Precision Agriculture
https://doi.org/10.1007/s11119-020-09736-0

Deep neural networks for grape bunch segmentation in natural images from a consumer-grade camera

R. Marani[1] · A. Milella[1] · A. Petitti[1] · G. Reina[2]

Abstract
Precision agriculture relies on the availability of accurate knowledge of crop phenotypic traits at the sub-field level. While visual inspection by human experts has been traditionally adopted for phenotyping estimations, sensors mounted on field vehicles are becoming valuable tools to increase accuracy on a narrower scale and reduce execution time and labor costs, as well. In this respect, automated processing of sensor data for accurate and reliable fruit detection and characterization is a major research challenge, especially when data consist of low-quality natural images. This paper investigates the use of deep learning frameworks for automated segmentation of grape bunches in color images from a consumer-grade RGB-D camera, placed on-board an agricultural vehicle. A comparative study, based on the estimation of two image segmentation metrics, i.e. the segmentation accuracy and the well-known Intersection over Union (*IoU*), is presented to estimate the performance of four pre-trained network architectures, namely the AlexNet, the GoogLeNet, the VGG16, and the VGG19. Furthermore, a novel strategy aimed at improving the segmentation of bunch pixels is proposed. It is based on an optimal threshold selection of the bunch probability maps, as an alternative to the conventional minimization of cross-entropy loss of mutually exclusive classes. Results obtained in field tests show that the proposed strategy improves the mean segmentation accuracy of the four deep neural networks in a range between 2.10 and 8.04%. Besides, the comparative study of the four networks demonstrates that the best performance is achieved by the VGG19, which reaches a mean segmentation accuracy on the bunch class of 80.58%, with *IoU* values for the bunch class of 45.64%.

Keywords Deep learning · Image processing · Grape bunch segmentation · RGB-D sensing

✉ R. Marani
roberto.marani@stiima.cnr.it

[1] Institute of Intelligent Industrial Technologies and Systems for Advanced Manufacturing, National Research Council, via G. Amendola 122D/O, 70126 Bari, Italy

[2] Department of Mechanics, Mathematics and Management, Polytechnic of Bari, Via Orabona 4, 70125 Bari, Italy

Introduction

The accurate knowledge of the crop characteristics at the sub-field level is crucial in precision agriculture. In the context of vineyard management, it is even more important, since early treatments through plant monitoring can increase the harvest yield, leading to significant economic benefits. Conventional approaches are based on human interventions, i.e. few analysis campaigns during which agronomists perform visual and even destructive inspections of each row of the vineyard. This process is highly time-consuming, subjective, and prone to human errors. However, the recent high-throughput phenotyping platforms (Reina et al. 2016) have opened further possibilities for the automatic inspections of cultivated fields. Specifically, auto-guided agricultural vehicles with perception systems, made of a sensor platform and a processing unit with dedicated algorithms, may generate dense information about the field.

Sensing a natural environment is a complex task that requires multi-domain information, ranging from the complete gathering of the color appearance and of three-dimensional models. The need for complete data affects the selection of the required technologies of the sensor platforms. As a first example, LIDAR sensors offer the possibility of mapping the field, enabling autonomous navigation, while simultaneously estimating the crop shapes (Bietresato et al. 2016; Malavazi et al. 2018). However, differences in color and shapes of the canopy and ambient light changes can produce outliers and artifacts in the 3D models. Besides, LIDARs are expensive sensors and do not enable the visual inference of the crops. For this reason, RGB-D cameras have been proposed as an alternative solution to recover 3D colored models of the plants. Among this family of sensors, the Microsoft Kinect (v1, v2 and Azure versions by Microsoft, Redmond, WA, USA) have been often used in several applications, including those related to precision agriculture (Chéné et al. 2012; Paulus et al. 2014; Narvaez et al. 2017). Although the resolution and accuracy of the Microsoft Kinects have increased along with the versions, keeping a consumer affordable cost, all the sensors are dedicated to indoor inspections and, thus, can effectively work only under laboratory conditions or in controlled environments, like greenhouses. In contrast, RGB-D cameras based on stereovision can work outdoor, i.e. in natural environments, still producing 3D models and RGB color images of the surroundings. Among this last category of sensors, the Intel RealSense R200 (Santa Clara, CA, USA) has attracted increasing interest since it is cost-effective and it can work under different conditions of ambient light, producing RGB and infrared images, together with 3D depth data. These features make it suitable for its use on-board an agricultural vehicle (Milella et al. 2019).

A high-throughput phenotyping platform can produce a huge volume of data, both colored and 3D, to be processed. Automatic analysis of such volumes of data is mandatory to extract informative contents of a high level, to assist agronomists in making targeted decisions. The accurate and reliable detection of fruits is an example of a complex task required from automated processing units involved in precision agriculture. This falls in the class of pattern recognition problems, and, more specifically, in the category of image segmentation, which, in general, refers to the partition of an image into a set of meaningful areas with similar semantic properties (Shapiro and Stockman 2001).

The process of image segmentation is a challenging task, even more complex when grape bunches are the target of the analysis. For instance, in the case of white grapes, the appearance of bunches, leaves, and background vegetation is very similar, since colors are comparable. This problem of segmentation of grape bunches from RGB color images has been tackled with several approaches based on conventional machine learning.

Murillo-Bracamontes et al. (2012) propose an object shape detection procedure implementing the Hough Transform to find the boundaries of the grape clusters. As a drawback, this technique is only demonstrated on laboratory images, where grapes are framed without any further object, such as cordons, trunks, or leaves. On the contrary, Reis et al. (2012) present a processing pipeline to distinguish grapes and bunch stems, regardless of light and brightness variations, from images acquired in natural environments. However, this procedure is applied to high-resolution images, which are preliminarily focused on the grape clusters. Liu and Whitty (2015) improve the previous results by Reis et al., removing the restrictions on the camera field-of-view and background. It is achieved by a Support Vector Machine (SVM) which receives color and texture features extracted from the input natural images. Although images are not too close to the single bunches and enable the fruit detection on large scales, input data are again of high quality and resolution. Besides, images are acquired under stationary conditions, without possible causes of degradation. Also Pérez et al. (2017) propose an SVM to process scale-invariant features transform (SIFT) and Bag of Features (BoF) to segment grapevine buds. Although this work focuses on the detection of a different part of the grapevines, it is of interest since it well performs on buds of both small and large sizes. Though it processes input images of resolution of 1600×1600 pixels, captured by consumer-grade cameras, but under controlled conditions (stable and well-focused images). Pérez-Zavala et al. (2018) present a comparative analysis of image feature vectors and SVMs to select the best strategy to separate bunches in single grapes, starting from the Histograms of Oriented Gradients (HOG) shape descriptor and the Local Binary Pattern (LBP) texture information. Luo et al. in their work of (2016), improved in (2018), describe a method for detecting summer black grapes onboard of a mobile vehicle moving a vineyard for undamaged fully-robotic harvesting. They propose several image processing blocks, achieving good results in bunch segmentation. Although results by Pérez-Zavala et al. and Luo et al. are encouraging, experiments are still performed on images of good quality, framing the grape bunches without invasive contributions of the leaves and the background. Standard machine learning approaches are also used to process dense 3D point clouds (Rist et al., 2018) to count the number of berries of the grape bunch and to achieve precise information about their shape. The capabilities of the so-called 3D-Bunch-Tool are tested on input data acquired under both laboratory and natural conditions but still focused on single bunches, without occlusions or other objects. A recent work by Lin et al. (2020) proposes the simultaneous analysis of RGB images and 3D point clouds acquired in the field by a Kinect v2 to segment clusters of fruits (pepper, eggplant, and guava). Although results reveal that the proposed algorithms are potentially universal, i.e. also for the segmentation of grape bunches, few points limit its applicability under the conditions of this study. All images are taken statically and, therefore, all issues related to motion blurring are not faced. Besides, the proposed sensor, which bad performs outdoor, is preliminary calibrated to find both the intrinsic and extrinsic camera parameters. Performing this task is time-consuming and requires competences in computer vision. For these reasons, it could be often unpracticable for standard final users.

Conventional machine learning solutions require data preprocessing aimed at highlighting significant features, selected by human experts. On the contrary, these solutions are very limited in processing the raw data produced by the sensor platforms. Recently, conventional machine learning approaches have been replaced by deep learning-based ones (LeCun et al. 2015), which belong to the wider category of representation learning. In this case, artificial networks are made of consecutive layers of filters which generate a hierarchical representation of the phenomena (or of the corresponding data), by producing maps of lower size but of increased meaning. This methodology has rapidly gained attention in many applications where complex understanding is mandatory, such as object detection form 2D (Cordts et al. 2016) and 3D (Maturana and Scherer 2015) data, remote sensing (Zhang et al. 2016), game analysis (Silver et al. 2017), and medical imagery (Litjens et al. 2017), just to mention a few.

In the agricultural context, deep learning has become the most promising technique in applications targeted at fruit detection and counting (Kamilaris and Prenafeta-Boldú 2018). Sa et al. (2016) present a Faster Region-based Convolutional Neural Network (Faster R-CNN) to detect seven kinds of fruit, such as sweet peppers and rockmelons, in natural images from greenhouses. Bargoti and Underwood (2017) export the previous results by Sa et al. to the more complex understanding of outdoor images of orchards. Rahnemoonfar and Sheppard (2017) propose a technique for data augmentation by feeding the network at the training phase with simulated images of fruits. This approach overcomes the need for huge datasets in training network architectures without any initialization. Recently, Koirala et al. (2019) present a comparative study on several deep learning architectures for the detection of mango fruits in orchards. Dataset images are acquired at night from vehicles equipped with lighting sources to keep control of the brightness level.

Parallel to the study of images of orchards, recent activities have focused on the use of deep learning also for the analysis of natural images of grapevines. For instance, Pereira et al. (2019) propose a deep neural network for the identification of grape plant species. In this case, a pre-trained net, namely the AlexNet (Krizhevsky et al. 2012), is tuned to recognize six varieties of red grapes in natural images of high resolution (5184 × 3456 pixels). Also Boulent et al. (2019) present the application of a deep learning framework for the identification of grapevine pest and disease, namely downy mildew (*Plasmopara viticola*) from RGB pictures, manually taken in the field. A recent work by Silver and Monga (2019) falls in the same context of this paper. It estimates vineyard grape yield from sets of images taken by a smartphone. The authors show that the combination of image processing and deep learning approach can lead to accurate estimations at harvest time when transfer learning is used.

