**Automatic weed detection using YOLOv5 object detector and segmentation based on deep learning**

**Keywords:** Computer vision, Deep Learning, Object detection, Agriculture, Weed detection, Pea culture, Image Segmentation

**Abstract**

Weed control is a key mission in agricultural productivity. In the context of SSVM, weed detection is the important step to be accomplished before applying the appropriate treatments. However, several applications have been developed for weed detection or localization or both using different machine learning and deep learning techniques, but the presence of large weed species complicates the detection. In this study, we propose a method based on deep learning for crop ultra-localization. By training the YOLOv5 model, we succeeded in detecting peas with a mAP accuracy that is higher than 99% and with a speed of 350 Fps! The results obtained from this trained model allowed us to perform crop/weed discrimination with the help of another CNN doing the segmentation. This network was trained to generate segmented images with a high accuracy and a speed reaching 217 Fps. The results obtained show the feasibility of our future system which will be able to the ultra-localized destruction of weeds.

**Keywords:** computer vision, deep learning, weed detection, precision agriculture, pea cultivation

1. **Introduction**

The fight against weed is a major challenge in agriculture, because it affects agricultural productivity. The traditional methods of weed management consist of using herbicides all along the field where the crops are treated, this causes a bad influence on the environment, people's health and also on the economic part.

In the context of SSVM (Site-Specific Weed Management) (Shaw, n.d.), which consists of applying control measures only where weeds are located with a high density, our research team has succeeded in responding to this need through its work. The first research was based on the combination of Haar's pseudo-features with the Ada Boost algorithm to detect the weeds of different crops in real time (Tannouche, Sbai, Rahmoune, Agounoune, et al., 2016). In a second place, in another one we developed a new adjacency descriptor for real-time weed selection (Tannouche, Sbai, Rahmoune, Zoubir, et al., 2016). These methods are own machine learning techniques among others like RF, KNN, SVM (Islam et al., 2021) etc. Such works have been developed with the objective of controlling sprayers in order to optimize the use of herbicides.

In the context of organic agriculture, weed control is based on mechanical weeding, where control measures target only weeds. Therefore, we need a new approach to ultra-localization of weeds.

The great success of deep learning in solving problems related to computer vision and specifically object detectors in images has opened its doors to thousands of researches in the agricultural field.

The most used object detector currently, and which has given amazing results, is the YOLO model. Its special architecture has been used to solve important problems in agriculture, such as the recognition of plant diseases (Chen et al., 2022) ] or the detection of insects (Önler, 2021), the identification and detection of diseases in food products (Yao et al., 2021) or the faulty products (Cengil & Çinar, 2021), the localization of products for harvesting applications (Yan et al., 2021). In the same token, several studies have used the YOLO architecture to detect weeds in different crops (Ahmad et al., 2021; B. Liu & Bruch, 2020; Osorio et al., 2020; Soni et al., 2021).

Except the YOLO object detector, other researches have been done in order to identify and detect weeds. The authors of (Khan et al., 2021) used the Faster RCNN ResNet 101 model to detect weeds in the carrot crop. In (Espejo-Garcia et al., 2020) the authors developed a classification CNN for crop/weed identification based on pre-segmented images. All these studies have focused on detecting weeds directly, which may be difficult in the presence of large species of weeds in the same field.

In our previous work, we succeeded in developing an algorithm based on deep learning, by training the Faster RCNN ResNet 50 model to detect and ultra-localize the pea crop among weeds. This method will be used to achieve crop/weed discrimination to correctly localize the weeds (Mohammed et al., 2022).

The objective of this research is to, first, make an improvement of the detection and the ultra-localization of the crop, using the YOLOv5 model. Next, we will proceed to the part of crop/weed discrimination using a segmentation method based on deep learning techniques that will lead us to the ultra-localized weed fight.

1. **MATERIALS AND METHODS**

In this study, we propose a method based on two main steps as shown in Figure1. The first step is to use the YOLOv5 object detector, after training, to ultra-localize peas from the input images. In parallel, a CNN will be trained for the plant/soil segmentation. By combining the results of the two neural networks, our resulting algorithm will be able to generate an image that contains only weeds, i.e. a weed/pea discrimination.

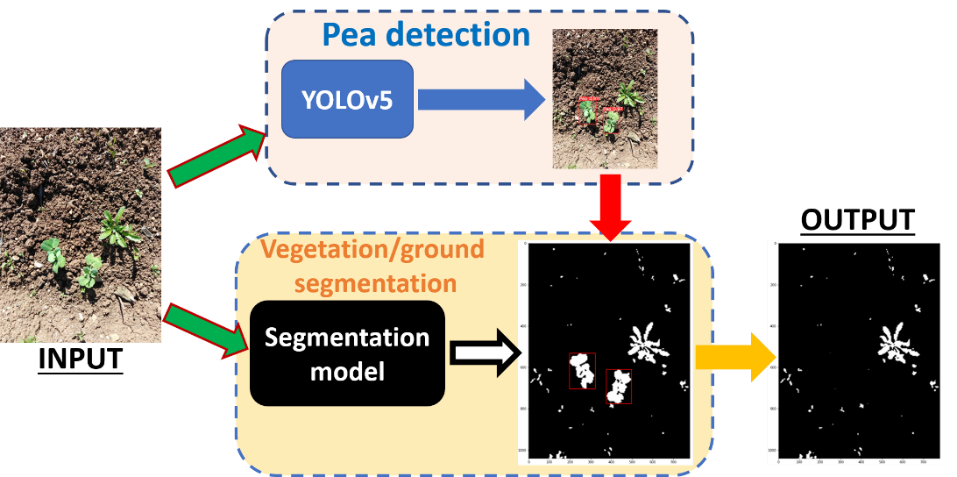


Figure 1: the methodology of this study

* 1. Yolov5

YOLO, abbreviation of You Look Only Once, is the most popular and fastest family of object detectors in the literature, and specifically real-time object detection.

Other recent object detectors, such as R-CNN, use region proposal methods to first generate Bounding boxes in an image, then run a classifier on these proposed boxes. These complex pipelines are slow and difficult to optimize because each component must be trained separately.

YOLO's philosophy is based on solving this problem. Thanks to its architecture, the YOLO model only looks at an image once to predict the objects present and their location (Redmon et al., 2015).

The research for the two recent versions of YOLO, YOLOv4 and YOLOv5 was quite near, as well as the architecture of YOLOv4 and YOLOv5 is very similar. But to avoid any conflict, Glenn, the author of YOLOv5 decided to name his version of YOLO, YOLOv5. The research of YOLOv4 and YOLOv5 integrates the latest innovations in computer vision (Bochkovskiy et al., 2020)

The YOLO network consists of three main elements as shown in Figure 2:

* Backbone: it is a network of features extractors that allow to obtain features from the input image, and to form them at different levels.
* Neck: this is the link between the backbone and the head. It contains a multi-layer network that allows mixing and combining image features to prepare them for prediction.
* Head: from the characteristics resulting from the Neck, it predicts the location and the class of the objects.

