A Low-Cost, Low-Power and Real-Time Image Detector for Grape Leaf Esca Disease Based on a Compressed CNN

A Low-Cost, Low-Power and Real-Time Image Detector for Grape Leaf Esca Disease Based on a Compressed CNN

Laura Falaschetti, *Member, IEEE,* Lorenzo Manoni, Romel Calero Fuentes Rivera, Danilo Pau, *Fellow, IEEE,* Gianfranco Romanazzi, Oriana Silvestroni, Valeria Tomaselli, and Claudio Turchetti, *Life Member, IEEE*

***Abstract*—Esca is one of the most common grape leaf diseases that seriously affect grape yield, causing a loss of global production in the range of 20%-40%. Therefore, a timely and effective identification of the disease could help to develop an early treatment approach to control its spread while reducing economic losses. For this purpose the use of computer vision and machine learning techniques for recognizing plant diseases have been extensively studied in recent years. The aim of this paper is to propose an image detector based on a high-performance convolutional neural network (CNN) implemented in a low cost, low power platform, to monitor the Esca disease in real-time. To meet the severe constraints typical of an embedded system, a new low-rank CNN architecture (LR-Net) based on CANDECOMP/PARAFAC (CP) tensor decomposition has been developed. The compressed CNN network so obtained has been trained on a specific dataset and implemented in a low-power, low-cost Python programmable machine vision camera for real-time classification. An extensive experimentation has been conducted and the results achieved show the superiority of LR-Net with respect to the state-of-the-art networks both in terms of inference time and memory occupancy.**

***Index Terms*—Image detector, Esca disease, convolutional neural network, tensor decomposition, embedded systems.**

I. Introduction

THE vineyards have different degrees of internal variability [1], [2] due to the complex interactions that are established among the morphology of the soil, its chemical and biological fertility, the seasonal meteorological trends and the management techniques applied. These interactions influence the physiological performance of the grapevines, changing the balance between the vegetative and productive components [3], [4]. Thus, the conditions for the vegeto-productive imbalance in plants may arise, negatively affecting the production of good quality grapes and the resilience to biotic and abiotic stresses. The monitoring of spatial and/or temporal variability within vineyards was used to calibrate water [5] or nitrogen requirements [6], to define new protocols of sampling grapes for maturation monitoring [7] and lately to detect foliar symptoms of the Esca grapevine trunk disease [8]. Esca is one of the main grapevine diseases, able to induce severe symptoms and lead to the death of symptomatic plants worldwide [9], [10]. A list of fungal pathogens have been associated with the disease, and the most common are *Phaeomoniella chlamydospora*, *Phaeoacremonium ultimum* and *Fomitiporia mediterranea* [11]. External disease symptoms, including leaf stripes, interveinal chlorosis and necrosis, linear pitting on berries, and dessiccation of a portion or of the whole plant appears on the canopy, even in young vineyards few years after planting, and become increasingly severe with the aging of the plant [12]. Infected plants do not show foliar symptoms every years, and investigations run in the Marche region (Italy) on Verdicchio have indicated that around one third of plants showed symptoms three years out of three, one third two years out of three, and the last third just one year out of three. Moreover, the symptoms displayed are highly variable according to the cultivator and are affected by the rootstock [13], [14]. Therefore, having tools that are useful for the grower to monitor the progress of the disease in real-time every year can help to explain the correlation with other factors and apply mitigation strategies.

Besides, a common practice to manage trunk disease in infected vineyards is the pruning in autumn (early pruning) or in winter (late pruning), removing pruning from the vineyard or burning, in order to prevent the spread of disease [15], [16], [17]. This operation is particularly expensive if carried out by hand, thus a mechanical solution is strongly required in order to reduce the cost of this practice. To be effective such a system must be able to automatically detect the disease and the localization of the infected plants, thus solving two different problems: disease detection and geolocalization of plants. This paper focuses on the former problem alone, that is the grape leaf Esca disease detection.

The availability of a vision system, simple to use and that could be combined with autonomous agricultural vehicles, would offer the agronomist a valuable assistance for the plant disease detection and diagnosis, through optical observation of infected leaves.

In this scheme both the image sensor and the geolocalization system are mounted on a vehicle moving in the field, and they act simultaneously giving the coordinate and the status of each

L. Falaschetti, L. Manoni, R. Calero Fuentes Rivera and C. Turchetti are with the Dipartimento di Ingegneria dell'Informazione, Università Politecnica delle Marche, Via Brecce Bianche 12, I-60131 Ancona, Italy; e-mail: l.falaschetti@univpm.it, l.manoni@pm.univpm.it, S1077689@studenti.univpm.it, c.turchetti@univpm.it.

D. Pau is with the System Research and Applications, STMicroelectronics, Agrate Brianza, Italy; e-mail: danilo.pau@st.com.

G. Romanazzi and O. Silvestroni are with the Dipartimento di Scienze Agrarie, Alimentari e Ambientali, Università Politecnica delle Marche, Via Brecce Bianche 12, I-60131 Ancona, Italy; e-mail: g.romanazzi@univpm.it, o.silvestroni@univpm.it.

V. Tomaselli is with the System Research and Applications, STMicroelectronics, Catania, Italy; e-mail: valeria.tomaselli@st.com.

Manuscript received x xx, xxxx; revised x xx, xxxx.

plant. Thus one of the main requirements of the vision system is that it must be able to process the images on the fly, i.e. by processing the images of one plant before moving on to the next one. Although this requirement may be considered as too restrictive, another option to manage this problem is a batch solution in which the images and the positions are collected on-line, while the vehicle moves on the field, store them in a mass memory and subsequently process them off-line, e.g. using a PC. However this solution has some main drawbacks. Firstly, such image storage of several hours at 50 frames/sec, e.g. QVGA resolution, requires about 659 Mbytes of storage per minute. This amount of information would fill a 32 GB SD Card in 50 minutes, thus requiring multiple memory cards for a 4 hours standard working turn. For higher picture resolutions, storage increase linearly by a factor equal to the ratio between them and QVGA, becoming more and more challenging to be managed by the application. Secondly, and more importantly, the geolocalization of plants is inevitably affected by large error. Thus, while such an error suffices for an automated system to coarsely localize the infected plants, a more accurate localization, i.e. through a real-time image detector, is required for both selective pruning and grape harvest on-line in order to prevent damages on healthy plants.

In recent years, convolutional neural networks (CNNs) [18] have shown an impressive performance and at present they contribute to the state-of-the-art solution to most computer vision problems. The main reasons for this widespread adoption is that they take advantage of large image datasets and are able to discover the discriminative features directly from original images, avoiding complicated image processing. Thanks to those properties several plant disease classification models based on CNNs have recently been suggested.

In [19] the pre-trained AlexNet CNN model to learn unsupervised feature representations for 44 different plant species, has been used. A very simple CNN architecture that comprises two convolutional layers alone has been suggested in [20] for classifying three different legume species. The applicability of CNNs for plant disease detection has been studied in [21], by adopting AlexNet and GoogleLeNet architectures to identify 14 crop species and 26 diseases. Sladojevic et al. [22] trained a 5-layers CNN on a dataset with 30880 images, original and augmented, to recognize 13 different types of plant diseases out of healthy leaves. Scratch and fine-tuned versions of GoogleLeNet and AlexNet architectures have been considered in [23] to classify plants from AgrilPlant, LeafSnap, and Folio datasets. Three families of detectors (Faster R-CNN, R-FCN, SSD) combined with VGGNet and ResNet have been used in [24] to recognize nine different types of plant diseases and pests. Agarwal et al. [25] have suggested a CNN model with six convolutional layers to identify the diseases in grape plants in early stages by analysing the leaf images from Plant Village dataset. In [26] a study for the automatic identification of plant diseases has been conducted with several different architectures, i.e. LeNet, AlexNet, CaffeNet, ResNet, VGG, Inception, DenseNet and ResNeXt, on images achieved from public datasets. A deep-learning-based and faster DR-IACNN model with higher feature extraction capability that uses Inception modules and SE blocks is presented in [27] for grape leaf disease detection. Recently, Z. Tang et al. [28] have proposed a lightweight convolutional neural network model, using an improved ShuffleNet architecture that embeds a squeeze-and-excitation(SE) block into a ShuffleNet, for grape disease image classification.

All these networks are able to guarantee a good accuracy for plant diseases recognition, however they require significant memory and computational resources, which prohibit their usage in embedded devices. This is a serious limitation for the development of low-power, low-cost image sensors that can be mounted on agricultural vehicles, in order to meet restrictions such as form factor, power consumption and cost.

The aim of this paper is to propose an image detector based on a high-performance CNN implemented in a low cost, low power platform, to monitor Esca disease in real-time. The main contributions of the paper are summarized as follows:

- Several CNN compression techniques based on tensor decomposition are preliminarily analysed and a comparative study has been carried out to determine the method with the best performance.
- A new CNN architecture based on CANDECOMP/PARAFAC (CP) tensor decomposition has been developed, that decomposes each convolution layer as two 1×1 convolution at the input and output ends, and a depthwise convolution in the middle. The same technique has been used to decompose the fully connected layers to reduce the storage cost.
- The compressed CNN network previously defined has been implemented in the OpenMV Cam STM32H7 Plus platform, a low-power low-cost Python programmable machine vision camera, embodying a powerful microcontroller (MCU) for real-time classification.

The rest of the paper is organized as follows. Section II deals with problem statement and specifications. In Section III we summarize the related work and we state the motivations for our work. In Section IV several CNN compression techniques are studied and compared in order to determine the method with the best compression factor. Section V describes the dataset adopted in the experiments. In Section VI some details on the development of the compressed CNN architecture in the TensorFlow/Keras environment are given. Section VII reports the main features, hardware and software, of the CNN-based image sensor implemented in the low-power, low-cost OpenMV Cam STM32H7 Plus platform. Finally experimental results are presented in Section VIII.

