Check for updates

Research Article
Open Access

MJ Cell Press

MJ Engineering Sciences

Available at: https://mjcellpress.com/article/mjes07/

MJ Engineering Sciences

Quaternion Convolutional Neural Networks for Multi-Class Classification of Cocoa Beans in Non-Standard Imaging Conditions

Ismael Boumsoumouna Kaoke[1,2*], Ousman Boukar[1,2], Esther Ngah[3], Robert Germain Beka[4], David Libouga Li Gwet [1,5], Laurent Bitjoka[1,2]

[1] Laboratory of Energy, Signal, Imaging and Automation, University of Ngaoundéré, P.O. Box 455, Ngaoundéré, Cameroon

[2] Department of Electrical, Energy, and Automation Engineering, National School of Agro-Industrial Sciences, University of Ngaoundéré, P.O. Box 455, Ngaoundéré, Cameroon

[3] Laboratory of Biochemistry and Food Technology, University of Ngaoundéré, P.O. Box 454, Ngaoundéré, Cameroon

[4] Department of Food Science and Nutrition, National School of Agro-Industrial Sciences, University of Ngaoundéré, P.O. Box 455, Ngaoundéré, Cameroon

[5] School of Chemical Engineering and Mineral Industries (EGCIM), P.O. Box 454, University of Ngaoundere, Cameroon

[*]Corresponding author: ismaelkaoke@gmail.com

Key words

Cocoa Beans, Cut Test, Computer vision, Deep Learning, Quaternion Convolutional Neural Network.

Abstract

Visual inspection of cocoa beans is vital for quality control in the agri-food industry, yet traditional methods such as the cut test remain subjective and labor-intensive. Conventional Machine Learning (ML) and Deep Learning (DL) models show good performance in binary classification but often struggle with multi-class tasks and non-standard image conditions.This study introduces a Quaternion Convolutional Neural Network (QCNN) designed to capture inter-channel correlations in color images within both RGB and CIE XYZ spaces. The model was trained and validated on 1,788 cocoa bean images divided into six quality classes, collected under uncontrolled real-world conditions. Data were split into 80% for training and 20% for testing while maintaining class balance. Experimental results show that the QCNN outperforms MobileNetV2 and ResNet50, achieving accuracies of 91% in RGB and 93% in CIE XYZ spaces. Compact and efficient, the QCNN (2.7 MB) processes each image in 12 ms, demonstrating robustness and practicality for automated cocoa bean classification under variable imaging conditions.

Received:23.09.2025 Accepted: 13.10.2025 Published online: 05.11.2025

How to cite this article: Kaoke, I. B., Boukar, O., Ngah, E., Beka, R. G., Li Gwet, D. L., & Bitjoka, L. (2025). *Quaternion Convolutional Neural Networks for multi-class classification of cocoa beans in non-standard imaging conditions. MJ Engineering Sciences. 1*(2), 94–117. https://doi.org/10.63156/mjes07

1. Introduction

Cocoa (*Theobroma cacao*) represents one of the world's most valuable agricultural commodities, appreciated for its distinctive organoleptic properties (cocoa flavor, chocolate aroma) and nutritional value (Alvarado et al., 2023). More than 70 % of global production originates from West Africa, where smallholder farmers play a dominant role. Côte d'Ivoire and Ghana remain the leading producers, followed by Nigeria and Cameroon (Alvarado et al., 2023; Aprotosoaie et al., 2016). Cocoa bean quality is traditionally evaluated through a combination of analytical approaches categorized as physical (color, size, shape) (Mite-Baidal et al., 2019), sensory (aroma, flavor) (Putri et al., 2024), and chemical (pH, free sugars, titratable acidity) analyses (Aprotosoaie et al., 2016). These evaluations strongly influence both market value and consumer acceptance (Levai et al., 2015; Moussoyi Moundanga et al., 2024). Among them, physical assessment remains essential because it allows rapid, practical evaluation of representative samples. This includes visual inspection (mold, insect damage, color) (Forte et al., 2022), moisture measurement, weight estimation, and dimensional analysis (de Jesus Silva et al., 2024; Sandoval et al., 2019).

The cut test, standardized by ISO 2451:2017, remains the reference method for visual assessment. It consists of cutting a representative sample (typically 300 beans) to examine cotyledon color and internal structure. Based on the proportion of defective beans, batches are graded as Grade I, II, or below-grade. Beans are further classified according to fermentation status (unfermented/slaty, partially fermented/violet, well-fermented/brown), visible defects (mold, insect infestation), flatness (absence of cotyledons), or other anomalies (Levai et al., 2015). Despite its extensive use, the cut test suffers from two major drawbacks: (i) slow, labor-intensive manual counting and (ii) subjective evaluation of batch homogeneity (Nguyen et al., 2022; Quelal-Vásconez et al., 2020). To overcome these limitations, automated inspection systems—particularly computer-vision-based approaches—have attracted increasing attention, combining digital imaging with algorithmic analysis.

Computer vision has proven effective in quality control across several agricultural value chains, including cocoa, coffee, fruits, cereals, and oilseeds (Velesaca et al., 2021), offering superior speed, accuracy, and reproducibility (Sood & Singh, 2021). Nevertheless, shape recognition and visual analysis remain challenging because of the structural complexity of biological samples. Two principal paradigms dominate the literature: Machine Learning (ML), which relies on handcrafted feature extraction (Carbonell et al., 1983; Carvalho et al., 2019), and Deep Learning (DL), which automatically learns hierarchical features from data (Guo et al., 2018; Rangel et al., 2024).

Numerous studies have applied ML and DL to cocoa bean inspection. ML models using colorimetric or textural descriptors have achieved high performance in binary classification—up to 100 % when differentiating fermented from unfermented beans (Basri et al., 2024) or good-quality from defective beans (Basri et al., 2025). However, their accuracy declines markedly in multi-class scenarios (e.g., whole, broken, split, damaged, fermented, unfermented, moldy beans), often falling to 65 % (Adhitya et al., 2020). Moreover, ML models are sensitive to image-acquisition conditions (lighting, noise) (Pal et al., 2024) and require extensive handcrafted features, limiting scalability. Although DL methods automate feature extraction and enhance performance, they still degrade under complex multi-class conditions. For instance, Jean et al. (2022) achieved 98.32 % accuracy across three classes using a CNN–SVM hybrid, whereas Essah et al. (2022) reported only 45 % accuracy across six classes using VGG-16. Furthermore, most existing datasets are acquired under standardized, controlled lighting, typically within light boxes.

The limited capacity of conventional ML and DL methods to manage visual variability and complex multi-class classification underscores the need for more robust architectures capable of exploiting multidimensional, correlated information within color images. To address this gap, the present study proposes and evaluates a Quaternion Convolutional Neural Network (QCNN) that leverages quaternionic representation of color channels to: (i) achieve higher classification accuracy than conventional CNNs such as ResNet50 and MobileNetV2; (ii) reduce misclassification between visually similar categories, notably moldy and infested beans; and (iii) provide a compact architecture suitable for embedded implementation.

The major contributions of this study are summarized as follows:

1. First reported application of QCNNs for cocoa bean quality inspection, addressing a key gap in agri-food computer vision.
2. Quaternionic representation of RGB and CIE XYZ color channels, enhancing discriminative feature learning under uncontrolled imaging conditions.
3. Experimental validation demonstrating that QCNNs combine predictive robustness with computational efficiency, enabling practical embedded deployment.

This work contributes to the modernization of the cocoa value chain and extends to broader domains such as precision agriculture, industrial food-quality monitoring, and low-resource AI implementation. The remainder of this paper is organized as follows: Section 1 presents the theoretical foundations of quaternion algebra and neural networks; Section 2 describes materials and methods; Section 3 reports and discusses experimental results in comparison with state-of-the-art techniques; and Section 4 concludes with perspectives for future research.

2. Quaternion-Valued Neural Networks

This section provides an overview of Quaternion-Valued Neural Networks (QNNs), beginning with the mathematical foundations of quaternion algebra and progressing toward the architecture of the Quaternion Multilayer Perceptron (QMLP). The QMLP framework is analyzed in terms of its forward propagation, backpropagation, and differentiation processes, which are rigorously formulated using the Generalized Hamiltonian Real (GHR) calculus. Subsequently, the section introduces the Quaternion Convolutional Neural Network (QCNN), emphasizing its principal architectural components, namely quaternion convolution, quaternion pooling, and quaternion batch normalization.

2.1. Quaternions

Quaternions, first introduced by William Rowan Hamilton in 1844, constitute a non-commutative extension of complex numbers. A quaternion $q \in \mathbb{H}$ (the set of quaternions) can be expressed as: $q = a + bi + cj + dk$,where $a, b, c, d \in \mathbb{R}$ and i, j, k are imaginary units. These units obey the multiplication rules: $i^2 = j^2 = k^2 = ijk = -1$, $ij = k, jk = i, ki = j$.

The fundamental operations in $\mathbb{H}$ are defined as follows (Equations 1–7):

For two quaternions: $q = a_q + b_q i + c_q j + d_q k$ and $p = a_p + b_p i + c_p j + d_p k$

- Addition or subtraction of p and q is defined as:

$$p \pm q = (a_p \pm a_q) + (b_p \pm b_q)i + (c_p \pm c_q)j + (d_p \pm d_q)k \quad (1)$$

- Conjugate of q is defined as:

$$q^* = a_q - b_q i - c_q j - d_q k \quad (2)$$

- Norm of q is defined as:

$$\|q\| = \sqrt{q q^*} = \sqrt{{a_q}^2 + {b_q}^2 + {c_q}^2 + {d_q}^2} \quad (3)$$

- Hamilton product is defined as:

$$q \otimes p = (a_q a_p - b_q b_p - c_q c_p - d_q d_p) + (a_q b_p + b_q a_p + c_q d_p - d_q c_p)i$$
$$+(a_q c_p - b_q d_p + c_q a_p + d_q b_p)j + (a_q d_p + b_q c_p - c_q b_p + d_q a_p)k \quad (4)$$

- Element-wise (Hadamard) product of p and q is defined as:

$$q \odot p = a_q a_p + b_q b_p i + c_q c_p j + d_q d_p k \quad (5)$$

- Quaternion involution: quaternion derivatives are particularly important for self-inverse quaternionic applications, also referred to as involutions. The quaternion involution is defined as (Ell & Sangwine, 2007):

$$q^{\mu} = \mu q \mu^{-1} = \mu q \mu^{*} = -\mu q \mu \quad (6)$$

where μ is a pure unit quaternion (i.e., a quaternion with norm $\|\mu\|$=1 and inverse $\mu^{*} = -\mu$). For any $\mu \in \mathbb{H}$, it is possible to define a generalized orthogonal quaternion basis $\{1, i^{\mu}, j^{\mu}, k^{\mu}\}$, which satisfies the following properties: $i^{\mu} i^{\mu} = j^{\mu} j^{\mu} = k^{\mu} k^{\mu} = i^{\mu} j^{\mu} k^{\mu} = -1$

- Partial derivative for a quaternion-valued function $f(q)$, $q = a + bi + cj + dk$, is defined as (Nitta, 1995; Parcollet, Ravanelli, et al., 2018):

$$\frac{\partial f}{\partial q} = \frac{\partial f}{\partial a} + \frac{\partial f}{\partial b} i + \frac{\partial f}{\partial c} j + \frac{\partial f}{\partial d} k \quad (7)$$

These mathematical principles form the foundation for the construction of Quaternion-Valued Neural Networks (QNNs). The integration of quaternions into neural architectures dates back to the pioneering works of *Arena et al.* (1994) and *Nitta* (1995), who demonstrated their potential for modeling inter-channel correlations. In recent years, Quaternion Neural Networks (QNNs), and particularly Quaternion Convolutional Neural Networks (QCNNs), have achieved state-of-the-art results in various domains, including RGB color image classification (Yin *et al.*, 2019), image denoising (Miao *et al.*, 2024), and audio recognition (Parcollet, Zhang *et al.*, 2018).