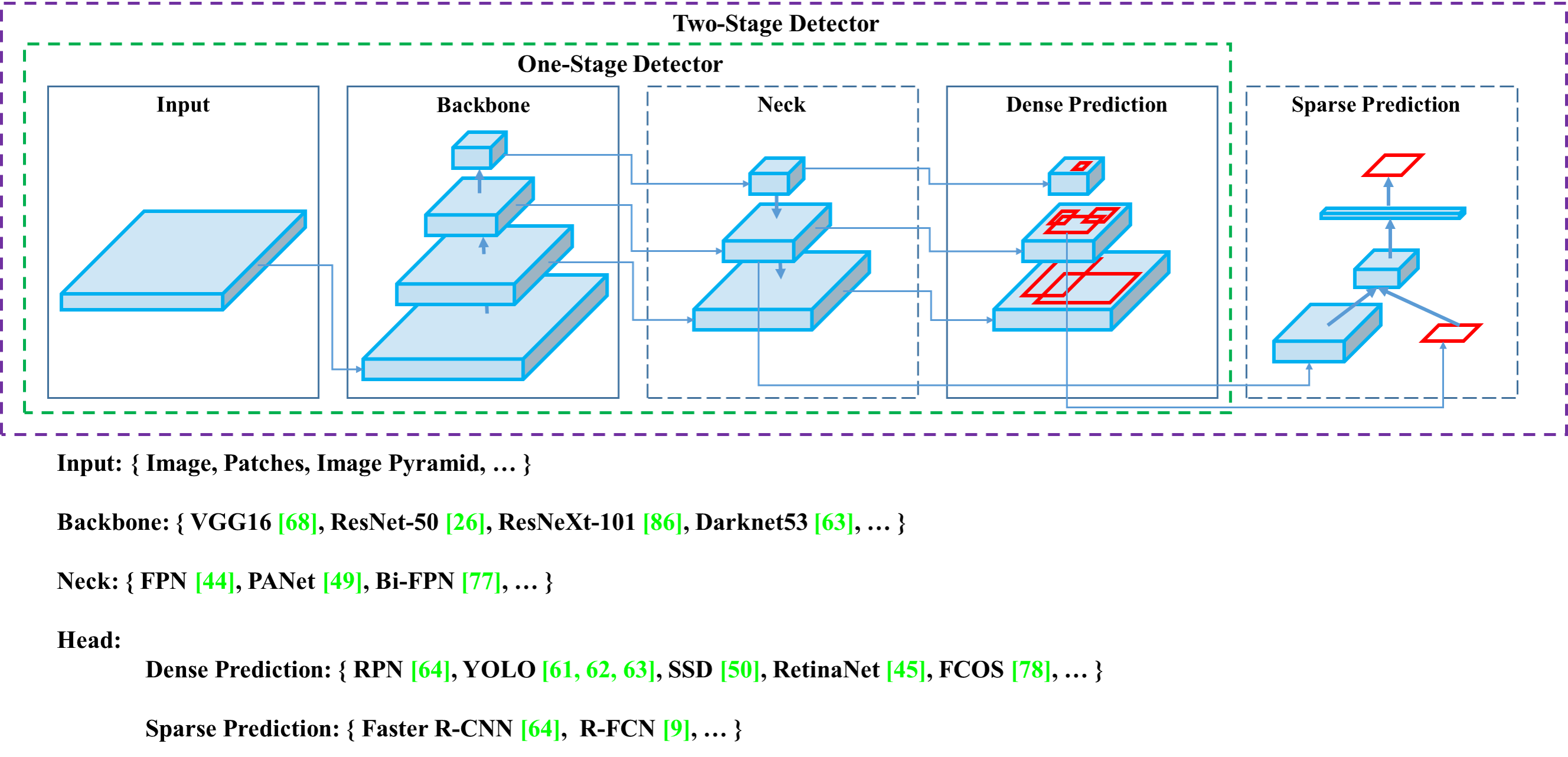


Figure 2 : the architecture of the YOLO model (Bochkovskiy et al., 2020)

The YOLOv5 model uses CSPDarknet53 as a backbone. This backbone solves the problem of repetition in large backbones and includes the gradient change in the feature map, which reduces the speed of inference, increases the accuracy and reduces the size of the model by decreasing the parameters. (Nepal & Eslamiat, 2022)

This CSPDarknet53 neural network uses the CSPNet strategy to split the base layer feature map by copying it and sending a copy through the dense block to the next stage and then fuses it with the original map. This split and fuse strategy allows for a larger gradient flow through the network.

The DarkNet53 network idea was first introduced in the development of the YOLOv3 model. Figure 3 shows the architecture of DarkNet53. This network uses successively 33 and 11 convolution layers, and it has shortened connections. Therefore, it has, in total, 53 convolutions layers (Redmon & Farhadi, 2018)

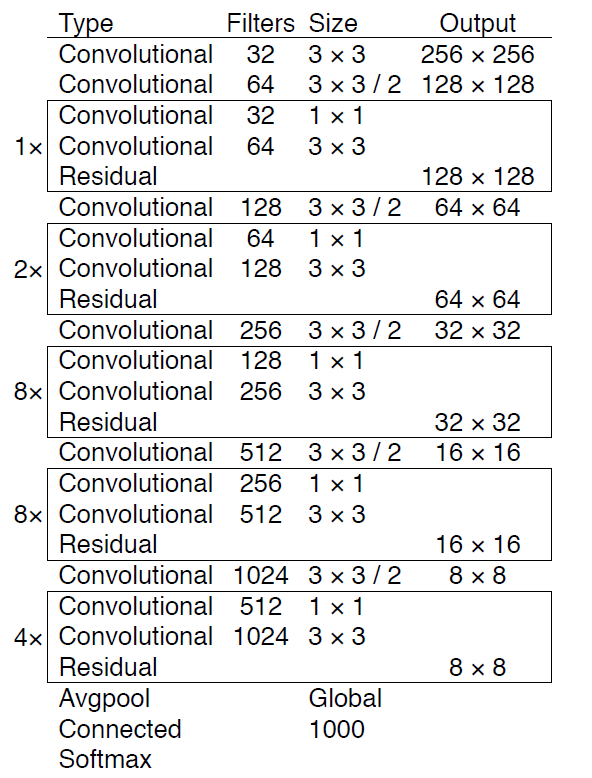


Figure 3: the architecture of the Darknet network (Redmon & Farhadi, 2018)

The YOLOv5 model uses the PANet path aggregation network as a Neck to support data flow. PANet provides improvements to the Pyramid Feature Network (FPN), which consists of several layers that distribute from bottom to top and top to bottom as shown in Figure 4. PANet improves localization in the lower layers, which improves the accuracy of object localization. (Nepal & Eslamiat, 2022)

In research conducted by Google Brain on EfficientDet's object detection architecture. EfficientDet authors found that PANet achieves better accuracy than FPN and NAS-FPN, but at the cost of more parameters and computation (Tan et al., 2019).

