Ingénierie des Systèmes d'Information
Vol. 31, No. 8, August, 2026, pp. 2753-2764
Journal homepage: http://iieta.org/journals/isi

Improvement of Quantum Feature Extraction Using Quantum Attention for Efficient Image Classification

Ifran Lindu Mahargya[1], Guruh Fajar Shidik[2*], Affandy[1], Pujiono[1]

[1] Faculty of Computer Science, Universitas Dian Nuswantoro, Semarang 50131, Indonesia
[2] Research Group Intelligence Distributed Surveillance and Security, Universitas Dian Nuswantoro, Semarang 50131, Indonesia

Corresponding Author Email: guruh.fajar@research.dinus.ac.id

https://doi.org/10.18280/isi.310820

Received: 25 April 2026
Revised: 28 July 2026
Accepted: 15 August 2026
Available online: 31 August 2026

Keywords:
image classification, quantum feature extraction, QFE-QA, ancilla-based quantum attention, quantum machine learning, parameterized quantum circuit

ABSTRACT

Quantum Feature Extraction (QFE) uses trainable quantum circuits to process image patches but lacks an explicit mechanism for emphasizing patches that are more informative for classification. This study proposed Quantum Feature Extraction-Quantum Attention (QFE-QA), which added a phase-sensitive ancilla-based quantum attention branch to QFE. For each patch, nine shared trainable controlled-RZ angles and one ancilla (auxiliary) qubit produced a relevance score. Softmax converted the patch scores into normalized weights, and the weighted QFE1 features were combined with the QFE2 output without Query-Key-Value representations or pairwise similarity matrices. QFE-QA was evaluated on balanced ten-class subsets of MNIST and Fashion-MNIST using a noiseless statevector simulator. Across five runs, it achieved mean test accuracies of 93.13% and 81.88%, compared with 92.13% and 80.73% for QFE, respectively; neither improvement was statistically significant. QFE-QA contained 27,817 trainable parameters, including nine attention parameters. The reported fully connected-only runtime excluded quantum-circuit execution and therefore did not establish end-to-end efficiency or speedup. These results support the feasibility of QFE-QA in the evaluated simulator setting, while validation on larger datasets, noisy circuits, and quantum hardware remains necessary.

1. INTRODUCTION

Convolutional Neural Networks (CNNs) learn local spatial patterns by applying shared filters across an image. Deeper CNNs can learn more complex features. However, they usually require more computation, parameters, and training time [1, 2]. Because the quality of feature extraction strongly affects computer-vision performance [3-11], hybrid quantum-classical alternatives have been investigated for classification, recognition, segmentation, and object tracking [12-15].

Quantum Feature Extraction (QFE) replaces a classical convolutional layer with a parameterized quantum circuit (PQC) whose trainable parameters are shared across image patches [16]. The original QFE design used a Quantum Approximate Optimization Algorithm (QAOA)-inspired circuit and described training through the Parameter-Shift Rule (PSR) and Chain Rule (CR) [17-19]. After each patch is processed, QFE projects and pools the resulting observables to obtain a compact feature vector. However, this aggregation does not explicitly assign larger input-dependent weights to the patches that are more informative for a particular image. Repeated experiments are also needed because early QFE validation used relatively simple settings [20].

Attention provides a way to assign different weights to different feature locations, so the model can emphasize information that is more useful for a given input [21-24]. Existing quantum-attention studies have implemented this idea through Query-Key-Value (QKV) representations, pairwise similarity scores, attention-matrix encoding, qubit-measurement weighting, quantum convolution, or variational quantum circuits [25-32]. These methods demonstrate several ways to combine quantum circuits and attention, but many of them require pairwise comparisons, classical convolutional backbones, or image-wide encoding. Their structures therefore cannot be inserted directly into the patch-wise QFE pipeline without substantial modification.

To address this gap, this study proposed Quantum Feature Extraction-Quantum Attention (QFE-QA). The method added an ancilla-based branch that converted each QFE1 patch-observable vector into one scalar relevance score. Nine controlled-RZ (CRZ) angles were shared across all patches, and one ancilla qubit provided the score used by softmax. This design avoided a full QKV module and was evaluated on controlled subsets containing all ten classes of MNIST and Fashion-MNIST (FMNIST).

In this study, efficiency was assessed only through the number of trainable parameters. The reported fully connected (FC)-only time measured the classifier head. It excluded execution of the QFE and QFE-QA circuits. It therefore cannot be interpreted as end-to-end speedup, computational efficiency, or quantum advantage.

This study makes two main contributions. First, it

introduces a QFE-specific, phase-sensitive ancilla-based attention branch. This branch maps each QFE1 patch-observable vector to a scalar score using nine shared CRZ angles and one ancilla qubit. Second, it integrates the Softmax-weighted QFE1 representation with QFE2 while preserving an 18-dimensional classifier input and avoiding QKV representations and pairwise patch-similarity matrices.

The remainder of this paper is organized into four sections. Section 2 reviews related work, and Section 3 explains the proposed method. Section 4 presents the experimental results and discussion, while Section 5 summarizes the conclusions and future work.

2. RELATED WORK

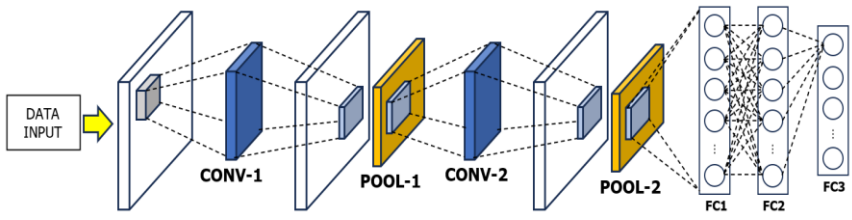
Table 1 compares the reviewed methods by architecture, data encoding, task, and reported resource configuration. It is intended to show how the methods organize attention rather than to rank their predictive performance. A direct ranking would be misleading because the studies use different datasets, class counts, circuit models, and definitions of quantum resources.

Methods [25, 26, 28, 29] use Q/K/V representations or pairwise similarity scores, whereas [27] encodes an attention matrix in a one-step or deferred-measurement circuit. Studies [30-32] use qubit-measurement weighting, quantum attention within a quantum-convolutional CNN, or a VQC-based replacement for global pooling. In contrast, QFE-QA begins with the patch-level observables produced by QFE1. Nine shared CRZ angles and one ancilla qubit produce one score per patch, and softmax converts the scores into normalized weights. QFE-QA therefore requires neither QKV representations nor pairwise patch-similarity matrices.

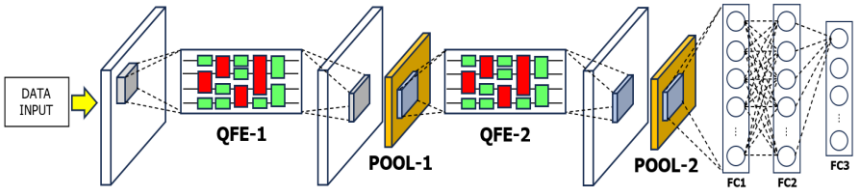
Table 1. Analytical comparison of representative quantum-attention and quantum-enhanced feature-aggregation methods with Quantum Feature Extraction-Quantum Attention (QFE-QA)

Methods	Attention / Circuit Structure and Analytical Relation	Datasets	Qubits / Resource ∗	Classification Task
QMSAN [25]	Mixed-state Q/K similarity using trace/SWAP-test formulation; retains pairwise Q–K scoring	MC, RP, Yelp, IMDb, Amazon	2–4	Binary text classification
QSANN [26]	Encoder and separate QKV PQCs with Gaussian-projected pairwise coefficients	MC, RP, Yelp, IMDb, Amazon	2–4	Binary text classification
QSAN [27]	QLS and QBSASM in a one-step/deferred-measurement circuit; produces an attention-matrix representation	MNIST	8 †	Binary image classification
QSAM [28]	Fully quantum ($U_e/U_1/U_2/U_3$) circuits with grouped QKV processing	Iris, MC, RP	32–96 †	Binary Iris/text classification
QKSAN [29]	Q/K quantum-kernel scores with conditional/deferred measurement; pairwise kernel attention	MNIST, FMNIST	4 †	Binary image classification
MQCNN-QA [30]	Multi-topology PQC quantum kernels; measurements from multiple qubits generate local feature maps that are adaptively weighted and fused through qubit-measurement attention	CIFAR-10	4 ‡	10-class image classification
QAttn-CNN [31]	NEQR-based quantum image representation, quantum convolution, and expectation-value-based quantum attention for adaptive feature selection within a CNN pipeline	MNIST, CIFAR-10, Skin Cancer, HAM10000	18 §	Binary, 7-class, and 10-class image classification
AE-VQC [32]	Amplitude-encoded variational quantum circuit replaces global pooling to preserve fine-grained feature-map information before the fully connected classifier	Croatian Fish, Aircraft, BUSI, Apples-or-Tomatoes	15–17 ‖	Binary and multiclass image classification
QFE-QA (Proposed)	**QFE1 observables → nine shared CRZ gates on one ancilla → phase-sensitive (Z) score → patch softmax; no QKV or pairwise similarity**	**MNIST, FMNIST**	**10 ϱ**	**10-class image classification**