2.2. Quaternion-Valued Multilayer Perceptron

These mathematical principles provide the theoretical foundation for the development of Quaternion-Valued Neural Networks (QNNs). The integration of quaternions into neural architectures traces back to the pioneering studies of Arena *et al.* (1994) and Nitta (1995), who demonstrated their effectiveness in capturing and modeling inter-channel correlations. In recent years, QNNs, particularly Quaternion Convolutional Neural Networks (QCNNs), have achieved state-of-the-art performance across multiple domains, including RGB image classification (Yin *et al.*, 2019), image denoising (Miao *et al.*, 2024), and audio recognition (Parcollet, Zhang *et al.*, 2018).

Notation

- M : Number of layers in the network
- l : Index of the layer, 1,…, M
- N_l : Number of neurons in the l^{th} layer
- n : Index of the neuron in a given layer
- $x_n^{(l)} = a_{x_n^{(l)}} + b_{x_n^{(l)}} i + c_{x_n^{(l)}} j + d_{x_n^{(l)}} k$: output of the n^{th}neuron in the l^{th} layer.
- $w_{nm}^{(l)}$: Quaternion weight connecting the n^{th} neuron in the l^{th} layer to the m^{th} neuron in the $(l-1)^{th}$ layer.
- $\theta_n^{(l)}$: Quaternion bias term of the n^{th} neuron in the l^{th} layer

2.2.1. Forward Propagation

Forward propagation, also referred to as the forward pass, consists of propagating the input data through the successive layers of the network to obtain the output. In the quaternion-valued case, the process is analogous to real-valued neural networks, except that the inputs, weights, and biases are represented as quaternions. For $l = 1, \dots, M$ and $n = 1, \dots, N_l$

Formally, the output of the n^{th} neuron in the l^{th} layer is given by:

$$s_n^{(l)} = \sum_{m=1}^{N_{l-1}} w_{nm}^{(l)} \otimes x_m^{(l-1)} + \theta_n^{(l)}, \tag{8}$$

$$x_n^{(l)} = \sigma\left(s_n^{(l)}\right) = \sigma(a_{s_n^{(l)}}) + \sigma(b_{s_n^{(l)}})i + \sigma(c_{s_n^{(l)}})j + \sigma(d_{s_n^{(l)}})k\,, \tag{9}$$

where

$$\sigma(\cdot) = \sigma(\cdot) + \sigma(\cdot)i + \sigma(\cdot)j + \sigma(\cdot)k \tag{10}$$

represents the split activation function, in which $\sigma(\cdot)$ is a real-valued activation function (e.g., ReLU, tanh, sigmoid) applied independently to each component (a, b, c, d) of the quaternion. Moreover, the pseudo-derivative $\sigma(\cdot)$ ˙ is given by:

$$\dot{\sigma}(\cdot) = \dot{\sigma}(\cdot) + \dot{\sigma}(\cdot)i + \dot{\sigma}(\cdot)j + \dot{\sigma}(\cdot)k \tag{11}$$

Finally, the loss function E, which quantifies the discrepancy between the predicted output $x_n^{(M)}$ and the target value y_n of the $n^{\text{ème}}$neuron in the final layer M, is defined as the mean squared error (MSE) computed over all neurons in this layer:

$$E = \frac{1}{2}\sum_{n=1}^{N_M}\left(y_n - x_n^{(M)}\right)^2 = \frac{1}{2}\sum_{n=1}^{N_M}\left[\begin{matrix}\left(a_{y_n} - a_{x_n^{(M)}}\right)^2 + \left(b_{y_n} - b_{x_n^{(M)}}\right)^2 i + \\ \left(c_{y_n} - c_{x_n^{(M)}}\right)^2 j + \left(d_{y_n} - d_{x_n^{(M)}}\right)^2 k\end{matrix}\right] \tag{12}$$

2.2.2. Error Backpropagation

Backpropagation is the algorithm used to compute the gradient of the loss function with respect to the network parameters (weights and biases), thereby enabling learning through the gradient descent algorithm (Rumelhart et al., 1986). To perform the backpropagation step, the pseudo-gradient error must first be defined as follows

$$\frac{\partial E}{\partial w_{nm}} = \frac{\partial E}{\partial a_{nm}} + \frac{\partial E}{\partial b_{nm}}i + \frac{\partial E}{\partial c_{nm}}j + \frac{\partial E}{\partial d_{nm}}k, \tag{13}$$

where the partial derivatives of the loss function are computed component-wise for each quaternion weight and each bias value in the network. The backpropagation rules for each layer $l = 1, \dots, M$ and each neuron $n = 1, \dots, N_l$ are given by:

$$\delta_n^{(l)} = \begin{cases}\left(y_n - x_n^{(M)}\right), & l = M \\ \dot{\sigma}\left(s_n^{(l)}\right) \odot \sum_{h=1}^{N_{l+1}} w_{hn}^{*(l+1)} \otimes \delta_n^{(l+1)}, & l \neq M\end{cases} \tag{14}$$

Thus, the update rules for the weight and the bias are given by:

$$w_{nm}^{(l)} = w_{nm}^{(l)} + \eta . \delta_n^{(l)} \otimes x_m^{*(l-1)}, \tag{15}$$

$$\theta_n^{(l)} = \theta_n^{(l)} + \eta . \delta_n^{(l)}. \tag{16}$$

where η denotes the learning rate. In matrix–vector form, the update rules can be expressed as follows:

$$\boldsymbol{\delta}^{(l)} = \begin{cases} \boldsymbol{e} = \left(\boldsymbol{y} - \boldsymbol{X}^{(M)}\right), & l = M \\ \dot{\sigma}\left(\boldsymbol{s}^{(l)}\right) \odot \left[\left(\boldsymbol{W}^{(l+1)}\right)^H \times \left(\boldsymbol{\delta}^{(l+1)}\right)\right], & l \neq M \end{cases} \tag{17}$$

with the weight updates given by:

$$\boldsymbol{W}^{(l)} = \boldsymbol{W}^{(l)} + \eta . \boldsymbol{\delta}^{(l)} \times (\boldsymbol{X}^{(l-1)})^H, \tag{18}$$

$$\boldsymbol{\theta}^{(l)} = \boldsymbol{\theta}^{(l)} + \eta . \boldsymbol{\delta}^{(l)}. \tag{19}$$

In matrix form, the symbol x represents the matrix product constructed from element-wise quaternion multiplications in $\mathbb{H}$, and the operator $(\cdot)^H$ represents the Hermitian transpose.

2.2.3. Generalized Hamiltonian Real (GHR) Calculus

The Generalized Hamiltonian Real (GHR) calculus, introduced by Xu *et al.* (2015), provides a framework for defining quaternion derivatives for real-valued, non-analytic functions. It restores fundamental rules such as the product rule, chain rule, mean value theorem, and Taylor expansion by exploiting quaternionic rotations within an orthogonal system (Zhu *et al.*, 2018). The rules of GHR calculus utilize involutions (Equation (6)) and quaternion rotations to establish a consistent set of differentiation rules specifically tailored to quaternions (Ell & Sangwine, 2007). This approach simplifies backpropagation in quaternion-valued neural networks by enabling gradients to be directly computed in $\mathbb{H}$ (Xu & Mandic, 2014).

Let $= a + bi + cj + dk \in \mathbb{H}$, and$f(q)$ be a function $f: \mathbb{H} \to \mathbb{R}$. The left GHR derivatives of f are defined as:

$$\frac{\partial f}{\partial q^{\mu}} = \frac{1}{4}\left(\frac{\partial f}{\partial a} - \frac{\partial f}{\partial b} i^{\mu} - \frac{\partial f}{\partial c} j^{\mu} - \frac{\partial f}{\partial d} k^{\mu}\right) \tag{20}$$

$$\frac{\partial f}{\partial q^{\mu^*}} = \frac{1}{4}\left(\frac{\partial f}{\partial a} + \frac{\partial f}{\partial b} i^{\mu} + \frac{\partial f}{\partial c} j^{\mu} + \frac{\partial f}{\partial d} k^{\mu}\right) \tag{21}$$

$$\nabla_{w^*} E = -\frac{1}{2} \sum_{\mu \epsilon \{1,i,j,k\}} (\boldsymbol{E}_{\boldsymbol{q}^{\mu}}^H \boldsymbol{e})^{\mu}, \tag{22}$$

where $\boldsymbol{e}$ denotes the error vector $\boldsymbol{y} - \boldsymbol{X}^{(M)}$ from Equation (17), and the operator $(\cdot)^H$ represents the Hermitian transpose of the quaternion-valued error vector. This gradient can be expressed layer by layer using the same notation as in the previous backpropagation algorithms.

$$\boldsymbol{\delta}^{(l)} = \begin{cases} \boldsymbol{e} = \left(\boldsymbol{y} - \boldsymbol{X}^{(M)}\right), & l = M \\ \left[\left(\boldsymbol{W}^{(l+1)} \times \bar{\sigma}\left(\boldsymbol{s}^{(l+1)}\right)\right)^H \times \left(\boldsymbol{\delta}^{(l+1)}\right)\right], & l \neq M \end{cases} \tag{23}$$

The weight update rules for the output layer are given as follows:

$$\boldsymbol{W}^{(l)} = \boldsymbol{W}^{(l)} + \eta . \boldsymbol{\delta}^{(l)} \times (\boldsymbol{X}^{(l-1)})^H, \tag{24}$$

$$\boldsymbol{\theta}^{(l)} = \boldsymbol{\theta}^{(l)} + \eta . \boldsymbol{\delta}^{(l)}, \tag{25}$$

whereas the weight updates for the hidden layers are given by:

$$\boldsymbol{W}^{(l)} = \boldsymbol{W}^{(l)} + \eta. \sum_{\mu\in\{1,i,j,k\}} (\boldsymbol{\delta}^{(l)})^{\mu} \times (\boldsymbol{X}^{(l-1)})^{H}, \tag{26}$$

$$\boldsymbol{\theta}^{(l)} = \boldsymbol{\theta}^{(l)} + \eta. \sum_{\mu\in\{1,i,j,k\}} (\boldsymbol{\delta}^{(l)})^{\mu}. \tag{27}$$