II. Problem Statement and Specifications

The Fig. 1 depicts an automated vision system for the detection of Esca grapevine disease. In this scheme the image sensor we propose in this paper is mounted on an agricultural vehicle moving through the cultivation field at constant speed v (km/h). The image sensor samples images of the plant at the rate f (frame/sec). As each plant is located between two poles which are distanced L (meters) one from the other, then the camera is able to capture N_f images

$$N_f = f \times \frac{L}{v} \tag{1}$$

of each plant.

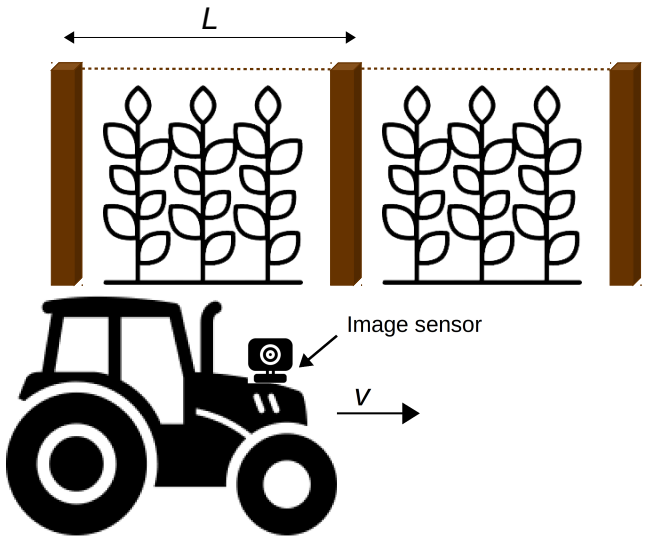


Fig. 1: The system for optical observation of the Esca disease on plant leaves.

Here we assume $v = 8$ km/h (2.2 m/sec), $f = 50$ frame/sec, $L = 3$ m, thus from (1) we have $N_f \cong 67$. Assuming each image is recognized by the sensor, then an inference time

$$t_{inf} < \frac{1}{f} = 20 \text{ msec} \tag{2}$$

would be required. However this specification constitutes a constraint too severe for the neural network to classify the images. A more realistic value for the number of images used for classification is $N_{inf} = kN_f$ with $k < 1$, which corresponds to an inference time

$$t_{inf} = \frac{1}{kf} \tag{3}$$

thus giving the following design constraint

$$t_{inf} = \frac{N_f}{N_{inf} \times f} = \frac{L}{N_{inf} \times v} \quad . \tag{4}$$

Choosing $N_{inf} = 20$ we obtain $t_{inf} = 68$ msec, which is one of the main constraints used in this paper in designing the sensor.

It is worth noticing that in this context the metric to measure the performance of plant disease recognition is the number of images recognized as true positive (TP) (meaning plant with disease) in the path between two consecutive poles. From the definition of accuracy we have

$$\text{accuracy} = (\text{TP} + \text{TN})/N_{inf} \tag{5}$$

where TN is the number of images recognized as true negative, i.e. with no disease. Setting TN=0 since for a plant attached by a disease no images classified as healthy occur, yields the following relationship

$$\text{TP} = \text{accuracy} \times N_{inf} \tag{6}$$

for the TP metric. This result clearly shows that in order to obtain the maximum value for TP, a compromise between accuracy and N_{inf} can be adopted.

III. Related Work

Convolutional neural networks (CNNs) are widely used for plant disease detection, however not all the architectures are suitable to be implemented in a low cost embedded MCU. In fact even though networks such as VGG and AlexNet are able to achieve a great learning accuracy, they require an exceeding inference cost both in terms of speed and memory footprint. For this reason in this paper we focus only on tiny architectures, i.e. nets that are able to learn a given dataset, can fit into scarce computing and memory resources made available by an MCU.

A. MobileNet

MobileNets are a class of efficient convolutional neural networks (CNNs) for mobile and embedded vision applications, that primarily focus on optimization for latency and small size networks. MobileNets V1 [29] is based on an architecture that uses depthwise separable convolution (DSC), which factorizes a standard convolution into two separate layers: depthwise convolution and 1×1 convolution. This factorization drastically reduces computation complexity and model size. MobileNet V2 [30] has been developed by observing that each layer in a CNN forms a manifold that could be embedded in a low-dimensional subspace. Thus in order to capture this property a novel layer module, the inverted residual with linear bottleneck, is inserted into the convolutional blocks. MobileNet V3 [31] uses a combination of two layers, called squeeze and excitation layers, into the bottleneck structure of the MobileNet V2 in order to optimize both accuracy and inference latency.

B. LeNet

The LeNet architecture [32] is one of the earliest CNN networks which was applied to MNIST digit dataset. It has a very basic structure formed, in the variant called LeNet-5, by 5 layers in total. Specifically, it consists of two convolutional layers each followed by a max-pooling operation and a single convolutional layer followed by a set of two fully connected layers towards the end of the model to act as a classifier on the extracted features.

C. SqueezeNet

SqueezeNet [33] achieves AlexNet-level accuracy on ImageNet with $50\times$ fewer parameters. Additionally, using model compression techniques, SqueezeNet can be compressed to less than 0.5 MB ($510\times$ smaller than AlexNet). To obtain these performances three strategies are adopted for designing the CNN architecture.

- *Strategy 1.* Replace 3×3 filters with 1×1 filters.
- *Strategy 2.* Decrease the number of input channels to 3×3 filters.
- *Strategy 3.* Downsample late in the network so that convolution layers have large activation maps.

D. ShuffleNet

ShuffleNet V1 [34] was designed specially for mobile devices with very limited computing power. On the basis of two operations, point wise group convolution and channel shuffle, the new architecture is able to greatly reduce computation cost while maintaining accuracy. The concept of group convolution was first introduced in AlexNet [35]. In this architecture the output of each stacked layer are distributed into several groups and each output channel only relates to the input channels of subsequent layers within the group. Channel shuffle operation is an efficient and elegant variant of group convolution to obtain input data from different groups, so that the input and output channels of two layers are fully related. ShuffleNet V2 [36] has been proposed to overcome some limitations of previously discussed techniques. In particular point wise group convolutions and bottleneck structure are two techniques adopted to increase the number of channels without significantly increasing complexity. A "channel shuffle" operation is then introduced to enable information communication between different groups of channels and to improve accuracy. However, both point wise group convolutions and bottleneck structure increase computational cost. ShuffleNet V2 introduces a simple operator called "channel shift" which splits the input feature channels into two branches. After convolution, the two branches are concatenated, so that the number of channels stay the same. ShuffleNet V2 is both efficient and accurate. Firstly because each building block enables using more feature channels and larger network capacity, secondly because half of feature channels directly join the next block. Several improved ShuffleNets architectures have recently been presented [28] to improve the performance of existing state-of-the-art models. Essentially these new architectures are based on the "squeeze-and-excitation" (SE) block [37], that adaptively recalibrates channel-wise feature response by explicitally modeling interdependencies between channels. Due to the flexibility of the SE block, it can be directly applied to existing network architectures, obtaining the new networks ShuffleNet V1 and ShuffleNet V2.

E. ResNet

ResNet architecture [38] is addressed to solve the degradation problem of deeper networks: with the network depth increasing, accuracy is saturated and then degrades rapidly. ResNet model solves this problem by introducing a 'deep residual learning' framework. More specifically this architecture is formed by a large number of stacked layers each of which asymptotically approximates a residual function instead of a generic mapping. In this way residual nets with a depth of up to 152 layers, $8\times$ deeper than that of VGG nets but still having lower complexity, have been evaluated [38] on ImageNet dataset.

F. PeleeNet

PeleeNet [39] is a variant of DenseNet [40], and it is designed to meet strict constraints on memory and computational budget. The key features of PeleeNet are: *i)* two-way dense layer to get different scales of receptive fields. One way uses a 3×3 kernel size, while the other uses two stacked 3×3 convolution to learn visual patterns for large objects. *ii)* Stem block before the first dense layer. This block improves the feature expression ability without increasing computational cost. *iii)* Dynamic number of channels in bottleneck layer, so that the number of channels in the bottleneck layer varies according to the input shape. PeleeNet achieved on ImageNet dataset an accuracy of 77%, higher than that of MobileNet by 2.1%, and it is only 66% of the model size of MobileNet.

G. Our Work

Even though all the networks previously discussed are based on lightweight architectures, that is with a reduced parameter set to learn, the performance achieved both in terms of storage cost and in particular in computational complexity do not suffice for embedded MCUs. With reference to the Esca grapevine disease recognition problem, the main goal of our work is to prove that an optimized CNN architecture can be implemented in an embedded platform to meet the requirements of low-power, low-cost and real-time detection. To our knowledge none of the previous works have addressed this problem, as they tested the CNN architectures on high performance processor units. Thus this paper presents for the first time a real-time image detector for Esca disease detection. To reach this goal we proceeded as follows.

- In order to reduce computational complexity the most common techniques for tensor decomposition have been analysed.
- A comparative study has been conducted to determine the method with the best compression factor.
- The method so derived, i.e. CP decomposition, has been applied to a generic convolutional layer of a CNN. As a result the layer is decomposed as 1×1 convolution at the input, $D \times D$ convolution in the middle and 1×1 convolution at the output.
- The same tensor decomposition is applied to the fully connected layer to reduce the storage cost.
- A new architecture based on CP decomposition is derived and trained using the ESCA-dataset [41], that contains photographs of healthy and infected leaves, as well as images obtained by data augmentation techniques.
- The optimized architecture is implemented in the Python programmable OpenMV Cam STM32H7 Plus platform, to meet the requirements of a low-power, low-cost, real-time image sensor for the detection of Esca grapevine disease.

IV. CNN Compression by Tensor Decomposition: A Comparative Study

The great accuracy of CNNs is achieved by paying the cost of large memory consumption and high computational complexity, thus in many emerging scenarios such as mobile and embedded applications, CNN compression [42] has become essential. The most common techniques for CNN compression are: *pruning*, *quantization*, *low-rank tensor decomposition*.

Pruning or network pruning has been widely used to compress CNN models in early work [43], [44], [45], [46], [47], and it has proven to be a valid way to reduce the network complexity. This technique starts by learning the weights via normal training, then proceeds by removing the small-weight connections, i.e. connections with weights below a threshold. Finally the network is retrained to learn the final weights of the remaining sparse connections.

