UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Visión Electrónica
https://doi.org/10.14483/issn.2248-4728

VISIÓN ELECTRÓNICA

A RESEARCH VISION

Smart System for Ripeness Detection in Blackberry Fruits

Sistema Inteligente de Detección de Madurez en Frutos de Mora

Rafael Augusto Núñez Rodríguez[1], Elkin Itamar Velasco Sánchez[2], Karen Julieth Camargo Flórez[2], Oscar David Rangel Jiménez[3]

INFORMACIÓN DEL ARTÍCULO

Historia del artículo:
Enviado: 01/01/2025
Recibido: 01/02/2025
Aceptado: 05/05/2025

Keywords:
Convolutional Neural Network
TensorFlow Lite
Blackberry Classification
Embedded Systems
Deep Learning
Ripeness Detection.

Palabras clave:
Red Neuronal Convolucional
Tensorflow Lite
Clasificación de Mora
Sistemas Embebidos
Aprendizaje Profundo
Detección de Madurez

ABSTRACT

This paper presents the development of an algorithm aimed at enhancing the classification process of blackberry fruits, which are traditionally harvested manually based on the experience of the workers. The algorithm leverages a VGG-16 convolutional neural network to recognize the ripeness level of blackberry fruits. To achieve this, a comprehensive image dataset was created to train the neural network. The network was designed, trained, and optimized based on specific performance parameters and then implemented on a Raspberry Pi using the TensorFlow Lite framework. Field tests were conducted on actual blackberry crops to evaluate the algorithm's performance, showing a high accuracy rate of 96.66 % The results highlight the potential of the proposed system to significantly improve the efficiency and accuracy of the fruit classification process, reducing the reliance on manual methods and enabling more consistent harvesting practices.

RESUMEN

Este trabajo presenta el desarrollo de un algoritmo destinado a mejorar el proceso de clasificación de los frutos de mora, que tradicionalmente se recolectan manualmente basándose en la experiencia de los trabajadores. El algoritmo aprovecha una red neuronal convolucional VGG-16 para reconocer el nivel de madurez de los frutos de mora. Para ello, se creó un amplio conjunto de datos de imágenes para entrenar la red neuronal. La red se diseñó, entrenó y optimizó en función de parámetros de rendimiento específicos y, a continuación, se implementó en una Raspberry Pi utilizando el marco TensorFlow Lite®. Se realizaron pruebas de campo en cultivos reales de mora para evaluar el rendimiento del algoritmo, que mostró una elevada tasa de precisión del 96,66 %. Los resultados destacan el potencial del sistema propuesto para mejorar los resultados

1. Electronic Engineer, Unidades Tecnológicas de Santander, Colombia. MSc. in Electronic Engineering, Universidad Pontificia Bolivariana, Colombia. Associate Professor, Unidades Tecnológicas de Santander, Colombia. E-mail: rrodriguez@correo.uts.edu.co. ORCID: https://orcid.org/0000-0001-6775-776X
2. Electronic Engineer, Unidades Tecnológicas de Santander, Colombia. MSc. candidate in Electrical Engineering, Universidade Federal do Paraná, Brazil. Associate Professor, Unidades Tecnológicas de Santander, Colombia. E-mail: eitamarvelasco@correo.uts.edu.co. ORCID: https://orcid.org/0009-0001-2832-3544
3. Electronic Engineer, Unidades Tecnológicas de Santander, Colombia. E-mail: ccamargo@correo.uts.edu.co. ORCID: https://orcid.org/0000-0001-6786-776X
4. Electronic Engineer, Unidades Tecnológicas de Santander, Colombia. E-mail: odrangel@correo.uts.edu.co. ORCID: https://orcid.org/0000-0001-6786-628X

Cite this article as: R. A. Núñez-Rodríguez, E. I. Vélasco-Sánchez, K. J. Camargo-Flórez, O. D. Rangel-Jiménez, "Smart System for Ripeness Detection in Blackberry Fruits", Visión Electrónica, vol. 19, no. 1, 2025.

destacan el potencial del sistema propuesto para mejorar significativamente la eficiencia y la precisión del proceso de clasificación de frutas, reduciendo la dependencia de los métodos manuales y permitiendo prácticas de cosecha más consistentes.