3. Quaternion-Valued Convolutional Neural Networks (QCNNs)

Conventional convolutional neural networks (real-valued CNNs)—such as those popularized by Krizhevsky et al. (2017)—process intra-component and inter-component interactions in an equivalent yet independent manner. More concretely, in standard CNNs applied to RGB color images, the response of the k^{th} filter is obtained through a spatial correlation computed independently on each channel, expressed as:

$$y_k^{(l)}(u,v) = \sum_{c\in\{R,G,B\}}^{C} \sum_{s=0}^{f-1}\sum_{t=0}^{f-1} W_{K,c}^{(l)}(s,t).x_c^{(l-1)}(u+s,v+t) \tag{28}$$

where $x_c^{(l-1)}(u,v)$represents the intensity (activation) on channel c at position (u,v) in layer $(l-1)$, $W_{k,c}^{(l)}(s,t)$ is the coefficient of the k^{th} filter in layer l for channel c, and f is the filter size. The output of the conventional convolution $y_k^{(l)}(u,v)$ is therefore a purely linear, channel-wise separated sum. This limitation is precisely addressed with the advent of quaternion-valued convolutional neural networks (QCNNs) (Gaudet & Maida, 2018).

In quaternion-valued convolutional neural networks (QCNNs), the three-color components are encoded into a single quaternion per pixel, and each filter coefficient is itself quaternion-valued. Denoting by $x^{(l-1)}(u,v) \in \mathbb{H}$ the quaternion activation at position (u,v) in layer $(l-1)$ (e.g., $x(u,v) = a(u,v) + b(u,v)i + c(u,v)j + d(u,v)k$), the quaternion pre-activation of the k^{th} filter in layer l is expressed as:

$$s_k{}^{(l)}(u,v) = \sum_{s=0}^{f-1}\sum_{t=0}^{f-1} w^{(l)}{}_k(s,t) \otimes x^{(l-1)}(u+s,v+t), \tag{29}$$

$$x_k^{(l)}(u,v) = \sigma\left(s_k{}^{(l)}(u,v)\right), \tag{30}$$

where $w^{(l)}{}_k(s,t) \in \mathbb{H}$ is the quaternion element of the kernel at position (s,t), and $\otimes$ denotes the Hamilton product. In Equation (29), the double summation applies only to the spatial coordinates of the filter; inter-channel (R, G, B) relationships no longer require explicit summation, as they are directly modeled by the Hamilton product between the quaternionic elements w and x. After exploring quaternionic convolution, it is essential to investigate how quaternionic subsampling can be integrated in order to optimize dimensionality reduction while preserving inter-channel relationships.

3.1. Quaternion Pooling

Pooling plays a central role in convolutional neural networks by reducing the spatial dimensionality of feature maps while preserving discriminative information. In a quaternionic neural network (QCNN), each entity q represents the output of a quaternionic convolution filter, i.e., a quaternion-valued feature map. A first approach, referred to as independent subsampling, applies conventional pooling operations (average or max) separately to each real component (Gaudet & Maida, 2018; Matsumoto et al., 2022; Parcollet, Zhang, et al., 2018), as expressed in Equation (31). Although simple, this method disregards the correlations between components.

$$QP_{avg}(q) = \left(\frac{1}{n}\sum_{k=1}^{n} a^{(k)}, \frac{1}{n}\sum_{k=1}^{n} b^{(k)}, \frac{1}{n}\sum_{k=1}^{n} c^{(k)}, \frac{1}{n}\sum_{k=1}^{n} d^{(k)}\right) \tag{31}$$

where n denotes the size of the pooling window.

Another approach, referred to as magnitude-based quaternion max-pooling (Muppidi & Radfar, 2021; Yin et al., 2019) (Equation (32)), does not simply select the maximum value per channel. Instead, for each region, it selects the quaternion with the highest magnitude, as defined in Equation (33).

$$\|q\| = \sqrt{a^2 + b^2 + c^2 + d^2}\,, \tag{32}$$

$$QP_{max-mag}(q) = argmax_{q^{(k)}} \|q^{(k)}\|. \tag{33}$$

After applying quaternionic subsampling layers to reduce dimensionality while preserving essential information, it is common to use quaternion batch normalization to further stabilize and accelerate the training of QCNNs.

3.2. Quaternion Batch Normalization

Batch Normalization (BN), introduced by Ioffe and Szegedy (2015), aims to stabilize and accelerate the training of deep networks by normalizing activations within each mini-batch (Bach, 2015; Shen et al., 2020). It re-centers activations around zero and rescales them to have unit variance, thereby improving the convergence of gradient descent. The most widespread approach for normalizing quaternion-valued activations is to apply conventional BN independently to each of the four real components of a quaternion. This strategy, referred to as split-BN, has been implemented in several architectures (Gaudet & Maida, 2018; Parcollet, Zhang, et al., 2018; Rumelhart et al., 1986) . In a quaternionic network (QCNN), each activation is a quaternion $q = a + bi + cj + dk$. The split-BN method applies BN separately to each of the four real components(a, b, c, d) according to the following steps:

- Computation of mini-batch statistics (the mean μ_c and the variance σ^2) (J. Wang et al., 2019):

$$\mu_c = \frac{1}{m}\sum_{k=1}^{m} q^{(k)} = a^k + b^k i + c^k j + d^k k \tag{34}$$

$$\sigma^2 = \frac{1}{m}\sum_{k=1}^{m} (q^{(k)} - \mu_c)(q^{(k)} - \mu_c)^*. \tag{35}$$

where m denotes the mini-batch size.

- The quaternion batch normalization $\mathrm{BN}(\tilde{x}^{(k)})$ is defined as follows:

$$\tilde{x}^{(k)} = \frac{{q_c}^{(k)} - \mu_c}{\sqrt{\sigma^2 + \epsilon}} \tag{36}$$

$$\mathrm{BN}(\tilde{x}^{(k)}) = \gamma\left(\frac{{q_c}^{(k)} - \mu_c}{\sqrt{\sigma^2 + \epsilon}}\right) + \beta \tag{37}$$

where γ is a scalar referred to as the scaling factor, β is a purely imaginary quaternion referred to as the shifting factor, and ϵ is a small nonzero constant. The parameters γ and β are updated during the training of the network.

4. Materials and Methods

This section describes the materials and methods used to develop the QCNN-based multi-class classification model for cocoa bean images, including data acquisition, preprocessing procedures, model architecture, and evaluation metrics. The experiments were conducted on an ASUS ROG laptop equipped with an Intel Core i7-9750H processor (2.6 GHz), 16 GB SSD RAM, running Windows 11, and supported by an NVIDIA GeForce GTX 1650 GPU. The

quaternion-valued convolutional neural network (QCNN) was implemented and trained using the Python programming language with the TensorFlow library.

The methodology followed a pipeline from data acquisition to model evaluation, as illustrated in Figure 1. Image preprocessing involved cropping (zooming), color space conversion (CIE XYZ), resizing, normalization, and quaternionic encoding. The proposed model is a quaternion-valued convolutional neural network (QCNN) that employs GHR calculus for backpropagation. Two benchmark models—ResNet50 (He *et al.*, 2016) and MobileNetV2 (Sandler *et al.*, 2018)—were used with transfer learning to provide a comparative analysis in terms of classification performance and computational efficiency.

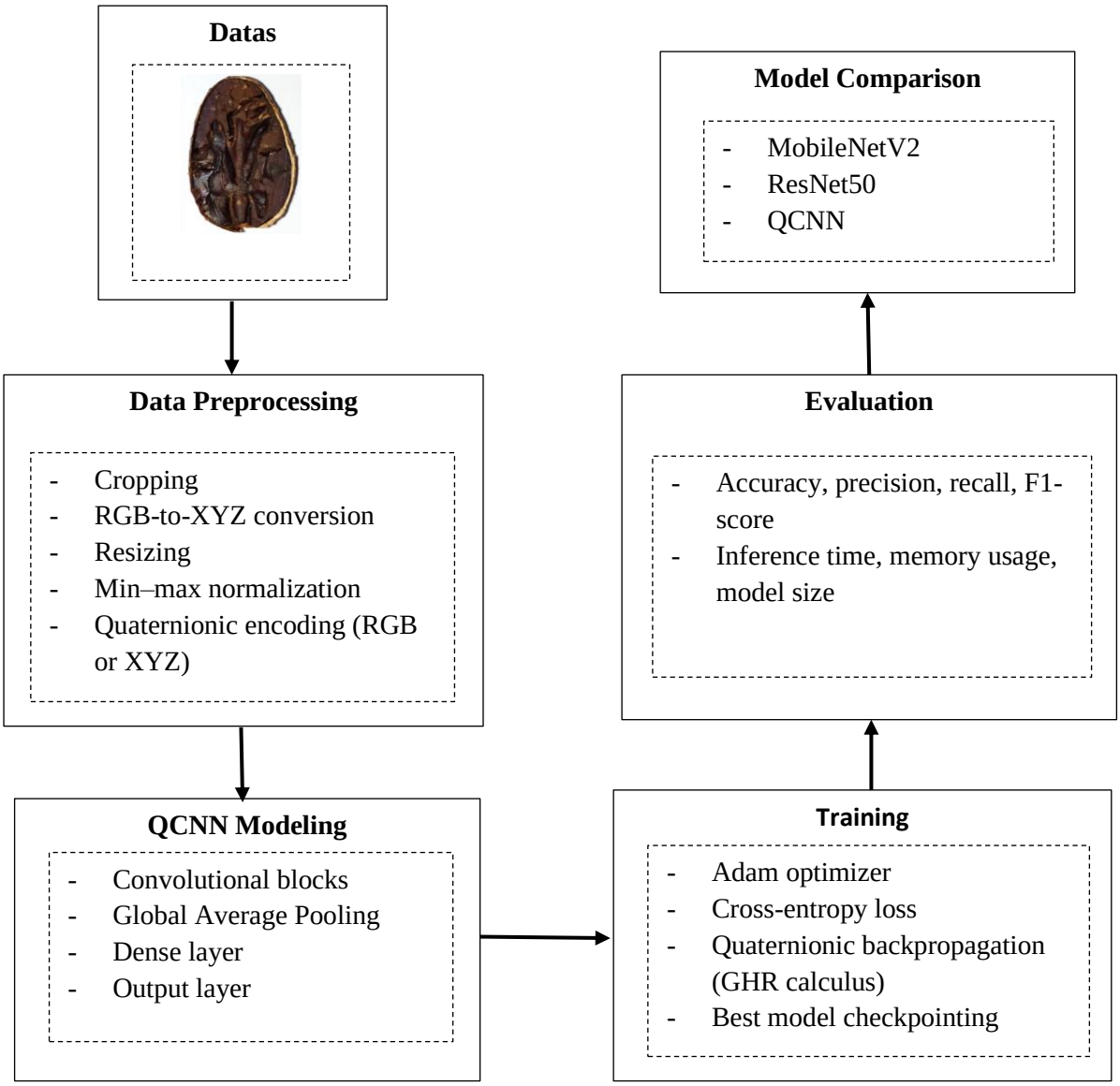

Figure 1: General pipeline of the proposed QCNN model

4.1. Data Acquisition

The cocoa beans used to construct the image dataset were obtained from smallholder cooperatives located in the Centre Region of Cameroon. These were commercial-grade beans derived from a mixture of several botanical varieties: Forastero, Criollo, and Trinitario. Data labeling was performed using the cut test, in accordance with standard quality control procedures, with the assistance of a specialized team from the Cameroon National Cocoa and Coffee Board (NCCB). A total of 1,788 beans were analyzed and classified into six categories: well-fermented, slaty, violet, infested, moldy, and flat (Figure 2).

