[Team 12] Plant Phenotyping

1st Jessica Mele (jmele) 2nd Krishnachaitanya Pullakandam (kpullak) 3rd Derek Martin (dmartin7)

***Abstract*—Identification of plant growth stages typically comes from physical indicators such as the number of leaves, number of collars, width of the plant, angles of the leaves or the height of the plant. The leaf-collar method, a popularly used metric among researchers using these specimen, considers plant growth in terms of stages classified by the number of leaves with fully developed collars. The goal of this project is to automate the detection of leaves and collars in images of plants so that researchers can easily classify and keep track of the plants' growth stages over time using image data. Our approach is to use the Faster R-CNN model for key-point detection of the leaves and collars. The model is comprised of a fully convolutional network (resnet 50) merged with an Region Proposal Network (RPN) which together work to identify features unique to each class. Trained on a (Tesla P100-PCIE-16GB) GPU, over 50 epochs our model achieved the below metrics: IoU - 0.70 (for leaf) and 0.66 (for collar), theta angle of intersection - (-)0.08 (for leaf) and 0.05 (for collar) and dice coefficient of 0.81 (for leaf) and 0.78 (for collar).**

***Index Terms*—plant, phenotyping, Faster R-CNN, keypoint detection**

I. Model Training and Selection

Given that the task at hand is to automate the detection of two object classes, leaves and collars, in images of plants via keypoint detection we took the approach of considering a handful well-known object detection models before selecting our final model. Object detecting in images utilizes models that can typically be classified into two groups: one-stage detectors and two-stage detectors [1]. The trade-off between the two is that two-stage detectors have high accuracy and one-stage detectors achieve high speeds. When considering recent advances in the field of two-stage detectors we decided that it would be best to aim for high accuracy and proceed this way.

When looking at well-known two-stage detectors such as R-CNN, Fast R-CNN, Faster R-CNN and Mask R-CNN we decided to go with Faster R-CNN, which is an improvement of the first two. Mask R-CNN is an extension of Faster R-CNN mainly used for instance segmentation [1]. Based on experimental data, Faster R-CNN models were able to achieve higher mAP values and lower computation times than Fast R-CNN when trained on the same VGG backbone [1]. A more complete description of the Faster R-CNN framework can be seen in the following subsection.

To implement the Faster R-CNN we utilized Python toolboxes Keras [2] and TensorFlow [3]. From Keras we used the Keras implementation of Faster R-CNN (keras_frcnn) and other modules such as Optimizers and Models. The model was trained on a Tesla P100-PCIE-16GB GPU.

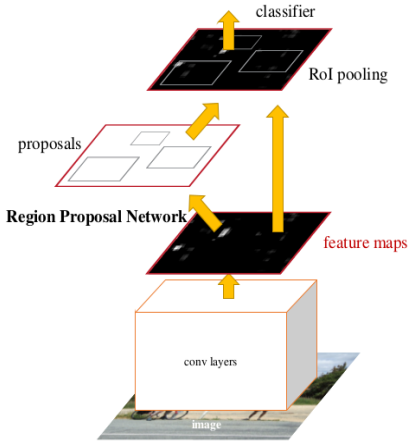

Fig. 1. **(Network Architecture)** A single unified network of a merged CNN with an RPN. [4]

A. The Model

The Faster R-CNN model consists of two modules: a fully convolutional network (resnet50/vgg) and a Regional Proposal Network (RPN) which is illustrated in Fig. 1. The main idea is that the region proposal network proposes regions for the convolutional layer to consider when extracting features for object detection [4]. The region proposal network itself is a fully convolutional network and can be trained end to end with backpropogation and stochastic gradient descent in efforts to improve the generated proposal regions [4]. Using the RPN rather than the selective search method found in Fast R-CNN accelerates the speed of region proposals by sharing its convolutional features with the detection network [1]. The unification of the two moduels happens via an ROI pooling layer and during training where the model alternates between fine tuning for object detection and region proposals [4]. The Keras implementation of Faster R-CNN utilizing the tensorflow backend performs a resize on the pooling region rather than a max pooling. According to Keras, this has little impact on the resulting bounding box predictions, but is more efficient [2]. The predictions from the model are presented as bounding boxes with corresponding objectness scores, which indicate the probability that the bounding box is actually representative of the class it is labeled as.

Our final model was a Faster R-CNN (resnet50) using fixed window sizes for the annotations for training and pre-processing the testing images to improve our training and testing accuracy. The final RPN module had a total of 145 layers and the CNN module had a total of 180 layers. We used

400 region of interests (ROIs) with a bounding box threshold (bbox threshold) of 0.25, and we trained it over 50 epochs.

