

Solution by: Novus Neurons

PSID: INTL-IVA-06

Team Leader: Sunny Kumar



– Drive Around Detection

Machine learning model which correctly labels objects such as Cars, Bikes, Scooties and Rickshaws, with an accuracy of 95% while predicting in images as well as videos.

Works well on partially visible objects in the frame.

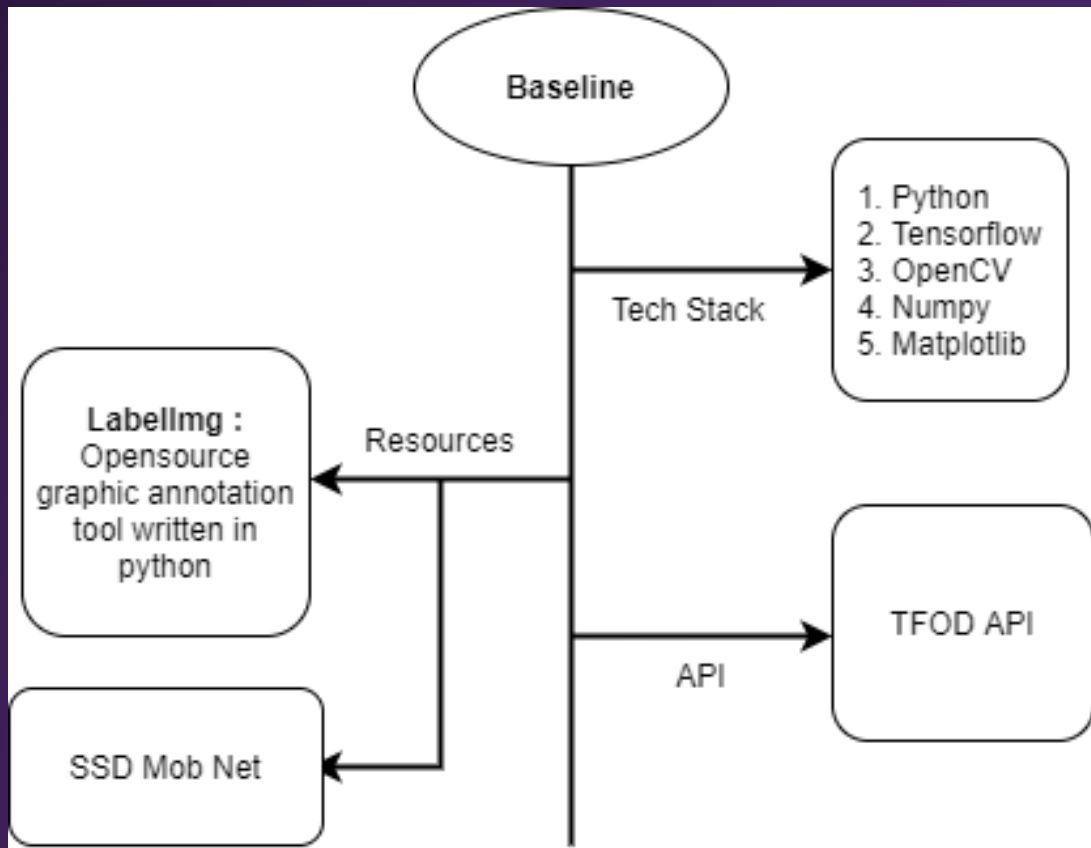
Using pretrained dense deep learning models and leveraging the model with transfer learning to enhance the results.

Base Line

To begin with will be using python and a machine leaning frame work.

Will use Labellmg to annotate our train and test images.

TFOD API and SSD Mob net at last for building the model.