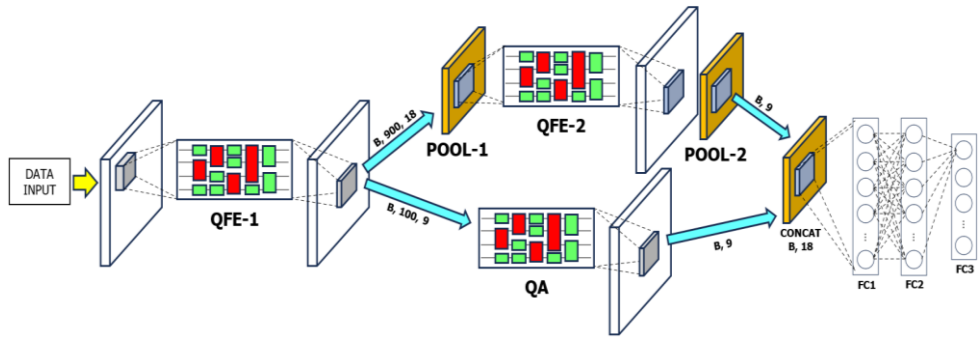
Notes:∗ Resource values follow the definitions and experimental configurations reported in the source studies and should not be interpreted as directly equivalent hardware costs. † QSAN is shown using its reported comparative configuration; QSAM reports task- and variant-dependent total circuit widths. ‡ MQCNN-QA uses four qubits per 2 × 2 q kernel; its circuit study evaluates 15 PQCs across five common topologies. § QAttn-CNN reports an 18-qubit NEQR register for a 32 × 32 grayscale image. ‖ AE-VQC requires 15–17 qubits depending on the backbone feature-map dimensions. ϱ Maximum QFE-QA attention-circuit width: nine feature qubits and one ancilla. PQC = parameterized quantum circuit, QKV = Query-Key-Value.



(a)

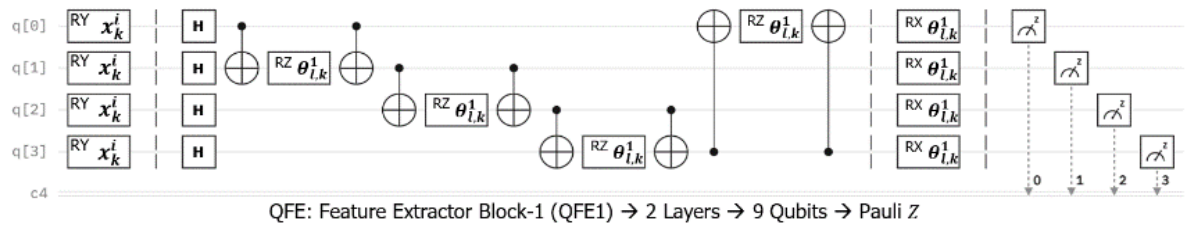


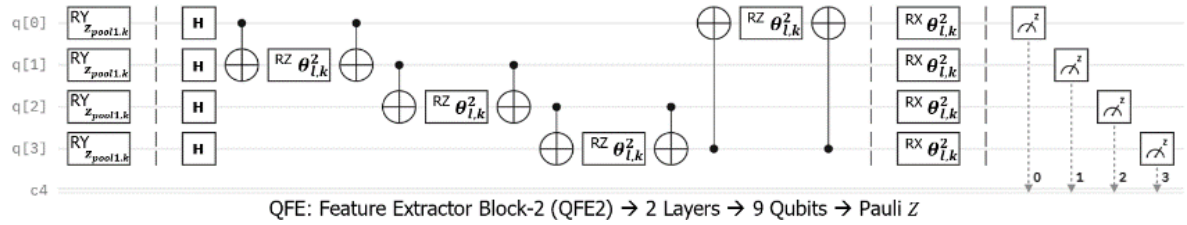
(b)



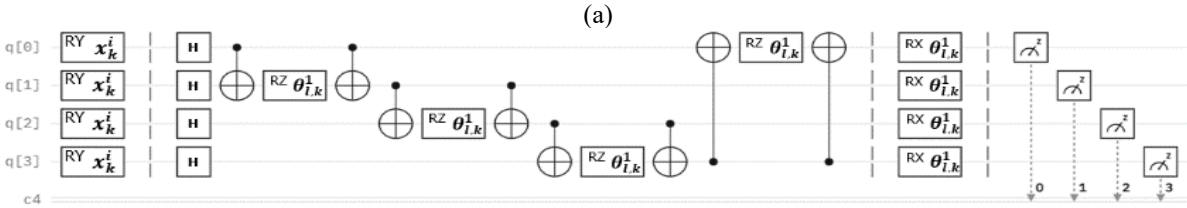
(c)

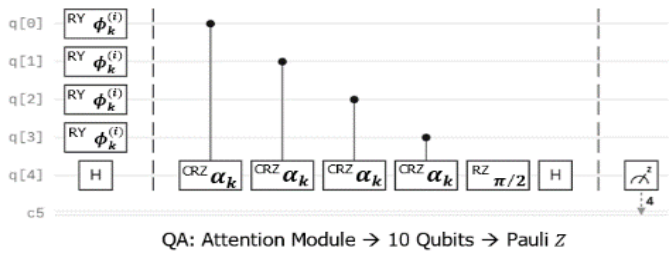
Figure 1. Model architecture: (a) Convolutional Neural Network (CNN) architecture, (b) Quantum Feature Extraction (QFE) architecture, (c) Quantum Feature Extraction-Quantum Attention (QFE-QA) architecture (proposed method)

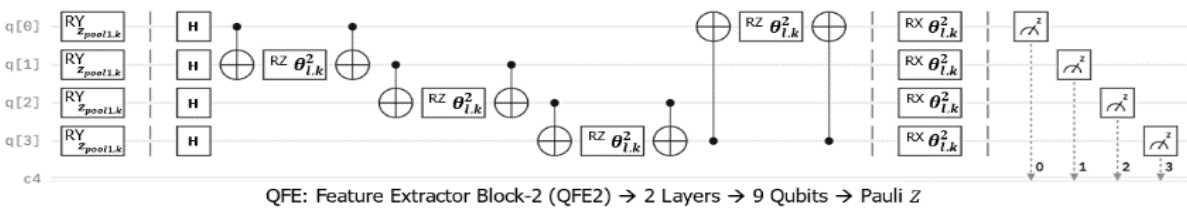




(a)







(b)

Figure 2. Illustrative quantum-circuit configurations, shown with four feature qubits for readability: (a) Quantum Feature Extraction (QFE) and (b) Quantum Feature Extraction-Quantum Attention (QFE-QA)

Note: Four feature qubits are shown only for readability. The implemented QFE blocks used nine feature qubits, whereas the QA circuit used nine feature qubits and one ancilla, giving a maximum width of ten qubits. Thus, $q[0]-q[3]$ and $q[4]$ in Figure 2(b) represent the implemented feature wires $q[0]-q[8]$ and ancilla $q[9]$, respectively. Each QFE1 observable was re-encoded as $\phi_k^{(i)} = \pi\,(z_k^{(i)}+1)/2$, and the feature qubits controlled nine trainable CRZ rotations on the ancilla. The QFE angles θ and CRZ angles α were trainable, whereas the encoding rotations and the $R_Z(\pi/2)$ ancilla phase were fixed. The final Hadamard enabled phase-sensitive Pauli Z readout.

2.1 Convolutional Neural Network

A CNN extracts local patterns through convolution and usually reduces their dimensions through pooling before classification. Deeper convolutional stacks can improve representation capacity but also increase computational cost [1, 2]. Lightweight CNNs address this cost through compact designs [3], but such models were outside this study's scope. The two-layer CNN from [16] was reproduced only as a classical reference under the original QFE setting, not as the current state of lightweight CNN research.

Eq. (1) defines convolution. Here, X is the input; W_1 and B_1 are the weights and bias; $*$ denotes convolution; and $g(\cdot)$ is the activation function. The filter size and input-channel count determine the shape of W_1.

$$Y_1(X) = g(W_1 * X + B_1) \tag{1}$$

Eq. (2) defines the subsequent learned transformation. Here, x is the input vector; W_2 and B_2 are the weight and bias; and $h(\cdot)$ is the activation function [2, 16]. Figure 1(a) shows these operations in the CNN reference.

$$Y_2(x) = h(W_2 x + B_2) \tag{2}$$

2.2 Quantum Feature Extraction

Dou et al. [16] introduced QFE as a hybrid architecture that replaced a convolutional feature extractor with a parameterized quantum circuit. The circuit processed small image patches, while classical pooling and fully connected classification were retained. Its trainable circuit parameters were optimized with gradient-based training, for which the original formulation used the Parameter-Shift Rule and Chain Rule. Figures 1(b) and 2(a) show the QFE architecture and an illustrative circuit, respectively.

Conceptually, the QFE circuit acts as a trainable patch transformation. A flattened 3 × 3 patch supplies nine values, each of which controls a rotation on one qubit. Measuring the transformed qubits produces a numerical feature vector for the subsequent classical layers.

