CAIR Lab

Instance Segmentation Model Training

By James Chun

The scope of the project was the development of an instance segmentation model for the use on hemp plants. The model will be able to identify each leaf as an instance. The base model used was a pre-trained fasterRCNN model on the MSCoco dataset. Using this, additional training and testing was performed on a custom data set to fine tune the model. Since a data set involving hemp plants was not available, the CVPPP data set was used instead. Specifically, the CVPPP 2017 data was used in training and validation, while the CVPPP 2015 data was used in testing. There was no further process for the training and validation sets, both were simply the entire CVPPP 2017 data. As a result of this, it should be noted that the models trained are not effective for actual hemp plants, and so the goal was shifted to simply develop a process which could be easily adjusted to train a model given the hemp plant data in the future.

For training, experiments were conducted to produce a model that yielded the best results, that being, the greatest ability to identify leaves and distinguish them from dense clusters of leaves. Experiments consisted of utilizing different PyTorch resources, such as different optimizers as well as learning rate schedulers (or whether one is needed at all). The training loss and validation results were then recorded on a tensorboard; additionally, at every 5 epochs of training, the model was saved.

The two main optimizers used were the SGD and Adam optimizers included with pytorch. The stochastic gradient descent (SGD) method runs through batches of the training data, rather than all of them at once, and adjusts model parameters accordingly. The first-order gradient-based optimization method (Adam) would run through all of the training data then adjust parameters. These two were used as they were the two simplest forms of optimizations. The other optimizers offered by pytorch are variants of these optimizers.

Different combinations of optimizers and schedulers were used and each combination was trained on multiple initial learning rates. Learning rates were determined either linearly, starting from 0.001 and incrementing by 0.0005, or logarithmically, starting from 0.001 and decreasing by a factor of 10. The results outlined below were the best performing models.

At first, the SGD optimizer was used with StepLR learning rate scheduling, which produced the following [results](https://drive.google.com/drive/folders/1evPYiJIJEzMBXXpNhHONeOL08OrmQQhv?usp=sharing) which are stored via tensorboard.With a learning rate of 0.0015, trained on 100 epochs, the lowest training loss recorded was 0.1996, with the following testing results:

Average Precision

IoU=0.50:0.95 | area=all | maxDets=100 : 0.7858509711695147

IoU=0.50 | area=all | maxDets=100 : 0.9688094852280265

IoU=0.70 | area=all | maxDets=100 : 0.9000939401701391

IoU=0.50:0.95 | area=small | maxDets=100 : 0.6067797541723786

IoU=0.50:0.95 | area=medium | maxDets=100 : 0.8065578060019372

IoU=0.50:0.95 | area=large | maxDets=100 : 0.8730333351229771

Average Recall

IoU=0.50:0.95 | area=all | maxDets= 1 : 0.055268199233716474

IoU=0.50:0.95 | area=all | maxDets=10 : 0.5388409961685824

IoU=0.50:0.95 | area=all | maxDets=100 : 0.8102011494252872

IoU=0.50:0.95 | area=small | maxDets=100 : 0.6654040404040404

IoU=0.50:0.95 | area=medium | maxDets=100 : 0.8332549019607842

IoU=0.50:0.95 | area=large | maxDets=100 : 0.876959619952494

Then the Adam optimizer was used, without a scheduler, which produced the (answer: empirically we showed that these models worked the best) following [results](https://drive.google.com/drive/folders/1TS6wHxBane1sn8PW5b-3pA0kYJzSXS3-?usp=sharing). With a learning rate of 0.0001, trained on 150 epochs, the lowest recorded training loss was 0.1667, with the following testing results:

Average Precision

IoU=0.50:0.95 | area=all | maxDets=100 : 0.8247514939484711

IoU=0.50 | area=all | maxDets=100 : 0.9694053759133986

IoU=0.70 | area=all | maxDets=100 : 0.9251917842414816

IoU=0.50:0.95 | area=small | maxDets=100 : 0.672873728126411

IoU=0.50:0.95 | area=medium | maxDets=100 : 0.8506231684880835

IoU=0.50:0.95 | area=large | maxDets=100 : 0.9023956548639323

Average Recall

IoU=0.50:0.95 | area=all | maxDets= 1 : 0.057183908045977005

IoU=0.50:0.95 | area=all | maxDets=10 : 0.5579022988505747

IoU=0.50:0.95 | area=all | maxDets=100 : 0.8519157088122606

IoU=0.50:0.95 | area=small | maxDets=100 : 0.7101010101010101

IoU=0.50:0.95 | area=medium | maxDets=100 : 0.8773333333333335

IoU=0.50:0.95 | area=large | maxDets=100 : 0.9087885985748219

These results indicate the performance of the model and the correctness of the inferences it generates when given the test data. The IoU, area, and maxDets variables determine the strictness in the evaluation, and the evaluations range from 0-1 with 1 being the best results. IoU specifies the overlap between the ground truth and the prediction bounding boxes.