AI

MDPI

Article

H-QNN: A Hybrid Quantum–Classical Neural Network for Improved Binary Image Classification

Muhammad Asfand Hafeez, Arslan Munir * and Hayat Ullah

Department of Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431, USA; mhafeez2024@fau.edu (M.A.H.); hullah2024@fau.edu (H.U.)
* Correspondence: arslanm@fau.edu

Abstract: Image classification is an important application for deep learning. With the advent of quantum technology, quantum neural networks (QNNs) have become the focus of research. Traditional deep learning-based image classification involves using a convolutional neural network (CNN) to extract features from the image and a multi-layer perceptron (MLP) network to create the decision boundaries. However, quantum circuits with parameters can extract rich features from images and also create complex decision boundaries. This paper proposes a hybrid QNN (H-QNN) model designed for binary image classification that capitalizes on the strengths of quantum computing and classical neural networks. Our H-QNN model uses a compact, two-qubit quantum circuit integrated with a classical convolutional architecture, making it highly efficient for computation on noisy intermediate-scale quantum (NISQ) devices that are currently leading the way in practical quantum computing applications. Our H-QNN model significantly enhances classification accuracy, achieving a 90.1% accuracy rate on binary image datasets. In addition, we have extensively evaluated baseline CNN and our proposed H-QNN models for image retrieval tasks. The obtained quantitative results exhibit the generalization of our H-QNN for downstream image retrieval tasks. Furthermore, our model addresses the issue of overfitting for small datasets, making it a valuable tool for practical applications.

Keywords: quantum machine learning; hybrid quantum–classical neural networks; quantum convolutional neural networks; convolutional neural networks; classification; image retrieval

Citation: Hafeez, M.A.; Munir, A.; Ullah, H. H-QNN: A Hybrid Quantum–Classical Neural Network for Improved Binary Image Classification. *AI* **2024**, *5*, 1462–1481. https://doi.org/10.3390/ai5030070

Academic Editor: Rafał Drężewski

Received: 2 July 2024
Revised: 8 August 2024
Accepted: 9 August 2024
Published: 19 August 2024

1. Introduction

Over the last two decades, neural networks have significantly impacted various domains through their capacity to model intricate patterns and generate accurate predictions. One area that has particularly advanced is image classification, which categorizes images into predefined classes [1]. Binary image classification, which aims to differentiate between two classes, is especially prevalent in applications such as medical imaging [2], object detection [3], and quality control in manufacturing. Traditional methodologies rely on convolutional neural networks (CNNs) for their adept feature extraction capabilities [4,5], allowing them to identify patterns within images. However, with the growing complexity and volume of data, these conventional models often necessitate substantial computational resources, resulting in processing speed and energy efficiency bottlenecks [6,7]. Hence, there is a need to explore more efficient and resilient computational models. This challenge calls for a novel approach capable of leveraging the computational advantages of quantum mechanics.

The integration of quantum computing with classical machine learning (ML) techniques has given rise to the innovative field of quantum ML (QML) [8,9]. The superposition and entanglement properties of quantum mechanics provide a theoretical basis for deducing complex correlations within data. Quantum algorithms, such as Grover's algorithm,

offer significant speedups in tasks like database searching, which is analogous to information retrieval in machine learning, including image classification. Quantum circuits can also utilize high-dimensional Hilbert spaces, making it easier to extract features that are difficult for classical networks to identify. This enhanced feature extraction capability is crucial for improving model accuracy and generalization. In addition, quantum algorithms exhibit the potential to facilitate more efficient optimization processes. By utilizing quantum parallelism, these algorithms can explore complex loss landscapes more effectively, leading to faster convergence and better optimization outcomes. This is particularly beneficial in the context of deep learning, where optimization challenges are prevalent.

Despite the advantages offered by quantum computing and QML, the practical realization of QML models is constrained by the current stage of quantum technology, characterized by noisy intermediate-scale quantum (NISQ) [10,11] devices that offer only a limited number of qubits and are prone to errors. The main challenge is to create a practical and efficient quantum–classical hybrid model that can leverage the strengths of both approaches to achieve high accuracy in image classification tasks. The model needs to be designed to work with the limited qubit capacities of NISQ devices while also demonstrating superior efficiency compared to classical-only models. The nascent state of quantum technologies and the intrinsic noise present in current quantum devices partially hinder the full exploitation of QML.

In the specific area of image classification, CNNs [12–15] have become the benchmark for image classification. Their capability in feature extraction and recognition tasks has set a high standard, consistently achieving high accuracy levels. Notably, various studies have demonstrated their effectiveness in different domains, such as medical imaging [13], general object classification [16], and image segmentation [17]. Recently, quantum neural networks (QNNs) have emerged to extend this success to the quantum realm. These networks utilize parameterized quantum circuits, which involve manipulation through rotation angles, entanglement, and measurement. Rotation angles are used to control qubit states, allowing precise management of qubit superposition. Entanglement operations create correlations between qubits, enabling intricate interactions crucial for quantum computation. Measurements, typically performed along specific bases like the Z-basis, extract information from the quantum system by converting qubit states into classical bits. Rotation angles, entanglement, and measurement work together to adjust and optimize the circuit parameters during the training process, similar to how weights and biases are adapted in classical neural networks.

This paper presents a novel model for binary image classification that employs a hybrid quantum–classical approach. By combining the strengths of both domains, the limitations of current quantum hardware and the computational demands of CNN models can be addressed. The main contributions of this work are as follows:

- We developed a two-qubit, six-layer H-QNN model specifically designed to meet the qubit limitations of NISQ devices for image classification. Our proposed model represents a significant advancement in quantum-enhanced image classification despite the limited quantum resources available.
- The proposed quantum convolutional architecture offers an efficient solution in terms of achieving high accuracy with a smaller number of images for binary classification tasks as compared to a similar CNN architecture. It also enhances the discriminatory power of high-accuracy image classification. Additionally, the proposed architecture provides the flexibility to handle multi-class classification.
- Our H-QNN model combines quantum computing with classical neural networks to optimize computational resources across the quantum–classical divide. This offers a scalable and effective approach to QML in the NISQ era.

The remainder of this paper is organized as follows. Section 2 presents the background of the proposed study and reviews related work in the literature. Section 3 elaborates the proposed H-QNN model. In Section 4, we present and analyze our experimental results.

Section 5 discusses the limitations of the proposed work. Finally, Section 6 concludes this paper.

2. Background

This section provides insight into QNNs by presenting essential concepts, background information, and related work. It covers fundamental topics such as quantum qubits, quantum gates, and circuits, providing an overview of the QNN's development, including its history, evolution, and current state. Additionally, it discusses the latest research in the field of QNNs, including advancements in quantum computing hardware, quantum algorithms, and the integration of QNNs with classical neural networks.

2.1. Preliminaries

2.1.1. Qubits, Superposition, Quantum Gates, Entanglement, and Measurement

Qubits: In classical computing, the basic unit of information is the bit, which can represent either *0* or *1*. In quantum computing, the fundamental unit is the qubit, which can represent *0*, *1*, or a combination of $|0\rangle$ and $|1\rangle$ simultaneously due to a property called superposition. This ability to exist in multiple states at once is what gives quantum computers their potential for exponentially increased computational power as compared to classical computers. There are several types of qubit technologies available, such as superconducting qubits, trapped ions, neutral atoms, and silicon spin qubits.

Superposition: Superposition is a fundamental principle of quantum mechanics that allows qubits to exist in multiple states simultaneously. For example, while a classical bit can be either *0* or *1*, a qubit can be in a superposition of both states, meaning it represents both *0* and *1* at the same time until measured. This property enables quantum computers to perform many calculations simultaneously, vastly speeding up certain types of computations.

Quantum Gates: Quantum gates are the building blocks of quantum circuits, analogous to classical logic gates. They operate on qubits to perform operations such as flipping the state of a qubit, entangling qubits, or creating superpositions. Common quantum gates include the Pauli-X gate (similar to a classical NOT gate), the Hadamard gate (creates superposition), and the CNOT gate (entangles two qubits). These gates manipulate the quantum states of qubits to perform computations. A quantum circuit often includes various gate operations.

