TRANTOR HIRING CHALLENGE

Task:

To train an object detection neural network and convert it to ONNX.

Data:

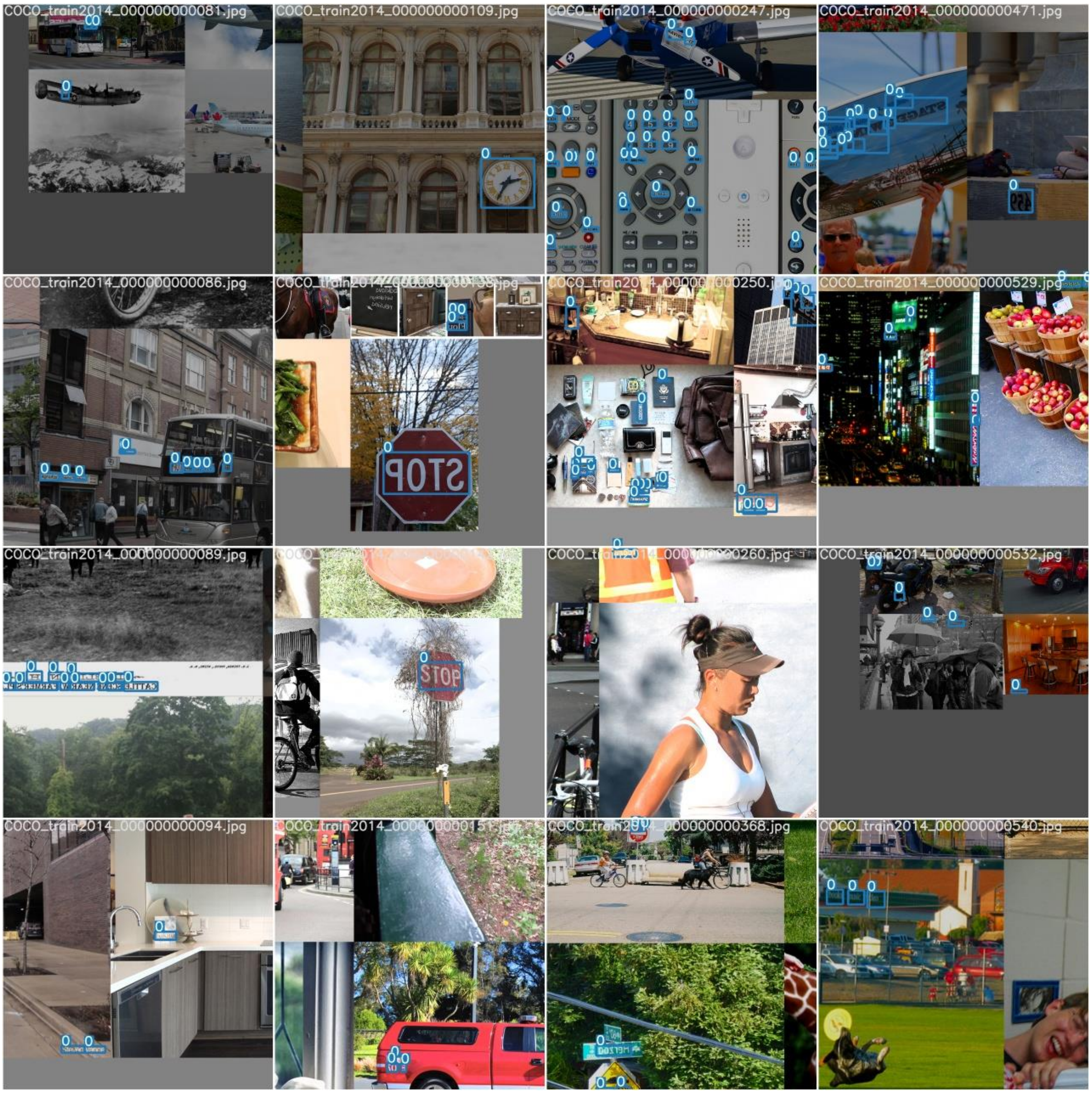
Given the three options, I went with the Coco-Text Dataset. The Coco-Text dataset is contains 63,686 images with 239,506 annotated text instances. Each annotation is labelled with three attributes: machine-printed vs. handwritten, legible vs. illegible, and English vs. non-English.

43,686 of these images were part of the train set and the test and validation set had 1000 images each.

To simplify the problem, I only select images with annotations marked as machine-printed, legible and English.

This took the number of images to 14324 images in the training set and 3346 in the validation set.

