```
!pip install -U --pre tensorflow=="2.*"
!pip install tf_slim
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: tensorflow==2.* in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: tensorboard<2.10,>=2.9 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/p
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: protobuf<3.20,>=3.9.2 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.7/dis
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: keras-preprocessing>=1.1.1 in /usr/local/lib/python3.7/d
Requirement already satisfied: keras<2.10.0,>=2.9.0rc0 in /usr/local/lib/python3.7/dist
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: flatbuffers<2,>=1.12 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: tensorflow-estimator<2.10.0,>=2.9.0rc0 in /usr/local/lib
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/pytho
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/local/lib/
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/dis
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python3.7/dist
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/li
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dis
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Requirement already satisfied: tf_slim in /usr/local/lib/python3.7/dist-packages (1.1.0
Requirement already satisfied: absl-py>=0.2.2 in /usr/local/lib/python3.7/dist-packages
```

```
import os
import pathlib


if "models" in pathlib.Path.cwd().parts:
  while "models" in pathlib.Path.cwd().parts:
    os.chdir('..')
elif not pathlib.Path('models').exists():
  !git clone --depth 1 https://github.com/tensorflow/models


%%bash
cd models/research/
protoc object_detection/protos/*.proto --python_out=.


%%bash
cd models/research
pip install .
```

```
ERROR: Directory '.' is not installable. Neither 'setup.py' nor 'pyproject.toml' found.
```

```
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile

from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image
from IPython.display import display


!pip install object_detection
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
ERROR: Could not find a version that satisfies the requirement object_detection (from v
ERROR: No matching distribution found for object_detection
```

```
import sys

sys.path.insert(0, '/content/models/research/')


from object_detection.utils import ops as utils_ops
```

```
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as vis_util

# patch tf1 into `utils.ops`
utils_ops.tf = tf.compat.v1

# Patch the location of gfile
tf.gfile = tf.io.gfile


def load_model(model_name):
  base_url = 'http://download.tensorflow.org/models/object_detection/'
  model_file = model_name + '.tar.gz'
  model_dir = tf.keras.utils.get_file(
    fname=model_name,
    origin=base_url + model_file,
    untar=True)

  print(model_dir)
  model_dir = pathlib.Path(model_dir)/"saved_model"
  (model_dir)
  print(model_dir)
  model = tf.saved_model.load(str(model_dir))

  return model


# List of the strings that is used to add correct label for each box.
PATH_TO_LABELS = 'models/research/object_detection/data/mscoco_label_map.pbtxt'
category_index = label_map_util.create_category_index_from_labelmap(PATH_TO_LABELS, use_display_name=True)


model_name = 'ssd_mobilenet_v1_coco_2017_11_17'
detection_model = load_model(model_name)
```

```
/root/.keras/datasets/ssd_mobilenet_v1_coco_2017_11_17
/root/.keras/datasets/ssd_mobilenet_v1_coco_2017_11_17/saved_model
INFO:tensorflow:Saver not created because there are no variables in the graph to restor
INFO:tensorflow:Saver not created because there are no variables in the graph to restor
```

```
model_name = 'faster_rcnn_resnet50_coco_2018_01_28'
detection_model = load_model(model_name)
```

```
/root/.keras/datasets/faster_rcnn_resnet50_coco_2018_01_28
/root/.keras/datasets/faster_rcnn_resnet50_coco_2018_01_28/saved_model
INFO:tensorflow:Saver not created because there are no variables in the graph to restor
INFO:tensorflow:Saver not created because there are no variables in the graph to restor
```

```
tensorflow.python.training.tracking.autotrackable.AutoTrackable
```

```
!wget http://download.tensorflow.org/models/object_detection/faster_rcnn_resnet50_coco_2018_01_28.tar.gz
```

```
--2022-08-02 19:18:13--  http://download.tensorflow.org/models/object_detection/faster_
Resolving download.tensorflow.org (download.tensorflow.org)... 74.125.199.128, 2607:f8b
Connecting to download.tensorflow.org (download.tensorflow.org)|74.125.199.128|:80... c
HTTP request sent, awaiting response... 200 OK
Length: 381355771 (364M) [application/x-tar]
Saving to: ‘faster_rcnn_resnet50_coco_2018_01_28.tar.gz’

faster_rcnn_resnet5 100%[===================>] 363.69M   329MB/s    in 1.1s

2022-08-02 19:18:14 (329 MB/s) - ‘faster_rcnn_resnet50_coco_2018_01_28.tar.gz’ saved [3
```

```
!unzip '/content/faster_rcnn_resnet50_coco_2018_01_28.tar.gz'
```

```
Archive:  /content/faster_rcnn_resnet50_coco_2018_01_28.tar.gz
  End-of-central-directory signature not found.  Either this file is not
  a zipfile, or it constitutes one disk of a multi-part archive.  In the
  latter case the central directory and zipfile comment will be found on
  the last disk(s) of this archive.