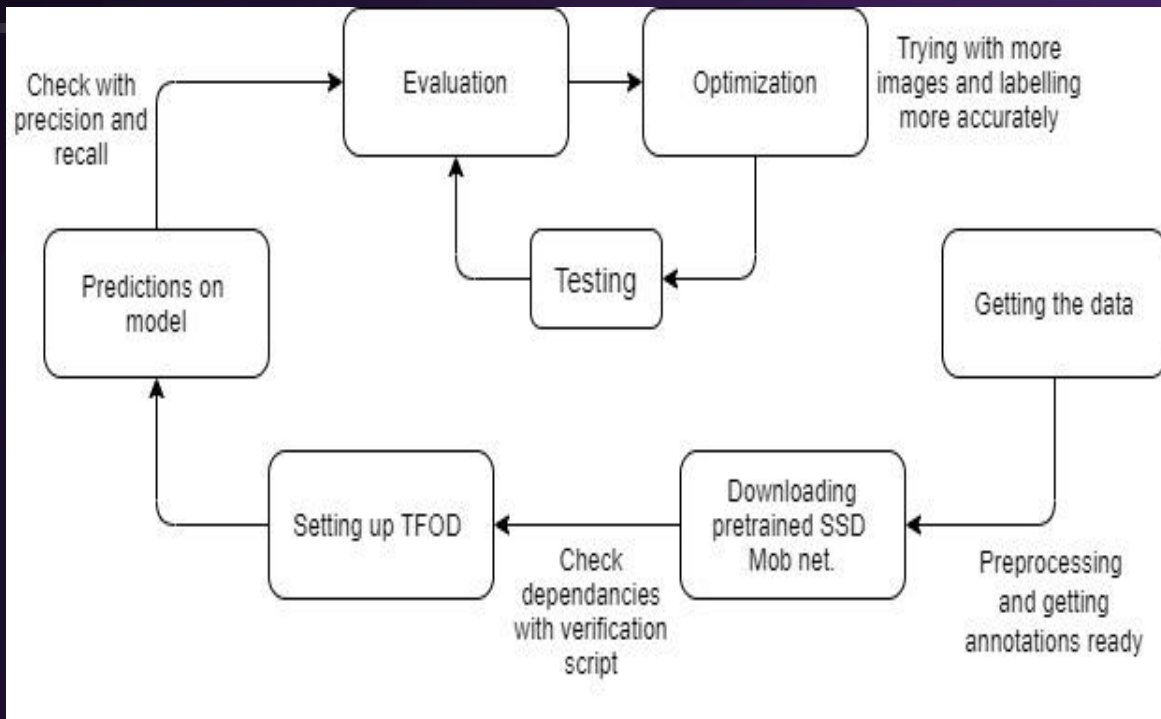


— Implementing Idea

Our approach towards the problem



Approach to the problem



We will be using a pretrained opensource deep learning model that predicts quite well - [SSD MobileNet V2 FPNLite 320x320](#) for our object recognition.

Then using the TFOD API we will localize the object in a better way.

Once our model is ready will evaluate its scores and test on random drive around videos to check for corrections and confusions then optimize accordingly.

– Insights in the approach

Getting a good accuracy is always an issue with image related problems. to overcome this we can leverage the models using transfer learning and hence increase our chances to get better accuracy.

Using the transfer learnt recognition model and TFOD API together, It levitates the accuracy of the model significantly. Focussing on less number of classes (like less than 5) also impact positively.

Using a more dense model to increase the accuracy and also to keep up with more classes on the cost of increasing the training time (will consume more resources) can be a solution as well.

