Automated identification of lateral plant root emergence points (Computer vision)

Meiqi Yang (meiqi@stanford.edu) & Vivian Zhong (vivzhong@stanford.edu)

I. Introduction

Studies concerning plant characteristics often require analysis of large amounts of images, and often require manual annotation of features of interest. In many cases, this amounts to an object detection task. For our project, we collaborated with Dr. Johannes Scharwies, who is investigating the hydropatterning phenotypes of nearly 300 different strains of corn, each with a different genome (DNA). Hydropatterning refers to how plant root development changes in response to the presence or absence of water in the soil[1], and could play a role in determining a plant's drought response. In particular, our collaborator is interested in identifying areas of corn DNA that influence the extent to which the plant initiates root development in dry vs wet conditions. This requires the tedious task of manually counting lateral (secondary) root emergence points across hundreds of root images, a task which we sought to automate. The input of our algorithm is a plant root image. We then use a YOLOv4-based single-class object detection model to predict bounding boxes for lateral root emergence points. For the purposes of our collaborator's study, the ultimate result of interest is the number of emergence points predicted by the model for a given root image.

II. Related work

Previous neural networks designed to analyze plant root images have all leveraged CNNs with encoder-decoder configurations[2–4]. However, these algorithms were all tailored for segmentation tasks such as tracing root length and counting total number of lateral roots. We are instead interested in identifying the point of lateral root emergence on a given side of the root. Segmentation performs well when there is a clear foreground and background, which is not present in our images. Neither we nor our collaborator were able to identify any pre-existing algorithm suited for this exact task.

YOLOv4[5] is considered the state-of-the-art model for fast and accurate object detection, thus we leveraged it as the base model for our project. Like the aforementioned algorithms, YOLOv4 also leverages convolutional layers for encoding and decoding input images. However, instead of learning to apply accurate segmentation masks, YOLOv4 learns to draw accurate bounding boxes around the objects of interest. YOLOv4 also accommodates a higher resolution input image (412x412) than the segmentation algorithms (256x256), which is useful for accurate detection of objects that are very small, as is the case with our images' root emergence points. YOLOv4 has previously been used to detect apples amongst foliage with a mean average precision of 96.9%[6].

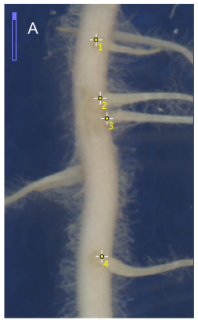
III. Dataset and Features

The initial dataset consisted of 1512 RGB root images with a combined 53097 human-annotated lateral root emergence points (example in Fig. 1A), with each point represented as a xy-coordinate (Fig. 1B). The 1512 images are all paired, with each root being imaged from two sides (the slice parameter in Fig. 1B). Each side represents a different growth environment (wet soil vs dry soil), hence the need to identify emergence points as opposed to simply counting total lateral roots present in an image. Each image has a width of 300 pixels and a variable height (median=5726, min/max=[1329, 7361]) (Fig. 2, Appendix A).

Image processing

The images were scaled by our collaborator such that each pixel in each image represents the same real-life distance. An algorithm[7] was also used to digitally straighten all the roots. YOLOv4 incorporates the Mosaic method of data augmentation, which combines 4 images into one for training, thus improving detection of objects outside their normal context.[5] Since we did not encounter high variance issues in our initial training, no additional data augmentation was performed.

All images were converted from TIFF to JPG, and bounding box labels were generated to fit the darknet format. We automatically generated bounding boxes for the training images by treating the human-annotated emergence point coordinates as centers of uniformly-sized square bounding boxes. For developing our model, we employed a 90/10 train/validation split. All bounding boxes were initially set to width and height of 40 pixels, as determined by sampling measurements using the ImageJ software.



B

	A	B	C	D
1	Position	X	Y	Slice
2	1	190.667	205.333	1
3	1	174.667	274.667	1
4	1	174.667	3344.667	1
5	1	177.333	3359.333	1
6	1	150.667	3854	1

Figure 1. Examples from the corn hydropatterning dataset. A) A corn root with annotated emergence points for the lateral roots that emerged on the imaged side. B) A set of coordinates of annotated emergence points for one side of a root.