Here's a batch of images from the training set:


COCO_train2014_000000000081.jpg
COCO_train2014_000000000109.jpg
COCO_train2014_000000000247.jpg
COCO_train2014_000000000471.jpg
COCO_train2014_000000000086.jpg
COCO_train2014_000000000250.jpg
COCO_train2014_000000000529.jpg
COCO_train2014_000000000089.jpg
COCO_train2014_000000000260.jpg
COCO_train2014_000000000532.jpg
COCO_train2014_000000000094.jpg
COCO_train2014_000000000151.jpg
COCO_train2014_000000000368.jpg
COCO_train2014_000000000540.jpg
STOP

Solution:

Model Selection:

It is very hard to have a fair comparison among different object detector algortihms. There is no straight answer on which model is the best. We make choices by trading off between accuracy and speed. The most commonly available object detection algorithms can be broadly divided into two categories:

1. Region-based Convolution Neural Networks with two shot detections

3. Single Shot Detectors

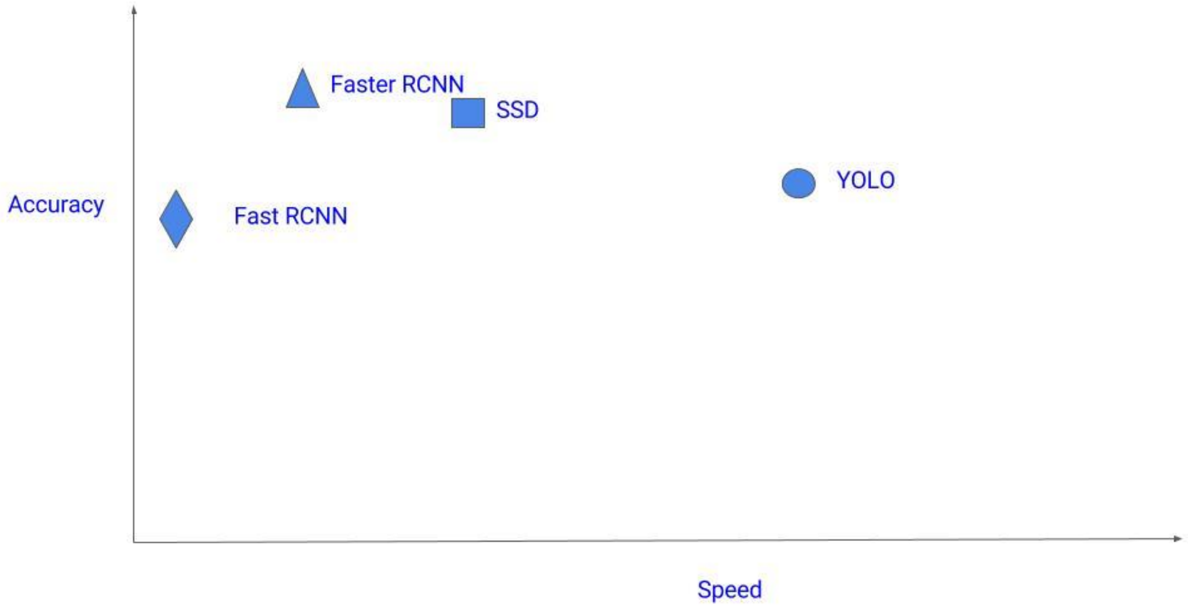


Figure 1: RCNN vs. SSD vs. YOLO Comparison

The two-shot detection models like Fast R-CNN and Faster R-CNN have two stages: region proposal and then classification of those regions and refinement of the location prediction. Single-shot detection skips the region proposal stage and yields final localization and content prediction at once. I found this excellent article on Medium and a paper by Google comparing different types of models.

While two-shot detection models achieve better performance, and YOLO the most speed, single-shot detection is in the sweet spot of performance and speed. Since the focus was on the most optimal solution and not the most accurate one, I went ahead with YOLO.

Within YOLO, there are various different versions with speed and accuracy trade-offs.

A YOLO History

"YOLO" refers to "You Only Look Once," a family of models that Joseph Redmon introduced in his May 2016 paper, "**You Only Look Once: Unified, Real-Time Object Detection**."

Redmon subsequently introduced YOLOv2 and v3 before announcing that he was stepping away from computer vision research.

Then, on April 23, 2020, Alexey Bochkovskiy **published YOLOv4**, and on June 9, 2020, Glenn Jocher added **released YOLOv5** .

A YOLOv5 Colab notebook, running a Tesla P100, could achieve a speed of 140 frames per second. By contrast, YOLOv4 achieved 50 FPS. Also, YOLOv5 is small weighing just 27 MB in comparison to ~270 MB for YOLOv4.