All the above mentioned solutions are proposed to analyze and understand images, even natural, captured under specific conditions. For instance, datasets are always acquired from stable points-of-view at fixed distances from the target plants or using high-resolution hardware. Even if images come from low-cost hardware, as in Silver and Monga (2019), data are acquired manually by human operators. In other words, humans may drive the image acquisition, implicitly selecting the best conditions. These images are not representative of the context of interest, where autonomous high-throughput phenotyping platforms gain the data from the environment: mobile platforms take the data directly as they are proposed to the sensors.

The first solution to the problem of bunch detection in natural images captured by low-cost consumer-grade RGB-D sensors on a moving agricultural vehicle is proposed by Milella et al. (2019). In this case, the dataset is processed by four pre-trained deep neural networks to detect grape bunches and enclose them in bounding boxes. This work is further improved by Marani et al. (2019), where a pre-trained AlexNet is focused on the classification of bunches within windowed portions of the input images. Then data are rearranged to achieve the exact per-pixel segmentation of the detected bunches. It results in the correct segmentation of 87.5% of the grape bunches of the test images.

This paper extends the previous work by Marani et al. (2019) for the segmentation of grape bunches in actual natural images from consumer-grade RGB-D cameras on mobile platforms. The proposed study aims at the most difficult target of detecting white grapes. Here, the specific ripeness level of the fruits makes the color appearance of the grapes very similar to the green leaves of the grapevines and to the vegetation of the background. The original contribution is three-fold:

- A comparative study on the performance of four pre-trained network architectures, namely the AlexNet, the GoogLeNet (Szegedy et al. 2015), the VGG16, and the VGG19 (both VGG16 and VGG19 are from Simonyan and Zisserman 2014), is presented. All investigations are performed on color images, neglecting the depth channel provided by the RGB-D camera used in the proposed experiments. This is to state whether the current form of this simple architectures is suitable to solve the task of segmentation of white grapes from low-quality natural images;
- Two metrics dedicated to image segmentation (pixel-by-pixel analysis), i.e. the segmentation accuracy and the well-known Intersection over Union (IoU) (Rezatofighi et al. 2019), are evaluated;
- A novel strategy for labeling bunch pixels is proposed. It is based on an optimal threshold selection of the bunch probability maps, instead of the conventional minimization of cross-entropy loss of mutually exclusive classes.

As a first result, the introduction of the new strategy for pixel labeling improves the performance of the deep neural network in image segmentation. Besides, the comparative study of the proposed networks demonstrates that the best performance is achieved by the VGG19, which reaches a mean segmentation accuracy on the bunch class of 80.58%, with IoU values for the bunch class of 45.64%. Alternatively, the AlexNet performs similarly in terms of IoU of the bunch class, but with lower values of the mean segmentation accuracy of the bunch class (74.70%). Anyway, its easier structure makes it more suitable for the fast processing of the input images.

The paper is organized as follows: it first describes the input datasets, the four networks, and the strategy for the final image segmentation based on the optimal threshold operation of the bunch probability map; then it discusses the results of the training and test phases; the last part of the manuscript adds final comments and remarks on future activities.

Materials and methods

This paper investigates the capability of four deep neural networks in the segmentation of grape bunches in natural images. Images are captured by consumer-grade hardware in real scenarios, from a moving vehicle, while processing is performed off-line at the end of the acquisitions.

The research was based on the following hypotheses:

- Input images:
    - o Fruits, leaves, wooden elements, and constitutive elements of the vineyard have a similar, but different appearance under daylight illumination. Human experts can always segment the acquired image, even if it requires high efforts;
    - o Images are acquired by consumer-grade hardware of small resolution on a moving platform. Although the image quality decreases as the effect of image blurring or compression artifacts, it is still possible to discern significant information for the exact segmentation;

  - During the whole acquisition, the camera frames the plants from a front view, within its depth-of-field;
  - Images under direct (opposite) sunlight are not considered since they are overexposed and colors saturate to white.

- Processing:

  - The use of deep neural networks can improve results in terms of segmentation accuracy since features are extracted from the images to obtain the best representation of the inputs;
  - Per-pixel segmentation can gain accuracy from multiple classifications of regions (patches) that insist on the same pixel to be labeled.

These hypotheses were contrasted to fulfill the goal of segmentation of targets of interest in natural images by:

- The application of pre-trained networks, re-adapted on this specific target. Results will be compared in terms of well-established metrics (pixel-by-pixel analysis and Intersection over Union);
- The development and testing of a novel strategy for the comparative analysis of classification scores aimed to achieve the best result in the labeling of bunch pixels.

The next sections will describe with more details the in-field hardware, the dataset used for the numerical experiments, the architectures of the deep neural networks, and the strategy implemented for the final image segmentation.

Input dataset

The scope of this work is the processing of natural images of resolution $W \times H$, captured in a vineyard, aimed at the exact segmentation of interesting objects, such as wooden elements (cordons, cranes, and trunks), plant leaves, and fruit clusters. The latter is the actual target of the whole processing since precise segmentation of grape clusters is particularly useful for the yield prediction and the estimation of important phenotyping traits.

The proposed architectures are intended to work on datasets of images acquired in real scenarios, using commercially available low-cost hardware. As stated previously, experiments performed under laboratory conditions, with specialized hardware, and/or by humans, can produce altered results which are practically not replicable in real conditions. Images were thus acquired by a consumer-grade hardware, namely the Intel RealSense R200 (Santa Clara, CA, USA) RGB-D camera, which takes RGB color images and depth maps, also outdoor. This camera is made of a stereo pair, which produces color images of a depth of 8 bits for each color channel. The presence of a stereo pair, together with an IR projector, allows the simultaneous acquisition of depth maps coded in a buffer of 16 bits. The resolution of the depth map is limited to the VGA resolution (640 × 480 pixels), whereas the color stream can reach a maximum resolution of 1920 × 1080 pixels (Full HD resolution). Nevertheless, both streams were kept to the VGA resolution to preserve data consistency ($W = 640$, $H = 480$).

This work focuses on the single stream of color images taken from the left camera (accordingly to the perspective of the camera module). Depth streams will be exploited in future investigations when grape bunch segments will be modeled in three dimensions to evaluate their size (see the conclusions of the manuscript for further discussions).

The Intel RealSense R200 was placed on a mobile platform, i.e. the commercial Niko caterpillar (Bühl/Baden, Germany), which was guided throughout a commercial vineyard of a white variety of wine grape (Räuschling) in Switzerland (N 47°14′ 27.6″, E 8° 48′ 25.2″). Datasets were completed in two days of acquisitions, during which the vehicle followed specific paths within the parallel rows of grapes (row spacing between 1.5 and 2 m) at an average speed of 1.5 m/s. The RGB-D camera pointed laterally at the rows of grapevines at a distance in the range between 0.75 and 1 m, producing video streams at a framerate of 5 *fps*. Images were acquired using the *librealsense* library (Librealsense online documentation, 2020), working on the Robot Operating System (ROS). The whole configuration resulted in about 500 RGB color images of grapevines.

Among the whole set of images, 84 were picked and manually annotated to divide five classes of interest:

- Bunch class: all pixels belonging to the bunches of white grapes (of main interest along these lines);
- Pole class: all pixels belonging to the vineyard poles;
- Wood class: all pixels belonging to wooden parts of the grapevines, such as trunks and cordons;
- Leaves class: all pixels whose appearance can be referred to leaves;
- Background class: all pixels not belonging to none of the previous classes.

An example of RGB natural images acquired by the proposed hardware and the corresponding annotated ground truth is reported in Fig. 1. Here, each color of the ground truth represents a specific class.

From the left column of Fig. 1, it is easy to understand that the process of image segmentation is of high complexity because of the quality of the input data generated in a real scenario. In particular:

- RGB color images suffer from motion blurring and JPEG compression artifacts;
- Target objects, such as grape bunches, have small sizes on the image, as the effect of the low resolution of the camera and the vehicle-plant distance.

The effect of image degradation is particularly critical for grape bunches. Their recognition and exact segmentation are the most challenging since their appearance is comparable to the background. For instance, image magnifications in Fig. 2 show typical grape clusters. The bunch of white grapes enclosed in the yellow circle was captured under lighting conditions which made its color appearance very similar to the vegetation of the background. This issue is typical of white grapes at the specific ripeness level of the ones of this study.

Image segmentation is the result of the classification of patches of known, small size. Focusing on the scheme of Fig. 3, a patch is a square-shaped window of size $N \times N$, which moves with a sample step (stride) of S_w in both dimensions on the input image. Every windowed patch is labeled with a specific class type, depending on the objects of the ground truth enclosed by the moving window. If a patch insists over a grape bunch, i.e. the number of labeled pixels of the bunch class is higher than a threshold, the whole patch is classified as a grape bunch. This process generates examples, i.e. couples of known input ($N \times N$ RGB color images) and output (classes).

