Approach for IDD Dataset Challenge

This report concerns with the work that our team, i.e TEAM-F , has done on the given problem statement.

Team Members: Ujjwal Upadhyay, Rishabh Banga, Divyansh Jha

What did we do? The Approach

Our solution is based on the modified retinanet based model. I decided to use retinanet as it's much simpler comparing to Faster-RCNN like models or SSD while having comparable results, this allows much easier experiments and debugging/tuning of the model.

My solution is based on:

1. https://github.com/yhenon/pytorch-retinanet
2. https://github.com/fizyr/keras-retinanet

I scaled the original images to 512x512 resolution, with 256 resolution I have seen results degradation and using the full resolution was not as practical with heavier base Models.

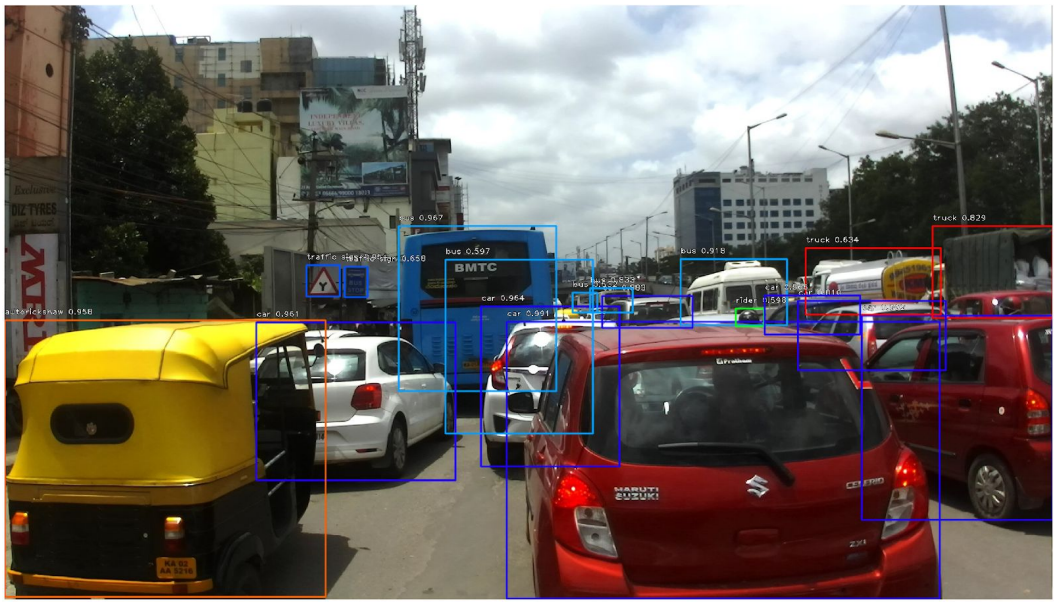
Modifications I have done to the original implementation:

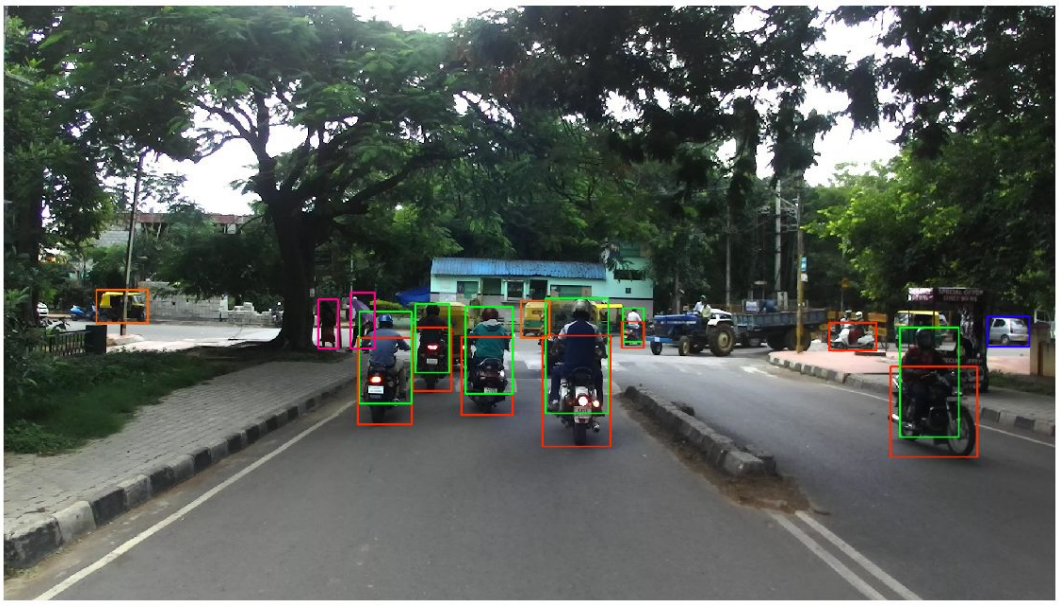
- added an extra output for smaller anchors (level 2 pyramid layer) to handle smaller boxes.
- added another classification output predicting the class of the whole image ('Person only', ‘Truck only’ or ‘Both’). I have not used the output but even making the model to predict other related function improved the result.
- As classification outputs overfit much faster comparing to anchors position/size regression outputs, I added dropout to anchor and the whole image class outputs. In addition to extra regularisation, it helped to achieve the optimal classification and regression results around the same epoch.

Augmentation used:

Shift, scale, and horizontal flip, for some images random level of blur and noise and gamma changes. I limited the amount of brightness/gamma augmentations as it was hard for me to verify if it does not invalidate labels.

Some of the results that we got from Retinanet:





Our second approach: YOLO-v3

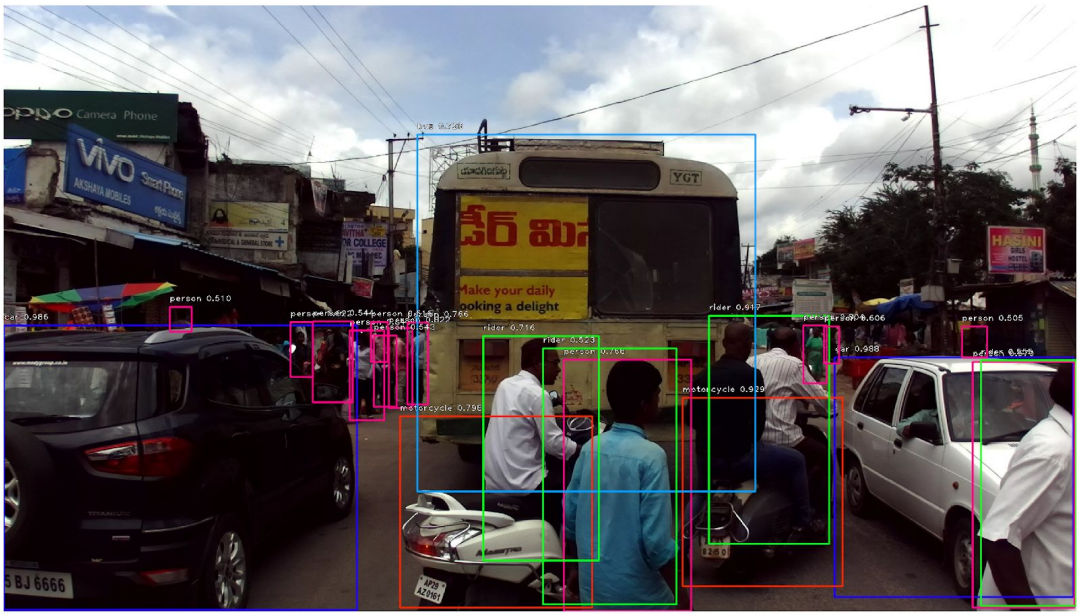
The second model we trained was Yolo-v3 due to ***complications that openvino has with retinanet***. YOLOv3 is extremely fast and accurate. In mAP measured at 0.5, IOU YOLOv3 is on par with Focal Loss but about 4x faster.