Entanglement: Entanglement is another unique property of quantum mechanics that allows qubits to be correlated in such a way that the state of one qubit depends on the state of another, even when they are separated by large distances. This phenomenon enables quantum computers to perform certain calculations much faster than classical computers.

Measurement: Measurement in quantum computing refers to the process of extracting information from a qubit. When a qubit is measured, its superposition collapses, and it takes on a definite state (either *0* or *1*) with a certain probability determined by the coefficients of its superposition. The act of measurement irreversibly changes the state of the qubit. In physical quantum computers, measurements are usually confined to a computational basis, such as the Z-basis. For example, in IBM quantum computers, measurements are typically performed along the Z-axis of the Bloch sphere. This Z-basis measurement determines whether the qubit state collapses to $|0\rangle$ or $|1\rangle$. Other bases, like the X-basis or Y-basis, can also be used, but they require additional quantum gates to rotate the qubit states into the computational basis before measurement. This approach simplifies the implementation of measurement but restricts direct measurement to the computational basis.

2.1.2. Quantum Neural Network

A QNN involves parameter optimization of a parameterized quantum circuit to obtain a desired input–output relationship. A QNN generally consists of three segments: (i) a classical-to-quantum data encoding (or embedding) circuit, (ii) a parameterized circuit, and (iii) measurement operations.

In the literature, several encoding methods are available [18]. The angle encoding scheme is the most widely used method for continuous variables. This method encodes a continuously variable input classical feature as a qubit rotation along the desired axis (X/Y/Z) [18–20]. If we have 'n' classical features, we require $\log_2 n$ qubits. We can encode multiple continuous variables in a single qubit using sequential rotations. This means applying multiple rotation gates successively on the same qubit, with each gate representing a different feature. The states produced by a qubit rotation along any axis repeat in 2π intervals, so we scale features between 0 and 2π (or $-\pi$ to π) in a data preprocessing step.

The parametric circuit consists of entangling operations and parameterized single-qubit rotations. The entanglement operations are a set of multi-qubit operations between all the qubits to generate correlated states [21]. This combination of entangling and single-qubit rotation operations is referred to as a parametric layer in the QNN [19,22].

2.2. Related Work

Recently, QML has emerged as a rapidly growing field in the realm of neural networks. Researchers all over the world are showing interest in QML due to its potential to leverage quantum computational supremacy to solve complex problems that classical computers cannot handle. QML combines principles from quantum physics with machine learning algorithms, leading to potential computational speed-ups in several areas like data analysis, image classification, and artificial intelligence. A study by Shephard et al. [23] showed that QML has significant potential in data analysis, while Ayoade et al. [24] worked on image classification using QML. Furthermore, Oroy [25] showed that QML has the potential to revolutionize the field of artificial intelligence.

One of the key advancements in this domain is the development of QNNs, which attempt to emulate the neural network architecture within a quantum framework. QNNs offer a novel approach to processing information through quantum bits (qubits) that can exist in superpositions, thereby drastically increasing the computational power for certain tasks [26,27]. The appeal of QNNs lies in their theoretical ability to capture correlations in quantum data more naturally than their classical counterparts, promising significant improvements in learning efficiency and performance metrics such as accuracy, precision, recall, and F1-score [28].

Due to the current limitation on the number of qubits and availability of only NISQ devices, researchers are exploring the use of the hybrid QCNN, which is a promising development in the quantum–classical interface. For instance, Alam et al. [29] proposed two quantum–classical H-QNN models for image classification, namely, the quanvolutional neural network and dimension reduction using a classical algorithm followed by QNN. Similarly, Mahajan [30] proposed a quantum-inspired hybrid model that uses quantum and classical neurons for commodity price prediction. Samuel et al. [31] developed a hybrid quantum–classical graph convolutional neural network (QGCNN) for learning high-energy physics (HEP) data. This demonstrates the broad applicability of hybrid QNNs. Furthermore, Huang et al. [32] recently assessed the feasibility of the hybrid quantum model in the context of classical adversarial examples, demonstrating its superior performance over CNNs.

Our proposed H-QNN model builds upon the advancements in the field of QML, which have been designed explicitly for binary image classification. Unlike previous models, our proposed H-QNN utilizes a compact quantum circuit integrated with a classical convolutional architecture, making it efficient for NISQ devices. This design addresses qubit limitations, enhances classification accuracy, and tackles overfitting in small datasets.

3. Methodology

This section presents our proposed combined classical and quantum computing methods. Specifically, we partially convert a classical neural network to a quantum neural network to create a hybrid quantum–classical neural network.

3.1. Overview

In order to integrate quantum and classical neural networks, a hidden layer is incorporated into our neural network, known as a parameterized quantum circuit (discussed in Section 2.1.2). The circuit employs a classical input vector to configure the rotation angles of the quantum gates. The output from the preceding neural network layer is fed as input to this parameterized circuit. The measurement statistics from the parameterized circuit are gathered and utilized as inputs for the succeeding layer of the neural network. This process is repeated until the final output layer is reached. The simplified architecture of the proposed model is depicted in Figure 1.

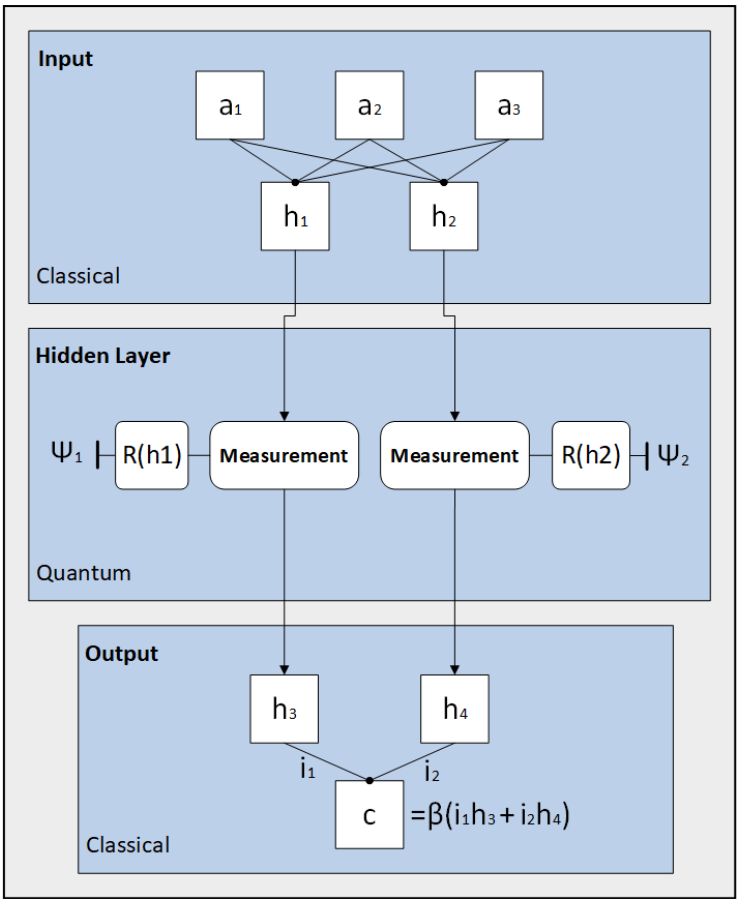


Figure 1. Overview of the proposed H-QNN model.

In Figure 1, β is a nonlinear function, and h_n is the value of neuron n at each hidden layer. $R(h_n)$ represents any rotation gate about an angle equal to h_n, and c is the final prediction value generated from the hybrid network.

3.2. Quantum Circuit (Parameterized Circuit)

Algorithm 1 provides a simplified interface for creating and executing parameterized quantum circuits. The *QuantumCircuit* class consists of two main procedures: *INITIALIZE* and *RUN*.

