Aid system for estimating agricultural yield using a deep learning technique: Tomato case

Abstract : The precision of traditional methods for estimating crop yield is a major challenge, particularly for large areas. To improve this process, we developed a tomato detection and localization system using deep learning techniques. The system uses Faster-RCNN, a cutting edge technology of object detection model, to detect and localize tomatoes in images. We trained the model on a database of 150 images, which were normalized to 100*100 pixels in RGB. The system estimates the real sizes of tomatoes using the Ground Sampling Distance method and predicts their masses using a regression model. The model produces an average absolute error of 42.365% and a quadratic error of 51.044%. Our system provides a more efficient and accurate way to estimate tomato crop yields on a large scale.

Keywords : Tomato, Object detection, Convolutional Neural Networks, Deep learning, Drone, Agricultural yield, Precision agriculture, Ground Sampling Distance, Regression model, Faster-RCNN.

1. Introduction

Precision yield predictions help farmers to improve the quality of their crops. In addition, it makes better decisions and, in consequence, reduces the cost of the operation regarding the intensity of the harvesting and the required labor. In general, estimation of crop yield is done using past data or workers manually count fruits at selected sampling locations in the field [a]. This method needs a lot of time and labor and sometimes, the chosen samples are not a good representation of the population, especially over a very large plot. A solution is to use a drone fitted with computer vision systems. It does not need to be remunerated and can tick off fruits in hundreds of frames per second and can assess a whole large area.

2. State of the art

Remote sensing techniques have been utilized to estimate crop yield, as demonstrated by Johnson et al. in their work. They use MODIS NDVI products and land surface temperature

products, along with rainfall data, to estimate the yield of corn and soybeans in 12 US states representing 75% of US corn and soybean production. The data is collected 32 times for 8-day periods from February to the end of October from 2006 to 2012. Johnson creates two datasets, one for corn and one for soybeans, and masks the pixels that do not belong to the fields. The pixel values are then averaged for all observations per count and per year. Pearson's product-moment correlation coefficient is used to understand the relationship between crop yield and remote sensing products. Results show a strong relationship between NDVI values and soybean yield during the summer months, and an inverse relationship in the spring, while rainfall showed no relationship to soybean yield. Johnson uses Decision Trees (DT) to predict feed yield and temperature data based on his findings and suggests that the method should be further investigated to check its applicability to other crops or regions [b]. Other methods are based on computer vision. There are two general categories of computer vision techniques used to estimate crop yields, namely area-based methods and count-based methods. While there is considerable literature on regional methods, count-based methods have not received as much attention. Wang et al. [c] developed an automatic stereoscopic camera in an apple orchard yield estimation system. To mitigate the impact of unpredictable natural lighting during the day, they acquired images at nighttime. Li et al. [c] have developed a system for detecting cotton in a field based on image segmentation data based on semantic regions. Lu et al. [c] developed color modeling by region for maize crops. In the literature, count-based techniques for estimating crop yield have received relatively little attention. Color images were utilized to estimate the count of apples obtained from orchards under natural light, resulting in an accuracy rate of over 85% [d]. However, this approach has drawbacks such as direct lighting and color saturation, which can lead to a significant number of false positives. Park et al. (2006) [e] developed a method to segment apple fruit from video using background modeling.

Computer vision algorithms face various challenges for fruit counting for yield estimation: variance of illumination during image capture, occlusion by foliage and tree branches or crop, degree of overlap between fruits to count, counting tomatoes that are in shadow, scale difference when capturing image. Counting objects accurately in various applications is challenging due to various factors, such as occlusion, illumination changes, and variability in shape and size. The task becomes even more complex in scenarios involving tomatoes or other fruits. This challenge is common in real-world applications, including counting cells in microscopic images, estimating the flora and fauna population in forest aerial images, monitoring crowds in shaded areas, and more. [c]. The method proposed by Kim et al. (2017) [c] detects and tracks moving people using a single fixed camera. Later, Lempitsky et al. (2017) [c] proposed a new supervised learning framework for counting visual objects. Convolutional neural networks have been utilized to extract features for various image analysis tasks, including but not limited to object recognition and semantic segmentation. These characteristics can also be regressed to count. This method makes it possible to explicitly estimate the number of tomatoes with a single glance at the entire image. In this way, it reduces

the overhead of detecting and locating objects. The main advantage of this work is that thousands of real images of tomatoes are not needed for training. The network has been trained in the use of synthetic images and tested on real images and works effectively.

In this article we introduce a new tomato detection model on an image coupled with a mass estimation model.

The rest of the document is organized as follows. In section 2, we present the materials and methods based on the notions of object detection using neural network techniques as well as the techniques for obtaining the sizes of an object on an image. In section 3 we presented the hardware and methods used in the implementation of our system. Finally in section 4 we present the results obtained and the discussions then we will conclude in section 5.

3. Conceptual clarification

In this section, we will successively present: agricultural yield in part II.1, object detection in part II.2, deep learning in part II.3, convolutional neural networks in part II.4, artificial neural networks in part II.5, convolutional neural networks in part II.6, transfer learning in part II.6, detection model architecture of objects in part II.7 and finally the performance metrics in part II.8.

3.1. Agricultural yield

Yield can be interpreted in various ways depending on the field in which it is used. Yield qualifies the proportion between the result obtained and the means used for this purpose. Thus, the term performance is expressed in a concrete and general way in the form of a ratio between the result obtained and the means provided for its effectiveness. In practice, this definition will be declined according to different formulations to correspond as closely and faithfully as possible to the real parameters of each activity. In agriculture, two aspects of yield can be identified: financial yield, which represents the producer's income after the difference between the selling price and the cost of production. This return is useful for managing producers' finances. On the other hand, the agricultural yield represents the quantity of product harvested on a given cultivated area. Also referred to as average yield. It is often expressed in tonnes per hectare for water-rich products (roots and tubers, fruits, etc.). It also refers to the number of plants per unit area (for leafy crops). It is this definition of performance that interests us. Depending on the culture being studied, one or the other is preferred. Yield is therefore a generalized concept, but its use depends on the interest one has in the information sought. When we take the example of a crop such as tomato, what interests the grower is the mass obtained on a given surface. Indeed, the price of the tomato is a function of the mass of the latter. On the other hand, for a leaf crop (such as lettuce for example) the number of plants is useful information, because the mass does not influence the price (which is a function of other parameters most often defined by the course of the market). The yield that interests us here is that of the proposal expressed in mass in relation to the area.

