Object Detection using TensorFlow

CPS-584 Advanced Intelligent Systems and Deep Learning

Department of Computer Science
University of Dayton
Professor: Mehdi R Zargham

Shravya Reddy Akmy – 101709698
Surya Venkatesh Vijjana – 101709607

Introduction

This project involved implementing Object Detection using TensorFlow and deploying the trained model to mobile environment. We are using Pre-Trained Models to detect Cat(s) and Dog(s) in the given input image.

Description

To develop a model that can do object detection, we are using 1000 images of cat and dogs. Each image may contain the picture of a cat, or a dog, or both and 80% of the dataset is used as training data and the rest 20% used as the test data for the model training.

We used the **ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8** pre-trained model from TensorFlow Model Zoo with 90 detection classes which is configured to detect 2 classes i.e., Cat and Dog as per our requirement.

Purpose of the project

The purpose of this project is to detect cats and dogs in any image given as an input to the trained model. The input can be an image from a local machine or from a live webcam footage or when deployed to mobile environment the input should be live camera footage from the primary camera.

Dataset

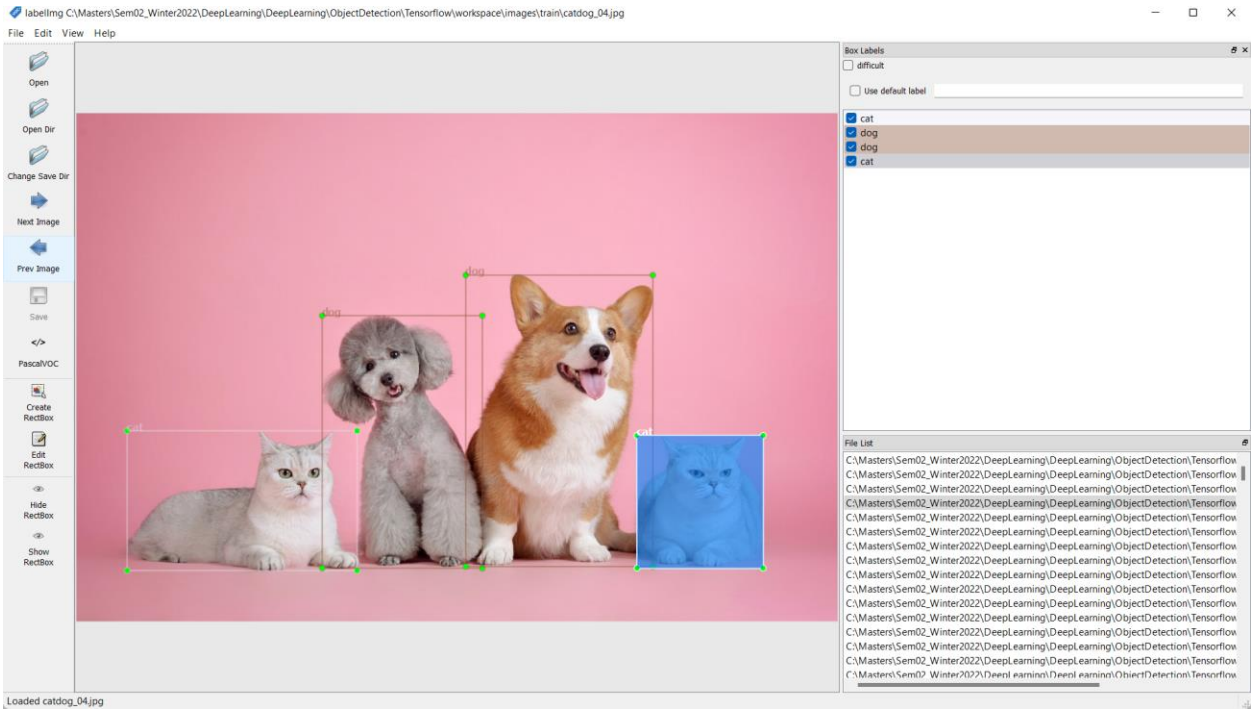
For the preparation of data set we used few images from the kaggle data set provided and from different web sources like Google Images, Pixabay and Unsplash.

Sample Images

Implementation

To train the model, first we need to download the pre-trained model from the TensorFlow model zoo and extract the pre trained model zip file to our workspace.