Referring to Algorithm 1, the *INITIALIZE* procedure creates a quantum circuit with a specified number of qubits (*n_qubits*), which is 2 qubits in our case, the *backend* (simulator) on which the circuit will run, and the number of repetitions for executing the circuit (*shots*). There are different options of *backend* simulators available, such as *qasm_simulator*, *statevector_simulator*, and *aer_simulator*. However, in our H-QNN model, we used the *aer_simulator* backend.

The algorithm introduces a parameter, θ, designed to facilitate the dynamic adjustment of gate operations within the circuit (line 4). The circuit configuration commences by applying a Hadamard gate H to all qubits, thus placing them into a superposition state to utilize quantum parallelism (line 5). Mathematically, the Hadamard gate is represented as

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{1}$$

Applying H to a basis state $|0\rangle$ or $|1\rangle$ results in

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \quad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{2}$$

This creates an equal probability of the qubit being in the state $|0\rangle$ or $|1\rangle$, laying the foundation for parallel processing of information.

Algorithm 1 Quantum circuit module.

```
1:  Class QuantumCircuit
2:  procedure INITIALIZE(n_qubits, backend, shots)
3:      C ← Create a quantum circuit with n_qubits
4:      θ ← Define a parameter θ
5:      C ← Apply a Hadamard gate to all qubits
6:      C ← Add a barrier
7:      C ← Apply an R_y gate parameterized by θ to all qubits
8:      C ← Measure all qubits
9:      Set the backend and number of shots for execution
10: end procedure
11:
12: procedure RUN(Θ)
13:     T_C ← Transpile the circuit for the specified backend
14:     for each θ_i in Θ do
15:         T_C ← T_C.assign_parameters({θ → θ_i})
16:         Results ← Execute the circuit on the backend with θ_i
17:     end for
18:     Ex ← []                                   ▷ List to store expected values
19:     for each result in Results do
20:         if isinstance(Results, list) then
21:             for each result in Results do
22:                 counts ← np.array(list(result()))
23:                 states ← np.array([int(state, result))
24:                 probabilities ← counts/shots
25:                 E ← ∑(states × probabilities)
26:                 Ex.append(E)
27:             end for
28:         else if isinstance(results, dict) then
29:             counts ← np.array(list(Results()))
30:             states ← np.array([int(state, result))
31:             probabilities ← counts/shots
32:             E ← ∑(states × probabilities)
33:             Ex.append(E)
34:         end if
35:     end for
36:     return np.array(Ex)
37: end procedure
```

Subsequently, a barrier is introduced (line 6) to demarcate different sections of the circuit. The barrier acts like a synchronization point in parallel computing, ensuring

that operations before and after the barrier are treated as separate during optimization (transpilation). This prevents certain optimizations from spanning across the barrier, maintaining the intended logical separation of circuit segments. Following this, a rotation gate (R_y), parameterized by θ, is applied to all qubits (line 7). This parameterized rotation gate $R_y(\theta)$ allows for dynamic adjustment of the qubit states based on classical inputs. Its matrix form is

$$R_y(\theta) = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \tag{3}$$

The rotation gate $R_y(\theta)$ rotates the qubit state around the Y-axis of the Bloch sphere by an angle θ, which is determined during the training process. The parameter θ enables the quantum circuit to learn and adapt based on the data, effectively parameterizing the quantum neural network. Finally, all qubits are measured (line 8) following the backend, and the number of shots (repetitions) is set (line 9).

The *RUN* procedure builds a quantum circuit for a set of parameter values Θ (lines 12–37). Initially, the circuit is optimized for the specified backend through a process called transpilation (line 13). This involves mapping the circuit to the backend's topology, optimizing gate operations, and reducing the overall complexity. Then, the circuit is parameterized and executed on the backend for each parameter value θ_i in the list Θ (lines 15–16). This allows us to observe how the quantum circuit behaves under various parameter settings. The parameter θ is dynamically assigned different values from the list Θ. Each time, the circuit is executed with a different parameter value during each iteration of the loop. The results of these executions are collected for further analysis. An empty list (**Ex**) (line 18) is initialized to store the expectation values computed from the results.

Each result is processed to compute the expectation values (**E**) (line 25). If the results are in the form of a list (indicating multiple results), the procedure iterates over each result, converting the measurement counts into a *NumPy* array and the binary state strings into integers. The probabilities of each state are calculated by normalizing the counts by the total number of *shots*. The expectation value (**E**) is then computed as the weighted sum of states and their respective probabilities, and this value is appended to the expectations list (**Ex**) (line 26). If the results are a single dictionary (indicating a single result), a similar process is followed: the counts are converted to a NumPy array, states are decoded, probabilities are computed, and the expectation value is calculated and appended to the list.

The *RUN* procedure concludes by returning the list of expectation values (**Ex**) as a *NumPy* array, providing a comprehensive output of the quantum circuit's behavior for the given parameter set (line 36). This structured approach can be used to create a quantum circuit efficiently.

3.3. Hybrid Module

The proposed hybrid module integrates the quantum layer with classical layers. It also comprises two classes: the Hybrid class and the Gradient Calculation (GradCalc) class. Algorithm 2 illustrates the proposed hybrid module that integrates a quantum circuit within a neural network architecture.

Algorithm 2 provides the forward and backward pass methods for gradient calculation (lines 1–8). The GradCalc class handles the integration of the quantum circuit into the gradient computation graph (line 2). Following this, a forward pass computes the expectation values using the quantum circuit (lines 3–5). Similarly, the backward pass computes the gradients of the expectation values (lines 6–8).

The *Hybrid* class extends *nn.Module*, which indicates that it is a custom module within the PyTorch framework (line 9). This inheritance allows the Hybrid class to integrate seamlessly with other PyTorch components. The Hybrid class is initialized with parameters specifying the quantum circuit configuration, such as the number of qubits (Ψ_n), *backend*, and *shots*. It calculates the gradient shift based on these parameters as described in

Section 3.4 (lines 10–13). The Hybrid class constructs a *QuantumCircuit* object and stores it along with the shift (φ) value for later computations.

The forward method is defined for the Hybrid class (lines 15–21), which adjusts the tensor input dimensions if necessary and then applies a gradient function (*GradFunction*). This function integrates the prepared quantum circuit into PyTorch's computational graph, enabling both forward and backward propagation through the network. The conditional input adjustment (lines 16–20) checks and modifies the tensor input shapes to ensure compatibility with the requirements of the quantum circuit. Specifically, if the input tensor has extra dimensions (i.e., the shape is not (1, 1)), it is squeezed to remove these singleton dimensions. If the shape is already (1, 1), the input tensor is indexed to ensure the correct format for the quantum circuit execution. Finally, the quantum circuit is executed using the adjusted input and returns the output of the forward method (line 21).

Algorithm 2 Hybrid quantum–classical module.

1: **Class GradCalc(Inherits Gradient Function)**
2: Integrates & computes gradient graph
3: **procedure** FORWARD_PASS($input$, $quan\_circuit$, φ)
4: Computes expectation values ← $backward\_pass$
5: **end procedure**
6: **procedure** BACKWARD_PASS($grad\_output$)
7: Compute gradients of the expected values
8: **end procedure**

9: **Class** Hybrid **extends** nn.Module
10: **procedure** INITIALIZE(Ψ_n, $backward$, $shots$, φ)
11: Call superclass initializer
12: quan_circuit ← $INITIALIZE(\Psi_n, back, shots)$ // QuantumCircuit Algorithm 1
13: self.shift ← φ
14: **end procedure**
15: **procedure** FORWARD($input$)
16: **if** $input.shape \neq (1,1)$ **then**
17: $input$ ← torch.squeeze($input$)
18: **else**
19: $input \leftarrow input[0]$
20: **end if**
21: **return** FORWARD_PASS($input$, $quan\_circuit$, φ)
22: **end procedure**