Yield estimation using a sample plot is done by marking a small part of a field, harvesting it separately from the rest, and calculating what the harvest would have been on one hectare. If the area of the field has also been measured, then the total production of the field can be calculated. The investigator can stake the sample plots at any time during the agricultural season, provided that he has finished before the start of the harvest. However, when the plants haven't yet reached the desired height, it is easier to plot the different plots, and the person in charge of the survey is less sensitive to the behavior of the crop when choosing the location of the plot. Sample plots must be clearly identified and their boundaries clearly marked. The investigator should regularly check that the stakes are still in place. The choice of the number and size of the sampling squares in a plot is decisive for the credibility of the yield estimate. The reliability of the estimate is all the greater as the plots are large. When deciding on the number of sample plots, account should be taken of the investigator's workload and the diversity of the crop. If several sample plots are set up, their position can be determined in two ways, either arbitrarily (which is the easiest), or systematically, so that all the plots are evenly distributed on the ground. In reality, the surveyor can rarely place more than one or two sample plots in a field. To lay the yield squares there are three steps to follow:

— locate the plot of the crop to be assessed;

— determine the longest diagonal of the field. This can be done visually;

— identify the diagonals of the plot and the laying points.

After this, it is necessary to lay 5 squares per diagonal avoiding borders for plots of one (1) hectare. For plots of less than one hectare, 2 to 3 squares must be laid per diagonal with a minimum of 4 squares per plot. For each square:

— count the number of plants or feet per square of yield laid;

— harvest the products (fruits or tubers), count them and place them in a container (bag or sachet, bucket, etc.);

— determine the weight of harvested products;

— record the data collected on the yield square sheet. For staggered harvests, add up, on the same square, the data collected from each harvest.

Qualitative approaches used by some countries are based on field information from specialists who, based on the agro-meteorological survey and taking into account the course of the campaign and the physiognomy of the crops and referring to the production and yields of previous years , issue an opinion that they are encrypted. They are also useful in determining losses caused by natural disasters. Other means relate to the prediction of yields and involve modeling using agro-meteorological, agronomic and remote sensing variables and indicators. The indicators generally used by these models relate to water balance, vegetation index, precipitation, temperature, agricultural inputs, etc. The indicators used in these models are water balance, vegetation index, precipitation, temperature, agricultural inputs, etc.

3.2. Object detection

In computer vision, to fully understand an image, one must not only focus on classifying the different images, but also attempt to accurately estimate the concepts and locations of the objects contained in the image. This task is called object detection.

Object detection is a computational technique related to computer vision and image processing that deals with the detection of instances of semantic objects of a certain class (such as human beings, buildings or cars) in digital images and videos. Object detection is able to provide valuable information for the semantic understanding of images and videos, and is related to many applications, such as image classification, human behavior analysis, facial recognition, autonomous driving etc.

The definition of the object detection problem is to determine where objects are located in a given image (object localization) and to which category each object belongs (object classification). Generally, the pipeline of traditional object detection models can be divided into three stages:

1. Informative region selection: As different objects may appear in all positions of the image and have different aspect ratios or sizes, it is natural to scan the entire image using a multi-scale sliding window.

2. Feature extraction: To recognize different objects, we need to extract visual features that can provide robust semantics and representation. SIFT [i], HOG [j] and Haar-like [k] are some of these representative features. This is because these features can produce representations associated with complex cells in the human brain [i]. However, due to the variety of appearances, lighting conditions and backgrounds, it is difficult to manually design a robust feature descriptor that can perfectly describe all kinds of objects.

3. Classification: Additionally, a classifier is needed to distinguish a target object from all other categories and to make the representations more hierarchical, semantic, and informative for visual recognition. Usually, Supported Vector Machine (SVM), AdaBoost and Deformable Parts Model (DPM) are good choices [l].

Thanks to the emergence of the Deep Neural Network (DNN), a more significant gain is obtained with the introduction of the Regions endowed with the characteristics. DNNs have the ability to learn more complex features than shallow networks. Moreover, the robust learning algorithms allow us to learn the informative representations of objects without the need to characterize them manually. Since the proposal of the R-CNN, a lot of progress in improving models have been suggested such as the Fast R-CNN which jointly optimizes the bounding box classification and regression tasks, Faster R-CNN which takes a sub -additional network to generate region proposals and YOLO which performs object detection via fixed-grid regression. All provide different degrees of improvement in detection performance over primary R-CNN and make real-time and accurate detection of smaller objects. We will present in the following the concepts that support the basis of object detection.

3.3. Deep learning

Deep Learning is a special type of Machine Learning that achieves tremendous power and flexibility by learning to represent the world as a nested hierarchy of concepts or abstractions. Deep Learning is inspired by the functionalities of our brain cells, most often with the models

of artificial neural networks. This model takes data connections between all the artificial neurons and adjusts them according to the data model. If the data size is large, the greater the number of neurons needed. It automatically incorporates learning at multiple levels of abstraction, thus allowing a system to learn the mapping of complex functions without depending on any specific algorithm. Deep learning techniques are a class of machine learning algorithms that:

- use various layers of non-linear processing units for the extraction and the transformation of features. Each layer uses the output of the preceding layer as input (exception for the first layer). The algorithms can be supervised or unsupervised, and their applications include shape matching and statistical clustering;
- Learning occurs at multiple levels of data detail or data representation; through the various layers, we move from low-level to higher-level metrics, corresponding to different levels of data abstraction.

3.4. Artificial neural networks

Neural networks are a family of Machine Learning models inspired by neuroscience research. They attempt to do the same thing as any other models, but with higher performance. Neural networks are multi-layer networks that are used to make prediction, to classify objects, etc.
Image 2 shows a simple neural network with five entries, five exits and two concealed layers of neurons. From right to left we have:

- the model's input layer in orange.
- the first hidden layer of neurons in blue.
- the second hidden layer of neurons in brown.
- the output layer (prediction layer) of the model in green.

The connecting arrows show the way all neurons are interlinked and the way the data flows from the entry layer to the exit layer. As can be seen, the fundamental unit of an artificial neural network is a node (neuron). These are mathematical models inspired by biology. The idea from the start was to model the functioning of a biological neuron.