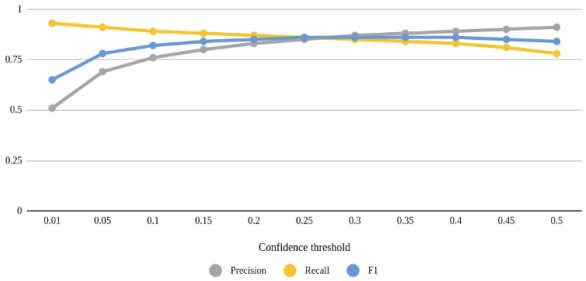


Figure 4. Precision, recall, and F1 scores with different confidence threshold. Generated on model 7 with IOU=0.4

IV. Methods

Our base algorithm was based on the most up-to-date darknet implementation of the YOLOv4 model (Fig. 3[8], Appendix A). Like previous YOLO object detectors, YOLOv4 employs a convolutional neural network to compress the features of the input image (backbone), followed by feature aggregation (neck), and ending with object detection (head). The YOLOv4 model learns to predict coordinates that describe a bounding box around the object of interest (localization) and assign the object a class label (classification). To do so, it trains on input images and their respective ground truth bounding boxes and class labels by optimizing a Complete Intersection over Union (Eq. 1)[6] loss function. Unlike basic IOU (Eq. 2), CIoU captures the distance between non-overlapping ground truth and predicted bounding boxes. A key aspect of YOLO is the use of a non-max suppression (NMS) algorithm to ensure that an object is detected only once: for a given set of overlapping boxes, as determined by a tunable IoU threshold, the algorithm keeps only the predicted box with the highest probability, filtering out any lower probability boxes.

V. Results

Measuring accuracy

The standard measure of accuracy when evaluating YOLO models is mean average precision (mAP, equivalent to AP in this context), generally defined as the area under the precision-recall curve (Eq. 3). Precision and recall are respectively calculated from the number of true positives, false positives, and false negatives (Eq. 4). Whether a predicted box is considered a true positive or false positive (compared to the ground truth bounding box) is determined by a tunable IoU threshold, typically set to 0.5. For deciding whether or not to make a bounding box prediction, we used the default confidence threshold of 0.25 for all our measurements, since in most cases it gave us the best F1 score (Eq. 5), indicating the optimal compromise between precision and recall (Fig. 4).

$$LOSS = 1 - IoU + \frac{\rho^2(b,b^{gt})}{c^2} + \alpha v -$$
$$\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{obj}\left[\hat{C}_i \log(C_i) + \left(1-\hat{C}_i\right)\log\left(1-C_i\right)\right] -$$
$$\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{noobj}\left[\hat{C}_i \log(C_i) + \left(1-\hat{C}_i\right)\log(1-C_i)\right] -$$
$$\sum_{i=0}^{S^2} I_{ij}^{obj}\sum_{c\in classes}\left[\hat{p}_i(c)\log(p_i(c)) + \left(1-\hat{p}_i(c)\right)\log(1-p_i(c))\right]$$

Equation 1

$$IOU = \frac{Area\ of\ Intersection\ of\ two\ boxes}{Area\ of\ Union\ of\ two\ boxes}$$

Equation 2

$$\mathrm{AP} = \int_0^1 p(r)dr$$

Equation 3

$$Precision = \frac{TP}{TP+FP}$$

$$Recall = \frac{TP}{TP+FN}$$

Equation 4

$$F_1 = 2\cdot\frac{\mathrm{precision}\cdot\mathrm{recall}}{\mathrm{precision}+\mathrm{recall}} = \frac{\mathrm{TP}}{\mathrm{TP}+\frac{1}{2}(\mathrm{FP}+\mathrm{FN})}$$

Equation 5

Baseline implementation

We used pre-trained weights (trained on the MS COCO dataset) and the original YOLOv4 architecture. For detection, we used weights from the 6000th iteration, the minimum recommended by the authors of YOLOv4. After that point, the loss tended to trend upward (Fig. 5) in our initial training. We initially followed all recommended defaults for hyperparameters (e.g. batch size, number of iterations, number of CNN filters, training iterations, and input image size) (Appendix B). Learning rate and anchor box dimensions were integrated into the source code and were left untouched. While testing different modifications to our model, we made various adjustments to the training data and hyperparameters, as described below.

