Implementation of Faster R-CNN

Faster R-CNN is an object detection model that improves Fast R-CNN by utilizing a region proposal network (RPN) with the CNN model. The RPN shares full-image convolutional features with the detection network, enabling nearly cost-free region proposals. It is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. The RPN is trained end-to-end to generate high-quality region proposals, which Fast R-CNN uses for detection. RPN and Fast R-CNN are merged into a single network by sharing their convolutional features: the RPN component tells the unified network where to look. As a whole, Faster R-CNN consists of two modules. The first module is a deep fully convolutional network that proposes regions, and the second module is the Fast R-CNN detector that uses the proposed regions.

Detectron 2 is a next-generation open-source object detection system from Facebook AI Research. It is used and trains the various state-of-the-art models as **Faster-RCNN** from the model zoo for detection tasks such as bounding-box detection, instance and semantic segmentation, and person keypoint detection.

In releasing Detectron2, the Facebook Artificial Intelligence Research team also released a model zoo. The Detectron2 model zoo includes pre-trained models for a variety of tasks: object detection, semantic segmentation, and keypoint detection

COCO Object Detection Baselines

Faster R-CNN:

Name	lr sched	train time (s/iter)	inference time (s/im)	train mem (GB)	box AP	model id	download
R50-C4	1x	0.551	0.102	4.8	35.7	137257644	model | metrics
R50-DC5	1x	0.380	0.068	5.0	37.3	137847829	model | metrics
R50-FPN	1x	0.210	0.038	3.0	37.9	137257794	model | metrics
R50-C4	3x	0.543	0.104	4.8	38.4	137849393	model | metrics
R50-DC5	3x	0.378	0.070	5.0	39.0	137849425	model | metrics
R50-FPN	3x	0.209	0.038	3.0	40.2	137849458	model | metrics
R101-C4	3x	0.619	0.139	5.9	41.1	138204752	model | metrics
R101-DC5	3x	0.452	0.086	6.1	40.6	138204841	model | metrics
R101-FPN	3x	0.286	0.051	4.1	42.0	137851257	model | metrics
X101-FPN	3x	0.638	0.098	6.7	43.0	139173657	model | metrics

Fig: 1 **Object detection models in the Detectron2 model zoo.**

We used a specific Faster-RCNpre-traineded model (COCO-Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml) from the model zoo the properties of this model as shown in Fig: 2 to train our own custom dataset

Steps of implementation

- **install the Detectron2 library**
- **we Downloaded our own custom object detection dataset that was preprocessed As we mentioned**
- **Register dataset (i.e., tell detectron2 how to obtain our own dataset).**
- **register metadata for the dataset**
- **Visualize Detectron2 training data (optional)**

Detectron2 makes it easy to view our training data to make sure the data has been imported correctly.

Samples of training data Visualization

Fig 2 training image

Fig 3 training image

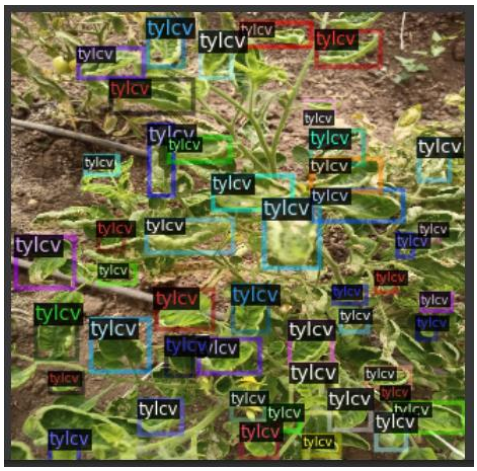

Fig 1 training ng image

❖ **Pre-train model in the model zoo of detectron2 for training.**

Faster-RCNN pre-trained model (COCO- Detection /faster_rcnn _X_101_32x8d _FPN_3x.yaml) with a learning rate of 0.001, a number of iterations are 800 epochs and batch size is 64 Model achieved an average precision 62.557

visualization of the model graph

TensorBoard allows tracking and visualizing metrics such as loss and accuracy, visualizing the model graph

