HardwareX 12 (2022) e00363

Contents lists available at ScienceDirect

HardwareX

journal homepage: www.elsevier.com/locate/ohx

# A CNN-based image detector for plant leaf diseases classification

Laura Falaschetti [a,*], Lorenzo Manoni [a], Denis Di Leo [a], Danilo Pau [b], Valeria Tomaselli [c], Claudio Turchetti [a]

[a] Department of Information Engineering, Università Politecnica delle Marche, Ancona, Italy
[b] System Research and Applications, STMicroelectronics, Agrate Brianza, Italy
[c] System Research and Applications, STMicroelectronics, Catania, Italy

ARTICLE INFO

*Keywords:*
Image detector
Esca disease
convolutional neural network
embedded systems
plant diseases recognition

ABSTRACT

Identifying diseases from images of plant leaves is one of the most important research areas in precision agriculture. The aim of this paper is to propose an image detector embedding a resource constrained convolutional neural network (CNN) implemented in a low cost, low power platform, named OpenMV Cam H7 Plus, to perform a real-time classification of plant disease. The CNN network so obtained has been trained on two specific datasets for plant diseases detection, the ESCA-dataset and the PlantVillage-augmented dataset, and implemented in a low-power, low-cost Python programmable machine vision camera for real-time image acquisition and classification, equipped with a LCD display showing to the user the classification response in real-time. Experimental results show that this CNN-based image detector can be effectively implemented on the chosen constrained-resource system, achieving an accuracy of about 98.10%/95.24% with a very low memory cost (718.961 KB/735.727 KB) and inference time (122.969 ms/125.630 ms) tested on board for the ESCA and the PlantVillage-augmented datasets respectively, allowing the design of a portable embedded system for plant leaf diseases classification. Source files are available at https://doi.org/10.17605/OSF.IO/UCM8D.



## Specifications table

| Hardware name | *OpenMV Cam Plant Diseases Detector* |
|---|---|
| **Subject area** | • Environmental, planetary and agricultural sciences<br>• General |
| **Hardware type** | • Imaging tools |
| **Closest commercial analog** | Wio Lite AI Single Board:https://www.hackster.io/news/seeed-s-new-wiolite-ai-packs-computer-vision-into-a-feather-sized-board-30a1eb8b08b3; Available at: https://www.seeedstudio.com/Wio-Lite-AI-Single-Board-p-5120.html |
| **Open source license** | MIT License |
| **Cost of hardware** | $136.47 |
| **Source file repository** | https://doi.org/10.17605/OSF.IO/UCM8D |

* Corresponding author.
*E-mail address:* l.falaschetti@univpm.it (L. Falaschetti).

https://doi.org/10.1016/j.ohx.2022.e00363

## 1. Hardware in context

Identifying diseases from images of plant leaves is one of the most important research areas in precision agriculture [1]. Several artificial intelligence approaches are currently used for detecting and classifying plant diseases. The most common approaches are the k-nearest neighbours (kNN), logistic regression, decision tree, support vector machine (SVM) [2]. Recently, deep neural networks (DNNs) and specifically convolutional neural networks (CNNs) have proven to be extremely effective to solve this task [3]. These approaches are combined with various image pre-processing methods in order to enhance feature extraction.

In the context of precision agriculture, the availability of a low-cost, low-power, portable, easy-to-use vision system incorporating deep learning techniques, which could be combined with autonomous agricultural vehicles, plays a great role since it would offer the agronomist a valuable assistance for the plant disease detection and diagnosis. Such devices could be drones and other autonomous agricultural vehicles, equipped with an embedded vision system, to be used by growers or agronomists for real-time monitoring and dynamic disease detection on large-scale open-field cultivations. The realization of a low-cost, low-power, portable, easy-to-use intelligent vision system leads to several challenges: th ction of computational complexity and memory occupation so that the DNN can be integrated directly into the vision system, i.e. an embedded device based on a microcontroller with limited computational power and very low energy consumption, together with the reduction of inference time in order to realize a real-time application, by preserving the performance in terms of accuracy.

Table 1 reports a summary of the state-of-the-art CNN models for plant diseases identification based on image classification.

As you can see in Table 1, the state-of-the-art plant diseases models achieve a very high classification accuracy, requiring a large number of parameters and higher computation cost, which prohibit their usage in embedded devices. In recent years the models have achieved very high accuracies, exploring different methods: transfer learning applied to existing architectures in literature [4,21,8,23,27], existing DNN models combined with different features extraction methods [11,17], modified versions of existing networks [31,33,41,36], novel network architectures [25,29,34,38–40]. Such a higher accuracy, in most cases, has been reached using complexed architectures that require a high number of parameters. In [25], the authors proposed a simple nine-layer CNN model to identify plant diseases, applying the proposed CNN to the PlantVillage dataset with data-augmentation techniques to increase the data size. Despite the simplicity of this architecture, the authors reported better accuracy than that of a traditional machine-learning-based approach. As mentioned above, in order to realized an embedded system that could be deployed in precision agriculture, the constrains must be the reduction of computational complexity and the memory occupation, still maintaining a good accuracy. According to our knowledge, few works in literature proposed a hardware implementation of these plant diseases models [23,34,39,40].

Following the above motivations, the aim of this paper is to propose an image detector for plant leaf diseases classification based on a lightweight and accurate CNN, that can be implemented on a low cost, low power core, while preserving good performance in terms of accuracy and inference time. This image detector has been presented in [42] where the severe constraints typical of embedded systems were satisfied with a network compression using tensor decomposition techniques. In this paper instead the final lightweight model was obtained with a filter pruning compression method [43] applied to a similar architecture as [42]. The pruning step is followed by a final retraining to recover the loss of accuracy introduced by compression. The CNN has been trained on two datasets specific for plant diseases recognition and implemented in a low-power, low-cost Python programmable machine vision camera, named OpenMV H7 Plus, for real-time classification. The obtained results show that the system is able to obtain good performances both in the case of binary and multi-class classification. In addition, a comparison with a state-of-the-art lightweight CNN for plant disease recognition [25] has been conducted.

The manuscript is organized as follows. Section 2 describes the CNN-based image detector in all its components. Section 3 provides a list and a description of the scripts implemented both to train the CNNs and to embed these networks on the OpenMV Cam H7 Plus. Section 4 lists the materials needed to reproduce the presented image detector. Detailed software building and installation instructions are given in Section 5 and Section 6. Section 7 presents the architecture of the proposed CNN and the experimental carried out to validate the performance of the vision system principally in terms of inference time, frame per second (FPS) and memory occupancy.

## 2. Hardware description

An example hardware setup is shown in Fig. 1 and Fig. 2.

Fig. 1a shows the OpenMV Cam connected to a laptop via micro USB cable. The PC is equipped with the OpenMV integrated development environment (IDE) that shows in the serial terminal the classification results of the images acquired by the camera of the OpenMV Cam H7 Plus. Fig. 1b shows how the OpenMV Cam H7 Plus can be supplied by a power bank, avoiding to use the PC and obtaining a portable system to classify the leaf image in real-time.

With the setup used in Fig. 1b, Fig. 2 shows an application example of the leaf image detector: Fig. 2a and Fig. 2b show the different responses of the system when acquires the frame of a healthy leaf or a leaf affected by esca disease. In the first case no message is expected but in the second case an alert message ('E') has shown on LCD display to notify the presence of the plant disease.

