Smart Agricultural Technology 4 (2023) 100183

Contents lists available at ScienceDirect

Smart Agricultural Technology

journal homepage: www.journals.elsevier.com/smart-agricultural-technology

Coffee disease classification at the edge using deep learning

João Vitor Yukio Bordin Yamashita, João Paulo R.R. Leite *

Universidade Federal de Itajuba - UNIFEI, Av. BPS, 1303, Itajuba,37500-903, Minas Gerais,Brasil

ARTICLE INFO

Keywords:
Artificial intelligence
Convolutional networks
Edge computing
Deep learning
Coffee diseases

ABSTRACT

Brazil is the world's largest producer and exporter of coffee and the second largest consumer of the beverage. The aim of this study is to embed convolutional networks in a low-cost microcontrolled board to classify coffee leaf diseases *in loco*, without the need for an internet connection. Early identification of diseases in coffee plantations is crucial for productivity and production quality. Two datasets were used, in addition to images taken with the development board itself, totaling more than 6000 images of six different types of diseases. The proposed architectures (cascade and single-stage), when embedded, presented accuracy values around 98% and 96%, respectively, demonstrating their ability to assist in the diagnosis of diseases in coffee farms, especially those managed by producers with less resources.

1. Introduction

Detecting diseases in coffee plantations is a difficult task, considering the diversity of pests that affect them. Usually, the analysis of coffee leaves is done visually by human experts, but it can be time-consuming and expensive. Identifying and treating diseases in their early stages is very important for efficient and high-quality coffee bean production [1], but small coffee producers do not have access to these specialists, which leads them to ineffective production and financial losses. Given these challenges, computerized methods are being developed for the automated detection of coffee diseases.

According to the International Coffee Organization, Brazil is the world's largest coffee producer [2]. In addition, information collected by the CNA (Brazilian Agriculture and Livestock Confederation) confirms the contribution of small properties to coffee production: of all coffee farms in the country, 88% have an area of less than 50 hectares. The majority (69%) have an area between 2 and 20 hectares [3]. In general, managers of these small properties have limited access to the internet and smartphones. Developing a standalone low-power system that can meet the demand for diseases classification on these farms, without the need for a cloud connection or expensive hardware, takes artificial intelligence to the edge, where it can process data locally, reducing latency and costs. By creating such a system, it is possible to increase productivity and broaden access to technology.

Many machine learning (ML) techniques have been used to address the disease classification problem [4–6]. Among them, Deep learning (DL) models, mainly convolutional neural networks (CNN), have been showing better performance and generalization [7] when used in new data. That said, the aim of this study is to embed convolutional networks in a low-cost microcontrolled board to automatically classify diseases on coffee leaves without the need for an internet connection and test it in a real-world scenario. This solution would especially benefit small farmers, who cannot afford more sophisticated resources and human expertise.

Several previous studies inspired our work. In [8], the authors developed a smartphone application for semantic segmentation and classification of disease symptoms using images of coffee leaves. Also, [9] created a multi-stage cascading system to classify disease types based on leaf photos. None of these works, however, have all the features necessary for our proposed goal - some need an internet connection, others were not tested in real coffee farm situations and others were not ready for embedded systems.

The rest of this paper is organized as follows. Section 2 discusses some similar works. Section 3 presents information about the convolutional network architecture used in this work, the development board, datasets, and the logical structure of the proposed model. Section 4 contains the development steps and performance metrics. Section 5 presents the results of the proposed systems and analyzes them. By the end, Section 6 summarizes our results, compare them to our goals and points future directions.

2. Related work

This project consists of three steps: 1) pre-processing, using

* Corresponding author.
E-mail addresses: joao.yamashita@unifei.edu.br (J.V.Y. Bordin Yamashita), joaopaulo@unifei.edu.br (J.P.R.R. Leite).

https://doi.org/10.1016/j.atech.2023.100183
Received 11 October 2022; Received in revised form 9 January 2023; Accepted 9 January 2023
Available online 13 January 2023

techniques such as data augmentation; 2) training the CNN deep learning model, using transfer learning and powerful GPUs; and, finally, 3) deployment, using optimization and quantization techniques to reduce the memory and processing power needed to run the model. All of them have been developed and improved by the scientific community in recent years.

The biggest problem of the CNN for plant disease classification is insufficient data, both in quantity and diversity. Mohanty et al. [10] tested CNN models, such as AlexNet [11] and GoogleLeNet [12], with a database of 54,306 images with 26 classes of diseases and obtained a training accuracy of 99.35%. However, when tested under conditions other than the training images, accuracy dropped to approximately 31%. This can be explained by the fact that the training dataset contains images of leaves in ideal conditions (good lighting, neutral background, single leaf facing up), which demonstrates the importance of a diverse dataset for practical applications. Fuentes et al. [13] aimed to develop a more robust detection model focused on real-world applications; the images were taken *in loco*, with various backgrounds and lighting conditions, various sizes and disease variations. This work performed well on the test dataset, but presented difficulties in classes with high pattern variation, tending to confuse with other classes, which could be improved with higher quality data.

To deal with the lack of quality data, data augmentation techniques can be used. Liu et al. [14] developed a deep learning model that classified four apple leaf diseases. They only had 1053 images. However, using data augmentation techniques such as rotation, mirror symmetry and light disturbance, they expanded their dataset to 13,689 images, and the accuracy improved by 10.83%, which highlights the importance of data augmentation in a scenario with limited resources.

Another problem, especially in the task of leaf disease classification, is the variation in symptoms, which are the result of the interaction of diseases, plant and environment [15]. For example, this variation can be caused by different disease stages [16] and/or multiple diseases affecting the same leaf [15]. Also, similar symptoms can appear in various diseases, which adds to the complexity of disease classification. To solve this problem, the most realist option is to gradually enrich the diversity of the dataset [16], using more data to make it robust to symptom variation.