- Create a virtual environment and install necessary dependencies including tensorflow-gpu and install CUDA and CUDNN to run the model using gpu.
- **Using LabelImg application**, label the cats and dogs in each image present in both train and test data folders and generate xml files.



- Create a pbtxt file(labelmap.pbtxt) with labelname and id's for both cat and dog.
- Using the python script from this github repo (Generate TFRecord), generate the tfrecord files for both train and test data with images, xml files and labelmap.pbtxt as input.
- Modify the config file of pre trained model as per our requirements.
    - Change the number of classes to 2 as we need to detect Cat and Dog.
    - Add the path(s) of labelmap.pbtxt file, test and train tf records to the config file.

```
pipeline_config.model.ssd.num_classes = len(labels)
pipeline_config.train_config.batch_size = 4
pipeline_config.train_config.fine_tune_checkpoint = os.path.join(paths['PRETRAINED_MODEL_PATH'],
                                                                 PRETRAINED_MODEL_NAME, 'checkpoint', 'ckpt-0')
pipeline_config.train_config.fine_tune_checkpoint_type = "detection"
pipeline_config.train_input_reader.label_map_path= files['LABELMAP']
pipeline_config.train_input_reader.tf_record_input_reader.input_path[:] = [os.path.join(paths['ANNOTATION_PATH'], 'train.record')]
pipeline_config.eval_input_reader[0].label_map_path = files['LABELMAP']
pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[:] = [os.path.join(paths['ANNOTATION_PATH'],
'test.record')]
```

- Train the model to get minimum loss values and good mAP and recall values. We trained the model for 20,000 steps.

```
python Tensorflow\models\research\object_detection\model_main_tf2.py
--model_dir=Tensorflow\workspace\models\my_ssd_mobnet --pipeline_con
fig_path=Tensorflow\workspace\models\my_ssd_mobnet\pipeline.config -
-num_train_steps=20000
```

- After training is completed, evaluate the model to get the metrics related to the model.

```
python Tensorflow\models\research\object_detection\model_main_tf2.py
--model_dir=Tensorflow\workspace\models\my_ssd_mobnet --pipeline_con
fig_path=Tensorflow\workspace\models\my_ssd_mobnet\pipeline.config -
-checkpoint_dir=Tensorflow\workspace\models\my_ssd_mobnet
```

```
Accumulating evaluation results...
DONE (t=0.08s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.618
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.881
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.716
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.418
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.636
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.543
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.719
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.748
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.559
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.762
INFO:tensorflow:Eval metrics at step 20000
I0416 00:10:55.588527 16936 model_lib_v2.py:1015] Eval metrics at step 20000
```

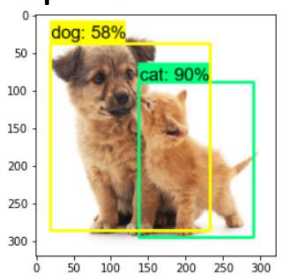
- Load saved pipeline config file and build the object detection model and restore the latest checkpoint file.

```
# Load pipeline config and build a detection model
configs = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
detection_model = model_builder.build(model_config=configs['model'],
is_training=False)
# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-21')).expect_partial()
```

- To Detect the Cat/Dog, pass the image path as an input.

```
img = cv2.imread(IMAGE_PATH)
img=cv2.resize(img,(320,320))
image_np = np.array(img)

input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0),
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)
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=10,
            min_score_thresh=.5,
            agnostic_mode=False)
plt.imshow(cv2.cvtColor(image_np_with_detections, cv2.COLOR_BGR2RGB))
plt.show()
```