Model Number	Model description	Precision	Recall	F1	mAP
1	Baseline	0.74	0.64	0.69	63.09%
2	Grayscale images	0.69	0.58	0.63	55.42%
3	Culled poorly-annotated images from dataset	0.74	0.64	0.69	62.71%
4	NMS IoU threshold = 0.75	0.73	0.64	0.68	62.78%
5	Bounding box = 35*35 pixels	0.67	0.62	0.64	54.58%
6	Input image size = 160*832 pixels	0.81	0.81	0.81	79.2%
7	Input image size = 160*832 pixels, bounding box = 35 pixels	0.78	0.80	0.79	74.81%

Table 1. Baseline model: default YOLOv4 (NMS IoU threshold = 0.45, input image size = 416x416); training bounding box = 40x40; trained on all images without culling. Unless otherwise mentioned, models used baseline settings. All detections were made with default confidence threshold = 0.25, and all accuracy measures were calculated with IoU threshold=0.5. Models *4-7* were all iterations on model *3*.

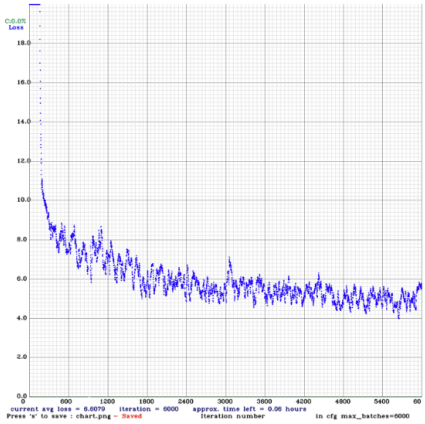


Figure 5. Training loss over 6000 iterations on model *1*.

Training data modification

Grayscale conversion

In our dataset, most of the roots have a yellowish color, whereas some root images have a distinct blue tint (Fig. 6, Appendix A), which might add unnecessary noise and make prediction more difficult. We converted all images to grayscale images in an attempt to eliminate this noise. However, model performance worsened after applying this change (Table 1, model *2*), possibly because the model relies on the color of the emergence points to make predictions, and converting to grayscale resulted in information loss.

Poor-quality data filtering

When training models *1* and *2*, we noticed that the validation accuracy was higher than the training accuracy, which suggested the presence of low quality data in the dataset, which would be more likely to appear in the larger training set than the validation set. Upon examining the dataset, we discovered images where the human-labelled emergence points were completely off the mark (Fig. 7, Appendix A). We manually removed all poorly-annotated images, which reduced the training set from 1512 images to 1491. While this did not improve model performance (Table 1, model *3*), it did result in the training accuracy becoming slightly higher than the validation accuracy, as expected.

Bounding box size

The baseline model yielded a much lower value for recall than precision, which suggested that the model failed to detect more emergence points than it falsely classified (more false negatives than false positives). During error analysis, we discovered that this usually happens when the emergence points are densely clustered together (Fig. 8A, 8B). In this case many of the bounding boxes overlap. Since the model's NMS algorithm removes overlapping bounding boxes, we hypothesized that reducing the bounding box size might help the model retain more bounding boxes when they overlap due to highly proximal emergence points.

Fig. 9 shows a comparison of models trained with 35x35 px bounding box and 40x40 px bounding box. As expected, reducing the bounding box size allows us to detect more overlapping emergence points, but at the cost of

reduced precision, since smaller boxes contain less information and makes it harder to detect emergence points accurately (Table 1, model *5*).

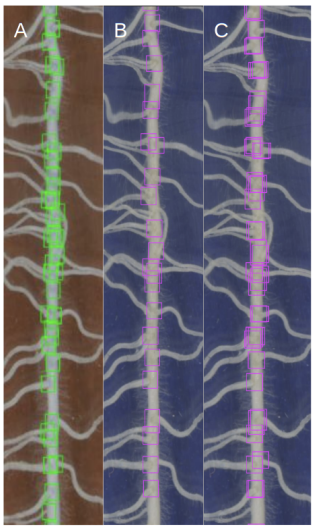


Figure 8. Example validation image GWAS2_137_2.jpg A) Bounding boxes (40x40) generated from human-annotated emergence points; B) model *3* predictions; C) model *5* (higher NMS threshold) predictions.