Quantization reduces the number of bits used to represent data, and this technique can be applied to several data objects in a neural network, involving weight, activation, gradient and weight update [42]. In [48] a method to train Binarized-Neural-Networks (BNNs), i.e. neural networks with binary weight and activations, has been proposed.

Tensor decomposition is a higher order extension of the singular value decomposition (SVD) for matrices, that reduces a tensor to a low-rank tensor that requires a reduced number of data and operations to be represented by preserving accuracy [49], [50], [51]. In a CNN the convolutional layers are the most time-consuming parts and involve convolutional operations on banks of filters represented by tensors. The size of these tensors can be very large, depending on the dataset to be learned, thus tensor decomposition is a useful technique to speed-up convolutional layers, and it will be applied in the following to derive a high performance CNN architecture. In this context a large variety of algorithms exist, thus to take advantage of this technique comparing the performance achieved with different algorithms is a preliminary task in order to derive the best solution. To this end four of the most beneficial methods known in literature, namely CP-decomposition, Tucker-decomposition (TD) [50], tSVD strategy 1 (tSVD$_1$) and tSVD strategy 2 (tSVD$_2$) [52], will be briefly summarized and a comparative study will be carried out to determine the method with the best performance.

A. CP Decomposition

The CANDECOMP/PARAFAC (CP) decomposition [50] approximates an nth-order tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_n}$ as a linear combination of R rank-1 tensors in the form

$$\hat{\mathcal{X}}_{CP} = \sum_{r=1}^{R} \lambda_r a_r^{(1)} \circ a_r^{(2)} \circ \ldots \circ a_r^{(n)} \tag{7}$$

where the symbol "$\circ$" represents the outer product and $a_r^{(1)}, a_r^{(2)}, \ldots$ are the columns of the matrices $A_1 = [a_1^{(1)}, \ldots, a_R^{(1)}]$, $A_2 = [a_1^{(2)}, \ldots, a_R^{(2)}], \ldots$ respectively.

The matrices $A_1, A_2, \ldots$ are determined by solving for a given value of R the following optimization problem

$$\min_{\hat{\mathcal{X}}_{CP}} \|\mathcal{X} - \hat{\mathcal{X}}_{CP}\|_F \tag{8}$$

where $\|\cdot\|_F$ is the Frobenius norm. It can be shown that the error in (8) can be rewritten as

$$\|\mathcal{X} - \hat{\mathcal{X}}_{CP}\|_F = \|\mathcal{X}_{(1)} - \hat{\mathcal{X}}_{CP(1)}\|_F = \|\mathcal{X}_{(2)} - \hat{\mathcal{X}}_{CP(2)}\|_F = \ldots \tag{9}$$

in which the matrices $\mathcal{X}_{(1)}, \hat{\mathcal{X}}_{CP(1)}, \mathcal{X}_{(2)}, \hat{\mathcal{X}}_{CP(2)}, \ldots$ denote the modal unfolding of tensors $\mathcal{X}$ and $\hat{\mathcal{X}}_{CP}$ respectively. Using (9) the minimization problem (8) reduces to a multilinear least-square problem, which can be solved with the alternating least-square (ALS) technique, i.e. by minimizing iteratively any one of the errors between matrices in (9) until the convergence is reached. The reduction of storage cost achieved with this decomposition for a 3rd order tensor is given by

$$C_{CP} = \frac{(n_1 + n_2 + n_3)R}{n_1 n_2 n_3} \quad . \tag{10}$$

B. Tucker Decomposition

Tucker decomposition [50] approximate an nth-order tensor $\mathcal{X}$ as a multilinear transformation in the form

$$\hat{\mathcal{X}}_{Tuck} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_n=1}^{R_n} g_{r_1,\ldots,r_n} U^{(1)}(:, r_1) \circ \circ U^{(2)}(:, r_2) \circ \ldots U^{(n)}(:, r_n) \tag{11}$$

where $U^{(1)}(:, r_1)$, $U^{(2)}(:, r_2), \ldots$ are columns vectors of the orthogonal matrices $U^{(1)} \in \mathbb{R}^{R_1 \times n_1}$, $U^{(2)} \in \mathbb{R}^{R_2 \times n_2}, \ldots$ respectively, and $g_{r_1,\ldots,r_n}$ is the generic term of the tensor $\mathcal{G}$.

Equivalently (11) can be rewritten in a more compact form as

$$\hat{\mathcal{X}}_{Tuck} = \mathcal{G} \times_1 U^{(1)} \times_2 U^{(2)} \ldots \times_n U^{(n)} \tag{12}$$

where the symbol $\times_n$ denotes the the n-mode product of a tensor with a matrix. The matrices $U^{(1)}, U^{(2)}, \ldots$ are determined by solving the following minimization problem

$$\min_{\hat{\mathcal{X}}_{Tuck}} \|\mathcal{X} - \hat{\mathcal{X}}_{Tuck}\|_F \tag{13}$$

while the tensor $\mathcal{G}$ can be derived from

$$\mathcal{G} = \mathcal{X} \times_1 (U^{(1)})^T \times_2 (U^{(2)})^T \times_3 \ldots \tag{14}$$

using the orthogonality property of matrices $U^{(1)}, U^{(2)}, \ldots$. The reduction of storage cost for a 3rd order tensor in this case is

$$C_{Tuck} = \frac{(n_1 R_1 + n_2 R_2 + n_3 R_3) + R_1 R_2 R_3}{n_1 n_2 n_3} \quad . \tag{15}$$

C. tSVD

tSVD is a generalization of the matrix SVD recently suggested in [53] for third-order tensors. The method is based on the notion of the t-product, a generalization of matrix multiplication for tensors of order three.

Given the tensors $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $\mathcal{B} \in \mathbb{R}^{n_1 \times l \times n_3}$ and denoting by $A_i = A(:, :, i)$, $B_i = B(:, :, i)$ their frontal slices, then the t-product

$$\mathcal{C} = \mathcal{A} * \mathcal{B} \tag{16}$$

gives the $n_1 \times l \times n_3$ tensor $\mathcal{C}$ defined by

$$\begin{bmatrix} C_1 \\ \ldots \\ C_{n_3} \end{bmatrix} = \begin{bmatrix} A_1 & A_{n_3} & \ldots & A_2 \\ A_2 & A_1 & \ldots & A_3 \\ \vdots & \vdots & \vdots & \vdots \\ A_{n_3-1} & A_{n_3-2} & \ldots & A_{n_3} \\ A_{n_3} & A_{n_3-1} & \ldots & A_1 \end{bmatrix} = \begin{bmatrix} B_1 \\ \ldots \\ B_{n_3} \end{bmatrix} \tag{17}$$

where $C_i = C(:,:,i)$ are the slices of the tensor $\mathcal{C}$ and the matrix derived from slices A_i in (13), is a block-circulant matrix of $\mathcal{A}$ denoted as $bcirc(\mathcal{A})$

By taking advantage of block-circulant matrix properties, it can be shown that, denoting with $\bar{\mathcal{A}}$, $\bar{\mathcal{B}}$, $\bar{\mathcal{C}}$ the tensors obtained as the result of the DFT on $\mathcal{A}, \mathcal{B}, \mathcal{C}$ along the 3-rd dimension, we can write

$$\bar{C}_i = \bar{A}_i \, \bar{B}_i, \; i = 1, \ldots, n_3 \quad . \tag{18}$$

On the basis of the t-product definition and (18) a 3rd-order tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ can be factorized as

$$\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T \tag{19}$$

where $\mathcal{U} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, $\mathcal{V} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ are orthogonal, i.e. are such that $\mathcal{U} * \mathcal{U}^T = \mathcal{I}$, $\mathcal{V}^T * \mathcal{V} = \mathcal{I}$ ($\mathcal{I}$ is the identity tensor) and $S \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is an f-diagonal tensor, i.e. a tensor in which each front-back is diagonal (see Fig. 2).

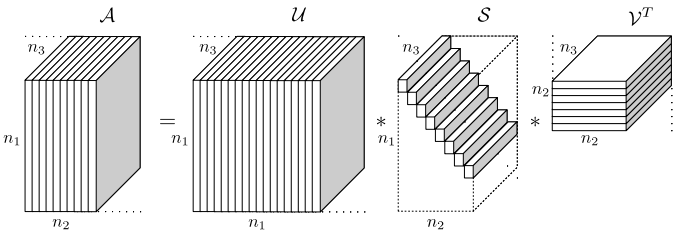


Fig. 2: An illustration of the t-SVD of an $n_1 \times n_2 \times n_3$ tensor.

On the basis of the theoretical results previously summarizes several techniques can be developed for tensor compression, among which tSVD strategy 1 (tSVD$_1$) and tSVD strategy 2 (tSVD$_2$) are the most beneficial.

1) tSVD$_1$: This technique [52] is based on the selection of the k eigenvalues of $\bar{\mathcal{S}}$, the tensor of singular values in the FFT domain, that assume the highest values.