Many machine learning techniques have been used to address the disease classification problem, *e.g.* support vector machines [4], K-nearest neighbor [5] and random forests [6]. These methods have disadvantages such as low performance [17] and the demand for feature selection, since it is often necessary to assemble a research team just to design and select relevant features from the data [18]. On the other hand, deep learning models, especially convolutional neural networks (CNN), have been used in various applications, from weed detection [19] to pest classification and diseases identification [20]. A CNN automatically extracts features from the raw data and presents better results against the disadvantages of other ML techniques, showing superior performance and generalization [7] when used in new data.

For classification, several CNN models have been developed. AlexNet [11] was developed in 2012 and won the ILSVRC competition [21] in the same year. It contains five convolutional layers and three fully connected layers. After that, deeper networks were developed, such as VGGNet [22]. ResNet [23] won the ImageNet competition in 2015 using residual blocks and shortcut connections. In 2017, the MobileNet [24] architecture was developed, and Google used it for mobile and embedded vision applications. In 2019, Google proposed another architecture, the EfficientNet [25], that uses a compound coefficient to uniformly scale all dimensions: depth, width and resolution.

In addition, many studies have proposed methods to reduce the model size and, consequently, the computational power needed to embed them [26–28]. Frameworks such as TensorFlow Lite converts a pre-trained model in TensorFlow [29] into a special format that can be optimized for speed or storage. It provides techniques such as dynamic range quantization, full integer quantization and Float16 quantization, which can make the model up to 4× smaller and 3× faster.[1]

3. Materials and methods

3.1. MobileNet architecture

The MobileNet network was developed by Google [24] to be used in TensorFlow [29] mobile applications. It aims to maximize performance on devices with limited processing power and memory. This architecture uses a type of convolution called "depth-wise separable convolution", which consists of a depth-wise convolution layer with batch normalization and a ReLU activation function followed by a point-wise convolution with batch normalization and a ReLU activation function, as shown in Table 1. This technique reduces computational costs and the number of parameters of the network [30].

An important network parameter is the "width multiplier", or α, which is used to uniformly reduce the width of each layer. By doing this, the processing power needed and number of parameters are reduced by a factor of approximately α^2. In Table 2, we can observe a MobileNet with $\alpha = 0.5$ uses 26% operations and 30% parameters when comparing with a MobileNet with $\alpha = 1$, which is roughly α^2 ($0.5^2 = 0.25$). We also have ρ, representing the "resolution multiplier" parameter, which controls the model input pixel resolution and, similarly to α, reduces the processing power needed by ρ^2.

The MobileNet network manages to achieve good results when compared to other popular networks, in terms of both performance and number of parameters. When compared to a VGG 16 type network [31] in a benchmark test on the ImageNet dataset [32], the MobileNet network performs similarly, with 32 times fewer parameters, as shown in Table 3.

3.2. Development board

For this work, we chose to use the MaixBiT development board[2], which has, at its core, a K210 processor by Kendryte [33]. This SoC has a 64-bit dual-core processor with RISC-V architecture and a clock rate of 400 MHz. The board also has a Knowledge Processor Unit (KPU), which is a processor used for convolution operations, activation functions, pooling operations and other operations characteristic of convolutional networks. This dedicated hardware for such operations reduces the CPU processing power required, and consequently, reduces the time and power needed to run CNN models.

In [34], the authors compared the performance of the K210 processor with other development boards focused on artificial intelligence such as Google Coral and demonstrated the great potential of this board. Finally, [35] implemented a convolutional neural network to detect belt failures in an industrial context using the MaixBit board.

To protect and facilitate the use of the development board, camera and LCD screen, a case was designed in Autodesk Fusion 360 software and printed on a 3D printer using PLA, as shown in Fig. 1.

Table 1
MobileNet architecture.

3 × 3 Depthwise Conv
Batch Normalization
ReLU
1 × 1 Conv
Batch Normalization
ReLU

[1] https://www.tensorflow.org/lite/performance/post_training_quantization
[2] https://wiki.sipeed.com/soft/maixpy/en/develop_kit_board/maix_bit.html

Table 2
Accuracy, number of operations and number of parameters in relation to α (when tested with ImageNet dataset) [24].

Width Multiplier α	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0	70.6%	569	4.2
0.75	68.4%	325	2.6
0.5	63.7%	149	1.3
0.25	50.6%	41	0.5

Table 3
Performance and number of parameters from different architectures.

Model	ImageNet Accuracy	Million Parameters
1.0 MobileNet-224	70.6%	4.2
GoogleNet	69.8%	6.8
VGG 16	71.5%	138

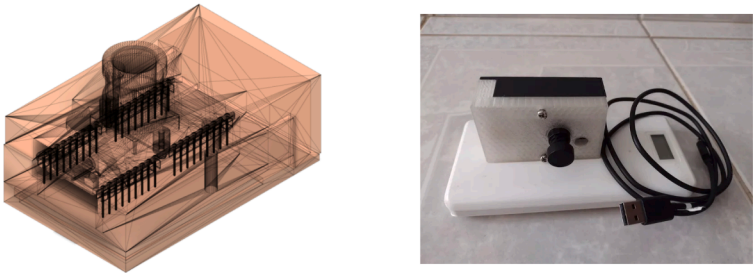


Fig. 1. 3D model and printed prototype.

3.3. Datasets

Two different datasets were used (BRACOL and LiCoLe), containing five classes of diseases that can affect coffee leaves and one of the healthy leaves: Rust (coffee rust), Sooty Molds, Cercospora (cercosporiosis), Phoma and Leaf Miner. We can also group diseases based on their similar characteristics: for example, Cercospora, Phoma and Leaf Miner are characterized by brown spots on the leaf surface.