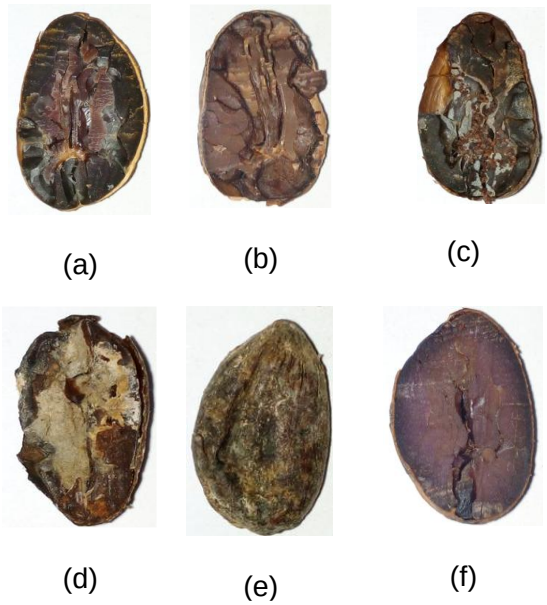

Figure 2: Cocoa bean images labeled by the cut test: (a) slaty, (b) well-fermented, (c) infested, (d) moldy, (e) flat, (f) violet.

Image acquisition was carried out using a Tecno Infinix smartphone equipped with an 8-megapixel camera under natural daylight conditions. The beans were placed on an A4-sized white background, and the capture distance was maintained between 20 and 25 cm to minimize shadow effects and ensure visual uniformity. Images were stored in PNG format within the RGB color space, with resolutions ranging from 1492×3147 to 1843×3076 pixels (Table 1).

Table 1: Distribution of cocoa bean images by class

Class	Number of images	Resolution (pixels)
Slaty	300	1843 × 3076
Well-fermented	300	1843 × 2807
Infested	293	1626 × 3013
Moldy	300	1545 × 3094
Flat	295	1708 × 2931
Violet	300	1492 × 3147
Total	1,788	—

4.2. Preprocessing

Prior to splitting the dataset into training (80%) and testing (20%) subsets, the images were first cropped (zoomed) by 30% to visually center the target bean against the background, thereby reducing the background's influence on the learning of relevant features. After cropping to center the region of interest, the images were converted from the RGB color space to CIE XYZ, which provides a perceptually uniform and device-independent color representation (Mohammadi *et al.*, 2024). The separation of luminance (Y) from chromaticity (X and Z) reduces sensitivity to variations in illumination intensity and color temperature, thus enhancing robustness under non-standard lighting conditions. The dataset was then randomly divided into 80% for training and 20% for testing, maintaining comparable class proportions across both subsets. To accelerate training and prevent gradient explosion, the images were resized to 150 × 150 pixels for the QCNN model and 224 × 224 pixels for real-valued CNNs. Subsequently, a min–max normalization was applied channel-wise, according to Equation (38), to scale each color component to the interval [0, 1] (Pablo Guerra & Cuevas, 2024):

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{38}$$

where X denotes the raw pixel value of the image, and X_{min} and X_{max}represent the minimum and maximum pixel values in the dataset, respectively.

After channel-wise normalization, the images are transformed into four-channel tensors to enable their representation in quaternion space. Specifically, the three colorimetric components from the RGB or CIE XYZ color spaces are projected onto the imaginary parts (i, j, k) of the quaternion, while a fourth component, initialized to zero, is added to constitute the real part (Huang et al., 2023; Miao et al., 2024).This can be formalized as follows: let a color image (RGB) with a resolution $\mathrm{M} \times \mathrm{N}$ pixels, its representation in quaternion space is defined within the quaternion matrix $Q = (q_{nm}) \in \mathbb{H}^{\mathrm{N}\times\mathrm{M}}, 1 \leq \mathrm{m} \leq \mathrm{M}, 1 \leq \mathrm{n} \leq \mathrm{N}$ as follow :

$$Q = 0 + Q_R i + Q_G j + Q_B k \tag{39}$$

where $Q_R, Q_G, Q_B \in \mathbb{R}^{\mathrm{M}\times\mathrm{N}}$ respectively contain the pixel values of the R, G, and B channels.

4.3. Modeling

The models developed for cocoa bean image classification comprise three architectures: QCNN, MobileNetV2, and ResNet50. The MobileNetV2 (Sandler *et al.*, 2018) and ResNet50 (He *et al.*, 2016) architectures were selected for comparison with the proposed QCNN model because they are widely recognized as benchmark models in image classification (Khatri *et al.*, 2025) and offer computational efficiency suitable for embedded deployment on low-resource devices such as Raspberry Pi boards, industrial smart cameras, and mobile devices (Lee *et al.*, 2023; Luo *et al.*, 2020).

4.3.1. Architecture

The proposed model is a quaternion-valued convolutional neural network (QCNN) composed of three convolutional blocks. Each block sequentially integrates a quaternion convolutional layer (Q-Conv2D), quaternion batch normalization (Q-BatchNorm), a LeakyReLU activation applied in split mode (Q-LeakyReLU), quaternion magnitude-based max pooling (Q-Max-Pooling), and a dropout layer with a fixed rate of 0.2. The feature tensors extracted from these blocks are subsequently aggregated using Global Average Pooling (GAP), followed by a fully connected (Dense) layer with 500 neurons, incorporating L2 regularization, a ReLU activation, and an additional dropout rate of 0.3. The network output is produced by a dense layer with a softmax activation, generating a probability distribution over the six target classes.

Within the quaternionic convolutional stack, the number of filters is set to 16, 32, and 64 for the three successive blocks, while the remaining hyperparameters are kept constant. The architecture consists of a total of 226,482 parameters, of which 225,586 are trainable and 896 are non-trainable (Table 2).

Table 2: Architecture of the Quaternion-Valued Convolutional Neural Network

Layer No.	Layer Type	Output Shape	Parameter Size
1	Input Layer	(150, 150, 4)	–
2	Q-Conv2D	(150, 150, 64)	640
3	Q-BatchNorm	(150, 150, 64)	256
4	Q-LeakyReLU	(150, 150, 64)	–
5	Q-MaxPooling	(75, 75, 64)	–
6	Dropout	(75, 75, 64)	–
7	Q-Conv2D	(75, 75, 128)	18,560
8	Q-BatchNorm	(75, 75, 128)	512
9	Q-LeakyReLU	(75, 75, 128)	–

Layer No.	Layer Type	Output Shape	Parameter Size
10	Q-MaxPooling	(37, 37, 128)	–
11	Dropout	(37, 37, 128)	–
12	Q-Conv2D	(37, 37, 256)	73,984
13	Q-BatchNorm	(37, 37, 256)	1,024
14	Q-LeakyReLU	(37, 37, 256)	–
15	Q-MaxPooling	(18, 18, 256)	–
16	Dropout	(18, 18, 256)	–
17	GAP (Global Avg. Pooling)	256	–
18	Dense	500	128,500
19	Dropout	500	–
20	Dense (Softmax)	6	3,006
Total	–	–	226,482

All parameters of the QCNN convolutional block are summarized in Table 3.

Table 3. Parameters of the QCNN Convolutional Block

Layer	Parameters
Q-Conv2D	Kernel size = (3, 3); Stride = (1, 1); Kernel initialization = He initializer
Q-BatchNorm	Momentum = 0.85; Epsilon = 1e-2
Q-MaxPooling	Pooling window size = (2, 2)

In this study, the MobileNetV2 (Sandler *et al.*, 2018) and ResNet50 (He *et al.*, 2016) architectures were employed within a transfer learning framework, with weights initialized from the ImageNet dataset. The base convolutional layers were kept frozen to concentrate training on a fully connected layer comprising 500 neurons, followed by an output layer with a softmax activation. This strategy ensures consistent training conditions across the baseline models and helps mitigate overfitting given the relatively small dataset size (Sun *et al.*, 2022; Y. Wang *et al.*, 2023). The MobileNetV2 model consists of 643,506 trainable parameters and 2,257,984 non-trainable parameters, whereas the ResNet50 model includes 1,027,506 trainable parameters and 23,587,712 non-trainable parameters. In both architectures, a dropout regularization rate of 30% was applied to reduce overfitting. The detailed configurations of both models are summarized in Table 4.

Table 4: Architectures of the MobileNetV2 and ResNet50 Networks

Layer No.	Layer Type	Output Shape	Parameter Size
MobileNetV2			
1	Input Layer	(224, 224, 3)	–
2	MobileNetV2_1.00_224	(7, 7, 1280)	2,257,984
3	GAP (Global Avg. Pooling)	1280	–
4	Dropout	1280	–
5	Dense	500	640,500
6	Dropout	500	–
7	Dense (Softmax)	6	7,686
Total	–	–	2,901,490
ResNet50			
1	Input Layer	(224, 224, 3)	–
2	ResNet50 Backbone	(7, 7, 2048)	23,587,712

Layer No.	Layer Type	Output Shape	Parameter Size
3	GAP (Global Avg. Pooling)	2048	–
4	Dropout	2048	–
5	Dense	500	1,024,500
6	Dropout	500	–
7	Dense (Softmax)	6	3,006
Total	–	–	24,615,218

4.3.2. Training Parameters

The QCNN, MobileNetV2 (Sandler *et al.*, 2018), and ResNet50 (He *et al.*, 2016) models share a common set of training parameters, as summarized in Table 5. However, a key distinction lies in the type of backpropagation employed: the QCNN utilizes quaternion-valued backpropagation based on GHR calculus, whereas MobileNetV2 and ResNet50 apply conventional real-valued backpropagation.