B. Baseline

Our baseline model was a Faster R-CNN with no modifications to the images or annotations. For our first experiment we used 400 ROIs with a bbox threshold of 0.8, and trained it over 50 epochs. This model returned no keypoint prediction for collars or leaves. As a result, we lowered the bbox threshold to 0.5, and our model was then able to identify only collars. We continued to change the bbox threshold to see what value would improve our model enough to identify both leaves and collars, but we could not find such a value. This led us to attempting data augmentation and annotation augmentation.

II. Experimental Section

We experimented with two types of data augmentations to improve our model: image/data augmentation and annotation/bounding box augmentation. We augmented our images because we noticed that our model was predicting/labelling all of the test images that were in a tinted rooms. It did not make a single prediction on rooms/images that were not tinted a different color. We decided to test if adding an artificial hue to images that were not already in a tinted room would improve our models ability to count the leaves and collars, and we found that it did make a significant difference. Another image augmentation method we tested was standardizing the image size. Half the images were 1920x1080 and the other half were 3280x2464, so we thought this was potentially a reason our model had trouble identifying leaves and collars in our test set. We had two hypothesis for scaling all the images to the size of 640x480: 1. if all the images have the same size, training will be more consistent, and 2. having smaller images may help improve the accuracy and predictions. We found that standardizing and shrinking the images did not produce better testing results, but it did speed up training and had more consistent training accuracy.

Our other observation and experiment was with the annotations for all the images. We noticed that there was a significant difference in terms of height and width of the bounding boxes for each annotation in all training images; some images have really big annotations and some have really small annotations. To remedy this, we decided to scale all the annotations to a fixed size as we believed this would improve our model accuracy and hence the ability to count leaves and collars in test images. Our solution for this was to take the maximum width and maximum height of all the annotations, and scale all the annotations to the same size. Our other modification to the annotations is to move the horizontally translate the bounding boxes so that the leaf tips and collars would be located closer to the center of the boxes. Originally we were using the leaf tip key point as a corner of the bounding box and as a result with no modifications, leaf tips would be located closer to corners of our bounding boxes. We believe this to be another reason why our model initially had trouble identifying and counting leaves and collars or mislabelled them completely meaning it would label part of the background as a leaf or collar.

A. Metrics

To evaluate the performance of the model we used standard object detection metrics and a keypoint metric to determine the performance of the model in terms of keypoints. To evaluate object detection perfomance we utilized Intersection over Union (IoU), Dice Coefficient, and the numbers of collars and leaves predicted with respect to the actual number of each class in the ground truth labels, which we represent as an accuracy value.

The IoU score is computed as the intersection over the union of the predicted bounding box with the corresponding actual bounding box. According to rules of the 2012 Pascal VOC Challenge, a predicted bounding box is considered a true positive if it has an IoU score with an actual label of 50% or more [5]. If two or more boxes surpass this 0.5 threshold then the prediction with the highest objectness score is taken as the true prediction. We used rule similar to this, with a slightly smaller threshold of 0.45. That is, if a predicted bounding box had an IoU score of 0.45 or more, and had the highest objectness score if this held for more than one prediction, with a true bounding box then it was considered a true positive. The others are then considered false positives. To present the results we considered the average IoU for both classes. Typically, an IoU score between 0.4 and 0.8 is indicative of a good model, above 0.8 is an excellent model and below 0.4 is a poor model. For our final model the average IoU scores for leaves and collars were 0.699 and 0.659 respectively.

Similar to the IoU score, the Dice Coefficient is two times the ratio of the overlapping bounding boxes with the sum of all of the pixels in both bounding boxes. A dice coefficient closer to 1 represents a better performing model whereas the coefficient closer to 0 represents a poor performing model. For our best performing model, the average dice coefficient for leaves and collars were 0.813 and 0.786 respectively.

We also felt it was important to show the accuracy of the model in terms of the numbers of leaves and collars predicted for each image in comparison with the true numbers of each in a given image. To compute this accuracy score we considered the ratio of the count of each class as predicted by the model versus the count of the respective class from the ground truth objects. The final model average accuracy scores for leaves and collars were 0.758 and 2.150 respectively. We will explain what these values mean more in depth under the performance section of the report.