The BRACOL dataset [36] was generated using cameras from 5 different cell phones and, to take the pictures, leaves were removed from the coffee trees and placed on a white background, with the back of the leaves facing up. It contains 1747 samples with 5 classes: Healthy, Leaf Rust, Cercospora, Phoma and Leaf Miner. The LiCoLe dataset [37] uses a 10.5-MP camera and the photos were taken in an isolated environment with white background. it has 4 classes (Leaf Rust, Cercospora, Sooty Molds and Healthy), totaling 4667 samples.

3.4. Proposed architectures

The first proposed architecture, named "single-stage", consists of a single model to classify an input image into one of six classes: healthy or one of the five diseases mentioned. The system flow is seen in Fig. 2, where the red arrows represent a possible path for classification (as "Leaf Miner").

The second proposal consists of a two-stage architecture, named "cascade model", built in order to increase classification performance. The first stage classifies the input image into one of four classes: Healthy, Rust, Sooty Molds or Brown Spots. If classified as "Brown Spots", there is a second stage which is another classifier that can distinguish between classes of diseases characterized by brown spots [38], namely Cercospora, Leaf Miner and Phoma. This architecture is shown in Fig. 3, where the red arrows also represent a possible path for classification (as "Brown Spots" and then "Leaf Miner").

By using the cascade model, we intend to remove some complexity from the initial stage, improving performance for the Healthy, Rust and Sooty Molds classes. Furthermore, by having a step focused on "Brown Spots", our intention is to improve the model performance for classes that share this characteristic, as this second step is specialized in identifying subtle differences between them.

4. Development

This section introduces the development steps: data preprocessing, model training and evaluation, the process of embedding the model on the development board and the definition of evaluation metrics.

4.1. Data processing and acquisition

Two data sources were used to train the models. The first comes from datasets that were obtained online: BRACOL and LiCoLe. The second data source consists of images produced by the development board itself and saved on an SD card. Their resolution is 224 × 224 and they were captured without removing the leaves from the coffee trees, trying to represent a real use situation. A total of 122 samples were captured on-field and divided in two classes: Leaf Miner and Rust.

4.2. Data augmentation

The data augmentation technique was used to increase the number of samples available for training and avoid overfitting, by adding slightly modified copies of already existing data: rotation, luminosity changes and the blur effect were used, and some examples can be seen in Fig. 4.

4.3. Train, validation and test sets

After the data augmentation process, samples were divided for the single-stage model according to Table 4. Samples for the cascade model were also divided, according to the Tables 5 and 6, where the "Brown Spots" class is a random subgroup of 2100 images from the second stage (Cercospora, Phoma and Leaf Miner). Such subgroup was selected to maintain a balance between the number of samples for each class in the first stage (around 2000 images).

The dataset was then divided in three: training, validation and test. The training set consist of 80% of the total samples, while the validation and test sets consist of 10% and 10%, respectively.

4.4. Training framework

For the training and conversion of the model, a framework called aXeleRate[3] was used. It is based on TensorFlow and Keras libraries and optimized to run on Google Collaboratory[4] for online training and access to GPUs. The main feature of this framework is the conversion from

[3] https://github.com/AIWintermuteAI/aXeleRate
[4] https://colab.research.google.com/

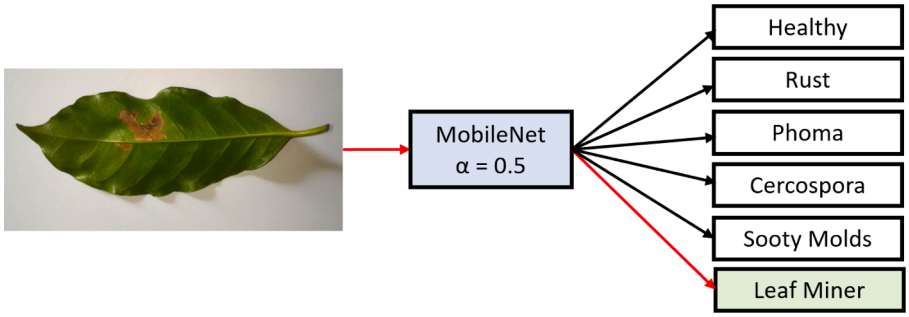


Fig. 2. First proposed system flow, for the single-stage model.

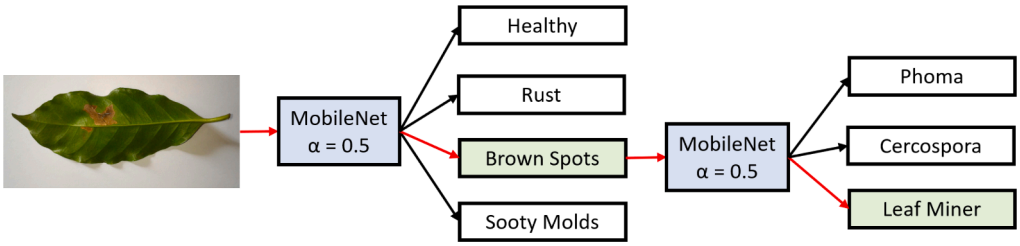
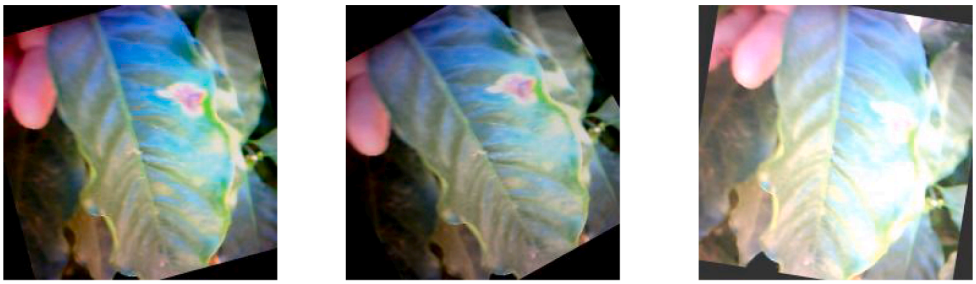


Fig. 3. Second proposed system flow, for the cascade model.