**Table 1**
Summary of the state-of-the-art CNNs for plant diseases detection.

| Author | Methods | Dataset | Classes | Accuracy | Parameters | Implementation on embedded system |
|---|---|---|---|---|---|---|
| Mohanty et al. [4] | AlexNet [5] and GoogleNet [6] | PlantVillage [7] | 38 | 99.27%, 99.34% | AlexNet: 60 million GoogleNet: 5 million | – |
| Ramcharan et al. [8] | Inception V3 [9] based on GoogleNet | Cassava dataset [10] | 6 | 93% | Inception V3: 27 million | – |
| Fuentes et al. [11] | Faster R-CNN [12], R-FCN [13], SSD [14] combined with VGG-16 [15] and ResNet [16] | custom Tomato Diseases and Pests Dataset 5000 images | 9 | 83% | Faster R-CNN with VGG: 2.4 million | – |
| Pawara et al. [17] | AlexNet [5] and GoogleNet [6] | AgriPlant Dataset [18] | 10 | 96.37%, 98.33% | AlexNet: 60 | – |
| | | LeafSnap Dataset [19] | 184 | 89.51%, 97.66% | millionGoogleNet: 5 | |
| | | Folio Dataset[20] | 32 | 97.67%, 97.63% | million | |
| Ferentinos et al. [21] | AlexNetOWTBn [22] and VGG [15] | custom dataset 87848 images | 58 | 99.49%, 99.53% | AlexNetOWTBn: 60 million VGG: 138 million | – |
| Ramacharan el al. [23] | MobileNet [24] - SSD [14] | Cassava dataset [10] | 6 | 80.6% on images 70.4% on video | MobileNet-SSD: 6 million | ✔Samsung Galaxy S5 Android device |
| Geetharamani et al. [25] | CNN | PlantVillage with data augmentation [26] | 39 | 96.46% | 212,543 | – |
| Chen et al. [27] | VGG-19 pre-trained on ImageNet with Inception module [28] | Maize PlantVillage [7] | 4 | 92% | VGG-19: 143 million | – |
| Chen et al. [29] | DenseNet [30] | Maize PlantVillage [7] | 4 | 98.50% | 33.97 million | – |
| Chen et al. [31] | MobileNet-V2 [32] | PlantVillage [7] | 38 | 99.71% | 3.83 million | – |
| Chen et al. [33] | DenseNet [30] | custom dataset 1000 images | 5 | 97.60% | 3.40 million | – |
| Li et al. [34] | CNN | NBAIR [35] | 50 | 95.4% | 0.75 million | ✔FPGA |
| | | Li et al. dataset [34] | 10 | 96.2% | | |
| Chen et al. [36] | MobileNet-V2 [32] and Attention Mechanism along with a Classification Activation Map [37] | Li et al. dataset [34] | 10 | 99.14% | 3.83 million | – |
| Chen et al. [38] | Semantic Segmentation and CNN | Grape PlantVillage [7] | 4 | 93.75% | 44.51 million | – |
| Mishra et al. [39] | CNN | PlantVillage [7] subset + custom images | 3 | 88.46% | 22.75 million | ✔ Intel Movidius NCS with Raspberry Pi 3 |
| Gajjar et al. [40] | SSD [14] combined with CNN | PlantVillage [7] subset | 20 | 96.88% | 6.07 million | ✔ NVIDIA Jetson TX1 |

(a) (b)

**Fig. 1.** OpenMV Cam with LCD display (a) used connected to a PC provided with OpenMV IDE that shows on the serial terminal the classification response applied to different leaves (CNN trained on PlantVillage-augmented dataset), (b) used in stand-alone mode powered by a power bank.

(a) (b)

**Fig. 2.** OpenMV Cam with LCD display that shows the classification response applied to Esca disease (CNN trained on ESCA-dataset). (a) Healthy leaf (no tag), (b) Leaf affected by Esca disease correctly detected as shown in the LCD display (tag 'E').

The main components of the embedded system are:

1. *OpenMV Cam H7 Plus*
   The OpenMV Cam STM32H7 Plus camera [1] is a low-power Python programmable machine vision camera that supports an extensive set of image processing functions and neural networks. It is based on the STM32H743II ARM Cortex-M7 MCU running at 480 MHz featuring 32 MBs off-chip SDRAM, 1 MB SRAM, 32 MB off-chip FLASH and 2 MB on-chip FLASH. The OV5640 image sensor can capture images up to size 2592 $\times$ 1944 but most algorithms run between 10–15-25–50 FPS on QVGA (320 $\times$ 240) resolutions and below.
2. *OpenMV LCD Shield*
   The LCD shield [2] gives OpenMV Cam the ability to display what it sees on-the-go while not connected to the computer. It features a 1.8" 128 $\times$ 160 16-bpp (RGB565) TFT LCD display with a controllable backlight. The OpenMV Cam's firmware has built-in support for controlling the LCD Shield using the `lcd` API module.
3. *PC equipped with OpenMV CAM IDE*
   OpenMV IDE[3] is the an integrated development environment specifically designated for use with OpenMV Cam. It features a text editor, a debug terminal and a frame buffer viewer with a histogram display. OpenMV IDE simplifies programming the OpenMV Cam and integrates a serial bootloader to flash the board.
4. *Micro SD card*
   The micro SD card is extremely useful to save the model (specifically tflite int8 format) previously trained on a specific dataset and the main.py script written for the classification application, delegating the execution of the application to the Micro Python interpreter embedded in the board.
5. *Micro USB cable*
   The micro USB cable can be used to connect the board to the PC or to a power supply.

[1] https://openmv.io/products/openmv-cam-h7-plus
[2] https://openmv.io/products/lcd-shield
[3] https://openmv.io/pages/download

6. *Power bank*
   The power bank can be used to powered the board avoiding to connecting it to PC and realizing a portable system.

## 3. Design files

### 3.1. Design files summary

| Design filename | File type | Open source license | Location of the file |
|---|---|---|---|
| CNN_Esca.ipynb | Google Colaboratory Notebook file | MIT License | OSF Repository |
| CNN_Esca_pruned.ipynb | Google Colaboratory Notebook file | MIT License | OSF Repository |
| CNN_PV.ipynb | Google Colaboratory Notebook file | MIT License | OSF Repository |
| CNN_PV_pruned.ipynb | Google Colaboratory Notebook file | MIT License | OSF Repository |
| CNN_Esca_model.h5 | model file | MIT License | OSF Repository |
| CNN_Esca_model.tflite | model file | MIT License | OSF Repository |
| CNN_Esca_pruned_model.h5 | model file | MIT License | OSF Repository |
| CNN_Esca_pruned_model.tflite | model file | MIT License | OSF Repository |
| CNN_PV_model.h5 | model file | MIT License | OSF Repository |
| CNN_PV_model.tflite | model file | MIT License | OSF Repository |
| CNN_PV_pruned_model.h5 | model file | MIT License | OSF Repository |
| CNN_PV_pruned_model.tflite | model file | MIT License | OSF Repository |
| conversion_DatasetEsca_into_bmp_format.ipynb | Google Colaboratory Notebook file | MIT License | OSF Repository |
| conversion_DatasetPV_into_bmp_format.ipynb | Google Colaboratory Notebook file | MIT License | OSF Repository |
| augmented_esca_dataset_9transformation_splitted.zip | data file | MIT License | OSF Repository |
| Dataset_Esca_Test_bmp_128x128.zip | data file | MIT License | OSF Repository |
| Dataset_PV_Test_bmp_128x128.zip | data file | MIT License | OSF Repository |
| openmv_demoEsca_SD_testingSet.py | micropython file | MIT License | OSF Repository |
| openmv_demoEsca_SD_realtime.py | micropython file | MIT License | OSF Repository |
| openmv_demoEsca_SD_realtime_responseOnLCD.py | micropython file | MIT License | OSF Repository |
| openmv_demoPV_SD_testingSet.py | micropython file | MIT License | OSF Repository |
| openmv_demoPV_SD_realtime.py | micropython file | MIT License | OSF Repository |
| demo_Esca-1.mp4 | media file | MIT License | OSF Repository |
| demo_Esca-2.avi | media file | MIT License | OSF Repository |
| demo_PV.mp4 | media file | MIT License | OSF Repository |