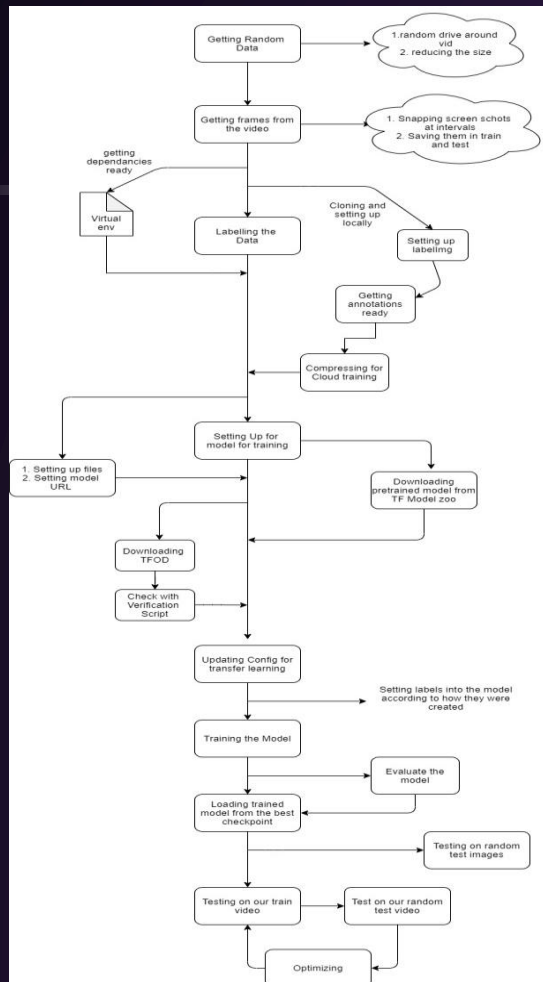
Development Pipeline

The overall workflow of the development pipeline we will be going through is shown to the left.

The workflow can be divided into 3 parts:

- 1) Getting Data ready
- 2) Setting Up Model
- 3) Training and Evaluating

In next 3 slides we will individually walk through each of these sections in detail.



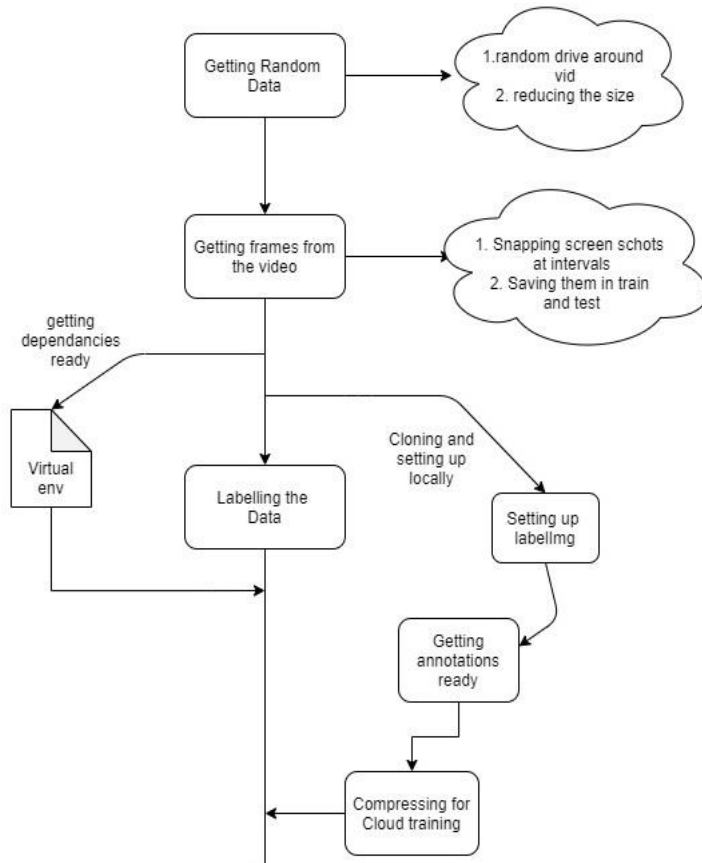
1. Getting Data Ready

In this section we will work on getting the data then pre-processing the data.

In our case we have a random drive around video in the Pune city which will be extracted in frames as the input data images.

Using the labelling to annotate the labels in our dataset to create train and test images.

Finally, setting up the virtual env for running the dependencies and compressing the images for cloud training.



