 × 16) dataset **(b)** (28 × 28) dataset

Figure 24. Classification accuracy of Convolutional Neural Network (CNN) configurations on the FashionMNIST dataset.

Quanvolutional Neural Networks: While quanvolutional neural networks were initially introduced without a trainable random quantum circuit in the quanvolutional layer, later work has suggested implementing parameterized and trainable quanvolutional layers. We, therefore, test both the trainable and non-trainable quanvolutional techniques and Figure 25 demonstrates that the trainable variant outperforms the other method in the (16 × 16) FashionMNIST dataset, although the differences are negligible on the (28 × 28) dataset. This is used as evidence behind our decision to use the trainable variant of the quanvolutional neural network as the baseline for comparison with the other models.

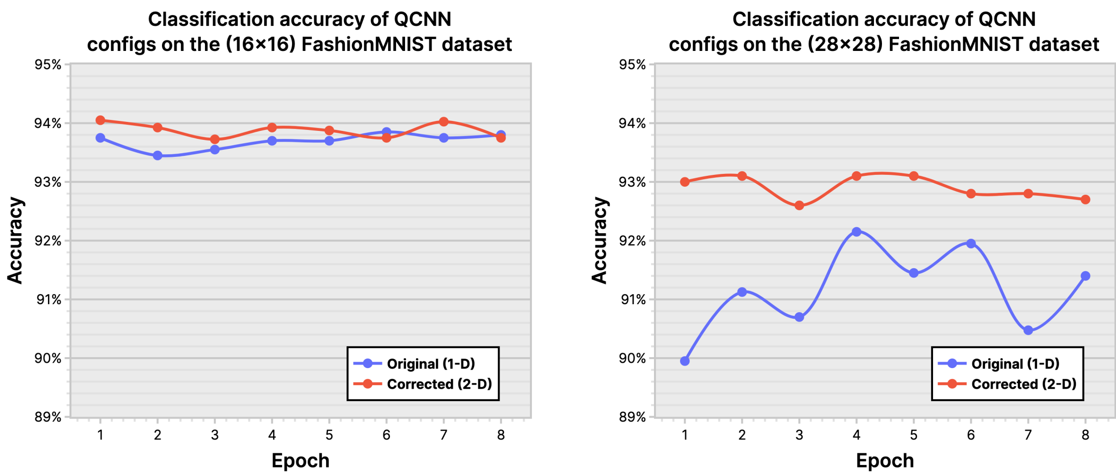
Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 March 2024 doi:10.20944/preprints202403.1060.v1



(a) (16 × 16) dataset **(b)** (28 × 28) dataset

Figure 25. Classification accuracy of Quanvolution Neural Network configurations on the FashionMNIST dataset.

Quantum Convolutional Neural Networks: We based our implementation of QCNN on [53], however, some modifications were made to the technique to work around limitations present in the data encoding method. When encoding data that is not of size 2^n in each dimension, the original method flattens (vectorizes) the input before padding with zeros, as opposed to padding each dimension. However, this sacrifices the advantage of multidimensional encoding, where each dimension is mapped to a region of qubits. To ensure a level field between QCNN and MQCC, the (16 × 16) and (28 × 28) FashionMNIST datasets were tested both for the original (1-D) and a corrected (2-D) data encoding configuration of QCNN, the results of which are shown in Figure 26. As expected, we see a clear improvement on the (28 × 28) dataset and based on this, we chose the corrected (2-D) data encoding method as our baseline QCNN for comparison against other ML models.



(a) (16 × 16) dataset **(b)** (28 × 28) dataset

Figure 26. Classification accuracy of Quantum Convolutional Neural Network (QCNN) configurations on the FashionMNIST dataset,

5.3. Results and Analysis

We first present the results of the quantum convolution operations on data with varying dimensionalities. Then, we compare the fidelity results of the quantum convolution under a noisy simulation environment with reference to classical convolution implementation. Finally, we present the results for MQCC.

5.3.1. Quantum Convolution Results

Similar to pooling, the fidelity of the quantum convolution technique was tested in both a noise-free and noisy environment against a classical implementation using common (3×3) and (5×5) filter kernels. These kernels, as described in [41–44[, include the Averaging F_{avg}, Gaussian blur F_{blur}, Sobel edge-detection $F_{\text{Sx}}/F_{\text{Sy}}$, and Laplacian of Gaussian blur (Laplacian) $F_{\mathcal{L}}$ filters. To enable a quantum implementation of these kernels, a classical pre-processing step zero-padded each kernel until the dimensions were an integer power of two. As negative values may occur in classical convolution, the magnitudes of the output values were cast into a single-byte range $[0, 255]$ in a classical post-processing step.

$$F_{\text{avg}}^{3\times3} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad (41a) \qquad F_{\text{avg}}^{5\times5} = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (41b)$$

$$F_{\text{blur}}^{3\times3} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, \quad (42a) \qquad F_{\text{blur}}^{5\times5} = \frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix} \quad (42b)$$

$$F_{\text{Sx}} = \frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \quad (43a) \qquad F_{\text{Sy}} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (43b)$$

$$F_{\mathcal{L}}^{3\times3} = \frac{1}{6}\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad (44a) \qquad F_{\mathcal{L}}^{5\times5} = \frac{1}{20}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & -24 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (44b)$$

A 1-D averaging kernel of sizes (1×3) and (1×5) was applied to audio files after pre-processing, described in Section 5.1, with data size ranging from 2^8 to 2^{20} data points. Figure 27 presents the reconstructed output data of this operation, with Figure 28 displaying the associated calculated fidelity.

The 2-D Averaging, Gaussian blur, Sobel edge-detection, and Laplacian kernels were applied to 2-D black-and-white and 3-D RGB *Jayhawk* images see Figure 23, ranging from (8×8) to (512×512) pixels and $(8 \times 8 \times 3)$ pixels to $(512 \times 512 \times 3)$ pixels, respectively. The reconstruction from convolution operations in classical, noise-free, and noisy environments on (128×128) and $(128 \times 128 \times 3)$-pixel input images can be seen in Tables 1 and 2, respectively.