unzip:  cannot find zipfile directory in one of /content/faster_rcnn_resnet50_coco_2018
        /content/faster_rcnn_resnet50_coco_2018_01_28.tar.gz.zip, and cannot find /cont
```

```
import tarfile


# open file
file = tarfile.open('/content/faster_rcnn_resnet50_coco_2018_01_28.tar.gz')

# extracting file
file.extractall('/content/')

file.close()
```

```
!wget http://download.tensorflow.org/models/object_detection/ssd_mobilenet_v1_coco_2017_11_17.tar.gz
```

```
--2022-08-02 19:22:56--  http://download.tensorflow.org/models/object_detection/ssd_mob
Resolving download.tensorflow.org (download.tensorflow.org)... 74.125.199.128, 2607:f8b
Connecting to download.tensorflow.org (download.tensorflow.org)|74.125.199.128|:80... c
HTTP request sent, awaiting response... 200 OK
Length: 76534733 (73M) [application/x-tar]
Saving to: ‘ssd_mobilenet_v1_coco_2017_11_17.tar.gz’

ssd_mobilenet_v1_co 100%[===================>]  72.99M   117MB/s    in 0.6s

2022-08-02 19:22:57 (117 MB/s) - ‘ssd_mobilenet_v1_coco_2017_11_17.tar.gz’ saved [76534
```

```
import tarfile


# open file
file = tarfile.open('/content/ssd_mobilenet_v1_coco_2017_11_17.tar.gz')

# extracting file
file.extractall('/content/')

file.close()


# Dependencies
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
import numpy as np


# load graphs using pb file path
def load_graph(pb_file):
    graph = tf.Graph()
    with graph.as_default():
        od_graph_def = tf.GraphDef()
        with tf.gfile.GFile(pb_file, 'rb') as fid:
            serialized_graph = fid.read()
            od_graph_def.ParseFromString(serialized_graph)
            tf.import_graph_def(od_graph_def, name='')
    return graph


# returns tensor dictionaries from graph
def get_inference(graph, count=0):
    with graph.as_default():
        ops = tf.get_default_graph().get_operations()
        all_tensor_names = {output.name for op in ops for output in op.outputs}
        tensor_dict = {}
        for key in ['num_detections', 'detection_boxes', 'detection_scores',
                    'detection_classes', 'detection_masks', 'image_tensor']:
            tensor_name = key + ':0' if count == 0 else '_{}:0'.format(count)
            if tensor_name in all_tensor_names:
                tensor_dict[key] = tf.get_default_graph().get_tensor_by_name(tensor_name)

        return tensor_dict


# renames while_context because there is one while function for every graph
def rename_frame_name(graphdef, suffix):
    for n in graphdef.node:
        if "while" in n.name:
            if "frame_name" in n.attr:
                n.attr["frame_name"].s = str(n.attr["frame_name"]).replace("while_context",
                                                                           "while_context" + suffix).encode(


if   name   == '  main  ':
```

```
# your pb file paths
frozenGraphPath2 = '/content/faster_rcnn_resnet50_coco_2018_01_28/frozen_inference_graph.pb'
frozenGraphPath1 = '/content/ssd_mobilenet_v1_coco_2017_11_17/frozen_inference_graph.pb'

# new file name to save combined model
combinedFrozenGraph = 'combined_frozen_inference_graph.pb'

# loads both graphs
graph1 = load_graph(frozenGraphPath1)
graph2 = load_graph(frozenGraphPath2)

# get tensor names from first graph

tensor_dict1 = get_inference(graph1)
with graph1.as_default():

    # getting tensors to add crop and resize step
    image_tensor = tensor_dict1['image_tensor']
    scores = tensor_dict1['detection_scores'][0]
    num_detections = tf.cast(tensor_dict1['num_detections'][0], tf.int32)
    detection_boxes = tensor_dict1['detection_boxes'][0]

    # I had to add NMS becuase my ssd model outputs 100 detections and hence it runs out of memory becua
    selected_indices = tf.image.non_max_suppression(detection_boxes, scores, 5, iou_threshold=0.5)
    selected_boxes = tf.gather(detection_boxes, selected_indices)

    # intermediate crop and resize step, which will be input for second model(FRCNN)
    cropped_img = tf.image.crop_and_resize(image_tensor,
                                           selected_boxes,
                                           tf.zeros(tf.shape(selected_indices), dtype=tf.int32),
                                           [300, 60] # resize to 300 X 60
                                           )
    cropped_img = tf.cast(cropped_img, tf.uint8, name='cropped_img')
gdef1 = graph1.as_graph_def()
gdef2 = graph2.as_graph_def()

g1name = "graph1"
g2name = "graph2"

# renaming while_context in both graphs
rename_frame_name(gdef1, g1name)
rename_frame_name(gdef2, g2name)

# This combines both models and save it as one

with tf.Graph().as_default() as g_combined:

    x, y = tf.import_graph_def(gdef1, return_elements=['image_tensor:0', 'cropped_img:0'])

    z, = tf.import_graph_def(gdef2, input_map={"image_tensor:0": y}, return_elements=['detection_boxes:0

    tf.train.write_graph(g_combined, "./", combinedFrozenGraph, as_text=False)
```

