USER GUIDE

1) Open notebook file model.ipynb in jupyter notebook.

2) Install tqdm and beautiful soup library in the conda environment using conda install.

3) Run the first cell to load and compile all the necessary libraries and supporting files.

```
]:
import sys

from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint, LearningRateScheduler, EarlyStopping, ReduceLROnPlateau, TensorBoard
from keras import backend as K
from keras.models import load_model
from math import ceil
import numpy as np
from matplotlib import pyplot as plt
import tensorflow as tf

from models.ssd_mobilenet import ssd_300
from misc.keras_ssd_loss import SSDLoss, FocalLoss, weightedSSDLoss, weightedFocalLoss
from misc.keras_layer_AnchorBoxes import AnchorBoxes
from misc.keras_layer_L2Normalization import L2Normalization
from misc.ssd_box_encode_decode_utils import SSDBoxEncoder, decode_y, decode_y2
from misc.ssd_batch_generator import BatchGenerator
from bs4 import BeautifulSoup

import os


import cv2
import time




Using TensorFlow backend.
```

4) Run the second cell to define all the necessary variables required by the model.

```
In [ ]:
img_height = 300  # Height of the input images
img_width = 300  # Width of the input images
img_channels = 3  # Number of color channels of the input images
subtract_mean = [123, 117, 104]  # The per-channel mean of the images in the dataset
swap_channels = True  # The color channel order in the original SSD is BGR
n_classes = 20  # Number of positive classes, e.g. 20 for Pascal VOC, 80 for MS COCO
scales_voc = [0.1, 0.2, 0.37, 0.54, 0.71, 0.88,
              1.05]  # The anchor box scaling factors used in the original SSD300 for the Pascal VOC datasets
scales_coco = [0.07, 0.15, 0.33, 0.51, 0.69, 0.87,
               1.05]  # The anchor box scaling factors used in the original SSD300 for the MS COCO datasets
scales = scales_voc

aspect_ratios = [[1.0, 2.0, 0.5],
                 [1.0, 2.0, 0.5, 3.0, 1.0 / 3.0],
                 [1.0, 2.0, 0.5, 3.0, 1.0 / 3.0],
                 [1.0, 2.0, 0.5, 3.0, 1.0 / 3.0],
                 [1.0, 2.0, 0.5],
                 [1.0, 2.0, 0.5]]  # The anchor box aspect ratios used in the original SSD300; the order matters
two_boxes_for_ar1 = True
steps = [8, 16, 32, 64, 100, 300]  # The space between two adjacent anchor box center points for each predictor layer.
offsets = [0.5, 0.5, 0.5, 0.5, 0.5,
           0.5]  # The offsets of the first anchor box center points from the top and left borders of the image as a fractio
limit_boxes = False  # Whether or not you want to limit the anchor boxes to lie entirely within the image boundaries
variances = [0.1, 0.1, 0.2,
             0.2]  # The variances by which the encoded target coordinates are scaled as in the original implementation
coords = 'centroids'  # Whether the box coordinates to be used as targets for the model should be in the 'centroids', 'corne
normalize_coords = True

# 1: Build the Keras model

K.clear_session()  # Clear previous models from memory.
```

5) Run the 5th cell to create the model structure and load trained weights to it.

```
model = ssd_300("training",
                image_size=(img_height, img_width, img_channels),
                n_classes=n_classes,
                l2_regularization=0.0005,
                scales=scales,
                aspect_ratios_per_layer=aspect_ratios,
                two_boxes_for_ar1=two_boxes_for_ar1,
                steps=steps,
                offsets=offsets,
                limit_boxes=limit_boxes,
                variances=variances,
                coords=coords,
                normalize_coords=normalize_coords,
                subtract_mean=subtract_mean,
                divide_by_stddev=127.5,
                swap_channels=swap_channels)
model.load_weights("ssd300_epoch-222.h5")

```

```
WARNING:tensorflow:From C:\Users\abhin\Anaconda3\envs\ml1913\lib\site-packages\tensorflow\python\framework\op_def_library.py:26
```

6) Store all the image files which need to be tested under a single folder.

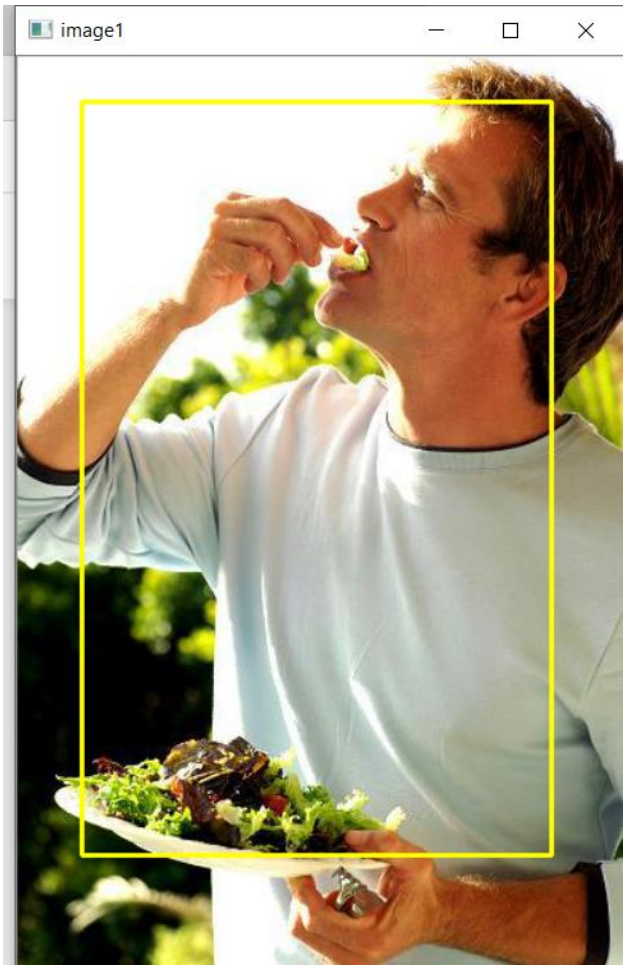
7) Go to the 6th cell and give path of this folder to the dir_path which is highlighted below.

```
in training mode
```

```
dir_path='data/val/'
for file in os.listdir(dir_path):
    filename = dir_path + file

    print(filename)

    img1 = cv2.imread(filename)
   # img = img.astype('uint8')

    img = cv2.cvtColor(img1,cv2.COLOR_BGR2RGB)


    # # img1 = ima[90:390,160:460]
    # img1 = cv2.resize(ima,dsize=(img_height,img_width))
    # im = img1
    orig_images = []  # Store the images here.
    input_images = []  # Store resized versions of the images here.
    orig_images.append(img1)

    # img1 = image.img_to_array(img1)
    # input_images.append(img1)
    # input_images = np.array(input_images)



    ima = img
    # img = img[:,a:a+320]
    image1 = cv2.resize(img,(300,300))
    image1 = np.array(image1,dtype=np.float32)
```

8) Run the 6th cell now to see the detection predictions made by the trained model.

9) The result will come it form of new opencv window namely ‘image1’ which will contain the bounding box on the original image.



10)The results will come one by one for each image. Press escape key to move to the next image results.

11)The opencv window will automatically close once all the results are displayed.