Below is a brief description of the file listened above:

- CNN_Esca.ipynb: This file contains the code to train, to validate and to test the initial CNN architecture for the ESCA-dataset [44]. It generates both the h5 and tflite models and then evaluates the models' performances, such as accuracy, memory cost, number of parameters.
- CNN_Esca_pruned.ipynb: This file contains the code to apply the filter pruning compression on the model trained on Esca dataset and then to perform the retraining. It generates the final model for Esca dataset in h5 and tflite format and then evaluates its performaces in terms of accurary, memory cost, number of parameters.
- CNN_PV.ipynb: This file contains the code to train, to validate and to test the initial CNN architecture for the PlantVillage-augmented dataset [7,26], together with a section to split the dataset in three partitions (training, validation, testing set). It generates both the h5 and tflite models and then evaluates the models performaces, such as accurary, memory cost, number of parameters.
- CNN_PV_pruned.ipynb: This file contains the code to apply the filter pruning compression on the model trained on PlantVillage-augmented dataset and then to perform the retraining. It generates the final model for Plant Village dataset in h5 and tflite format and then evaluates its performaces in terms of accurary, memory cost, number of parameters.

- CNN_Esca_model.h5: The Keras model file (h5 format) generated by CNN_Esca.ipynb.
- CNN_Esca_model.tflite: The TensorFlow Lite model file (tflite format) generated by CNN_Esca.ipynb.
- CNN_Esca_pruned_model.h5: The Keras model file (h5 format) generated by CNN_Esca_pruned.ipynb.
- CNN_Esca_pruned_model.tflite: The TensorFlow Lite model file (tflite format) generated by CNN_Esca_pruned.ipynb.
- CNN_PV_model.h5: The Keras model file (h5 format) generated by CNN_PV.ipynb.
- CNN_PV_model.tflite: The TensorFlow Lite model file (tflite format) generated by CNN_PV.ipynb.
- CNN_PV_pruned_model.h5: The Keras model file (h5 format) generated by CNN_PV_pruned.ipynb.
- CNN_PV_pruned_model.tflite: The TensorFlow Lite model file (tflite format) generated by CNN_PV_pruned.ipynb.
- conversion_DatasetEsca_into_bmp_format.ipynb: This file converts the images of the Esca dataset from jpg format to bmp format in order to be used in the OpenMV Cam.
- conversion_DatasetPV_into_bmp_format.ipynb: This file converts the images of the PlantVillage-augmented dataset from jpg format to bmp format in order to be used in the OpenMV Cam.
- augmented_esca_dataset_9transformation_splitted.zip: In this work we use a version of the ESCA-dataset with applied 9 data augmentations transformations. This dataset can be also generated by the scripts available in the ESCA-dataset repository, but, for your convenience, we provides the dataset already generated.
- Dataset_Esca_Test_bmp_128x128.zip: The ESCA-dataset converted in bpm format (pixel size 128 $\times$ 128).
- Dataset_PV_Test_bmp_128x128.zip: The PlantVillage-augmented dataset converted in bpm format (pixel size 128 $\times$ 128).
- openmv_demoEsca_SD_testingSet.py: This file contains the MicroPython code to be used in OpenMV Cam. This file loads the tflite model file and the bpm version of the testing set of ESCA-dataset, both saved in the OpenMV SD card, and it performs the classification on the testing set.
- openmv_demoEsca_SD_realtime.py: This file contains the MicroPython code to be used in OpenMV Cam. This file loads the tflite model file saved in the OpenMV SD card, and it performs a real-time classification of the images captured by the on board camera.
- openmv_demoEsca_SD_realtime_responseOnLCD.py: This file contains the MicroPython code to be used in OpenMV Cam. This file loads the tflite model file saved in the OpenMV SD card, and it performs a real-time classification of the images captured by the on board camera, displaying the results on the LCD display.
- openmv_demoPV_SD_testingSet.py: This file contains the MicroPython code to be used in OpenMV Cam. This file loads the tflite model file and the bpm version of the testing set of PlantVillage-augmented dataset, both saved in the OpenMV SD card, and it performs the classification on the testing set.
- openmv_demoPV_SD_realtime.py: This file contains the MicroPython code to be used in OpenMV Cam. This file loads the tflite model file generated from the training on the PlantVillage-augmented dataset and then saved in the OpenMV SD card, and it performs a real-time classification of the images captured by the on board camera.
- demo_Esca-1.mp4: This video shows how the board performs when a healthy/unhealthy leaf is captured (ESCA-dataset).
- demo_Esca-2.avi: Same as demo_Esca-1.mp4 but using pre-saved images of leaves displayed on a tablet (ESCA-dataset).
- demo_PV.mp4: This video shows how the board performs when different species of healthy/unhealthy leaves are captured (PlantVillage-augmented dataset).

## 4. Bill of materials

| Designator | Component | Number | Cost per unit - USD | Total cost - USD | Source of materials |
|---|---|---|---|---|---|
| OpenMV Cam H7 Plus | OC1 | 1 | 80.00 | 80.00 | https://openmv.io/products/openmv-cam-h7-plus |
| OpenMV LCD Shield | LCD1 | 1 | 20.00 | 20.00 | https://openmv.io/products/lcd-shield |
| Micro SD Card | SD1 | 1 | 8.69 | 8.69 | https://www.amazon.com/SanDisk-Ultra-UHS-I-Memory-Adapter/dp/B00M55C0VU |
| micro USB cable | USB1 | 1 | 7.79 | 7.79 | https://www.amazon.com/AmazonBasics-Male-Micro-Cable-Black/dp/B07232M876/ |
| power bank | PW1 | 1 | 19.99 | 19.99 | https://www.amazon.it/OKZU-10000mAh-Caricabatterie-Portatile-Powerbank/dp/B07JNSXRP2 |

## 5. Build instructions

- *Generate networks:*

1. Load on your Google Drive the notebook files: CNN_Esca.ipynb, CNN_Esca_pruned.ipynb, CNN_PV.ipynb, CNN_PV_pruned.ipynb.
2. Load on your Google Drive the augmented_esca_dataset_9transformation_splitted.zip to be used with the CNN_Esca.ipynb and CNN_Esca_pruned.ipynb files in order to train, prune and then retrain the Esca model.
3. Load on your Google Drive the PlantVillage-augmented Dataset from [26] to be used with CNN_PV.ipynb and CNN_PV_pruned.ipynb files in order to train, prune and then retrain PlantVillage model.
4. Open the file CNN_Esca.ipynb with Google Colaboratory, choose runtime session from the Menu and execute. Repeat this step for the same procedure for the file CNN_Esca_pruned.ipynb.
5. The output model files will be saved in the same directory of the notebook file, download them for the next steps.
6. Repeat steps 4 and 5, but using CNN_PV.ipynb and CNN_PV_pruned.ipynb files, to obtain the CNN models for the PlantVillage-augmented dataset.