3.5. Convolutional neural networks (CNNs)

Nowadays, the convolutional neural network (CNN) is the most representative model of deep learning. It is inspired by the functioning of the human visual cortex system. Convolutive neural networks have a similar approach to traditional supervised learning methods: they receive input images, detect the features (characteristics) of each of them, and then train a classifier on them. The features are learnt systematically. CNNs carry out the time-consuming task of extracting and describing the features themselves: during the training phase, the classification error is minimized in order to optimize the classifier settings and the features. Furthermore, the specific network architecture enables the retrieval of different complexity features, from the most simple to the most advanced ones. However, unlike supervised learning techniques, convolutional neural networks learn the features of each image.

CNNs represent a sub-category of neural networks: they therefore have all the characteristics. However, CNNs are particularly suitable for image processing. Their configuration is then more specific: it is composed of two main blocks: the feature extractor and the classifier. [n]
As with ordinary neural networks, the layer parameters are determined by gradient backpropagation. In CNN, these parameters designate in particular the characteristics of the images. CNNs are generally composed of 4 layers (which can be repeated):

- The Convolutional Layer (CONV) : It is an important component of convolutional neural networks and is usually the first layer. Its main function is to identify a set of features in the input images. This is accomplished through convolution filtration, where a window representing the feature is dragged over the image, and the convoluted product is calculated from the feature to each part of the scanned image. The filters are acting as characteristics that we want to detect in the images, and the convolution layer gets several images as input and computes the convolution of each with each filter. For each pair (image, filter), an activation map or feature map is produced, which shows where the features are in the image. The larger the value of the map, the more similar the corresponding location in the picture is to the characteristic.
- the pooling layer (POOL): This type of layer is often placed between two convolution layers: it receives several feature maps as input, and applies the pooling operation to each of them.
- The pooling operation is employed to shrink the size of the images while retaining their significant properties. The process involves dividing the image into uniform cells and preserving the maximum value in each cell. In practical applications, small square cells are used to minimize information loss. Common options include non-overlapping adjacent cells measuring 2x2 pixels or cells of size 3x3 pixels with a 2-pixel separation. The output contains the same number of feature maps as the input, but in a much smaller size. This layer helps reduce the number of parameters and computations in the network, thus enhancing efficiency and preventing overfitting. The feature maps produced after pooling are less precise compared to the input, which is an advantage since precise feature localization is not always necessary in image recognition tasks. Therefore, the pooling layer makes the network less sensitive to the position or orientation of features, and slight variations should not result in significant changes in image classification.
- the ReLU correction layer: ReLU (Rectified Linear Units) designates the nonlinear real function defined by $\mathrm{ReLU}(x) = \max(0, x)$.

The ReLU correction layer therefore replaces all the negative values received as inputs with zeros. It acts as an activation function.

- The fully-connected layer (FC): The fully-connected layer always constitutes the last layer of a neural network, convolutional or not; it is therefore not characteristic of a CNN. This type of layer receives a vector as input and produces a new vector as output.

To do this, it applies a linear combination and then possibly an activation function to the values received as input. The last fully-connected layer is used to classify the input image of the network: it returns a vector of size N, where N is the number of classes in our image classification problem. Each element of the vector indicates the probability for the input image to belong to a class. For example, if the problem consists in distinguishing cats from dogs, the final vector will be of size 2: the first element (respectively, the second) gives the probability of belonging to the class "cat" (respectively "dog"). Thus, the vector [0.9, 0.1] means that the image has a 90% chance of representing a cat. Every entry array value "votes" in favor of a class. The votes do not all have the same weight: the layer gives them weights that are dependent on the array item and the cluster. To compute the likelihoods, the entirely linked layer multiplies each entry by a weight, calculates the sum, then applies an enabling function (logistic if N=2, softmax if N>2) : This process is equivalent to multiply the entry vector by the matrix that contains the weights. The fact that each entry value is linked to all the exit values explains the term "entirely linked".

3.6. Transfer learning

A convolutional neural network is very expensive to train: the more layers are stacked, the larger the number of convolutions and parameters to be optimized. The machine must be able to store many gigabytes of information and carry out the computations in an efficient way. This is the reason why hardware makers are increasing their efforts to supply powerful graphics processing units (GPUs) capable of quickly train a deep neural network by parallelizing the computations.

Transfer learning is a technique of machine learning in which a trained model for one job is re-used for another associated job. Transfer learning is linked to such issues as multi-tasking and concept drift and is not solely an area of deep learning research. Transfer learning has several benefits in addition to accelerating the training of networks. One of these benefits is the capability to avoid overfitting. When operating with a small collection of input images, it is not advisable to train a neural network from scratch, as the number of parameters to be learnt is much higher than the number of images, leading to a high potential for overfitting. Based on the size and the similarity of the input dataset to the pretrained set, we can use the pretrained neural network in various ways.

- Strategy 1: Total fine-tuning

It involves replacing the last fully-connected layer of a pre-trained network with a randomly initialized classifier that is adapted to the new problem, such as SVM or logistic regression. All layers of the pre-trained network are then trained on the new images. This approach is suitable when the new collection of images is large enough, as it can avoid the risk of overfitting. In addition, since the parameters of all the layers (except the last) are initialized with those of the pre-trained network, the learning phase can be faster than starting from a random initialization.

- Strategy 2: feature extraction

In order to represent the images of a new problem, we can utilize the features of a pre-trained network. This involves removing the last fully-connected layer and setting all other parameters, allowing the truncated network to calculate the representation of each input image using previously learned features. Next, a classifier initialized randomly is trained on these representations to address the new problem. This approach is recommended for a small new image collection that is similar to the pre-training images, as there is a significant risk of overfitting when training the network on a small number of images. Additionally, if the new images are similar to the old ones, they can be represented by the same features.

- Strategy 3: partial fine-tuning

This approach combines strategies 1 and 2, where the last fully-connected layer is replaced with a new classifier that is randomly initialized, and certain layers of the pre-trained network are fixed. Therefore, in addition to training the new classifier, the unfixed layers are also trained on the new images, which typically correspond to the highest layers of the network.