The final metric used to evaluate model performance was in terms of keypoints. Since we were originally given labels in terms of keypoints we wanted to have a way to describe the performance in terms of predicted keypoints. Taking the diagonal of the predicted bounding box as our predicted keypoint for each leaf and collar we decided to quantify the results in terms of orientation. This is especially important when considering the leaf class. The leaves are all oriented in different manners depending on how developed the plant is.

Here, we compute the angle between the predicted keypoint and the actual keypoint in order to evaluate the orientation of our keypoints. A smaller angle of intersection implies that the keypoints were oriented in similar manners, which is what we are aiming for, and if the value is high that implies that the predicted keypoints are close to perpendicular to the actual keypoints. For our final model, the average angle of intersection for leaves and collars were -0.0851 and 0.0523 respectively.

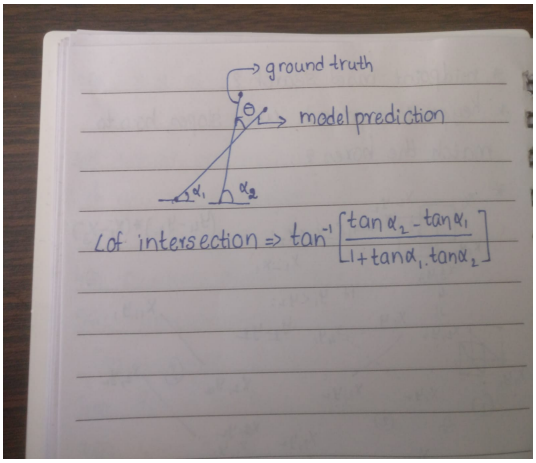

Fig. 2. Tangent Formula

B. Pre-Processing

List of all pre-processing steps that we have done:
a) Segment an object from an image based on color:

We found that by altering the pixel values in the image segments more towards red (255) the model accuracy improved
b) Image Smoothing using OpenCV Gaussian Blur:

As in any other signals, images also can contain different types of noise, especially because of the source (camera sensor). Image Smoothing techniques help in reducing the noise. Using this techniques, sharp edges in images are smoothed while minimizing too much blurring.Gaussian blur achieved using OpenCV: cv2.GaussianBlur(src,(5,5), cv2.BORDER_DEFAULT)

C. Model Selection

To perform model selection we first spent time evaluating the bounding box predictions for the baseline model and then when adjusting specific parameters. The final model we selected was trained over 50 epochs, with a bbox threshold of 0.25, fixed annotation bounding box sizes, and image pre-processing of introducing a pinkish hue to the images. As you can see from Fig. 4 and Fig. 5 there were significant improvements made to the model after introducing the image pre-processing steps, adjusting the bbox threshold and centering the tips of the leaves and collars to that they are more contained within the bounding boxes during training. When considering which model should be our final model

TABLE I
PERFORMANCE EVALUATION BETWEEN BASELINE MODEL AND FINAL MODEL

	Model 1		Final Model	
	Leaves	Collars	Leaves	Collars
Average IoU	0	0.618	0.700	0.659
Average Theta	0	0.102	-0.085	0.0527
Average Dice	0	0.758	0.813	0.786
Average Accuracy	0	1.705	0.758	2.150

choice we considered the above metrics, which can be seen in Table I. As you can see from the numbers in this table, we were able to obtain higher average IoU scores for both classes in the final model, as well as smaller Theta values, higher dice coefficients and better leaf accuracy in our final model when compared to our first model, which was only predicting collars. Now we will discuss the accuracy metrics and what this means for the model. Recall our accuracy metric is a ratio of the number of predicted objects in a specific class versus the number of actual objects in the image as determined by our ground truth labels. It is important to note that with our rule of considering a prediction a true positive if it had an IoU score of 0.45 or higher with a true label box that we were able to identify a match for each label in our ground truth. While this is great news, we still felt it was necessary to report this accuracy ratio to showcase one of the downfalls of the model. As you can see from the Table Ithe final model has an average accuracy of 2.150 for collars. This implies that there were more predicted collars than actual collars, indicating that we have a high false positive value for collar prediction for this model. Now, in comparison to the first model that was only predicting collars this ratio increased, but all of the other metrics improved significantly, which is why we chose this to be our final model.

Fig. 3. **(Model 1)** Modification of baseline model of adjusting bbox threshold from 0.8 to 0.5. Here there are no laves detected and collars are bounded by red boxes. Each prediction has an objectness score which is the probability of the box actually being the respective label.