Table 5; Training Parameters of the Models

Parameter	Value
Batch size	8
Epochs	50 ;150
Optimizer	Adam
Learning rate	1e-3

Following Masters and Luschi (2018), a small batch size (m=8m = 8m=8) was adopted to promote better generalization and training stability. The Adam optimizer (Kingma & Ba, 2017) with a learning rate of $1\times10^{-3}$was employed, as it is widely recognized as a standard choice for training deep neural networks. To prevent overfitting and enhance convergence, early stopping was applied by monitoring the validation loss (patience=15, best weight restoration = true). Model checkpoints were saved based on the highest validation accuracy. To further improve robustness under non-standard imaging conditions, on-the-fly data augmentation was implemented, consisting exclusively of contrast adjustments of ±20%.

4.4. Evaluation Metrics

To evaluate the performance of the different classification models, a standard assessment protocol commonly employed in computer vision and agricultural imaging was adopted (Kanakala & Ningappa, 2025). The selected metrics include precision, recall, accuracy, and the F1-score, which collectively provide a comprehensive evaluation of predictive performance. The underlying confusion matrix is constructed from four fundamental indicators: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). These indicators respectively represent the number of correctly identified positive and negative cases, as well as the corresponding misclassifications. From these values, the primary performance metrics are derived, enabling a detailed assessment and objective comparison of the analyzed models.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{40}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{41}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{42}$$

$$\mathrm{F1-score} = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \quad (43)$$

Beyond classification performance, the evaluation of a model must also consider its operational feasibility, particularly in the context of industrial quality control applications. To this end, three key indicators—widely employed in the literature (Canziani *et al.*, 2016; Lin *et al.*, 2020; Ratul *et al.*, 2025)—were selected to characterize the computational efficiency of the analyzed models:

- **Model size (MB):** the memory size of the saved weight file, which reflects model compactness and determines ease of deployment in resource-constrained environments.
- **Memory usage (RAM):** the amount of memory required to perform inference with a given batch size, representing the model's memory footprint under real deployment conditions.
- **Inference time (ms/image):** the average duration required to generate a prediction from an input image—a critical parameter for applications requiring rapid or real-time processing.

These indicators were measured for each architecture (QCNN, MobileNetV2, and ResNet50), together with classical performance metrics (accuracy, precision, recall, and F1-score), to enable a comparative analysis that integrates both predictive performance and computational efficiency. Inference time was measured on an NVIDIA GeForce GTX 1650 GPU with a batch size of eight images; the reported value corresponds to the average processing time per batch and per image, computed over five consecutive runs. Memory usage was monitored using the *psutil* library, which records the total CPU RAM allocated by the TensorFlow process, including model weights, temporary tensors, and background buffers.

5. Results and Discussions

This section presents and discusses the experimental results, beginning with an analysis of overall performance and training stability, followed by an examination of class-wise performance. It further evaluates the computational efficiency of the models and concludes with a comparative analysis against state-of-the-art approaches.

5.1. Overall Performance Analysis and Training Stability

Figure 3 shows the accuracy curves of the ResNet50, MobileNetV2, and QCNN models, evaluated on cocoa bean images in both RGB and CIE XYZ color spaces. Overall, across 50 training iterations, all models converge properly on both the training and testing datasets, demonstrating satisfactory generalization capability. However, some instability is observed during the testing phase, which may indicate sensitivity to inter-class variations and image acquisition conditions. This instability remains relatively moderate for ResNet50 and MobileNetV2 in RGB, owing to their optimized architectures and robustness to visual noise (Arnia *et al.*, 2021). Conversely, it appears more pronounced for the QCNN in RGB, suggesting that this model is slightly more sensitive to illumination changes and background noise. In contrast, transitioning to the CIE XYZ color space improves the stability of the QCNN, indicating that this colorimetric representation facilitates better separation of discriminative features.

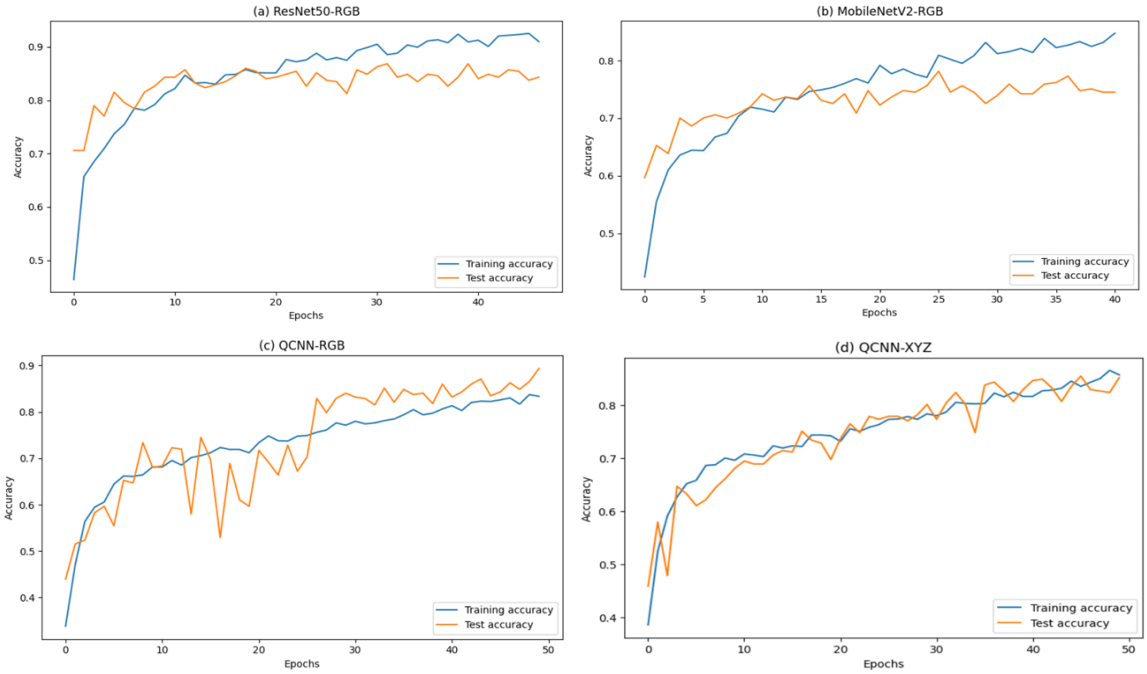

Figure 3: Accuracy curves of the models during training and testing over 50 epochs: (a) ResNet50 on RGB images, (b) MobileNetV2 on RGB images, (c) QCNN on RGB images, and (d) QCNN on XYZ images**.**

During the testing phases, illustrated by the orange curves, the accuracy of the ResNet50 and MobileNetV2 models rapidly exceeds 50% within the initial iterations, reaching respective peaks of 88% at the 13th epoch for ResNet50 and 78% at the 14th epoch for MobileNetV2. However, a gradual divergence of the curves is observed from the 25th epoch onward, indicating increased instability during the generalization phase. In comparison, the QCNN model begins with lower accuracy (below 40%) on both RGB and XYZ images but demonstrates a steady upward trend: it continuously improves its predictions throughout the iterations, showing faster adaptation from the 20th epoch for quaternionic RGB data and from the outset for quaternionic XYZ data. At convergence, the QCNN achieves a maximum accuracy of 89% in RGB and 87% in XYZ, with overall performance comparable to or exceeding that observed during training. This behavior suggests that the QCNN still possesses additional learning capacity and could further enhance its generalization ability with an increased number of training iterations.

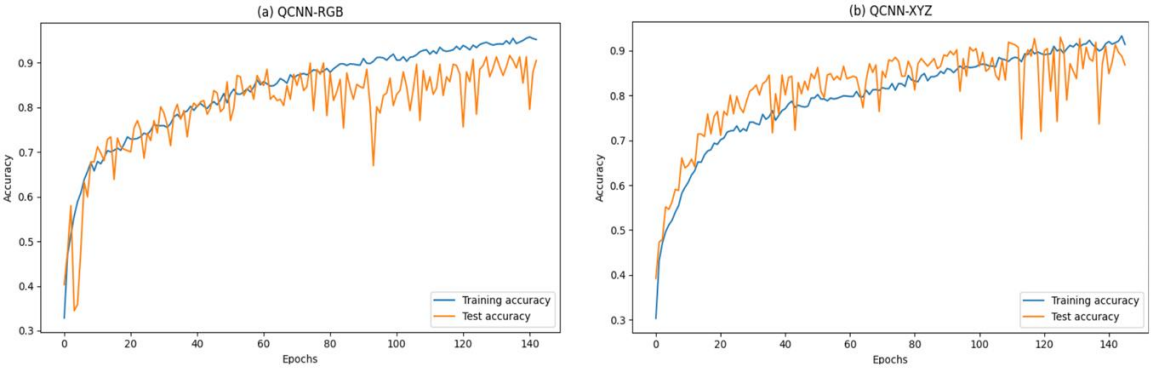

Figure 4: Accuracy curves of the QCNN model during training and testing over 150 epochs: (a) QCNN on RGB images, and (b) QCNN on XYZ images.

After increasing the number of epochs to 150, Figure 4 highlights a significant improvement in the performance of the QCNN model on the test data for both RGB and XYZ images. In the RGB configuration, the model maintains accuracy comparable to, or slightly higher than, that of the training phase up to the 70th epoch, reaching a maximum of 91% at the 80th epoch. However, marked instability appears after the 85th epoch, with accuracy dropping to 65% at the 95th epoch, followed by a gradual recovery, reaching 85% by the 140th epoch. In the XYZ configuration, the

QCNN demonstrates overall superior performance compared with training up to the 110th epoch, achieving a maximum accuracy of 93%. Although a phase of instability is observed around the 120th epoch, the model rapidly readjusts and regains convergence similar to the training curve from the 140th epoch onward. The best-performing checkpoints were therefore selected at the 80th epoch for the RGB model and the 110th epoch for the XYZ model, corresponding to the highest accuracy values observed on the test set during training. These behaviors can be attributed to the quaternionic nature of the network, which more effectively captures inter-channel (RGB/XYZ) correlations, but exhibits a more gradual convergence compared with conventional models (Gaudet & Maida, 2018; Huang *et al.*, 2023).

5.2. Analysis of Model Performance per Class

The performance of the models for each class was assessed using the classification metrics reported in Tables 6–9, derived from the confusion matrix data presented in Figure 5.

5.2.1. ResNet50

The ResNet50 model achieved an overall accuracy of 0.87, with a precision of 0.87, a recall of 0.86, and an F1-score of 0.86. As shown in Table 6, certain classes were classified with high reliability, such as flat beans (precision and recall = 0.98) and well-fermented beans (F1-score = 0.90). However, notable misclassifications occurred in defective beans, particularly for infested beans (F1-score = 0.72) and moldy beans (F1-score = 0.84), reflecting the model's difficulty in discriminating classes with strong visual similarities.