When dealing with a small and significantly different collection of images than the pre-training images, we employ this strategy: replacing the last fully-connected layer with a new classifier that is randomly initialized, while also fixing the parameters of certain layers of the pre-trained network. This is because strategy 1, which involves training the whole network, is not feasible due to the high risk of overfitting, and strategy 2, which utilizes the pre-trained network's features to represent the new images, is not suitable because of the dissimilarities between the two image sets. Nonetheless, it's worth noting that the lower layers' features are generally simple and generic, whereas the higher layers' are complex and specific to the problem. Consequently, fixing the lower layers and training the classifier and upper layers is a reasonable compromise.

3.7. Object detection architectures models

There are currently several object detection models. Each model presents a different but sometimes complementary and useful architecture for various operations: better performance, better inference time, real-time use, etc.

3.8. Performance Metrics

A common evaluation metric used in many object recognition and detection tasks like Faster R-CNN, SSD, etc is the COCO standard "mAP", short for "mean average precision". from 0 to 100; the higher the value, the better.

3.8.1. Precision and recall

- ❖ Precision measures how accurate your predictions are, that is, the percentage of your predictions that are correct.
- ❖ Recall measures the quality of all positives. For example, we can find 80% of the possible positive cases in our main predictions.

Here are their mathematical definitions:
That is :
VP: true positive
VN: true negative
FP: false positive
FN: false negative

precision = VP/ (VP + FP)
recall = VP/ (VP + FN)

3.8.2. IoU (Intersection over union)

The IoU measures the overlap between two borders. We use it to measure the degree of overlap between the intended boundary and the actual object boundary. In some datasets, we pre-define a unit threshold (e.g. 0.5) to determine if the prediction is a true positive or a false positive.

4. Materials and methods

In this section we will present the R-CNN model in part III.1, then follow the Faster R-CNN model in part III.2, then the YOLO model in part III.3 and finally part III.4 will expose the method of determining the actual sizes of objects in an image.

4.1. R-CNN Models

R-CNN is short for "Region-Based Convolutional Neural Networks". The main idea is composed of two steps. First, using a Selective search algorithm, it identifies a manageable number of candidate regions (2000) to the enclosing object area ("region of interest" or "RoI"). And then it extracts the CNN features of each region independently for classification.
The operation of R-CNN can be summarized as follows:
— Pre-training a CNN network on image classification tasks (e.g. VGG or ResNet which have been trained on the ImageNet dataset). The classification task involves N classes.
— Proposal of regions of interest independent of the category by selective search (approximately 2,000 candidates per image). These regions can contain target objects and are of different sizes. Region candidates are warped to have a fixed size as required by CNNs.
— Continue to refine the CNN on the deformed proposition regions for K+1 classes. For each image region, forward propagation through the CNN generates a feature vector. This feature vector is then consumed by a binary SVM formed independently for each class. Positive samples are proposed regions with IoU overlap threshold (intersection on union) >= 0.3, and negative samples are irrelevant. To reduce localization errors, a regression model is trained to correct the predicted detection window on the bounding box correction offset using CNN functions.

4.2. Problems with R-CNN

It still takes a long time to train the network, because 2000 region proposals would have to be classified per image.

It cannot be implemented in real time because it takes around 47 seconds for each test image. The selective search algorithm is a fixed algorithm. Therefore, no learning takes place at this stage. This could lead to the generation of bad candidate region proposals.

4.3. Faster R-CNN model

The R-CNN algorithm (as well as its optimized version Fast R-CNN [l]) use selective search to find proposed regions. Selective searching is a slow and tedious process that affects network performance. Therefore, some researchers in [l] have developed an object detection algorithm which eliminates the selective search algorithm and allows the network to learn the region proposals. As with the Faster R-CNN, the image is provided as input to a convolutional network which provides a convolutional feature map. Instead of using a selective feature map search algorithm to identify region proposals, a separate network is used to predict region proposals. The proposed predicted regions are then reshaped using a resolution pooling layer which is then used to classify the image into the proposed region and predict offset values for the bounding boxes.

4.4. YOLO model (You Only Look Once)

Previously, object detection algorithms used regions to locate objects within an image. These algorithms examined specific portions of the image that were most likely to contain the object, rather than analyzing the entire image. In contrast, YOLO is a unique object detection algorithm that operates differently from these region-based methods. YOLO uses a single convolutional network to predict both the bounding boxes and the class probabilities for those boxes. The YOLO approach involves dividing an image into an S * S grid, with each grid containing m bounding boxes. The network produces a class probability and offset values for each bin, and bounding boxes with class probabilities exceeding a certain threshold are chosen to locate the object (of that class) within the image. YOLO is faster than other object detection algorithms, achieving 45 frames per second. However, the YOLO algorithm struggles to detect small objects in the image, such as a flock of birds, due to spatial limitations.

4.5. Methods for determining the actual sizes of objects in an image

To allow us to have the real dimensions of an object captured in image (by drone for example), we need to determine the GSD. The Ground Sampling Distance (GSD) refers to the distance between the centers of two adjacent pixels measured on the ground surface. A higher GSD value indicates a lower spatial resolution of the image, which means fewer details are visible. Generally, the GSD value is inversely proportional to the flight height, meaning that as the flight altitude increases, the GSD value also increases. For the calculation of the GSD:

$$GSDx = (H * Sh)/(Fr * imgH)$$

$$GSDy = (H * Sw)/(Fr * imgW)$$

$$GSD = \max(GSD\ x\ ,\ GSD\ y\ )(cm/pixel))$$

So to calculate the width of the object for example:

$$GSD = W' = GSD * Wpixel$$

— H: the distance between the camera and the target (the ground)
— S h: the length of the sensor (camera);
— S w: the width of the sensor (camera);
— F r: the focal length (in mm);
— imgH: the length of the image (in pixels);
— imgW: the width of the image (in pixels);
— GSD x: GSD along the x axis;
— GSD y: GSD along the y axis;
— W pixel: the dimension (length or width) in image (in pixel);
— W': the actual dimension (length or width) (in cm);

4.6. Database design for detection

For the constitution of the database, we used a total set of 150 images distributed as follows:

- 25 images taken in a tomato field with a drone;
- 30 images from various online sources;
- 95 images from the official fruit-360 database [r].