Eqs. (3) and (4) define the angle-encoding stage. The flattened patch is written as $x = (x_1, \ldots, x_{nq})^{\dagger}$ with $n_q = 9$. Each component x_j determines an R_Y rotation on qubit j. The combined rotations prepare the encoded quantum state $|\psi_{enc(x)}\rangle$.

$$|\psi_{enc}(x)\rangle = U_{enc}(x)|0\rangle^{\otimes n_q} \tag{3}$$

$$U_{enc}(x) = \bigotimes_{j=1}^{n_q} R_Y(x_j) \tag{4}$$

After encoding, the PQC applies a trainable unitary transformation $U(\theta)$, as shown in Eq. (5). The vector θ contains the circuit parameters that are updated during training. Increasing the number of parameters can increase representation capacity, but it also increases circuit and optimization complexity. The QFE circuit therefore used a compact QAOA-inspired ansatz [16, 18].

$$|\psi_{enc}(x;\theta)\rangle = U(\theta)|\psi_{enc}(x)\rangle \tag{5}$$

QFE converts the quantum state back into numerical features by measuring $Pauli\ Z$ observables. Eq. (6) writes this measurement as an expectation value. The equation then adds the bias b. The activation σ maps the resulting scalar to the interval $[-\pi, \pi]$, allowing it to serve as an input to the next QFE layer.

$$Y_3(x) = \sigma\left(Tr\left(|x;\theta\rangle\langle x;\theta|\hat{O}\right) + b\right) \tag{6}$$

Eq. (7) gives the explicit form $\sigma(z) = \pi\ tanh(z)$. The factor π changes only the output range and does not add a trainable parameter. Eq. (8) combines encoding, the trainable circuit, measurement, bias, and activation into one expression for a single QFE layer.

$$\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \times \pi \tag{7}$$

$$Y(x) = \sigma\left(Tr\left(U(\theta)U(x)|0\rangle\langle 0|U(x)^{\dagger}U(\theta)^{\dagger}\hat{O}\right) + b\right) \tag{8}$$

Training requires the loss gradient to be propagated through both the classical and quantum parts of the model. Eq. (9) expresses this propagation through the Chain Rule, where ξ can represent a trainable parameter or an intermediate input. Eq. (10) gives the Parameter-Shift Rule for a compatible parameterized quantum gate. The rule evaluates the circuit at positively and negatively shifted parameter values.

$$\frac{\partial L}{\partial \xi} = \frac{\partial L}{\partial A}\frac{\partial A}{\partial \xi} \tag{9}$$

$$\frac{\partial f(\phi)}{\partial \phi} = \frac{1}{2}\left[f\left(\phi + \frac{\pi}{2}\right) - f\left(\phi - \frac{\pi}{2}\right)\right] \tag{10}$$

In the original two-layer QFE formulation, the Parameter-Shift Rule was applied to derivatives with respect to circuit parameters and encoded inputs [16, 17, 19, 33]. This formulation is relevant to gradient evaluation on quantum hardware. However, the simulator experiments in the present study used PennyLane backpropagation rather than explicit parameter-shift evaluations.

3. PROPOSED METHOD

This study extended the two-block QFE architecture with an ancilla-based quantum attention branch. QFE1 produced one nine-dimensional observable vector for each image patch. The new branch converted each vector into a scalar relevance score. Softmax then compared the patch scores and produced normalized weights for a weighted QFE1 representation.

The weighted QFE1 representation and the QFE2 output each contained nine values. Their concatenation therefore produced an 18-dimensional input to the classifier. The QA circuit used nine feature qubits and one ancilla qubit, giving a maximum width of ten qubits. Figures 1(c), 2(b), and 3 show the architecture, circuit configuration, and end-to-end workflow, respectively.

Each 22 × 22 input image was divided into $P = 100$ overlapping 3 × 3 patches. Flattening each patch produced a nine-dimensional vector. Consequently, the patch tensor X had shape $P \times d$ with $d = 9$, as defined in Eq. (11). This operation is analogous to selecting the local receptive fields processed by a CNN filter.

$$X = \left[x_p^{(1)}, x_p^{(2)}, \ldots, x_p^{(P)}\right] \in \mathbb{R}^{P \times d} \tag{11}$$

QFE1 processed each patch separately and measured one *Pauli Z* expectation value from each of the nine feature qubits. Eq. (12) denotes the resulting vector for patch i as $z^{(i)}$. Eq. (13) then stacks the vectors from all patches into Z. Thus, Z contains both the patch dimension and the nine quantum features associated with each patch.

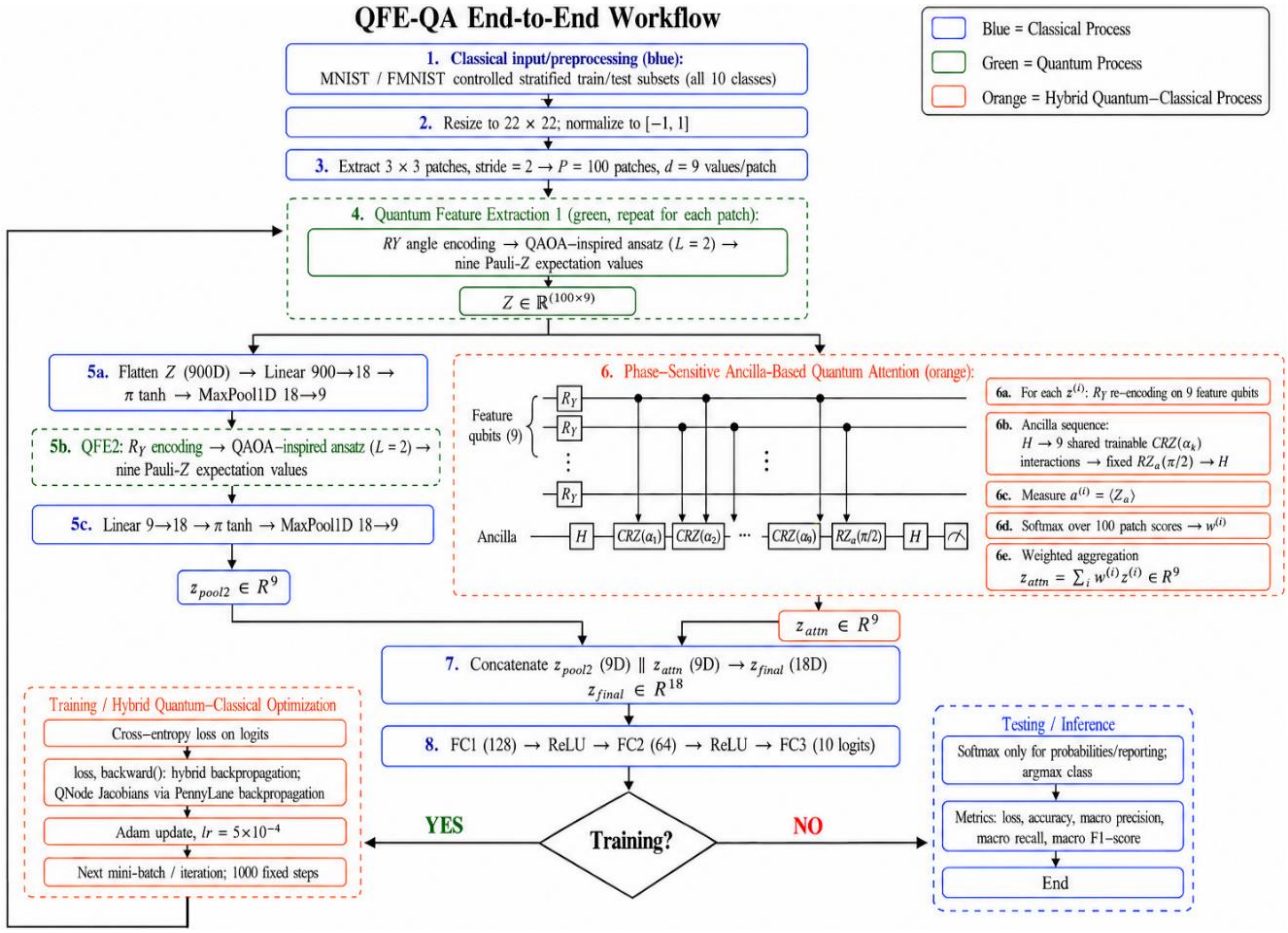


Figure 3. End-to-end workflow of the compact Quantum Feature Extraction-Quantum Attention (QFE-QA) architecture
Note: The QFE2 and phase-sensitive ancilla-based attention branches produce two nine-dimensional representations, which are concatenated into the 18-dimensional classifier input. The reported simulator experiments use PennyLane backpropagation for hybrid-model differentiation.

$$z^{(i)} = \left[\langle \hat{Z}_1 \rangle, \langle \hat{Z}_2 \rangle, \dots, \langle \hat{Z}_{n_q} \rangle\right]^{(i)} \in \mathbb{R}^{n_q} \tag{12}$$

$$Z = \left[z^{(1)}, z^{(2)}, \dots, z^{(P)}\right] \in \mathbb{R}^{P \times n_q} \tag{13}$$

The attention branch received each QFE1 vector $z^{(i)}$ and produced one scalar score $a^{(i)}$, as defined in Eq. (14). A larger score did not have a fixed meaning before training. Instead, the classification loss learned which score patterns were useful. The same parameter vector α was reused for every patch, so the number of attention parameters did not grow with the number of patches.