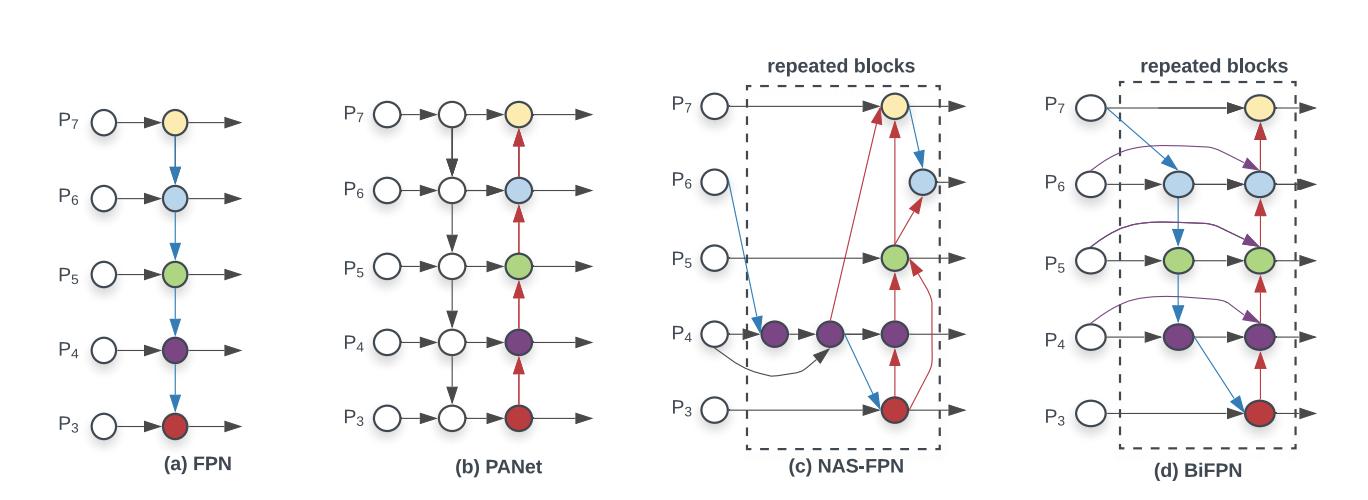


Figure 4 : the architecture of the PANet network (Tan et al., 2019)

Each of the above Pi represents a feature layer in CSP backbone.

For the head of YOLOv5, it is similar to that of YOLOv4 and YOLOv3, which generates three different feature map outputs to perform multi-scale prediction. This also effectively improves the prediction of small to large objects in the YOLOv5 model.

The image is passed to CSPDarknet53 for feature extraction and then to PANet for feature fusion. Finally, the YOLO layer generates the results (Nepal & Eslamiat, 2022).

For activation and optimization, YOLOv5 uses sigmoid and leaky ReLU activation functions, as well as SGD and ADAM as optimization functions.

And for loss calculation, it uses binary cross entropy with logit loss.

There are 5 sizes of Yolov5 (n, s, m, l and xl). The larger the network is, then the training parameters increase, as well as the performance of the network. But more parameters mean more learning time and detection takes longer.

For this research, we studied the 5 versions of YOLOv5, to choose the best one, in terms of accuracy and speed as well as its flexibility to implement it in our future system.

One of the important particularities of YOLOv5 is the use of PyTorch instead of Darknet. The latter was written mainly in C language to provide modification and control of encoded operations in the network. This can make porting new search results slower, as custom gradient calculations must be written for each new addition. The benefit of the PyTorch library is that it allows for easy control of the network parameters.

* The annotations for YOLOv5

For learning purposes, YOLO v5 accepts only annotations, for each image in the form of a .txt file where each line of the text file describes a bounding box.

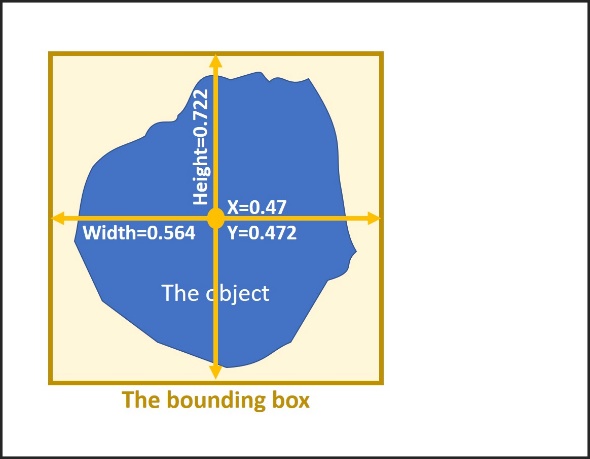


Figure 5 : the bounding box coordinates supported by YOLO

In figure 5, there is an object, the bounding box that surrounds it is characterized by the following coordinates: the coordinates of the center (X, Y), the length and width of the bounding box, The coordinates of the bounding box must be normalized by the dimensions of the image to have values between 0 and 1.

In the .txt file that describes this image, the object must be represented by the following line:

''0 0.47 0.472 0.564 0.722'', with x\_center=0.47, y\_center=0.472, width=0.564, height=0.722 and the class of the object=0. Each line of the file contains the coordinates of each object in the image.

* 1. Image segmentation:

Among the important areas of application of Deep Learning in computer vision, we find the segmentation of images, it has been very successful in solving problems related to this area.

The segmentation consists in labeling any object belonging to an image, determining their limits with precision in relation to the detection of objects, it allows the classification of each pixel in the image, so it can be considered, in the deep learning, as a classification by pixel.

We differentiate between two types of segmentation: instance segmentation and semantic segmentation. The instance segmentation allows to give different labels to the same objects of the image contrary to the semantic segmentation which does not differentiate between objects of the same class.

* + 1. Supervised segmentation

It is the set of traditional techniques that allows to do the image segmentation and that requires the human control to correct the errors that can appear from the image processing.

It is based on computer methods such as optimization algorithms that adopt a local vision of the characteristics of the images, the adaptive thresholding, the genetic algorithm of Otsu ...

Among the simplest methods, we find the thresholding that allows from a fixed threshold to specify the pixels belonging to the objects, only the pixels whose value is higher than the threshold are kept.

The result is a binarization of the image, and for a better thresholding it is necessary to choose the optimal threshold. The most used thresholding is based on the color space where the threshold is a color intensity.

There are many color spaces, the most popular is the RGB model. It is the most used color space. But it has a lack of precision in the applications of artificial vision (Mythili & Kavitha, 2012)

The HSV (hue, saturation, value) color space is the most used by color specialists, because it corresponds better than the RGB color space to the way people see colors.

* + 1. Segmentation using deep learning

The amazing results of deep learning in different domains open its doors to new applications, among them image segmentation. Many works have been developed to solve the problems related to segmentation.

Among the application areas that have made great contributions in his research, the segmentation of medical/biomedical images. In (Ronneberger et al., 2015) the authors proposed the U-Net for the segmentation of biological microscopy images (Minaee et al., 2020)

Their network shown in Figure 6 consists of two parts, a subsampling path to extract features at an FCN-like architecture with 3 × 3 convolutions, and a symmetric oversampling path that uses upward convolution to reduce the number of feature maps while increasing their dimensions.