We notice a certain heterogeneity of the images. The goal is to allow our model to learn the characteristics of our tomatoes under various hazards both in the natural environment and in controlled environments. All images have been normalized to a resolution of 100*100 pixels. Since we are doing supervised learning, we have labeled each object contained in each image ourselves. This was done with the labelImage tool. We obtained the annotations in XML file in PASCAL VOC format. Figure 10 shows the use of the LabelImage tool which allows to have the labels.

4.7. System Architecture

The architecture of the system revolves around:
1. Tomato detection on the image : we use the tomato detection model that we had previously trained to detect and locate the tomatoes contained in the image.
2. This image is then processed to extract each detected tomato in isolation.
3. Each of the tomatoes is then segmented to detect only the tomato in the image. Using a real size estimation technique from the image, we determined the air of the projected area of the captured tomato.
4. From this value of the surface of the tomato, we used a model of estimation of weight by the projected surface. This model is based on the studies of Amin Taheri-Garavand et al. [s].

The main mass estimation models from the projected surface of a tomato are given in linear and non-linear form by the following equations:

$y = 0.009 * PA + 64.80$ (1) with $R^2 = 0.94$

$y = 0.001*PA^2 + 0.006*PA + 70$ (2) with $R^2 = 0.94$

$y = 36.52\ln(PA) - 201.0$ (3) with $R^2 = 0.936$
with PA the projected area. PA is determined by
$PA = \pi * H * W$ (2)
H: the length of the tomato detection box
W: the width of the tomato detection box.
After several tests we opted for model (2) because it gave results closer to our test sample.

5. Results and discussion

We used, for the training of our detection model, 100 annotated images which contain on average 5 tomatoes. These images have been downloaded from the internet in various image banks, others taken on a field with a DJI Phantom 4 drone and the main part of the Fruit-360 database. In addition, we used 20 other images for verification. We therefore used the model trained on the 20 images of which it is not aware to verify its performance. MaP (Average Accuracy) is a commonly used metric to measure the accuracy of object detectors such as Faster R-CNN, SSD, etc. Average Precision calculates the average precision value for the recall value between 0 and 1 (or as a percentage).
For training, we used 10000 iterations saving the model every 500 iterations. We can observe on the learning graph of the model that the model has its high performance point from the 6500 iteration. At this level the mAp is close to 0.84 or 84% accuracy.
In figure 13, we can see that almost all of the tomatoes are identified. However, some deviations in the detection boxes are identified. This could introduce bias when extracting the actual dimensions of each tomato. However, the shifts observed are of the order of a millimeter. Similarly, we notice that partially covered tomatoes are partially identified. Their estimate will then be more biased. On the other hand, tomatoes that are covered by leaves are not identified. This is a limitation that must be taken into account when making a decision.
We used each tomato detection box from the tomato detection model for the estimation of the mass of each tomato. We can see some results in the table below:

N°	Masse réelle (g)	Masse estimée (g)	Taux de précision(%)
1	200	235,55	81,10
2	100	110,98	90.10
3	100	128,15	78,03
4	250	312,44	80,12
5	200	305,02	65,53

Table 1. Some mass estimates from the estimator

Various statistical indicators for estimating errors were also used to assess the relationship between estimated and actual fruit masses in grams (g). In particular, we used Mean Absolute Error (MAE), Root Mean Squared Error (MSE), Root Mean Squared Deviation (RMSE) for our results.

$$MAE=i=1ny1-yn \quad (3)$$

$$MSE=1n(y1-y)2 \quad (4)$$

$$RMSE =1ni=1ny1 - \hat{y}^2 \quad (5)$$

We thus obtain after experiment:
AEM = 42.365
MSE = 2605.534
RMSE = 51.0444
The figure 14 presents the distribution of the estimates. We notice that we obtain a certain linearity in our model.

Given the amount of variation in the training data samples and the error in the geometry modulus, we can consider these results acceptable. Similarly, we can say that the error of mass estimates is related to the quality of the detection box of each tomato by the detection model as well as the efficiency of the detection system of the real dimensions. In addition, the polynomial mass estimator used gives good results in general but was implemented with a very large quantity of tomato of various species. Therefore, the study of a particular species will not be as effective. The model seems too generic, which adds a certain discrepancy to the results of the tomato crop used during the experiments. Because, in fact, the density of a tomato is strongly linked to the species [o]. To overcome this problem, studies are underway to set up a mass estimation model based on certain morphological characteristics (extracted from a 2D image of the tomato) for a particular tomato species.

6. Conclusion and prospects

During our work, we first developed an object detection model to identify tomatoes on an image and then used the morphological characteristics of tomatoes identified earlier in a mass estimator by AI projected to estimate the mass of each tomato. The operation is carried out on each tomato in each field in order to have an approximation of the final yield. Detection and localization gave good results with a mAp of 84%. The trained model is also effective in detecting and locating multiple tomato fruit occurrences in complex environmental scenarios. However, fruits that contain leaves or other tomatoes will not be accurately detected. There will be some loss of precision in the detection boxes. The mass estimator used has a coefficient of determination R^2 of 0.94 for all kinds of tomatoes. However, the use in our system during the experiments gave a MAE of 42.365, 2605.534 for the MSE and 51.0444 for the RMSE. These results show that the mass estimator is too generic and that a model specific to a particular tomato species should be put in place. In addition to detecting and estimating the mass of individual fruits, this approach can be effective in identifying and estimating the mass of multiple types of fruit. Moreover, this method is appropriate for systems where data collection is calibrated using a single camera positioned perpendicular to the object surface. While this setup reduces costs, it also results in lower accuracy in estimating fruit masses.

Nonetheless, this proposed system shows promise as a foundation for creating computer vision-based technologies for automatic sorting, grading, and measuring.