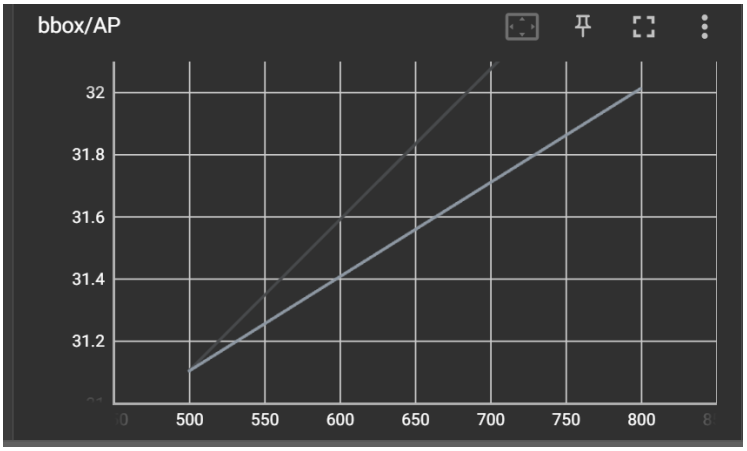

Figure 4 average precision

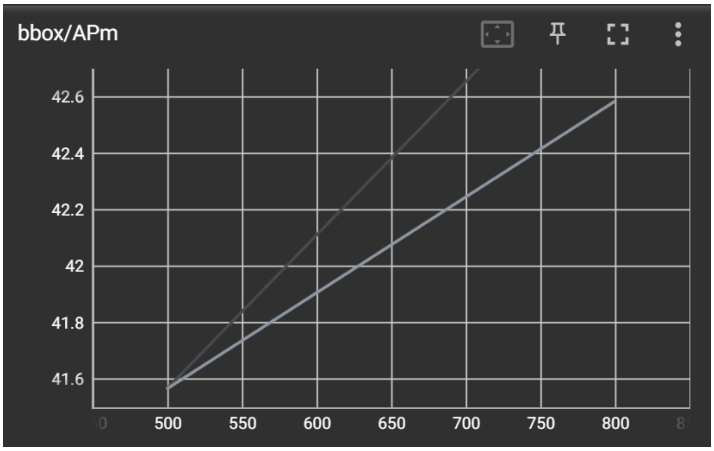

Figure 3 mean average precision

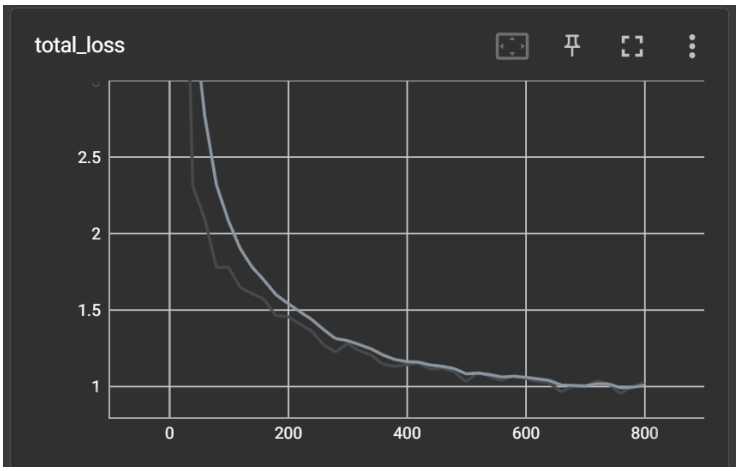

Figure 1 total loss

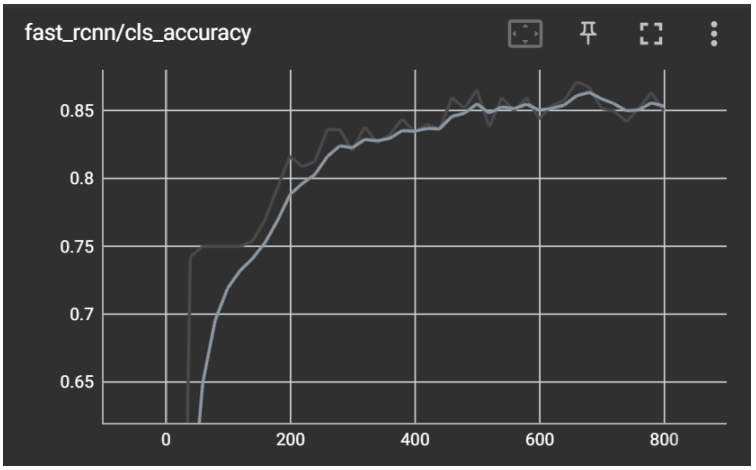

Figure 2 accuracy

❖ Test Evaluation

Evaluation is done on a Test set that model has not seen and achieved average precision of 61.386

Samples of the test set that evaluated

Figure 6 test image that is evaluated

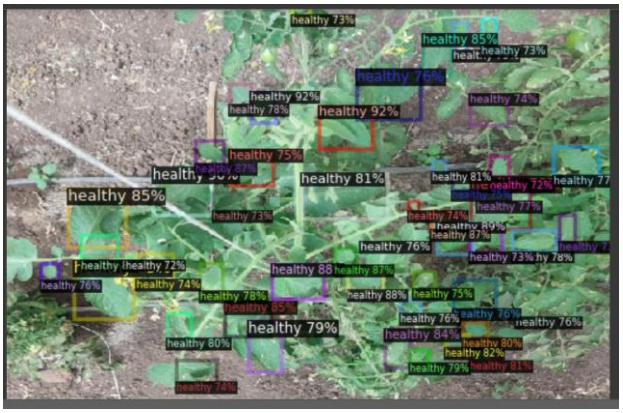