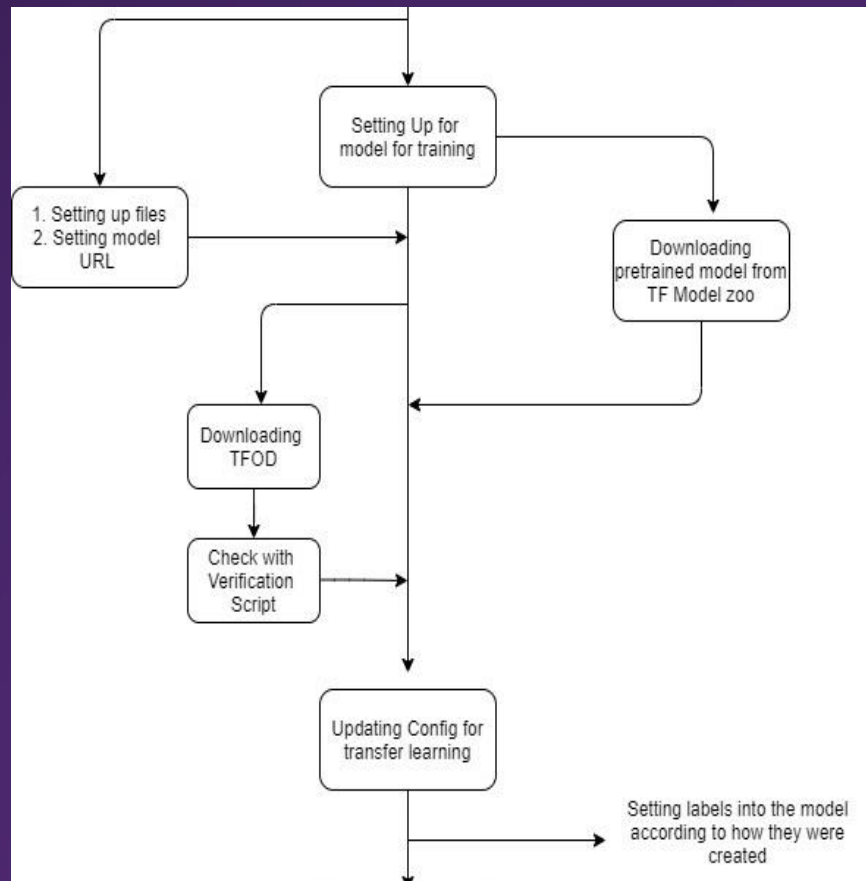
2. Setting Up Model

To begin with, in this section we will download the SSD Mob net model from the TF Model zoo for object recognition task.

Next we will be setting up the TFOD API to localize the object in the images.

The original model will come with different configurations and we would have to change them according to our requirement for this problem state.

Labels will be set here for the model.



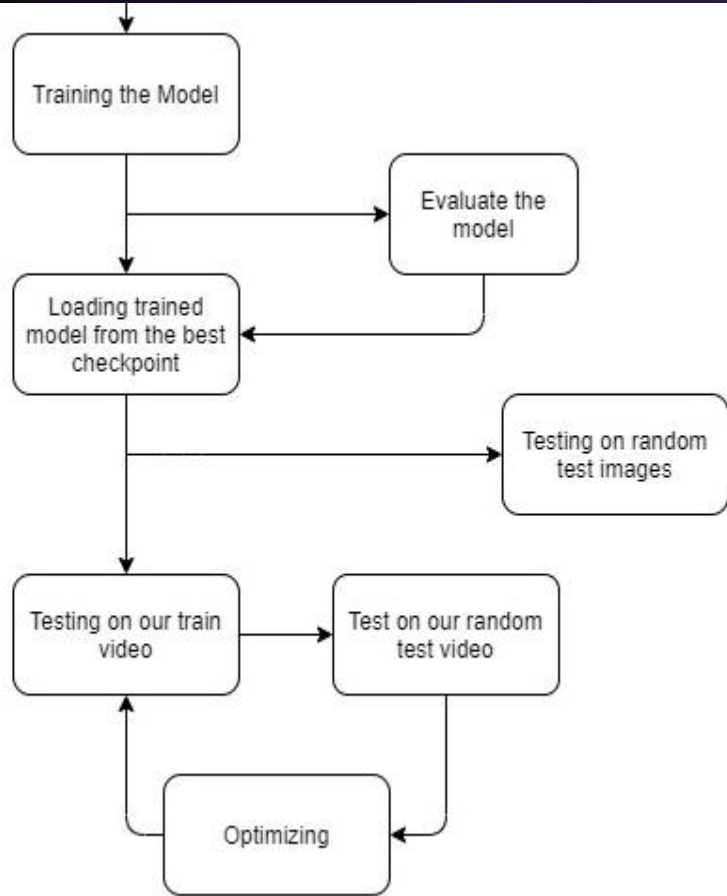
3. Training and Evaluating Model

In this section we will be training our model for 4000 to 4500 epochs. (takes approx. 5min on Tesla K80).