References

[a] Q. Wang, S. Nuske, M. Bergerman, and S. Singh, "Automated crop yield estimation for apple orchards," in Experimental Robotics, 2013, pp. 745-758.
[b] Helena Russello. Convolutional neural networks for crop yield prediction using satellite images. June 2018.
[c] Maryam Rahnemoonfar and Clay Sheppard. Real-time yield estimation based on deep learning. page 1021809, 05 2017.
[d] O. Cohen R. Linker and A. Naor. Determination of the number of green apples in rgb images recorded in orchards. Computers and Electronics in Agriculture, 2012.
[e] D. L. Peterson A. L. Tabb and J. Park. Segmentation of apple fruit from video via background modeling. 2006 ASAE Annual Meeting, 2006.
[f] Richard J. Maude Stefan Jaeger Mahdieh Poostchi, Kamolrat Silamut and George Thoma. Acm mm. 2014.
[g] Z. Yang and R. Nevatia. A multi-scale cascade fully convolutional network face detector. ICPR, 2016.
[h] A. L. Kornhauser C. Chen, A. Seff and J. Xiao. Deepdriving : Learning affordance for direct perception in autonomous driving. ICCV, 2015.
[i] D. G. Lowe. Distinctive image features from scale-invariant keypoints. ICCV, 2004.
[j] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. CVPR, 2005.
[k] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection.ICIP, 2002.
[l] Zhong-Qiu Zhao, Member, IEEE, Peng Zheng, Shou-tao Xu, and Xindong Wu, Fellow "Object Detection with Deep Learning: A Review" in IEEE
[m] Yannis Chaouche. Identify differents types of automatic learning, 20/09/2019. https://openclassrooms.com/fr/courses/, last visit the 21st of september 2019.
[n] Kimia Nadjahi Pascal Monasse. Classify and segment visual data
, 23/05/2019. https://openclassrooms.com/fr/courses/, last visit the 26th of september 2019.
[o] Chang ShuCOMP. Introduction to Computer VisionDr. 4900CWinter, 2008.16
[p] https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

[q] Ground sampling distance (gsd), 15/06/2019. https://support.pix4d.com/hc/en-us/articles/202559809-Ground-sampling-distance-GSD, last visit the 29th of september 2019.
[r] Horea Muresan, Mihai Oltean "Fruit recognition from images using deep learning", Acta Univ. Sapientiae, Informatica, 10, 1 (2018) 26–42
[s] Amin Taheri-Garavand, Shahin Rafiee, and Alireza Keyhani. Study on some morphological and physical characteristics of tomato used in mass models to characterize best post harvesting options. Australian Journal of Crop Science, 5 :433–438, 04 2011.

[t] Lee, Jaesu; Nazki, Haseeb; Baek, Jeonghyun; Hong, Youngsin; Lee, Meonghun. 2020. "Tomato Mass Estimation in Precision Agriculture" Sustainability 12, no. 21: 9138. https://doi.org/10.3390/su12219138.

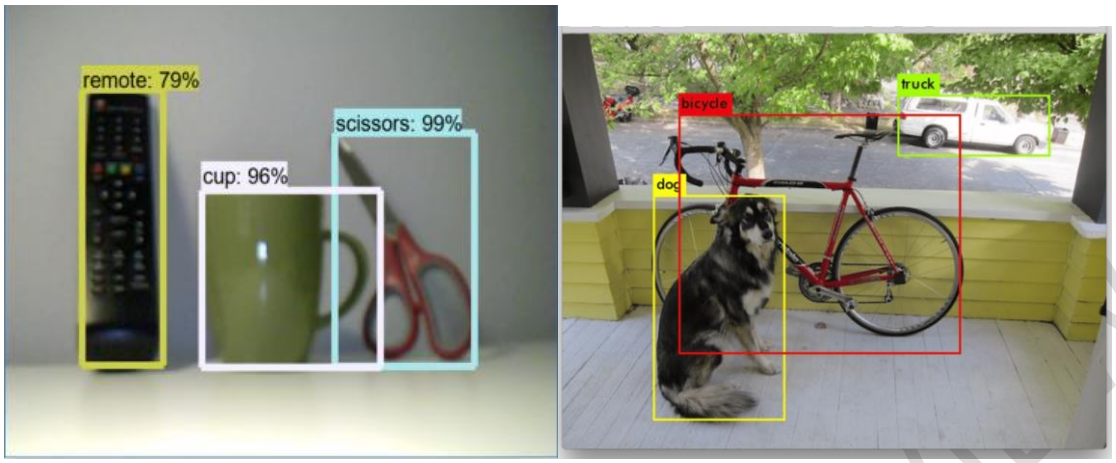

fig. 1. Illustrations of some objects detections [l]

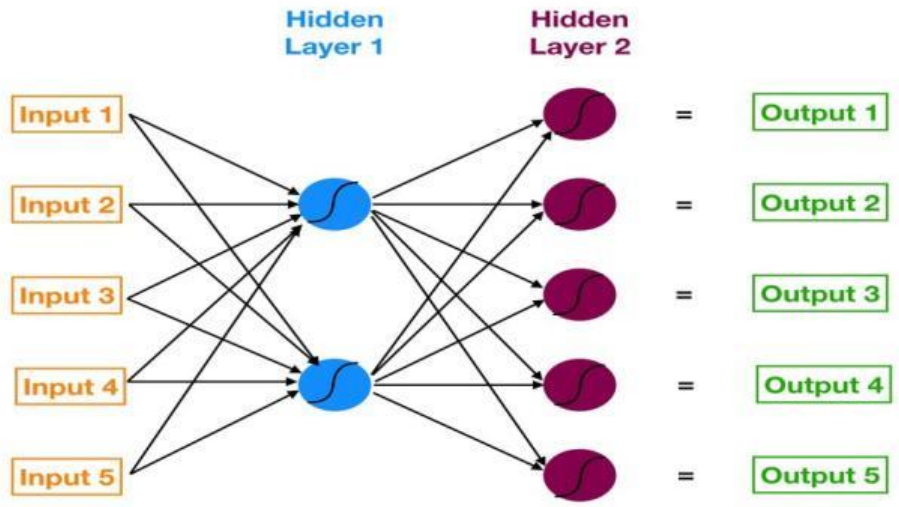

fig. 2. Neural network with two hidden layers [m]

$$f(u) = \max(0, u)$$

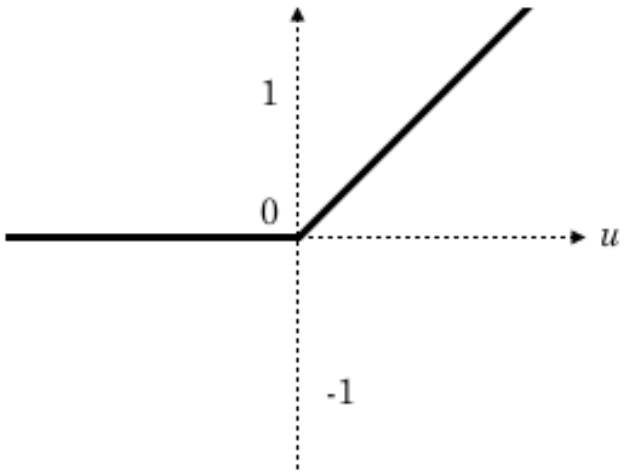

fig. 3. ReLU function appearance [n]