Table 6: Classification performance metrics of ResNet50 on RGB cocoa bean images.

Class	Precision	Recall	F1-score	Number of Images
0 -Slaty	0.93	0.82	0.88	51
1 -Well-fermented	0.91	0.89	0.90	75
2-Infested	0.76	0.69	0.72	55
3-Moldy	0.80	0.88	0.84	59
4-Flat	0.98	0.98	0.98	60
5-Violet	0.83	0.91	0.87	57
Average	0.87	0.86	0.86	357
Accuracy	–	–	0.87	357

5.2.2. MobileNetV2

MobileNetV2 achieved an overall accuracy of 0.87, with a precision of 0.87, a recall of 0.86, and an F1-score of 0.86 on RGB images. These results are slightly inferior compared to those obtained with ResNet50. As shown in Table 7, the model exhibits substantial variability in per-class performance: while flat beans are well classified (F1-score = 0.92), the model demonstrates a limited ability to discriminate infested and moldy beans (F1-score= 0.62 and 0.69, respectively). These findings indicate that MobileNetV2 is more prone to interclass confusion when confronted with high intra-class visual variability.

Table 7: Classification metrics of MobileNetV2 on RGB images of cocoa beans

Class	Precision	Recall	F1-score	Number of images
0 -Slaty	0.96	0.84	0.90	51
1 -Well-fermented	0.76	0.85	0.81	75
2-Infested	0.62	0.62	0.62	55
3-Moldy	0.69	0.69	0.69	59
4-Flat	0.93	0.90	0.92	60

Class	Precision	Recall	F1-score	Number of images
5-Violet	0.77	0.75	0.76	57
Average	0.79	0.78	0.78	357
Overall Accuracy	–	–	0.78	357

5.2.3. QCNN on RGB Images

When applied to RGB images of cocoa beans, the QCNN model demonstrated a marked improvement, achieving an overall accuracy of 0.91, with a precision of 0.91, a recall of 0.91, and an F1-score of 0.91. As reported in Table 8, the slaty, violet, and flat classes achieved particularly high performance (F1-score ≥ 0.93). Although the infested bean class remained slightly lower (F1-score = 0.82), QCNN substantially outperformed conventional CNNs by better exploiting inter-channel correlations, thereby reducing misclassification errors.

Table 8: Classification metrics of QCNN on RGB images of cocoa beans

Class	Precision	Recall	F1-score	Number of images
0 -Slaty	1.00	0.96	0.98	51
1 -Well-fermented	0.91	0.93	0.92	75
2-Infested	0.86	0.78	0.82	55
3-Moldy	0.96	0.80	0.87	59
4-Flat	0.87	1.00	0.93	60
5-Violet	0.90	1.00	0.95	57
Average	0.92	0.91	0.91	357
Overall Accuracy	–	–	0.91	357

3.2.4. QCNN on XYZ Images

The classification of cocoa bean images converted into the CIE XYZ color space using QCNN yielded overall high and stable performance compared with the methods previously discussed in this manuscript, achieving an accuracy of 0.93, with a precision of 0.93, a recall of 0.93, and an F1-score of 0.93. As shown in Table 9, all classes achieved F1-scores above 0.84, with particularly outstanding performance for well-fermented beans (F1-score = 0.96), flat beans (F1-score = 0.95), and violet beans (F1-score = 0.95).

Table 9: Classification metrics of QCNN on XYZ images of cocoa beans

Class	Precision	Recall	F1-score	Number of images
0 -Slaty	0.96	0.96	0.96	51
1 -Well-fermented	0.94	0.99	0.96	75
2-Infested	0.88	0.80	0.84	55
3-Moldy	0.91	0.88	0.90	59
4-Flat	0.91	1.00	0.95	60
5-Violet	0.98	0.93	0.95	57
Average	0.93	0.93	0.93	357
Overall Accuracy	–	–	0.93	357

These results confirm the advantage of combining quaternionic representation with the CIE XYZ color space, which more effectively captures the multidimensional structure and inter-channel correlations of color images. Indeed, the CIE XYZ space, being closer to a photometrically linear representation(Ohno, 2000), reduces sensitivity to lighting variations and thereby enhances class separability, which may explain the improved stability and superior scores achieved by QCNN on XYZ images. These findings are consistent with prior reports in the literature, where

quaternion-valued convolutional neural networks have demonstrated superior efficiency in handling complex color image classification problems (Gaudet & Maida, 2018; Zhu et al., 2018).

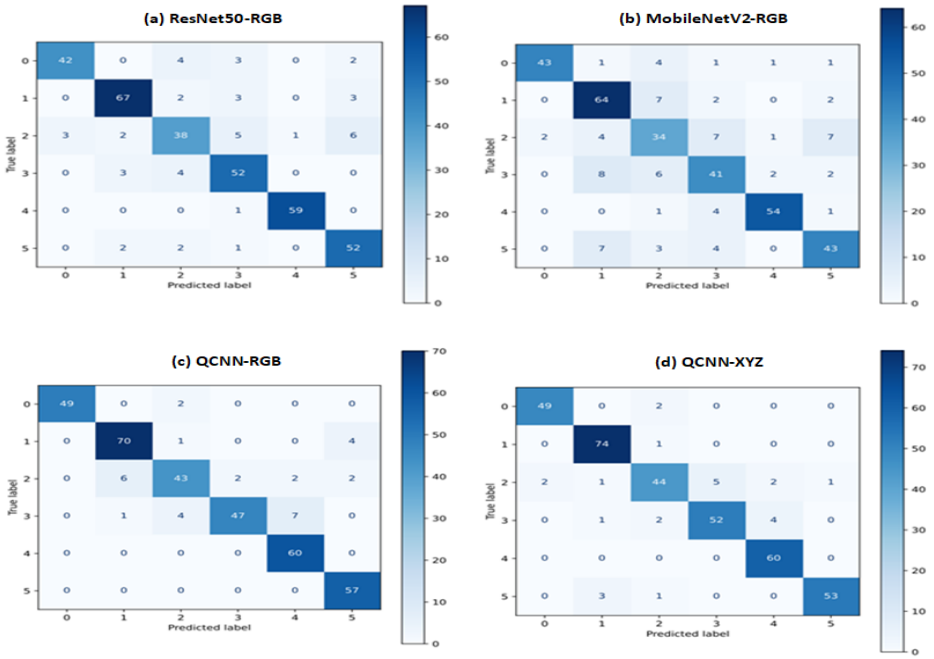

Figure 5: Confusion matrices for cacao bean classification using (a) ResNet50 on RGB images, (b) MobileNetV2 on RGB images, (c) QCNN on RGB images, and (d) QCNN on XYZ images.

5.2.5. Misclassification rates by class and model for cocoa bean

Figure 6 summarizes the dominant misclassification rates across the four evaluated models. ResNet50 (deep blue) demonstrates relatively strong performance in classifying flat and well-fermented beans but exhibits substantial confusion between visually similar categories, with 15% of moldy beans misclassified as infested and 12% of slaty beans mislabeled as moldy. MobileNetV2 (orange-red) shows even greater inter-class confusion, particularly with 20% of infested beans and 18% of moldy beans incorrectly assigned to each other, underscoring its limited ability to distinguish subtle visual differences. In contrast, the QCNN in the RGB configuration (teal green) markedly reduces these errors, lowering moldy–infested confusion to 7% and maintaining misclassification rates for slaty, violet, and flat beans below 5%. The best results are obtained with the QCNN in the XYZ color space (magenta), where misclassification rates remain consistently below 5% across all categories and near-perfect separation is achieved for flat, violet, and well-fermented beans (>95% accuracy). These findings confirm the superiority of quaternionic modeling, particularly when combined with the XYZ color representation.

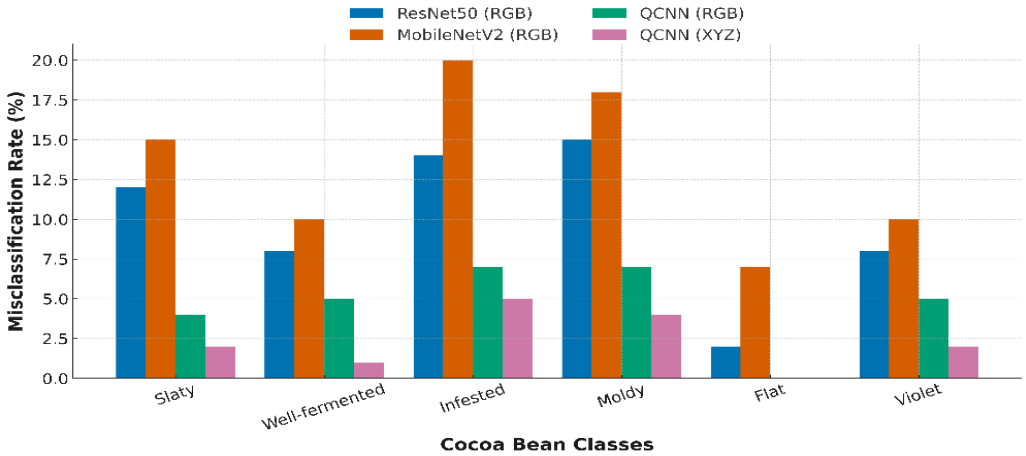

Figure 6: Misclassification rates (%) by class and model for cocoa bean classification.

The comparative analysis of misclassification patterns highlights the practical relevance of adopting quaternion convolutional neural networks (QCNNs) for automated cocoa bean quality inspection. Conventional deep learning architectures such as ResNet50 and MobileNetV2 achieved acceptable overall accuracies; however, their error distributions reveal a critical limitation—high confusion between visually similar defect classes, particularly between infested and moldy beans, reaching up to 20% misclassification. From an industrial standpoint, such errors are especially costly, as they directly affect grading reliability and market value. In contrast, QCNNs exhibited not only higher overall accuracy but also substantially lower misclassification rates, with the QCNN–XYZ configuration reducing all class errors to below 5%. This robustness can be attributed to the quaternionic representation, which effectively captures cross-channel correlations and improves class discrimination under non-standard imaging conditions.

5.3. Computational Efficiency of the Classification Models

The results presented in Table 11 highlight the trade-offs in computational efficiency among ResNet50, MobileNetV2, and QCNN. The reported values correspond to the mean ± standard deviation computed over five independent runs. ResNet50 (RGB) achieves the lowest inference time (10.00 ± 0.15 ms/image), confirming its effectiveness for fast processing. However, this advantage is offset by its very high memory footprint (3641 ± 10 MB) and large model size (102 MB), which significantly limit its deployment on resource-constrained devices. This observation aligns with the findings of He *et al.* (2016), who describe ResNet50 as a robust architecture but one that is computationally expensive in terms of parameters and memory usage. In comparison, MobileNetV2 (RGB) offers a more balanced trade-off, combining reduced storage requirements (16 MB) with fast inference (7.00 ± 0.12 ms/image), making it suitable for lightweight embedded applications. Its compactness results from an optimized architecture that employs depthwise separable convolutions and linear bottlenecks, specifically designed for deployment in resource-limited environments (Sandler *et al.*, 2018).
The QCNN, in both RGB and XYZ configurations, stands out as the most compact architecture (2.7 MB), exhibiting memory consumption approximately 375 MB lower than that of MobileNetV2. Although its inference time is slightly higher (12 ms/image), it remains compatible with near real-time applications. This efficiency stems from the quaternionic representation, which enables weight sharing across components and explicitly models inter-channel correlations, thereby substantially reducing the number of parameters without compromising performance.