Finally, a 3-D averaging kernel of sizes $(3 \times 3 \times 3)$ and $(5 \times 5 \times 5)$ was applied to hyperspectral images from the KSC dataset [43] with pre-processing applied to ensure images were resized to a power of two, ranging from $(8 \times 8 \times 8)$ pixels to $(128 \times 128 \times 128)$ pixels in size. Figure ?? shows the reconstructed output images, while Figure 30 illustrates the fidelity of the operation compared to its classical counterpart.

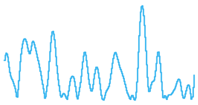
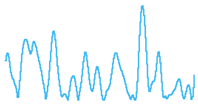
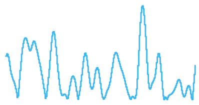
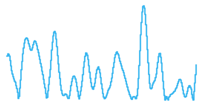
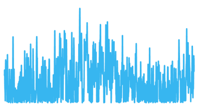
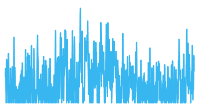
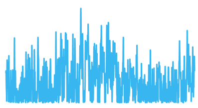
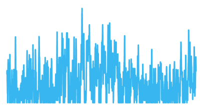
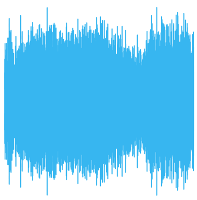
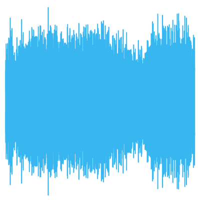
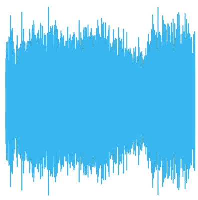
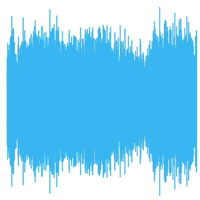
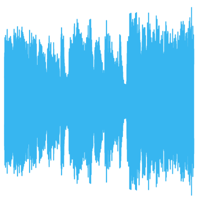
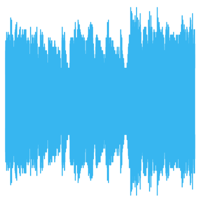
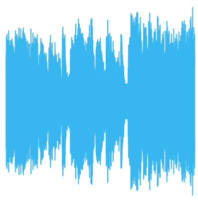
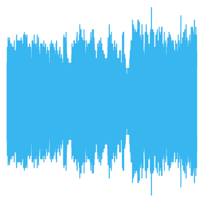
Data Size (No. of Sample Points)	(3) Averaging Classical/ Noise-Free	(3) Averaging Noisy (10^6 shots)	(5) Averaging Classical/ Noise-Free	(5) Averaging Noisy (10^6 shots)
256				
4096				
65,536				
1,048,576				

Figure 27. The 1-D convolution (averaging) filters applied to 1-D audio samples [41].

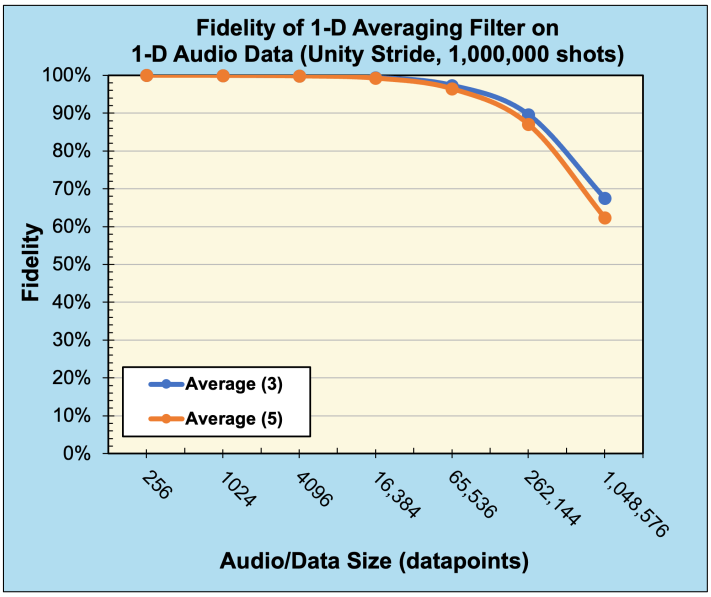
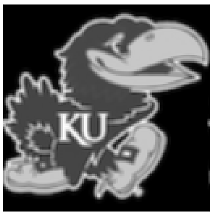
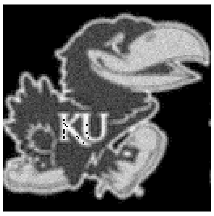
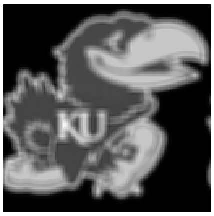
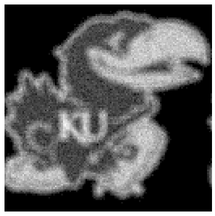
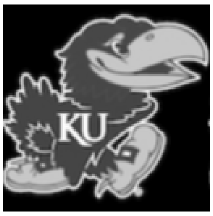
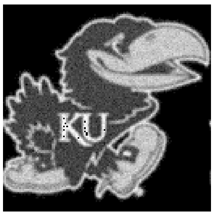
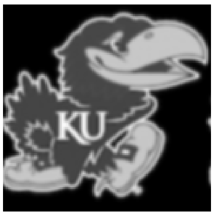
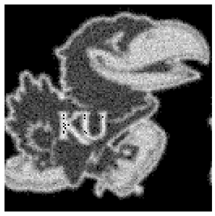
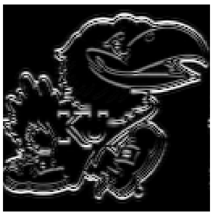
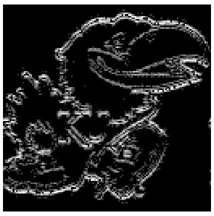


Figure 28. Fidelity of 1-D convolution (averaging) filters with unity stride on 1-D audio data (sampled with 1,000,000 shots).