3.4. Gradients Calculation

In our proposed method, we use the parameter-shift rule developed by Gavin E. Crook [33] to calculate gradients in quantum circuits. The parameter-shift rule enables us to compute the gradient of a parameterized quantum circuit, which is essential for optimizing quantum algorithms. The rule allows us to evaluate gradients of quantum circuits without the need for ancilla qubits or controlled operations, making it an efficient and practical approach for quantum computing tasks. Moreover, we can also extend this gradient rule to multi-qubit gates and gate decompositions to facilitate the optimization of quantum circuits, especially in the current NISQ era. The basic setup involves a quantum circuit with a parameterized gate $U_G(\theta)$, as described in Equation (4).

$$U_G(\theta) = e^{-ia\theta G} \tag{4}$$

where G is a Hermitian generator and a is a real constant. The objective function $f(\theta)$ related to this circuit can be written as

$$f(\theta) = \langle \psi | U_G^\dagger(\theta) A U_G(\theta) | \psi \rangle \tag{5}$$

Here, ψ is the state vector and $\mathbf{A}$ is a Hermitian operator. The $f(\theta)$ with respect to the parameter θ using the parameter-shift rule is given by

$$\frac{d}{d\theta} f(\theta) = r\left[f\left(\theta + \frac{\pi}{4r}\right) - f\left(\theta - \frac{\pi}{4r}\right)\right] \tag{6}$$

where the shift parameter r is defined as $r = \frac{a}{2}(e_1 - e_0)$, and e_0 and e_1 are the eigenvalues of the generator G. The parameter-shift rule for implementations of single- and multi-qubit gates is discussed below.

3.4.1. Single-Qubit Gates

Single-qubit gates, such as the Pauli rotation gates, are foundational in quantum computing. These gates can be defined, and their gradients can be computed using the parameter-shift rule:

- The X-rotation gate is defined as

  $$R_x(\theta) = e^{-i\frac{1}{2}\theta X}$$

  with a shift constant $r = \frac{1}{2}$.
- The Y-rotation gate

  $$R_y(\theta) = e^{-i\frac{1}{2}\theta Y}$$

  also has a shift constant $r = \frac{1}{2}$.
- The Z-rotation gate

  $$R_z(\theta) = e^{-i\frac{1}{2}\theta Z}$$

  similarly uses $r = \frac{1}{2}$.

These rotation gates manipulate quantum states by rotating them around the respective axes on the Bloch sphere.

3.4.2. Multi-Qubit Gates

For more complex quantum operations involving multiple qubits, such as the two-qubit canonical gate, the parameter-shift rule can be applied by decomposing the gate into simpler interactions:

- The two-qubit canonical gate can be expressed as

  $$U_{\text{CAN}} = \exp\left(-i\frac{\pi}{2}(t_x(X \otimes X) + t_y(Y \otimes Y) + t_z(Z \otimes Z))\right)$$

  where t_x, t_y, and t_z are coefficients that determine the strength of the interaction terms in a multi-qubit gate for $X \otimes X$, $Y \otimes Y$, and $Z \otimes Z$, respectively. This gate can be decomposed into interactions involving XX, YY, and ZZ.

This decomposition allows the evaluation of gradients for each component of the gate, facilitating the optimization of quantum circuits that incorporate entangling operations.

3.5. Error Handling

Quantum circuits are inherently prone to various types of errors, including decoherence, gate errors, and measurement errors. These errors can significantly impact the performance and reliability of QNNs. However, the H-QNN model addresses these challenges by incorporating several error mitigation techniques.

3.5.1. Transpilation and Gate Optimization

Transpilation is performed using Qiskit's transpile function before executing the quantum circuits. This process effectively optimizes the quantum circuit for the specific backend by proficiently mapping the quantum operations to the available qubits and gates. Transpilation often results in a reduction in the overall gate count and circuit depth, which minimizes the potential for errors stemming from gate imperfections and decoherence.

Given a quantum circuit $\mathbf{T}_{\mathrm{QC}}$ with n qubits, the transpilation process can be mathematically described as

$$\mathbf{T}_{\mathrm{transpiled}} = \mathrm{transpile}(\mathbf{T}_{\mathrm{QC}}, \mathbf{C}, backend) \tag{7}$$

where $\mathbf{T}_{\mathrm{transpiled}}$ is the optimized circuit for the specific quantum hardware and $\mathbf{C}$ is the cost function used to evaluate and optimize the quantum circuit.

3.5.2. Measurement Correction

After running the quantum circuits, measurement outcomes are collected and processed to calculate the expectation values. This step involves converting the raw counts into probabilities and computing the expectation values of the measured states. To handle potential discrepancies in the measurement outcomes, a consistent format for processing the results is used as discussed below.

Let $\{p_i\}$ be the probability of measuring the quantum state $\{|i\rangle\}$. The expectation value E of the quantum measurement is given by

$$E = \sum_i i \cdot p_i \tag{8}$$

where i represents the measured state and p_i is the probability of measuring state i. To calculate p_i, we use the measurement counts from multiple shots:

$$p_i = \frac{n_i}{N} \tag{9}$$

where n_i is the count of measuring state i and N is the total number of shots. In H-QNN, we handle potential discrepancies in the measurement outcomes by processing the results as given in Algorithm 1 (lines 19–34).

This approach ensures accurate interpretation of measurement results, even in the presence of noise and errors, through the practice of averaging over multiple shots and employing statistical methods to extract the expectation values.

3.6. Proposed Model Architecture

Figure 2 depicts the proposed H-QNN architecture. The details of each component are given below:

1. **Input:** The initial step involves image data preprocessing, where all the images are resized to a standard size. For H-QNN architecture, we used the image size of 720 × 720. This resizing ensures uniformity in input dimensions. Furthermore, the pixel values are normalized to fit within a specific range of around $[-1, 1]$.
2. **Classical neural network components:** After preprocessing the data, a CNN begins with six convolutional layers (Conv2d) (Figure 2). These layers process pixel data from images by applying various filters (kernels). The filters capture spatial hierarchies and features such as edges, textures, and shapes. Each convolutional layer is followed by a ReLU activation function. This introduces nonlinearity, which assists the model in learning more complex patterns. After each convolutional layer, max-pooling layers (MaxPool2d) are applied. Pooling reduces the dimensionality of each feature map while retaining the most important information. It helps identify consistent features regardless of variations in the size and orientation of objects in the image. Additionally, it also reduces the dimensions of the feature maps, which in turn lowers

the computational load, making the processing faster and more efficient. After the final convolutional layer, an adaptive average pooling layer (AdaptiveAvgPool2d) is applied. It reduces the spatial dimensions to a size of (1, 1), effectively summarizing the features extracted by the convolutions into a single number per feature map.

3. **Fully connected layers:** The network then transitions from convolutional layers to three fully connected layers (linear). These layers map the learned feature representations to the desired output size. The output of the last convolutional layer sequence is flattened and passed through the fully connected layers to produce a feature vector.
4. **Quantum circuit component/parameterized circuit:** After the fully connected layers are the hidden layers of the H-QNN model that consist of a quantum component only. The input generated by the last classical layer (i.e., fully connected classical layer) is processed through a quantum circuit, which includes Hadamard gates, parameterized rotation gates (R_y), and measurements. The circuit processes inputs through quantum states and produces an output that captures integrated information from the input.

   In our H-QNN model (Figure 2), the parameterized circuit consists of 2 qubits and has 4 quantum gates. To incorporate the quantum circuit into the forward pass of the network, we integrate it with PyTorch's Function class through an *autograd* function. The forward method computes the expectation values of the quantum circuit by running the quantum operations and then measuring the final states to produce a classical output. These expectation values serve as the input to the next layer or the final output of the network.

   During the backward pass, we employ backpropagation to train the network by slightly shifting the inputs using the parameter-shift rule, as described in Section 3.4. We compute the gradients of the quantum circuit with respect to the input parameters, enabling the network to learn and update these parameters. After calculating the gradients, we run the circuit again to verify and adjust the parameters, completing the backward pass.
5. **Intergation of both classical and quantum components:** The Hybrid module combines the quantum circuit with classical neural network functionalities explained in Section 3.3. It is integrated into one of the final layers of the model, specifically after the last fully connected layer of the classical CNN architecture. This final classical layer is responsible for converting the high-level features extracted by the convolutional layers into a format suitable for processing by the quantum circuit.