- *Generate OpenMV Cam application through MicroPython code:*

1. *Generate MicroPython code:*

1.1 Open the OpenMV IDE to test the scripts openmv_demoEsca_SD_testingSet.py, openmv_demoEsca_SD_realtime.py and openmv_demoPV_SD_testingSet.py, openmv_demoPV_SD_realtime.py in oder to change, if necessary, the path of ESCA and PlantVillage models respectively and to test the model performance on board.
Using the scripts openmv_demoEsca_SD_testingSet.py and openmv_demoPV_SD_testingSet.py the board makes the predictions on the test sets Dataset_Esca_Test_bmp_128x128.zip and Dataset _PV_Test_bmp_128x128.zip respectively, both saved on SD together with the corresponding tflite models CNN_Esca_model.tflite and CNN_PV_model.tflite.
Using the scripts openmv_demoEsca_SD_realtime.py and openmv_demoPV_SD_realtime.py the board makes the predictions on the images acquired by the camera, dumping also the FPS and the inference time as well as the accuracy for each image.
Using the script openmv_demoEsca_SD_realtime_responseOnLCD.py a feedback for the detection of an unhealthy leaf has shown in the LCD display.

2. *Setting up OpenMV Cam for MicroPython code:*

2.1 Once chosen the more appropriate script for your application, save the selected script on the SD card renaming it as main.py. This script will be executed when the board starts.
2.2 Power the board with a power bank connected with a micro USB cable in order to have your image detector.

## 6. Operation instructions

Please refer to Hardware description and Build instructions sections, specifically to step *Setting up OpenMV Cam for MicroPython code* in Build instructions section.

## 7. Validation and characterization

In order to validate the image detector for real-time plant disease detection previously discussed, two different experiments, *i)* the experiment on ESCA-dataset [44] and *ii)* the experiment on the PlantVillage-augmented dataset [7,26], were conducted. The first aims to validate the image detector performances in a binary classification and the second aims to determine that, with the same model, the system shows good performances also for the multi-class classification task.

The experiments were conducted using the TensorFlow v. 2.5.0 and Keras v. 2.5.0 to train the models on Google Colaboratory with the ESCA-dataset and the PlantVillage-augmented datasets partitioned as reported in Table 2 and Table 3 respectively, the STM32Cube.AI extension X-CUBE-AI v. 7.1.0 [4] to analyse the generated models, and the OpenMV firmware v. 4.1.1 on board.

[4] https://www.st.com/en/embedded-software/x-cube-ai.html

**Table 2**
Consistency of the Esca dataset partition considered for training, validation and testing.

| Category | Training Samples 60% | Validation Samples 15% | Testing Samples 25% | Total samples |
|---|---|---|---|---|
| esca | 5328 | 1332 | 2220 | 8880 |
| healthy | 5292 | 1323 | 2205 | 8820 |
| Total | 10620 | 2655 | 4425 | 17700 |

**Table 3**
Consistency of the PlantVillage-augmented dataset partition considered for training, validation and testing.

| Category | Training Samples 60% | Validation Samples 15% | Testing Samples 25% | Total samples |
|---|---|---|---|---|
| Apple_scab | 600 | 150 | 250 | 1000 |
| Apple_black_rot | 600 | 150 | 250 | 1000 |
| Apple_cedar_apple_rust | 600 | 150 | 250 | 1000 |
| Apple_healthy | 987 | 246 | 412 | 1645 |
| Background_without_leaves | 685 | 171 | 287 | 1143 |
| Blueberry_healthy | 901 | 225 | 376 | 1502 |
| Cherry_powdery_mildew | 631 | 157 | 264 | 1052 |
| Cherry_healthy | 600 | 150 | 250 | 1000 |
| Corn_gray_leaf_spot | 600 | 150 | 250 | 1000 |
| Corn_common_rust | 715 | 178 | 299 | 1192 |
| Corn_northern_leaf_blight | 600 | 150 | 250 | 1000 |
| Corn_healthy | 697 | 174 | 291 | 1162 |
| Grape_black_rot | 708 | 177 | 295 | 1180 |
| Grape_black_measles | 829 | 207 | 347 | 1383 |
| Grape_leaf_blight | 645 | 161 | 270 | 1076 |
| Grape_healthy | 600 | 150 | 250 | 1000 |
| Orange_haunglongbing | 3304 | 826 | 1377 | 5507 |
| Peach_bacterial_spot | 1378 | 344 | 575 | 2297 |
| Peach_healthy | 600 | 150 | 250 | 1000 |
| Pepper_bacterial_spot | 600 | 150 | 250 | 1000 |
| Pepper_healthy | 886 | 221 | 371 | 1478 |
| Potato_early_blight | 600 | 150 | 250 | 1000 |
| Potato_late_blight | 600 | 150 | 250 | 1000 |
| Potato_healthy | 600 | 150 | 250 | 1000 |
| Raspberry_healthy | 600 | 150 | 250 | 1000 |
| Soybean_healthy | 3054 | 763 | 1273 | 5090 |
| Squash_powdery_mildew | 1101 | 275 | 459 | 1835 |
| Strawberry_leaf_scorch | 665 | 166 | 250 | 1081 |
| Strawberry_healthy | 600 | 150 | 278 | 1028 |
| Tomato_bacterial_spot | 1276 | 319 | 532 | 2127 |
| Tomato_early_blight | 600 | 150 | 250 | 1000 |
| Tomato_late_blight | 1145 | 286 | 250 | 1681 |
| Tomato_leaf_mold | 600 | 150 | 444 | 1194 |
| Tomato_septoria_leaf_spot | 1062 | 265 | 420 | 1747 |
| Tomato_spider_mites_two-spotted_spider_mite | 1005 | 251 | 352 | 1608 |
| Tomato_target_spot | 842 | 210 | 1340 | 2392 |
| Tomato_yellow_leaf_curl_virus | 3214 | 803 | 399 | 4416 |
| Tomato_mosaic_virus | 600 | 150 | 250 | 1000 |
| Tomato_healthy | 954 | 238 | 478 | 1670 |
| Total | 36884 | 9213 | 15389 | 61486 |

### 7.1. Datasets

*ESCA-dataset:* The ESCA-dataset [44] containing 1770 photographs of the leaves of healthy and infected plants was used for the training, validation and testing of the CNN architecture. The sizes of the acquired images are $1920 \times 1080$ and $1280 \times 720$ pixels with random portrait and landscape orientation. To enhance the size and quality of training dataset a data augmentation technique has been adopted, by using geometric transformations (horizontal and vertical flip, rotation, width and height shift), color transformations (brightness, contrast, saturation, hue, gamma), plus other image manipulations like zoom and blur. The classes and the consistency of the Esca dataset used in this experiment are reported in Table 2.