(a) Rotation (b) Rotation+blur (c) Rotation+brightness

Fig. 4. Three samples generated from one source.

Table 4
samples for each class - single-stage.

Single-stage System	
Rust	1800
Sooty Molds	2000
Healthy	2000
Cercospora	1500
Phoma	1500
Leaf Miner	1703

Table 5
samples for each class of the 1st stage - cascade.

First stage	
Rust	1800
Brown Spots	2100
Sooty Molds	2000
Healthy	2000

Table 6
samples for each class of the 2nd stage - cascade.

Second stage	
Cercospora	1500
Phoma	1500
Leaf Miner	1703

TensorFlow Lite model to the KModel[5], which is needed to embed it in the MaixBit board.

Also, to embed the model and perform the inference on the board, the MaixPy IDE and the MicroPython[6] language were used. The IDE is based on the OpenMV IDE and was adapted to interface with the KPU, in addition to acquiring images from the camera and displaying them. MicroPython is a lean and efficient implementation of the Python 3 programming language that includes a small subset of the Python standard library. It is optimized to run on micro-controllers and in constrained environments.

[5] https://wiki.sipeed.com/soft/maixpy/en/course/ai/basic/maixpy_hardware_ai_basic.html
[6] https://micropython.org/

4.5. Model training

For the models, the chosen architecture was MobileNet with $\alpha = 0.5$ and $\rho = 1$, due to the fact that the reduction in performance is small when compared to $\alpha = 1$, with an important reduction in the number of parameters. A MobileNet with $\alpha = 0.5$ pre-trained with the ImageNet dataset was used to train the model using transfer learning and, in that way, speed up the training process. The chosen validation metric was accuracy. Furthermore, a learning rate of 0.001, batch size of 32, dropout equal to 0.2 and a number of epochs equal to 30 were used.

4.6. Evaluation metrics

The performance metrics for evaluating the model were: accuracy (1), precision (2), recall (3) and F1-score (4). We have TP as true positive, FP as false positive, TN true negative and FN as false negative.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

$$precision = \frac{TP}{TP + FP} \tag{2}$$

$$recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \tag{4}$$

5. Results

5.1. Single-stage model

The first architecture was trained using a Tesla K80 GPU, and it took 176 min. The single-stage model with the best training result obtained the following metrics for the test set: loss: 0.0604 - accuracy: 0.9797 - val loss: 0.1123 - val accuracy: 0.9693. Using the test dataset, we have obtained the confusion matrix from Fig. 5 containing the accuracy values for each class and the classification report[7] in Table 7, where "#" refers to the number of samples of each class used to calculate the metrics.

The application of the single-stage model resulted in an overall accuracy of 96%, though some classes had better performance than others. The confusion matrix in Fig. 5 shows that instances from the Cercospora class, for example, which presented the poorer performance (83% accuracy), are often misunderstood as Leaf Miner or Healthy. This may occurs because of similarities between Leaf Miner and Cercospora and because of subtle symptoms that are difficult for the simple model to identify. Other diseases such as Phoma and Leaf Miner are misunderstood with other classes with similar characteristics as well. This issue is expected to be minimized with the adoption of the cascade model.

5.2. Cascade model

The first stage of the cascade model was trained using a Tesla K80 GPU and took 136 min. The first stage model with the best training result obtained the following metrics: loss: 0.0277 - accuracy: 0.9909 - val loss: 0.0330 - val accuracy: 0.9883. Using the test dataset we obtain the confusion matrix shown in Fig. 6 (BS stands for 'Brown Spots' and SM stands for 'Sooty Molds') and classification report in Table 8.

The second stage of the model was trained using a Tesla K80 GPU, and took 97 min. The best training result obtained the following metrics: loss: 0.0206 - accuracy: 0.9840 - val loss: 0.2047 - val accuracy: 0.9472. The confusion matrix is shown in Fig. 7 and the classification report, which contains the accuracy, recall and F1-score, in Table 9.

Results presented in Fig. 7 and Table 9 show that the second stage is able to distinguish between "Brown Spot" classes, once most performance metrics were above 90%. However, to fairly compare results from both models, we need to consider "Brown Spots" accuracy from the first stage, which was 99%. If we multiply this accuracy with those from the second stage, we obtain real accuracy values: around 88% for Cercospora, 92% for Phoma and 96% for Leaf Miner. A similar reasoning can be applied to the other metrics, with similar results. Table 10 shows the performance metrics for the cascade model, combining the two stages. Values marked with (*) are higher than those obtained in the single-step model (Table 7).

Based on these results, we conclude that the Cascade Model improves the overall performance, once accuracy rises from 96% to 98% in the first stage. Also, it increases the accuracy for four of five classes (Healthy, Phoma, Cercospora and Leaf Miner), especially for Cercospora, which rose from 83% to 88% and Leaf Miner, which rose from 91% to 96%. Precision, Recall and F1-score were also slightly increased by the cascade model for most classes. Performance indices for Rust class remained very close to those obtained with the Single-stage Model.

5.3. Onboard performance evaluation

During the process of embedding the models, quantization and optimization steps were carried out, including the conversion to TensorFlow Lite and from TensorFlow Lite to KModel. Furthermore, the difference of hardware can impact the performance, once our board has limited resources when compared to the processors used in Sections 5.1 and 5.2.

In order to evaluate the performance of our models on the chosen embedded system and compare them with those from the previous sections, we used the same test dataset. The images were resized and saved in a SD card, and a script was written in MicroPython to read, classify and save the results. For the Single-Stage model, the obtained confusion matrix is depicted in Fig. 8 and the other performance indices in Table 11.

Despite our MaixBit board limited resources and the fact that the classification model needed to go through simplification steps to be embedded, results are quite similar to those obtained in Section 5.1, which used a powerful GPU. Overall accuracy remained the same, and all other performance metrics had slight variations. This fact demonstrates the feasibility of an efficient low-cost embedded solution to work at the edge.