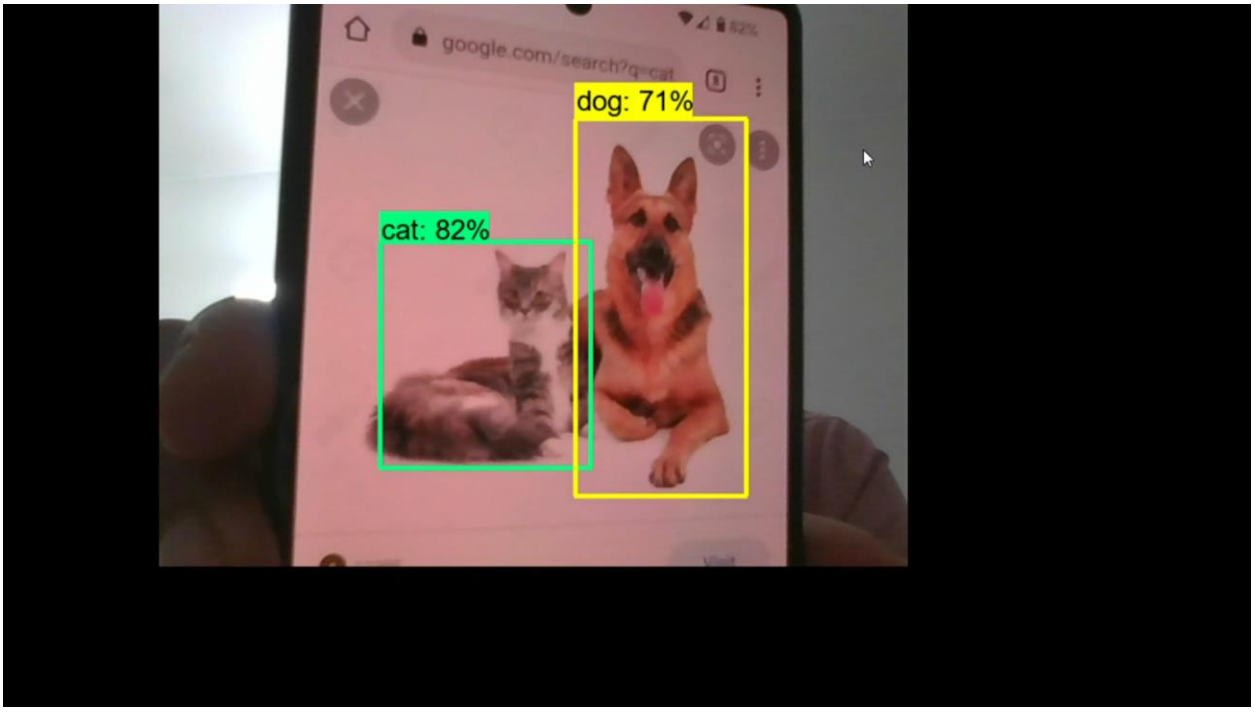
Output



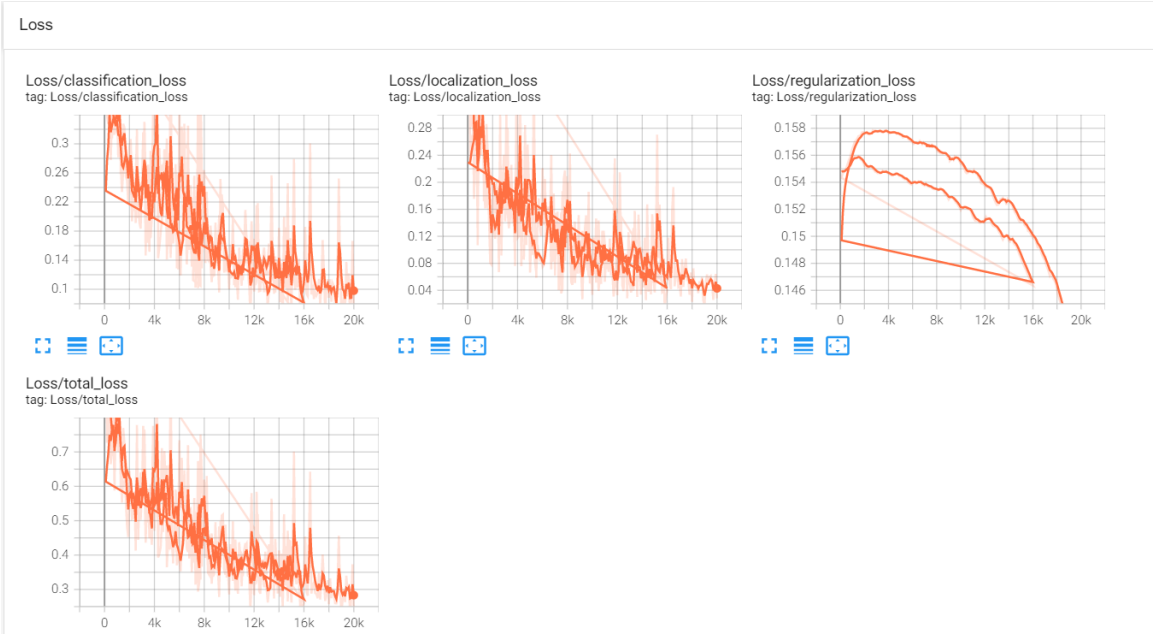
Webcam Implementation

```
cap = cv2.VideoCapture(0)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

while cap.isOpened():
    ret, frame = cap.read()
    image_np = np.array(frame)

    input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0),
dtype=tf.float32)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=10,
                min_score_thresh=.5,
                agnostic_mode=False)

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

    if cv2.waitKey(10) & 0xFF == ord('q'):
        cap.release()
        cv2.destroyAllWindows()
        break
```