To avoid information lost, a copy of each subsampling map is merged with the equivalent oversampling map. Each feature map is processed with a 1×1 convolution to categorize each pixel of the input image.

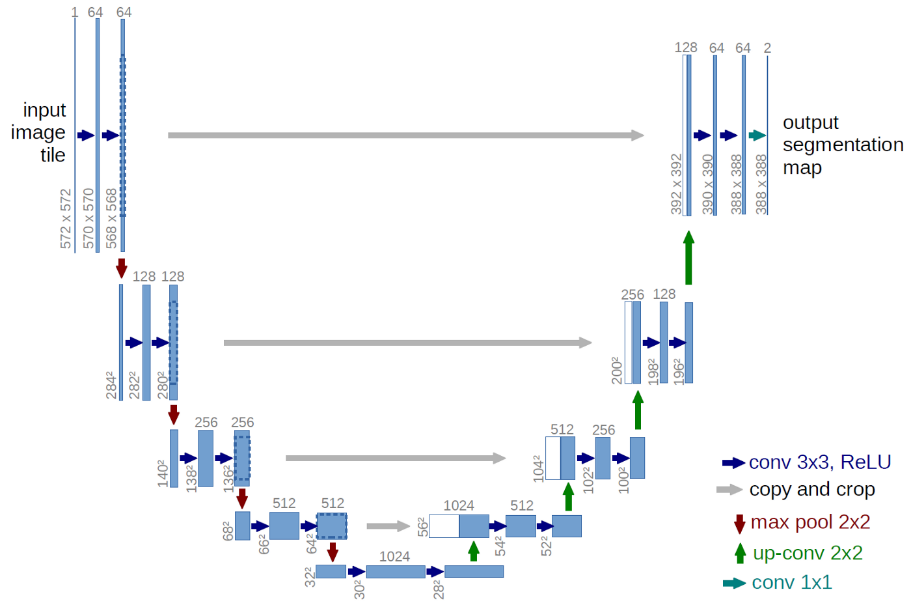


Figure 6 : the architecture of the UNet network (Minaee et al., 2020)

Different architectures have been developed to improve the performance of U-Net. Hasib Zunair et al. proposed the Sharp U-Net model which is "a novel depth convolutional encoder-decoder network architecture for binary and multi-class image segmentation. The principal idea is to convolve the output of the encoder feature map with a spatial enhancement kernel/filter, before performing fusion with the decoder features in the skip connections" (Zunair & ben Hamza, 2021)

Other models as popular as UNet, such as LinkNet and FPN, LinkNet (figure 7.1) differs from Unet by adding the subsampled features with the oversampled ones instead of concatenating them. The FPN model works by creating feature maps in the shape of two pyramids (Figure 7.b) by summing up features of similar size and concatenating them to generate feature-rich segmentation maps at each level (Parmar et al., 2020).



Figure 7 : the architecture of the two networks LinkNet (a) and FPN (b)(Parmar et al., 2020)

In this research, we focused on the different models presented before: UNet, LinkNet, SharpUNet and FPN.

* 1. Dataset

For the training of the detection and segmentation models, we used the dataset already collected and prepared in our previous work (Mohammed et al., 2022). Figure 8 illustrates the techniques used for the acquisition of the images. To enrich the dataset, we proceeded to augment it using image processing tools.

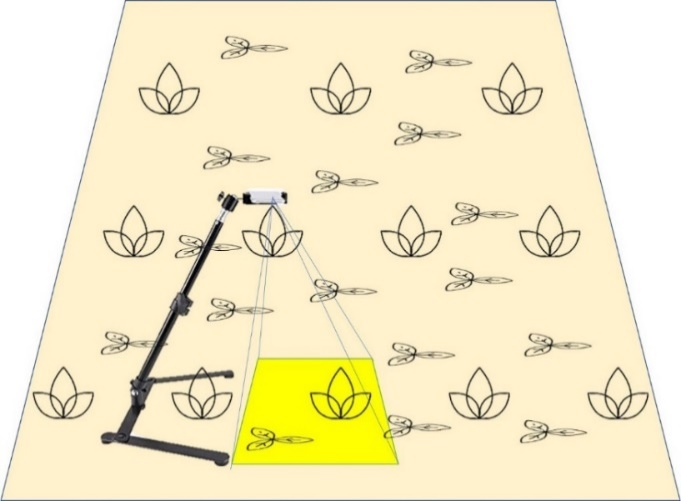


Figure 8 : image acquisition techniques, the device used is the digital camera phone (Huawei Y7 prime), fixed in 40cm from the soil

* + 1. YOLO annotations

The images were annotated in XML format which is the most commonly used format for annotating images, but, as mentioned earlier, the YOLO models have their own annotation format.

Since annotating images is a fastidious and slow process, we opted to convert, automatically, the annotations in xml format to the txt format accepted by YOLO, by determining the link between the both.

In xml format, each bounding box is defined by the two points min and max (its coordinates successively are xmin, ymin, xmax and ymax). In YOLO format, the bounding box is determined by its center point (its coordinates are X and Y), so the relations that link the two are :

; ; ;

For the normalization of the coordinates, they must be divided by the length and width of the image.

The conversion of the files, reading xml files and writing files in txt format, has been done by using the libraries (xmltodict and os) and python functionalities.

Finally, we divided the dataset in 3 parts, 7360 images for training, 1840 images for validation and 54 test images.

* + 1. Annotations of segmentation models

Our research is also interested in the segmentation of images in order to discriminate the herbs on the soil (background), so we have only one class to identify. The segmentation, in this case, is to determine whether each pixel belongs to the herbs or the background. So, the annotations in our case is a binary mask that contains the pixels belonging to the herb class.

If we look for how to separate the herbs from the background, we quickly see that they can be differentiated by their colors. So, we opted to use the segmentation based on colors.

The color segmentation consists first of all, in specifying the color space to be used to select the target colors. It is necessary first of all to analyze our images in the spaces of the colors by visualizing the distribution of the colors of the pixels as it shows the figure 9.

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

Figure 9 : 3D plot of the distribution of a sample image in the two spaces RGB (a) and HSV (b)

In the RGB space (figure 9.a), we can see that the herbs (green color) extend over all RGB colors, so the herbs are difficult to separate in this space, but in the HSV space (figure 9.b), they are located in well-defined areas. For this, our segmentation was made in HSV space (*Image Segmentation Using Color Spaces in OpenCV + Python – Real Python*, n.d.).

By manually adjusting the color margins, we performed the supervised segmentation of 500 images in order to generalize them using deep learning techniques.

* 1. Software

For importing, processing our dataset, building, training and testing the models, we chose to use the "Google colab pro" platform, which is characterized by its modern performance in terms of memory and speed. It offers us the powerful GPU "Tesla P100-PCIE-16GB", and a large RAM memory reaches 26GB, as well as it gives us a wider working time (*Colab Pro: Is It Worth the Money? | by Dario Radečić | Towards Data Science*, n.d.).