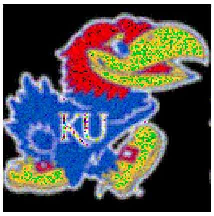
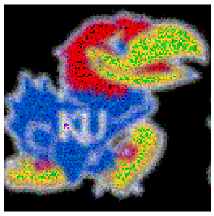
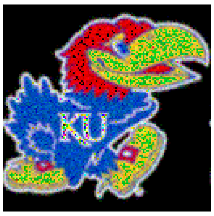
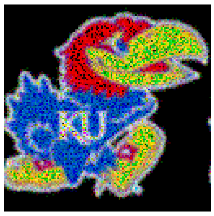
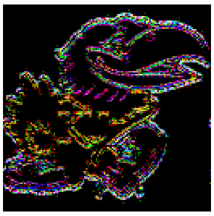
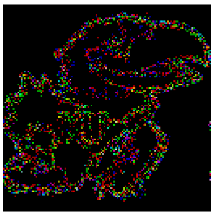
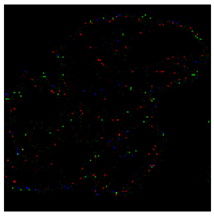
 doi:10.20944/preprints202403.1060.v1

Table 1. Two-dimensional (2-D) convolution kernels applied to a (128 × 128) black-and-white (B/W) image [42].

Kernel	(3 × 3) kernel Classical / Noise-Free	(3 × 3) kernel Noisy (10^6 shots)	(5 × 5) kernel Classical / Noise-Free	(5 × 5) kernel Noisy (10^6 shots)
Average				
Gaussian				
Sobel-X			N/A	N/A
Sobel-Y			N/A	N/A
Laplacian				

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 March 2024 doi:10.20944/preprints202403.1060.v1

Table 2. 2-D convolution kernels applied to a (128 × 128 × 3) color (RGB) image [42].

Kernel	(3 × 3) kernel Classical / Noise-Free	(3 × 3) kernel Noisy (10^6 shots)	(5 × 5) kernel Classical / Noise-Free	(5 × 5) kernel Noisy (10^6 shots)
Average				
Gaussian				
Sobel-X			N/A	N/A
Sobel-Y			N/A	N/A
Laplacian				

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 March 2024 doi:10.20944/preprints202403.1060.v1

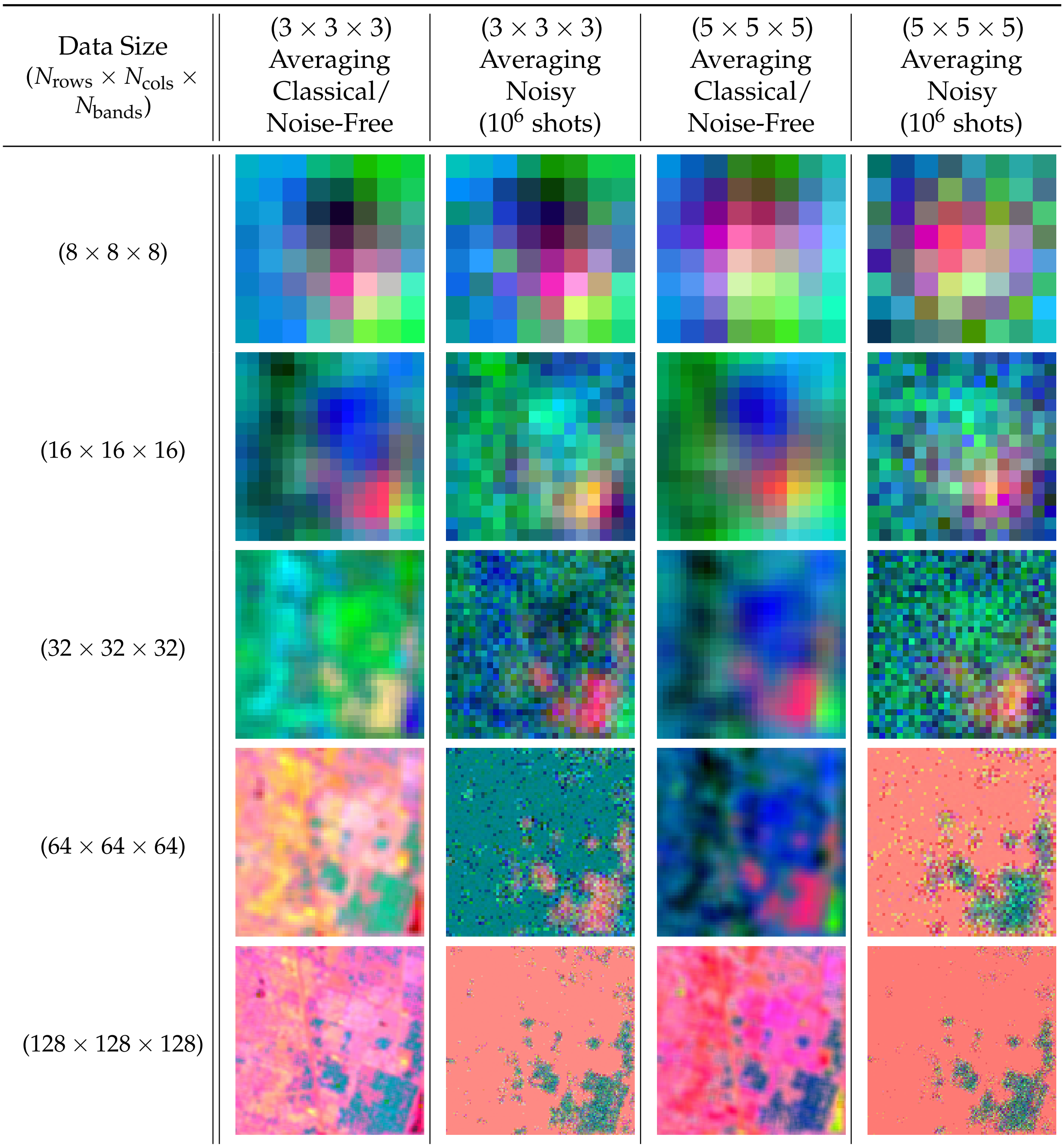


Figure 29. The 3-D convolution (averaging) filters applied to 3-D hyperspectral images (bands 0, 1, and 2) [43].