Output



Training Plots

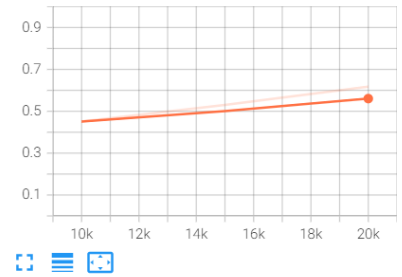


Testing Plots

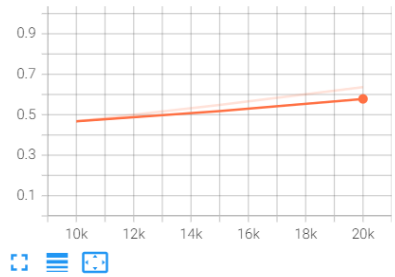
Precision

DetectionBoxes_Precision

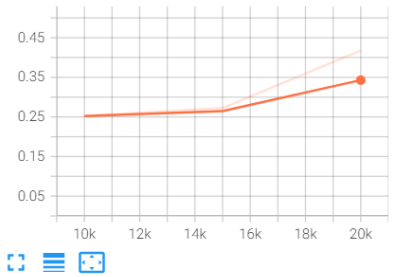
DetectionBoxes_Precision/mAP
tag: DetectionBoxes_Precision/mAP



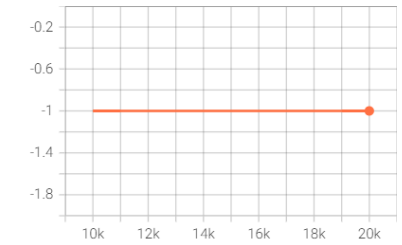
DetectionBoxes_Precision/mAP (large)
tag: DetectionBoxes_Precision/mAP (large)



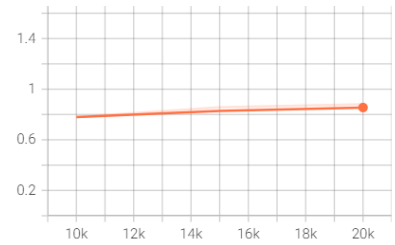
DetectionBoxes_Precision/mAP (medium)
tag: DetectionBoxes_Precision/mAP (medium)



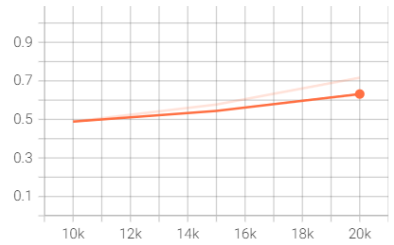
DetectionBoxes_Precision/mAP (small)
tag: DetectionBoxes_Precision/mAP (small)



DetectionBoxes_Precision/mAP@.50IOU
tag: DetectionBoxes_Precision/mAP@.50IOU



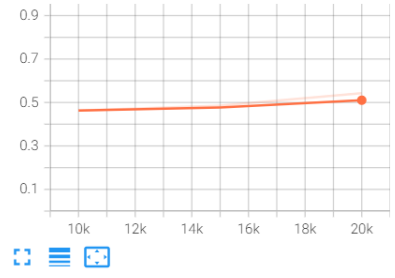
DetectionBoxes_Precision/mAP@.75IOU
tag: DetectionBoxes_Precision/mAP@.75IOU