Eq. (15) applies softmax across the P patch scores. Softmax converts the scores into positive weights $w^{(i)}$ that sum to one, which makes the weights comparable across patches. Eq. (16) then computes z_{attn} as the weighted sum of the QFE1 vectors, producing one nine-dimensional representation for the complete image.

$$a^{(i)} = f_{attn}\left(z^{(i)}; \alpha\right) \in \mathbb{R} \tag{14}$$

$$w^{(i)} = \frac{exp\left(a^{(i)}\right)}{\sum_{j=1}^{P} exp\left(a^{(j)}\right)} (Softmax) \tag{15}$$

$$z_{attn} = \sum_{i=1}^{P} w^{(i)} z^{(i)} \in \mathbb{R}^{n_q} \tag{16}$$

The main QFE path processed the same QFE1 outputs independently of the attention branch. The matrix Z was first flattened and projected by W_{p1}. It was then reduced by one-dimensional max pooling, as shown in Eqs. (17) and (18). This step compressed the features from all patches into the nine-dimensional vector z_{pool1}.

$$z_{proj1} = \pi \, tanh\left(W_{p1} z_{flat} + b_{p1}\right) \in \mathbb{R}^{d_1} \tag{17}$$

$$z_{pool1} = MaxPool1D\left(z_{proj1}\right) \in \mathbb{R}^{n_q} \tag{18}$$

where, $W_{p1} \in \mathbb{R}^{d_1 \times (Pn_q)}$ is the QFE1 projection matrix and $MaxPool1D$ is the pooling of the projection channels. Here W_{p1} is a linear projection matrix that maps the flattened representation of the entire patches, of dimension Pn_q, to the feature space of dimension d_1.

QFE2 received z_{pool1} and produced nine *Pauli Z* expectation values, as defined in Eq. (19). These values were projected and pooled again through Eqs. (20) and (21). The resulting vector z_{pool2} had the same dimension as z_{attn}, which

allowed the two branches to be combined directly.

$$z_{qfe2}^{raw} = \left[\langle Z_1\rangle, \langle Z_2\rangle, \ldots, \langle Z_{n_q}\rangle\right] = QFE2\left(z_{pool1}\right) \in \mathbb{R}^{n_q} \quad (19)$$

$$z_{proj2} = \pi\, tanh\left(W_{p2} z_{qfe2}^{raw} + b_{p2}\right) \in \mathbb{R}^{d_2} \quad (20)$$

$$z_{pool2} = MaxPool1D\left(z_{proj2}\right) \in \mathbb{R}^{n_q} \quad (21)$$

where, $W_{p2} \in \mathbb{R}^{d_2 \times n_q}$ is the projection weight of QFE2, and the output $z_{pool2} \in \mathbb{R}^{n_q}$ has the same dimensionality as the attention output.

Eq. (22) concatenates z_{pool2} and z_{attn} rather than adding them. Because each vector has nine components, the combined vector z_{final} has 18 components. The concatenation preserves information from both branches. This vector was passed to the fully connected classifier.

$$z_{final} = \left[z_{pool2} \parallel z_{attn}\right] \in \mathbb{R}^{2n_q} \quad (22)$$

with $n_q = 9 \longrightarrow$ then $z_{final} \in \mathbb{R}^{18}$, and $\parallel$ denotes vector concatenation.

The classifier contained hidden layers with 128 and 64 units followed by a ten-class output layer, as defined in Eqs. (23)–(25). ReLU was used in the two hidden layers. The final layer produced logits for the ten classes. Eq. (26) computes cross-entropy directly from the logits and the class-index label y, and softmax was used only to obtain class probabilities for reporting.

$$h_1 = ReLU\left(W_{fc1} \cdot z_{final} + b_1\right) \in \mathbb{R}^{128} \quad (23)$$

$$h_2 = ReLU\left(W_{fc2} \cdot h_1 + b_2\right) \in \mathbb{R}^{64} \quad (24)$$

$$\ell = W_{fc3} \cdot h_2 + b_3 \in \mathbb{R}^{C} \quad (25a)$$

$$\hat{y} = Softmax(\ell) \quad (25b)$$

$$L = CrossEntropy(\ell, y) \quad (26)$$

where, $\hat{y}$ denotes the predicted class-probability vector, C is the number of classes, b is the bias, y denotes the class-index ground-truth labels, ℓ denotes logits, and L is the loss computed at each training iteration.

Algorithm 1 summarizes one QFE-QA training iteration.

Algorithm 1. Quantum Feature Extraction-Quantum Attention (QFE-QA)

Input:
$x \in \mathbb{R}^{B\times 1\times 22\times 22}$ # Batch input images
$y \in \{0, \ldots, 9\}^B$ # Class-index labels
Model: QFE-QA neural model with parameter Θ.
Optimizer and $loss_{fn}$: training utilities.

Procedure:
Step 1: Patch Extraction
$patches \leftarrow extract_{patches}(x, patch_{size} = 3, stride = 2)$
$patches \in \mathbb{R}^{B\times 100\times 9}$
Step 2: QFE1
$\left(qfe1_{pool}, qfe1_{patchwise}\right) \leftarrow QFE1(patches)$
$qfe1_{patchwise} \in \mathbb{R}^{B\times 100\times 9}$ # Patch-level observables (Pauli Z)
$qfe1_{pool} \in \mathbb{R}^{B\times 9}$ # Pooled & projected from QFE1
Step 3: Ancilla-Based Quantum Attention
Initialize shared trainable angles $\alpha \in R^{nq}$ and fixed $\delta = \frac{\pi}{2}$.
For each patch $i = 1, \ldots, P$:
$\phi^{(i)} \leftarrow \frac{\pi}{2}\left(z^{(i)} + 1\right)$
$a^{(i)} \leftarrow \langle Z_a\rangle$ obtained from $U_{QA}\left(z^{(i)}, \alpha, \delta\right)$, as defined in Eq. (28)
$w \leftarrow Softmax(a, dim = patch)$
$z_{attn} \leftarrow \sum_i w^{(i)} z^{(i)}$
Step 4: QFE2
$qfe2_{raw} \leftarrow QFE2\left(qfe1_{pool}\right)$
$qfe2_{raw} \in \mathbb{R}^{B\times 9}$
Step 5: Pooling2
$qfe2_{proj} \leftarrow \pi\, tanh\left(W_{p2} \cdot qfe2_{raw} + b_{p2}\right)$
$qfe2_{pool} \leftarrow MaxPool1D\left(qfe2_{proj}\right)$
$qfe2_{pool} \in \mathbb{R}^{B\times 9}$
Step 6: Feature Concatenation
$z_{final} \leftarrow concatenation\left(qfe2_{pool}, z_{attn}\right)$
$z_{final} \in \mathbb{R}^{B\times 18}$
Step 7: FC Network
$h_1 \leftarrow ReLU\left(W_{fc1} \cdot z_{final} + b_{fc1}\right)$ # $\mathbb{R}^{B\times 128}$
$h_2 \leftarrow ReLU\left(W_{fc2} \cdot h_1 + b_{fc2}\right)$ # $\mathbb{R}^{B\times 64}$
$logits \leftarrow W_{fc3} \cdot h_2 + b_{fc3}$ # $\mathbb{R}^{B\times 10}$
$logits \in \mathbb{R}^{B\times 10}$
Step 8: Compute Loss and Backpropagation
$loss \leftarrow CrossEntropy(logits, y)$
$\hat{y} \leftarrow Softmax(logits)$ # Only for probabilities/reporting
$loss.backward()$
$optimizer.step()$
Output:
Updated model parameters Θ after one training iteration.

In the reported simulator implementation, loss.backward() propagates the classification gradient through the hybrid model, while the quantum-node Jacobians are computed using PennyLane backpropagation.

Note: The same trainable attention vector α was shared across all patches. The fixed $R_Z(\pi/2)$ phase and both Hadamard gates contained no trainable parameters. Consequently, the attention branch added nine trainable CRZ angles rather than a separate set of angles for every patch.

Eqs. (27) and (28) expand the score function introduced in Eq. (14). For patch i, the QA circuit started from the all-zero state ρ_0 of the nine feature qubits and one ancilla qubit. The feature register encoded $z^{(i)}$, and a Hadamard gate prepared the ancilla. The nine feature qubits then controlled CRZ rotations on that ancilla.

$$\begin{aligned} a^{(i)} &= f_{attn}\left(z^{(i)}; \alpha\right) \\ &= \langle Z_a\rangle_i = Tr\left[(I_F \otimes Z_a) U_{QA}\left(z^{(i)}, \alpha\right) \rho_0 U_{QA}^{\dagger}\left(z^{(i)}, \alpha\right)\right] \\ &= P_a^{(i)}(0) - P_a^{(i)}(1) \in [-1, 1] \end{aligned} \quad (27)$$

$$\begin{aligned} U_{QA} &= (I_F \otimes H_a) \\ &\left(I_F \otimes R_{Z_a}(\delta)\right) U_{int}(\alpha) \left[U_{enc}\left(z^{(i)}\right) \otimes H_a\right], \\ U_{int}(\alpha) &= \prod_{k=1}^{n_q} CRZ_{k\to a}\left(\alpha_k\right), \delta = \frac{\pi}{2} \end{aligned} \quad (28)$$

where, I_F is the identity on the feature register, $U_{enc}\left(z^{(i)}\right) = \bigotimes_{k=1}^{n_q} R_{Y_k}\left(\phi_k^{(i)}\right)$, with $\phi_k^{(i)} = \frac{\pi}{2}\left(z_k^{(i)} + 1\right)$, and $\rho_0 = |0\rangle\langle 0|^{\otimes(n_q+1)}$ is the all-zero initial state of the nine feature

qubits and one ancilla qubit.