*PlantVillage-augmented dataset:* The PlantVillage dataset [7] consists of 54303 healthy and unhealthy leaf images divided into 38 categories by species and disease. The original dataset is not yet available from the original source [5], therefore it can be downloaded from several well-known online repository, such as GitHub [6] and Kaggle [7]. Moreover, an augmented version of this dataset can be used directly by the `plant_village` struct embedded in TensorFlow [8] or can be downloaded from the source Mendeley Data repository [26]. Particularly, the dataset in [26] has been presented by Geetharamani et al. in [25], where the authors, starting from the original PlantVillage dataset, realize an augmented version of this dataset applying several data augmentation techniques in order to improve classification performance and propose a simple nine-layer CNN specifically designed for this dataset. Authors use six different data augmentation techniques for increasing the data-set size: image flipping, Gamma correction, noise injection, PCA color augmentation, rotation, and scaling. In the final augmented data-set, 39 different classes of plant leaf and background images are available, for a total of 61486 images. In this work, the aforementioned version of the PlantVillage dataset, which for simplicity will be called PlantVillage-augmented dataset, has been chosen and a comparison with the simple CNN proposed by Geetharamani et al. has been performed, to validate the proposed approach. The classes and the consistency of the PlantVillage-augmented dataset used in the experimentation are reported in Table 3.

### 7.2. CNN architecture

The starting CNN architecture is depicted in Fig. 3 and comprises 5 weight layers in total: it consists of 3 convolutional layers each followed by a ReLU activation function and a max-pooling operation, and 2 fully-connected layers with a final softmax classifier. A detailed description of this simple architecture is provided in Table 4. The network was trained for 30 epochs on the previously described ESCA-dataset and PlantVillage-augmented dataset with images of size $128 \times 128$, by using an Adadelta optimizer with categorical cross entropy, a learning rate of 0.5 and a batch size of 64.

Despite the high number of classes, the simple proposed network can be efficiently applied also to the Plant Village dataset. Furthermore a pruning compression technique was applied to the proposed CNN in order to implement this network on the OpenMV Cam H7 constrained-resource system, maintaining a good accuracy.

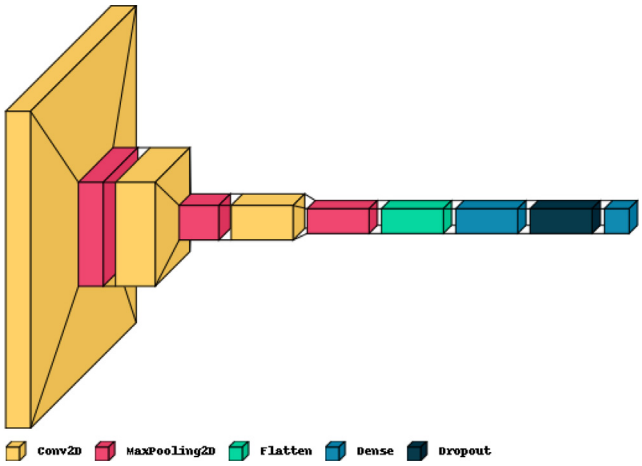



**Fig. 3.** Design TensorFlow of CNN architecture.

### 7.3. CNN pruning

The previously trained network was then compressed with a pruning method in order to reduce computational complexity and to obtain acceptable values for storage cost and inference time on embedded platform.

A filter pruning technique was used with a similar approach as in [43]. For each layer the filters with the lowest total norms were discarded by consequently pruning the corresponding input channels of the subsequent layer. The finetuning was simply performed in a single step without freezing any layer.

Details about model compression and finetuning for Esca and Plant Village datasets can be found in Tables 5, 6. The final pruned network architecture is described in Table 7.

[5] https://plantvillage.psu.edu/
[6] https://github.com/spMohanty/PlantVillage-Dataset
[7] https://www.kaggle.com/emmarex/plantdisease
[8] https://www.tensorflow.org/datasets/catalog/plant_village

**Table 4**
CNN architecture in detail.

| Type | Filter shape | Input size | Number of parameters |
|---|---|---|---|
| conv_1 | $3 \times 3 \times 3 \times 16$ | $128 \times 128 \times 3$ | 448 |
| relu_1 | – | $128 \times 128 \times 16$ | 0 |
| maxpool_1 ($3 \times 3$) | – | $128 \times 128 \times 16$ | 0 |
| conv_2 | $3 \times 3 \times 16 \times 32$ | $42 \times 42 \times 16$ | 4640 |
| relu_2 | – | $42 \times 42 \times 32$ | 0 |
| maxpool_2 ($3 \times 3$) | - | $42 \times 42 \times 32$ | 0 |
| conv_3 | $3 \times 3 \times 32 \times 64$ | $14 \times 14 \times 32$ | 18496 |
| relu_3 | – | $14 \times 14 \times 64$ | 0 |
| maxpool_3 ($2 \times 2$) | – | $14 \times 14 \times 64$ | 0 |
| flatten | – | $7 \times 7 \times 64$ | 0 |
| dense_1 | $3136 \times 512$ | $1 \times 3136$ | 1606144 |
| relu_4 | – | $1 \times 512$ | 0 |
| dropout (0.5) | – | $1 \times 512$ | 0 |
| dense_2 | $512 \times 2/512 \times 39^*$ | $1 \times 512$ | $1026/20007^*$ |
| softmax | – | $1 \times 2/1 \times 39^*$ | 0 |

* Different number of parameters for the architecture applied to ESCA and PlantVillage-augmented dataset, since the dense layer depends on the number of classes (2 and 39 classes respectively).

**Table 5**
Compression factors for the pruned models.

| | Esca dataset | Plant Village dataset |
|---|---|---|
| conv_1 | 0.35 | 0.35 |
| conv_2 | 0.5 | 0.5 |
| conv_3 | 0.5 | 0.5 |
| dense_1 | 0.9 | 0.9 |

**Table 6**
Details for the finetuning of the pruned models.

| | Esca dataset | Plant Village dataset |
|---|---|---|
| pruning method | $l_2$ norm | $l_1$ norm |
| optimizer | Adadelta | Adadelta |
| learning rate | 1.0 | 1.0 |
| epochs | 20 | 25 |

**Table 7**
Pruned CNN architecture.

| Type | Filter shape | Input size | Number of parameters |
|---|---|---|---|
| conv_1 | $3 \times 3 \times 3 \times 5$ | $128 \times 128 \times 3$ | 140 |
| relu_1 | – | $128 \times 128 \times 5$ | 0 |
| maxpool_1 ($3 \times 3$) | – | $128 \times 128 \times 5$ | 0 |
| conv_2 | $3 \times 3 \times 5 \times 16$ | $42 \times 42 \times 5$ | 736 |
| relu_2 | – | $42 \times 42 \times 16$ | 0 |
| maxpool_2 ($3 \times 3$) | – | $42 \times 42 \times 16$ | 0 |
| conv_3 | $3 \times 3 \times 16 \times 32$ | $14 \times 14 \times 16$ | 4640 |
| relu_3 | – | $14 \times 14 \times 32$ | 0 |
| maxpool_3 ($2 \times 2$) | – | $14 \times 14 \times 32$ | 0 |
| flatten | – | $7 \times 7 \times 32$ | 0 |
| dense_1 | $1568 \times 460$ | $1 \times 1568$ | 721740 |
| relu_4 | – | $1 \times 460$ | 0 |
| dropout (0.5) | – | $1 \times 460$ | 0 |
| dense_2 | $460 \times 2/460 \times 39^*$ | $1 \times 460$ | $920/17979^*$ |
| softmax | – | $1 \times 2/1 \times 39^*$ | 0 |

* Different number of parameters for the architecture applied to ESCA and PlantVillage-augmented dataset, since the dense layer depends on the number of classes (2 and 39 classes respectively).