Since $\mathcal{S}$ is f-diagonal it is easy to show that

$$\mathcal{X} = \sum_{i=1}^{\min(n_1,n_2)} \mathcal{U}(:,i,:) * \mathcal{S}(i,i,:) * \mathcal{V}(:,i,:)^T \quad . \tag{20}$$

In order to compress the tensor $\mathcal{X}$ an index $k < \min(n_1, n_2)$ is chosen so that $\mathcal{X}$ is approximately by

$$\mathcal{X}_{\text{tSVD1}} = \sum_{i=1}^{k} \mathcal{U}(:,i,:) * \mathcal{S}(i,i,:) * \mathcal{V}(:,i,:)^T \quad . \tag{21}$$

The reduction of storage cost is given by

$$C_{\text{tSVD1}} = \frac{k\,(n_1 + n_2 + 1)}{n_3 n_1 n_2} \quad . \tag{22}$$

2) tSVD$_2$: This method [52] is based on the following result: given the tSVD decomposition $\mathcal{X} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T$, and defining the sums

$$\begin{aligned} X &= \sum_{k=1}^{n_3} X_k \,, \quad U = \sum_{k=1}^{n_3} U_k \\ S &= \sum_{k=1}^{n_3} S_k \,, \quad V = \sum_{k=1}^{n_3} V_k \,, \end{aligned} \tag{23}$$

where X_k, U_k, S_k, and V_k are the k-th frontal faces of $\mathcal{X}, \mathcal{U}, \mathcal{S}$, and $\mathcal{V}$ respectively, thus U, S, V represent the SVD decomposition of X

$$X = USV^T \quad . \tag{24}$$

Choosing $k_1 \ll n_1$, $k_2 \ll n_2$ and defining

$$\begin{aligned} \widetilde{U} = U(:, 1:k_1), \, \widetilde{V} = V(:, 1:k_2), \, \widetilde{S} = S(1:k_1, 1:k_2) \\ \mathcal{T}(:,:,k) = \widetilde{U}^T \mathcal{X}(:,:,k) \widetilde{V}, \; k = 1, \ldots, n_3 \end{aligned} \tag{25}$$

the compressed tensor is computed as

$$\mathcal{X}_{\text{tSVD2}} = \sum_{i=1}^{k_1} \sum_{j=1}^{k_2} \widetilde{U}(:,i) \circ \widetilde{V}(:,j) \circ \mathcal{T}(i,j,:) \quad . \tag{26}$$

The reduction of storage cost is given by

$$C_{\text{tSVD2}} = \frac{k_1 k_2 n_3 + n_1 k_1 + n_2 k_2}{n_3 n_1 n_2} \tag{27}$$

D. A Comparative Study

In order to select the technique that is able to ensure the best performance both in terms of accuracy and storage gain, a comparative study using the methods previously discussed has been carried out. The datasets used for this purpose are the following:

- USPS. A set of 16×16 images in gray scale representing hand written digits. A tensor $\mathcal{X}$ of dimension $16 \times 16 \times 1000$ has been formed from the dataset.
- MNIST. Dataset of hand written digit in gray scale. A tensor of dimension $28 \times 28 \times 1000$ has been extracted from the dataset.
- AT&T. The set contains face images in gray scale collected from distinct subjects. A tensor of dimension $56 \times 46 \times 100$ has been formed from the dataset.
- COIL-20. The dataset contains images of objects in gray scale. The size of the tensor extracted from this dataset is $32 \times 32 \times 170$.
- CIFAR-10. Dataset of RGB images containing 10 different objects. A tensor of size $32 \times 32 \times 1000$ is obtained from this dataset converting the images in gray scale.

The results achieved using the four tensor decomposition techniques - CP, Tucker, tSVD$_1$, tSVD$_2$ - on the five datasets just mentioned, are reported in Figs. 3, 4 and 5. The reconstruction error adopted in these experiments is the following measure

$$MMSE = \frac{\|\mathcal{X} - \hat{\mathcal{X}}\|_F^2}{\|\mathcal{X}\|_F^2} \quad . \tag{28}$$

The comparison reported in the Figures clearly shows that CP method for tensor decomposition outperforms the other techniques both in terms of accuracy and compression factor. As additional considerations, tSVD$_1$ and tSVD$_2$ have performance strictly depending on the dataset and on the depth of tensor, i.e. the n_3 dimension. For tensors of reduced dimension, in particular for USPS and COIL-20 datasets, tSVD$_1$ shows the best performance.

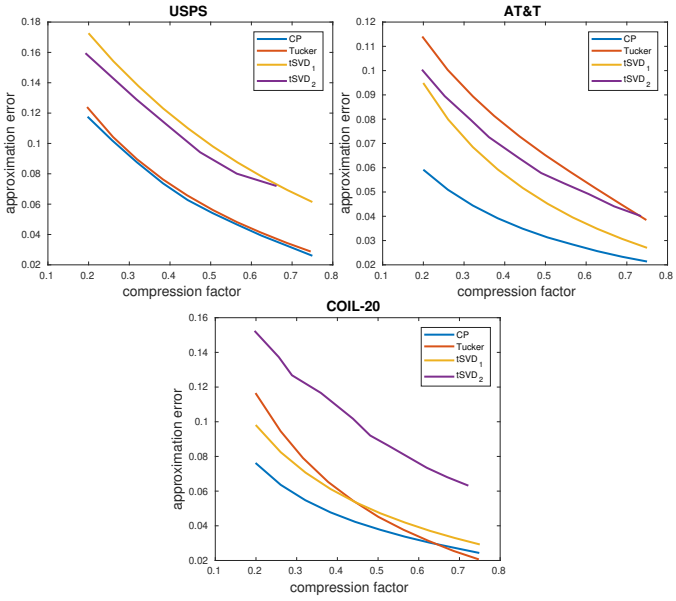


Fig. 3: Experimentation on USPS, AT&T and COIL-20 datasets.

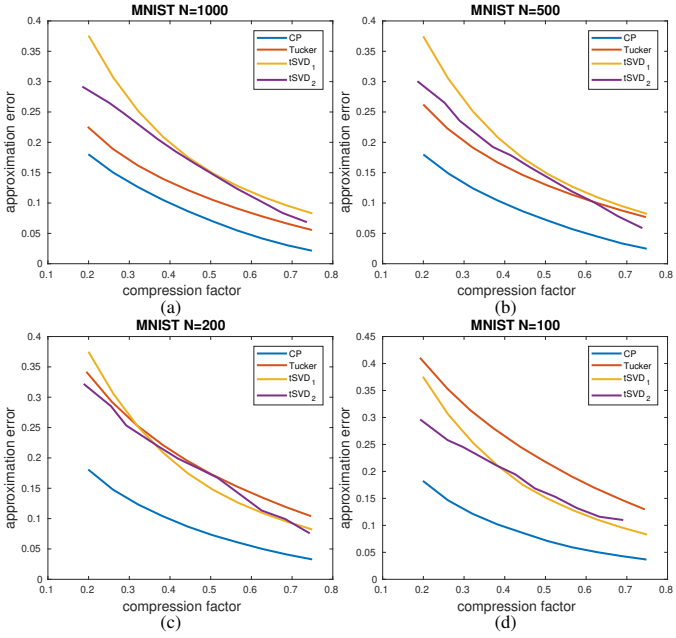


Fig. 4: Experimentation on MNIST dataset by selecting a different number of images, (a) $N = 1000$, (b) $N = 500$, (c) $N = 200$, (d) $N = 100$.

E. CNN Compression

To address the CNN compression problem it is worth to notice that in such a network the convolutional layers are the most time-consuming part, while the fully-connected layers involve most storage cost, thus these two layers will be treated separately.

1) Convolutional Layer: Having proven on the basis of the comparative study previously discussed that CP decomposition ensures the best performance, in the following we want to reformulate the convolution operation by decomposition the kernel with this technique.

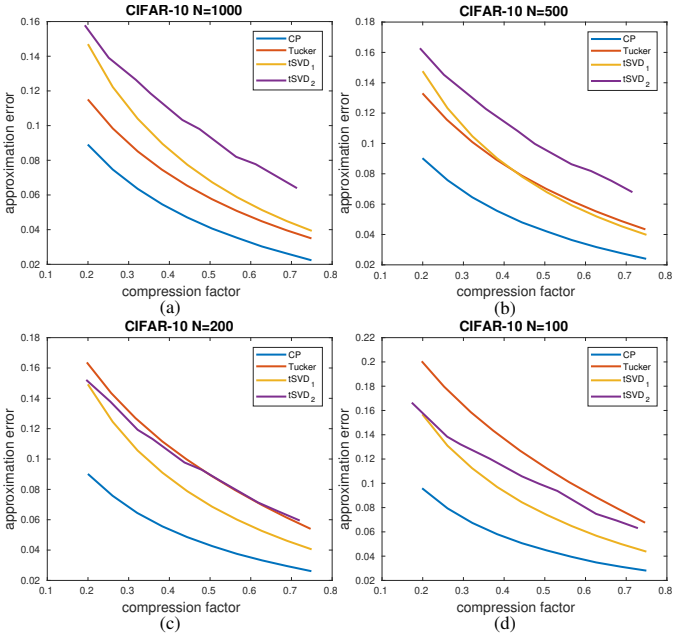


Fig. 5: Experimentation on CIFAR-10 dataset by selecting a different number of images, (a) $N = 1000$, (b) $N = 500$, (c) $N = 200$, (d) $N = 100$.

To this end let us refer to a generic convolution layer of a CNN

$$\mathcal{Y}_{h',w',t} = \sum_{s=1}^{S} \sum_{i,j} \mathcal{X}_{h_i,w_j,s}\, \mathcal{K}_{i,j,s,t} \tag{29}$$

where $\mathcal{X} \in \mathbb{R}^{I_x \times I_y \times S}$ is the input tensor and $\mathcal{K}_{i,j,s,t}$ is the kernel dimension $D \times D \times S \times T$. Using the CP decomposition the kernel can be approximated by

$$\hat{k} = \sum_{i=1}^{R} a_r \circ b_r \circ c_r \circ d_r \quad . \tag{30}$$

Combining (29) and (30) yields

$$\begin{aligned} \mathcal{Y}_{h',w',t} &= \sum_{r=1}^{R} \sum_{s=1}^{S} \sum_{i,j} \mathcal{X}_{h_i,w_j,s}\, (a_{i,r} b_{j,r} c_{s,r} d_{t,r}) = \\ &= \sum_{r=1}^{R} \sum_{s=1}^{S} \sum_{i,j} \mathcal{X}_{h_i,w_j,s}\, \mathcal{Q}_{i,j,r}\, c_{s,r}\, d_{t,r} \end{aligned} \tag{31}$$

where $\mathcal{Q}_{i,j,r} = a_{i,r} b_{j,r}$ and $Q \in \mathbb{R}^{D \times D \times R}$.