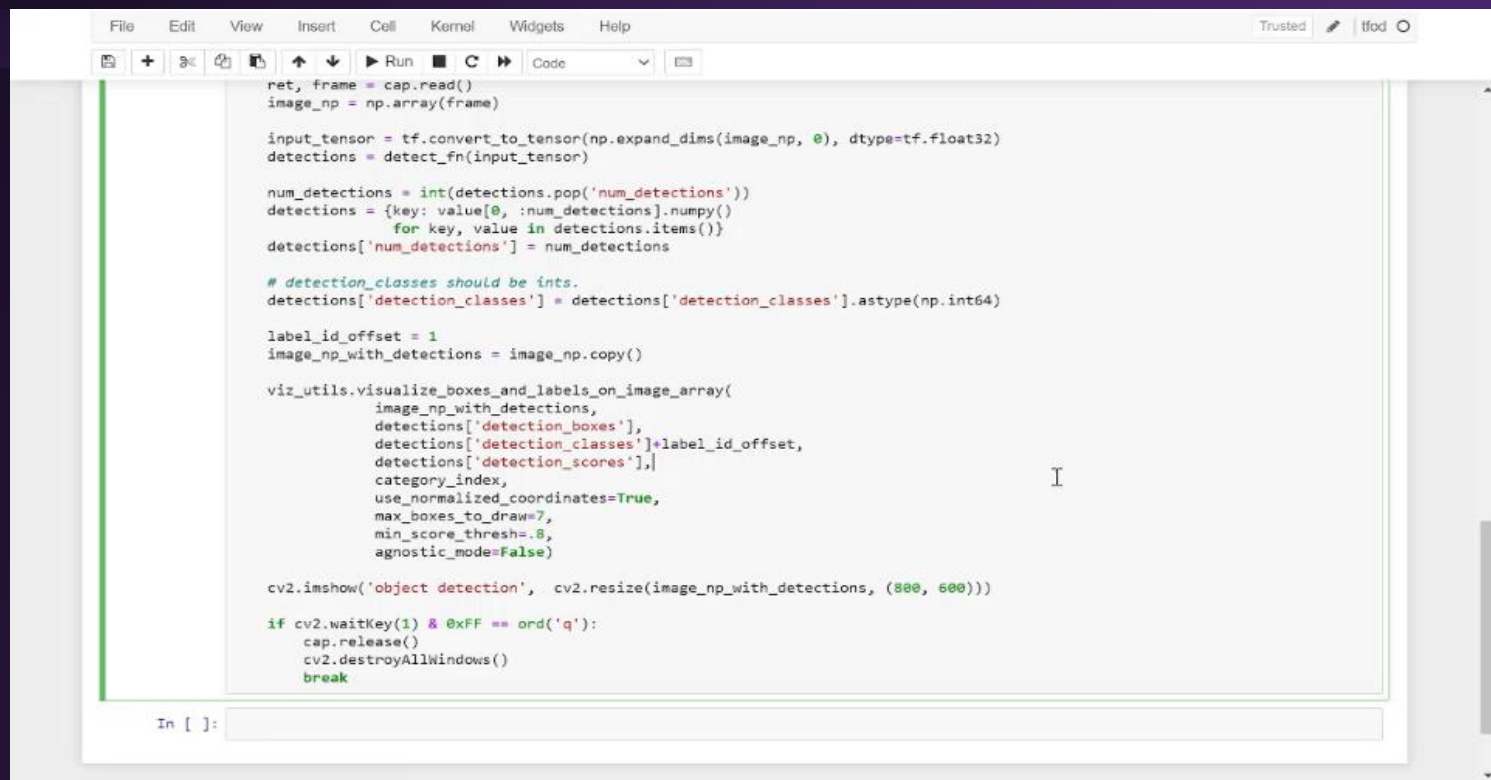
Will evaluate the model on targeting the precision and recall metrics. Check on random test images will be made.

Will run it on the train video to check its performance then move to test video. Here we will find classes which are been poorly classified.

This final step will be a continuous testing and optimization phase until satisfied results.