### 7.4. Results

To show the validity of the described models, firstly the performance achieved on PC has been considered and then the model has been evaluated on the chosen embedded system, the OpenMV Cam H7 Plus.

Regarding the performance on PC, Table 8 and Table 9 report the results achieved by the initial and the pruned CNN in terms of storage cost, number of parameters, testing accuracy, inference time, MACC (multiply-accumulate operation), ROM Bytes and RAM Bytes, on ESCA and PlantVillage-augmented dataset respectively. Moreover, as reported in Table 10 and Table 11, a detailed analysis of the implemented models has been performed using the STM32Cube.AI extension X-CUBE-AI v.7.1.0. Specifically, MACC, ROM Bytes and RAM Bytes have been computed using the X-CUBE-AI to analyse the models using in the terminal the following command:

- `$> stm32ai analyze -m $model_name.$ext --allocate-inputs --allocate-outputs -o $destination_dir`

where `$model_name` is the name of the model to analyze, `$ext` the extension of the file (h5 or tflite) and `$destination_dir` is the destination directory chosen to save the log file.

Regarding the performance on board Table 12 and Table 13 report the results achieved for testing accuracy, inference time and frame per second (FPS), on ESCA and PlantVillage-augmented dataset respectively.

As you can see, the pruned model preserves a high accuracy for both binary and multi-class classification.

**Table 8**
Model performance on PC — ESCA-dataset.

| | Model type | Input size | Memory cost [*KB*] | Test accuracy [%] | Inference time [ms/img] |
|---|---|---|---|---|---|
| CNN | h5 | 128 × 128 | 19164.75 | 97.88 | 53.22 |
| | tflite | 128 × 128 | 1602.555 | 97.90 | 75.45 |
| Pruned CNN | h5 | 128 × 128 | 5739.289 | 97.79 | 51.95 |
| | tflite | 128 × 128 | 718.961 | 97.69 | 19.94 |

**Table 9**
Model performance on PC — PlantVillage-augmented dataset.

| | Model type | Input size | Memory cost [KB] | Test accuracy [%] | Inference time [ms/img] |
|---|---|---|---|---|---|
| CNN | h5 | 128 × 128 | 19386.688 | 96.52 | 52.93 |
| | tflite | 128 × 128 | 1621.195 | 96.50 | 78.67 |
| Pruned CNN | h5 | 128 × 128 | 8786.457 | 95.87 | 44.44 |
| | tflite | 128 × 128 | 735.727 | 95.72 | 22.70 |

**Table 10**
Details of the model analyzed with X-CUBE-AI — ESCA-dataset.

| | Model type | Parameters number | MACC | ROM Bytes | RAM Bytes |
|---|---|---|---|---|---|
| CNN | h5 | 1630754 | 21081040 | 6523016 | 334088 |
| | tflite | 1630754 | 20749400 | 1632632 | 88515 |
| Pruned CNN | h5 | 728178 | 5338585 | 2912712 | 242000 |
| | tflite | 728178 | 5221717 | 729724 | 62315 |

**Table 11**
Details of the model analyzed with X-CUBE-AI — PlantVillage-augmented dataset.

| | Model type | Parameters number | MACC | ROM Bytes | RAM Bytes |
|---|---|---|---|---|---|
| CNN | h5 | 1649735 | 21100576 | 6598940 | 334236 |
| | tflite | 1649735 | 20769084 | 1651724 | 88552 |
| Pruned CNN | h5 | 745235 | 5356197 | 2980940 | 242148 |
| | tflite | 745235 | 5239477 | 746892 | 62352 |

**Table 12**
Model performance on OpemMV Cam H7 Plus — ESCA-dataset.

| | Model type | Input size | Test accuracy [%] | Inference time [ms/img] | FPS |
|---|---|---|---|---|---|
| CNN | tflite | 128 × 128 | 98.40 | 276.404 | 3.61 |
| Pruned CNN | | 128 × 128 | 98.10 | 122.969 | 8.13 |

**Table 13**
Model performance on OpemMV Cam H7 Plus — PlantVillage-augmented dataset.

| | Model type | Input size | Test accuracy [%] | Inference time [ms/img] | FPS |
|---|---|---|---|---|---|
| CNN | tflite | 128 × 128 | ***96.24*** | 283.30 | 3.52 |
| Pruned CNN | | 128 × 128 | 95.24 | ***125.63*** | ***7.95*** |
| Geetharamani et al. [25] | tflite | 128 × 128 | 94.34 | 270.61 | 3.69 |

Meanwhile a considerable reduction is obtained for storage cost, inference time and FPS for the tflite models implemented on the OpenMV Cam thus obtaining a suitable model for embedded platform applications.

In addition, a comparison with a state-of-the-art lightweight CNN for plant disease recognition by Geetharamani et al. [25] has been shown in Table 13, showing the better performance achieved by the proposed pruned CNN in terms of accuracy and inference time.

A further experimentation has been conducted by testing the models trained on the CNNs trained on ESCA and PlantVillage-augmented dataset on a different testing set realized following the procedure of Mohanty et al. [4]. In this work, in order to assess the performance of the model on a "real world" data set, the authors downloaded images from Bing Image performing a search with the query "{crop and disease name} leaf leaves", where {crop and disease name} was replaced by the crop and disease name pairs given for each of the PlantVillage dataset classes. Following this procedure, firstly a research on Bing Image (on July 4, 2022) has been conducted to download the top 10 images for each class (2 classes for ESCA and 39 for PlantVillage-augmented models) and then a visual verification step has been made to verify that the image was by a reputable source and that it was showing leaves in approximately the same configuration. The final testing set so realized consists of 20 and 390 images to test the Esca leaf disease and the plant leaf diseases through the models trained on ESCA and PlantVillage-augmented dataset (both pruned and not pruned). Authors show that this test leads to an accuracy drop from 99.34% to about 31.69%, using the best model trained on PlantVillage dataset. This is reasonable since the PlantVillage dataset contains images collected in a controlled environment unlike the new images that contains different background, such as fruits and stems. Also in the proposed work, the accuracy drops from 96.24% to 36.41% with the not-pruned CNN and from 95.24% to 30.51% with the pruned CNN, both tested on 39 classes, testing the models on OpenMV H7 Cam board. However, the achieved accuracy is better than that obtained using the model Geetharamani et al. [25] with this testing set: 27.69%. The experimental results show that the proposed models trained on ESCA-dataset reaches a lower accuracy drop: from 98.40% to 51.00% with the original CNN and from 98.10% to 50.00% with the pruned CNN, both tested on 2 classes and with the models embedded on board. In this case the drop is mitigated by the fact that, unlike the PlantVillage-augmented dataset, the ESCA-dataset contains images that has been directly taken in-place by different cameras with various resolutions, considering multiple lighting brightness, different sides and backgrounds, i.e. including not only single healthy/infected leaf but also groups of leaves and other parts of the plant, such as stems. Future works regard the improvement of this aspect in order to deploy the CNN-based plant leaf diseases detector in real scenarios without loss in accuracy.