   The hybrid layer processes these inputs through the quantum circuit, and the resulting output is treated the same way as any other layer output in a neural network. After the quantum circuit processes, the inputs and measurements are taken, and the output from these measurements is then passed on to subsequent layers, if any are present. In this hybrid setup, the measurements from the quantum circuit are essentially the final outputs of the hybrid layer. These outputs can either feed into further layers in a more complex network or directly contribute to the final prediction of the model.
6. **Output:** The output layer of the H-QNN model integrates the outcomes of the quantum circuit with those of the classical neural network to produce the final classification output. The quantum-enhanced features from the parameterized quantum circuit are integrated into the overall decision-making process, which improves the final accuracy and performance of the model.

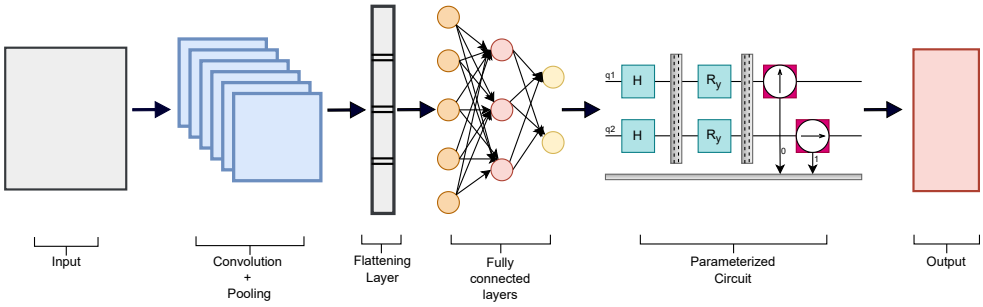


Figure 2. The proposed H-QNN network consists of six convolutional layers, three fully connected layers, and a two-qubit parameterized quantum circuit. In the parameterized circuit, $q1$ and $q2$ denote qubits, the H block signifies data encoding, R_y represents the parameterized gate, and arrows indicate measurements. The square and the rectangle represent the input and output of the network, respectively.

4. Results

This section presents the experimental results of our proposed H-QNN model. We conducted experiments on a computer equipped with an Intel(R) Core i7-13700 processor with processor frequency of 3.50 GHz, 32.0 GB of random access memory (RAM), and running Ubuntu 22.04.4, version LTS.

4.1. Comparative Performance Analysis

We evaluated the effectiveness of our hybrid model by utilizing three distinct datasets sourced from Kaggle. We used 2000 images for each class in the Car vs. Bike dataset, 5000 images for each class in the Dump vs. Recycle Waste dataset, and 1500 images for each class in the Football vs. Rugby dataset. We selected the images randomly from the datasets.

The results of our proposed H-QNN model are presented in Table 1. Our proposed H-QNN model shares the same parameters as the traditional CNN model (six convolutional layers and three fully connected layers). The results indicate that our H-QNN model outperforms the traditional CNN model in terms of accuracy. Specifically, our model achieved an accuracy of 90.1% for the Car vs. Bike dataset, compared to the 88.2% accuracy of the CNN model, despite having identical parameters.

Furthermore, Table 2 compares the performance of H-QNN and CNN on three datasets. The comparison is made for two sets of experiments: one with 1000 images per class and the other with 500 images per class. The purpose of these experiments is to evaluate the models' performance with both ample and limited training data in order to gain insights into their generalization capabilities.

In the experiments with 1000 images per class, H-QNN consistently outperforms CNN across all datasets, demonstrating its robustness with larger training data. Similarly, when the number of images per class is reduced to 500, H-QNN maintains a more stable performance than CNN. This stability indicates H-QNN's superior generalization capabilities, making it more suitable for applications with limited training data. In contrast, CNN tends to overfit with reduced data, underscoring the advantages of utilizing H-QNN for reliable and consistent performance in image classification tasks.

Further, we evaluate the performance of our H-QNN model and classical CNN using the confidence interval metric (with 95% confidence) over Car vs. Bike, Football vs. Rugby, and Dump vs. Waste datasets. It is worth mentioning here that we conducted experiments on two different variants of each dataset, where the first variant contains 1000 images per class and the second variant contains 500 images per class. The obtained results are tabulated in Table 3. As can be noticed in Table 3, our H-QNN achieves better confidence intervals in comparison with the classical CNN in most of the cases. For example, our H-QNN has a narrow confidence interval on Football vs. Rugby dataset when using 1000 images per class. Similarly, when testing on 500 images per class, our H-QNN yields a better confidence interval on the Car vs. Bike, Football vs. Rugby, and Dump vs. Waste

datasets. The obtained confidence interval values indicate that our model prediction confidence lies in a narrow yet higher confidence interval across each variant of the dataset, as shown in Table 3.

Table 1. Accuracy analysis of our proposed H-QNN with CNN model for three datasets: Car vs. Bike, Football vs. Rugby, and Dump vs. Recycle.

Datasets	Test Accuracy	
	H-QNN	CNN
Car vs. Bike	90.1%	88.2%
Football vs. Rugby	72.0%	70.4%
Dump vs. Recycle Waste	86.7%	84.9%

Table 2. Performance analysis of our proposed H-QNN with CNN model with different numbers of images for three datasets: Car vs. Bike, Football vs. Rugby, and Dump vs. Recycle.

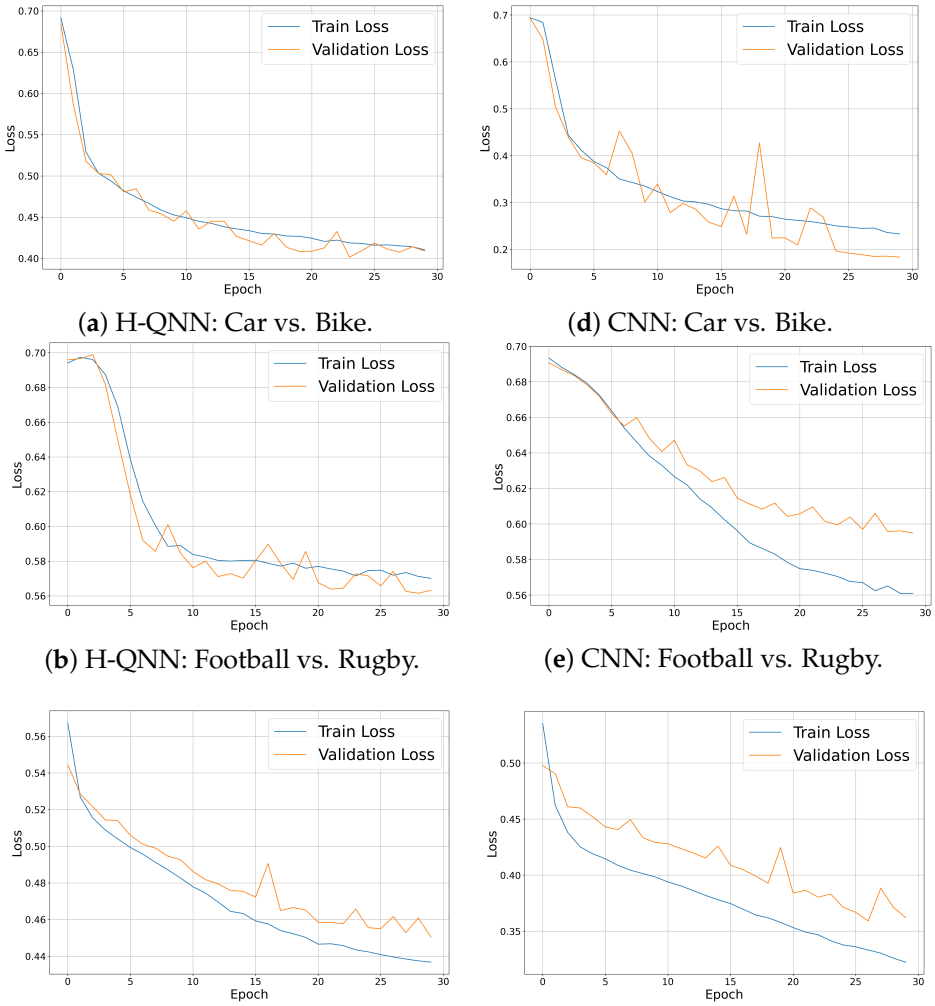
Accuracy	1000 Images per Class						500 Images per Class					
	Car vs. Bike		Football vs. Rugby		Dump vs. Recycle		Car vs. Bike		Football vs. Rugby		Dump vs. Recycle	
	H-QNN	CNN	H-QNN	CNN	H-QNN	CNN	H-QNN	CNN	H-QNN	CNN	H-QNN	CNN
Train	88.93	87.36	72.36	72.02	89.07	88.43	85.14	88.71	76.29	72.29	90.43	89.43
Validation	87.00	85.67	73.33	72.67	89.33	87.67	86.67	83.33	79.33	77.33	88.00	85.33
Test	87.71	86.33	72.00	68.00	91.00	88.33	83.13	79.32	75.40	71.30	92.71	82.00