Each QFE1 value $z_k^{(i)}$ lies in $[-1,1]$ and was mapped monotonically to $\phi_k^{(i)} = \pi\,(z_k^{(i)} + 1)/2$. This mapping preserved the input order and set the population of feature qubit k. The shared angle α_k then controlled how strongly that qubit influenced the ancilla phase.

The fixed $R_Z(\pi/2)$ rotation changed the ancilla readout from a cosine-sensitive to a sine-sensitive response. Near the zero initialization of α, it provided the nonzero first-order response $a^{(i)} \approx -\Sigma_k\, p_k^{(i)} \alpha_k$. The final Hadamard converted the accumulated phase into the Pauli Z difference $P_a^{(i)}(0) - P_a^{(i)}(1)$, which formed the scalar patch score.

Eqs. (29) and (30) convert the P patch scores into normalized weights and form the weighted representation z_{attn}. Index i denotes the weighted patch, j is the dummy index in the softmax denominator, and k indexes the $n_q = 9$ feature qubits. The score was not assumed to represent semantic importance before training; its useful interpretation was learned through the classification loss.

$$w^{(i)} = \frac{exp(a^{(i)})}{\sum_{j=1}^{P} exp(a^{(j)})},\ i = 1, \dots, P \tag{29}$$

$$z_{attn} = \sum_{i=1}^{P} w^{(i)} z^{(i)} \in \mathbb{R}^{n_q} \tag{30}$$

where, $\alpha = \left(\alpha_1, \alpha_2, \dots, \alpha_{n_q}\right)^{\dagger}$ contains one trainable CRZ angle per feature qubit, while $a^{(i)}$, $w^{(i)}$, and z_{attn} denote the score of patch i, its normalized weight, and the weighted aggregation, respectively.

4. EXPERIMENTAL RESULTS AND DISCUSSION

4.1 Datasets

MNIST and Fashion-MNIST (FMNIST) were selected to preserve the all-class image-classification setting used in the original QFE evaluation [16]. The original training and test splits contained 60,000 and 10,000 images, respectively [34, 35]. From these splits, balanced subsets of 12,000 training images and 2,000 test images were sampled across all ten classes. Table 2 summarizes the original dataset properties.

Table 2. Dataset descriptions

Datasets	Dimension	Original Train Set	Original Test Set	Class/ Label	Color Channel
MNIST	28 × 28	60,000	10,000	10	Grayscale (1 Dimension)
FMNIST	28 × 28	60,000	10,000	10	Grayscale (1 Dimension)

All images were resized to 22 × 22 pixels, which produced 100 patches per image. This setting limited the maximum circuit width to ten qubits. This controlled configuration was intended as a proof of concept rather than a scalability test. Although the nine attention angles were shared across patches, QFE1 and QA still required one circuit evaluation for each patch, so performance on larger datasets, higher-resolution images, and RGB inputs remains unverified.

4.2 Preprocessing and tools

All images were resized to 22 × 22 pixels and normalized to $[-1,1]$. The experiments were implemented in Python 3.12.13 with PyTorch 2.11.0 on the CPU and PennyLane 0.45.1 using the default.qubit backend. The circuits returned analytic expectation values with shots = None, which means that finite-shot sampling noise was not simulated. PennyLane backpropagation was used to differentiate the hybrid model, whereas the PSR in Section 2.2 was retained as the theoretical and hardware-compatible formulation of the original QFE method. Training used shuffled mini-batches of 64 samples without replacement. With 12,000 training samples and the final incomplete mini-batch retained, one epoch contained 188 mini-batches, so 1,000 update steps corresponded to approximately 5.3 epochs.

The QFE R_Z and R_X parameters were initialized from $U(0,2\pi)$, while the nine CRZ attention parameters were initialized from $N(0,0.1^2)$. The projection layers retained the default PyTorch linear-layer initialization. Adam was used with a learning rate of 5×10^{-4}, $\beta = (0.9,0.999)$, $\varepsilon = 10^{-8}$, zero weight decay, and no learning-rate scheduler. A distinct automatically generated seed controlled Python, NumPy, PyTorch, stratified subset selection, and mini-batch shuffling in each run. The five runs were independent and were not seed-matched across models.

4.3 Comparison of methods

QFE-QA was compared primarily with its direct predecessor, QFE, while the CNN from the study [16] served only as a classical reference. All models used the same subsets, input resolution, preprocessing, mini-batch size, optimizer, learning rate, update steps, and classifier widths. Because their feature representations, architectures, and parameter counts differed, the experiment standardized data and training but was not capacity matched. Recorded outcomes comprised training and test loss, accuracy, macro precision, macro recall, macro F1-score, and FC-only forward time.

The FC-only time measured the classifier head and excluded all QFE and QFE-QA circuit executions. Simulator time would also include classical statevector overhead, so the reported timing was treated only as a classifier-head diagnostic and did not support claims of end-to-end speedup or quantum advantage. Execution on a real quantum processing unit would introduce finite-shot sampling, gate and readout errors, decoherence, and backend-dependent routing. These effects could alter the ancilla scores and increase the effective circuit depth. QFE1 and QA also required 100 patch-circuit evaluations per image, while hardware-compatible gradients would add circuit executions. The noiseless results therefore did not establish hardware robustness, scalability, or practical quantum advantage.

Five independently initialized runs were conducted to characterize variation caused by parameter initialization and stochastic training. Performance was summarized by the arithmetic mean and sample standard deviation. No additional hyperparameter tuning, batch normalization, gradient clipping, weight decay, dropout, or other regularization was applied. All models were trained for 1,000 parameter-update steps under the same restricted protocol [16]. With the selected subset and mini-batch size, this budget corresponded to

approximately 5.3 epochs.

4.4 Results

Tables 3 and 4 present the results from five independent runs and provide the primary evidence for model comparison. Because the runs were not seed-matched, QFE-QA and QFE were compared with an exploratory two-sided Mann-Whitney U test. The test used exhaustive permutation p-values and a significance threshold of 0.05. The smaller of the two U statistics is denoted by U_{min}.

On MNIST, mean test accuracy was 0.9313 ± 0.0029 for QFE-QA, 0.9213 ± 0.0085 for QFE, and 0.9239 ± 0.0056 for CNN. On FMNIST, it was 0.8188 ± 0.0068, 0.8073 ± 0.0068, and 0.7891 ± 0.0095, respectively. QFE-QA therefore had the smallest sample standard deviation on MNIST and tied QFE on FMNIST. It exceeded QFE by 1.00 and 1.15 percentage points, respectively, but neither difference was statistically significant (MNIST: $U_{min} = 5.0$, $p = 0.1349$; FMNIST: $U_{min} = 3.5$, $p = 0.0635$). The evidence indicates numerically higher mean accuracy, not statistically established superiority.

Table 5 presents the experimental configurations and selected results from the best test-accuracy run. Figure 4 shows representative training curves, and Figure 5 shows representative confusion matrices from those runs. These best-run displays are descriptive examples and were not used as the main basis for statistical or comparative conclusions.

Table 3. Five-run performance and run-to-run variability on MNIST (bold is the best value)

Setup		Performance							
Run	**Methods**	**Train Loss**	**Test Loss**	**Train Accuracy**	**Test Accuracy**	**Precision**	**Recall**	**F1-Score**	**Fully Connected (FC) Time**
1	CNN	**0.0821**	0.2760	**0.9844**	0.9145	0.9169	0.9145	0.9138	0.86s
1	QFE	0.4049	0.3066	0.8750	0.9165	0.9172	0.9165	0.9164	**0.23s**
1	QFE-QA	0.2934	**0.2511**	0.9219	**0.9265**	**0.9274**	**0.9265**	**0.9264**	**0.23s**
2	CNN	**0.2029**	0.2431	**0.9531**	0.9260	0.9285	0.9260	0.9254	0.86s
2	QFE	0.2507	0.2579	0.9219	0.9215	0.9254	0.9215	0.9216	0.31s
2	QFE-QA	0.2332	**0.2319**	0.9062	**0.9335**	**0.9353**	**0.9335**	**0.9334**	**0.23s**
3	CNN	0.2537	0.2428	0.9062	0.9265	0.9278	0.9265	0.9265	0.89s
3	QFE	0.3449	0.3018	0.8750	0.9155	0.9165	0.9155	0.9152	**0.24s**
3	QFE-QA	**0.0593**	**0.2240**	**0.9844**	**0.9325**	**0.9329**	**0.9325**	**0.9325**	0.26s
4	CNN	0.2600	0.2371	0.8906	0.9290	0.9292	0.9290	0.9287	1.03s
4	QFE	0.0791	**0.2273**	**1.0000**	**0.9360**	**0.9362**	**0.9360**	**0.9359**	0.25s
4	QFE-QA	**0.0391**	0.2325	**1.0000**	0.9305	0.9307	0.9305	0.9303	**0.22s**
5	CNN	0.2504	**0.2347**	0.8750	0.9235	0.9250	0.9235	0.9234	0.87s
5	QFE	0.3041	0.2532	0.9219	0.9170	0.9179	0.9170	0.9166	**0.24s**
5	QFE-QA	**0.1059**	0.2533	**0.9531**	**0.9335**	**0.9348**	**0.9335**	**0.9336**	**0.24s**
					Five-Run Mean				
	CNN	0.2098	0.2467	0.9219	0.9239	0.9255	0.9239	0.9236	0.90s
	QFE	0.2767	0.2694	0.9188	0.9213	0.9226	0.9213	0.9211	0.25s
	QFE-QA	**0.1462**	**0.2386**	**0.9531**	**0.9313**	**0.9322**	**0.9313**	**0.9312**	**0.24s**