## Ethics statements

*The work did not involve any human or animal subjects, nor data from social media platforms.*

## CRediT author statement

**Laura Falaschetti***: Conceptualization, Methodology, Investigation, Data curation, Software, Validation, Writing - Original Draft, Writing - Review & Editing, Visualization;* **Lorenzo Manoni***: Conceptualization, Methodology, Investigation, Software, Validation, Writing - Review & Editing, Visualization;* **Denis Di Leo***: Data curation, Software, Validation;* **Danilo Pau***: Conceptualization, Methodology, Investigation, Writing - Review & Editing, Visualization, Supervision, Project administration, Funding acquisition;* **Valeria Tomaselli***: Conceptualization, Writing - Review & Editing, Visualization, Funding acquisition;* **Claudio Turchetti***: Conceptualization, Methodology, Investigation, Writing - Review & Editing, Visualization, Supervision, Project administration.*

## Declaration of Competing Interest

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.

## Acknowledgements

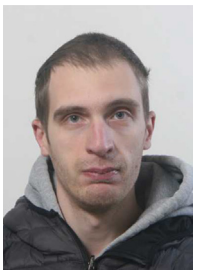
*This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.*

## References


[1] S.D. Khirade, A.B. Patil, Plant disease detection using image processing, in: 2015 International Conference on Computing Communication Control and Automation, 2015, pp. 768–771, https://doi.org/10.1109/ICCUBEA.2015.153.
[2] E.M.F. El Houby, A survey on applying machine learning techniques for management of diseases, J. Appl. Biomed. 16 (3) (2018) 165–174, https://doi.org/10.1016/j.jab.2018.01.002.
[3] J. Boulent, S. Foucher, J. Théau, P.-L. St-Charles, Convolutional neural networks for the automatic identification of plant diseases, Front. Plant Sci. 10 (2019) 941, https://doi.org/10.3389/fpls.2019.00941.
[4] S.P. Mohanty, D.P. Hughes, M. Salathé, Using deep learning for image-based plant disease detection, Front. Plant Sci. 7 (2016) 1419, https://doi.org/10.3389/fpls.2016.01419.
[5] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in Advances in Neural Information Processing Systems, eds F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger (Curran Associates, Inc.), (2012), 1097–1105. doi: 10.1145/3065386..
[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, https://doi.org/10.48550/arXiv.1409.4842.
[7] D.P. Hughes, M. Salathe, An open access repository of images on plant health to enable the development of mobile disease diagnostics. arXiv preprint arXiv:1511.08060, 2015, URL:http://arxiv.org/abs/1511.08060.
[8] A. Ramcharan, K. Baranowski, P. McCloskey, B. Ahmed, J. Legg, D.P. Hughes, Deep Learning for Image-Based Cassava Disease Detection, Front. Plant Sci. 8 (2017) 1852, https://doi.org/10.3389/fpls.2017.01852.
[9] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (Las Vegas), 2016, pp. 2818–2826, https://doi.org/10.48550/arXiv.1512.00567.
[10] Cassava Disease Classification Dataset. URL: https://www.kaggle.com/competitions/cassava-disease/data, 2019 (accessed 06.07.22)..
[11] 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) 2022, https://doi.org/10.3390/s17092022.
[12] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, IEEE Trans. Pattern Anal. Mach. Intell. 39 (2016) 1137–1149, https://doi.org/10.1109/TPAMI.2016.2577031.
[13] J. Dai, Y. Li, K. He, J. Sun, R-FCN: Object Detection via Region-based Fully Convolutional Networks. arXiv (2016). doi: 10.48550/arXiv.1605.06409..
[14] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, A.C. Berg, SSD: Single Shot MultiBox Detector, in: Proceedings of the European Conference on 9905, Computer Vision–ECCV, Amsterdam, The Netherlands, 2016, pp. 21–37, https://doi.org/10.1007/978-3-319-46448-0_2.
[15] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv (2014), https://doi.org/10.48550/arXiv.1409.1556.
[16] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the 2016 IEEE Conference on Computer, Vision, Pattern Recognition, Las Vegas, NV, USA, 2016, pp. 770–778, https://doi.org/10.1109/CVPR.2016.90.
[17] P. Pawara, E. Okafor, O. Surinta, L. Schomaker, M. Wiering, Comparing local descriptors and bags of visual words to deep convolutional neural networks for plant recognition, International Conference on Pattern Recognition Applications and Methods, SciTePress 2 (2017) 479–486, https://doi.org/10.5220/0006196204790486.
[18] P. Pawara, E. Okafor, M. Groefsema, S. He, L. Schomaker, M. Wiering, AgriPlant Dataset. URL: https://www.ai.rug.nl/p.pawara/dataset.php, 2017 (accessed 06.07.22)..
[19] N. Kumar, P.N. Belhumeur, A. Biswas, D.W. Jacobs, W.J. Kress, I.C. Lopez, J.V.B. Soares, Leafsnap: A Computer Vision System for Automatic Plant Species Identification. Computer Vision – ECCV 2012 Lecture Notes in Computer Science, 7573, (2012). Springer, Berlin, Heidelberg. doi: 10.1007/978-3-642-33709-3_36..
[20] T. Munisami, M. Ramsurn, S. Kishnah, S. Pudaruth, Plant Leaf Recognition Using Shape Features and Colour Histogram with K-nearest Neighbour Classifiers, Procedia Comput. Sci. 58 (2015) 740–747, https://doi.org/10.1016/j.procs.2015.08.095.
[21] K.P. Ferentinos, Deep learning models for plant disease detection and diagnosis, Comput. Electron. Agricult. 145 (2018) 311–318, https://doi.org/10.1016/j.compag.2018.01.009.
[22] A. Krizhevsky, One weird trick for parallelizing convolutional neural networks (2014), https://doi.org/10.48550/arXiv.1404.5997.
[23] A. Ramcharan, P. McCloskey, K. Baranowski, N. Mbilinyi, L. Mrisho, M. Ndalahwa, J. Legg, D.P. Hughes, A mobile-based deep learning model for cassava disease diagnosis, Front. Plant Sci. 10 (2019) 272, https://doi.org/10.3389/fpls.2019.00272.
[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) doi: 10.48550/arXiv.1704.04861..
[25] G. Geetharamani, A. Pandian J., Identification of plant leaf diseases using a nine-layer deep convolutional neural network. Comput. Electr. Eng., 76 (2019) 323-338. doi: 10.1016/j.compeleceng.2019.04.011..
[26] A. Pandian J., G. Geetharamani, Data for: Identification of Plant Leaf Diseases Using a 9-layer Deep Convolutional Neural Network. Mendeley Data, V1, (2019) doi: 10.17632/tywbtsjrjv.1.
[27] J. Chen, J. Chen, D. Zhang, Y. Sun, Y.A. Nanehkaran, Using deep transfer learning for image-based plant disease identification, Comput. Electron. Agricul. 173 (2020), https://doi.org/10.1016/j.compag.2020.105393 105393.
[28] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1251–1258, https://doi.org/10.1109/CVPR.2017.195.
[29] J. Chen, W. Wang, D. Zhang, A. Zeb, Y.A. Nanehkaran, Attention embedded lightweight network for maize disease recognition, Plant. Pathol. 70 (3) (2021) 630–642, https://doi.org/10.1111/ppa.13322.
[30] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely Connected Convolutional Networks, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2261–2269, https://doi.org/10.1109/CVPR.2017.243.
[31] J. Chen, D. Zhang, M. Suzauddola, A. Zeb, Identifying crop diseases using attention embedded MobileNet-V2 model, Appl. Soft Comput. 113 (2021), https://doi.org/10.1016/j.asoc.2021.107901 107901.
[32] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, MobileNetV2: Inverted Residuals and Linear Bottlenecks, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520, https://doi.org/10.1109/CVPR.2018.00474.
[33] J. Chen, A. Zeb, S. Yang, D. Zhang, Y.A. Nanehkaran, Automatic identification of commodity label images using lightweight attention network, Neural Comput. Appl. 33 (2021) 14413–14428, https://doi.org/10.1007/s00521-021-06081-9.
[34] Y. Li, J. Yang, Few-shot cotton pest recognition and terminal realization, Comput. Electron. Agricult. 169 (2020), https://doi.org/10.1016/j.compag.2020.105240 105240.
[35] National Bureau of Agricultural Insect Resources (NBAIR). URL: https://www.nbair.res.in/index.php/databases, 2015 (accessed 06.07.22)..
[36] J. Chen, W. Chen, A. Zeb, D. Zhang, Y.A. Nanehkaran, Crop pest recognition using attention-embedded lightweight network under field conditions, Appl. Entomol. Zool. 56 (2021) 427–442, https://doi.org/10.1007/s13355-021-00732-y.
[37] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, A. Torralba, Learning Deep Features for Discriminative Localization, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2921–2929, https://doi.org/10.1109/CVPR.2016.319.
[38] J. Chen, J. Chen, D. Zhang, Y.A. Nanehkaran, Y. Sun, A cognitive vision method for the detection of plant disease images, Mach. Vis. Appl. 32 (2021) 31, https://doi.org/10.1007/s00138-020-01150-w.
[39] S. Mishra, R. Sachan, D. Rajpal, Deep Convolutional Neural Network based Detection System for Real-time Corn Plant Disease Recognition, Procedia Comput. Sci. 167 (2020) 2003–2010, https://doi.org/10.1016/j.procs.2020.03.236.