1. **RESULTS AND DISCUSSIONS**
   1. YOLO training
      1. The Hyperparameters

The choice of the initial values of the hyperparameters is a sensitive and very important phase, a good choice means a good training and optimal results. There are several search methods such as grid searches, these traditional methods are difficult to implement because they are expensive (requires a lot of computation), the search space is of multiple dimensions and the absence of links between these dimensions, hence the need to find other method.

Glenn Jocher (*GitHub - Ultralytics/Yolov5: YOLOv5 🚀 in PyTorch > ONNX > CoreML > TFLite*, n.d.) the author of YOLOv5 suggests a method of searching for the most optimal hyperparameters using the genetic algorithm (GA) with genetic operators the crossing and the mutation.

Starting from the algorithm predefined by the author, we launched 50 steps of search for hyperparameters that corresponds to our dataset, after each 10 steps the algorithm evaluates the performance of the chosen hyperparameters, we visualize the results obtained in Figure 10, with The values are on the x axis and the fitness on the y axis. More fitness implies good results. The best values obtained are shown at the top of each graph.

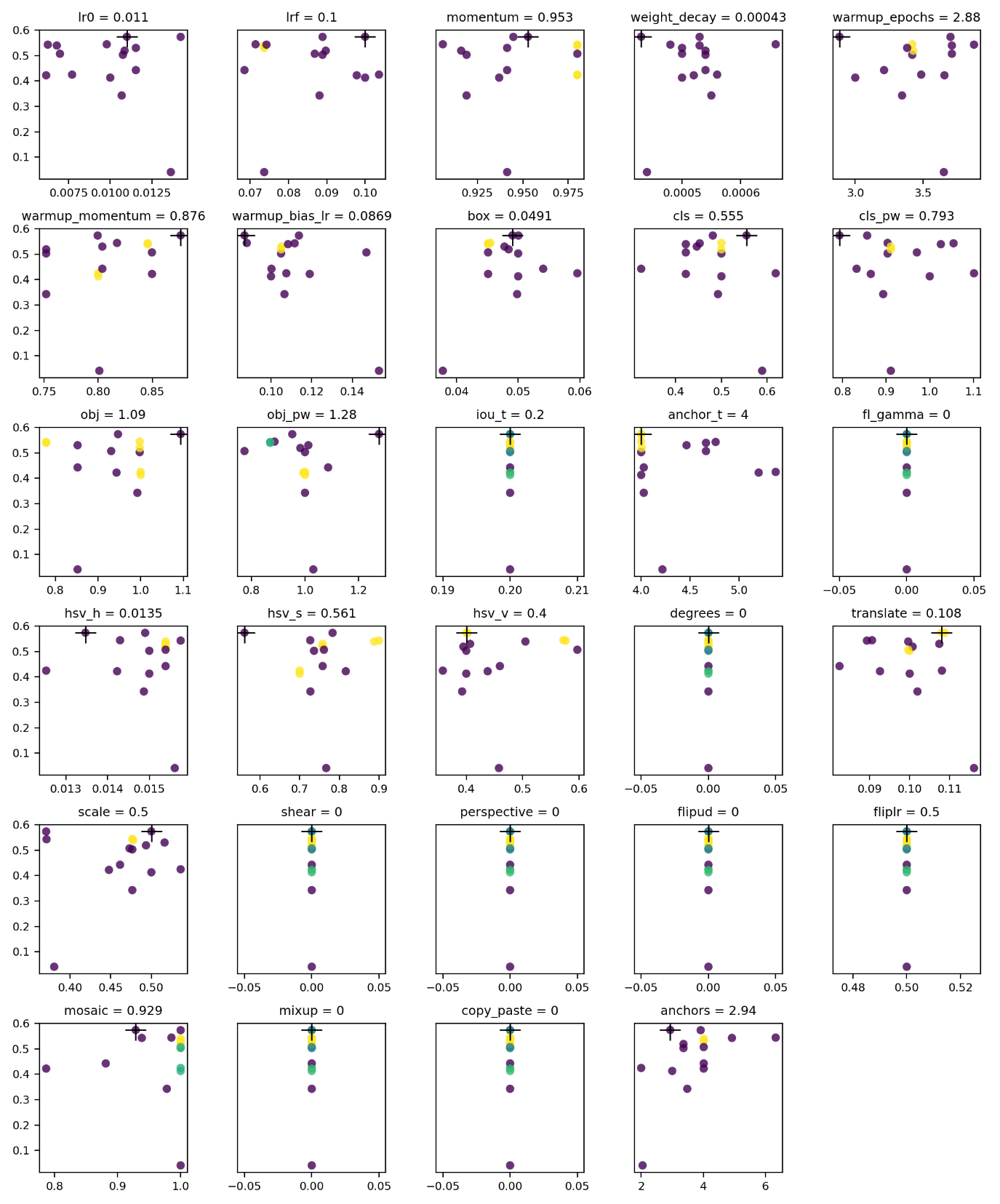


Figure 10 : Visualization of the best hyperparameters search results after 50 steps

YOLOv5 has 29 hyperparameters used for various training parameters. The totality of the final hyperparameters obtained is shown in table 1.

Table 1 : the hyperparameters chosen after the search and their meanings

|  |  |  |
| --- | --- | --- |
| Hyperparameter | Value | Meaning |
| lr0 | 0.011 | initial learning rate |
| lrf | 0.1 | final OneCycleLR learning rate |
| momentum | 0.935 | SGD momentum/Adam beta1 |
| weight\_decay | 0.00043 | optimizer weight decay |
| warmup\_epochs | 2.88 | warmup epochs |
| warmup\_momentum | 0.876 | warmup initial momentum |
| warmup\_bias\_lr | 0.08669 | warmup initial bias lr |
| box | 0.0491 | box loss gain |
| cls | 0.555 | cls loss gain |
| cls\_pw | 0.79300 | cls BCELoss positive\_weight |
| obj | 1.09 | obj loss gain (scale with pixels) |
| obj\_pw | 1.28 | obj BCELoss positive\_weight |
| iou\_t | 0.2 | IoU training threshold |
| anchor\_t | 4 | anchor-multiple threshold |
| anchors | 0 | anchors per output grid |
| fl\_gamma | 0.0135 | focal loss gamma |
| hsv\_h | 0.561 | image HSV-Hue augmentation |
| hsv\_s | 0.4 | image HSV-Saturation augmentation |
| hsv\_v | 0 | image HSV-Value augmentation |
| degrees | 0.108 | image rotation |
| translate | 0.5 | image translation |
| scale | 0 | image scale |
| shear | 0 | image shear |
| perspective | 0 | image perspective |
| flipud | 0.5 | image flip up-down |
| fliplr | 0.929 | image flip left-right |
| mosaic | 0 | image mosaic |
| mixup | 0 | image mixup |

* + 1. YOLOv5 models training

After the preparation of the training environment and the setting of the hyperparameters, we started the training of the 5 sizes of YOLOv5, using the two training techniques, Training From Scratch, and Transfer Learning with pre-trained weights. With 64 as batch-size, the image size is 320 and 200 training steps.