Table 3. Computed confidence interval values (with 95% confidence) for our proposed H-QNN and classical CNN over Car vs. Bike, Football vs. Rugby, and Dump vs. Waste datasets.

Car vs. Bike		Football vs. Rugby		Dump vs. Waste	
H-QNN	CNN	H-QNN	CNN	H-QNN	CNN
1000 Images					
[85.45–90.30]	[84.33–88.56]	[70.85–74.27]	[64.61–77.18]	[87.19–92.40]	[87.11–89.16]
500 Images					
[80.56–89.39]	[70.54–96.36]	[71.88 – 82.12]	[65.60–81.67]	[84.52–96.23]	[76.34–94.83]

4.2. Loss Curve Analysis

Figure 3 shows the loss curves of the H-QNN and CNN models across three different datasets over the training epochs. These curves are essential to understand the learning and stabilization rate of each model. The H-QNN model has a steep decline in loss, indicating a faster learning rate and potentially better generalization capabilities. The curves also show less fluctuation, indicating that the quantum components help stabilize the learning dynamics. On the other hand, the CNN models have a more gradual decline in loss with noticeable fluctuations, especially during the early stages of training, which may indicate difficulty in fitting the data or getting stuck in local minima. Based on the comparative analysis of these loss curves, it can be concluded that the H-QNN model learns faster, achieves a lower loss, and thus performs better on these datasets.



(**a**) H-QNN: Car vs. Bike.

(**d**) CNN: Car vs. Bike.

(**b**) H-QNN: Football vs. Rugby.

(**e**) CNN: Football vs. Rugby.

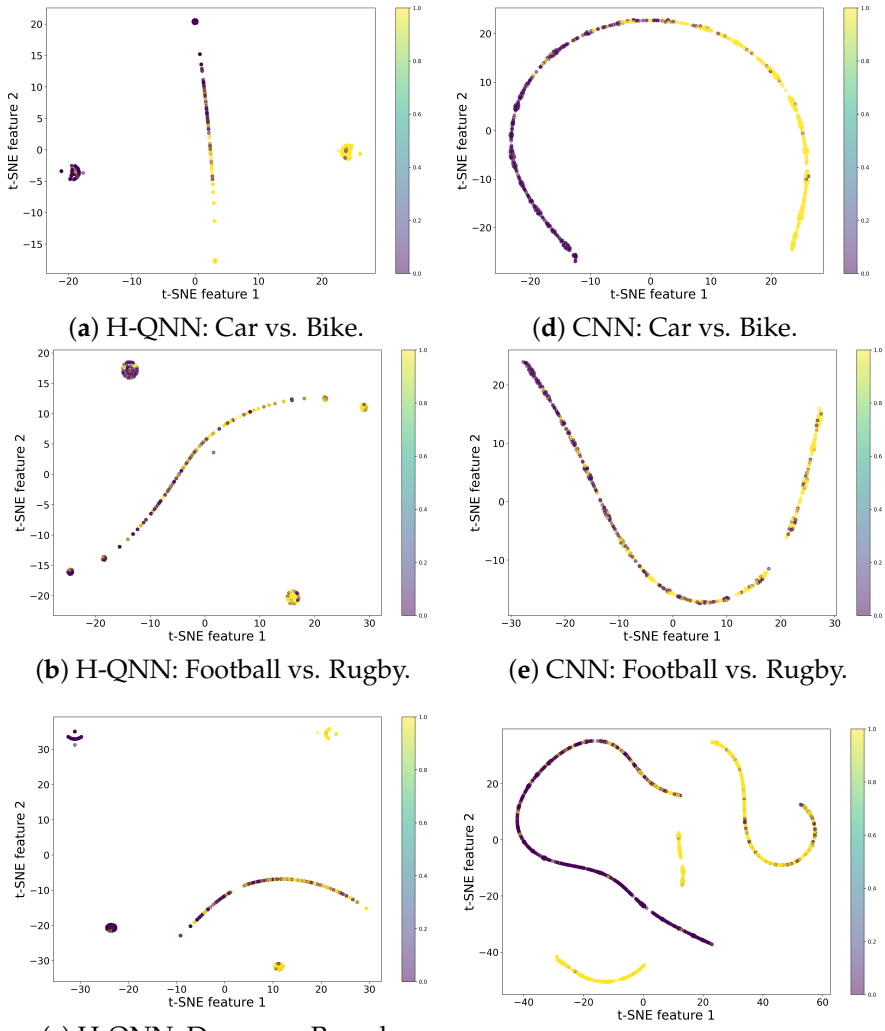
(**c**) H-QNN: Dump vs. Recycle Waste.

(**f**) CNN: Dump vs. Recycle Waste.

Figure 3. Loss curves for three datasets: Car vs. Bike, Football vs. Rugby, and Dump vs. Recycle. Results for our proposed H-QNN model are shown in (**a**–**c**), while results for the CNN model are shown in (**d**–**f**).

4.3. t-SNE Feature Extraction

Figure 4 illustrates the t-distributed stochastic neighbor embedding (t-SNE) technique for feature extraction, which is used for dimensionality reduction to visualize the high-dimensional features learned by the H-QNN and CNN models. This enables us to evaluate the quality of clustering and separation of features between different classes. The feature plots for the H-QNN model exhibit well-defined and distinct clusters, indicating its effectiveness in learning discriminative features that are well separated in the feature space. Conversely, the CNN feature plots show less distinct clusters with some overlap between different classes, which could lead to higher misclassification rates as the model may struggle to distinguish between closely clustered classes. These t-SNE plots show that the H-QNN model performs better than CNN because it can learn more useful and separate features for image classification tasks.



(**a**) H-QNN: Car vs. Bike.

(**d**) CNN: Car vs. Bike.

(**b**) H-QNN: Football vs. Rugby.

(**e**) CNN: Football vs. Rugby.

(**c**) H-QNN: Dump vs. Recycle Waste.

(**f**) CNN: Dump vs. Recycle Waste.

Figure 4. t-SNE feature extraction for three datasets: Car vs. Bike, Football vs. Rugby, and Dump vs. Recycle. Results for our proposed H-QNN model and CNN model are shown in (**a**–**c**) and (**d**–**f**), respectively.