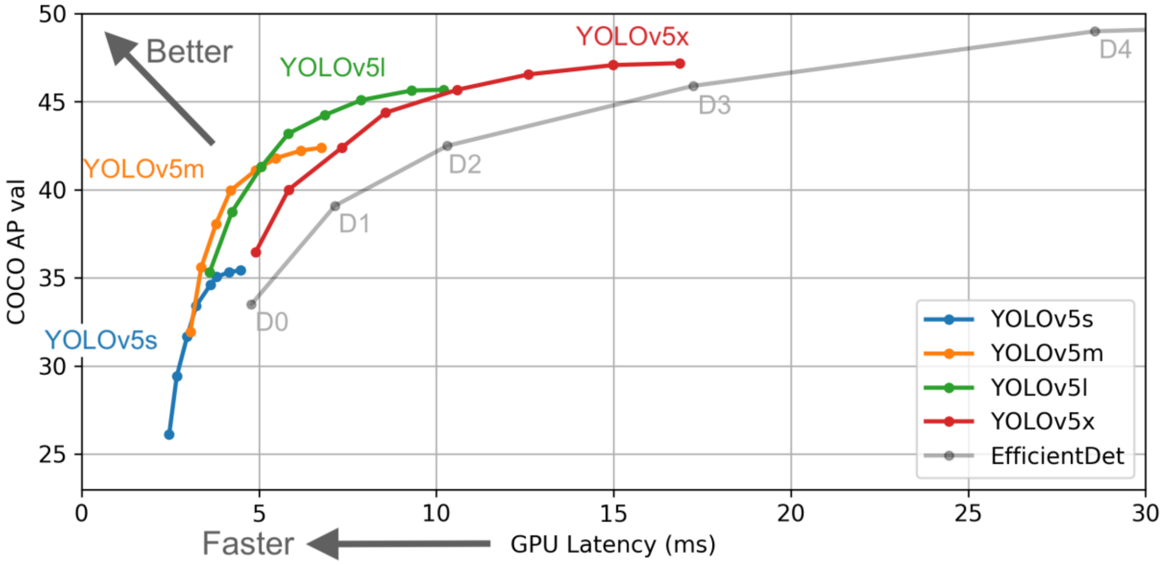


Figure 2: YOLOv5 Performance

Model Training

I initially ran the model training for 5 epochs and then again for 15 more epochs. Making a total of 20 epochs.

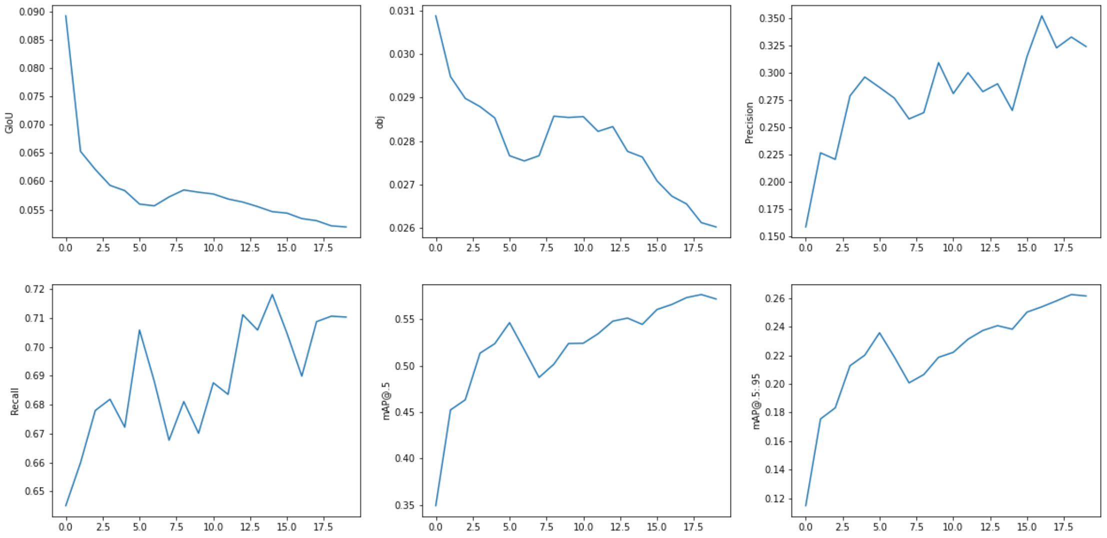


Figure 3: Results on 20 epochs

The metrics used are:

- GIoU loss gain
- Objective loss gain
- Precision
- Recall
- Mean Average Precision at 50% IOU
- Mean Average Precision at 95% IOU

Inference:

The average metrics for the test dataset were:

- Precision: 0.34
- Recall: 0.704
- mAP@0.5: 0.58
- mAP@0.95: 0.265

The inference time for each image was 4.5 milliseconds.

You can watch this video to watch the inference in real time.

Conclusion:

At around 60% mAP@0.5, the model is certainly not the best model in the world. But at just 14 MB, it's a very small model which can be easily used on mobile and embedded devices. Also, depending on the actual application, the precision might improve even more as YOLO is not great at detecting very small object s, a lot of which were present in the dataset, which brought down the accuracy.