Same augementations were used in Yolo-v3 as in retinanet.

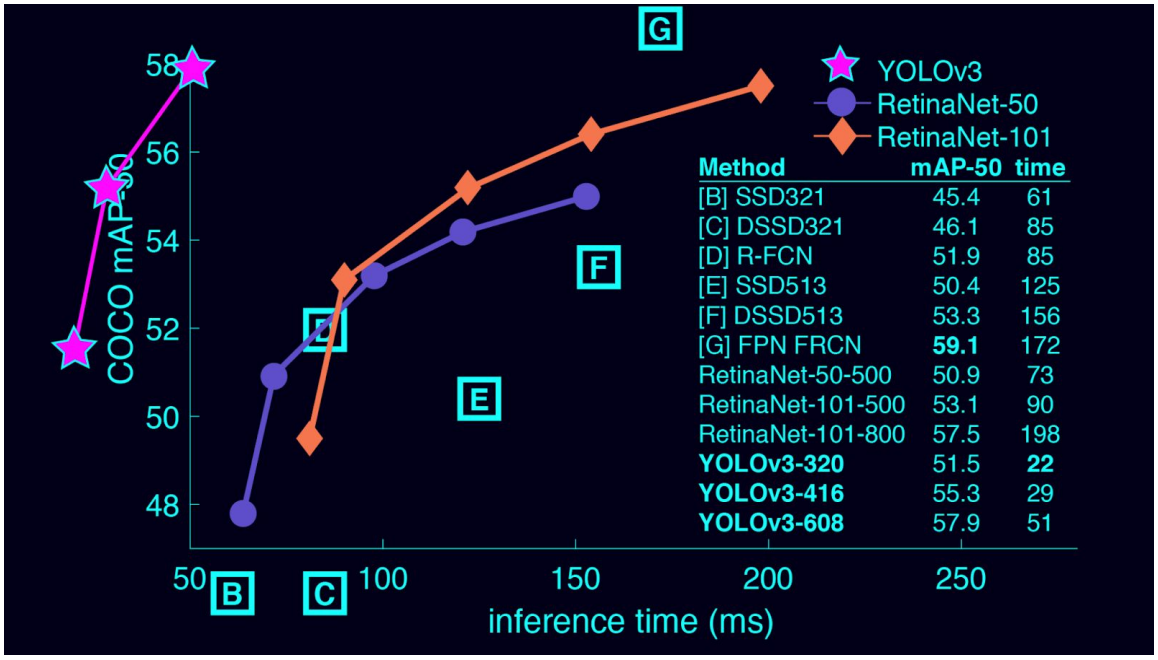
My solution is based on:

1. https://github.com/qqwweee/keras-yolo3
2. https://github.com/experiencor/keras-yolo3

Results from Yolo-v3:



Comparison between them (taken from the internet):



Conversion between different model type

1. Conversion of our model from h5 to protobuf format is done using this repository. https://github.com/amir-abdi/keras_to_tensorflow
2. The conversion of our protobuf model to IR (Intermediate Representation) is done using the help of this repository. https://github.com/PINTO0309/OpenVINO-YoloV3

What we plan to do now that competition is over

We are trying to distill our model so as to make it faster on CPU and it will, therefore, reduce inferencing time. We are making use of implementation provided at Intel Neural Network Distiller https://github.com/NervanaSystems/distiller and blogs on intel

website like this
https://software.intel.com/en-us/articles/knowledge-distillation-with-keras .
We are trying to convert the trained model to the IR model which can run on NCS.
and has quite good inference speed.