4.4. Image Retrieval Performance Analysis

This section presents the detailed quantitative and qualitative evaluation of the baseline CNN and our proposed H-QNN for the image retrieval task on three different datasets (including Car vs. Bike, Football vs. Rugby, and Dump vs. Recycle dataset). The motivation behind the evaluation of trained models (including both CNN and H-QNN) for image retrieval tasks is to ensure their efficiency in terms of feature extraction and adaptability for downstream tasks (i.e., image retrieval, object detection, and image segmentation). To conduct image retrieval experiments, we collected 100 images per class from three datasets, resulting in a test set containing 600 images. Typically, for an image retrieval task, the performance of a model can be examined based on correct retrievals for a given query image. Following the same evaluation criteria, our proposed evaluation scheme for the image retrieval task uses the top-k metric, where k $\in$ {5, 10, and 15}. The variable k represents the number of retrievals based on their similarity with the query image. The final retrieval accuracy is computed as follows in Equation (10):

$$Top\text{-}k\ accuracy = \frac{Number\ of\ correct\ retrivals}{Number\ of\ total\ retrivals} \tag{10}$$

The obtained quantitative retrieval results of baseline CNN and our H-QNN on three datasets are presented in Table 4. The tabulated values demonstrate the effectiveness of our proposed H-QNN model, which obtains an average improvement of 12.41% over baseline CNN on the Car vs. Bike dataset. On the Football vs. Rugby and Dump vs. Recycle datasets, the baseline CNN obtains better results as compared to H-QNN across each top-k metric. Furthermore, the obtained qualitative results from three tests with different query images using top-10 metrics are depicted in Figure 5. From the results listed in Table 4, it can be perceived that the results obtained on the image retrieval task are different from the classification results due to their distinct objectives and evaluation criteria. In image classification, a model aims to assign the correct label to each image from a predefined set of categories, while in image retrieval, it focuses on finding and ranking images based on their similarity to a query image.

Table 4. Image retrieval performance analysis on three datasets including Car vs. Bike, Football vs. Rugby, and Dump vs. Recycle. Each model is evaluated based on three different metrics that include top-5, top-10, and top-15.

Datasets	Models	Top-5	Top-10	Top-15	Avg
Car vs. Bike	CNN	60	56.67	50.05	55.57
	H-QNN	76.67	66.67	60.61	67.98
Football vs. Rugby	CNN	63.33	46.67	42.52	51.84
	H-QNN	50	45	46.97	47.32
Dump vs. Recycle	CNN	73.33	61.67	62.52	65.84
	H-QNN	63.33	60	50.45	57.92

Figure 5. Visual illustration of image retrieval tests using Top-10 retrieval metric. Three different query images are used to retrieve Top-10 similar images from the test set of 600 images.

5. Discussion and Limitations

The results presented in Section 4 demonstrate the potential of H-QNN for binary image classification. The H-QNN model effectively combines the advantages of both quantum computing and classical neural networks, resulting in improved accuracy and

generalization capabilities compared to traditional CNN models. Specifically, the H-QNN model achieved a 90.1% accuracy rate on binary image datasets, outperforming CNN models. The integration of a two-qubit quantum circuit within the classical convolutional architecture has shown to be computationally efficient on NISQ devices, making it a promising approach for practical quantum computing applications. Furthermore, the experimental results indicate that the H-QNN model can address the issue of overfitting with small datasets, which is a common challenge in machine learning. The loss curve analysis and t-SNE feature extraction demonstrate that the H-QNN model learns faster, achieves lower loss, and produces well-separated feature clusters, contributing to its superior performance in image classification and image retrieval tasks.

Despite the advantages offered by the H-QNN model, there are some limitations to consider. The current implementation of the H-QNN model is limited by the capabilities of NISQ devices, which have a restricted number of qubits and are prone to noise and errors. Gate fidelity and quantum noise are critical factors affecting the performance of quantum circuits. Low gate fidelities can introduce errors in the computation, while quantum noise, including decoherence and operational errors, can degrade the model's accuracy. Our experimental observations reveal that gate errors and noise can lead to significant performance degradation.

To mitigate the limitations of NISQ devices, we employed several strategies, including error mitigation techniques such as measurement correction and transpilation. These techniques help to counteract the adverse effects of noise, allowing our H-QNN model to achieve more reliable results. Continued advancements in quantum hardware, particularly in enhancing gate fidelities and reducing noise, will be crucial for the practical deployment of quantum neural networks. While the H-QNN model shows significant improvement in binary image classification, its scalability to multi-class classification and larger datasets remains to be fully explored. The efficiency and accuracy of the H-QNN model in handling more complex tasks with a higher number of classes and larger data volumes need further investigation.

The quantum circuit design within the H-QNN model is relatively simple, involving only a two-qubit circuit. More sophisticated quantum circuit designs could enhance the model's performance but require more advanced quantum hardware and optimization techniques. Training the H-QNN model involves significant computational resources and time, particularly when simulating quantum circuits on classical hardware. This can be a barrier to QML widespread adoption, as it requires access to high-performance computing resources and efficient quantum simulators.

The current study focuses on binary image classification tasks. The generalization of the H-QNN model to other domains and types of data, such as natural language processing or time-series analysis, has yet to be explored and remains an open question. Quantum computations are susceptible to errors due to decoherence and gate imperfections. Effective error mitigation strategies are crucial for improving the reliability and accuracy of the H-QNN model. The quantum circuits' error mitigation strategies are still an area of ongoing research. Addressing these limitations will be essential for advancing the practical applications of H-QNN models in various domains and fully realizing the potential of quantum-enhanced machine learning. Future work in this area should focus on developing more robust quantum circuits, improving error mitigation techniques, and exploring the scalability and generalization of H-QNN models to broader applications.

6. Conclusions and Future Work

Quantum neural networks (QNNs) hold significant promise for advancing image classification tasks by leveraging the unique properties of quantum mechanics. This study illustrates the potential of hybrid QNNs (H-QNNs) in advancing binary image classification. Our proposed H-QNN model, specifically tailored for binary image classification tasks, seamlessly integrates the computational advantages of quantum mechanics with traditional machine learning techniques. The proposed model significantly improves classification

accuracy on NISQ devices, achieving an impressive 90.1% accuracy rate on the Car vs. Bike dataset, surpassing the performance of the classical CNN model by 1.9%.

Our proposed H-QNN model comprises a two-qubit, six-layer architecture designed to address the qubit limitations of NISQ devices, along with a novel quantum convolutional architecture that achieves high accuracy with a smaller number of images. The hybrid quantum–classical approach optimizes computational resources across the quantum–classical spectrum, offering a scalable and effective solution in the NISQ era. The adaptability of our proposed architecture to handle multi-class classification further underscores its potential in various real-world applications.

This work also underscores the robustness of our H-QNN model to quantum noise and its ability to generalize well even with limited training data, in contrast to the traditional CNN's tendency to overfit. These findings suggest that hybrid models represent a paradigm shift in quantum-enhanced artificial intelligence, unlocking further possibilities for addressing the limitations of purely classical or quantum approaches in high-stakes, real-world applications.

Despite the promising results for H-QNN models, there are several areas for future work that could improve the applicability and performance of H-QNN models. Future research could focus on developing more sophisticated quantum circuit designs that take advantage of advanced quantum hardware and optimization techniques to further enhance the model's performance. Additionally, extending the H-QNN model to handle multi-class classification and larger datasets could significantly broaden its applicability.

Exploring the generalization of the H-QNN model to other domains, such as natural language processing and time-series analysis, could provide insights into its versatility. Furthermore, advancing error mitigation strategies for quantum circuits will be crucial for improving the reliability and accuracy of H-QNN models. Finally, investigating the scalability of H-QNN models to larger quantum systems and integrating them with emerging quantum technologies (e.g., quantum annealing and topological qubits) could lead to more practical and widespread applications in quantum-enhanced machine learning.

Author Contributions: Conceptualization, M.A.H. and A.M.; methodology, M.A.H.; software, M.A.H.; validation, M.A.H. and H.U.; formal analysis, M.A.H. and H.U.; investigation, A.M.; resources, A.M.; data curation, M.A.H.; writing—original draft preparation, M.A.H.; writing—review and editing, A.M.; visualization, M.A.H. and H.U.; supervision, A.M.; project administration, A.M.; funding acquisition, A.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Datasets used in this research are available open source at the following links: Car vs. Bike dataset at https://www.kaggle.com/datasets/utkarshsaxenadn/car-vs-bike-classification-dataset (accessed on 26 February 2024); Football vs. Rugby dataset at https://www.kaggle.com/datasets/ligtfeather/football-vs-rugby-image-classification (accessed on 28 March 2024); Dump vs. Recycle dataset at https://www.kaggle.com/datasets/techsash/waste-classification-data (accessed on 28 March 2024).