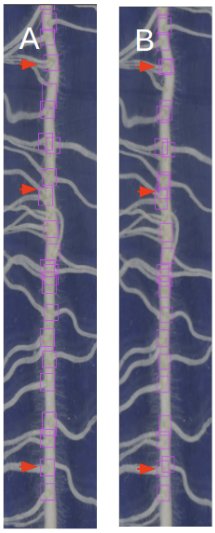


Figure 9. Comparison of predictions on GWAS2_137_2.jpg made by model trained with A) 40px bounding boxes (model *3*) and B) 35px bounding boxes (model *5*) Arrows indicate areas where the two models made different predictions.

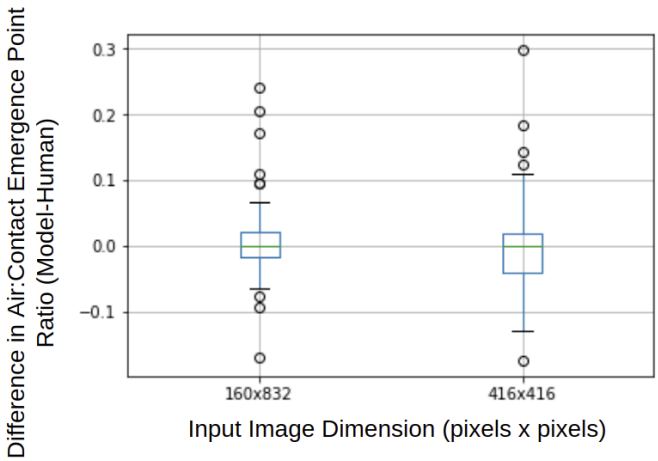


Figure 10. Boxplot comparison of the distribution of difference from human annotations for two models (*6* and *3*) that were identical in every way except in input image dimension. The difference is calculated as the ratio in number of emergence points on the air vs contact sides of a root, as determined from model detection, minus the ground truth ratio as determined from human annotations.

Hyperparameter tuning

Non-max suppression IoU threshold

In addition to changing the training annotation bounding box size, we hypothesized that another way to reduce the likelihood that a legitimate bounding box would be removed by the NMS algorithm would be to increase the IoU threshold for considering two boxes to be duplicates. The default YOLOv4 threshold was 0.45; we increased this to 0.75 across all instances when NMS was called in the detector source code. Recall and mAP increased slightly compared to the baseline for lower IoU thresholds (Table 1, model *4*) while precision decreased slightly or remained unchanged, suggesting a possible reduction in false negatives and/or increase in false positives. Example validation images with model-annotated boxes seem to confirm this (Fig. 8C). However, this is not entirely congruous with the unexpected result that the total number of detected emergence points decreased slightly from 9452 to 9031.

Input image dimension

By default, YOLOv4 resizes images to 412x412 before using it as an input to the convolutional network. Since our images have, on average, an aspect ratio of 0.05:1, we hypothesized that making the aspect ratio less square would reduce the distortion of features and loss of information in the height dimension, thus leading to improved feature learning by the convolutional network. This change could be especially helpful for making the correct NMS calls on overlapping bounding boxes, particularly given the small size of our objects. We trained a model with image input dimensions of 160x832. Since this results in a decrease in number of pixels (160*832=133,120 compared to 412*412=173,056), any improvement in model performance would not be attributable to an increase in input image resolution.

This modification resulted in a remarkable improvement in model performance (Table 1, models *6*, *7*). Model *6* achieved the highest mAP score on the validation set of all the models we tested. In fact, it achieved greater accuracy

(mAP@0.5=79.2%) than the original YOLOv4 did on MS COCO using a higher input image resolution of 608x608 (mAP@0.5=65.7%)[5].