For the second architecture, using the cascade model, the confusion matrix obtained from the first stage is shown in Fig. 9a and the performance data in Table 12. For the second stage, the confusion matrix can be seen in Fig. 9b and performance data in Table 13.

Table 14 shows the performance metrics for the embedded cascade model, combining the two stages. Values marked with (*) are higher than those obtained in the single-stage embedded model (Table 11). It can be observed that, as mentioned before, our models showed little to none loss in performance when embedded.

Again, most performance indices remained above 90%, which is an indication that the model can classify properly and help small farmers to quickly identify diseases in coffee leaves. Classes like Cercospora and Leaf Miner, which are very similar and difficult to distinguish, had the largest increase in accuracy for the cascade model, and this improvement can be explained by the use of a second-stage specialized in "Brown Spots". These classes, along with Phoma, had improvements in all performance metrics, going from accuracy values of 83% and 91% to 87% and 95%, respectively. Almost all performance indices were increased in the cascade model (Table 14) in relation to those from the single-stage model (Table 11), except for those related to Rust class, which had a small decrease (around 1 to 3%).

[7] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report

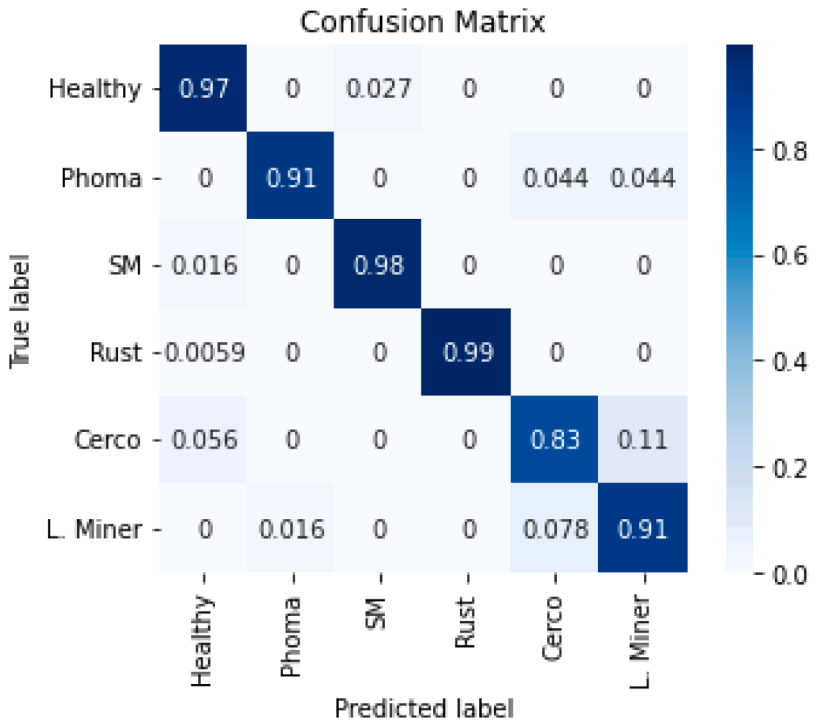


Fig. 5. Confusion matrix - Single-stage model (SM stands for 'Sooty Molds').

Table 7
Evaluation metrics - Single-stage model.

	Accuracy	Precision	Recall	f1-score	#
Healthy	0.97	0.97	0.97	0.97	200
Phoma	0.91	0.98	0.91	0.95	150
S. Molds	0.98	0.97	0.98	0.98	200
Rust	0.99	1.00	0.99	1.00	180
Cerco.	0.83	0.65	0.83	0.73	150
L. Miner	0.91	0.92	0.91	0.91	170
Accuracy				0.96	1050

Table 8
Evaluation metrics - Cascade model (First stage only).

	Precision	Recall	f1-score	support
Healthy	0.98	0.98	0.98	200
Rust	0.99	0.97	0.98	180
BS	0.97	0.99	0.98	200
SM	0.98	0.99	0.99	200
Accuracy			0.98	780

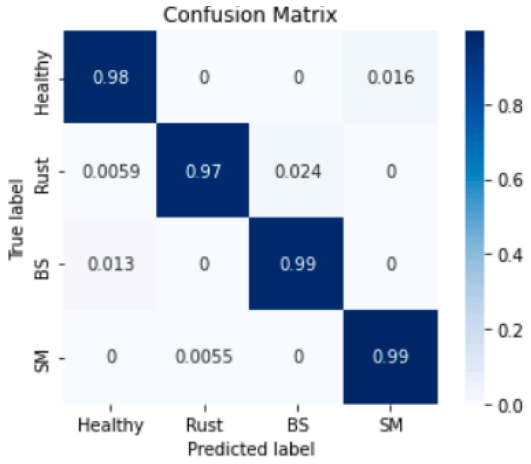


Fig. 6. Confusion matrix - Cascade model (First stage only).

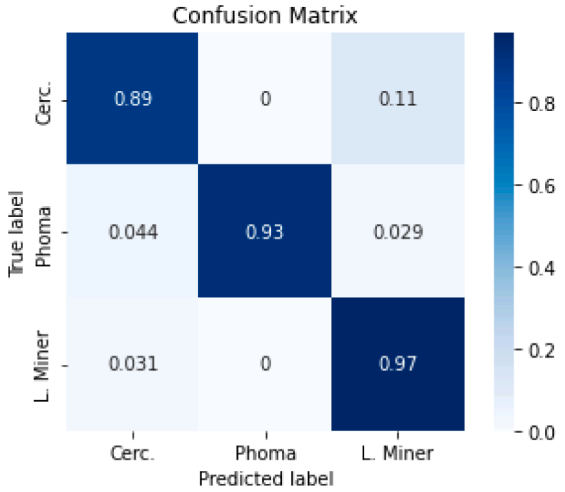


Fig. 7. Confusion matrix - Cascade model (Second stage only).