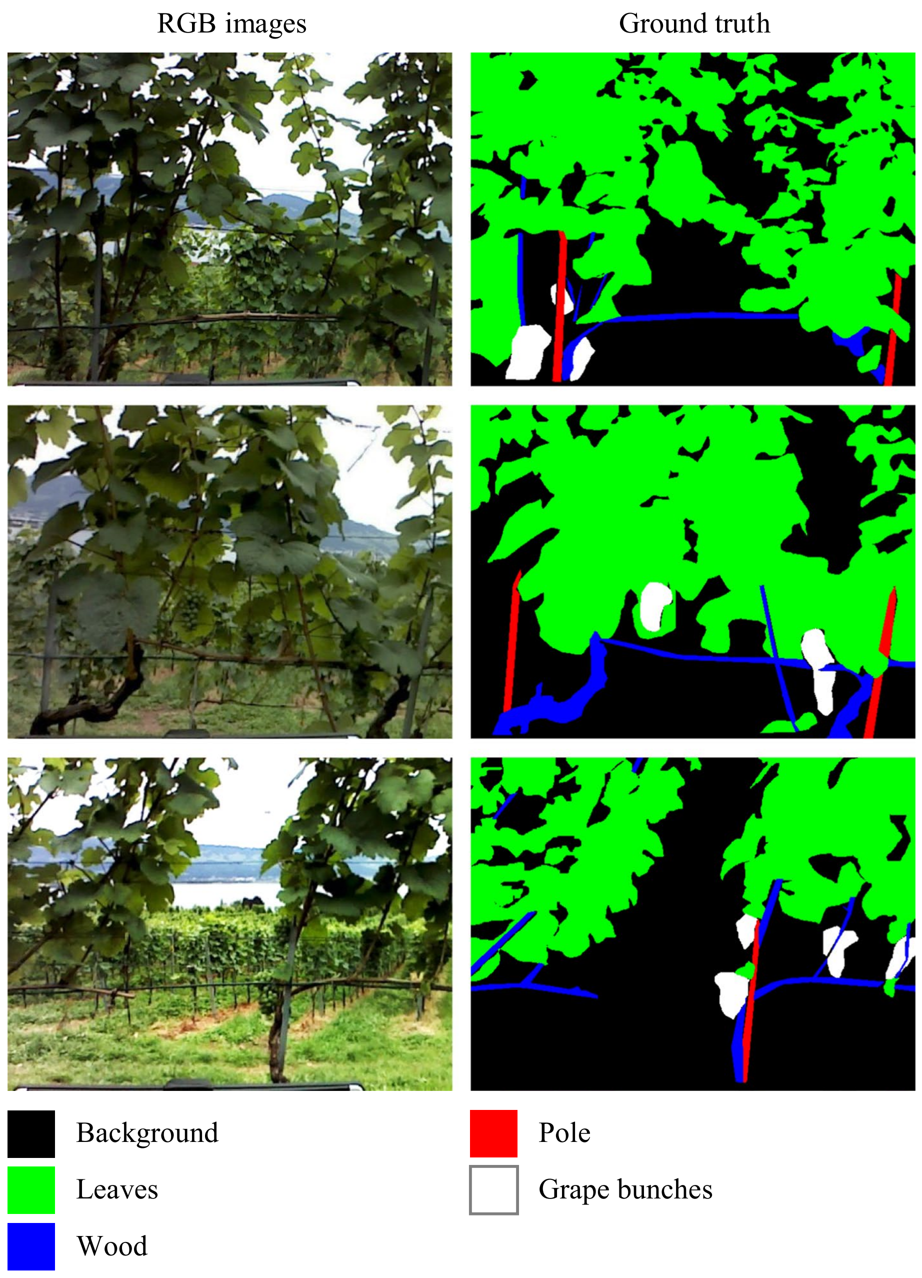

Fig. 1 Set of input images (left column) acquired by the camera and corresponding image annotation with the 5 classes of interest. Each color refers to a class. See the legend for color correspondence (Color figure online)

The selection of the class of each windowed patch is performed following the same flow scheme of Milella et al. (2019), shown in Fig. 4. Since the segmentation of the

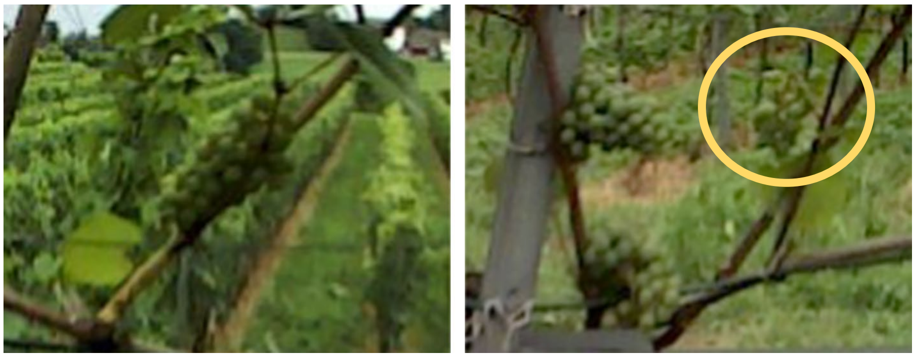

Fig. 2 Image magnification of areas enclosing grape bunches, whose appearance is comparable to the background. The yellow circle encloses a hardly visible grape cluster (Color figure online)

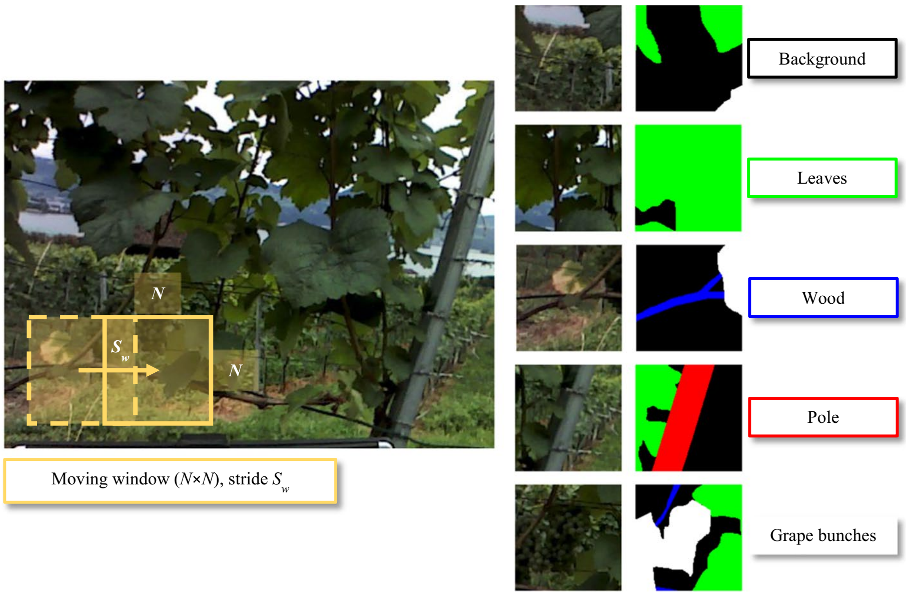

Fig. 3 Representation of the process of dataset generation. A square window (in yellow) of size $N \times N$ moves into the image with a stride S_w, and frames patches, which are labeled with a class, depending on the ground truth (Color figure online)

grape bunch is the aim of the whole processing, the pixels of grape clusters are first counted. If the number of bunch pixels N_{bunch} is higher than 20% of the whole number of pixels, the patch is labeled as a bunch, regardless of the content of the remaining 80% of the patch. The process continues counting the number of wood pixels, pole pixels, and leaves pixels. In the end, if none of the previous classes satisfies the conditions in Fig. 4, the patch is labeled as background. Threshold values have been set

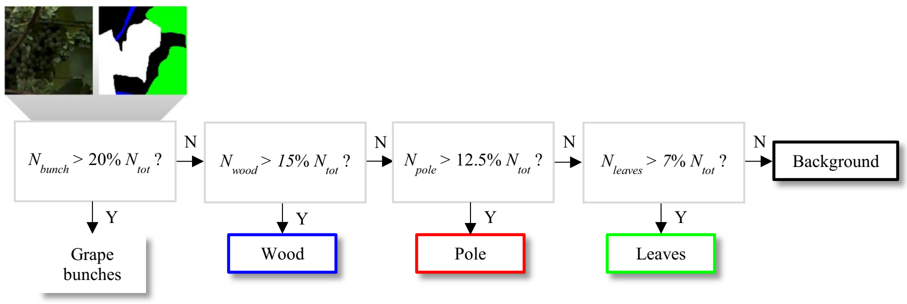
N_{bunch} > 20% N_{tot} ?
N
N_{wood} > 15% N_{tot} ?
N
N_{pole} > 12.5% N_{tot} ?
N
N_{leaves} > 7% N_{tot} ?
N
Background
Y
Grape bunches
Y
Wood
Y
Pole
Y
Leaves

Fig. 4 Scheme for patch labeling as in Milella et al. (2019). The corresponding ground truth is framed, and its pixels are counted to determine the population of each class. The single label of the whole patch is thus selected to favor the classes with smaller populations

experimentally to emphasize the less populated classes, thus resulting in a balanced dataset.

Network architectures

In deep learning, chains of filters and non-linear operations are applied to the input data, i.e. images. When the data flow through the filters, from the input to the output, newer representations of smaller size, but with richer meanings are created. The filter kernels are determined during a training phase when the net receives known examples. At this stage, the net automatically searches for the best data representations, i.e. features, that can introduce the highest discrimination among the output classes, similarly to what domain experts do in approaches of conventional machine learning.

The main drawback of such schemes is the need for large datasets to train from scratch the millions of weights of the whole net. When large datasets made of million labeled examples are not available, it is possible to leverage knowledge from a similar domain and transfer it to perform a new task. This strategy is known as transfer learning (Yosinski et al., 2014). This theory assumes that the first layers of the nets similarly interpret the data, regardless of the specific goal. In image classification, the first layers of the nets typically try to extract standard features, such as corners and edges. If an existing pre-trained network is used, it is not necessary to intensively train again these first layers, since the results would be close to the initial condition. The pre-trained network is thus gently tuned to perform the new task. Weights change their values from the initial conditions to the final convergence to achieve the best classification with much more efficiency.

In this paper, the classification of the input patches is performed by using four pre-trained deep neural networks, namely the AlexNet, the GoogLeNet, the VGG16, and the VGG19. All these networks are on a common basis: they are designed to classify images among the same number of generic classes from the ImageNet dataset (https://www.image-net.org). Among all the possible networks working on this dataset, only these four ones have been selected since a comparative benchmark proposed by Russakovsky et al. (2015) shows that the GoogLeNet, the VGG16, and the VGG19 produce the best all-time results in image classification and single-object detection., whereas the AlexNet produces the best results of 2012, with a significantly less complex architecture than the other three networks, proposed in 2014.