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

References

1. Lu, D.; Weng, Q. A survey of image classification methods and techniques for improving classification performance. *Int. J. Remote Sens.* **2007**, *28*, 823–870. [CrossRef]
2. Dureja, A.; Pahwa, P. Medical image retrieval for detecting pneumonia using binary classification with deep convolutional neural networks. *J. Inf. Optim. Sci.* **2020**, *41*, 1419–1431. [CrossRef]
3. Sun, S.; Yin, Y.; Wang, X.; Xu, D.; Wu, W.; Gu, Q. Fast object detection based on binary deep convolution neural networks. *CAAI Trans. Intell. Technol.* **2018**, *3*, 191–197. [CrossRef]

4. Chaganti, S.Y.; Nanda, I.; Pandi, K.R.; Prudhvith, T.G.; Kumar, N. Image Classification using SVM and CNN. In Proceedings of the 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA), Gunupur, India, 13–14 March 2020 ; pp. 1–5.
5. Dhruv, P.; Naskar, S. Image classification using convolutional neural network (CNN) and recurrent neural network (RNN): A review. In *Machine Learning and Information Processing: Proceedings of ICMLIP 2019*; Springer: Singapore, 2020; pp. 367–381.
6. Joshi, S.; Verma, D.K.; Saxena, G.; Paraye, A. Issues in training a convolutional neural network model for image classification. In Proceedings of the Advances in Computing and Data Sciences: Third International Conference, ICACDS 2019, Ghaziabad, India, 12–13 April 2019; Revised Selected Papers, Part II 3; Springer: Singapore, 2019; pp. 282–293.
7. Yuan, L. Remote sensing image classification methods based on CNN: Challenge and trends. In Proceedings of the 2021 International Conference on Signal Processing and Machine Learning (CONF-SPML), Stanford, CA, USA, 14 November 2021; pp. 213–218.
8. Ramezani, S.B.; Sommers, A.; Manchukonda, H.K.; Rahimi, S.; Amirlatifi, A. Machine learning algorithms in quantum computing: A survey. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8.
9. Martín-Guerrero, J.D.; Lamata, L. Quantum machine learning: A tutorial. *Neurocomputing* **2022**, *470*, 457–461. [CrossRef]
10. Brooks, M. Beyond quantum supremacy: The hunt for useful quantum computers. *Nature* **2019**, *574*, 19–22. [CrossRef] [PubMed]
11. Preskill, J. Quantum computing in the NISQ era and beyond. *Quantum* **2018**, *2*, 79. [CrossRef]
12. Guo, T.; Dong, J.; Li, H.; Gao, Y. Simple convolutional neural network on image classification. In Proceedings of the 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA), Beijing, China, 10–12 March 2017; pp. 721–724.
13. Li, Q.; Cai, W.; Wang, X.; Zhou, Y.; Feng, D.D.; Chen, M. Medical image classification with convolutional neural network. In Proceedings of the 2014 13th International Conference on Control Automation Robotics & Vision (ICARCV), Singapore, 10–12 December 2014; pp. 844–848.
14. Hussain, M.; Bird, J.J.; Faria, D.R. A study on CNN transfer learning for image classification. In Proceedings of the Advances in Computational Intelligence Systems: Contributions Presented at the 18th UK Workshop on Computational Intelligence, Nottingham, UK, 5–7 September 2018; pp. 191–202.
15. Elngar, A.A.; Arafa, M.; Fathy, A.; Moustafa, B.; Mahmoud, O.; Shaban, M.; Fawzy, N. Image classification based on CNN: A survey. *J. Cybersecur. Inf. Manag.* **2021**, *6*, 18–50. [CrossRef]
16. Liu, L.; Ouyang, W.; Wang, X.; Fieguth, P.; Chen, J.; Liu, X.; Pietikäinen, M. Deep learning for generic object detection: A survey. *Int. J. Comput. Vis.* **2020**, *128*, 261–318. [CrossRef]
17. Sultana, F.; Sufian, A.; Dutta, P. Evolution of image segmentation using deep convolutional neural network: A survey. *Knowl.-Based Syst.* **2020**, *201*, 106062. [CrossRef]
18. Schuld, M.; Sweke, R.; Meyer, J.J. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Phys. Rev. A* **2021**, *103*, 032430. [CrossRef]
19. Abbas, A.; Sutter, D.; Zoufal, C.; Lucchi, A.; Figalli, A.; Woerner, S. The power of quantum neural networks. *Nat. Comput. Sci.* **2021**, *1*, 403–409. [CrossRef] [PubMed]
20. Schuld, M.; Bocharov, A.; Svore, K.M.; Wiebe, N. Circuit-centric quantum classifiers. *Phys. Rev. A* **2020**, *101*, 032308. [CrossRef]
21. Lloyd, S.; Schuld, M.; Ijaz, A.; Izaac, J.; Killoran, N. Quantum embeddings for machine learning. *arXiv* **2020**, arXiv:2001.03622.
22. Sim, S.; Johnson, P.D.; Aspuru-Guzik, A. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Adv. Quantum Technol.* **2019**, *2*, 1900070. [CrossRef]
23. Shephard, N.; Xiu, D. Econometric analysis of multivariate realised QML: Estimation of the covariation of equity prices under asynchronous trading. *J. Econom.* **2017**, *201*, 19–42. [CrossRef]
24. Ayoade, O.; Rivas, P.; Orduz, J.; Rafi, N. Satellite image classification using quantum machine learning. In *Artificial Intelligence in Earth Science*; Elsevier: Amsterdam, The Netherlands, 2023; pp. 337–355.
25. Oroy, K.; Jhon, R. Quantum Machine Learning: Bridging Quantum Computing and Artificial Intelligence. *EasyChair* **2024** . Available online: https://easychair.org/publications/preprint/fpn2 (accessed on 1 July 2024).
26. Choi, J.; Chuang, P.I.; Wang, Z.; Venkataramani, S.; Srinivasan, V.; Gopalakrishnan, K. Bridging the accuracy gap for 2-bit quantized neural networks (qnn). *arXiv* **2018**, arXiv:1807.06964.
27. Sher, A.; Trusov, A.; Limonova, E.; Nikolaev, D.; Arlazarov, V.V. Neuron-by-Neuron Quantization for Efficient Low-Bit QNN Training. *Mathematics* **2023**, *11*, 2112. [CrossRef]
28. Mani, V.R.S.; Saravanaselvan, A.; Arumugam, N.J.M.J. Performance comparison of CNN, QNN and BNN deep neural networks for real-time object detection using ZYNQ FPGA node. *Microelectron. J.* **2022**, *119*, 105319. [CrossRef]
29. Alam, M.; Kundu, S.; Topaloglu, R.O.; Ghosh, S. Quantum-classical hybrid machine learning for image classification (iccad special session paper). In Proceedings of the 2021 IEEE/ACM International Conference on Computer Aided Design (ICCAD), Munich, Germany, 1–4 November 2021; pp. 1–7.
30. Mahajan, R.P. Hybrid quantum inspired neural model for commodity price prediction. In Proceedings of the 13th International Conference on Advanced Communication Technology (ICACT2011), Gangwon, Republic of Korea, 13–16 February 2011; pp. 1353–1357.
31. Chen, S.Y.; Wei, T.C.; Zhang, C.; Yu, H.; Yoo, S. Hybrid quantum-classical graph convolutional network. *arXiv* **2021**, arXiv:2101.06189.

32. Huang, S.-Y.; An, W.-J.; Zhang, D.-S.; Zhou, N.-R. Image classification and adversarial robustness analysis based on hybrid quantum-classical convolutional neural network. *Opt. Commun.* **2023**, *533*, 129287. [CrossRef]
33. Crooks, G.E. Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition. *arXiv* **2019**, arXiv:1905.13311.