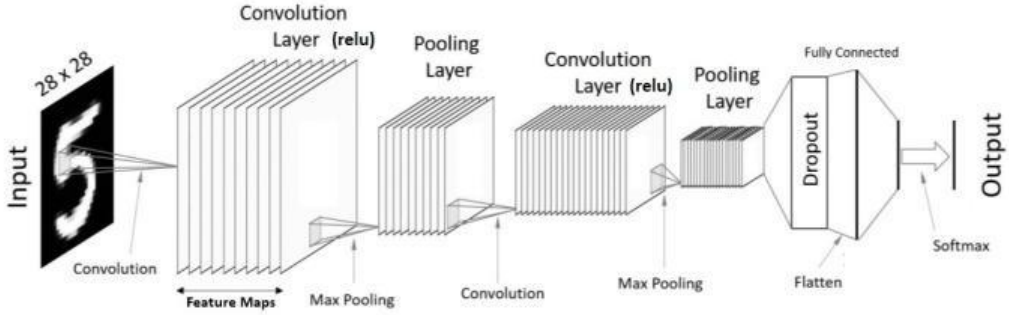

fig. 4. Example of convolutional neural network allowing five (5) recognition on an image [o]

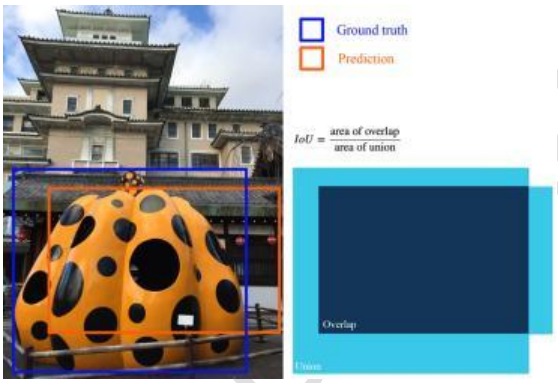

fig. 5. IoU illustration [p]

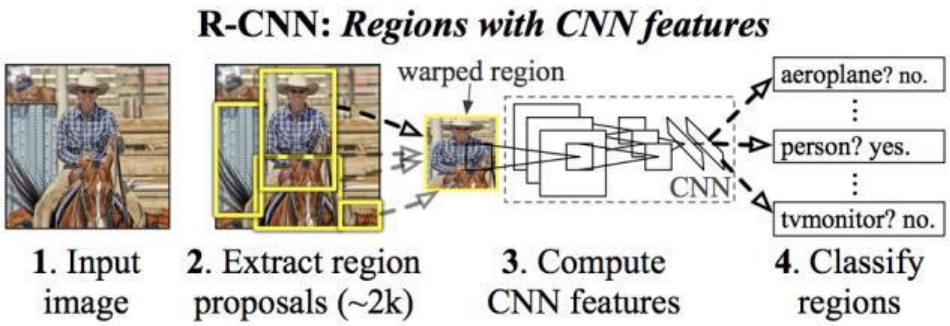

fig. 6. R-CNN model [l]

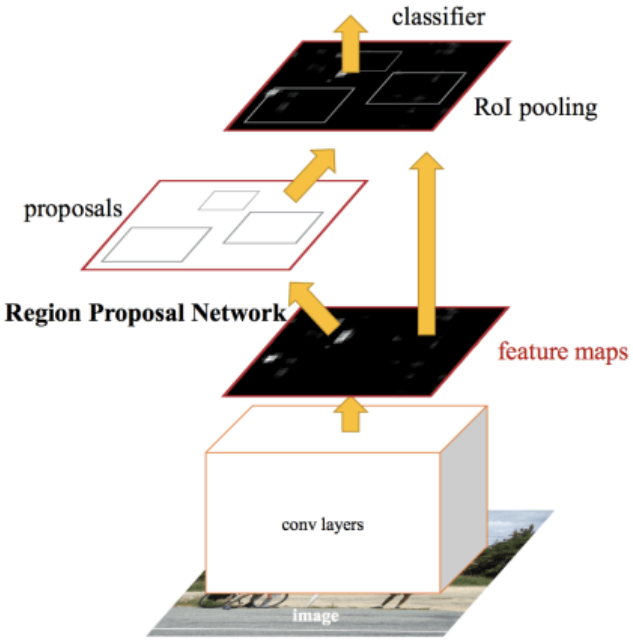

fig. 7. Faster R-CNN model [l]

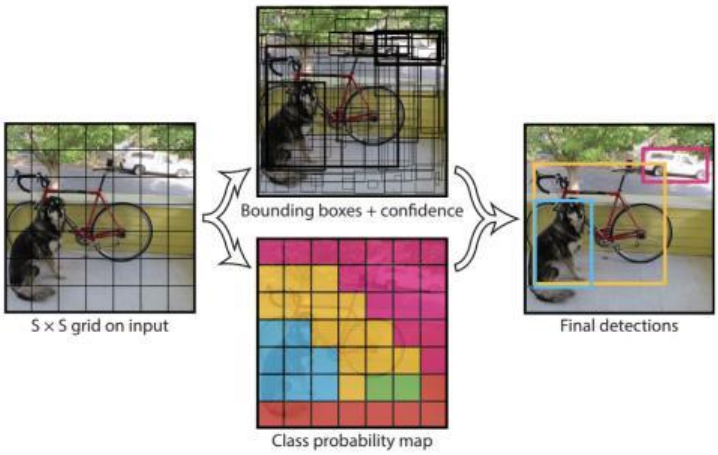

fig. 8. YOLO model [l]