Different metrics are used for model evaluation during training, such as error, precision, recall etc. mAP, mean Average Precision, remains the most used metric to properly evaluate the performance of training object detectors (Mohammed et al., 2022). The general definition of mean average precision (mAP) is:

(1)

With p the precision and r the recall, which are defined by :

(2)

(3)

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

Figure 11 : training results of YOLOv5 models trained from scratch (a) and with the transfer learning technique (b) with a threshold of IoU=0.5.

For the results of figure 11, we can see that in the from scratch training the accuracy of the models is very perturbed compared to the transfer learning during the first 40 steps. In the last steps, we notice that the precision of all the models stabilizes around 99% for the transfer learning, and for the from scratch training the final precision varies from 98% to more than 99% depending on the size of the model, more size implies more precision. We can see that the precision increases if the size of the model increases as well as the precision when training pre-trained weights better than when training from scratch weights.

It appears well if we vary the threshold of IoU from 0.5 to 0.95, as it is shown in table 2, for the training of the pre-trained weights the final mAP varies from 80% to the most of 92%, on the other hand the mAP of the models trained from scratch starts from 76% and does not exceed 87%. The table 2 confirms the difference in performance between the different sizes of YOLOv5.

We resume the precision of all the trained models in different sizes and training mode, after 150 training steps, in table 2, and comparing them with the model trained in our previous work.

Table 2 : evaluation results of the YOLOv5 models trained from scratch and with the transfer learning technique and the Faster RCNN ResNet 50 model (our previous work)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| The model | mAP IoU=0.5 | | mAP IoU=0.5:0.95 | |
| Transfer Learning | Learning From Scratch | Transfer Learning | Learning From Scratch |
| YOLOv5n | 99.03% | 98.04% | 79.83% | 76.12% |
| YOLOv5s | 99.30% | 98.97% | 84.81% | 80.75% |
| YOLOv5m | 99.47% | 98.66% | 89.91% | 83.92% |
| YOLOv5l | 99.45% | 99.29% | 90.97% | 85.43% |
| YOLOv5x | 99.45% | 99.32% | 92.25% | 86.04% |
| Faster RCNN ResNet 50 | 85.07% | - | - | - |

* + 1. YOLOv5 models evaluation

After the training, we proceeded to the evaluation of the resulting models, to extract their performance, precisely in terms of precision and speed relationship. This evaluation was done with the evaluation dataset. At each step we have enlarged the size of the incoming images to see the reaction of the model in speed (the larger the input means the longer the processing time). Figure 12 illustrates the results of the evaluation, with the coordinate axis representing the precision and the x-axis representing the processing time of a single image which includes the pre-processing time, the inference time and the Non-Maximum Suppression time (NMS to remove the overlapping bounding boxes).

The evaluation was performed with the same processor used in the training, Tesla P100-PCIE-16GB.

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

Figure 12 : the evolution of speed and precision as a function of input image size for YOLOv5 models trained from scratch (a) and pre-trained models (b).

By analyzing the graphs, we can see that the variation margin of response time of the large size models is very high in comparison with the small size models. For the nano and small sizes, the time does not exceed 20ms (50 Fps). Compared to the two training modes, from scratch and transfer learning, we do not see a big difference in terms of speed, but the graphs in the figure 12 confirm the difference in precision already observed (the precision of the models trained from scratch decreases to at least 20%, while those trained with pre-trained weights remain above 30%).

Table 3 shows the response time per frame as well as its equivalent speed by choosing 256 as the size of the input images which is the size we defined during the training.

Table 3 : speeds of the YOLOv5 trained from scratch and pre-trained models.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Learning from scratch | | Transfer Learning | | |
|  | Time (ms) | Speed (Fps) | | Time (ms) | Speed (Fps) |
| YOLOv5n | **2.468** | 405 | **1.977** | | 506 |
| YOLOv5s | **2.837** | 352 | **2.801** | | 357 |
| YOLOv5m | **3.363** | 297 | **3.960** | | 253 |
| YOLOv5l | **4.796** | 209 | **4.744** | | 211 |
| YOLOv5x | **6.857** | 146 | **7.905** | | 127 |

* 1. Segmentation models learning

For the segmentation models, we chose to train the 5 models UNet, SharpUNet, miniUNet, FPN and LinkNet. The UNet model was created following the UNet architecture with 5 levels of convolution, by decreasing these levels to 3 we obtained the miniUNet, the SharpUNet is similar to UNet but adding convolutions of the encoder feature map outputs with a kernel before performing the fusion, the LinkNet model has the LinkNet architecture of 3 levels and the FPN model has the pyramid architecture of 5 levels.

We have chosen to train these models with the same parameters that are shown in table 4, only the batch size that has been varied to see the reaction of the models with this parameter.

Table 4 : training parameters of the 5 networks.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Epochs | Batch size | Optimizer | Loss | Metrics | lr0 | lrf |
| 100 | 4, 12 and 32 | Adam | Binary crossentropy | Jaccard and Dice | 0.0010 | 1e-8 |

For the evaluation of the model we used the Dice and Jaccard indices, which are the most widely used metrics for the evaluation of segmentation models (X. Liu et al., 2021). The Dice and Jaccard indices are used to calculate and measure the similarity or overlap between two object shapes A and B, they are defined by :

The value of these two indices is between 0 and 1. More the closer the value is to 1 more the similarity is complete.

|  |
| --- |
|  |
| (a) |
|  |
| (b) |
|  |
| (c) |

Figure 13 : the evaluation results of the 5 networks represented by the Jaccard and Dice similarity indices, changing the Batch Size parameters, (a) BZ=4, (b) BZ=12, (c) BZ=32.

Analyzing the results in Figure 13, we can see that the similarity indices (Dice and Jaccard) decrease with increasing batch size, as well as the difference between the performances of the trained models appears well in the large batch size (c), in this case we can see that the UNet architectures have obtained good results compared to the other architectures and precisely the SharpUNet which has exceeded 90% for both indices in several cases.

Figure 14 shows the segmentation result of an image belonging to the validation dataset, comparing the prediction with the annotation mask, we observe that there is almost a perfect similarity between the two. The Jaccard index for this result is 0.905.