Figure 5 test image that is evaluated

Figure 8 test image that is evaluated

Figure 7 test image that is evaluated

❖ Run Prediction :

And finally, we can run our new custom Faster R-CNN model on real images.

Note, these are images that the model has never seen (Test Set)

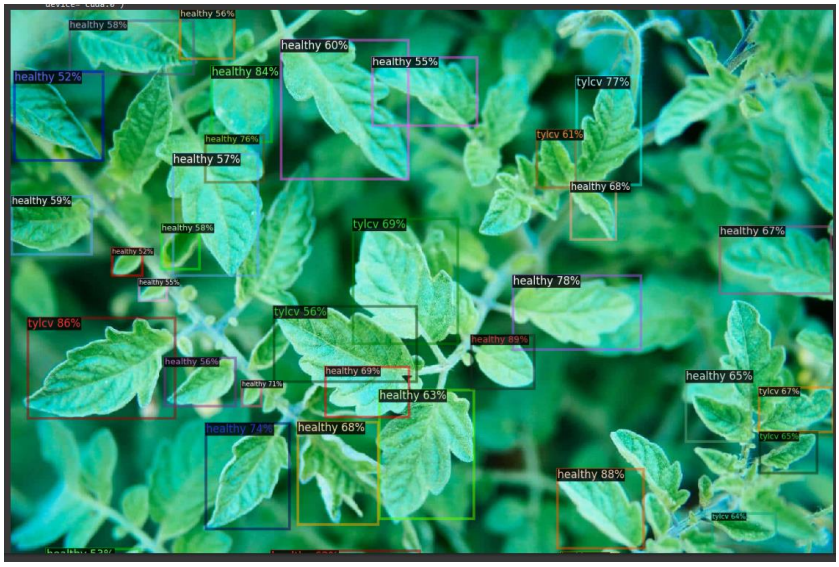

Figure 10 predicted image

Figure 9 predicted image

Source code

https://colab.research.google.com/drive/1GQRcAX5mYHbrkRzfD7dtLxmCc0avC5t-#scrollTo=FgjUxtCy5MfD