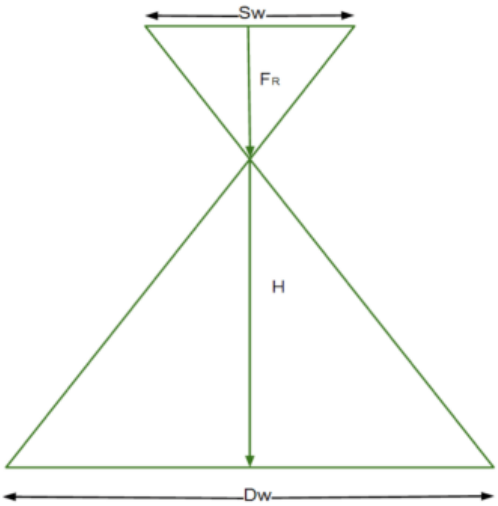

fig. 9. Informations for GSD computation [q]

fig. 10. Labeling of tomatoes with the LabelImage tool

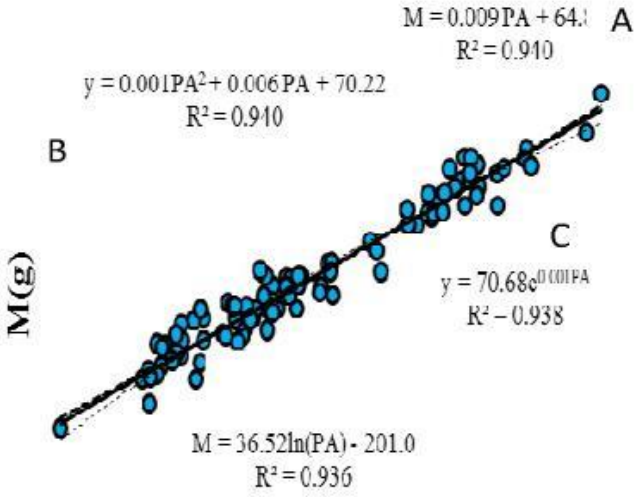

fig. 11. Mass estimator by the projected surface [s]

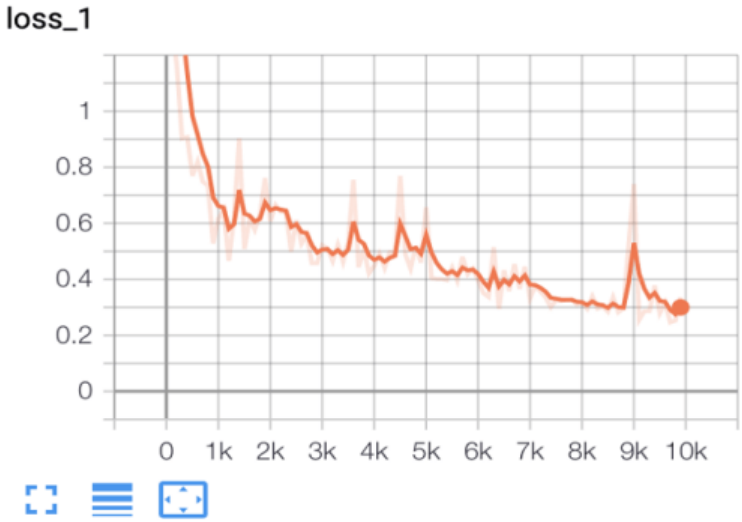

fig. 12. Learning loss curve

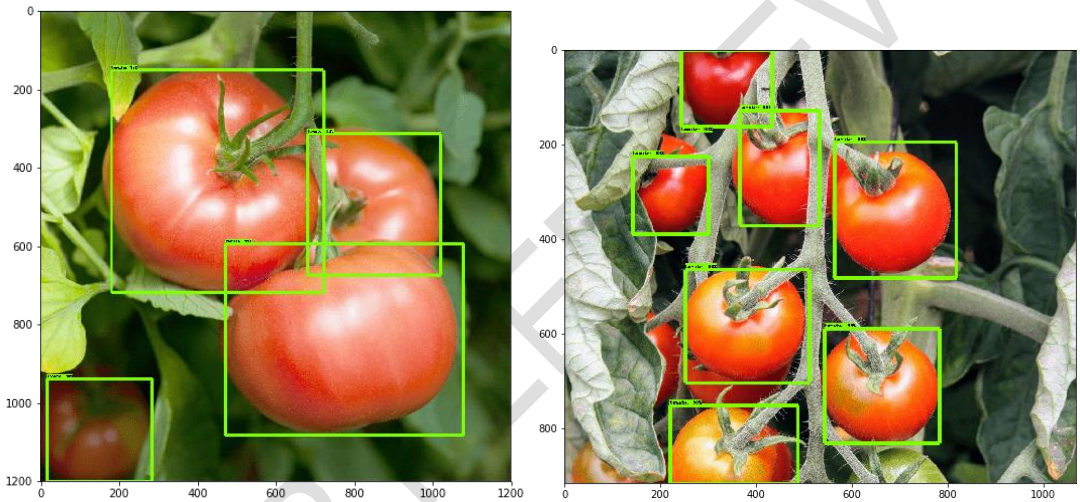

fig. 13. Examples of tomato detection with our model

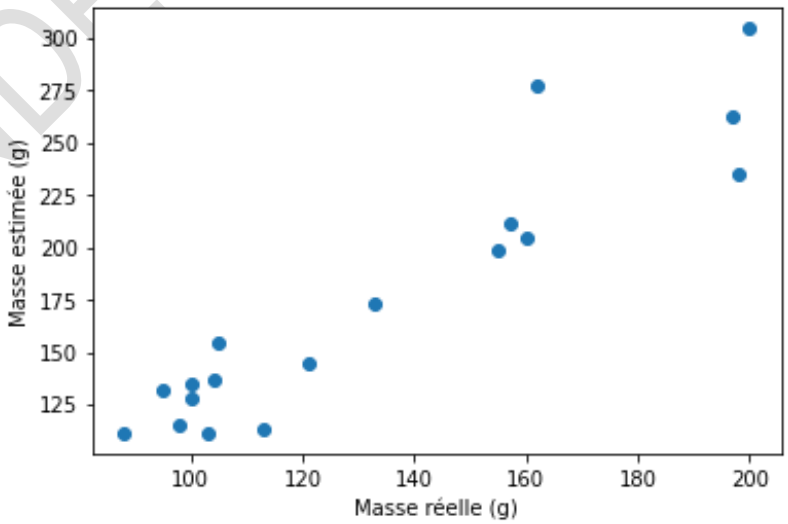

fig. 14. Comparison of the estimated mass and the actual mass of each tomato