1. Introduction

Machine vision technologies have been widely used in agriculture to improve the performance of agricultural production units. Current methods for detecting ripeness in blackberry fruits encompass a range of technologies, each with its own set of limitations. One prominent method involves the use of an electronic nose (e-nose) to sense aromatic volatiles released by the fruit, which, when combined with artificial neural networks (ANN), principal components analysis (PCA), and linear discriminant analysis (LDA), can classify ripeness with high precision, achieving 100 % accuracy for blackberries in ANN analysis. However, this method's effectiveness can vary with different berry types, as seen with lower accuracy in whiteberries [1]. Another approach utilizes machine learning and deep learning techniques, which are advantageous due to their ability to process raw data without complex feature engineering. These methods, while effective, require substantial computational resources and large datasets for training, which can be a limitation in practical applications [2]. Additionally, tactile sensors using nanoneedle-patterned polydimethylsiloxane (PDMS) have been developed to assess fruit firmness, a key indicator of ripeness. These sensors are highly sensitive and can be used in robotic applications, but they require recalibration for different fruit types, which can be a constraint [3]. Optical methods, such as hyperspectral imaging and near-infrared spectroscopy, are also employed for nondestructive firmness evaluation. Yet, they face challenges in achieving high accuracy for online sorting due to poor correlations with traditional firmness measures [4]. Furthermore, color-based systems using NFC sensors offer a low-cost solution for ripeness classification. Still, their accuracy varies significantly depending on the fruit type and algorithm used, ranging from 80 % to 93 % [5]. Lastly, advanced detection techniques incorporating artificial intelligence, such as computer vision and electronic nose systems, are being explored for their potential in high-throughput berry processing, though they have not yet been fully integrated into commercial applications [6]. Overall, while these methods provide valuable tools for detecting ripeness, they each have limitations related to accuracy, cost, and the need for specific calibration or computational resources. From the context of a portable embedded system for blackberry fruit ripeness recognition, the following contributions can be proposed:

- Development of a Lightweight Convolutional Neural Network for Real-Time Blackberry Ripeness Detection on Embedded Systems.
- Systematic Evaluation Framework for Optimizing Ripeness Detection Algorithms on Embedded Platforms.

This paper presents descriptive research with a quantitative approach and a systematic observation method since the training process of a convolutional neural network will be analyzed, and the algorithm will be evaluated in terms of accuracy and percentage of hits. First, we propose designing the recognition algorithm that integrates the following phases: image bank collection, digital image processing, digital image transformation, and the creation of a new CNN model from a pre-trained structure. Secondly, the implementation of the previously designed algorithm in an R-Pi embedded system using the TensorFlow Lite framework. Then, the performance of the recognition algorithm is evaluated, different validation scenarios are considered, and the results are presented. Finally, the results of the various phases are presented, along with the conclusions of the work.

2. Materials and Methods

2.1. Design of the recognition algorithm

A blackberry maturation level recognition algorithm using convolutional neural networks is proposed. It consists of four stages: data collection, data processing, data transformation, and creation of a new CNN model.

2.1.1. Image bank compilation

An image bank is built with the specific characteristics of the ripening levels established by the ICONTEC standard according to the ripening table of blackberry fruits [7]. The images were taken from different blackberry crops in the Planada's district in Piedecuesta, Santander.

2.1.2. Digital Image Processing

The first step of data processing consists of reviewing the photographs taken in the previous step and discarding those that are not of good quality, for example, images that, due to their light intensity, distort the natural colors of the Blackberry or that, due to their sharpness do not guarantee a correct classification by the algorithm. Secondly, sections of the initial image belonging to the blackberry fruit are extracted using the cropping and rotation tool to eliminate areas that do not provide relevant information to the algorithm. Finally, the dataset is constructed according to the ripening characteristics. 3000 photographs were categorized into 3 classes or labels of output data: Green, Red, and Purple.

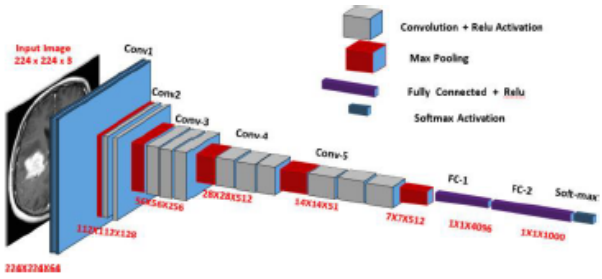
2.1.3. Digital Image Transformation

Once the images have been classified, the CNN is not yet ready to develop the learning transfer algorithm since the photos to be entered do not meet the input parameters required by the network. However, one of the most important adjustments is the size of the images, which must be 224 x 224 pixels since this is the network's default value, according to the kernel of the VGG-16 pre-trained network.