Input Image Dimension, Bounding Box Dimension *(Model)*	160x832, 35x35 *(7)*	416x416, 35x35 *(5)*	160x832, 40x40 *(6)*	416x416, 40x40 *(3)*
p-value (Wilcoxon signed-rank test, two-sided)	0.44	1.16e-7	0.97	3.70e-9

Table 2. Comparison of p-values for two-sided Wilcoxon signed-rank tests between models trained with different combinations of input image dimension (hyperparameter) and bounding box dimensions (dataset modification). The null hypothesis is that the emergence point counts for the paired root images (air and contact sides) as determined by the model and by the original human come from the same distribution. The lower the p-value, the lower the match between the two distributions.

Evaluation of model in the research study context

The original research motivation for developing this object detector was to count the number of lateral roots that emerged from either side of the primary root, in order to determine the degree to which emergence occurred preferentially on the "contact" side (in contact with moisture) as compared to the "air" side (exposed only to dry air). To gauge whether or not our best models were accurate enough for the purposes of the study, we used the model's detection results to calculate the number of emergence points on either side of a given root. We then used a Wilcoxon signed-rank test[9] as a proxy for gauging how closely the model's predictions resembled the human annotations. As shown in Table 2, the distribution of point counts for each root image predicted by the two models employing non-square input image dimensions (*6*, *7*) were not significantly different from the corresponding ground truth point counts. Meanwhile, the difference between points counts predicted by the two models (*3*, *5*) employing square input image dimensions and the ground truth points counts would be considered significant for any commonly-used p-value threshold.

The ultimate metric that our collaborator requires for his research is the ratio of emergence points on the air side versus on the contact side ("air:contact ratio"). Fig. 10 shows that the distribution of error (difference) for this metric when determined by model *6* is more centered and tightly clustered around 0 as compared to model *3*. A better metric of error is taking the absolute value of the difference and dividing by the human-annotated value; however, due to division by 0, the distribution is difficult to show. The average of this error metric was lower for model *6* (0.24) than for model *3* (0.35), as expected.

VI. Conclusion and Future Work

Setting the dimension of the image that is fed into the model's convolutional network to an aspect ratio more similar to the dataset image dimensions resulted in the clearest improvement in model performance, beating the basic YOLOv4 model's MS COCO performance. Decreasing the size of the bounding box in the training dataset annotations and increasing the IoU threshold for the NMS algorithm decreased the number of false negatives at the cost of increased false positives; further tuning of these attributes could result in a more significant increase in model accuracy.

One almost guaranteed improvement that can be made would involve manually re-annotating the dataset images, since we noticed many instances of unlabelled emergence points (Fig. 11, Appendix A) that likely provided conflicting information to the model. Another easy improvement would involve increasing the input image resolution (requires longer training time), which should be particularly beneficial since our objects are very small (<1% of the average image area). Furthermore, since using an input image aspect ratio with greater similarity to dataset image aspect ratio seems to improve model performance, we can try splitting all of our images into equally-sized chunks and preserve the chunk aspect ratio for the input image, thus avoiding any distortion. Finally, an updated version of YOLOv4,

Our collaborator will be generating another batch of root images in the future, which will require the same emergence point annotation as the dataset we worked with here. Since YOLO algorithms have been shown to perform well when detecting the same object in a different context[10], we expect the model developed here to retain its accuracy.

Contributions

Both team members contributed to all aspects of the project.

References

1. Robbins, N. E. & Dinneny, J. R. Growth is required for perception of water availability to pattern root branches in plants. *Proc. Natl. Acad. Sci.* 115, E822–E831 (2018).
2. Yasrab, R. *et al.* RootNav 2.0: Deep learning for automatic navigation of complex plant root architectures. *GigaScience* 8, giz123 (2019).
3. Gaggion, N. *et al. ChronoRoot: High-throughput phenotyping by deep segmentation networks reveals novel temporal parameters of plant root system architecture*. http://biorxiv.org/lookup/doi/10.1101/2020.10.27.350553 (2020) doi:10.1101/2020.10.27.350553.
4. Falk, K. G. *et al.* Computer vision and machine learning enabled soybean root phenotyping pipeline. *Plant Methods* 16, 5 (2020).
5. Bochkovskiy, A., Wang, C.-Y. & Liao, H.-Y. M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *ArXiv200410934 Cs Eess* (2020).
6. Wu, L., Ma, J., Zhao, Y. & Liu, H. Apple Detection in Complex Scene Using the Improved YOLOv4 Model. *Agronomy* 11, 476 (2021).
7. Lobet, G., Pagès, L. & Draye, X. A Novel Image-Analysis Toolbox Enabling Quantitative Analysis of Root System Architecture1[W][OA]. *Plant Physiol.* 157, 29–39 (2011).
8. Lyu, J. *et al.* Extracting the Tailings Ponds from High Spatial Resolution Remote Sensing Images by Integrating a Deep Learning-Based Model. *Remote Sens.* 13, 743 (2021).
9. Wilcoxon signed-rank test. *Wikipedia* (2021).
10. Science, O.-O. D. Overview of the YOLO Object Detection Algorithm. *Medium* https://medium.com/@ODSC/overview-of-the-yolo-object-detection-algorithm-7b52a745d3e0 (2018).