5.4. Usability and affordability

Our models achieved performances close or better then projects embedded in smartphones [39] and single board computers, such as Jetson Nano and Raspberry Pi [40,41], while being several times less expensive. The prototype has a very low-cost, since the board, camera and display cost US$30 and the 3D case costs around 5$. Considering the price of the devices in Reais (R$), which is the currency of Brazil, we have that the cost of the developed device is around R$200, while a smartphone has an average price of R$1845 and the Jetson Nano is around R$3000. Therefore, our device presents results similar to devices nine to fifteen times more expensive in the Brazilian market.

Table 9
Evaluation metrics - Cascade model (Second stage only).

	Precision	Recall	f1-score	support
Cerc.	0.86	0.89	0.82	150
Phoma	1.00	0.93	0.96	150
L. Miner	0.94	0.97	0.95	170
Accuracy			0.94	470

Table 10
Evaluation metrics - Cascade model (Combining two stages).

	Accuracy	Precision	Recall	f1-score	#
Healthy	0.98*	0.98*	0.98*	0.98*	200
Phoma	0.92*	0.92	0.94*	0.92	150
S. Molds	0.99*	0.99*	0.99*	0.99*	200
Rust	0.97	0.97	0.98	0.97	180
Cerco.	0.88*	0.88*	0.80*	0.88*	150
L. Miner	0.96*	0.96	0.93*	0.96*	170

Furthermore, as the focus of the development is the small farmer, with little or no familiarity with the technology, the developed model presents a very intuitive and simple interface. To obtain the information, the user simply has to point the device at the coffee leaf and the classification will automatically appear on the screen, along with the probability of the result (Fig. 10). As a result, we conclude that the project achieved its objectives, obtaining results comparable to the state of the art, while remaining affordable and easy to use by the small farmer.

6. Conclusion

The objective of this study was to embed convolutional networks in a low-cost microcontrolled board to classify diseases in coffee leaves at the edge. The feasibility of using convolutional neural networks on edge devices for disease classification based on coffee tree leaf images was verified. The prototype proved to be an efficient embedded system, that can be powered by batteries, does not require an internet connection, and can be used in real-world scenarios for people with little familiarity with technology. Due to its low cost and independence, it could benefit especially small producers, with fewer resources.

Both proposed architectures - single-stage and cascade - presented promising results and proved to be efficient and able to assist the coffee producer. Most performance indices obtained are above 90%, including accuracy, precision, recall and f1-score. The cascade model stands out for its best overall accuracy (around 98% for the first and 94% for the second stage). Almost all performance indices were increased in the cascade model in relation to those from the single-stage model, except for those related to Rust class, which had a small decrease. The single-stage model showed difficulties in diseases with more subtle or similar characteristics, such as Cercospora, Phoma and Leaf Miner. Some of these difficulties were attenuated in the two-stage system, proving the effectiveness of the cascade model. Nevertheless, it must be stated that the improvement in performance for some classes in cascade model comes with a cost in memory and latency, and there is a cost-benefit ratio that must be taken into account when choosing the model.

There is some future work be made. First, some aspects related to the information contained in the dataset can be improved, such as using only samples taken in real use situations, in order to improve the accuracy of the on-board system. Also, new classes can be created containing two or more diseases, because some leaves may be affected by more than one condition. For specific applications, it may be recommended to remove some classes from the model, containing diseases that do not exist in a certain geographical region, aiming to increase the accuracy of the system.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence

Table 11
Evaluation metrics - Single-stage model - Embedded.

	Accuracy	Precision	Recall	f1-score	support
Healthy	0.97	0.96	0.97	0.96	200
Phoma	0.90	0.98	0.90	0.94	150
S. Molds	0.98	0.97	0.98	0.97	200
Rust	0.99	1.00	0.99	0.99	180
Cerco.	0.83	0.65	0.83	0.73	150
L. Miner	0.91	0.91	0.91	0.91	170
Accuracy				0.96	1050

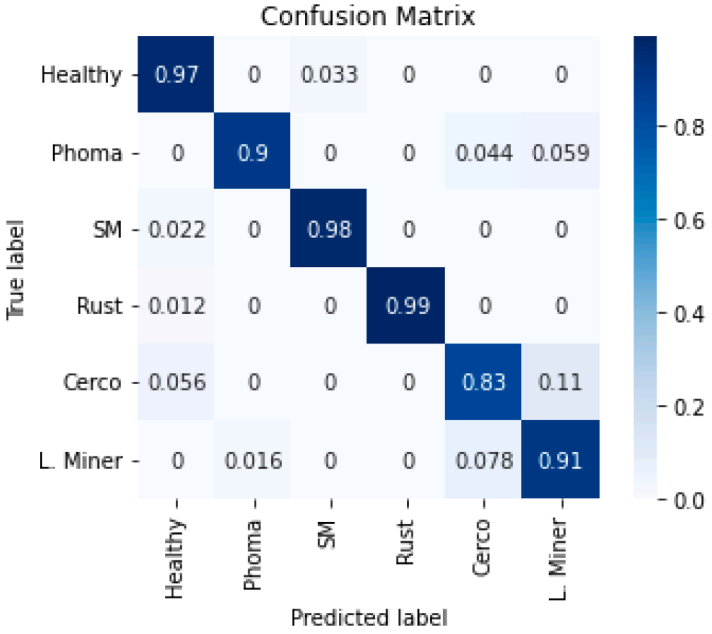
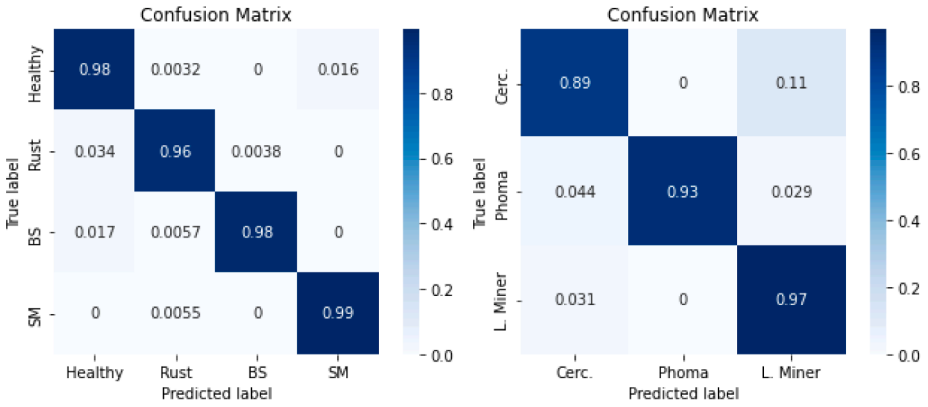