2.1.4. Neural Network Parameterization

This section presents how the CNN model should be adjusted, trained, and evaluated. For this purpose, the pre-trained network VGG-16 is established. According to Subham Tewari, it achieves 92.7 % accuracy in object recognition [8]. The neural network VGG – 16 is structured in Python programming language, so it is necessary to have a code editor to modify the output parameters of the neural network; Figure 1 summarizes the parameters and architecture of the network. The number of classes to work with is defined, considering that the original network has 1000 classes as output. Then, the images selected for training are distributed equally among the classes to which they correspond, and a label is assigned to each data set. To ensure better results when training the algorithm, the data must be organized randomly; likewise, the data are divided into two sets: 80 % of the images are taken for training and 20 % for algorithm validation.

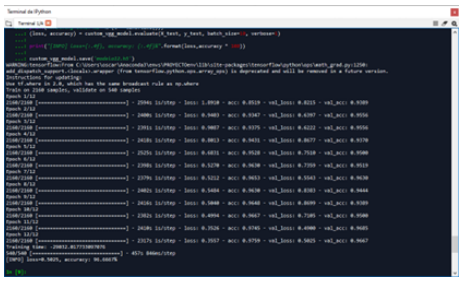
Figure 1: VGG – 16 Architecture



Source: [9].

The learning transfer stage starts by configuring the size of the input images. Subsequently, the pre-trained VGG-16 network is imported from the TensorFlow library. This network allows the input of a new image bank and the modification of the output labels according to the problem to be treated.

Figure 2: CNN Training Results



Source: [Own].

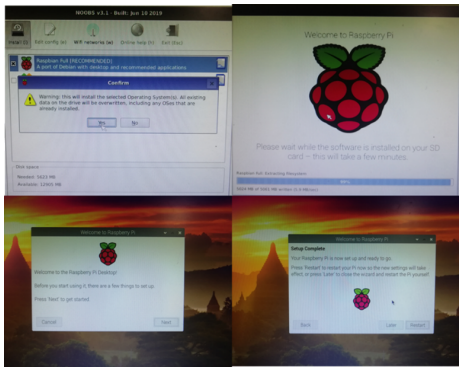
This method is called transfer learning. To apply such learning, the learned parameters of the original network are maintained, and only the output block is extracted to modify them with new labels. Finally, the new neural network architecture is compiled. Once the network architecture has been adjusted, the training stage begins. This consists of training the network with 80 % of the previously processed images. Once the training is finished, the algorithm is validated with 20 % of the remaining images, resulting in the percentage of loss and accuracy as shown in Figure 2. Thus, the matrix of weights adjusted after the training process is stored and then implemented in the R-Pi embedded system.

2.2. Implementation of the algorithm in an embedded system

2.2.1. Development Environment Configuration

To develop this section, an embedded system and a device to visualize the operation of the previously designed algorithm must be acquired. For this project, Raspberry Pi 3 model B was used.

Figure 3: Initial OS Raspberry Pi configurations.



Source: [Own].

First, installing the respective operating system (Raspbian) on an SD storage card is necessary, which can be obtained from the official Raspberry website. Once the installation is complete, specific system parameters are configured, such as setting the language, changing the location, and network configuration, which can be seen in Figure 3. Secondly, to implement the algorithm, the OpenCV library is needed, which allows the development of applications for object recognition, as well as being the main library to meet the proposed objective. Finally, installing the Tensorflow and Keras libraries is necessary since they allow the efficient execution of the code contained in the CNN.

2.2.2. Implementation of CNN in a development environment

Once the algorithm design is obtained, the next step is to test its operation. This section describes how the implementation is carried out on the Raspberry Pi; a graphical interface is also designed with the Python tool "Tkinter" so that the user can visualize the algorithm's operation. Table 1 describes the implementation procedure on the R – Pi.

Table 2: Pseudocode CNN implementation in R–Pi

CNN implementation code in R–Pi	
Start	
1.	Import functions for code development.
2.	Create visual interface elements (titles, buttons, typography).
3.	Create a function for video transmission.
4.	Create a function to take, save, and classify photos (CNN code).
End	

Source: Own

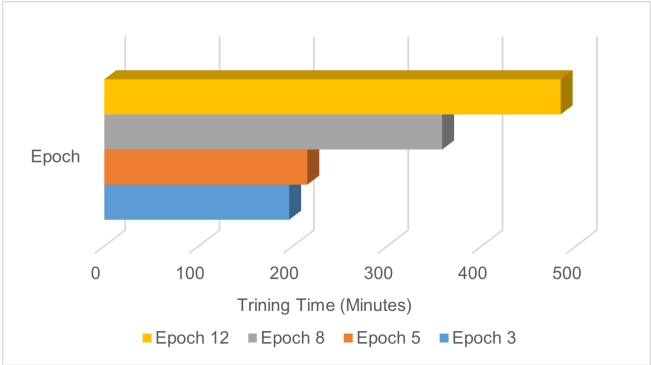
Figure 4: Blackberry crops, Planadas – Santander.