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 March 2024 doi:10.20944/preprints202403.1060.v1

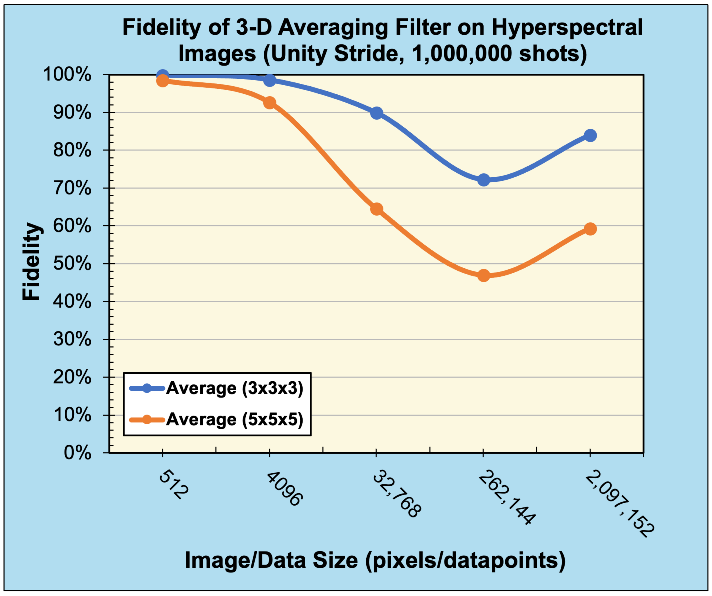


Figure 30. Fidelity of 3-D convolution (averaging) filters with unity stride on 3-D hyperspectral data (sampled with 1,000,000 shots).

Compared to the expected, classically generated results, the noise-free quantum results tested at 100% fidelity across all trials. Therefore, in a noise-free environment, given the same inputs, the proposed convolution techniques have no degradation compared to classical convolution. The fidelity of noisy simulation using 1-D audio and 3-D hyperspectral data are presented in Figures 28 and 30, respectively. The fidelity degradation in both figures is due to the statistical noise where the constant shot count (number of circuit samples) becomes less and less sufficient to describe the increasing data size.

6. Discussion

In this section, we discuss the results of our experiments with MQCC and compare them against the other models in terms of the number of required training parameters, the accuracy of the model, and circuit depth of the implemented model.

6.1. Number of parameters

Among the ML models evaluated, MQCC had the fewest trainable parameters, see Figure 31. This implies potential advantages such as reduced memory usage and faster performance when using classical gradient descent. Although the reduction in parameter decreases from (MLP to CNN) and then further from (CNN to MQCC), parameter reduction diminishes from (MLP to CNN) and further from (CNN to MQCC), there is still a significant 83.62% decrease in parameter count.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 March 2024 doi:10.20944/preprints202403.1060.v1

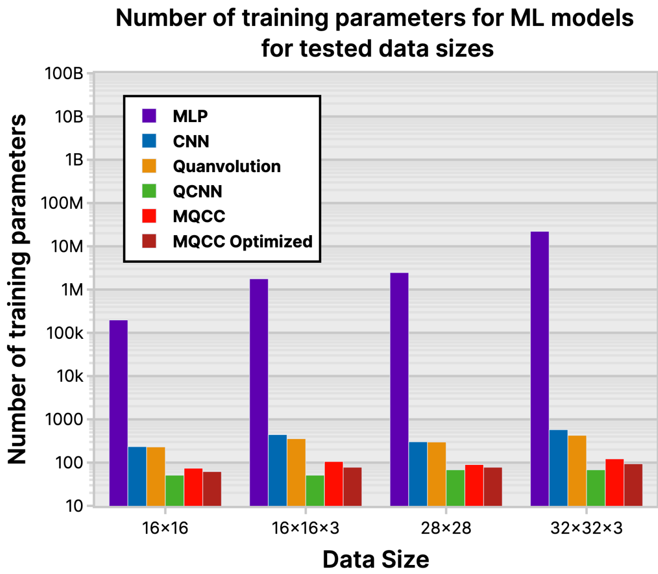
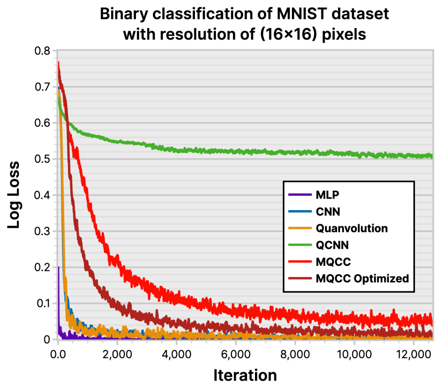


Figure 31. Number of training parameters for ML models for tested data sizes

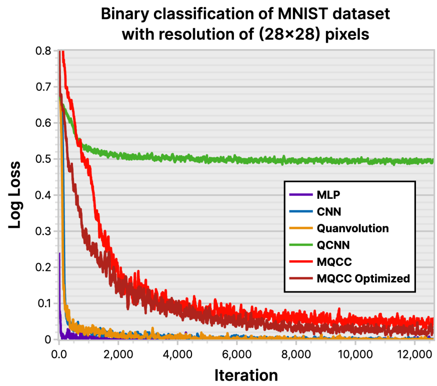
6.2. Loss History and Accuracy

ML-based classifiers aim to maximize the accuracy of their classifications, measured by a loss function during training to estimate the accuracy that may be exhibited when deployed. Hence, Figures 32 and 33 depict the performance of the ML models across the experimental datasets in their plotting of log-loss history and classification accuracy. The MNIST [45] dataset is not complex enough to effectively distinguish models, however, differences begin to emerge in the FashionMNIST [46] and CIFAR10 [47] datasets. MLP consistently achieves the highest accuracy across trials due to its larger parameter count, allowing for greater flexibility in adapting to nuances in input. CNN showcases its ability to select relevant input features using convolution and data locality, demonstrating the second-highest accuracy. Among the tested models, QCNN generally performs the poorest, displaying its inability to properly leverage data locality. However, comparing the accuracy of MQCC and quanvolutional neural networks is inconclusive. Quanvolutional neural networks performed better on FashionMNIST, whereas MQCC performed better on CIFAR10.