Table 11: Computational efficiency of the classification models on cocoa bean images.

Model	Color space	Inference Time (ms/image)	Memory usage (RAM, MB)	Model size (MB)
MobileNetV2	RGB	7.00 ± 0.12	3589 ± 8	16.0
ResNet50	RGB	10.00 ± 0.15	3641 ± 10	102.0
QCNN	RGB	12.36 ± 0.18	3213 ± 7	2.7
QCNN	XYZ	12.83 ± 0.16	3244 ± 6	2.7

Overall, the results in Table 11 emphasize that, unlike ResNet50—whose deployment is constrained by its high memory and model size requirements—MobileNetV2 and QCNN offer compact and efficient alternatives. QCNN combines compactness, computational speed, and predictive robustness, positioning it as a more suitable solution for real-time industrial deployment, especially in resource-constrained environments.

5.4. Comparison of QCNN Performance with the State of the Art

Table 11 presents a comparative performance analysis between previous studies and the proposed approach. Traditional machine learning models based on manually extracted colorimetric or textural descriptors achieve near-perfect accuracy in simple binary classification tasks (100%) (Guo *et al.*, 2018; Rangel *et al.*, 2024), but their performance drops sharply as the number of classes increases (65% for seven classes) (Adhitya *et al.*, 2020). Deep learning methods, such as VGG-16 or certain hybrid CNN architectures, improve robustness yet remain sensitive to

inter-class variability and require large, annotated datasets (Essah *et al.*, 2022; Jean *et al.*, 2022). It should be noted that these results were obtained using datasets different from those employed in the present study, which limits the possibility of a strict comparison. Nevertheless, within our experimental framework, the proposed QCNN outperforms the reference architectures (MobileNetV2 and ResNet50), demonstrating superior generalization ability and higher accuracy in the multi-class classification of cocoa beans.

Table 11: Comparative performance of the proposed QCNN model for cocoa bean classification relative to prior studies.

Study	Dataset	Methodology	Accuracy
Basri et al. (2024)	1,000 images, 2 classes	Texture features + SVM	100%
Basri et al. (2025)	500 images, 2 classes	Texture and color features + SVM	99%
Adhitya et al. (2020)	7,428 images, 7 classes	Texture features + XGBoost	65%
Jean et al. (2022)	3,470 images, 3 classes	CNN + SVM	98%
Essah et al. (2022)	207 images, 6 classes	VGG-16 with transfer learning	45%
QCNN–RGB (proposed)	1,780 beans, 6 classes	Quaternion CNN + RGB channels	91%
QCNN–XYZ (proposed)	1,780 beans, 6 classes	Quaternion CNN + XYZ channels	93%

SVM: support vector machine, XGBoost : extreme gradient boosting, VGG-16: visual geometry group network (16 layers)

Conclusion

This study demonstrated the potential of quaternion convolutional neural networks (QCNNs) as a robust solution for multi-class classification of cocoa beans under non-standard imaging conditions. By embedding RGB and CIE XYZ color channels within a quaternionic representation, the proposed model effectively captured inter-channel correlations, resulting in superior discriminative power compared with conventional deep learning architectures. The QCNN achieved accuracies of 91% (RGB) and 93% (CIE XYZ), outperforming MobileNetV2 and ResNet50, particularly by reducing misclassifications among visually similar classes such as moldy and infested beans. Beyond predictive performance, the QCNN proved highly compact (2.7 MB) and computationally efficient (12 ms/image inference), underscoring its suitability for deployment in real-time, resource-constrained industrial environments. These findings highlight its relevance as a scalable, low-cost, and transferable tool for automated cocoa quality control, with broader implications for agro-food monitoring systems. When benchmarked against prior research, the QCNN demonstrated a clear advancement over both traditional machine learning and conventional CNN-based approaches, exhibiting notable resilience under challenging imaging conditions. Future work should focus on scaling QCNNs to larger datasets, integrating multimodal inputs (e.g., spectral, hyperspectral, or 3D imaging), and extending their application to other agricultural products, thereby reinforcing their role in smart agriculture and precision quality management.

Financial supports

No funds, grants, or other financial support was received for conducting this study or for preparing this manuscript.

Competing Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Adhitya, Y., Prakosa, S. W., Köppen, M., & Leu, J. S. (2020). Feature extraction for cocoa bean digital image classification prediction for smart farming application. Agronomy, 10. https://doi.org/10.3390/agronomy10111642.
2. Alvarado, M. C., Sanchez, P. D. C., & Polongasa, S. G. N. (2023). Emerging rapid and non-destructive techniques for quality and safety evaluation of cacao: Recent advances, challenges, and future trends. Food Production, Processing and Nutrition, 5(1), 40. https://doi.org/10.1186/s43014-023-00157-w.

3. Aprotosoaie, A. C., Luca, S. V., & Miron, A. (2016). Flavor Chemistry of Cocoa and Cocoa Products—An Overview. Comprehensive Reviews in Food Science and Food Safety, 15(1), 73–91. https://doi.org/10.1111/1541-4337.12180.
4. Arena, P., Fortuna, L., Occhipinti, L., & Xibilia, M. G. (1994). Neural networks for quaternion-valued function approximation. Proceedings of IEEE International Symposium on Circuits and Systems-ISCAS'94, 6, 307–310. https://doi.org/ 10.1109/ISCAS.1994.409587.
5. Arnia, F., Saddami, K., & Munadi, K. (2021). DCNet: Noise-Robust Convolutional Neural Networks for Degradation Classification on Ancient Documents. Journal of Imaging, 7(7), 114. https://doi.org/10.3390/jimaging7070114.
6. Bach, F. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. Proc 32nd Int Conf Mach Learn, 37, 448.
7. Basri, B., Indrabayu, I., Achmad, A., & Areni, I. S. (2024). Fermented and Unfermented Cocoa Beans for Quality Identification Using Image Features. JOIV: International Journal on Informatics Visualization, 8(3), 1109–1117. http://dx.doi.org/10.62527/joiv.8.3.2578.
8. Basri, B., Indrabayu, I., Achmad, A., & Areni, I. S. (2025). Cocoa bean quality identification using a computer vision-based color and texture feature extraction. International Journal of Advances in Intelligent Informatics, 11(1), 86–101. https://doi.org/10.26555/ijain.v11i1.1609.
9. Bill, J., Cox, B. A., & Champagne, L. (2023). A comparison of quaternion neural network backpropagation algorithms. Expert Systems with Applications, 232, 120448. https://doi.org/10.1016/j.eswa.2023.120448.
10. Buchholz, S., & Sommer, G. (2008). On Clifford neurons and Clifford multi-layer perceptrons. Neural Networks, 21(7), 925–935. https://doi.org/10.1016/j.neunet.2008.03.004.
11. Canziani, A., Paszke, A., & Culurciello, E. (2016). An Analysis of Deep Neural Network Models for Practical Applications (Version 4). arXiv. https://doi.org/10.48550/ARXIV.1605.07678.
12. Carbonell, J. G., Michalski, R. S., & Mitchell, T. M. (1983). An overview of machine learning. Machine Learning, 3–23. https://doi.org/10.1016/B978-0-08-051054-5.50005-4.
13. Carvalho, T. P., Soares, F. A., Vita, R., Francisco, R. da P., Basto, J. P., & Alcalá, S. G. (2019). A systematic literature review of machine learning methods applied to predictive maintenance. Computers & Industrial Engineering, 137, 106024. https://doi.org/10.1016/j.cie.2019.106024.
14. de Jesus Silva, N. M., de Lima, C. L. S., Dos Santos, R. M., Rogez, H., & de Souza, J. N. S. (2024). Exploring variations in quality parameters of Theobroma cacao L. beans from Eastern Amazonia. Heliyon, 10(21). https://doi.org/10.1016/j.heliyon.2024.e39295.
15. Ell, T. A., & Sangwine, S. J. (2007a). Quaternion involutions and anti-involutions. Computers & Mathematics with Applications, 53(1), 137–143. https://doi.org/10.1016/j.camwa.2006.10.029.
16. Essah, R., Anand, D., & Singh, S. (2022). An intelligent cocoa quality testing framework based on deep learning techniques. Measurement: Sensors, 24, 100466. https://doi.org/10.1016/j.measen.2022.100466.
17. Forte, M., Currò, S., Van de Walle, D., Dewettinck, K., Mirisola, M., Fasolato, L., & Carletti, P. (2022). Quality evaluation of fair-trade cocoa beans from different origins using portable near-infrared spectroscopy (NIRS). Foods, 12(1), 4. https://doi.org/10.3390/foods12010004.
18. Gaudet, C. J., & Maida, A. S. (2018). Deep quaternion networks. 2018 International Joint Conference on Neural Networks (IJCNN), 1–8. https://doi.org/10.1109/IJCNN.2018.8489651.
19. Guo, Y., Liu, Y., Bakker, E. M., Guo, Y., & Lew, M. S. (2018). CNN-RNN: a large-scale hierarchical image classification framework. Multimedia Tools and Applications, 77(8), 10251–10271. https://doi.org/10.1007/s11042-017-5443-x
20. Gyan, J. K., & Bajan, B. (2022). Market analysis on cocoa beans export: The case of Ghana and Cote d'ivoire in West Africa. Journal of Agribusiness and Rural Development, 66(4), 375–384. http://doi.org/10.17306/J.JARD.2022.01646.
21. He, K., Zhang, X., Ren, S., & Sun, J. (2016, June). Deep Residual Learning for Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
22. Huang, C., Li, J., & Gao, G. (2023). Review of quaternion-based color image processing methods. Mathematics, 11(9), 2056.https://doi.org/10.3390/math11092056.
23. Jean, A. K., Diarra, M., Bakary, B., Pierre, G., & Jérôme, A. K. (2022). Application based on Hybrid CNN-SVM and PCASVM Approaches for Classification of Cocoa Beans. International Journal of Advanced Computer Science and Applications (IJACSA), 13(9). https://doi.org//10.14569/IJACSA.2022.0130927.
24. Kanakala, S., & Ningappa, S. (2025). Detection and Classification of Diseases in Multi-Crop Leaves using LSTM and CNN Models (Version 1). arXiv. https://doi.org/10.48550/ARXIV.2505.00741.