Fig. 8. Confusion matrix - Single-stage model - Embedded.



(a) First stage (b) Second stage

Fig. 9. Confusion matrix - Cascade Model - Embedded.

Table 12
First stage evaluation metrics for the embedded model.

	Precision	Recall	f1-score	support
Healthy	0.96	0.98	0.97	200
Rust	0.99	0.96	0.98	180
BS	0.99	0.98	0.99	200
SM	0.97	0.99	0.98	200
Accuracy			0.98	780

Table 13
Second stage evaluation metrics for the embedded model.

	Precision	Recall	f1-score	support
Cerc.	0.86	0.89	0.82	150
Phoma	1.00	0.93	0.96	150
L. Miner	0.94	0.97	0.95	170
Accuracy			0.94	470

Table 14
Evaluation metrics - Cascade model (Combining two stages) - Embedded.

	Accuracy	Precision	Recall	f1-score	#
Healthy	0.98*	0.96	0.98*	0.97*	200
Phoma	0.91*	0.99*	0.91*	0.95*	150
S. Molds	0.99*	0.97	0.99*	0.98*	200
Rust	0.96	0.99	0.96	0.98	180
Cerco.	0.87*	0.85*	0.87*	0.81*	150
L. Miner	0.95*	0.93*	0.95*	0.94*	170

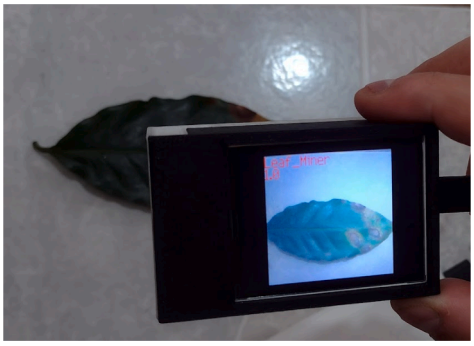


Fig. 10. A classification made with a real leaf.

the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] C. M. Oliveira, A. M. Auad, S. M. Mendes, M. R. Frizzas, Crop losses and the economic impact of insect pests on Brazilian agriculture, https://www.journals.elsevier.com/crop-protection(2014).
[2] ICO, Monthly coffee market report, 2021, https://www.ico.org/Market-Report-20-21-e.asp.
[3] CNA, Pesquisa safra cafeeira, 2019, http://www.sapc.embrapa.br/arquivos/consorcio/publicacoes_tecnicas/Pesquisa_Safra_Cafeeira_2019_CNA.pdf.
[4] U. Mokhtar, N. El Bendary, A.E. Hassenian, E. Emary, M.A. Mahmoud, H. Hefny, M. F. Tolba, SVM-based detection of tomato leaves diseases, in: D. Filev, J. Jabłkowski, J. Kacprzyk, M. Krawczak, I. Popchev, L. Rutkowski, V. Sgurev, E. Sotirova, P. Szynkarczyk, S. Zadrozny (Eds.), Intelligent Systems'2014, Springer International Publishing, Cham, 2015, pp. 641–652.
[5] M. Rahaman, E. Hossain, M. Hossain, A color and texture based approach for the detection and classification of plant leaf disease using KNN classifier, 2019, pp. 1–6, https://doi.org/10.1109/ECACE.2019.8679247.
[6] R.M. Mohana, C.K.K. Reddy, P. Anisha, B.R. Murthy, Random forest algorithms for the classification of tree-based ensemble, Mater. Today. Proc. (2021), https://doi.org/10.1016/j.matpr.2021.01.788.https://www.sciencedirect.com/science/article/pii/S2214785321008853
[7] G. Zhong, X. Ling, L.-N. Wang, From shallow feature learning to deep learning: benefits from the width and depth of deep architectures, WIREs Data Min. Knowl. Discovery 9 (1) (2019) e1255, https://doi.org/10.1002/widm.1255.https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1255
[8] G.L. Manso, H. Knidel, R.A. Krohling, J.A. Ventura, A smartphone application to detection and classification of coffee leaf miner and coffee leaf rust, CoRR (2019). abs/1904.00742.http://arxiv.org/abs/1904.00742
[9] F.J.P. Montalbo, Stage wise aggregated triple deep convolutional neural network (2020).
[10] S.P. Mohanty, D.P. Hughes, M. Salathé, Using deep learning for image-based plant disease detection (2016).
[11] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: F. Pereira, C. Burges, L. Bottou, K. Weinberger (Eds.), Advances in Neural Information Processing Systems Vol. 25, Curran Associates, Inc., 2012.https://www.proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
[13] A. Fuentes, S. Yoon, S.C. Kim, D.S. Park, A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition, Sensors 17 (9) (2017), https://doi.org/10.3390/s17092022.https://www.mdpi.com/1424-8220/17/9/2022
[14] B. Liu, Y. Zhang, D. He, Y. Li, Identification of apple leaf diseases based on deep convolutional neural networks, Symmetry 10 (1) (2018), https://doi.org/10.3390/sym10010011.https://www.mdpi.com/2073-8994/10/1/11
[15] J.G.A. Barbedo, A review on the main challenges in automatic plant disease identification based on visible range images, Biosyst. Eng. 144 (2016) 52–60, https://doi.org/10.1016/j.biosystemseng.2016.01.017.https://www.sciencedirect.com/science/article/pii/S1537511015302476

[16] A. Fuentes, S. Yoon, S.C. Kim, D.S. Park, A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition, Sensors 17 (9) (2017), https://doi.org/10.3390/s17092022.https://www.mdpi.com/1424-8220/17/9/2022

[17] M. Turkoglu, D. Hanbay, Plant disease and pest detection using deep learning-based features, Turkish J. Electr. Eng.Comput. Sci. 27 (2019) 1636–1651, https://doi.org/10.3906/elk-1809-181.