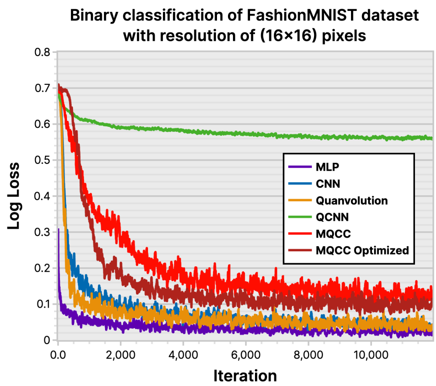
Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 March 2024 doi:10.20944/preprints202403.1060.v1



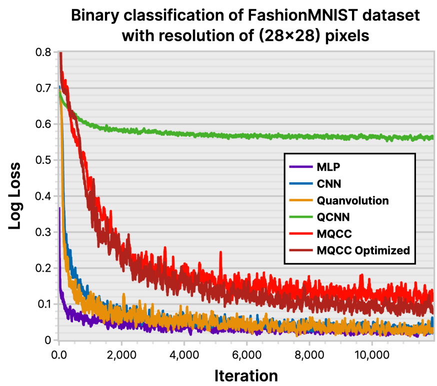
(a) MNIST (16 × 16)



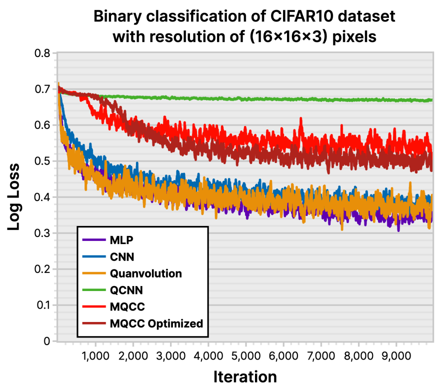
(b) MNIST (28 × 28)



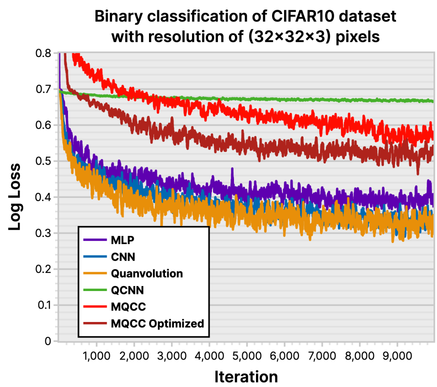
(c) FashionMNIST (16 × 16)



(d) FashionMNIST (28 × 28)



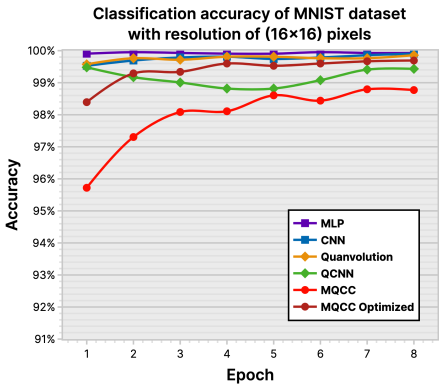
(e) CIFAR10 (16 × 16 × 3)



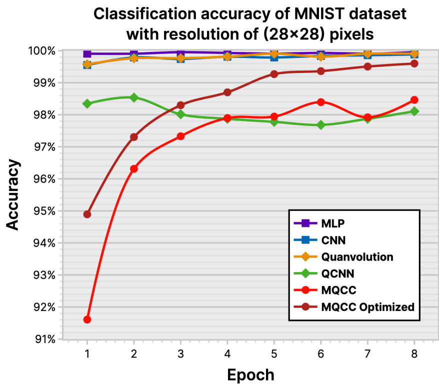
(f) CIFAR10 (32 × 32 × 3)

Figure 32. Loss history of ML models on various datasets.

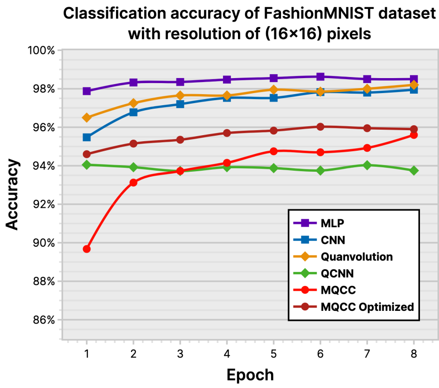
Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 18 March 2024 doi:10.20944/preprints202403.1060.v1



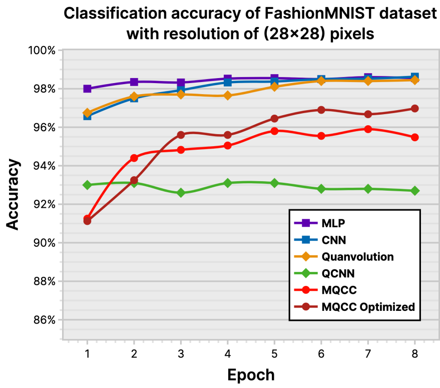
(a) MNIST (16 × 16)



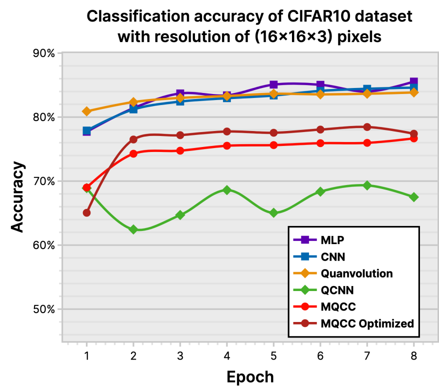
(b) MNIST (28 × 28)



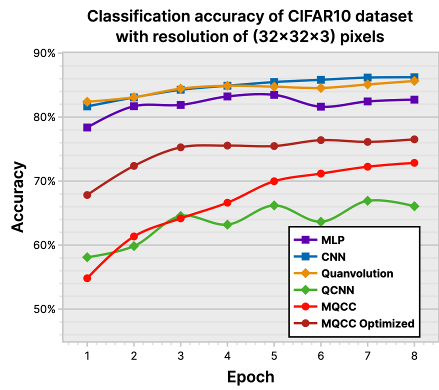
(c) FashionMNIST (16 × 16)



(d) FashionMNIST (28 × 28)



(e) CIFAR10 (16 × 16 × 3)



(f) CIFAR10 (32 × 32 × 3)