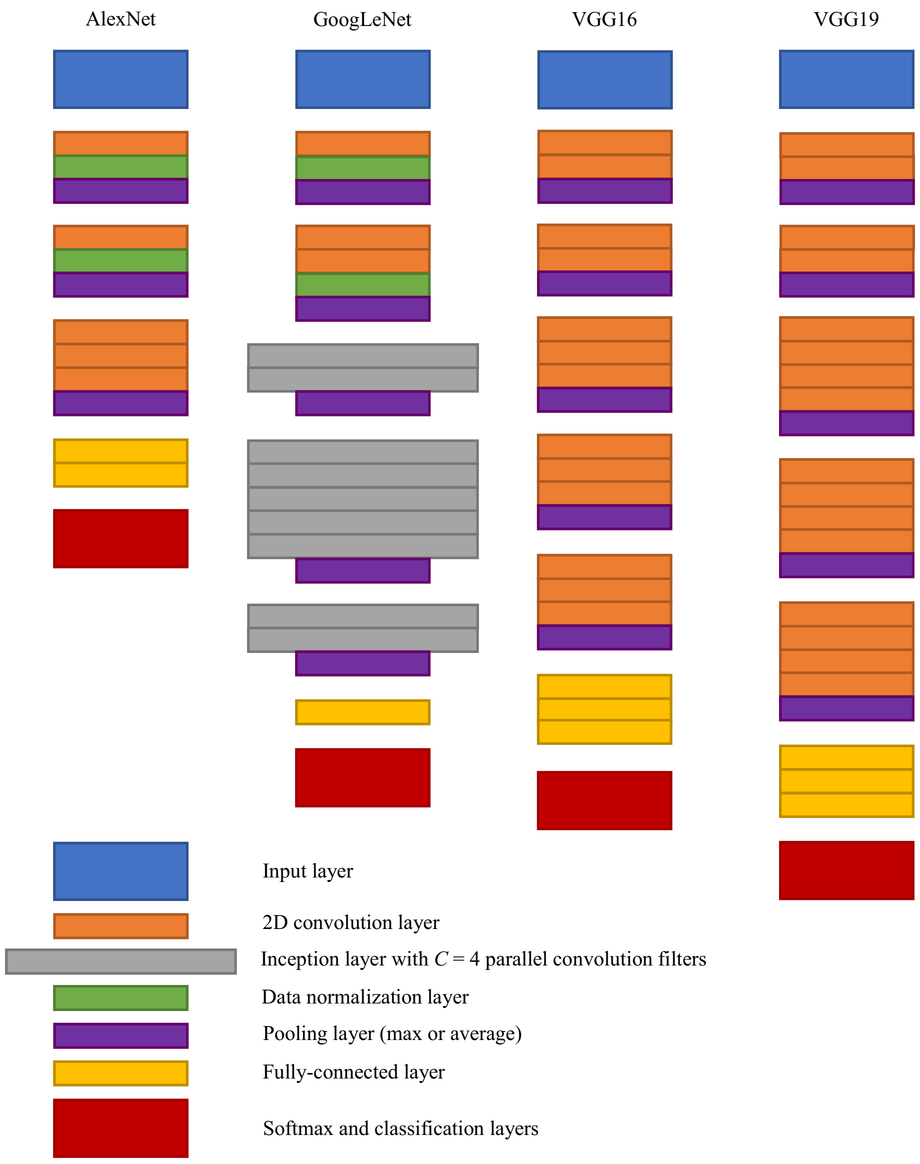
AlexNet
GoogLeNet
VGG16
VGG19
Input layer
2D convolution layer
Inception layer with $C = 4$ parallel convolution filters
Data normalization layer
Pooling layer (max or average)
Fully-connected layer
Softmax and classification layers

Fig. 5 Standard architectures of the AlexNet, GoogLeNet, VGG16, and VGG19 deep neural networks

A schematic representation of the four networks is depicted in Fig. 5, which also shows their complexities at a glance. Each architecture of Fig. 5 is made of an arrangement of consecutive blocks (or layers), which perform complex operations, even non-linearly. Typical blocks are:

- Input processing layer (blue box): data feed the network at the first layer into the input processing layer. Depending on the architecture, this layer can accept different input sizes $I \times J \times 3$.
- 2D convolution layer (orange box): it is made of a set of N_f 2D convolutional filters which work parallelly on the same inputs and produce N_f output maps, sum of the results of the convolution of the i-th filter kernel ($i = 1, \ldots, N_f$) to every input. Kernels have a size of $K_i \times K_i$. A 2D convolutional layer is often followed by an activation layer, which applies the REctified Linear Unit (ReLU), i.e. a threshold operation with threshold level set to zero).
- Pooling layer (purple box): it groups neighbor data within the input maps and outputs only the highest (max-pooling) or the average (mean-pooling) value of the input group. The neighborhoods are grouped by a moving mask of size $P \times P$, with a sample step (stride) of $S_{p,row} \times S_{p,col}$. The output map has a lower size than the input one, but it is only filled by significant information.
- Inception layer (gray box): it is an extension of the 2D convolution layer, aiming at allowing the net to select the most proper kernel size for the processing of the input maps. In general, it is an arrangement of C parallel convolution layers, with different kernel sizes. In the case of the GoogLeNet, which is the only net to use this layer, C is equal to 4. The kernel sizes of the first three branches are equal to 1×1, 3×3, and 5×5. The last branch of the inception layer is a sequence of a max-pooling layer ($P = 3$) and a convolution layer with kernel size 1×1. Results are finally concatenated.
- Data normalization layer (green box): it performs data normalization among all inputs to regularize parallel features which span over different ranges of amplitude. This layer is typically placed on the top of the network, where map amplitudes are like those of input data. Flowing through the architecture, values are no longer related to the inputs and, thus, the data normalization layer may become redundant.
- Fully-connected layer (yellow box): it implements a fully connected layer, made of U units, which return a weighted summation of the inputs and a bias. Since the layer is fully connected, it is possible to write it in a matrix form: the layer linearly multiplies the input by a matrix of weights and then add a vector of bias. If the fully-connected layer is at the end of the network, U is also the number of the final mutually exclusive classes.
- Softmax and classification layers (red box): this block is the sequence of two layers. The softmax layer first assigns a probability for each class, for that specific input (classification score). The sum of all scores is equal to 1.0. Then, a classification layer receives the set of input classification scores for the mutually exclusive classes, computes the cross-entropy loss, and outputs the result of the classification (the class with the lowest loss).

The four deep neural networks in Fig. 5 are all feed-forward and designed to label input square images between 1000 possible classes. This means that the last fully-connected layer is always made of $U = 1000$ units. Nevertheless, each architecture has its characteristics:

- AlexNet: it is made of 25 layers. It accepts image patches of size $227 \times 227 \times 3$ which are then passed by a first 2D convolutional layer of 96 filters, with size 11×11. The output is further convolved by four other layers (the first with a kernel size of 5×5 and the other of a size of 3×3). Normalization layers and max-pooling layers are properly placed within the network. At the end of this chain, there are 256 parallel representations of size 6×6 of the input patch. The 256 6-by-6 representations are finally rearranged in a feature vector of 9216 entries, which passes through two fully-connected layers (4096 and 1000 units each). The network can thus perform the classification of the input with the sequence of softmax and classification layers.

- GoogLeNet: it is made of 144 layers. As input, it accepts patches of size equal to $224 \times 224 \times 3$ pixels. The first segment of the network performs standard 2D convolutions, data normalization, and max-pooling so that at the end of the sequence, there are 192 parallel representations of the data of size 28×28. This set of data flows through a sequence of two inception layers with $C = 4$ and then is max-pooled to have 480 parallel representations of the data of size 14×14. These representations are processed by 5 inception layers and a max-pooling layer, becoming 832 parallel representations of size 7×7. These are finally processed by the last two inception layers and a mean-pooling layer, producing a final vector of 1024 features. This final vector is then classified by the standard classification sequence made of a fully-connected ($U = 1000$), a softmax, and a classification layer.
- VGG16: it is made of 41 layers. As the GoogLeNet, input images must be of $224 \times 224 \times 3$ pixels to be accepted by the input layer. It is a sequence of two segments of convolutional filters. The first segment is made of two blocks of two convolutional layers and a max-pooling layer. It produces 128 parallel representations, having size 56×56 pixels, of the input data. These representations feed the second segment made of three blocks of three convolutional layers and a max-pooling layer. At the end of the second segment, input data are represented by 512 maps of size 7×7. These are rearranged in a feature vector of 25,088 entries and placed at the input of a sequence of three fully-connected layers with U equal to 4096, 4096, and 1000, respectively. The softmax and classification layers end the architecture, providing the final label.
- VGG19: it is an extension of the VGG16, made of 47 layers. The first segment of the network architecture is completely equal to the corresponding part of the VGG16. On the contrary, the second segment adds one convolution layer to the three constitutive blocks, before the max-pooling layers. This difference does not alter the size and the number of representations of the input data, which are still rearranged in a feature vector of 25,088 entries. The final layers of the architecture are the same as the VGG16.

The strategy of transfer learning is thus applied to all the proposed networks. Since the required number of the target classes is equal to 5, the last fully-connected layer of every deep network is downsized to $U = 5$. It is thus not initialized and is the only layer that will be trained out of the scratch. It means that AlexNet, the VGG16, and the VGG19 have 4096×5 weights and 5 biases to be completely trained, whereas the GoogLeNet has only 1024×5 weights and 5 biases to be determined without any initial condition.

Image segmentation with probability map analysis

All the networks are designed for the task of image classification. Given an input image of a fixed square size, they label it with the class that best matches the informative content of the input image. Image segmentation can be reached by gathering classification scores into probability maps. The whole classification of the inputs must be placed in a more complex pipeline made of the following steps:

1. A moving window of size $N \times N$ moves with a spatial sample step S_w in both dimensions on the input image to extract square patches, similarly to what proposed for the generation of the dataset.
2. Each input patch is tagged with the position (i_{p1}, j_{p2}) of the center of the moving window, where $p1$ and $p2$ are elements of the series $N/2$, $S_w + N/2$, $2 \cdot S_w + N/2$, …, bounded by the input frame resolution $W \times H$ (in this case 640×480).
3. Every input patch is resized to $I \times J$ pixels to meet the specification of the input layer of the deep neural network under analysis and then classified accordingly.
4. Two results are taken from of the classification of each input patch at (i_{p1}, j_{p2}):
   a. The classification score of the five classes after the softmax layer;
   b. The single label determined by the cross-entropy loss analysis, at the output of the whole net (standard classification result).

It is worth noticing that the computational cost of this processing depends on the number of patches to be classified. This number is squared proportional to the inverse of the stride S_w and directly proportional to W and H. On the contrary, it is independent of the size N of the input patch since their size must be rescaled to the fixed allowed value of the input layer of the deep network ($I \times J$).

The two results of the per-patch classification are related to the position (i_{p1}, j_{p2}) of the input patch. Therefore, the whole analysis leads to two results:

- Standard segmentation: every single label generated at (i_{p1}, j_{p2}), after the classification layer, is placed accordingly in a final prediction map, whose size only depends on the stride S_w chosen for the moving window. If $S_w = 1$, the size of the final prediction map is $(W - N + 1) \times (H - N + 1)$. In this case, the segmentation is driven by the minimization of the cross-entropy loss computed among the five mutually exclusive classes.
- Set of five probability maps: the softmax layer produces a vector of five classification scores at (i_{p1}, j_{p2}). Each entry of the vector fills the corresponding probability map, one for each class, in the positions subtended by the initial window. In this way, each point of the probability map accumulates many contributions. For instance, if $S_w = 1$, each point of the probability map collects (N–1) classification scores. By averaging the whole contributions insisting at the pixel, it is possible to obtain five smooth probability maps. In this case, the probability maps have the same 2D dimensions of the input images, namely $W \times H$ pixels.