[18] F. Jiang, Y. Lu, Y. Chen, D. Cai, G. Li, Image recognition of four rice leaf diseases based on deep learning and support vector machine, Comput. Electron. Agric. 179 (2020) 105824, https://doi.org/10.1016/j.compag.2020.105824.https://www.sciencedirect.com/science/article/pii/S016816992030795X

[19] J. Yu, S.M. Sharpe, A.W. Schumann, N.S. Boyd, Deep learning for image-based weed detection in turfgrass, Eur. J. Agron. 104 (2019) 78–84, https://doi.org/10.1016/j.eja.2019.01.004.https://www.sciencedirect.com/science/article/pii/S1161030118306129

[20] P. Bansal, R. Kumar, S. Kumar, Disease detection in apple leaves using deep convolutional neural network, Agriculture 11 (7) (2021), https://doi.org/10.3390/agriculture11070617.https://www.mdpi.com/2077-0472/11/7/617

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, L. Fei-Fei, ImageNet large scale visual recognition challenge, Int. J. Comput. Vis. (IJCV) 115 (3) (2015) 211–252, https://doi.org/10.1007/s11263-015-0816-y.

[22] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, https://arxiv.org/abs/1409.1556. 10.48550/ARXIV.1409.1556.

[23] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, CoRR (2015). abs/1512.03385.http://arxiv.org/abs/1512.03385

[24] A.G. HOWARD, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, MobileNets: efficient convolutional neural networks for mobile vision applications (2017).

[25] M. Tan, Q.V. Le, EfficientNet: rethinking model scaling for convolutional neural networks, CoRR (2019). abs/1905.11946.http://arxiv.org/abs/1905.11946

[26] H. Wu, P. Judd, X. Zhang, M. Isaev, P. Micikevicius, Integer quantization for deep learning inference: principles and empirical evaluation, CoRR (2020). abs/2004.09602.https://arxiv.org/abs/2004.09602

[27] A. Jain, S. Bhattacharya, M. Masuda, V. Sharma, Y. Wang, Efficient execution of quantized deep learning models: a compiler approach, CoRR (2020). abs/2006.10226.https://arxiv.org/abs/2006.10226

[28] P. Kluska, M. Zieba, Post-training quantization methods for deep learning models, in: N.T. Nguyen, K. Jearanaitanakij, A. Selamat, B. Trawiński, S. Chittayasothorn (Eds.), Intelligent Information and Database Systems, Springer International Publishing, Cham, 2020, pp. 467–479.

[29] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: large-scale machine learning on heterogeneous systems, 2015, Software available from tensorflow.org, https://www.tensorflow.org/.

[30] A. SARKAR, Understanding depthwise separable convolutions and the efficiency of mobilenets, 2021, https://www.towardsdatascience.com/understanding-depthwise-separable-convolutions-and-the-efficiency-of-mobilenets-6de3d6b62503.

[31] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556(2014).

[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: a large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition, Ieee, 2009, pp. 248–255.

[33] KENDRYTE, K210 datasheet, 2019. https://www.s3.cn-north-1.amazonaws.com.cn/dl.kendryte.com/documents/kendryte_datasheet_20181011163248_en.pdf.

[34] S.A. TIRADAS, Aceleracin de ai en dispositivos de bajo consumo ai acceleration on low-power devices (2020).

[35] E. Klippel, R.A. Oliveira, D. Maslov, A. Bianchi, S.E. Silva, C.T. Garrocho, Towards to an embedded edge ai implementation for longitudinal rip detection in conveyor belt. Anais Estendidos do X Simpósio Brasileiro de Engenharia de Sistemas Computacionais, SBC, 2020, pp. 97–102.

[36] R. Krohling, J. Esgario, J.A. Ventura, Bracol - a brazilian arabica coffee leaf images dataset to identification and quantification of coffee diseases and pests(2019).

[37] F.J.P. Montalbo, A.A. Hernandez, Classifying Barako coffee leaf diseases using deep convolutional models (2020).

[38] F.J.P. Montalbo, Automated diagnosis of diverse coffee leaf images through a stage-wise aggregated triple deep convolutional neural network, Mach. Vis. Appl. 33 (1) (2022) 1–22.

[39] U. Barman, R.D. Choudhury, D. Sahu, G.G. Barman, Comparison of convolution neural networks for smartphone image based real time classification of citrus leaf disease, Comput. Electron. Agric. 177 (2020) 105661, https://doi.org/10.1016/j.compag.2020.105661.https://www.sciencedirect.com/science/article/pii/S0168169920302258

[40] T. Aboneh, A. Rorissa, R. Srinivasagan, A. Gemechu, Computer vision framework for wheat disease identification and classification using Jetson GPU infrastructure, Technologies 9 (3) (2021) 47.

[41] M. Sankar, D.N. Mudgal, T. varsharani jagdish, N.w. Geetanjali Laxman, M. Mahesh Jalinder, Green leaf disease detection using raspberry pi. 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT), 2019, pp. 1–6, https://doi.org/10.1109/ICIICT1.2019.8741508.