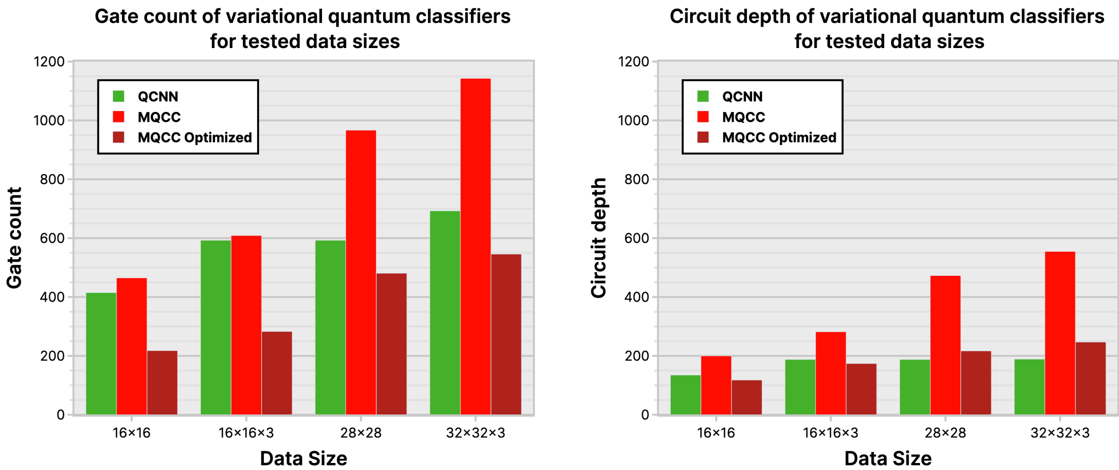
Figure 33. Classification accuracy of ML models on various datasets.

6.3. Gate Count and Circuit Depth

Although comparing MQCC, MQCC Optimized, and QCNN with quantum metrics like gate count and circuit depth is viable, see Figure 34, it is challenging to include quanvolutional neural networks in the comparison due to their significant differences from the other models. These differences are due to the quantum component within quanvolutional neural networks constituting a small fraction of the entire algorithm, bringing it closer to a classical algorithm than a quantum algorithm. Meanwhile, comparing the techniques of MQCC and QCNN in Figure 34, highlights the rationale behind developing MQCC Optimized. Initially, MQCC performed worse than QCNN in gate count and circuit depth. However, after optimizations, MQCC matches the performance of QCNN and even outperforms it in best-case scenarios. While the QCNN architecture appears more suitable for shallower quantum circuits than MQCC, it is because the high parallelization of each QCNN layer

 doi:10.20944/preprints202403.1060.v1

halves the active qubits. Despite QCNN using half the active qubits per layer than MQCC, MQCC utilizes the extra qubits for weights and features, with each pooling layer reducing the qubit count by a constant amount, n_k. However, as QCNN's structure was motivated by the classical convolution operation, they usually need more complex and deeper "convolution" and "pooling" ansatz to attain a higher accuracy.



(a) Gate count **(b)** Circuit depth

Figure 34. Gate count and circuit depth of MQCC vs QCNN.

7. Conclusions

In this paper, we presented the multidimensional quantum convolutional classifier (MQCC) that consists of quantum convolution, quantum pooling, and a quantum fully-connected layer. We leveraged existing convolution techniques to support multiple features/kernels and utilized it in our proposed method. Furthermore, we proposed a novel width-optimized quantum circuit that reuses freed-up qubits from pooling layer in the subsequent convolutional layer. The proposed MQCC additionally preserves data locality in the input data which has shown to improve data classification in convolutional classifiers. The MQCC methodology is generalizable for any arbitrary multidimensional filter operation and pertinent for multi-feature extraction. The proposed method can also support data of arbitrary dimensionality since the underlying quantum pooling and convolution operations are generalizable across data dimensions. We experimentally evaluated the proposed MQCC on various real-world multidimensional images, utilizing several filters through simulations on state-of-the-art quantum simulators from IBM Quantum and Xanadu. In our experiments, MQCC produced higher classification accuracy while having a lower training time and reduced quantum gate count. In our future work, we are planning on expanding MQCC with additional convolution capabilities such as arbitrary striding and dilation and further optimizing it for deployment on real-world quantum processors. In addition, we will investigate using our proposed quantum techniques for real-life applications such as medical imaging and classification.

Author Contributions: Conceptualization: E.E., M.J., D.K., and V.J.; Methodology: E.E., M.J., D.K., and V.J.; Software: E.E., M.J., D.K., and D.L.; Validation: E.E., M.J., V.J., A.N., D.K., and D.L.; Formal analysis: E.E., M.J., D.L., D.K., and V.J.; Investigation: E.E., M.J., V.J., A.N., D.K., D.L., I.I., M.C., A.F., E.V., M.S., A.A., and E.B.; Resources: E.E., M.J., A.N., D.L., V.J., M.C., D.K., A.F., I.I., E.B., M.S., A.A., and E.V.; Data curation: E.E., M.J., V.J., A.N., D.K., D.L., I.I., M.C., A.F., E.V., M.S., E.B., and A.A.; Writing—original draft preparation: E.E., A.N., M.J., V.J., D.K., D.L., I.I., and M.C.; Writing—review and editing: E.E., M.J., V.J., A.N., D.K., D.L., I.I., M.C., A.F., E.V., M.S., E.B., and A.A.; Visualization: E.E., M.J., V.J., A.N., D.K., D.L., I.I., M.C., A.F., E.V., M.S., E.B., and A.A.; Supervision: E.E.; Project administration: E.E.; Funding acquisition: E.E. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

 doi:10.20944/preprints202403.1060.v1

Data Availability Statement: The audio samples used in this work are publicly available from the European Broadcasting Union at https://tech.ebu.ch/publications/sqamcd (accessed on 23 February 2024) as file `64.flac` [41]. The hyperspectral data used in this work are publicly available from the Grupo de Inteligencia Computacional (GIC) at https://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes#Kennedy_Space_Center_(KSC) (accessed on 23 February 2024) under the heading Kennedy Space Center (KSC) [43].