Rearranging the terms (31) can be interpreted as a sequence of two convolutions

$$\mathcal{Y}_{h',w',t} = \sum_{r=1}^{R} \sum_{i,j} \mathcal{W}_{h_i,w_j,r} \mathcal{Q}_{i,j,r} d_{t,r} = \sum_{r=1}^{R} \mathcal{Z}_{h',w',r} d_{t,r} \tag{32}$$

where the tensors $\mathcal{W}$ and $\mathcal{Z}$ are given by

$$\begin{aligned} \mathcal{W}_{h_i,w_j,r} &= \sum_{s=1}^{S} \mathcal{X}_{h_i,w_j,s} c_{s,r} \\ \mathcal{Z}_{h',w',r} &= \sum_{i,j} \mathcal{W}_{h_i,w_j,r} \mathcal{Q}_{i,j,r}\,. \end{aligned} \tag{33}$$

As a result the convolution layer is decomposed as two 1×1 convolutions at the input and output ends, and a depthwise convolution in the middle. Fig. 6 shows the effect of CP decomposition on the convolution layer.

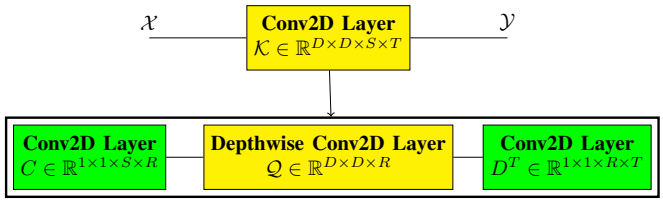


Fig. 6: The effect of CP decomposition on Conv2D Layer.

The architecture of convolutional layer as decomposed by CP technique is depicted in Fig. 7.

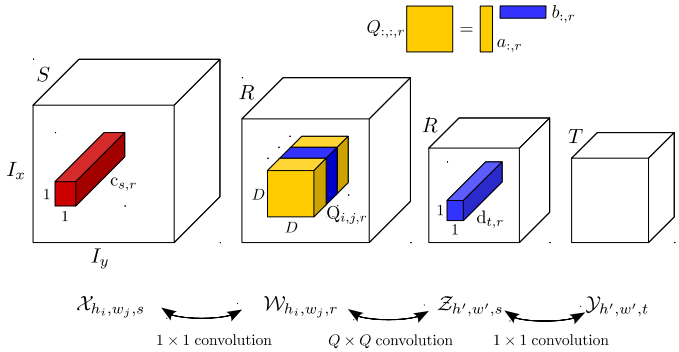


Fig. 7: The architecture of a convolutional layer decomposed by CP technique.

It can be proven that using this decomposition the compression factor

$$C_{\text{CP}}^{\text{conv}} = \frac{R\left(D^2+S+T\right)}{D^2ST} \tag{34}$$

is obtained.

2) Fully Connected Layer: As previously stated the fully-connected layers involve most storage cost. This layer is represented by a linear transformation

$$y = Wx + b,\ x \in \mathbb{R}^n, W \in \mathbb{R}^{m\times n}\ b \in \mathbb{R}^m \quad . \tag{35}$$

Applying CP decomposition to (35) is equivalent to low-rank factorization of matrix W

$$\hat{W} = BA^T,\ B \in \mathbb{R}^{m\times R},\ A \in \mathbb{R}^{n\times R} \tag{36}$$

thus in this way (35) becomes

$$\begin{aligned} \hat{y} &= Bz + b \\ z &= A^T x \end{aligned} \tag{37}$$

Fig. 8 shows the effect of CP decomposition on fully-connected layer.

The compression factor so obtained is

$$C_{\text{CP}}^{\text{dense}} = \frac{R\left(m+n\right)}{mn} \tag{38}$$

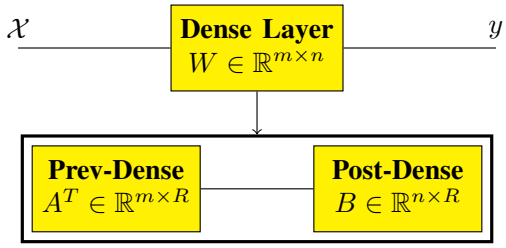


Fig. 8: The effect of CP decomposition on Dense Layer.

V. Dataset

The ESCA-dataset [41] 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 1920×1080 and 1280×720 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.

Table I reports the description of the dataset for each of the two classes and the size of the dataset used in this work before and after augmentation.

TABLE I: Esca dataset consistency.

Class name	Class ID	Number of original images	Number of images after all data augmentation transformations	Number of images after considered data augmentation transformations
esca	1	888	12432	8880
healty	2	882	12348	8820
Total		1770	24780	17700

Fig. 9 shows two examples of leaf images for the class "esca" and "healthy" respectively.

a) b)

Fig. 9: Example of grapevine leaves belonging to different classes: a) Esca disease, b) healthy.

Fig. 10 and Fig. 11 depict several examples of images extracted from the dataset, showing the considered data augmentation transformations.

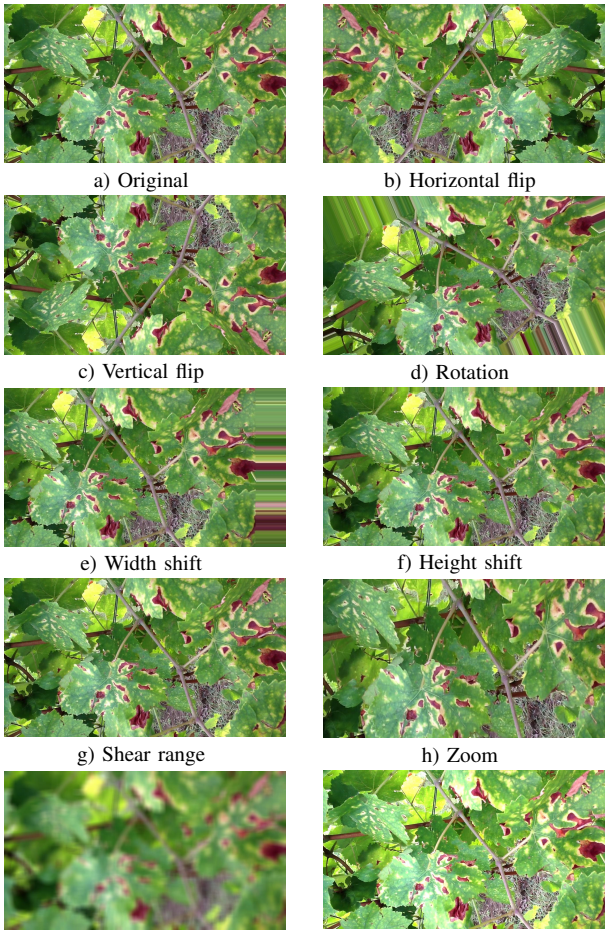


Fig. 10: Examples of augmentation for the class "esca" where a) is the original image and b), c), d), e), f), g), h), i), l) are the augmented images.

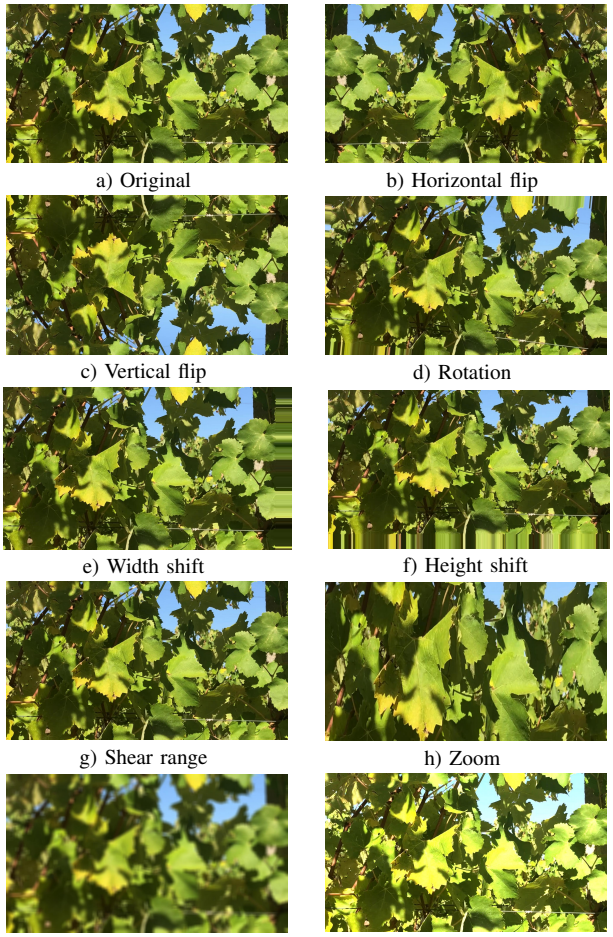


Fig. 11: Examples of augmentation for the class "healthy" where a) is the original image and b), c), d), e), f), g), h), i), l) are the augmented images.

VI. Design of TensorFlow CNN Architecture

A new CNN architecture based on CP tensor decomposition was developed in TensorFlow/Keras v. 2.4.0 environment. The design-flow, depicted in Fig. 12, can be divided in three main steps.

- First a lightweight architecture was designed to meet the accuracy requirement. In this network all the tensors representing both feature layers and kernel filters are full-rank tensors, i.e. not decomposed by tensor decomposition technique, so this network will be called full-rank Net (FR-Net).
- Second, in order to reduce storage cost and computational complexity, CP decomposition technique was applied to FR-Net as explained in Section IV obtaining a new architecture which will be called low-rank Net (LR-Net).
- Finally in order to compensate for the loss of accuracy caused by weights' approximation, LR-Net was iteratively fine-tuned on the same training dataset used for FR-Net.

The trained model and the source code to test the LR-Net are available at https://codeocean.com/capsule/8031070/tree/v1.

A. FR-Net

The FR-Net architecture is shown in Fig. 12 and comprises 5 weight layers in total. Specifically, 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 architecture description is provided in Table II.

The network was trained for 30 epochs on the ESCA-dataset described in Section V, by using an Adadelta optimizer with categorical cross entropy, a learning rate of 0.5 and a batch size of 64, obtaining the performance reported in Table VI.