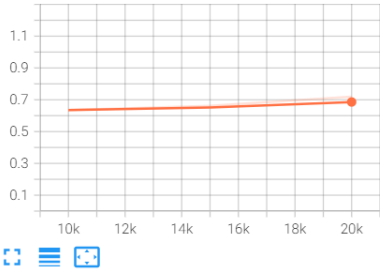
Recall

DetectionBoxes_Recall

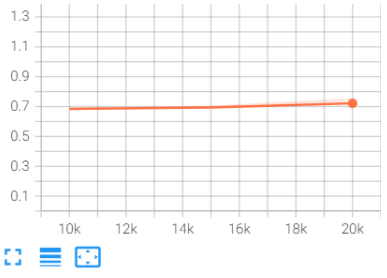
DetectionBoxes_Recall/AR@1
tag: DetectionBoxes_Recall/AR@1



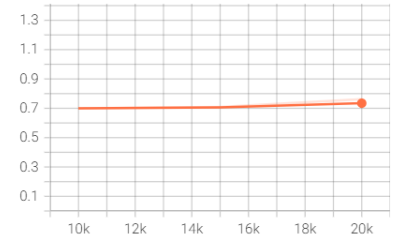
DetectionBoxes_Recall/AR@10
tag: DetectionBoxes_Recall/AR@10



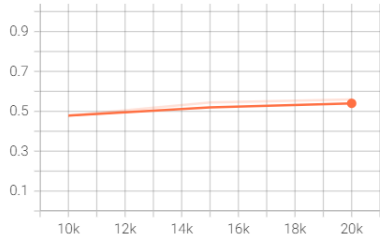
DetectionBoxes_Recall/AR@100
tag: DetectionBoxes_Recall/AR@100



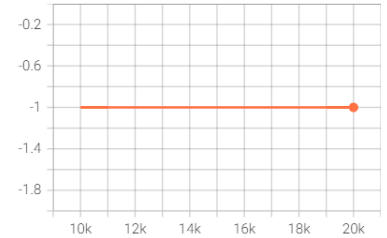
DetectionBoxes_Recall/AR@100 (large)
tag: DetectionBoxes_Recall/AR@100 (large)



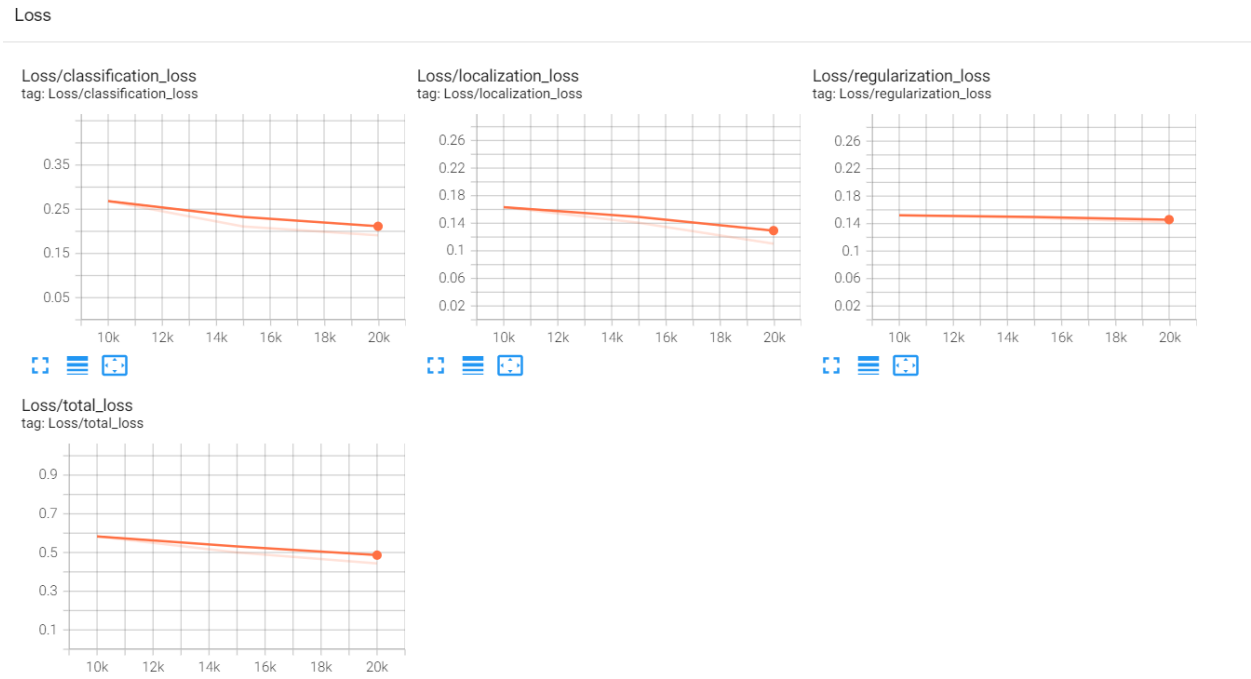
DetectionBoxes_Recall/AR@100 (medium)
tag: DetectionBoxes_Recall/AR@100 (medium)