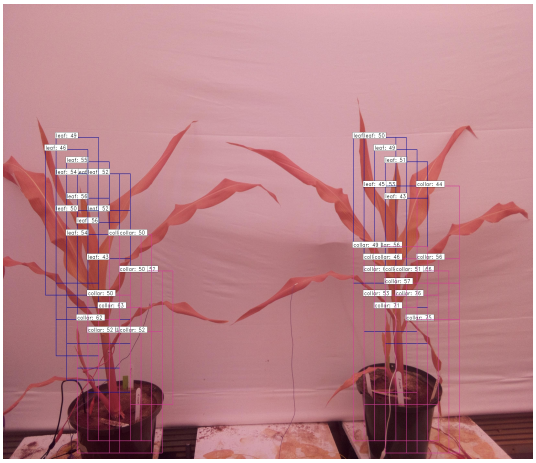

Fig. 4. **(Model 2)** Modification of model by adjusting bbox threshold from 0.5 to 0.25 and switching to fixed annotation bounding box sizes.Here detected leaves are bounded by blue boxes and collars with pink boxes. Each prediction has an objectness score which is the probability of the box actually being the respective label.

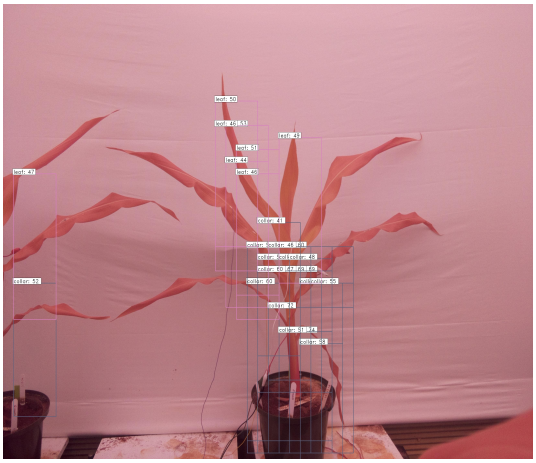

Fig. 5. **(Model 3 - Final)** Modification of model to 0.25 bbox threshold, fixed annotation bounding box sizes, and image pre-processing. Here detected leaves are bounded by pink boxes and collars with grey boxes. Each prediction has an objectness score which is the probability of the box actually being the respective label.

D. Performance and Comparison with Baseline

In comparison with our baseline model our final model had a significant improvement in terms of evaluation metrics. Our baseline was not able to predict a single leaf or collar in any of the test set images so the metrics obtained from that model are meaningless in the context of this task. To show the progression of the model with our improvements we have decided to show the first meaningful set of metrics with that of the final model in Table I. As you can see, the metrics indicate that we have a good working model for the detection of leaves and collars in the plant image data set. In terms of key point predictions our average theta values for both classes are relatively low and close to zero. That means for both classes our key point predictions shared similar orientations with those of the true labels. Especially for leaves, this metrics is an important indicator of a good prediction. Considering the average IoU and Dice coefficients for both classes we can say that the model is a decent fit here as well. Recall that an IoU between 0.4 and 0.8 indicates a good fit and our values fall within those bounds for both classes. Similarly, the dice coefficients are both close to 1, which is a positive metric for this model. Again, we can note that the model is predicting too many collars and that we have a lot of false positives within this object class. In terms of performance, this model performs reasonably well for the given task. There are a few additional things we would consider exploring to improve metrics further. For instance, there are a few images where there is a human subject included. It would be beneficial to the model and improve accuracy if we could perform more pre-processing and attempt to blur anything that is not a plant. This could help clear up confusion and allow the model to more accurately detect leaves. Additionally, we believe there was a drawback with the data set. We were not given a lot of images and so even with a 90/10 split of training and testing data we were not able to train the model as well as we would have liked. Given more images we could have exposed the model to more complex images throughout training. Another note on the data set are the annotations. We noticed that some images had multiple sets of annotations which created a problem with training as well. For example, some images had labels indicating more leaves and collars then were actually present in the image. This could have confused the model and also led to our false positives being so high. If we were to continue with this data set we would spend more time thoroughly cleaning the data and ensuring proper annotations to improve performance overall.

REFERENCES

[1] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A survey of deep learning-based object detection," *CoRR*, vol. abs/1907.09408, 2019. [Online]. Available: http://arxiv.org/abs/1907.09408

[2] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[4] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[5] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[6] P. Sharma. A practical implementation of the faster r-cnn algorithm for object detection.

[7] R. Nielson. (2019) Determining corn leaf stages. [Online]. Available: https://www.agry.purdue.edu/ext/corn/news/timeless/VStageMethods.html