B. LR-Net

CP decomposition was applied to all but the first convolutional layer and the final dense layer used for classification. These layers have a relatively low number of parameters and do not add a significant overload to storage and computational cost. Moreover, compression of these layers would result in a significant accuracy loss difficult to be recovered. For the same reason the compression factors achieved on the other two convolutional layers is much higher than that of dense layer.

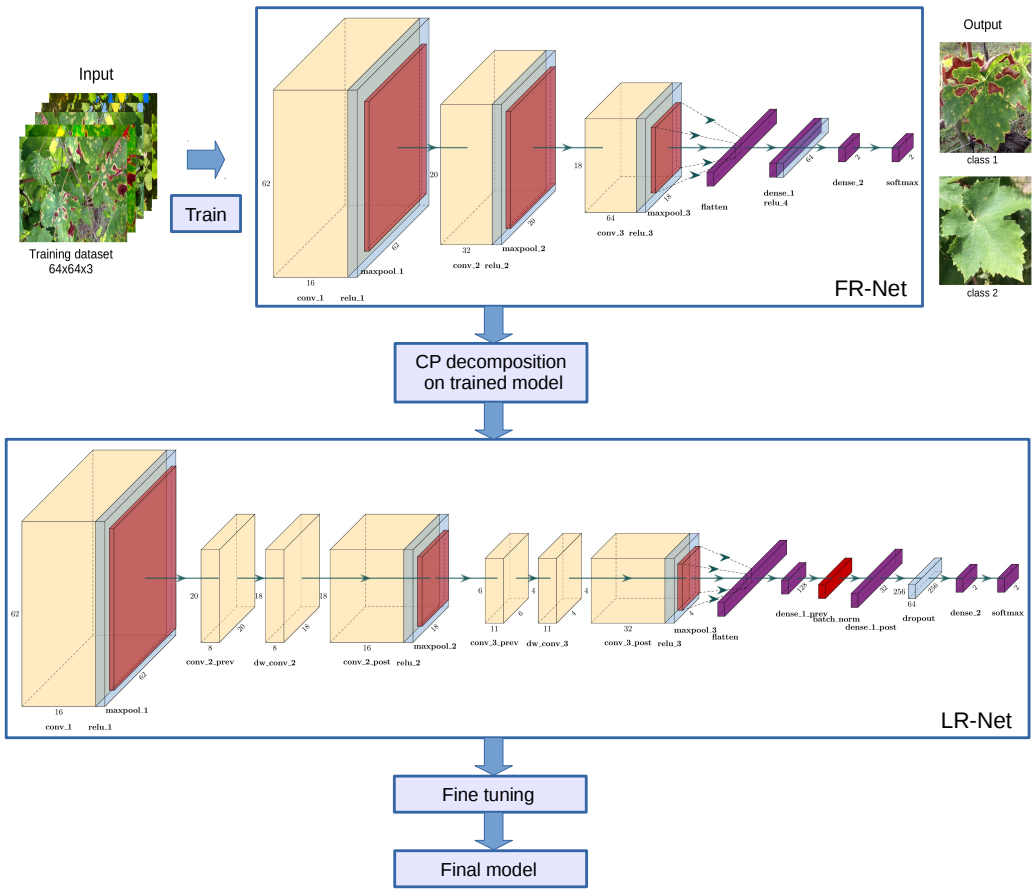


Fig. 12: Design TensorFlow of CNN architecture.

TABLE II: FR-Net architecture.

Type	Filter shape	Input size	Number of parameters
conv_1	$3 \times 3 \times 3 \times 16$	$64 \times 64 \times 3$	448
relu_1	–	$62 \times 62 \times 16$	0
maxpool_1 (3×3)	–	$62 \times 62 \times 16$	0
conv_2	$3 \times 3 \times 16 \times 32$	$20 \times 20 \times 16$	4640
relu_2	–	$18 \times 18 \times 32$	0
maxpool_2 (3×3)	–	$18 \times 18 \times 32$	0
conv_3	$3 \times 3 \times 32 \times 64$	$6 \times 6 \times 32$	18496
relu_3	–	$4 \times 4 \times 64$	0
maxpool_3 (2×2)	–	$4 \times 4 \times 64$	0
flatten	–	$2 \times 2 \times 64$	0
dense_1	256×64	1×256	16448
relu_4	–	1×64	0
dropout (0.5)	–	1×64	0
dense_2	64×2	1×64	130
softmax	–	1×2	0

TABLE III: LR-Net architecture.

Type	Filter shape	Input size	Number of parameters
conv_1	$3 \times 3 \times 3 \times 16$	$64 \times 64 \times 3$	448
relu_1	–	$62 \times 62 \times 16$	0
maxpool_1 (3×3)	–	$62 \times 62 \times 16$	0
conv_2_prev	$1 \times 1 \times 16 \times 11$	$20 \times 20 \times 16$	176
dw_conv_2	$3 \times 3 \times 11$	$20 \times 20 \times 11$	99
conv_2_post	$1 \times 1 \times 11 \times 32$	$18 \times 18 \times 11$	384
relu_2	–	$18 \times 18 \times 32$	0
maxpool_2 (3×3)	–	$18 \times 18 \times 32$	0
conv_3_prev	$1 \times 1 \times 32 \times 23$	$6 \times 6 \times 32$	736
dw_conv_3	$3 \times 3 \times 23$	$6 \times 6 \times 23$	207
conv_3_post	$1 \times 1 \times 23 \times 64$	$4 \times 4 \times 23$	1536
relu_3	–	$4 \times 4 \times 64$	0
maxpool_3 (2×2)	–	$4 \times 4 \times 64$	0
flatten	–	$2 \times 2 \times 64$	0
dense_1_prev	256×26	1×256	6656
batch_norm	–	1×26	104
dense_1_post	26×64	1×26	1728
relu_4	–	1×64	0
dropout (0.5)	–	1×64	0
dense_2	64×2	1×64	130
softmax	–	1×2	0

The architecture graph and the description of the low-rank Network (LR-Net) obtained with the decomposition can be found in Fig 12 and Table III respectively.

Details about the performance of CP decomposition are provided in Table IV. CNN decomposition was implemented

by using *Tensorly* [54], a specific library for tensor description in Python.

TABLE IV: Performance of FR-Net CP decomposition.

Layer	Compression factor	Decomposition rank	Weights approximation error
conv_2	8	11	0.729
conv_3	8	23	0.751
dense_1	2	26	0.550

C. Fine-tuning

Fine-tuning is a technique needed to overcome the accuracy loss due to decomposition and it was performed with an iterative procedure.

Instead of decomposing all layers and retraining the whole network, a single fine-tuning was applied after each layer decomposition with decreasing learning rates as shown in Table V. This procedure limits the possibility of finding local minima which could compromise network performance.

TABLE V: LR-Net finetuning.

Layer	Optimizer	Learning rate	Epochs
conv_2	Adadelta	0.01	20
conv_3	Adadelta	0.002	15
dense_1	Adadelta	0.01	30

Table VI reports a comparison between the FR-Net and the LR-Net performance.

TABLE VI: Comparison between the baseline proposed architecture (FR-Net) and the same architecture with CP decomposition (LR-Net).

Model	Memory cost [KB]	Compression factor	Parameters number	Test accuracy	Test loss
FR-Net	48.047	4	40 162	**0.990**	**0.025**
LR-Net	**25.094**	**13**	**12 204**	0.984	0.041

VII. CNN-Based Image Sensor

A. Hardware

The image detector has been implemented in the OpenMV Cam STM32H7 Plus platform, a low-power Python programmable machine vision camera that supports an extensive set of image processing functions and neural networks. The OpenMV Cam STM32H7 Plus camera 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 2592×1944 but most algorithms run between 10-15-25-50 FPS on QVGA (320×240) resolutions and below.

B. Software

The OpenMV Cam is particularly suitable for the implementation of machine learning applications. It can be directly programmed in high level Python scripts, thanks to the MicroPython Operating System.

The main features are:

- STM32Cube.AI to automatically generate low-level C code from the pre trained neural networks. This allows users to develop applications in TensorFlow, Keras, Caffe rather than using C/C++ language. The process for using STM32Cube.AI with OpenMV is described in Fig. 13.
- OpenMV firmware source code available. With this feature the programmer can eventually modify the firmware.

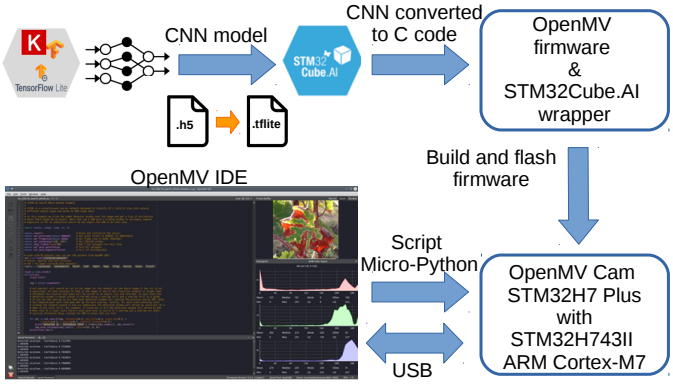


Fig. 13: Process of integrating the CNN into the OpenMV environment with STM32Cube.AI.

Once the model has been trained to a satisfactory accuracy, it must be converted to an executable code that runs on the embedded device. This can be a complex process, but TensorFlow offers the TensorFlow Lite converter Python API for this purpose, that converts the model into a FlatBuffer, reducing the model size, and modifies it to use TensorFlow Lite operations.

To obtain the smallest possible model size, quantization is a recommended approach. Quantization is a tricky and involved process, and it is still an active area of research, so taking the float graph of trained net and converting it down to 8 bit takes quite a bit of code. Quantization can be applied during the training stage (quantization aware training) or directly on the model file (post training quantization). We rely on the TFLiteConverter class to handle the quantization and conversion into the TensorFlow Lite FlatBuffer file (tflite) that we need for the inference engine. The last step consists of converting the tflite model into a C source file with the help of the STM32Cube.AI tool converter. This executable file can be directly stored in OpenMV Cam Flash using the OpenMV IDE. Fig. 14 resumes the described procedure.