The availability of probability maps allows a different strategy for image segmentation, aimed at the best result in terms of the Intersection over Union (*IoU*) parameter (Rezatofighi et al. 2019). This metric is often used for similarity analysis between objects of arbitrary shapes, such as image segments or corresponding bounding boxes. The *IoU* computed for the generic c-th class ($c = 1, \ldots, 5$) has the following formulation:

$$\mathrm{IoU_c} = \frac{\mathrm{S_c} \cap \mathrm{S_{c,GT}}}{\mathrm{S_c} \cup \mathrm{S_{c,GT}}} \tag{1}$$

where S_c is the set of segmented points and $S_{c,GT}$ is the corresponding set of points of the ground truth for the c-th class. Since detected and ground truth segments are normalized to their area extensions, the *IoU* is invariant to the scale. Its values are thus in the range between 0 (segment dissimilarity) and 1 (perfect segmentation).

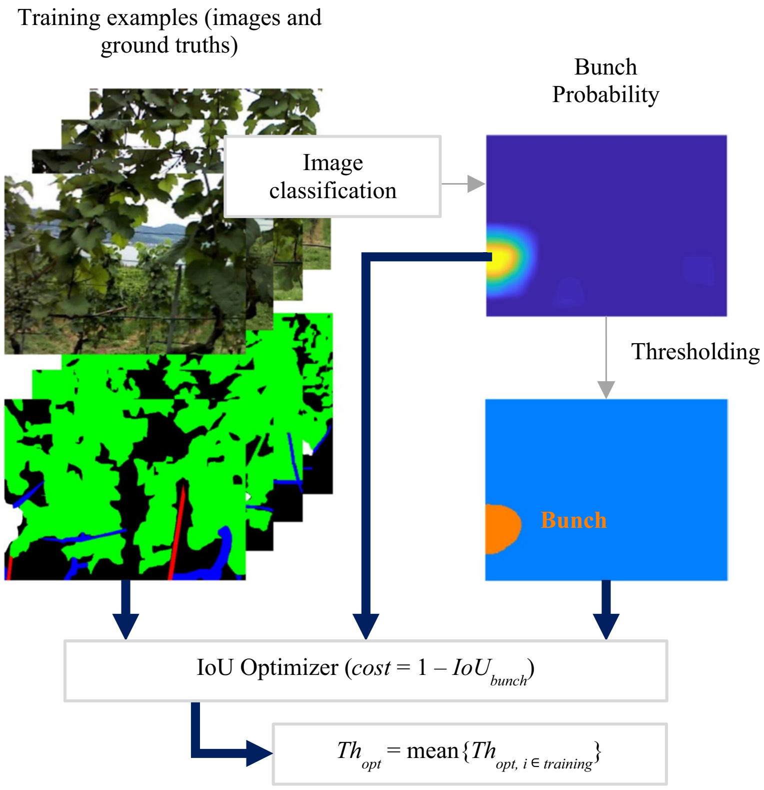
Training examples (images and ground truths)
Image classification
Bunch Probability
Thresholding
Bunch
IoU Optimizer ($cost = 1 - IoU_{bunch}$)
$Th_{opt} = \text{mean}\{Th_{opt,\, i \in training}\}$

Fig. 6 Selection of the probability threshold Th_{opt} for the bunch class. Training examples are used to iteratively set the probability threshold to minimize the cost function 1–IoU_{bunch}. The outputs of the optimization on each input example of the training set ($Th_{opt,\, i \in training}$) are finally averaged

Image segmentation is performed by a threshold operation on the probability map of the bunch class. Regarding Fig. 6, the set of training examples, which are known and already exploited for training the deep neural networks, is the input of an IoU_{bunch} optimizer. It processes every training image and the corresponding bunch probability map, given by the deep neural network, and searches for the best threshold value (Th_{opt}) that optimizes the objective function 1–IoU_{bunch}. The final Th_{opt} is defined as the average of the single values that optimize IoU_{bunch} for each input image of the training set.

During testing, the threshold operation is applied on the bunch probability map for segmenting grape bunches. The remaining areas are instead labeled with the other classes following the standard classification. In this case, the cross-entropy loss is computed considering the four classes other than the bunch one.

It is important to state that the proposed strategy for image segmentation emphasizes the detection of grape bunches, which is relevant in this scope, by maximizing the IoU_{bunch} value. Anyway, it is possible to favor any other class, e.g. the wood class, if its segmentation is the main target of the whole process.

Results and discussion

As stated in the previous sections, the input dataset of 84 images acquired by the Intel RealSense R200 sensor is divided into training and test sets.

The training set is generated by applying the moving window on 60 training images, with N and S_w parameters properly chosen. N must be high enough to completely frame a whole grape bunch. This condition depends on the relative distance between the camera and the grapevines, the field of view of the Intel RealSense and its resolution, and the fruit dimensions. For the proposed experimental setup, $N = 80$ resulted in a suitable value. On the other hand, the stride parameter S_w can be ideally equal to the unity, though it can lead to very similar images with small relative informative contents. However, increasing this value returns small populated datasets. For instance, if S_w is set to 5, the whole dataset becomes made of 7.7×10^5 examples, to be further divided into the training and test sets. This dataset size may be not enough to guarantee the convergence of the training, especially for the architectures with a higher number of layers, such as the GoogLeNet. A good balance between the size and the significance of the dataset can be reached with S_w equal to 3. As a result, the 60 training images generate about 1.5 million sub-patches for tuning the weights of the four pre-trained networks.

The remaining 24 images, randomly picked, constitute the test dataset. It will be used to show the capability of each pre-trained deep neural network and to quantitatively state their performance in terms of IoU and mean segmentation accuracy of the c-th class (MSA_c):

$$MSA_c = \text{mean}\left(\left. \frac{TP_c}{n_{obs,c}} \right|_{\forall image \in dataset} \right) \tag{2}$$

where TP_c is the number of pixels of the c-th class labeled in agreement with the ground truth (true positives or true detections), and $\mathrm{n}_{\mathrm{obs,c}}$ is the number of expected observations of the c-th class. The argument of the mean operator is computed for every image of the dataset, both training and test, alternatively. The MSA_c can be further averaged among the 5 classes to obtain the total segmentation accuracy MSA_{TOT}.

In summary, 84 input images generate 2104872 sub-patches, further divided into the training and test datasets, made of 1503480 and 601392 examples, respectively. Each patch is labeled with a specific class following the scheme of Fig. 4. The populations of each class of the training and test sets are detailed in Table 1.

It is worth noticing that the population of the grape bunch set is about 4.3% of the total number of examples, as a result of the strategy for pixel labeling, depicted in Fig. 4. This is the best achievable balance of the dataset, since background areas and plant leaves occupy

Table 1 Per-class population of the training and test datasets

Class	Population		
	Training	Test	Total
Background	863 938	318 930	1 182 868
Leaves	465 352	188 869	654 221
Wood	34 478	27 493	61 971
Pole	83 711	31 760	115 471
Grape bunch	56 001	34 340	90 341

most of the input images and, thus, their populations are implicitly higher than the ones of the other classes, including the grape bunch class. However, if the proposed strategy was not adopted, i.e. pixels of the five classes have the same weight (equal to 1), the resulting dataset would be heavily unbalanced: only about 0.03% of the resulting square patches of the whole dataset would be labeled as 'grape bunch'. As expected, the strategy in Fig. 4 mitigates this problem returning a much better balancing of the five classes.

For the sake of completeness, numerical investigations have been performed on a workstation equipped with a 64-bit Intel Xeon CPU (Santa Clara, CA, USA) with a clock frequency of 2.3 GHz, 192 GB of RAM, and a single Quadro K5200 GPU by NVIDIA (Santa Clara, CA, USA).

Training results

The first step is the training of the four deep neural networks with the training patches. With more details, the training of all the considered networks is defined as an optimization problem where the stochastic gradient descent with momentum (SGDM) solver is applied. The momentum is always set to 0.9. This process tries to converge to an optimal solution, at minimum cost (or loss), by heavily changing the weight of the last fully-connected layer (not initialized) and slightly tuning the transferred ones. For this reason, different learning rates are proposed for the transferred layers (lower) and the last fully-connected (much higher). Specifically, the learning rates of the first layers is set to 10^{-4}, whereas the last fully-connected layers of the four networks learn 20 times faster (learning rate of 2×10^{-3}).

These parameters allow the convergence of the learning phase performed on the AlexNet, the GoogLeNet, the VGG16, and VGG19, in 17, 15, 11, and 22 epochs, respectively. It corresponds to a total time requirement of 241, 1212, 2948, and 3450 min, respectively (Milella et al. 2019).

As an example of the result of the training, Fig. 7 shows an input image of the training set and the corresponding ground truth, together with the probability maps of the five classes, at the output of the softmax layer of each deep neural network. Although probability maps are comparable, quantitative results on test images may differ in terms of the final performance.

After the training of the deep neural networks, it is necessary to determine the optimal threshold value Th_{opt} for grape bunch segmentation, by running the optimization process with all the natural images of the training set. An example of the application of the IoU_{bunch} optimizer on a training image feeding the AlexNet is reported in Fig. 8.

The training image is first processed by the deep neural network (the AlexNet in the plots of Fig. 8) to create the probability map of the bunch class. A guess threshold Th in the range [0, 1] is used to create a binary image of the bunch probability map. This binary image is then compared with the ground truth to evaluate the corresponding IoU_{bunch}. The profile of the cost ($1–IoU_{bunch}$) as a function of Th is in Fig. 8d. The final classification performed by the threshold operation reaches an IoU_{bunch} of 62.77%, higher than the one obtained with the standard classification (cross-entropy loss minimization), which is equal to 57.89%. It is due to the different target of the two strategies: standard classification well performs in image labeling, but is less accurate when exact segmentation is required, producing an overestimation of the segment area.

The application of this segmentation strategy to the deep neural networks with the whole training dataset returns the results in Table 2, which shows the values of Th_{opt} and the corresponding mean values of the IoU_{bunch} resulting from the optimization ($IoU_{bunch,opt}$).