Table 4. Five-run performance and run-to-run variability on FMNIST (bold is the best value)

Setup		Performance							
Run	**Methods**	**Train Loss**	**Test Loss**	**Train Accuracy**	**Test Accuracy**	**Precision**	**Recall**	**F1-Score**	**Fully Connected (FC) Time**
1	CNN	**0.3952**	0.5468	**0.9062**	0.7985	0.7978	0.7985	0.7970	1.33s
1	QFE	0.5664	**0.4977**	0.7656	0.8140	0.8185	0.8140	0.8148	**0.25s**
1	QFE-QA	0.4352	0.4978	0.8594	**0.8220**	**0.8240**	**0.8220**	**0.8203**	0.26s
2	CNN	0.4942	0.5780	0.8125	0.7810	0.8148	0.7810	0.7807	0.83s
2	QFE	0.5420	0.5396	0.7812	0.7970	0.7955	0.7970	0.7941	**0.24s**
2	QFE-QA	**0.3119**	**0.5101**	**0.8750**	**0.8130**	**0.8082**	**0.8130**	**0.8085**	0.26s
3	CNN	**0.4790**	0.5876	**0.7812**	0.7830	0.7830	0.7830	0.7771	0.85s
3	QFE	0.5265	0.5463	0.7656	0.8055	0.8012	0.8055	0.7987	**0.25s**
3	QFE-QA	0.6208	**0.5290**	0.7656	**0.8100**	**0.8069**	**0.8100**	**0.8043**	0.27s
4	CNN	0.5000	0.5292	0.8125	0.8005	0.8035	0.8005	0.7968	0.90s
4	QFE	**0.3086**	0.5568	**0.9219**	0.8070	0.8088	0.8070	0.8035	0.25s
4	QFE-QA	0.6380	**0.4954**	0.8281	**0.8245**	**0.8248**	**0.8245**	**0.8241**	**0.23s**
5	CNN	0.6876	0.5921	0.7188	0.7825	0.7874	0.7825	0.7809	0.98s
5	QFE	0.6040	0.5243	0.7188	0.8130	0.8199	0.8130	0.8131	**0.25s**
5	QFE-QA	**0.4301**	**0.5126**	**0.8281**	**0.8245**	**0.8301**	**0.8245**	**0.8248**	0.28s
					Five-Run Mean				
	CNN	0.5112	0.5667	0.8062	0.7891	0.7973	0.7891	0.7865	0.98s
	QFE	0.5095	0.5329	0.7906	0.8073	0.8088	0.8073	0.8048	**0.25s**
	QFE-QA	**0.4872**	**0.5090**	**0.8312**	**0.8188**	**0.8188**	**0.8188**	**0.8164**	0.26s

Note: CNN = Convolutional Neural Network, QFE = Quantum Feature Extraction, QA = Quantum Attention, PQC = parameterized quantum circuit, QKV = Query-Key-Value, FC = Fully Connected.

Table 5. Experimental configurations and selected best-test-run results (bold is the best value)

Modules and Parameters	Convolutional Neural Network (CNN)	Quantum Feature Extraction (QFE)	Quantum Feature Extraction-Quantum Attention (QFE-QA)
First layer	Convolution 1 (1, 18, 3 × 3, stride = 2, pad = 1)	QFE1 (9 qubits – 2 layers)	QFE1 (9 qubits – 2 layers)
Number of output channels	1 × 18 × 11 × 11	18 channels (with linear dimension projection)	18 channels (with linear dimension projection)
Pooling 1	Channel Pooling (MaxPooling2d) 1 × 9 × 11 × 11	Channel Pooling (MaxPooling1d) 9 channels	Channel Pooling (MaxPooling1d) 9 channels
Second layer	Convolution 2 (9, 36, 3 × 3, stride = 2, pad = 1)	QFE2 (9 qubits – 2 layers)	QFE2 (9 qubits – 2 layers)
Number of output channels	1 × 36 × 6 × 6	36 channels (with linear dimension projection)	18 channels (with linear dimension projection)
Pooling 2	Channel Pooling (MaxPooling2d) 1 × 18 × 6 × 6	Channel Pooling (MaxPooling1d) 18 channels	Channel Pooling (MaxPooling1d) 9 channels
Additional layer	-	-	Quantum Attention 9 channels
Classifier	Fully Connected Network FC1(128) – FC2(64) – FC3(10)	Fully Connected Network FC1(128) – FC2(64) – FC3(10)	Fully Connected Network FC1(128) – FC2(64) – FC3(10)
Batch size	64	64	64
Optimizer	Adam	Adam	Adam
Learning rate	0.0005	0.0005	0.0005
Iteration	1000 fixed steps (~5.3 epoch-equivalent)	1000 fixed steps (~5.3 epoch-equivalent)	1000 fixed steps (~5.3 epoch-equivalent)
Controlled subset (all 10 classes)	Train = 12,000 Test = 2,000	Train = 12,000 Test = 2,000	Train = 12,000 Test = 2,000
MNIST Dataset			
Trainable parameters	95110	27988	**27817**
Classifier runtime (Fully Connected (FC) time)	1.03 s	0.25 s	**0.23 s**
Test error rate	0.0710	**0.0640**	0.0665
Test accuracy	0.9290	**0.9360**	0.9335
Precision	0.9292	**0.9362**	0.9353
Recall	0.9290	**0.9360**	0.9335
F1-score	0.9287	**0.9359**	0.9334
FMNIST Dataset			
Trainable parameters	95110	27988	**27817**
Classifier runtime (FC time)	0.90s	0.25s	**0.23s**
Test error rate	0.1995	0.1860	**0.1755**
Test accuracy	0.8005	0.8140	**0.8245**
Precision	0.8035	0.8185	**0.8248**
Recall	0.8005	0.8140	**0.8245**
F1-score	0.7968	0.8148	**0.8241**

Table 6. Comparison of trainable parameters (bold is the best value)

Methods	Image Dimensions	Block 1	Block 2	FC1 (128)	FC2 (64)	FC3 (10)	Total
Convolutional Neural Network (CNN)	1 × 22 × 22	(1 grayscale × 3 × 3 kernels × 18 dimensions) + 18 bias = 180	(9 dimensions × 3 × 3 kernels × 36 dimensions) + 36 bias = 2,952	(648 dimensions × 128 neurons) + 128 bias = 83,072	(128 × 64) + 64 = 8,256	(64 × 10) + 10 = 650	95,110
Quantum Feature Extraction (QFE)	1 × 22 × 22	2 layers (9 RZ + 9 RX) + (100 patches × 9 observables × 18 dimensions) + 18 bias = 16,254	2 Layers (9 RZ + 9 RX) + (9 observables × 36 dimensions) + 36 bias = 396	(18 dimensions × 128 neurons) + 128 bias = 2,432	(128 × 64) + 64 = 8,256	(64 × 10) + 10 = 650	27,988
Quantum Feature Extraction-Quantum Attention (QFE-QA)	1 × 22 × 22	2 layers (9 RZ + 9 RX) + (9 CRZ) + (100 patches × 9 observables × 18 dimensions) + 18 bias = 16,263	2 Layers (9 RZ + 9 RX) + (9 observables × 18 dimensions) + 18 bias = 216	(18 dimensions × 128 neurons) + 128 bias = 2,432	(128 × 64) + 64 = 8,256	(64 × 10) + 10 = 650	**27,817**

Note: Trainable parameters comprise all optimized scalar circuit angles and convolutional, projection, and FC weights and biases. Each two-layer QFE block has 36 shared angles [2 × (9 RZ + 9 RX)], and QFE-QA adds nine shared CRZ attention angles. These angles are counted once because they are reused across 100 patches. Encoding, fixed gates, observables, the ancilla, pooling, concatenation, and softmax have no trainable parameters. Totals represent unmatched native architectures evaluated under the same protocol.

Accuracy, precision, recall, and F1-score were calculated for each run. Precision, recall, and F1-score were macro-averaged across classes so that every class had equal weight; because each test class contained 200 samples, macro recall

could equal or closely approach accuracy. Eq. (31) then averages each metric across the five runs, where m_r is the value from run r and $R = 5$.

$$\bar{m} = \frac{1}{R}\sum_{r=1}^{R} m_r \, , R = 5 \tag{31}$$

Table 6 compares trainable-parameter counts for the three evaluated architectures. The quantum models had fewer total parameters mainly because their 18-dimensional classifier input required 2,432 parameters in FC1, whereas the CNN's 648-dimensional input required 83,072. QFE-QA added nine shared CRZ angles but reduced the Block 2 projection from 396 to 216 parameters. These counts describe model size only; they do not establish trainability, runtime efficiency, superiority to lightweight CNNs, or quantum advantage [36-39].