Alternatively, the generated tflite model can be directly loaded and run by the OpenMV STM32H7 Plus Cam, with the constraint that the model and the model's required scratch

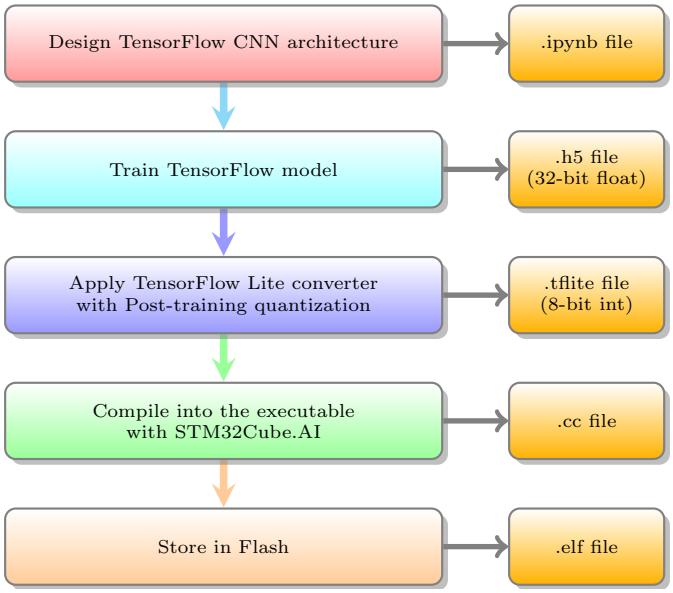


Fig. 14: OpenMV Cam code flow.

RAM must fit within the available 31 MB frame buffer stack RAM.

VIII. Experimental Results

In order to validate the image detector for real-time grapevine plant disease detection previously discussed, two different experiments, *A)* the experiment on desktop and *B)* the experiment on the OpenMV Cam STM32H7 Plus, were conducted. The first aims to compare the performance of LR-Net with those achieved with the state-of-the-art networks. The second aims to determine the final performance, obtained by the network implemented in the embedded processor.

The experiments were conducted using the TensorFlow Keras v. 2.4.0 to train the models on Google Colaboratory (GPU runtime) with the ESCA-dataset partitioned as reported in Table VII, the STM32Cube.AI v. 5.2.0 to analyse the generated models, and the OpenMV firmware v. 3.9.3 on board.

TABLE VII: 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

A. Experiment on Desktop

In this experiment a wide variety of CNN architectures have been trained on ESCA-dataset in order to compare their performance with those achieved with LR-Net. In particular the following networks have been used: MobileNet V1 [29], MobileNet V2 [30] , MobileNet V3 [31], ResNet [38], LeNet [32], SqueezeNet [33], ShuffleNet V1 [34], ShuffleNet V1 0.25x [34], Improved ShuffleNet V1 [28], ShuffleNet V2 [36], ShuffleNet V2 0.25x [36], Improved ShuffleNet V2 [28], PeleeNet [39].

Table VIII reports the results achieved for storage cost, compression, complexity and accuracy. As can be seen the LR-Net network proposed in this paper outperforms all the other networks both in terms of memory cost and parameter numbers (complexity). The gains achieved with LR-Net for these performance are relevant, i.e. in some cases of several order of magnitude. All the networks but LR-Net are compressed by a factor of 4 due to the post training quantization that is performed by the TensorFlow Lite converter Python API. An extra compression factor is guaranteed using CP decomposition technique in LR-Net, thus obtaining a total compression factor of 13. As far as the accuracy is concerned, the value achieved with LR-Net is just a few percentage below the best performance obtained with other networks.

As we will discuss later this does not constitute a real issue for the final plant disease detection, since this lack of accuracy is compensated by the greater number of frame per second the LR-Net is able to perform.

B. Experiment on the OpenMV Cam STM32H7 Plus

In this case a subset of the networks used in the first experiment is considered, since some of them are not supported by the OpenMV Cam STM32H7 Plus platform.

Table IX reports the results achieved for accuracy, inference time, number of recognized frame per second (FPS), the number of images (N_{inf}) processed by the sensor during the time interval $L/v = 1.36$ sec, i.e. the time required by the tractor to cover the distance L, and the TP metric.

These results clearly show the superiority of LR-Net with respect to all other networks both in terms of accuracy, inference time, FPS, N_{inf}, TP metric and memory occupancy. In particular LR-Net is the only network that is able to meet the severe constraint of $t_{inf} = 68$ msec derived as design specification in Section II.

IX. Conclusion

To design and develop a low-cost, low-power and real-time image detector based on a CNN architecture for grape leaf Esca disease, some several constraints have to be met. In particular accuracy, inference time as well as memory occupancy are of primary concern. However, even though a wide variety of CNN architectures have recently been proposed for plant disease detection, the performance achieved with those networks, in terms of storage cost and computational complexity, do not suffice for such an embedded application. This paper shows that adopting a CNN compression technique based on tensor-decomposition, a low-rank architecture (LR-Net) with a very low complexity can be derived by preserving the accuracy.

TABLE VIII: CNNs comparison on Esca dataset - Performance on desktop.

Model	Memory cost [KB]	Compression factor	Parameters number	Inference time [msec]	Test accuracy	Test loss
LR-Net	**25.094**	**13**	**12 204**	**12.044**	0.984	0.041
MobileNet V1 [29]	3525.992	4	3 237 058	132.881	0.968	0.328
MobileNet V2 [30]	2793.797	4	2 268 226	75.932	0.964	0.385
MobileNet V3 [31]	1905.031	4	1 538 162	25.988	0.920	0.603
ResNet [38]	23 948.109	4	23 581 186	924.519	0.955	0.572
LeNet [32]	335.773	4	337 806	15.795	0.943	0.368
SqueezeNet [33]	843.656	4	736 450	152.698	0.984	0.052
ShuffleNet V1 [34]	1811.867	4	1 359 914	72.221	0.992	0.024
ShuffleNet V1 0.25x [34]	328.297	4	105 404	14.841	0.980	0.067
Improved ShuffleNet V1 [28]	2354.695	4	5 720 606	70.311	**0.995**	**0.016**
ShuffleNet V2 [36]	1810.172	4	1 329 722	58.591	0.984	0.060
ShuffleNet V2 0.25x [36]	328.562	4	105 404	14.553	0.962	0.228
Improved ShuffleNet V2 [28]	6369.516	4	5 720 606	147.065	0.992	0.033
PeleeNet [39]	2325.406	4	2 113 250	130.662	0.884	0.253

TABLE IX: CNNs comparison on Esca dataset - Performance on the OpenMV Cam STM32H7 Plus.

Model	Test accuracy	Inference time [msec/img]	FPS	N_{inf} [FPS$\times(L/v)$]	TP [accuracy$\times N_{inf}$]	ROM bytes
LR-Net	**0.980**	**64.213**	**15.574**	**21.180**	**20.756**	**13 015**
MobileNet V1 [29]	0.974	577.618	1.731	2.354	2.292	3 237 064
MobileNet V2 [30]	0.954	526.646	1.899	2.582	2.463	2 268 232
ResNet [38]	0.944	3687.71	0.271	0.368	0.347	23 528 194
LeNet [32]	0.942	85.726	11.665	15.864	14.944	338 493
SqueezeNet [33]	0.976	448.866	2.228	3.030	2.957	745 384
PeleeNet [39]	0.938	411.456	2.240	3.046	2.857	2 108 392

REFERENCES

[1] R. Bramley and R. Hamilton, "Understanding variability in winegrape production systems 1. within vineyard variation in yield over several vintages," *Australian Journal of Grape and Wine Research*, vol. 10, no. 1, pp. 32–45, 2004.

[2] C. Acevedo-Opazo, B. Tisseyre, S. Guillaume, and H. Ojeda, "The potential of high spatial resolution information to define within vineyard zones related to vine water status," *Precision Agriculture*, vol. 9, no. 5, pp. 285–302, 2008.

[3] L. G. Santesteban, S. Guillaume, J. B. Royo, and B. Tisseyre, "Are precision agriculture tools and methods relevant at the whole-vineyard scale?" *Precision Agriculture*, vol. 14, no. 1, pp. 2–17, 2013.

[4] S. F. Di Gennaro, R. Dainelli, A. Palliotti, P. Toscano, and A. Matese, "Sentinel-2 validation for spatial variability assessment in overhead trellis system viticulture versus UAV and agronomic data," *Remote Sensing*, vol. 11, no. 21, 2019.

[5] L. J. Klein, H. F. Hamann, N. Hinds, S. Guha, L. Sanchez, B. Sams, and N. Dokoozlian, "Closed loop controlled precision irrigation sensor network," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4580–4588, 2018.

[6] M. Gatti, C. Squeri, A. Garavani, T. Frioni, P. Dosso, I. Diti, and S. Poni, "Effects of variable rate nitrogen application on cv. Barbera performance: Yield and grape composition," *American Journal of Enology and Viticulture*, vol. 70, no. 2, pp. 188–200, 2019.

[7] J. M. Meyers, N. Dokoozlian, C. Ryan, C. Bioni, and J. E. Vanden Heuvel, "A new, satellite NDVI-based sampling protocol for grape maturation monitoring," *Remote Sensing*, vol. 12, no. 7, 2020.

[8] N. Bendel, A. Kicherer, A. Backhaus, H.-C. Klück, U. Seiffert, M. Fischer, R. T. Voegele, and R. Töpfer, "Evaluating the suitability of hyper- and multispectral imaging to detect foliar symptoms of the grapevine trunk disease Esca in vineyards," *Plant methods*, vol. 16, no. 1, pp. 1–18, 2020.

[9] L. Mugnai, A. Graniti, and G. Surico, "Esca (black measles) and brown wood-streaking: Two old and elusive diseases of grapevines," *Plant Disease*, vol. 83, no. 5, pp. 404–418, 1999.

[10] V. Mondello, A. Songy, E. Battiston, C. Pinto, C. Coppin, P. Trotel-Aziz, C. Clément, L. Mugnai, and F. Fontaine, "Grapevine trunk diseases: A review of fifteen years of trials for their control with chemicals and biocontrol agents," *Plant Disease*, vol. 102, no. 7, pp. 1189–1217, 2018.