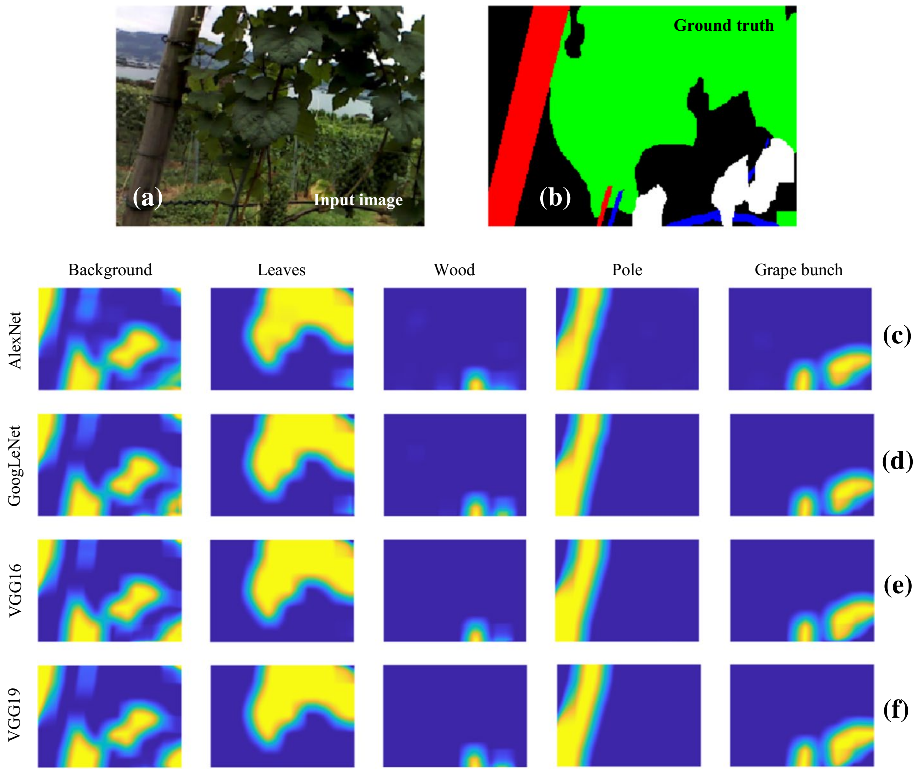

Fig. 7 Probability maps of each class obtained by processing a training image (**a**), displayed with its ground truth (**b**). Results are computed for the **c** AlexNet, **d** GoogLeNet, **e** VGG16, and **f** VGG19

The insight of Table 2 reveals that the GoogLeNet requires a lower threshold to achieve the best IoU_{bunch}. Therefore, this network produces low values of the bunch probabilities at the segment boundary, even if it correctly approaches the unity in the center of the segment. It is worth underlining that the example of Fig. 8 is an outlier in the distribution of the Th_{opt} values resulting from the analysis of the training images proposed to the AlexNet. It is shown to give more visual evidence to the consequent improvement of the segmentation result. However, most of the Th_{opt} values are below 9.19% and only three training images produce higher Th_{opt}, as in Fig. 8.

The results of the training in terms of MSA_{TOT} and MSA_{bunch} are in Table 3. As expected, for all networks, the MSA_{bunch} is higher than the total one, which also includes the contribution of the other classes. In other words, the segmentation is performed with more accuracy for the bunch class, which is favored by the segmentation strategy.

Test of the networks

The performance of the networks can be evaluated by processing input natural images, which have never been proposed to the networks before. The testing phase requires the same set of operations of the training: per-patch classification, collection of probability maps, and rearrangement of results to the final segmentation. The most cost is spent in

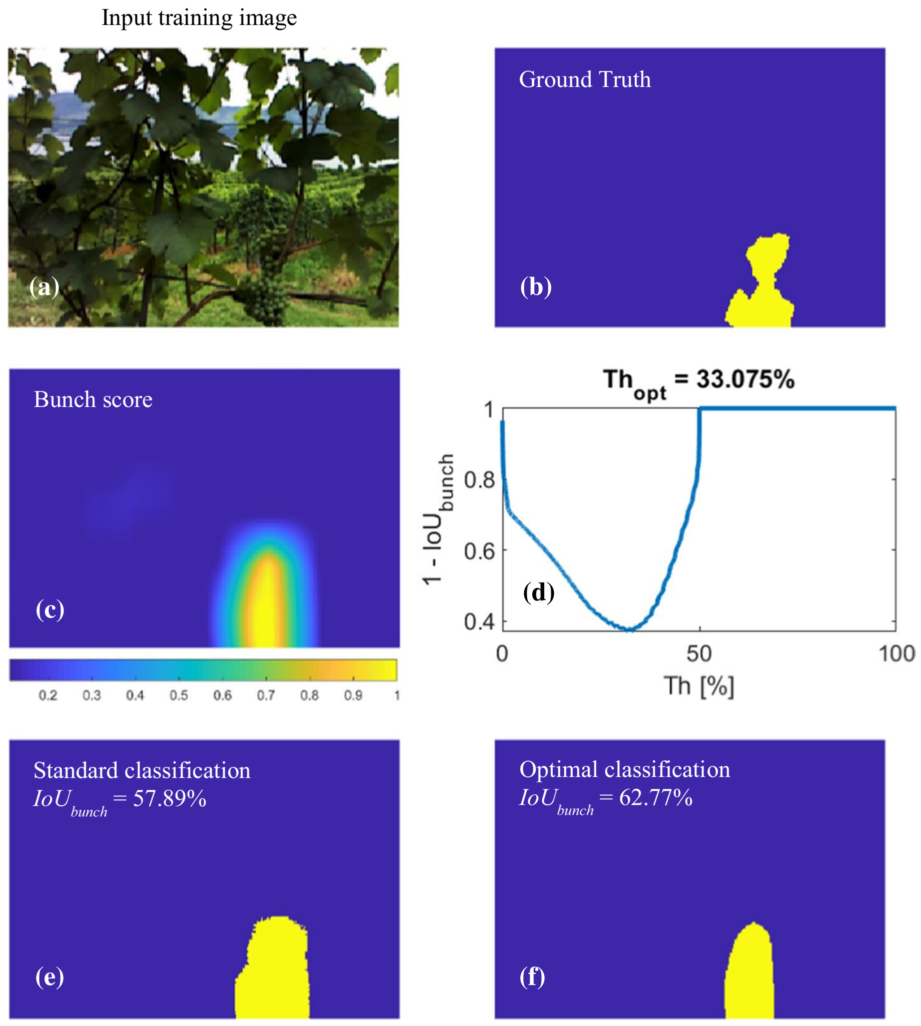

Fig. 8 Process of threshold selection via IoU_{bunch} optimization of training images. **a** Input image of the training set and **b** corresponding ground truth; **c** probability of the bunch class; **d** cost function ($1–IoU_{bunch}$) vs. a guess threshold in the range [0, 1] (the optimal value is $Th_{opt}=33.07\%$ which returns an $IoU_{bunch,opt}$ of 62.77%); **e** result of a standard classification layer with cross-entropy loss function; **f** result of the proposed classification with IoU_{bunch} optimization. These plots are the output of the AlexNet

Table 2 Probability threshold of the grape bunch class, selected after the process of optimization of IoU_{bunch}, performed on the training images, and final corresponding value

Net	Th_{opt} (%)	$IoU_{bunch,opt}$ (%)
AlexNet	6.71	50.26
GoogLeNet	3.28	54.13
VGG16	6.49	51.64
VGG19	7.26	47.79

Table 3 Mean segmentation accuracy from the application of the proposed nets to label the training images

Net	Mean segmentation accuracy	
	MSA_{TOT} (%)	MSA_{bunch} (%)
AlexNet	71.17	74.83
GoogLeNet	68.04	70.65
VGG16	67.73	73.47
VGG19	68.43	73.32

Values are computed for all the classes (MSA_{TOT}) and detailed for the bunch one (MSA_{bunch})

the classification of the input patches, whose size is set accordingly with the one of the training ($N = 80$) and stretched to the admitted input size ($I \times J$ of 227×227 pixels for the AlexNet and 224×224 pixels for the other networks). The execution of this task for a single patch on the proposed hardware takes 2.2 ms for the AlexNet, 26.9 ms for the GoogLeNet, 33.9 ms for the VGG16, and 41.6 ms for the VGG19 (Milella et al. 2019). In the test phase, the stride parameter S_w of the moving window has been set to 6 to speed-up the classification. Test images are thus divided into 50 350 sub-patches, individually processed to find the probability maps.

The analysis of the probability map of the bunch class and the threshold operation with the levels in Table 2, together with the application of standard classification on the other classes, produces the final segmentation of the test images. Fig. 9 shows an example of segmentation results produced by the four networks (Fig. 9c–f) on the test image in Fig. 9a, plotted with its ground truth (Fig. 9b). It is worth noticing that results are displayed as they are returned by the proposed processing, without the application of further processing, such as morphological filters. Although these further operations may improve results, they are too dependent on the specific outputs of the segmentation and thus may distort the comparison of the outputs of the deep neural networks.

The inspection of Fig. 9 denotes that the segments of the bunch, wood, and pole classes are overestimated, i.e. their extension is higher than the actual one in the image. This behavior is due to the preliminary selection of the labels of the patches of the training dataset. As stated previously, each training patch is labeled following the scheme of Fig. 4. It is the result of a pipeline that gives different weights to the labels of the pixels of the ground truth. It is aimed at giving more emphasis to pixels of the bunch, wood and pole classes, whose size within the patches may be much smaller than the one of leaves and background objects. In turn, the classification scores of these classes are high enough to lead to the correct segmentation, even if the corresponding patches have only a few pixels of those specific segments. On the other hand, this overestimation of the segment extensions can cause the merging of close segments of bunch, wood, or poles, into a single spread cluster. This is the case of the two grape bunches enclosed in the yellow circle of Fig. 9b. None of the networks can discriminate between the two bunches since the overestimation results from the segmentation method and not from the way the networks classify the sub-patches.

The comparison of results in Fig. 9 also shows that the VGG16 (Fig. 9e) is not able to segment the small, dark bunch within the orange circle in Fig. 9b. On the contrary, all the other classes cut out the bunch with good accuracy.