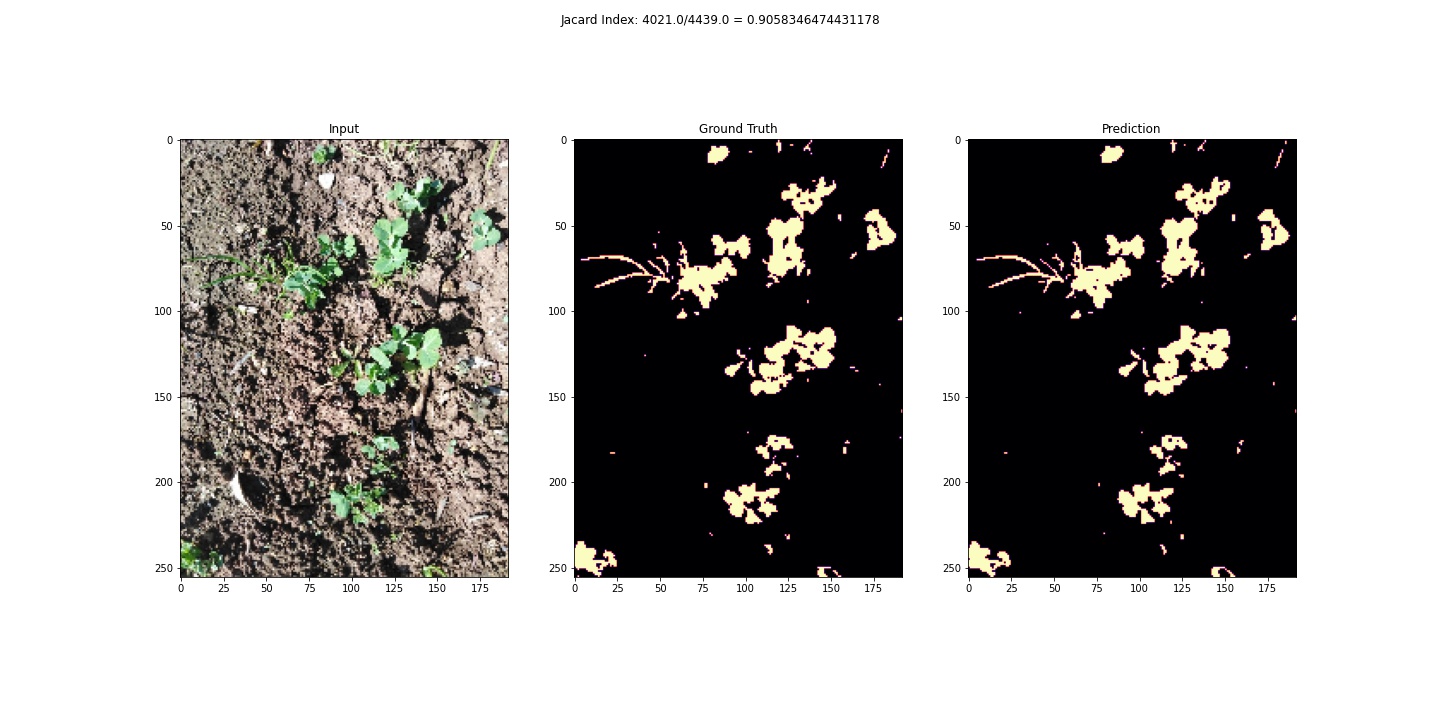


Figure 14 : comparison between the prediction of an image and the mask already generated in the annotation

After training, we ran an evaluation of the resulting models with test images (54 images), to measure the models' performance in terms of accuracy and speed.

Table 5 summarizes the results of this evaluation, it contains the prediction time for 54 test images, its equivalent in speed, the average of the Jaccard and Dice indices for the 54 images and finally the relationships of the indices to the equivalent time.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Speed for 54 prediction | | | Jaccard | Dice | J/t | D/t |
|  | (ms) for 54 images | (ms) per image | Fps |
| SharpUNet | 294 | 5.44 | 184 | 0.9046 | 0.9497 | 0.166152 | 0.174433 |
| FPN | 248.6 | 4.60 | 217 | 0.894 | 0.944 | 0.194191 | 0.205052 |
| Unet | 282.4 | 5.23 | 191 | 0.904 | 0.949 | 0.172861 | 0.181466 |
| miniUnet | 187.2 | 3.47 | 288 | 0.901 | 0.947 | 0.259904 | 0.273173 |
| LinkNet | 181.2 | 3.36 | 298 | 0.894 | 0.943 | 0.266424 | 0.281026 |
|  | | | | | Average : | 0.211906 | 0.22303 |

Table 5 : the results of the 5 trained networks in terms of speed, accuracy, and accuracy/speed ratio

The SharpUNet model is the best in prediction and LinkNet is the least accurate, but on the other hand, LinkNet is the fastest and SharpUNet the slowest of the models. We notice that the similarity indexes are almost equal, but there are important differences in the speed of the models, this is shown in the index/speed ratios, the fast models have better ratios than the slow models and the closest to the average is the FPN model.

Combining the two models, we obtain the results of figure 15, the YOLOv5 model was designed for the detection of the crop in images that contains in addition to the crops the weeds (figure 15.a). The segmentation model is used to discriminate weeds from the ground in parallel to the detection (Figure 15.b).

With the help of bounding boxes resulting from YOLOv5 the crop is removed to keep only the weeds (figure 15.c).

|  |  |  |
| --- | --- | --- |
|  | | |
| (a) | (b) | (c) |

Figure 15 : samples of pea/weed discrimination results (c) from input images (a) by combining localization and segmentation (b).

* 1. Discussions

Our method allowed to extract weeds from the images which contains in addition to the weeds the crop. This method is divided into two great parts, the detection and the ultra-localization of the crop and the segmentation.

* For the detection of the crop, it has been observed that the training of the pre-trained weights is better than the training from scratch, and that the increase of the size of the model YOLOv5 does not cause a great change in precision, but in relation to the speed the models of small size are faster than those of big size. For the application that we want to realize it our choice will be limited to the medium models that are fast and of light sizes, these are YOLOv5s and YOLOv5m.
* For segmentation, we also notice that the studied models do not have a big difference in terms of accuracy. Comparing the index/speed ratios, we notice that the pyramidal FPN architecture is the right choice because it is nearer to the average of all ratios.

Examining the results obtained, it was found that for object detection the choice of a lower IoU threshold (0.5) caused errors related to the occlusion of the bounding box, i.e. some bounding boxes do not contain the whole crop as shown in Figure 16, this problem affects the final results because the parts not contained in the bounding box are considered as weeds after the crop/weed discrimination. In this case, it is absolutely necessary to increase the IoU threshold to correct these errors.

|  |  |
| --- | --- |
|  |  |
| (a) | (b) |

Figure 16 : an occlusion error (yellow box) and its influence on the final results (b)

It was also noticed that the resulting images contain "salt" type noises due to two reasons, the first one is the presence of medium and small weeds that must be preserved, the second one is the errors related to the segmentation results in the annotations or in the results of the segmentation model, these can be removed by using morphological filters.

1. **CONCLUSION AND FUTUR WORK**

In this work, we succeeded in building an algorithm based totally on deep learning, this algorithm is able to detect and ultra-localize in real time the weeds in the pea crop. We have achieved the two objectives set at the beginning of this research.

The first step was to improve the results obtained in our previous work. Using the YOLOv5 object detector, we were able to identify and ultra-localize the pea among weeds with predictions reaching 96%. The study of different sizes of YOLOv5 led us to choose the best size in terms of precision and speed. The model we chose has a mAP accuracy that exceeds 99% (85% in the previous research) and the most important is the speed that reaches more than 350 Fps!

Secondly, we used deep learning techniques to segment the input images. By studying different segmentation CNNs, we chose the best one that exceeds 89% for similarity tests (the Jaccard and Dice indices), and its speed is about 217 Fps.