```
# If you want to test the code with your images, just add path to the images to the TEST_IMAGE_PATHS.
PATH_TO_TEST_IMAGES_DIR = pathlib.Path('models/research/object_detection/test_images')
TEST_IMAGE_PATHS = sorted(list(PATH_TO_TEST_IMAGES_DIR.glob("*.jpg")))
TEST_IMAGE_PATHS
```

```
[PosixPath('models/research/object_detection/test_images/image1.jpg'),
 PosixPath('models/research/object_detection/test_images/image2.jpg'),
 PosixPath('models/research/object_detection/test_images/image3.jpg')]
```

```
# Dependencies
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()
import numpy as np

IMAGE_SIZE = (12, 8)


def get_inference(graph, count=0):
    with graph.as_default():
        ops = tf.get_default_graph().get_operations()
        all_tensor_names = {output.name for op in ops for output in op.outputs}
        tensor_dict = {}
        for key in ['num_detections', 'detection_boxes', 'detection_scores',
                    'detection_classes', 'detection_masks', 'image_tensor']:
            tensor_name = key + ':0' if count == 0 else '_{}:0'.format(count)
            if tensor_name in all_tensor_names:
                tensor_dict[key] = tf.get_default_graph().get_tensor_by_name(tensor_name)

        return tensor_dict


def load_image_into_numpy_array(image):
  (im_width, im_height) = image.size
  return np.array(image.getdata()).reshape(
      (im_height, im_width, 3)).astype(np.uint8)

# load graphs using pb file path
def load_graph(pb_file):
    graph = tf.Graph()
    with graph.as_default():
        od_graph_def = tf.GraphDef()
        with tf.gfile.GFile(pb_file, 'rb') as fid:
            serialized_graph = fid.read()
            od_graph_def.ParseFromString(serialized_graph)
            tf.import_graph_def(od_graph_def, name='')
    return graph

combinedFrozenGraph = '/content/combined_frozen_inference_graph.pb'

# loads both graphs
detection_graph = load_graph(combinedFrozenGraph)
tensor_dict1 = get_inference(detection_graph)

print(tensor_dict1)
```

```
with detection_graph.as_default():
  with tf.Session(graph=detection_graph) as sess:
    for image_path in TEST_IMAGE_PATHS:
      image = Image.open(image_path)
      # the array based representation of the image will be used later in order to prepare the
      # result image with boxes and labels on it.
      image_np = load_image_into_numpy_array(image)
      # Expand dimensions since the model expects images to have shape: [1, None, None, 3]
      image_np_expanded = np.expand_dims(image_np, axis=0)
      image_tensor = tensor_dict1['image_tensor']
      # Each box represents a part of the image where a particular object was detected.
      scores = tensor_dict1['detection_scores'][0]

      num_detections = tf.cast(tensor_dict1['num_detections'][0], tf.int32)

      detection_boxes = tensor_dict1['detection_boxes'][0]

      # Actual detection.
      (boxes, scores, classes, num_detections) = sess.run(
          [boxes, scores, classes, num_detections],
          feed_dict={image_tensor: image_np_expanded})
      # Visualization of the results of a detection.
      vis_util.visualize_boxes_and_labels_on_image_array(
          image_np,
          np.squeeze(boxes),
          np.squeeze(classes).astype(np.int32),
          np.squeeze(scores),
          category_index,
          use_normalized_coordinates=True,
          line_thickness=8)
      plt.figure(figsize=IMAGE_SIZE)
      plt.imshow(image_np)


import numpy as np

def nms(dets, thresh):
    x1 = dets[:, 0]
    y1 = dets[:, 1]
    x2 = dets[:, 2]
    y2 = dets[:, 3]
    scores = dets[:, 4]

    areas = (x2 - x1 + 1) * (y2 - y1 + 1)
    order = scores.argsort()[::-1]

    keep = []
    while order.size > 0:
        i = order[0]
        keep.append(i)
        xx1 = np.maximum(x1[i], x1[order[1:]])
        yy1 = np.maximum(y1[i], y1[order[1:]])
        xx2 = np.minimum(x2[i], x2[order[1:]])
        yy2 = np.minimum(y2[i], y2[order[1:]])

        w = np.maximum(0.0, xx2 - xx1 + 1)
        h = np.maximum(0.0, yy2 - yy1 + 1)
```

```
            inter = w * h
            ovr = inter / (areas[i] + areas[order[1:]] - inter)

            inds = np.where(ovr <= thresh)[0]
            order = order[inds + 1]

        return keep
```

✓ 9s completed at 04:28