Another example of the application of the proposed processing for the segmentation of a test image is in Fig. 10. Each subplot is in correspondence to those of Fig. 9. This case is significant for two reasons:

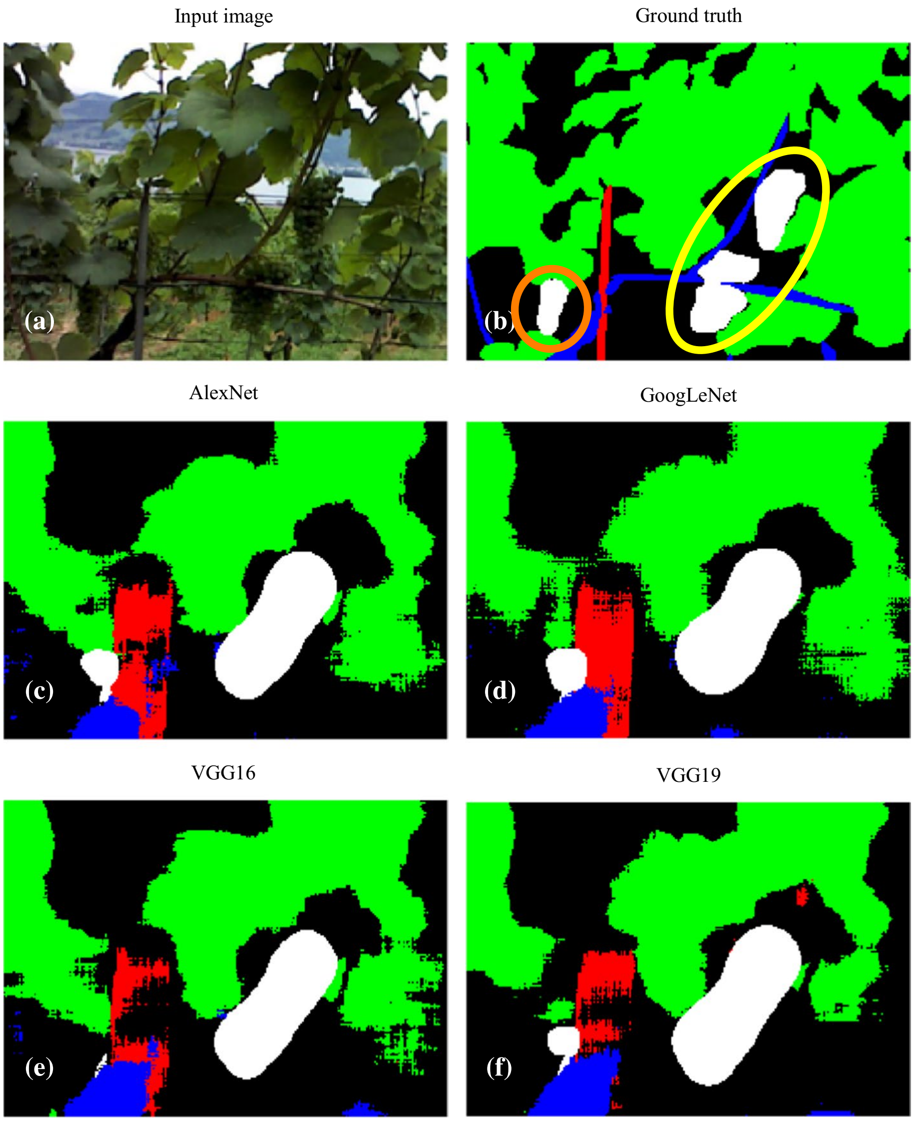

Fig. 9 Result of segmentation: **a** Input image and **b** corresponding ground truth; **c**–**f** result from AlexNet, GoogLeNet, VGG16, and VGG19, respectively. Here the VGG16 transferred net is not able to segment a small grape bunch (orange circle). All the networks merge two clusters in the yellow ellipse of **b** a single segment (Color figure online)

- All the four deep neural networks label the area of the orange circle in Fig. 10b with the leaf class, even if the ground truth presents the background. The inspection of the input image in the same region shows an ambiguity. It is full of leaves, which, however, are oriented towards the next row of grapevines. This causes the labeling of the ground

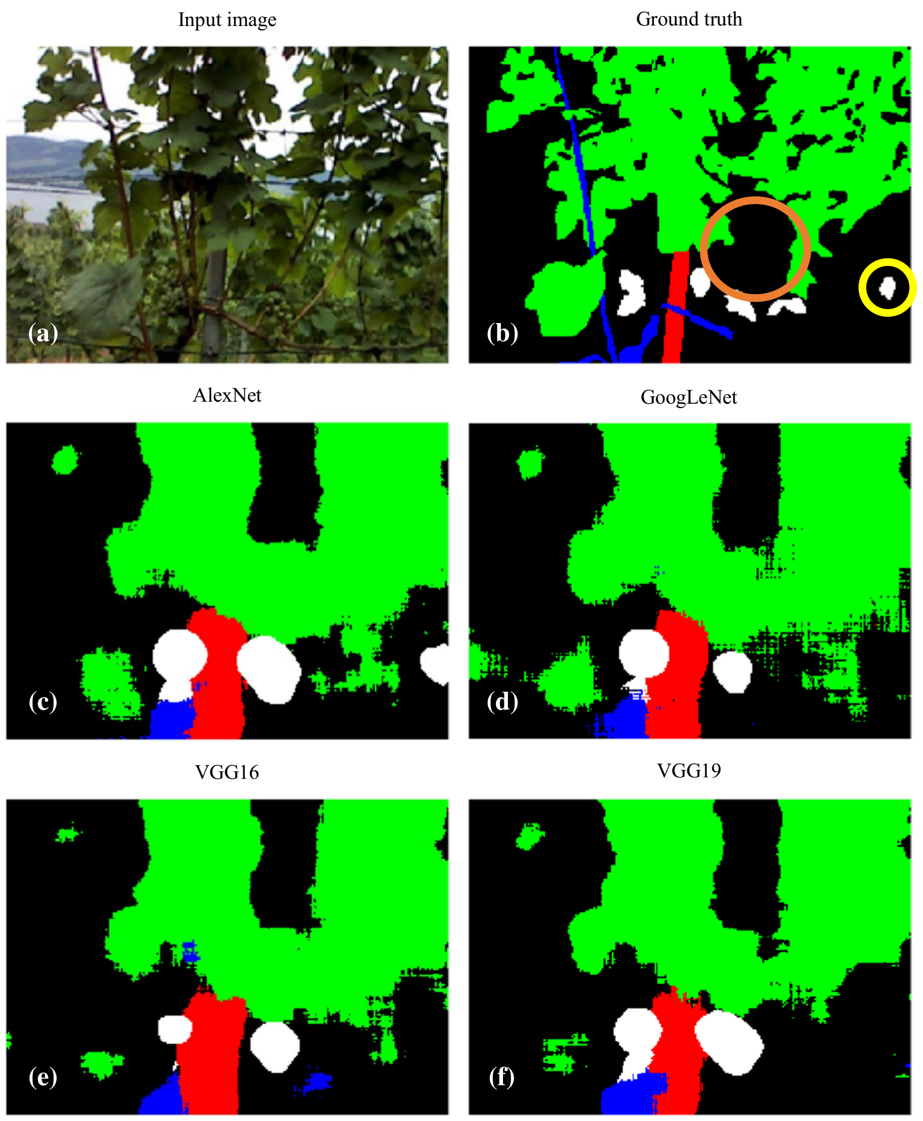

Fig. 10 Result of segmentation: **a** Input image and **b** corresponding ground truth; **c**–**f** result from AlexNet, GoogLeNet, VGG16, and VGG19, respectively. All the nets recognize the leaf class within the orange circle in (**b**). Moreover, the AlexNet is the only net able to segment the grape bunch on the right (see the yellow circle in (**b**)) (Color figure online)

truth, which is in contrast with the results of the networks, although both classifications are eligible. In turn, it will down the test performance in terms of MSA_{leaves}.

- The segmentation of the grape bunch in the yellow circle of Fig. 10b is challenging, because of its dimensions in the image and its dark appearance. Nevertheless, the

Table 4 Mean segmentation accuracy from the application of the proposed nets to label the test images

Net	Mean segmentation accuracy	
	MSA_{TOT} (%)	MSA_{bunch} (%)
AlexNet	69.52	74.70
GoogLeNet	65.49	74.41
VGG16	69.77	73.73
VGG19	68.63	80.58

Values are computed for all the classes (total) and detailed for the bunch one

Table 5 Comparison of IoU_{bunch} values resulting from the classification of the test images with a standard classification layer and with the Th_{opt} values of Table 2

Net	mean(IoU_{bunch})		std(IoU_{bunch})	
	Std class. layer (%)	Class. with Th_{opt} (%)	Std class. Layer (%)	Class. with Th_{opt} (%)
AlexNet	39.91	45.82	16.85	7.67
GoogLeNet	35.03	37.13	19.18	10.46
VGG16	42.75	45.44	15.94	9.59
VGG19	37.60	45.64	13.22	10.48

The comparison is proposed in terms of mean (mean(IoU_{bunch})) and standard deviation (std(IoU_{bunch})) values computed for the images of the test set

AlexNet correctly finds and segments its contours, whereas the other networks are not able to classify it. This outcome is interesting since the AlexNet has the easiest architecture, made of the lowest number of layers, and performs classification in the smallest time.

Quantitative analyses are finally proposed to evaluate the performance of the four deep neural networks. Specifically, the total mean segmentation accuracy (MSA_{TOT}) and the mean segmentation accuracy of the bunch class (MSA_{bunch}) are in Table 4.

The results in Table 4 are similar to the ones in Table 3, which are obtained for the analysis of training images. It means that overfitting is avoided since the four networks reliably predict the labels of the training patches, never seen before, with the same accuracy of the training, although the images of the two datasets differ in their appearance. Furthermore, all networks perform with comparable MSA_{TOT}, whereas the VGG19 produces the best performance in MSA_{bunch}, with a consistent gap of about 6%.

The performance of the networks can be evaluated in terms of the IoU_{bunch} metrics, which have computed for each image of the test set under two conditions: with the standard classification layer, whose prediction is based on cross-entropy loss optimization, and with the optimal threshold on the bunch probability.

As a first result, Table 5 shows that the application of the threshold operation on the bunch probability better performs than the standard classification layer, even if the threshold level is optimized on the training images. This is clear from the inspection of both mean(IoU_{bunch}) and std(IoU_{bunch}) which are always better when the optimal threshold strategy is selected. Then, the insight of Table 5 demonstrates that the AlexNet, the VGG16, and VGG19 equivalently perform better than the GoogLeNet, which ends with the worst results in both mean(IoU_{bunch}) and std(IoU_{bunch}) values. This is also expected from the analysis of the optimal threshold for the GoogLeNet in Table 2, which highlighted that the classification score of the grape bunch class is low at the boundaries of the grape bunches.

In summary, the best segmentation results on the test image are achieved by the VGG19 deep neural network. However, also the AlexNet shows a comparable performance in terms of the mean of IoU_{bunch} values and even better results in the standard deviations of the IoU_{bunch} values. Although the MSA_{bunch} of the AlexNet is lower than the one of the VGG19, it can be preferable for the task of bunch segmentation, since it requires much less computational time, more than 14 times shorter than the VGG19. These outcomes are valid for the analysis of the proposed images, targeted to the recognition of bunches of white grapes. In the case of different varieties of grapes, the training of the network has to be tuned to the new task, and thus the comparative analysis may produce different results. Nevertheless, since the problem of detection of white grapes is the most challenging, the good results of this study suggest the applicability of the deep neural networks to segment the other kinds of grapes, even easier to be detected as well-contrasted compared to the leaves and to the background vegetation.