25. Khatri, M., Sahoo, M., Sayyad, S., & Sayyad, J. (2025). Adaptive and User-Friendly Framework for Image Classification with Transfer Learning Models. Future Internet, 17(8), 370. https://doi.org/10.3390/fi17080370.
26. Kingma, D. P., & Ba, J. (2017). Adam: A Method for Stochastic Optimization (No. arXiv:1412.6980). arXiv. https://doi.org/10.48550/arXiv.1412.6980.
27. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84–90. https://doi.org/10.1145/3065386.
28. Lee, J., Mukhanov, L., Molahosseini, A. S., Minhas, U., Hua, Y., Martinez Del Rincon, J., Dichev, K., Hong, C.-H., & Vandierendonck, H. (2023). Resource-Efficient Convolutional Networks: A Survey on Model-, Arithmetic-, and Implementation-Level Techniques. ACM Computing Surveys, 55(13s), 1–36. https://doi.org/10.1145/3587095.
29. Levai, L. D., Meriki, H. D., Adiobo, A., Awa-Mengi, S., Akoachere, J.-F. T. K., & Titanji, V. P. (2015). Postharvest practices and farmers' perception of cocoa bean quality in Cameroon. Agriculture & Food Security, 4(1), 28. https://doi.org/10.1186/s40066-015-0047-z.
30. Lin, J., Chen, W.-M., Lin, Y., Cohn, J., Gan, C., & Han, S. (2020). MCUNet: Tiny Deep Learning on IoT Devices (No. arXiv:2007.10319). arXiv. https://doi.org/10.48550/arXiv.2007.10319.
31. Luo, C., He, X., Zhan, J., Wang, L., Gao, W., & Dai, J. (2020). Comparison and Benchmarking of AI Models and Frameworks on Mobile Devices (Version 1). arXiv. https://doi.org/10.48550/ARXIV.2005.05085.
32. Masters, D., & Luschi, C. (2018). Revisiting Small Batch Training for Deep Neural Networks. arXiv. https://doi.org/10.48550/ARXIV.1804.07612.
33. Matsumoto, Y., Natsuaki, R., & Hirose, A. (2022). Full-learning rotational quaternion convolutional neural networks and confluence of differently represented data for PolSAR land classification. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 15, 2914–2928. https://doi.org/10.1109/JSTARS.2022.3164431.
34. Miao, J., Kou, K. I., Yang, Y., Yang, L., & Han, J. (2024). Quaternion matrix completion using untrained quaternion convolutional neural network for color image inpainting. Signal Processing, 221, 109504. https://doi.org/10.1016/j.sigpro.2024.109504.
35. Mite-Baidal, K., Solís-Avilés, E., Martínez-Carriel, T., Marcillo-Plaza, A., Cruz-Ibarra, E., & Baque-Bustamante, W. (2019). Analysis of Computer Vision Algorithms to Determine the Quality of Fermented Cocoa (Theobroma Cacao): Systematic Literature Review. In R. https://doi.org/10.1007/978-3-030-10728-4_9.
36. Valencia-García, G. Alcaraz-Mármol, J. D. Cioppo-Morstadt, N. Vera-Lucio, & M. Bucaram-Leverone (Eds), ICT for Agriculture and Environment (Vol. 901, pp. 79–87). Springer International Publishing. https://doi.org/10.1007/978-3-030-10728-4_9.
37. Mohammadi, V., Ansari, K., Gouton, P., & Attig, H. (2024). Developing a New Method of Transformation for Obtaining XYZ Color Values from RGB Images for Agricultural Applications. Sensors, 24(23), 7728.https://doi.org/10.3390/s24237728.
38. Moussoyi Moundanga, S., Petit, J., Ndangui, C. B., Scher, J., & Nzikou, J.-M. (2024). Impact of cocoa variety on merchant quality and physicochemical characteristics of raw cocoa beans and roasted cocoa mass. Discover Food, 4(1), 115. https://doi.org/10.1007/s44187-024-00188-3.
39. Muppidi, A., & Radfar, M. (2021). Speech emotion recognition using quaternion convolutional neural networks. ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 6309–6313. https://doi.org/10.1109/ICASSP39728.2021.9414248.
40. Nguyen, D. T., Pissard, A., Pierna, J. A. F., Rogez, H., Souza, J., Dortu, F., Goel, S., Hernandez, Y., & Baeten, V. (2022). A method for non-destructive determination of cocoa bean fermentation levels based on terahertz hyperspectral imaging. International Journal of Food Microbiology, 365, 109537. https://doi.org/10.1016/j.ijfoodmicro.2022.109537.
41. Nitta, T. (1995). A quaternary version of the back-propagation algorithm. Proceedings of ICNN'95-International Conference on Neural Networks, 5, 2753–2756. https://doi.org/10.1109/ICNN.1995.488166.
42. Ohno, Y. (2000). CIE Fundamentals for Color Measurements. NIP & Digital Fabrication Conference, 16(1), 540–545. https://doi.org/10.2352/ISSN.2169-4451.2000.16.1.art00033_2.
43. Pablo Guerra, J., & Cuevas, F. (2024). Application of Digital Image Processing Techniques for Agriculture: A Review. In F. Cuevas, P. L. Mazzeo, & A. Bruno (Eds), Digital Image Processing—Latest Advances and Applications. IntechOpen. https://doi.org/10.5772/intechopen.1004767.
44. Pal, C., Das, S., Akuli, A., Adhikari, S. K., & Dey, A. (2024). Cocoa-Net: Performance Analysis on Classification of Cocoa Beans Using Structural Image Feature. Informatica, 48(12). https://doi.org/10.31449/inf.v48i12.5762.

45. Parcollet, T., Ravanelli, M., Morchid, M., Linarès, G., Trabelsi, C., De Mori, R., & Bengio, Y. (2018). Quaternion recurrent neural networks. https://doi.org/10.48550/arXiv.1806.04418.
46. Parcollet, T., Zhang, Y., Morchid, M., Trabelsi, C., Linarès, G., De Mori, R., & Bengio, Y. (2018). Quaternion convolutional neural networks for end-to-end automatic speech recognition. https://doi.org/10.48550/arXiv.1806.07789.
47. Putri, D. N., De Steur, H., Juvinal, J. G., Gellynck, X., & Schouteten, J. J. (2024). Sensory attributes of fine flavor cocoa beans and chocolate: A systematic literature review. Journal of Food Science, 89(4), 1917–1943. https://doi.org/10.1111/1750-3841.17006.
48. Quelal-Vásconez, M. A., Lerma-García, M. J., Pérez-Esteve, É., Talens, P., & Barat, J. M. (2020). Roadmap of cocoa quality and authenticity control in the industry: A review of conventional and alternative methods. Comprehensive Reviews in Food Science and Food Safety, 19(2), 448–478. https://doi.org/10.1111/1541-4337.12522.
49. Rangel, G., Cuevas-Tello, J. C., Nunez-Varela, J., Puente, C., & Silva-Trujillo, A. G. (2024). A survey on convolutional neural networks and their performance limitations in image recognition tasks. Journal of Sensors, 2024(1), 2797320. https://doi.org/10.1155/2024/2797320.
50. Ratul, I. J., Zhou, Y., & Yang, K. (2025). Accelerating Deep Learning Inference: A Comparative Analysis of Modern Acceleration Frameworks. Electronics, 14(15), 2977. https://doi.org/10.3390/electronics14152977.
51. Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533–536. https://doi.org/10.1038/323533a0.
52. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018, June). MobileNetV2: Inverted Residuals and Linear Bottlenecks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). arXiv:1801.04381v4 [cs.CV] 21 Mar 2019
53. Sandoval, A. J., Barreiro, J. A., De Sousa, A., Valera, D., & Müller, A. J. (2019). Determination of the physical properties of fermented and dried Venezuelan Trinitrio cocoa beans (Theobroma cacao). Revista Técnica de La Facultad de Ingeniería, Universidad Del Zulia, 42(2), 50–56.
54. Shen, W., Zhang, B., Huang, S., Wei, Z., & Zhang, Q. (2020). 3d-rotation-equivariant quaternion neural networks. European Conference on Computer Vision, 531–547. https://doi.org/10.1007/978-3-030-58565-5_32.
55. Sood, S., & Singh, H. (2021). Computer vision and machine learning based approaches for food security: A review. Multimedia Tools and Applications, 80(18), 27973–27999. https://doi.org/10.1007/s11042-021-11036-2.
56. Sun, Y., Chen, Q., He, X., Wang, J., Feng, H., Han, J., Ding, E., Cheng, J., Li, Z., & Wang, J. (2022). Singular Value Fine-tuning: Few-shot Segmentation requires Few-parameters Fine-tuning (Version 2). arXiv. https://doi.org/10.48550/ARXIV.2206.06122.
57. Velesaca, H. O., Suárez, P. L., Mira, R., & Sappa, A. D. (2021). Computer vision based food grain classification: A comprehensive survey. Computers and Electronics in Agriculture, 187, 106287. https://doi.org/10.1016/j.compag.2021.106287.
58. Wang, J., Li, T., Luo, X., Shi, Y.-Q., & Jha, S. Kr. (2019). Identifying Computer Generated Images Based on Quaternion Central Moments in Color Quaternion Wavelet Domain. IEEE Transactions on Circuits and Systems for Video Technology, 29(9), 2775–2785. https://doi.org/10.1109/TCSVT.2018.2867786.
59. Wang, Y., Sun, D., Chen, K., Lai, F., & Chowdhury, M. (2023). Egeria: Efficient DNN Training with Knowledge-Guided Layer Freezing (No. arXiv:2201.06227). https://doi.org/10.48550/arXiv.2201.06227.
60. Wessel, M., & Quist-Wessel, P. F. (2015). Cocoa production in West Africa, a review and analysis of recent developments. NJAS: Wageningen Journal of Life Sciences, 74(1), 1–7. https://doi.org/10.1016/j.njas.2015.09.001.
61. Xu, D., Jahanchahi, C., Took, C. C., & Mandic, D. P. (2015). Enabling quaternion derivatives: The generalized HR calculus. Royal Society Open Science, 2(8), 150255. https://doi.org/10.1098/rsos.150255.
62. Xu, D., & Mandic, D. P. (2014). Quaternion gradient and hessian. https://doi.org/10.1109/TNNLS.2015.2440473.
63. Yin, Q., Wang, J., Luo, X., Zhai, J., Jha, S. K., & Shi, Y.-Q. (2019). Quaternion convolutional neural network for color image classification and forensics. IEEE Access, 7, 20293–20301. https://doi.org/ 10.1109/ACCESS.2019.2897000.
64. Zhu, X., Xu, Y., Xu, H., & Chen, C. (2018). Quaternion convolutional neural networks. Proceedings of the European Conference on Computer Vision (ECCV), 631–647. https://doi.org/ 10.48550/arXiv.1903.00658.