Appendix

A. Figures

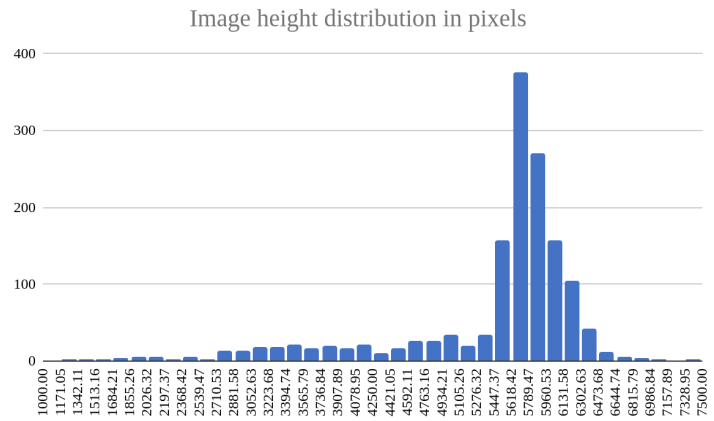


Figure 2. Image height distribution.
Mean = 5429, min = 1329, max = 7136, median = 5726.

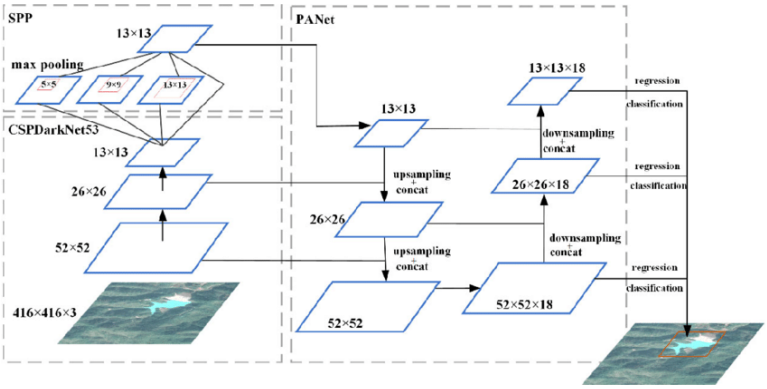


Figure 3. Architecture of YOLOv4

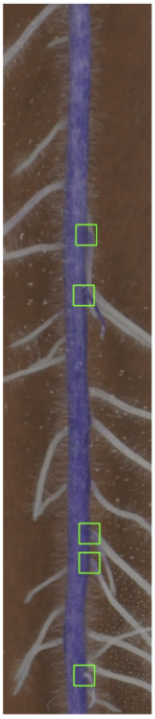


Figure 6. An example of root with blue tint.

Figure 7. An example of a training image with incorrect human annotations.

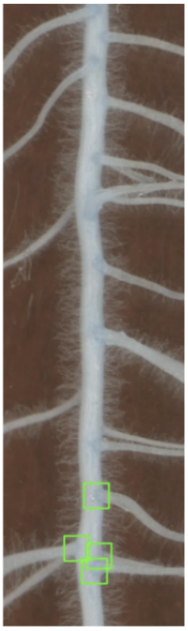


Figure 11. An example of unlabelled emergence points.

B. Initial training configurations modified in cfg/yolov4-custom.cfg file:
   - height = 416
   - width = 416
   - max_batches = 6000
   - steps = 4800, 5400 - (80% of max_batches), (90% of max_batches)
   - filters = 18 - (# of classes + 5) * 3
   - classes = 1