Source: Own.

3. Results

This section presents the training results, showing the time duration for each training performed and its accuracy percentages. As evidenced in Figure 5, increasing 3 or 4 epochs increases the time it takes to train by approximately 25 %.

Figure 5: Training time CNN tests



Source: Own.

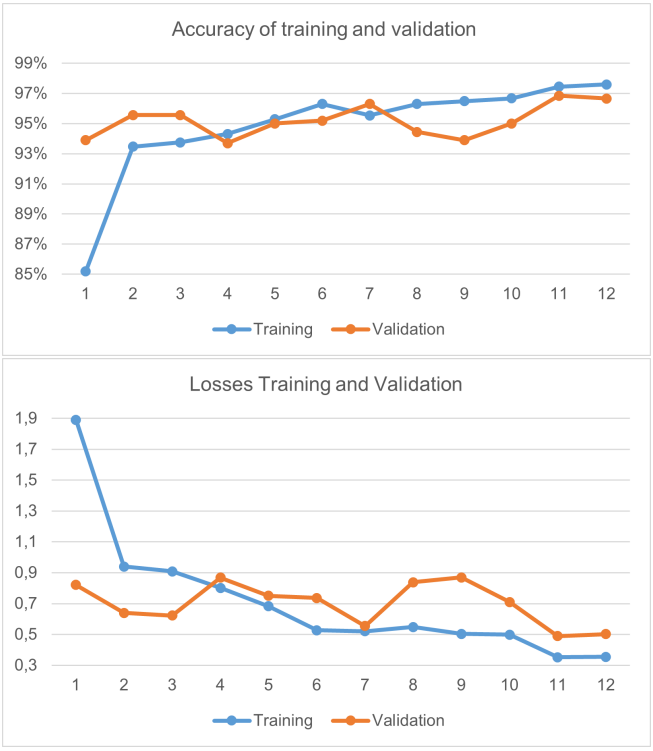
Table 2 shows the training process results compared to the validation concerning model losses and accuracy. As shown in Figure 6 (a), the blue line represents the accuracy achieved during training, and the green line corresponds to the trained model validation. Figure 6 (b) shows the training losses (blue color) and validation losses (orange color) obtained during model training.

Table 3: CNN training results with 12 epochs

Epochs	Time (Minutes)	Training		Validation	
		Data loss	Accuracy	Data Loss	Accuracy
1	43.23	1.8910	85.19 %	0.8215	93.89 %
2	40.00	0.9403	93.47 %	0.6397	95.56 %
3	39.85	0.9087	93.75 %	0.6222	95.56 %
4	40.30	0.8013	94.31 %	0.8677	93.70 %
5	42.08	0.6831	95.28 %	0.7510	95.00 %
6	39.97	0.5270	96.30 %	0.7359	95.19 %
7	39.65	0.5212	95.53 %	0.5543	96.30 %
8	40.03	0.5484	96.30 %	0.8383	94.44 %
9	40.27	0.5040	96.48 %	0.8699	93.89 %
10	39.70	0.4994	96.67 %	0.7105	95.00 %
11	40.17	0.3526	97.45 %	0.4900	96.85 %
12	38.62	0.3557	97.59 %	0.5025	96.66 %
Total Training Time [minutes]					483.87
Loss					0.5025
Final Accuracy					96.66 %

Source: Own

Figure 6: Accuracy and training graphs CNN training with 12 epochs.



Source: Own.

On the other hand, Table 3 shows the accuracy percentages when training the network with different numbers of epochs, 3, 5, 8, and 12, to determine which is more efficient. As can be seen, the accuracy percentage was 96.66 % when training the model with 12 epochs. The same behavior is presented in the neural network training with 3, 5, and 8 epochs.