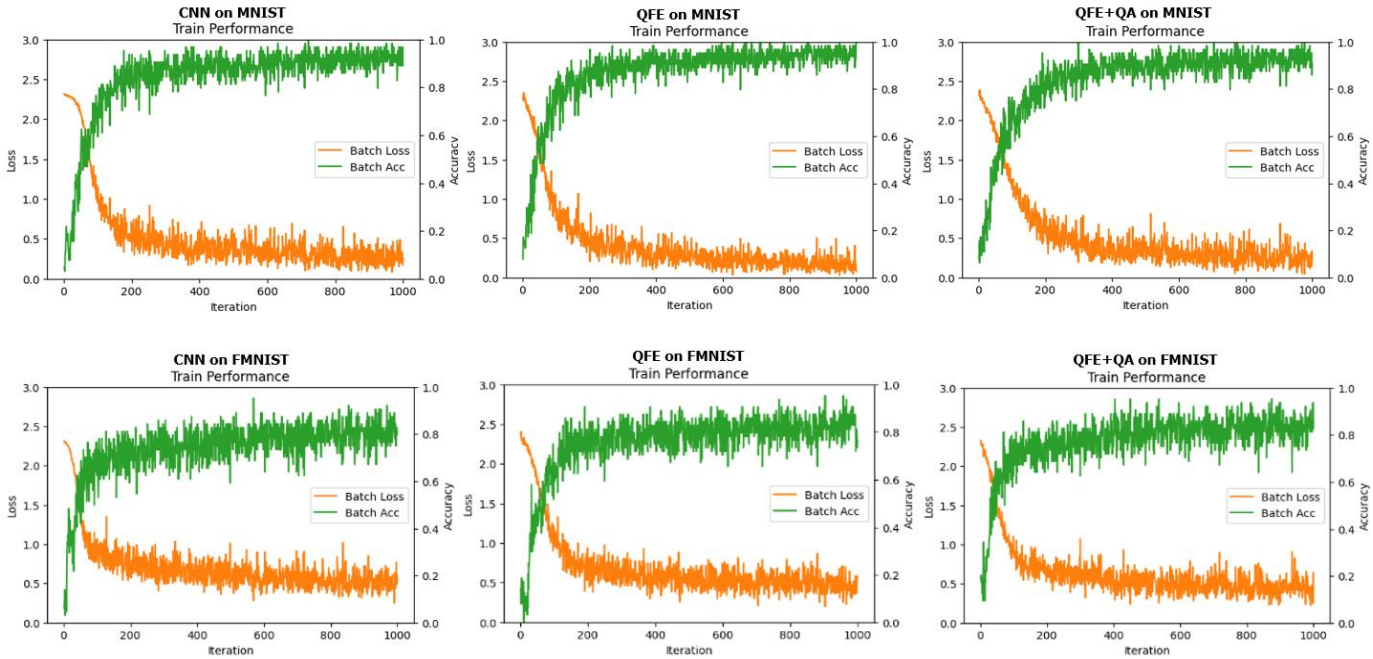


Figure 4. Representative best-run training curves of the evaluated methods

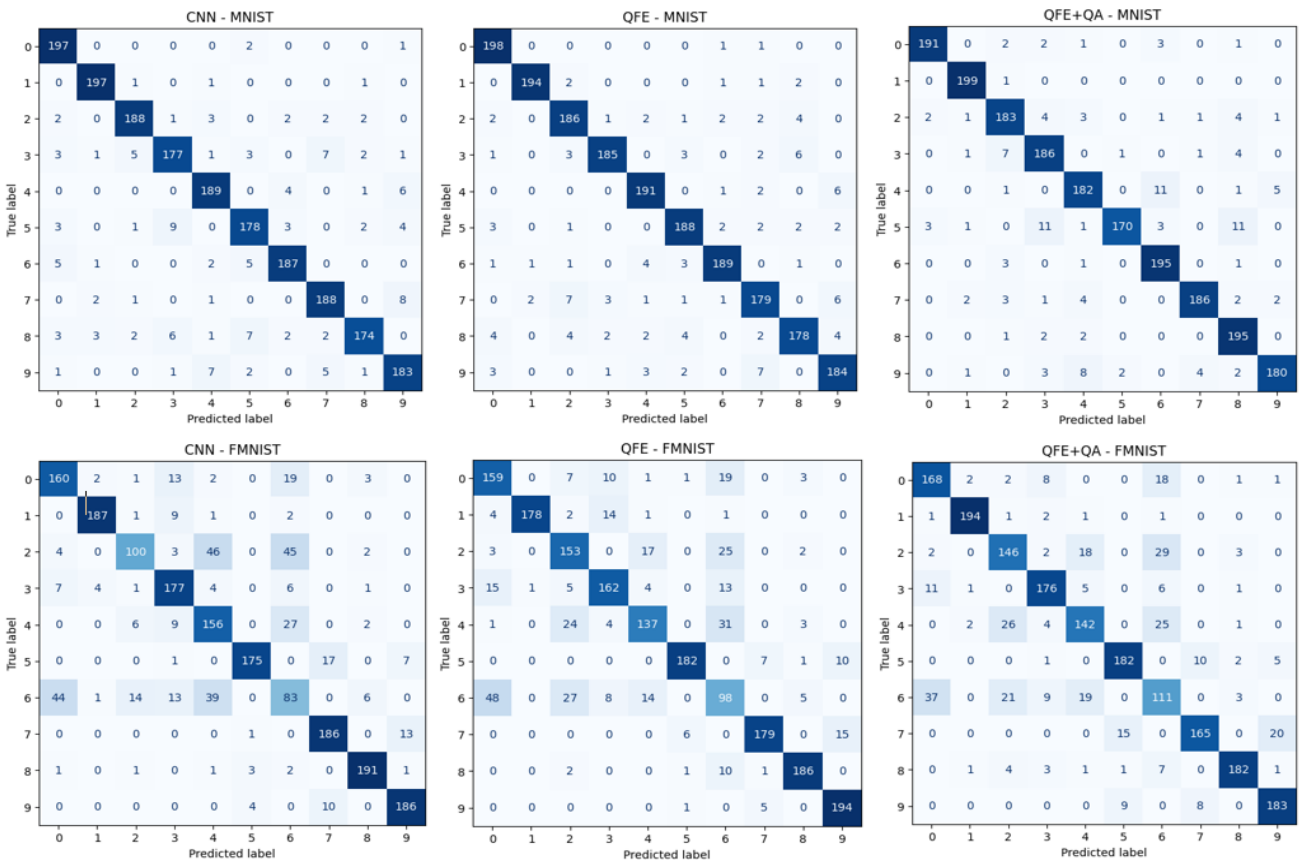


Figure 5. Representative best-run confusion matrices for the evaluated methods

Table 7. Architecture-level ablation of the integrated QA branch and Block 2 configuration (bold is the best value)

Methods	Five-Run Mean Test Accuracy	Five-Run Mean F1-Score	Five-Run Mean FC Time	Trainable Parameters
		MNIST Dataset		
QFE	0.9213	0.9211	0.25 s	27,988
QFE-QA	0.9313	0.9312	**0.24 s**	**27,817**
QFE-QA-18D	**0.9334**	**0.9333**	0.25 s	29,149
		FMNIST Dataset		
QFE	0.8073	0.8048	**0.25 s**	27,988
QFE-QA	0.8188	0.8164	0.26 s	**27,817**
QFE-QA-18D	**0.8234**	**0.8210**	0.26 s	29,149

Note: QFE-QA-18D retains the original 18-dimensional Block 2 output. Its nine-dimensional attention vector produces a 27-dimensional FC1 input and 29,149 trainable parameters. QFE = Quantum Feature Extraction; QA = Quantum Attention.

Table 7 summarizes the ablation of the integrated attention branch and the Block 2 output dimension. QFE-QA concatenated nine-dimensional QFE2 and attention-weighted QFE1 vectors, whereas QFE-QA-18D retained the original 18-dimensional Block 2 output. QFE-QA-18D achieved numerically higher mean accuracy than QFE on both datasets, indicating that the gains were not caused only by reducing the Block 2 dimension. However, classical-attention and non-ancilla controls, attention maps, latent-space analysis, and component-level ablations were not evaluated. The evidence therefore supports the integrated architecture but not the superiority of its individual quantum components.

5. CONCLUSION

This study evaluated QFE-QA, which integrated a phase-sensitive ancilla-based attention branch with QFE. The method produced numerically higher mean test accuracy than QFE on both evaluated datasets, but the differences were not statistically significant. The controlled grayscale subsets and noiseless statevector simulator demonstrated feasibility only within the tested setting and did not establish scalability, end-to-end efficiency, generalizability, or quantum advantage. Future work should evaluate larger RGB datasets, higher resolutions, noisy and transpiled circuits, real quantum hardware, parameter-matched lightweight CNNs, attention maps, latent representations, and component-level ablations.

ACKNOWLEDGMENT

The authors thank the Faculty of Computer Science, Universitas Dian Nuswantoro; the Intelligence Distributed Surveillance and Security Research Group; and the Research Center for Materials Informatics.