Conclusions and future perspectives

This paper has presented a methodology for image analysis targeted at the detection and exact segmentation of grape bunches in RGB images. The cutting-edge approach of deep learning has served to the processing of low-quality images with VGA resolution (640×480 pixels), acquired by consumer-grade hardware mounted on-board an agricultural vehicle which moves in real environments. Four pre-trained deep neural networks, namely the AlexNet, the GoogLeNet, the VGG16, and the VGG19, have been tuned to classify input square patches of small size into five classes of interest, including the bunch class. Classification results are the input for the exact image segmentation. These results have been rearranged in probability maps, further processed during the training by an IoU optimization module. It selects the optimal threshold levels that can lead to the best IoU of the bunch class at the training stage. Then, these optimal thresholds have been used in the test of independent images to produce the final segmentation of grape bunches. Experiments have been run on 84 input images (60 for training and 24 for testing), windowed into 2104872 squared patches (1503480 and 601392 examples for the training and test datasets, respectively). Results have shown the capability of the methodology, based on deep learning pre-trained networks and the comparative analysis of classification scores, to fulfill the goal of segmentation of targets, under the hypotheses of this research. Specifically, outcomes have shown that the optimal threshold increases the mean segmentation accuracy from 2.10% (GoogLeNet) to 8.04% (VGG19). Besides, the comparison of the four deep neural networks shows that the VGG19 better performs than the other networks in terms of MSA_{bunch} (80.58%), with IoU_{bunch} of 45.64%. On the contrary, the GoogLeNet has returned the worst IoU_{bunch} results since the probability values of the bunch class are too small at the boundaries of the grape bunch. At the same time, the AlexNet has shown comparable IoU_{bunch} but lower MSA_{bunch} (74.70%) than the VGG19. Nevertheless, it can be preferable since its easier architecture, made of 25 layers, leads to less computational time. In this way, grape clusters can be exactly located in an automatic and non-invasive way, thus overcoming time and cost problems related to traditional manual measurements.

Future investigations will address the use of the depth data produced by the RGB-D sensor. This aspect will be explored for three purposes. As a first step, 3D data can directly enable the complete assessment of the geometrical properties of the segmented grape bunches, thus giving quantitative analytics of the harvest yield. Furthermore, depth data will be used to guide the selection of the size *N* of the moving window of the proposed processing. As known, the selection of this parameter depends on the vehicle-grapevine distance, which is implicitly measured by the sensor. Additionally, the depth map can enrich the input RGB image and can be another input to the proposed deep architectures. The transfer learning principles are still applicable, provided the modification of the input layers (blue boxes of Fig. 5) and the first 2D convolutional layers (the very first orange boxes of each chain in Fig. 5), in order to manage the new depth channel. The kernels of the 2D convolutional filters operating on the color channels are already initialized, as they are part of the initial pre-trained networks. On the contrary, the convolutional kernels working on the depth channel and the last fully-connected layers of the four networks will be trained from scratch with fast learning rates. In this way, the greater amount of input data will presumably result in an even better segmentation of the grape bunches.

Acknowledgements This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement ATLAS, No. 857125. The authors are also grateful to M. Attolico, G. Bono, S. Rilling, P. Frölich and M. Nielsen for the support in performing the experiments and gathering the data. The sole responsibility for the content of this publication lies with the authors. It does not necessarily represent the opinion of the European Union. Neither the EASME nor the European Commission is responsible for any use that may be made of the information contained therein.

References

Bargoti, S., & Underwood, J. (2017). Deep fruit detection in orchards. In F. Arai, F. Arrichiello, D. Burschka, J. Castellanos, K. Hauser, V. Isler, et al. (Eds.), *Proceedings of the IEEE international conference on robotics and automation (ICRA)* (pp. 3626–3633). Singapore: IEEE.

Bietresato, M., Carabin, G., Vidoni, R., Gasparetto, A., & Mazzetto, F. (2016). Evaluation of a LiDAR-based 3D-stereoscopic vision system for crop-monitoring applications. *Computers and Electronics in Agriculture, 124*, 1–13.

Boulent, J., Beaulieu, M., St-Charles, P.-L., Théau, J., & Foucher, S. (2019). Deep learning for in-field image-based grapevine downy mildew identification. In J. V. Stafford (Ed.), *In precision agriculture '19* (pp. 141–148). Wageningen: Wageningen Academic Publishers.

Chéné, Y., Rousseau, D., Lucidarme, P., Bertheloot, J., Caffier, V., Morel, P., et al. (2012). On the use of depth camera for 3D phenotyping of entire plants. *Computers and Electronics in Agriculture, 82*, 122–127.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., et al. (2016) The cityscapes dataset for semantic urban scene understanding. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3213–3223.

Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. *Computers and Electronics in Agriculture, 147*, 70–90.

Koirala, A., Walsh, K. B., Wang, Z., & McCarthy, C. (2019). Deep learning for real-time fruit detection and orchard fruit load estimation: Benchmarking of ‘MangoYOLO’. *Precision Agriculture, 20*(6), 1107–1135.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *In Proceedings of the 25th International Conference on Neural Information Processing Systems, 1*, 1097–1105.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444.

Librealsense online documentation: https://wiki.ros.org/librealsense. Retrieved 15 May 2020.

Lin, G., Tang, Y., Zou, X., Xiong, J., & Fang, Y. (2020). Color-, depth-, and shape-based 3D fruit detection. *Precision Agriculture, 21*(1), 1–17.

Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., et al. (2017). A survey on deep learning in medical image analysis. *Medical Image Analysis, 42*, 60–88.

Liu, S., & Whitty, M. (2015). Automatic grape bunch detection in vineyards with an SVM classifier. *Journal of Applied Logic, 13*(4), 643–653.

Luo, L., Tang, Y., Zou, X., Ye, M., Feng, W., & Li, G. (2016). Vision-based extraction of spatial information in grape clusters for harvesting robots. *Biosystems Engineering, 151*, 90–104.

Luo, L., Tang, Y., Lu, Q., Chen, X., Zhang, P., & Zou, X. (2018). A vision methodology for harvesting robot to detect cutting points on peduncles of double overlapping grape clusters in a vineyard. *Computers in Industry, 99*, 130–139.

Malavazi, F. B. P., Guyonneau, R., Fasquel, J.-B., Lagrange, S., & Mercier, F. (2018). LiDAR-only based navigation algorithm for an autonomous agricultural robot. *Computers and Electronics in Agriculture, 154*, 71–79.

Marani, R., Milella, A., Petitti, A., & Reina, G. (2019). Deep learning-based image segmentation for grape bunch detection. In J. V. Stafford (Ed.), *In precision agriculture ’19* (pp. 791–797). Wageningen: Wageningen Academic Publishers.

Maturana, D., Scherer, S. (2015) Voxnet: A 3d convolutional neural network for real-time object recognition. *In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 922–928.

Milella, A., Marani, R., Petitti, A., & Reina, G. (2019). In-field high throughput grapevine phenotyping with a consumer-grade depth camera. *Computers and Electronics in Agriculture, 156*, 293–306.

Murillo-Bracamontes, E. A., Martinez-Rosas, M. E., Miranda-Velasco, M. M., Martinez-Reyes, H. L., Martinez-Sandoval, J. R., & Cervantes-de-Avila, H. (2012). Implementation of hough transform for fruit image segmentation. *Procedia Engineering, 35*, 230–239.

Narvaez, F. Y., Reina, G., Torres-Torriti, M., Kantor, G., & Cheein, F. A. (2017). A survey of ranging and imaging techniques for precision agriculture phenotyping. *IEEE/ASME Transactions on Mechatronics, 22*, 2428–2439.

Paulus, S., Behmann, J., Mahlein, A.-K., Plümer, L., & Kuhlmann, H. (2014). Low-cost 3D systems: Suitable tools for plant phenotyping. *Sensors, 14*, 3001–3018.

Pereira, C. S., Morais, R., & Reis, M. J. C. S. (2019). Deep learning techniques for grape plant species identification in natural images. *Sensors, 19*(22), 4850.

Pérez, D. S., Bromberg, F., & Diaz, C. A. (2017). Image classification for detection of winter grapevine buds in natural conditions using scale-invariant features transform, bag of features and support vector machines. *Computers and Electronics in Agriculture, 135*, 81–95.

Pérez-Zavala, R., Torres-Torriti, M., Cheein, F. A., & Troni, G. (2018). A pattern recognition strategy for visual grape bunch detection in vineyards. *Computers and Electronics in Agriculture, 151*, 136–149.

Rahnemoonfar, M., & Sheppard, C. (2017). Deep count: Fruit counting based on deep simulated learning. *Sensors, 17*(4), 905.

Reina, G., Milella, A., Rouvere, R., Nielsen, M., Worst, R., & Blas, M. R. (2016). Ambient awareness for agricultural robotic vehicles. *Biosystems Engineering, 146*, 114–132.

Reis, M. J. C. S., Morais, R., Peres, E., Pereira, C., Contente, O., Soares, S., et al. (2012). Automatic detection of bunches of grapes in natural environment from color images. *Journal of Applied Logic, 10*(4), 285–290.

Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S. (2019). Generalized intersection over union: A metric and a loss for bounding box regression. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 658–666.

Rist, F., Herzog, K., Mack, J., Richter, R., Steinhage, V., & Töpfer, R. (2018). High-precision phenotyping of grape bunch architecture using fast 3D sensor and automation. *Sensors, 18*(3), 763.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*(3), 211–252.

Sa, I., Ge, Z., Dayoub, F., Upcroft, B., Perez, T., & McCool, C. (2016). Deep-fruits: A fruit detection system using deep neural networks. *Sensors, 16*, 1222.

Shapiro, L. G., & Stockman, G. C. (2001). *Computer vision*. Upper Saddle River, NJ, USA: Prentice Hall.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., et al. (2017). Mastering the game of go without human knowledge. *Nature, 550*(7676), 354.

Silver, D. L., & Monga, T. (2019). In vino veritas: Estimating vineyard grape yield from images using deep learning. In M. J. Meurs & F. Rudzicz (Eds.), *Advances in artificial intelligence lecture notes in computer science, 11489, Proceedings of the Canadian conference on artificial intelligence* (pp. 212–224). Cham: Springer.

Simonyan, K., Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. Retrieved Dec 4, 2019 from https://arxiv.org/pdf/1409.1556.pdf.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2015).Going deeper with convolutions. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–9.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *27th International conference on neural information processing systems* (Vol. 2, pp. 3320–3328). Cambridge, MA, USA: MIT Press.

Zhang, L., Zhang, L., & Du, B. (2016). Deep learning for remote sensing data: A technical tutorial on the state of the art. *IEEE Geoscience and Remote Sensing Magazine, 4*(2), 22–40.