[40] R. Gajjar, N. Gajjar, V.J. Thakor, N.P. Patel, S. Ruparelia, Real-time detection and identification of plant leaf diseases using convolutional neural networks on an embedded platform, Visual Comput. (2021), https://doi.org/10.1007/s00371-021-02164-9.
[41] J. Chen, D. Zhang, S. Yang, Y.A. Nanehkaran, Intelligent monitoring method of water quality based on image processing and RVFL-GMDH model, IET Image Proc. 14 (2020) 4646–4656, https://doi.org/10.1049/iet-ipr.2020.0254.
[42] L. Falaschetti, L. Manoni, R. Calero Fuentes Rivera, D. Pau, G. Romanazzi, O. Silvestroni, V. Tomaselli, C. Turchetti, A Low-Cost, Low-Power and Real-Time Image Detector for Grape Leaf Esca Disease Based on a Compressed CNN. IEEE J. Emerg. Select. Top. Circuits Syst., 11(3), (2021) 468-481. doi: 10.1109/JETCAS.2021.3098454..
[43] H. Li, A. Kadav, I. Durdanovic, H. Samet, H.P. Graf, Pruning filters for efficient ConvNets, 2017, URL:https://arxiv.org/abs/1608.08710v3.
[44] M. Alessandrini, R. Calero Fuentes Rivera, L. Falaschetti, D. Pau, V. Tomaselli, C. Turchetti, A grapevine leaves dataset for early detection and classification of esca disease in vineyards through machine learning. Data in Brief, 35, (2021) 106809. doi: 10.1016/j.dib.2021.106809..

**Laura Falaschetti** received the B.Sc., the M.Sc. and the Ph.D. degree in electronics engineering from the Universitá Politecnica delle Marche, Ancona, Italy, in 2008, 2012 and 2016 respectively. Since 2017 she is a Postdoctoral Research Fellow at the Department of Information Engineering (DII), Universitá Politecnica delle Marche. She is currently a Researcher and an Assistant Professor of Electronic Systems with the DII, Universitá Politecnica delle Marche. Her current research interests include: embedded systems, machine learning, neural networks, manifold learning, pattern recognition, signal processing, image processing, speech processing, bio-signal analysis.



**Lorenzo Manoni** received the B.Sc., the M.Sc. and the Ph.D. degrees in electronics engineering from the Universitá Politecnica delle Marche, Ancona, Italy, in 2015, 2018 and 2022 respectively. His current research interests include signal processing, embedded systems, machine learning, algorithms analysis and design, bio-signal analysis.

**Denis Di Leo** received the B.Sc. degree in electronics engineering from the Universitá Politecnica delle Marche, Ancona, Italy, working on the development of convolutional neural networks for image classification. He is currently pursuing the M.Sc. degree at the same University.

**Danilo Pau** One year before graduating from the Politecnico di Milano in 1992, Danilo Pau joined STMicroelectronics, where he worked on HDMAC and MPEG2 video memory reduction, video coding, embedded graphics, and computer vision. Today, his work focuses on developing solutions for deep learning tools and applications. Since 2019 Danilo is an IEEE Fellow. Currently serves as member of IEEE Region 8 Action for Industry and Member of the Machine Learning, Deep Learning and AI in the CE (MDA) Technical Stream Committee IEEE Consumer Electronics Society (CESoc). With over 80 patents, 104 publications, 113 MPEG authored documents and 39 invited talks/seminars at various worldwide Universities and Conferences, Danilo's favorite activity remains mentoring undergraduate students, MSc engineers and PhD students from various universities.

**Valeria Tomaselli** is senior engineer and Project Leader at STMicroelectronics of Catania. She holds a Master's Degree in Computer Engineering from the University of Catania. Since 2003 she has been working at STMicroelectronics, in the System Research and Applications group, where she conducted research activities and developed application solutions in the fields of image processing and computer vision. Her current projects focus on machine learning, deep learning and artificial intelligence applications and tools. She is the author of patents and papers on image processing, computer vision and artificial intelligence. She has also participated in numerous national and international research projects.

**Claudio Turchetti** received the Laurea degree in electronics engineering from the University of Ancona, Ancona, Italy, in 1979. He joined the Universitá Politecnica delle Marche, Ancona, in 1980, where was the Head of the Department of Electronics, Artificial Intelligence and Telecommunications for five years and is currently a Full Professor of micro-nanoelectronics and design of embedded systems. His current research interests include: statistical device modeling, RF integrated circuits, device modeling at nanoscale, computational intelligence, signal processing, pattern recognition, system identification, machine learning and neural networks. He has published more than 160 journal and conference papers, and two books. The most relevant papers were published in IEEE J. of Solid-State Circuits, IEEE Trans. on Electron Devices, IEEE Trans. on CAD of IC's and Systems, IEEE Trans. on Neural Networks and Learning Systems, IEEE Trans. on Signal Processing, IEEE Trans. On Cybernetics, IEEE J. of Biomedical and Health Informatics, IEEE Trans. on Consumer Electronics, Information Sciences. He has held a variety of positions as Project Leader in several applied research programs developed in cooperation with small, large, and multinational companies in the field of microelectronics. Prof. Turchetti has served as a Program Committee Member for several conferences and as a reviewer of several scientific journals. He is a Member of the IEEE, Computational Intelligence and Signal processing Society. He has been an Expert Consultant of the Ministero dell'Universitá e Ricerca.