REFERENCES

[1] Elharrouss, O., Akbari, Y., Almadeed, N., Al-Maadeed, S. (2024). Backbones-review: Feature extractor networks for deep learning and deep reinforcement learning approaches in computer vision. Computer Science Review, 53: 100645. https://doi.org/10.1016/j.cosrev.2024.100645

[2] Zhao, X., Wang, L., Zhang, Y., Han, X., Deveci, M., Parmar, M. (2024). A review of convolutional neural networks in computer vision. Artificial Intelligence Review, 57(4): 99. https://doi.org/10.1007/s10462-024-10721-6

[3] Liu, Y., Xue, J., Li, D., Zhang, W., Chiew, T.K., Xu, Z. (2024). Image recognition based on lightweight convolutional neural network: Recent advances. Image and Vision Computing, 146: 105037. https://doi.org/10.1016/j.imavis.2024.105037

[4] Wu, J., Meng, H., Yan, T., Yuan, M. (2024). Feature-enhanced composite backbone network for object detection. Multimedia Tools and Applications, 83(30): 75387-75405. https://doi.org/10.1007/s11042-024-18448-w

[5] Zhou, T., Liu, F., Ye, X., Guo, Y., Niu, Y., Lu, H. (2024). RNE-DSNet: A re-parameterization neighborhood enhancement-based dual-stream network for CT image recognition. Engineering Science and Technology, an International Journal, 56: 101760. https://doi.org/10.1016/j.jestch.2024.101760

[6] Qiu, J., Lu, X., Wang, X., Chen, C., Chen, Y., Yang, Y. (2024). Research on image recognition of tomato leaf diseases based on improved AlexNet model. Heliyon, 10(13): e33555. https://doi.org/10.1016/j.heliyon.2024.e33555

[7] Chintalapati, B., Precht, A., Hanra, S., Laufer, R., Liwicki, M., Eickhoff, J. (2025). Opportunities and challenges of on-board AI-based image recognition for small satellite earth observation missions. Advances in Space Research, 75(9): 6734-6751. https://doi.org/10.1016/j.asr.2024.03.053

[8] Zhang, H., Li, M., Miao, D., Pedrycz, W., Wang, Z., Jiang, M. (2023). Construction of a feature enhancement network for small object detection. Pattern Recognition, 143: 109801. https://doi.org/10.1016/j.patcog.2023.109801

[9] Lee, S.H., Bae, S.H. (2023). AFI-GAN: Improving feature interpolation of feature pyramid networks via adversarial training for object detection. Pattern Recognition, 138: 109365. https://doi.org/10.1016/j.patcog.2023.109365

[10] Chagnon, J., Hagenbuchner, M., Tsoi, A.C., Scarselli, F. (2024). On the effects of recursive convolutional layers in convolutional neural networks. Neurocomputing, 591: 127767. https://doi.org/10.1016/j.neucom.2024.127767

[11] Yan, T., Shen, S.L., Zhou, A. (2025). Data augmentation-assisted muck image recognition during shield tunnelling. Underground Space, 21: 370-383. https://doi.org/10.1016/j.undsp.2024.10.001

[12] Das, M., Naskar, A., Mitra, P., Basu, B. (2024). Shallow quantum neural networks (SQNNs) with application to crack identification. Applied Intelligence, 54(2): 1247-1262. https://doi.org/10.1007/s10489-023-05192-1

[13] Farina, M., Magri, L., Menapace, W., Ricci, E., Golyanik, V., Arrigoni, F. (2023). Quantum multi-model fitting. In 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, pp. 13640-13649. https://doi.org/10.1109/CVPR52729.2023.01311

[14] Mahargya, I.L., Shidik, G.F., Affandy, Pujiono, Rustad, S. (2025). A systematic literature review of quantum object detection and recognition: Research trend, datasets, topics and methods. Intelligent Systems with Applications, 26: 200499. https://doi.org/10.1016/j.iswa.2025.200499

[15] Syed, M., Garcia, P. (2023). A hybrid quantum-classical machine learning approach to vision sensor data analysis in aerospace applications. In 2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC), Barcelona, Spain, pp. 1-7. https://doi.org/10.1109/DASC58513.2023.10311313

[16] Dou, T., Zhang, G., Cui, W. (2023). Efficient quantum feature extraction for CNN-based learning. Journal of the Franklin Institute, 360(11): 7438-7456. https://doi.org/10.1016/j.jfranklin.2023.06.003

[17] Crooks, G.E. (2019). Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition. arXiv preprint arXiv:1905.13311. https://doi.org/10.48550/arXiv.1905.13311

[18] Farhi, E., Goldstone, J., Gutmann, S. (2014). A quantum approximate optimization algorithm. arXiv preprint arXiv:1411.4028. https://doi.org/10.48550/arXiv.1411.4028

[19] Mitarai, K., Negoro, M., Kitagawa, M., Fujii, K. (2018). Quantum circuit learning. Physical Review A, 98(3): 032309. https://doi.org/10.1103/PhysRevA.98.032309

[20] Hubregtsen, T., Pichlmeier, J., Stecher, P., Bertels, K. (2021). Evaluation of parameterized quantum circuits: On the relation between classification accuracy, expressibility, and entangling capability. Quantum Machine Intelligence, 3(1): 9. https://doi.org/10.1007/s42484-021-00038-w

[21] Delowar, K.E., Uddin, M.B., Khaliluzzaman, M., Rabbi, R.I., Hossen, M.J., Hossen, M.M. (2025). PolyNet: A self-attention based CNN model for classifying the colon polyp from colonoscopy image. Informatics in Medicine Unlocked, 56: 101654. https://doi.org/10.1016/j.imu.2025.101654

[22] Jumaili, M.L.F., Sonuc, E. (2025). An attention-based CNN framework for Alzheimer's disease staging with multi-technique XAI visualization. Computers, Materials and Continua, 83(2): 2947-2969. https://doi.org/10.32604/cmc.2025.062719

[23] Shaari, F.N., Nasir, A.S.A., Mustafa, W.A., Ahmed, W.A.W., Sukor, A.S.A. (2025). Attention-enhanced hybrid CNN–LSTM network with self-adaptive CBAM for COVID-19 diagnosis. Array, 26: 100424. https://doi.org/10.1016/j.array.2025.100424

[24] Shen, X., Liu, Z., Qin, W., et al. (2025). Accurate machine vision identification of GCHD symptom using a self-attention-based CNN model with adaptive fish separation. Smart Agricultural Technology, 11: 100871. https://doi.org/10.1016/j.atech.2025.100871

[25] Chen, F., Zhao, Q., Feng, L., Chen, C., Lin, Y., Lin, J. (2025). Quantum mixed-state self-attention network. Neural Networks, 185: 107123. https://doi.org/10.1016/j.neunet.2025.107123

[26] Li, G., Zhao, X., Wang, X. (2024). Quantum self-attention neural networks for text classification. Science China Information Sciences, 67(4): 142501. https://doi.org/10.1007/s11432-023-3879-7

[27] Shi, J., Zhao, R.X., Wang, W., Zhang, S., Li, X. (2025). QSAN: A near-term achievable quantum self-attention network. IEEE Transactions on Neural Networks and Learning Systems, 36(8): 13995-14008. https://doi.org/10.1109/TNNLS.2024.3504828

[28] Shi, S., Wang, Z., Li, J., et al. (2023). A natural NISQ model of quantum self-attention mechanism. arXiv preprint arXiv:2305.15680. https://doi.org/10.48550/arXiv.2305.15680

[29] Zhao, R.X., Shi, J., Li, X. (2024). QKSAN: A quantum kernel self-attention network. IEEE Transactions on Pattern Analysis and Machine Intelligence, 46(12): 10184-10195. https://doi.org/10.1109/TPAMI.2024.3434974

[30] Wu, Q.S., Liu, W.J., Li, X.R., Su, Z., Lei, J. (2025). A multi-topology quantum convolutional neural network with qubit-measurement attention for image classification. Engineering Applications of Artificial Intelligence, 160: 111705. https://doi.org/10.1016/j.engappai.2025.111705

[31] Pandey, P., Mandal, S. (2026). A hybrid quantum–classical convolutional neural network with a quantum attention mechanism for skin cancer. Scientific Reports, 16: 1639. https://doi.org/10.1038/s41598-025-31122-x

[32] Chen, Y.X. (2024). A novel image classification framework based on variational quantum algorithms. Quantum Information Processing, 23: 362. https://doi.org/10.1007/s11128-024-04566-9

[33] Schuld, M., Bergholm, V., Gogolin, C., Izaac, J., Killoran, N. (2019). Evaluating analytic gradients on quantum hardware. Physical Review A, 99(3): 032331. https://doi.org/10.1103/PhysRevA.99.032331

[34] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11): 2278-2324. https://doi.org/10.1109/5.726791

[35] Xiao, H., Rasul, K., Vollgraf, R. (2017). Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747. https://doi.org/10.48550/arXiv.1708.07747

[36] Cerezo, M., Sone, A., Volkoff, T., Cincio, L., Coles, P.J. (2021). Cost function dependent barren plateaus in shallow parametrized quantum circuits. Nature Communications, 12(1): 1791. https://doi.org/10.1038/s41467-021-21728-w

[37] McClean, J.R., Boixo, S., Smelyanskiy, V.N., Babbush, R., Neven, H. (2018). Barren plateaus in quantum neural network training landscapes. Nature Communications, 9(1): 4812. https://doi.org/10.1038/s41467-018-07090-4

[38] Pesah, A., Cerezo, M., Wang, S., Volkoff, T., Sornborger, A.T., Coles, P.J. (2021). Absence of barren plateaus in quantum convolutional neural networks. Physical Review X, 11(4): 041011. https://doi.org/10.1103/PhysRevX.11.041011

[39] Sim, S., Johnson, P.D., Aspuru-Guzik, A. (2019). Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. Advanced Quantum Technologies, 2(12): 1900070. https://doi.org/10.1002/qute.201900070