By combining the two techniques we were able to discriminate peas from weeds and obtain binary masks that contain only weeds.

In our future work, we have the vision to implement the resulting algorithm in an automated embedded system, able to simulate manual weed control by mechanical, thermal or chemical destruction of weeds by targeting only them.

**References**

Ahmad, A., Saraswat, D., Aggarwal, V., Etienne, A., & Hancock, B. (2021). Performance of deep learning models for classifying and detecting common weeds in corn and soybean production systems. *Computers and Electronics in Agriculture*, *184*. https://doi.org/10.1016/j.compag.2021.106081

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). *YOLOv4: Optimal Speed and Accuracy of Object Detection*. http://arxiv.org/abs/2004.10934

Cengil, E., & Çinar, A. (2021). Poisonous Mushroom Detection using YOLOV5. *Turkish Journal of Science & Technology*, *16*(1), 119–127.

Chen, Z., Wu, R., Lin, Y., Li, C., Chen, S., Yuan, Z., Chen, S., & Zou, X. (2022). Plant Disease Recognition Model Based on Improved YOLOv5. *Agronomy*, *12*(2). https://doi.org/10.3390/agronomy12020365

*Colab Pro: Is it Worth the Money? | by Dario Radečić | Towards Data Science*. (n.d.). Retrieved May 30, 2022, from https://towardsdatascience.com/colab-pro-is-it-worth-the-money-32a1744f42a8

Espejo-Garcia, B., Mylonas, N., Athanasakos, L., Fountas, S., & Vasilakoglou, I. (2020). Towards weeds identification assistance through transfer learning. *Computers and Electronics in Agriculture*, *171*. https://doi.org/10.1016/j.compag.2020.105306

*GitHub - ultralytics/yolov5: YOLOv5 🚀 in PyTorch > ONNX > CoreML > TFLite*. (n.d.). Retrieved May 30, 2022, from https://github.com/ultralytics/yolov5

*Image Segmentation Using Color Spaces in OpenCV + Python – Real Python*. (n.d.). Retrieved May 30, 2022, from https://realpython.com/python-opencv-color-spaces/

Islam, N., Rashid, M. M., Wibowo, S., Xu, C. Y., Morshed, A., Wasimi, S. A., Moore, S., & Rahman, S. M. (2021). Early weed detection using image processing and machine learning techniques in an australian chilli farm. *Agriculture (Switzerland)*, *11*(5). https://doi.org/10.3390/agriculture11050387

Khan, S., Tufail, M., Khan, M. T., Khan, Z. A., & Anwar, S. (2021). Deep learning-based identification system of weeds and crops in strawberry and pea fields for a precision agriculture sprayer. *Precision Agriculture*, *22*(6), 1711–1727. https://doi.org/10.1007/s11119-021-09808-9

Liu, B., & Bruch, R. (2020). Weed Detection for Selective Spraying: a Review. *Current Robotics Reports*, *1*(1), 19–26. https://doi.org/10.1007/s43154-020-00001-w

Liu, X., Song, L., Liu, S., & Zhang, Y. (2021). A review of deep-learning-based medical image segmentation methods. *Sustainability (Switzerland)*, *13*(3), 1–29. https://doi.org/10.3390/su13031224

Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., & Terzopoulos, D. (2020). *Image Segmentation Using Deep Learning: A Survey*. http://arxiv.org/abs/2001.05566

Mohammed, H., Tannouche, A., & Ounejjar, Y. (2022). Weed Detection in Pea Cultivation with the Faster RCNN ResNet 50 Convolutional Neural Network. *Revue d’Intelligence Artificielle*, *36*(1), 13–18. https://doi.org/10.18280/ria.360102

Mythili, C., & Kavitha, V. (2012). Color Image Segmentation using ERKFCM. *International Journal of Computer Applications*, *41*(20), 975–8887.

Nepal, U., & Eslamiat, H. (2022). Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs. *Sensors*, *22*(2). https://doi.org/10.3390/s22020464

Önler, E. (2021). REAL TIME PEST DETECTION USING YOLOv5. *International Journal of Agricultural and Natural Sciences E*, *14*(3), 232–246.

Osorio, K., Puerto, A., Pedraza, C., Jamaica, D., & Rodríguez, L. (2020). A Deep Learning Approach for Weed Detection in Lettuce Crops Using Multispectral Images. *AgriEngineering*, *2*(3), 471–488. https://doi.org/10.3390/agriengineering2030032

Parmar, V., Bhatia, N., Negi, S., & Suri, M. (2020). *Exploration of Optimized Semantic Segmentation Architectures for edge-Deployment on Drones*. http://arxiv.org/abs/2007.02839

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). *You Only Look Once: Unified, Real-Time Object Detection*. http://arxiv.org/abs/1506.02640

Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*. http://arxiv.org/abs/1804.02767

Ronneberger, O., Fischer, P., & Brox, T. (2015). *U-Net: Convolutional Networks for Biomedical Image Segmentation*. http://arxiv.org/abs/1505.04597

Shaw, D. R. (n.d.). *Remote sensing and site-specific weed management*. www.frontiersinecology.org

Soni, J. A., Hetal, D., & Patel, N. (2021). ITEE Journal Deep Learning-Based Weed Identification for Precision Farming. In *International Journal of Information Technology and Electrical Engineering ITEE* (Vol. 10, Issue 3).

Tan, M., Pang, R., & Le, Q. v. (2019). *EfficientDet: Scalable and Efficient Object Detection*. http://arxiv.org/abs/1911.09070

Tannouche, A., Sbai, K., Rahmoune, M., Agounoune, R., & Rahmani, A. (2016). Real time weed detection using a boosted cascade of simple features. *International Journal of Electrical and Computer Engineering*, *6*(6), 2755–2765. https://doi.org/10.11591/ijece.v6i6.11878

Tannouche, A., Sbai, K., Rahmoune, M., Zoubir, A., Agounoune, R., Saadani, R., & Rahmani, A. (2016). A fast and efficient shape descriptor for an advanced weed type classification approach. *International Journal of Electrical and Computer Engineering*, *6*(3), 1168–1175. https://doi.org/10.11591/ijece.v6i3.9978

*Train Custom Data 📌 - YOLOv5 Documentation*. (n.d.). Retrieved May 30, 2022, from https://docs.ultralytics.com/tutorials/train-custom-datasets/

Yan, B., Fan, P., Lei, X., Liu, Z., & Yang, F. (2021). A real-time apple targets detection method for picking robot based on improved YOLOv5. *Remote Sensing*, *13*(9). https://doi.org/10.3390/rs13091619

Yao, J., Qi, J., Zhang, J., Shao, H., Yang, J., & Li, X. (2021). A real-time detection algorithm for kiwifruit defects based on yolov5. *Electronics (Switzerland)*, *10*(14). https://doi.org/10.3390/electronics10141711

Zunair, H., & ben Hamza, A. (2021). Sharp U-Net: Depthwise convolutional network for biomedical image segmentation. *Computers in Biology and Medicine*, *136*. https://doi.org/10.1016/j.compbiomed.2021.104699