DetectionBoxes_Recall/AR@100 (small)
tag: DetectionBoxes_Recall/AR@100 (small)



Loss



Evaluation

Mobile Deployment

- Freeze our current trained model and export the model and convert the model to tflite model.

  Freeze the model to export using this command

```
python Tensorflow\models\research\object_detection\exporter_main_v2.
py  --input_type=image_tensor --pipeline_config_path=Tensorflow\work
space\models\my_ssd_mobnet\pipeline.config --trained_checkpoint_dir=
Tensorflow\workspace\models\my_ssd_mobnet --output_directory=Tensorf
low\workspace\models\my_ssd_mobnet\export
```

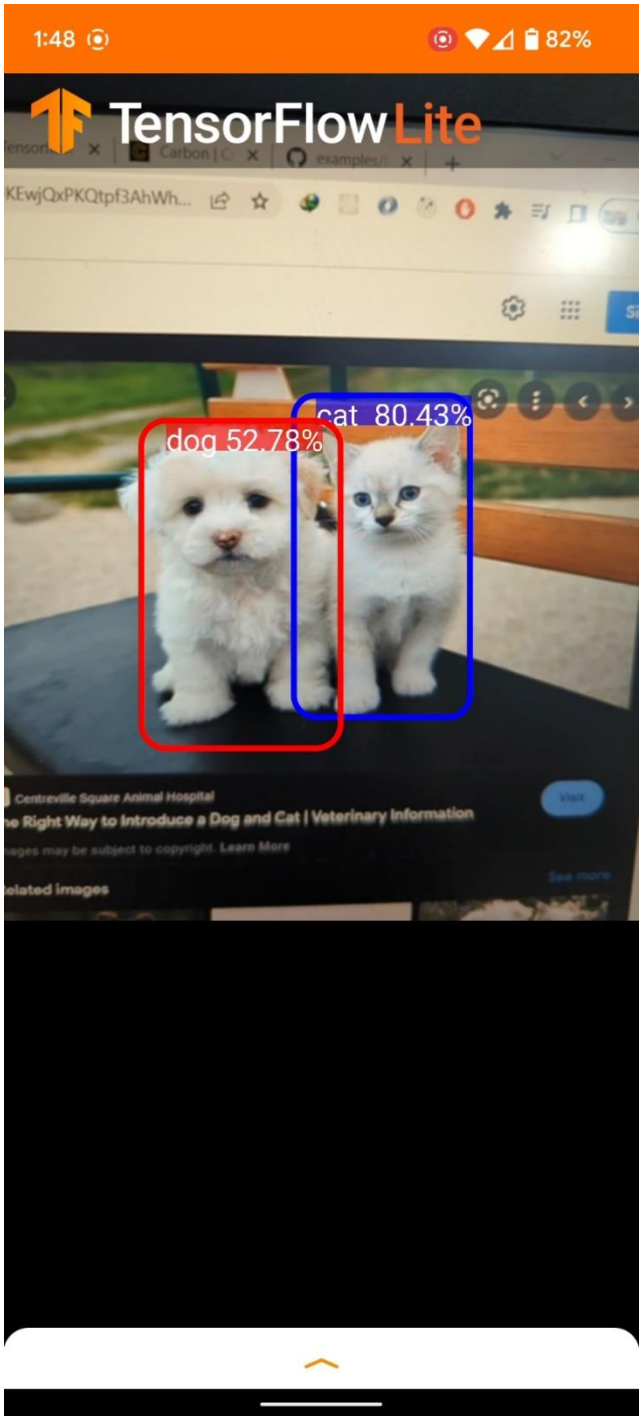
  Convert into tflite model using these 2 commands

```
python Tensorflow\models\research\object_detection\export_tflite_gra
ph_tf2.py  --pipeline_config_path=Tensorflow\workspace\models\my_ssd
_mobnet\pipeline.config --trained_checkpoint_dir=Tensorflow\workspac
e\models\my_ssd_mobnet --output_directory=Tensorflow\workspace\model
s\my_ssd_mobnet\tfliteexport
```

```
tflite_convert --saved_model_dir=Tensorflow\workspace\models\my_ssd_
mobnet\tfliteexport\saved_model --output_file=Tensorflow\workspace\m
odels\my_ssd_mobnet\tfliteexport\saved_model\detect.tflite --input_s
hapes=1,300,300,3 --input_arrays=normalized_input_image_tensor --out
put_arrays='TFLite_Detection_PostProcess','TFLite_Detection_PostProc
ess:1','TFLite_Detection_PostProcess:2','TFLite_Detection_PostProces
s:3' --inference_type=FLOAT --allow_custom_ops
```