Demo



```
ret, frame = cap.read()
image_np = np.array(frame)

input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
detections = detect_fn(input_tensor)

num_detections = int(detections.pop('num_detections'))
detections = {key: value[0, :num_detections].numpy()
               for key, value in detections.items()}
detections['num_detections'] = num_detections

# detection_classes should be ints.
detections['detection_classes'] = detections['detection_classes'].astype(np.int64)

label_id_offset = 1
image_np_with_detections = image_np.copy()

viz_utils.visualize_boxes_and_labels_on_image_array(
    image_np_with_detections,
    detections['detection_boxes'],
    detections['detection_classes']+label_id_offset,
    detections['detection_scores'],
    category_index,
    use_normalized_coordinates=True,
    max_boxes_to_draw=7,
    min_score_thresh=.8,
    agnostic_mode=False)

cv2.imshow('object detection', cv2.resize(image_np_with_detections, (800, 600)))

if cv2.waitKey(1) & 0xFF == ord('q'):
    cap.release()
    cv2.destroyAllWindows()
    break
```

In []:

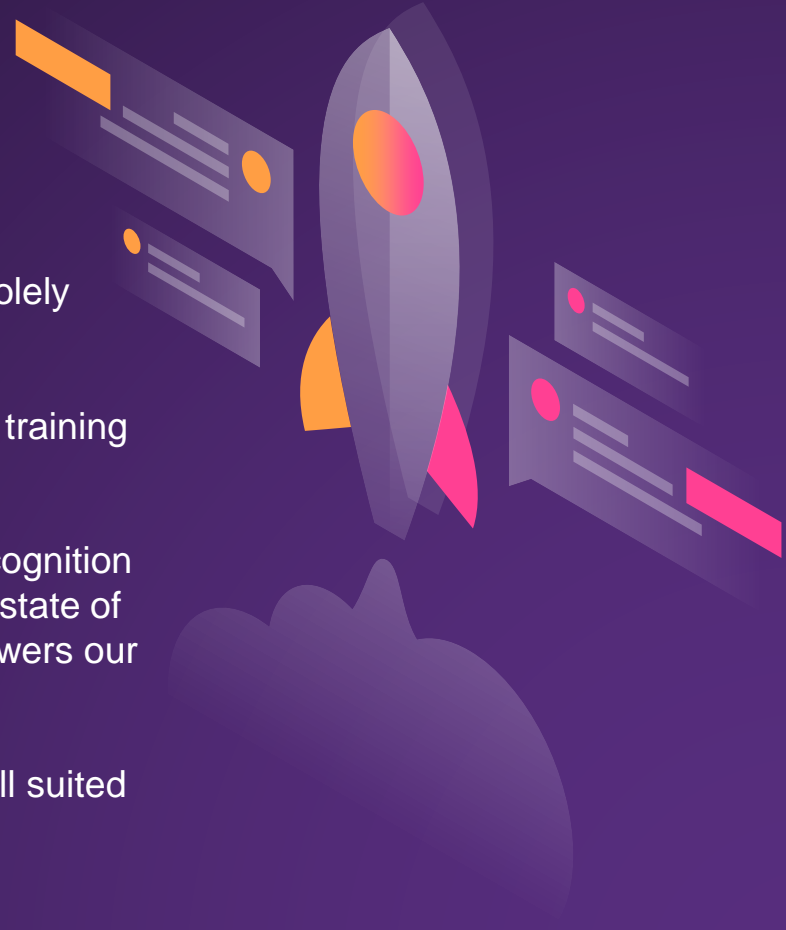
— Vision of the Idea

Considering the data set is already available the model solely depends on time taken to set up the files and training.

The model already yields high accuracy on a small set of training images.

Most of the Object detection model will fail in either of recognition or localization unlike this one because we will be using a state of the art deep learning model on top of TFOD API which lowers our chances of losing accuracy.

Considering the already know difficulties this model is well suited for Advanced Object Detection needed in our case.



Members

1. Sunny Kumar
2. Riddhi Gujarathi
3. Saniya Mulla
4. Prathamesh Parit
5. Shriraj Pawar
6. Hrushikesh Kachgunde



Thanks!