Acknowledgments: This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Villalobos, P.; Sevilla, J.; Besiroglu, T.; Heim, L.; Ho, A.; Hobbhahn, M. Machine learning model sizes and the parameter gap. *arXiv* **2022**, *preprint arXiv:2207.02852*.
2. LeCun, Y.; Kavukcuoglu, K.; Farabet, C. Convolutional networks and applications in vision. In Proceedings of the 2010 IEEE International Symposium on Circuits and Systems, Paris, France, 30 May–2 June 2010; IEEE: New York, NY, USA, 2010, pp. 253–256.
3. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. https://doi.org/10.1038/nature14539.
4. Chakraborty, S.; Dey, L. *Computing for Data Analysis: Theory and Practices*; Springer Nature Singapore 2023, pp. 161–178.
5. Peruzzo, A.; McClean, J.; Shadbolt, P.; Yung, M.H.; Zhou, X.Q.; Love, P.J.; Aspuru-Guzik, A.; O'Brien, J.L. A variational eigenvalue solver on a photonic quantum processor. *Nature Commun.* **2014**, *5*, 4213. https://doi.org/10.1038/ncomms5213.
6. Havlíček, V.; Córcoles, D. and Temme, K.; Harrow, W.; Kandala, A.; Chow, J.; Gambetta, M. Supervised learning with quantum-enhanced feature spaces. *Nature* **2019**, *567*, 209–212. https://doi.org/10.1038/s41586-019-0980-2.
7. Farhi, E.; Goldstone, J.; Gutmann, S. A quantum approximate optimization algorithm. *arXiv* **2014**, *preprint arXiv:1411.4028*.
8. Farhi, E.; Neven, H. Classification with quantum neural networks on near term processors. *arXiv* **2018**, *preprint arXiv:1802.06002*.
9. Schuld, M.; Bocharov, A.; Svore, M. and Wiebe, N. Circuit-centric quantum classifiers. *Physical Review A* **2020**, *101*, 032308. https://doi.org/10.1103/PhysRevA.101.032308.
10. Anton, F. K.; Franco, N. *Quantum Bits with Josephson Junctions*; Springer, Cham. 2019.
11. Williams, C.P.; *Explorations in Quantum Computing*; 2nd edition: Springer-Verlag London Limited 2011.
12. Shende, V.; Bullock, S.; Markov, I. Synthesis of quantum-logic circuits. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2006**, *25*, 1000–1010. https://doi.org/10.1109/TCAD.2005.855930.
13. Balauca, S.; Arusoaie, A. Efficient Constructions for Simulating Multi Controlled Quantum Gates. Computational Science – ICCS, 22nd International Conference, London, UK, June 21–23, 2022, *25*, 179–194. Springer International Publishing https://doi.org/10.1007/978-3-031-08760-8_16.
14. Jeng, M.; Islam, S.I.U.; Levy, D.; Riachi, A.; Chaudhary, M.; Nobel, M.A.I.; Kneidel, D.; Jha, V.; Bauer, J.; Maurya, A.; Mahmud, N.; El-Araby, E. Improving quantum-to-classical data decoding using optimized quantum wavelet transform. *J. Supercomput.* **2023**, *79*, 20532–20561. https://doi.org/10.1007/s11227-023-05433-7.
15. Jeng, M.; Nobel, A.; Jha, V.; Levy, D.; Kneidel, D.; Chaudhary, M. and Islam, I.; Rahman, M.; El-Araby, E. Generalized Quantum Convolution for Multidimensional Data. *Entropy* **2023**, *25*, *11*. https://doi.org/10.3390/e25111503.
16. Yao, X.; Wang, H.; Liao, Z.; Chen, M.; Pan, J.; Li, J.; Zhang, K.; Lin, X.; Wang, Z.; Luo, Z.; Zheng, W.; Li, J.; Zhao, M.; Peng, X.; Suter, D. Quantum Image Processing and Its Application to Edge Detection: Theory and Experiment. *Phys. Rev. X* **2017**, *7*, 031041. https://doi.org/10.1103/PhysRevX.7.031041.
17. Li, X. and Yang, G. and Torres, M.; Zheng, D.; Wang, L. A Class of Efficient Quantum Incrementer Gates for Quantum Circuit Synthesis. *International Journal of Modern Physics B* **2014**, *28*, 1350191. https://doi.org/10.1142/S0217979213501919 (Preprint).