- Clone the Tensorflow Android Deployment repository and import the directory as a project in the Android Studio and in the assets folder replace the tflite and labelmap.txt files with our exported tflite model and labelmap.txt file with cat and dog as labels.
- Install the required dependencies and connect your android device and run the build to install the apk file to the device connected.

Output



Issues Encountered

- While generating the tfrecord files, we encountered an error because in few xml files file format is missing (.jpg extension for filename tag in xml file). To fix this issue, we used **xml.etree** library to add the file format to the filename tag where it was missing.

```
from xml.etree import ElementTree as et

for img in os.listdir(os.path.join(paths['IMAGE_PATH'],'train')):
    if(img.split('.')[1]=='xml'):
        tree =
et.parse(os.path.join(paths['IMAGE_PATH'],'train',img))
        if(tree.find('.//filename').text.find('.jpg')==-1):
            tree.find('.//filename').text+='.jpg'
            tree.write(os.path.join(paths['IMAGE_PATH'],'train',img))
```

- Missing few dependencies while training our model. To fix this issue, installed those missing dependencies.
  pip install <missing library>

- When we tried different object detection models like resnet, inceptionResNet, efficientDet we got a version mismatch error and out of memory errors. So used **MobileNet** for training.

- While deploying the tflite model to the mobile, application keeps on crashing and when referred to the github issues section found that metadata for tflite model is missing. To fix this issue, we used tflite_support library and add metadata to our tflite model.

```
ObjectDetectorWriter = object_detector.MetadataWriter
_MODEL_PATH = os.path.join('Tensorflow','workspace','models',
                           'my_ssd_mobnet','tfliteexport','saved_model','detect.tflite')
# Task Library expects label files that are in the same format as the one below.
_LABEL_FILE = os.path.join("Tensorflow","examples","lite","examples",

"object_detection","android","app","src","main","assets","labelmap.txt")
_SAVE_TO_PATH = os.path.join("Tensorflow","workspace","models",
                             'my_ssd_mobnet','tfliteexport','saved_model','detect.tflite')

_INPUT_NORM_MEAN = 127.5
_INPUT_NORM_STD = 127.5

# Create the metadata writer.
writer = ObjectDetectorWriter.create_for_inference(
    writer_utils.load_file(_MODEL_PATH), [_INPUT_NORM_MEAN], [_INPUT_NORM_STD],
    [_LABEL_FILE])

# Verify the metadata generated by metadata writer.
print(writer.get_metadata_json())

# Populate the metadata into the model.
writer_utils.save_file(writer.populate(), _SAVE_TO_PATH)
```

References

- Object Detection using Tensorflow Tutorial by Nicholas Renotte - Tutorial
- Error Guide [Github Repo]: TFOD Error Guide
- TF Record Generator Script: GenerateTFRecord
- Android Deployment: Tensorflow : Android
- Generate Metadata for TFLITE Model : Metadata Generator
- Dataset : Kaggle, pixabay, unsplash