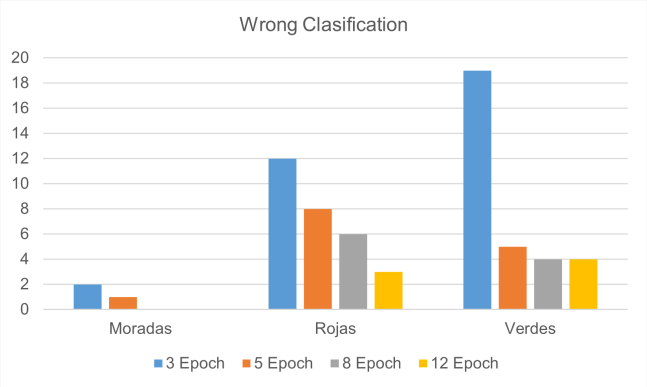
Table 4: Percentage of accuracy CNN training

Number of Epochs	Percentage of Accuracy (Hits)
3	85.55 %
5	95.74 %
8	95.74 %
12	96.66 %

Source: Own

Considering the most accurate model, the following are the results of the tests described during the algorithm's validation. Initially, 150 images were taken with which the algorithm was validated. As shown in Figure 7, the variables are inversely proportional, i.e., as the epochs increase, the misclassified images decrease significantly.

Figure 7: Number of CPU misclassified images.



Source: Own.

On the other hand, tests were performed directly on the BlackBerry crops with the Raspberry Pi and its respective camera module. Of the 60 images, only 7 were misclassified, as shown in Table 4 shows that the fruit classifier algorithm has an accuracy of approximately 88.3 %.

Table 5: Number of misclassified images in R – Pi

Label	Epoch 12
Purple	1
Reds	4
Greens	2

Source: Own

4. Conclusions

Despite the diverse environmental factors with which the image bank was created, and according to the results described above, it is concluded that the greater the number of epochs in the algorithm's training, the greater its accuracy will be. Implementing GPU is recommended to train the CNN, because any modification in the network parameters can take more than 6 hours to complete the training again. Therefore, it is logical to conclude that the fine tuning of the algorithm is not feasible to carry out in the development of the project. When implementing the algorithm on the board, there were drawbacks related to the Raspberry Pi camera, since it did not have a light filter and did not have an autofocus system, which generated low quality images when capturing the fruit directly on the crops. Thus, in the classification of images, the algorithm yielded an 11.6 % error rate during the test.

References

[1] N. Aghilinategh, M. J. Dalvand, and A. Anvar, "Detection of ripeness grades of berries using an electronic nose," *Food Science & Nutrition*, vol. 8, no. 9, pp. 4919–4928, Sep. 2020, doi: 10.1002/fsn3.1788.

[2] M. Rizzo, M. Marcuzzo, A. Zangari, A. Gasparetto, and A. Albarelli, "Fruit ripeness classification: A survey," *Artificial Intelligence in Agriculture*, vol. 7, pp. 44–57, 2023, doi: 10.1016/j.aiia.2023.02.004.

[3] V. Maharshi, S. Sharma, R. Prajesh, S. Das, A. Agarwal, and B. Mitra, "A Novel Sensor for Fruit Ripeness Estimation Using Lithography Free Approach," *IEEE Sensors Journal*, vol. 22, no. 22, pp. 22192–22199, Nov. 2022, doi: 10.1109/JSEN.2022.3210439.

[4] S. Tian and H. Xu, "Mechanical-based and Optical-based Methods for Nondestructive Evaluation of Fruit Firmness," *Food Reviews International*, vol. 39, no. 7, pp. 4009–4039, Aug. 2023, doi: 10.1080/87559129.2021.2015376.

[5] A. Lazaro, M. Boada, R. Villarino, and D. Girbau, "Color Measurement and Analysis of Fruit with a Battery-Less NFC Sensor," *Sensors*, vol. 19, no. 7, p. 1741, Apr. 2019, doi: 10.3390/s19071741.

[6] D. Wang, M. Zhang, A. S. Mujumdar, and D. Yu, "Advanced Detection Techniques Using Artificial Intelligence in Processing of Berries," *Food Engineering Reviews*, vol. 14, no. 1, pp. 176–199, Mar. 2022, doi: 10.1007/s12393-021-09298-5.

[7] ICONTEC, *Frutas frescas. Mora de castilla. Especificaciones - NTC 4106*. Colombia: Instituto Colombiano de Normas Técnicas y Certificación, 1997.

[8] S. Tewari, "CNN Architecture Series — VGG-16 with implementation (Part I)," *Medium*, 2019. Available: https://medium.com/datadriveninvestor/cnn-architecture-series-vgg-16-with-implementation-part-i-bca79e7db415.

[9] H. M. Rai and K. Chatterjee, "Detection of brain abnormality by a novel Lu-Net deep neural CNN model from MR images," *Machine Learning with Applications*, vol. 2, p. 100004, Dec. 2020, doi: 10.1016/j.mlwa.2020.100004.