[11] D. Gramaje, L. Mostert, J. Z. Groenewald, and P. W. Crous, "Phaeoacremonium: From esca disease to phaeohyphomycosis," *Fungal Biology*, vol. 119, no. 9, pp. 759–783, 2015.

[12] G. Romanazzi, S. Murolo, L. Pizzichini, and S. Nardi, "Esca in young and mature vineyards, and molecular diagnosis of the associated fungi," *European Journal of Plant Pathology*, vol. 125, no. 2, pp. 277–290, 2009.

[13] S. Murolo and G. Romanazzi, "Effects of grapevine cultivar, rootstock and clone on esca disease," *Australasian Plant Pathology*, vol. 43, no. 2, pp. 215–221, 2014.

[14] M. Borgo, G. Pegoraro, and E. Sartori, "Susceptibility of grape varieties

to esca disease," *BIO Web of Conferences, 39th World Congress of Vine and Wine*, vol. 7, p. 01041, 01 2016.

[15] F. Fontaine, D. Gramaje, J. Armengol, R. Smart, Z. A. Nagy, M. Borgo, C. Rego, and M.-F. Corio-Costet, *Grapevine trunk diseases. A review*. OIV publications, 05 2016, OIV - International Organisation of Vine and Wine.

[16] K. Baumgartner. Guide to managing vineyard trunk disease in lodi. [Online]. Available: https://www.lodigrowers.com/guide-to-managing-vineyard-trunk-disease-in-lodi/

[17] M. L. Cooper, L. J. Bettiga, R. J. Smith, R. Travadon, and K. Baumgartner. Guide to vineyard trunk diseases in california. [Online]. Available: http://ipm.ucanr.edu/PDF/PMG/grape_trunk_disease_view.pdf

[18] V. Sze, Y. Chen, T. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[19] S. H. Lee, C. S. Chan, P. Wilkin, and P. Remagnino, "Deep-plant: Plant identification with convolutional neural networks," in *2015 IEEE International Conference on Image Processing (ICIP)*, 2015, pp. 452–456.

[20] G. L. Grinblat, L. C. Uzal, M. G. Larese, and P. M. Granitto, "Deep learning for plant identification using vein morphological patterns," *Computers and Electronics in Agriculture*, vol. 127, pp. 418–424, 2016.

[21] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in Plant Science*, vol. 7, p. 1419, 2016.

[22] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, "Deep neural networks based recognition of plant diseases by leaf image classification," *Computational intelligence and neuroscience*, vol. 2016, 2016.

[23] P. Pawara, E. Okafor, O. Surinta, L. Schomaker, and M. Wiering, "Comparing local descriptors and bags of visual words to deep convolutional neural networks for plant recognition," in *International Conference on Pattern Recognition Applications and Methods*, vol. 2. SCITEPRESS, 2017, pp. 479–486.

[24] A. Fuentes, S. Yoon, S. C. Kim, and D. S. Park, "A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition," *Sensors*, vol. 17, no. 9, 2017.

[25] M. Agarwal, S. K. Gupta, and K. K. Biswas, "Grape disease identification using convolution neural network," in *2019 23rd International Computer Science and Engineering Conference (ICSEC)*, 2019, pp. 224–229.

[26] J. Boulent, S. Foucher, J. Théau, and P.-L. St-Charles, "Convolutional neural networks for the automatic identification of plant diseases," *Frontiers in Plant Science*, vol. 10, p. 941, 2019.

[27] X. Xie, Y. Ma, B. Liu, J. He, S. Li, and H. Wang, "A deep-learning-based real-time detector for grape leaf diseases using improved convolutional neural networks," *Frontiers in Plant Science*, vol. 11, p. 751, 2020.

[28] Z. Tang, J. Yang, Z. Li, and F. Qi, "Grape disease image classification based on lightweight convolution neural networks and channelwise attention," *Computers and Electronics in Agriculture*, vol. 178, p. 105735, 2020.

[29] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[30] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *CoRR*, vol. abs/1801.04381, 2018.

[31] A. Howard, M. Sandler, G. Chu, L. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for MobileNetV3," *CoRR*, vol. abs/1905.02244, 2019.

[32] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[33] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "Squeezenet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size," *CoRR*, vol. abs/1602.07360, 2016.

[34] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6848–6856.

[35] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, 01 2012.

[36] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet V2: Practical guidelines for efficient CNN architecture design," in *Computer Vision – ECCV 2018*. Cham: Springer International Publishing, 2018, pp. 122–138.

[37] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7132–7141.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[39] R. J. Wang, X. Li, S. Ao, and C. X. Ling, "Pelee: A real-time object detection system on mobile devices," *CoRR*, vol. abs/1804.06882, 2018.

[40] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269.

[41] M. Alessandrini, R. Rivera, L. Falaschetti, D. Pau, V. Tomaselli, and C. Turchetti, "A grapevine leaves dataset for early detection and classification of Esca disease in vineyards through machine learning," *Data in Brief*, p. 106809, 2021.

[42] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485–532, 2020.

[43] Y. L. Cun, J. S. Denker, and S. A. Solla, *Optimal Brain Damage*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, pp. 598–605.

[44] S. J. Hanson and L. Y. Pratt, "Comparing biases for minimal network construction with back-propagation," in *Proceedings of the 1st International Conference on Neural Information Processing Systems*, ser. NIPS'88. Cambridge, MA, USA: MIT Press, 1988, p. 177–185.

[45] N. Ström, "Phoneme probability estimation with dynamic sparsely connected artificial neural networks," *The Free Speech Journal*, vol. 5, no. 1-41, p. 2, 1997.

[46] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 243–254.

[47] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," 2016.

[48] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1," 2016.

[49] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, September 2009.

[50] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Processing Magazine*, vol. 32, no. 2, p. 145–163, Mar 2015.

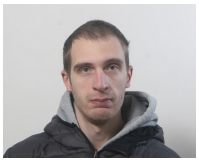
[51] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.

[52] M. E. Kilmer, C. D. Martin, and L. Perrone, "A third-order generalization of the matrix SVD as a product of third-order tensors," *Tufts University, Department of Computer Science, Tech. Rep. TR-2008-4*, 2008.

[53] M. E. Kilmer and C. D. Martin, "Factorization strategies for third-order tensors," *Linear Algebra and its Applications*, vol. 435, no. 3, pp. 641–658, 2011, special Issue: Dedication to Pete Stewart on the occasion of his 70th birthday.

[54] J. Kossaifi, Y. Panagakis, A. Anandkumar, and M. Pantic, "Tensorly: Tensor learning in python," *Journal of Machine Learning Research*, vol. 20, no. 26, pp. 1–6, 2019.

Laura Falaschetti (S'15–M'16) received the B.Sc., the M.Sc. and the Ph.D. degree in electronics engineering from the Università Politecnica delle Marche, Ancona, Italy, in 2008, 2012 and 2016 respectively. She collaborated as research fellow with the Department of Information Engineering (DII) at the Università Politecnica delle Marche, from 2012 to 2013. She is currently a post-doctoral research fellow at the DII and she is a contract professor for the course Electronic Systems, at Electronic and Biomedical Engineering, Università Politecnica delle Marche. Her current research interests include: embedded systems, machine learning, neural networks, manifold learning, pattern recognition, signal processing, image processing, speech recognition, speaker identification, speech synthesis, bio-signal analysis.

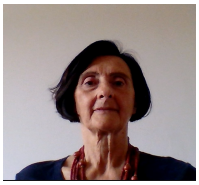


Lorenzo Manoni received the B.Sc. And M.Sc. degrees in electronics engineering from the Università Politecnica delle Marche, Ancona, Italy, in 2015 and 2018, respectively, where he is currently pursuing the Ph.D. degree with Department of Information Engineering (DII). His current research interests include signal processing, embedded systems, machine learning, algorithms analysis and design, bio-signal analysis.

Romel Calero Fuentes Rivera received his B.Sc and M.Sc. degrees in electronics engineering from the Università Politecnica delle Marche, Ancona, Italy, in 2016 and 2021, respectively, working on the development of convolutional neural networks for image classification. He is currently working as a firmware developer, dealing with signal processing and programming of embedded systems.

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.

Gianfranco Romanazzi got Degree 'cum laude' in Agricultural Sciences in 1995 and PhD in Crop Protection in 1999 at the University of Bari. He joined Marche Polytechnic University in Ancona in 2001, where he is professor of Plant pathology and chairs the BSc in Agricultural Science and Technology and MSc in Land and Agricultural Technology programs. He coordinates several projects, including PRIMA StopMedWaste and Euphresco BasicS, and research activity focuses on alternatives to synthetic fungicides for pre and postharvest disease management. He investigated the distribution of esca disease in vineyards of the area, according to the cultivar and rootstock, identifying associated agents with classical and molecular tools. Since June 2020, he is President of the Italian Association for Plant Protection (AIPP).



Oriana Silvestroni graduated 'cum laude' in Agricultural Sciences in 1979 at the University of Bologna. She collaborated with the Arboriculture Institute of the University of Bologna from 1979 to 1992, first as a fellow, then as a university researcher. She joined the Università Politecnica delle Marche in 1992 as an associated professor and became full professor since 2000 at the same university, where she was the Head of the Department of Envinronmental and Crop Science for 6 years. She has been teaching viticulture and ampelography since 1992 and coordinated the PhD course in Plant Production and Environment for 7 years. Her current research interests include: grapevine physiology in relation to environmental stress and to vineyard management; innovation in vineyard management for mitigation and adaptation to climate change; berry ripening and grape quality; study and management of viticultural biodiversity. Member of the National Academy of Agriculture (ANA) and of the Italian Academy of Vine and Wine (AIVV), she has been on the Board of Directors of the latter since 2013. She is part of the Italian Horticultural Society, where she founded the Working Group "Viticulture" in 2005, which she coordinated for 16 years.

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 (M'86) received the Laurea degree in electronics engineering from the University of Ancona, Ancona, Italy, in 1979. He joined the Università 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'Università e Ricerca.