18. Fan, P.; Zhou, R.; Hu, W.; Jing, N. Quantum image edge extraction based on classical Sobel operator for NEQR. *Quantum Information Processing* **2018**, *18*, 24. https://doi.org/10.1007/s11128-018-2131-3.
19. El-Araby, E.; Mahmud, N.; Jeng, M.J.; MacGillivray, A.; Chaudhary, M.; Nobel, M.A.I.; Islam, S.I.U.; Levy, D.; Kneidel, D.; Watson, M.R.; Jack, B.; Andrew, R. Towards Complete and Scalable Emulation of Quantum Algorithms on High-Performance Reconfigurable Computers. *IEEE Trans. Comput.* **2023**, *72*, 2350–2364. https://doi.org/10.1109/TC.2023.3248276.
20. IBM Quantum. Qiskit: An Open-source Framework for Quantum Computing. *Zenodo*. Available online: https://doi.org/10.5281/zenodo.2573505 (accessed on 28 February 2024)
21. Nation, P. How to measure and reset a qubit in the middle of a circuit execution. **2021**. Available online: https://www.ibm.com/quantum/blog/quantum-mid-circuit-measurement (accessed on 26 February 2024).
22. Mi, A.; Deng, S.; Szefer, J. Securing Reset Operations in NISQ Quantum Computers. *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* **2022** 2279–2293.
23. Mahmud, N.; Jeng, M.; Nobel, Md. Alvir I.; Chaudhary, M.; Ul Islam, S M I.; Levy, D.; El-Araby, E. Efficient Data Encoding and Decoding for Quantum Computing. 2022 IEEE International Conference on Quantum Computing and Engineering (QCE), Broomfield, CO, USA, September 18–23, 2022, 765–768. https://doi.org/10.1109/QCE53715.2022.00110
24. Le, Q.; Dong, F.; Hirota, K. A flexible representation of quantum images for polynomial preparation, image compression, and processing operations. *Quantum Information Processing*, *10*, **2011**, 63–84. https://doi.org/10.1007/s11128-010-0177-y.
25. Zhang, Y.; Lu, K.; Gao, Y.;Wang, M. NEQR: a novel enhanced quantum representation of digital images. *Quantum Information Processing*, *12*, **2013**, 2833-2860. https://doi.org/10.1007/s11128-013-0567-z.
26. Burrus, C.S. S.; Parks, Thomas W. *DFT/FFT and Convolution Algorithms: Theory and Implementation*; John Wiley & Sons, Inc., USA, 1991.
27. Podlozhnyuk, V. FFT-based 2D convolution. *NVIDIA* **2007**. Available online: https://developer.download.nvidia.com/compute/cuda/1.1-Beta/x86_64_website/projects/convolutionFFT2D/doc/convolutionFFT2D.pdf (accessed on 18 February 2024).
28. NVIDIA. CUTLASS Convolution. *Github* **2023**. Available online: https://github.com/NVIDIA/cutlass/blob/main/media/docs/implicit_gemm_convolution.md (accessed on 18 February 2024).
29. NVIDIA. Convolution Algorithms. *Github* **2023**. Available online: https://docs.nvidia.com/deeplearning/performance/dl-performance-convolutional/index.html#conv-algo (accessed on 18 February 2024).
30. Ma, Y.; Ma, H.; Chu, P. Demonstration of Quantum Image Edge Extration Enhancement Through Improved Sobel Operator. *IEEE Access*, *8*, **2020**, 210277-210285. https://doi.org/10.1109/ACCESS.2020.3038891.
31. Zhang, Y.; Lu, K.; Gao, Y. QSobel: A novel quantum image edge extraction algorithm. *Science China Information Sciences*, *58*, **2015**, 1-13. https://doi.org/10.1007/s11432-014-5158-9.
32. Zhou, R.; Yu, H.; Cheng, Y.; Li, F. Quantum image edge extraction based on improved Prewitt operator. *Quantum Information Processing*, *18*, **2019**, 261. https://doi.org/10.1007/s11128-019-2376-5.
33. Li, P. Quantum implementation of the classical Canny edge detector. *Multimedia Tools and Applications*, *81*, **2022**, 11665-116941. https://doi.org/10.1007/s11042-022-12337-w.
34. Cong, I.; Choi, S.; Lukin, M.D. Quantum convolutional neural networks. *Nature Phys.* **2019**, *15*, 1273–1278. https://doi.org/10.1038/s41567-019-0648-8.
35. Henderson, M.; Shakya, S.; Pradhan, S.; Cook, T. Quanvolutional neural networks: powering image recognition with quantum circuits. *Quantum Machine Intelligence* **2020**, 2, 2. https://doi.org/10.1007/s42484-020-00012-y.
36. Bieder, F.; Sandkühler, R.; Cattin, P. Comparison of methods generalizing max-and average-pooling. *arXiv* **2021**,preprint arXiv:2103.01746 https://doi.org/10.48550/arXiv.2103.01746.
37. Jeng, M.; Nobel, A.; Jha, V.; Levy, D.; Kneidel, D.; Chaudhary, M.; Islam, I.; Baumgartner, E.; Vanderhoof, E.; Facer, A.; Singh, M.; Arshad, A.; El-Araby, E. Optimizing Multidimensional Pooling for Variational Quantum Algorithms. *Algorithms* **2024**, *17*, 82. https://doi.org/10.3390/a17020082.
38. PyTorch. torch.nn.LPPool1d [Computer Software]. 2023. Available online: https://pytorch.org/docs/stable/generated/torch.nn.LPPool1d.html (accessed on 18 February 2024).
39. IBM Quantum. qiskit.circuit.library.NLocal [Computer Software]. 2023. Available online: https://qiskit.org/documentation/stubs/qiskit.circuit.library.NLocal.html (accessed on 18 February 2024).

40. Xanadu. qml.BasicEntanglerLayers [computer software]. 2023. Available online: https://docs.pennylane.ai/en/stable/code/api/pennylane.BasicEntanglerLayers.html (accessed on 18 February 2024).
41. Geneva, S. Sound Quality Assessment Material: Recordings for Subjective Tests. Available online: https://tech.ebu.ch/publications/sqamcd, 1988. (accessed on 18 February 2024).
42. Brand Center, University of Kansas. Jayhawk Images. Available online: https://brand.ku.edu/ (accessed on 18 February 2024).
43. Graña, M.; Veganzons, M.A.; Ayerdi, B. Hyperspectral Remote Sensing Scenes. Available online: https://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes#Kennedy_Space_Center_(KSC) (accessed on 18 February 2024).
44. Jeng, M.; Nobel, M.A.I.; Jha, V.; Levy, D.; Kneidel, D.; Chaudhary, M.; Islam, S.I.U.; El-Araby, E. Multidimensional Quantum Convolution with Arbitrary Filtering and Unity Stride. In Proceedings of the IEEE International Conference on Quantum Computing and Engineering (QCE23), Bellevue, WA, USA, 17–22 September 2023.
45. Deng, L. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine; IEEE*, **2012**, *29*, 141–142. https://doi.org/10.1109/MSP.2012.2211477.
46. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv*, **2017**. https://doi.org/10.48550/arXiv.1708.07747.
47. Krizhevsky, A. Learning Multiple Layers of Features from Tiny Images. **2009**. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf (accessed on 18 February 2024).
48. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; Chintala, S. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* **2019** *32*. https://dl.acm.org/doi/10.5555/3454287.3455008.
49. Bergholm, V.; Izaac, J.; Schuld, M.; Gogolin, C.; Ahmed, S.; Ajith, V.; Alam, M.S.; Alonso-Linaje, G.; AkashNarayanan, B.; Asadi, A.; Arrazola, J.M. PennyLane: Automatic differentiation of hybrid quantum-classical computations. *arXiv* **2022**. Available online: https://pennylane.ai/ (accessed on 18 February 2024).
50. KU Community Cluster, Center for Research Computing, University of Kansas. Available online: https://crc.ku.edu/systems-services/ku-community-cluster (accessed on 18 February 2024)
51. Kingma, D. P.; Ba, J. Adam: A method for stochastic optimization. *arxiv* **2014**. preprint arXiv:1412.6980.
52. Arora, R.; Basu, A.; Mianjy, P.; Mukherjee, A. Understanding deep neural networks with rectified linear units. *arxiv* **2014**. preprint arXiv:1611.01491.
53. Hur, T.; Kim, L.; Park, D. Quantum convolutional neural network for classical data classification. *Quantum Machine Intelligence, Springer* **2022** *4*. https://doi.org/10.1007/s42484-021-00061-x.