View on TensorFlow.org Run in Google Colab View on GitHub Download notebook See TF Hub models

TensorFlow Hub Object Detection Colab

Welcome to the TensorFlow Hub Object Detection Colab! This notebook will take you through the steps of running an "out-of-the-box" object detection model on images.

More models

This collection contains TF 2 object detection models that have been trained on the COCO 2017 dataset. Here you can find all object detection models that are currently hosted on tfhub.dev.

Imports and Setup

Let's start with the base imports.

[1]
```
# This Colab requires TF 2.5.
!pip install -U tensorflow>=2.5
```

[2]
```
import os
import pathlib

import matplotlib
import matplotlib.pyplot as plt

import io
import scipy.misc
import numpy as np
from six import BytesIO
from PIL import Image, ImageDraw, ImageFont
from six.moves.urllib.request import urlopen

import tensorflow as tf
import tensorflow_hub as hub

tf.get_logger().setLevel('ERROR')
```

Utilities

Run the following cell to create some utils that will be needed later:

- Helper method to load an image
- Map of Model Name to TF Hub handle
- List of tuples with Human Keypoints for the COCO 2017 dataset. This is needed for models with keypoints.

Run this!!

[3]
```
# @title Run this!!

def load_image_into_numpy_array(path):
  """Load an image from file into a numpy array.

  Puts image into numpy array to feed into tensorflow graph.
  Note that by convention we put it into a numpy array with shape
  (height, width, channels), where channels=3 for RGB.

  Args:
    path: the file path to the image

  Returns:
    uint8 numpy array with shape (img_height, img_width, 3)
  """
  image = None
  if(path.startswith('http')):
    response = urlopen(path)
    image_data = response.read()
    image_data = BytesIO(image_data)
    image = Image.open(image_data)
  else:
    image_data = tf.io.gfile.GFile(path, 'rb').read()
    image = Image.open(BytesIO(image_data))

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


ALL_MODELS = {
'CenterNet HourGlass104 512x512' : 'https://tfhub.dev/tensorflow/centernet/hourglass_512x51
'CenterNet HourGlass104 Keypoints 512x512' : 'https://tfhub.dev/tensorflow/centernet/hourg
'CenterNet HourGlass104 1024x1024' : 'https://tfhub.dev/tensorflow/centernet/hourglass_102
'CenterNet HourGlass104 Keypoints 1024x1024' : 'https://tfhub.dev/tensorflow/centernet/hou
'CenterNet Resnet50 V1 FPN 512x512' : 'https://tfhub.dev/tensorflow/centernet/resnet50v1_fp
```

```
  'CenterNet Resnet50 V1 FPN Keypoints 512x512' : 'https://tfhub.dev/tensorflow/centernet/res
  'CenterNet Resnet101 V1 FPN 512x512' : 'https://tfhub.dev/tensorflow/centernet/resnet101v1_
  'CenterNet Resnet50 V2 512x512' : 'https://tfhub.dev/tensorflow/centernet/resnet50v2_512x5
  'CenterNet Resnet50 V2 Keypoints 512x512' : 'https://tfhub.dev/tensorflow/centernet/resnet5
  'EfficientDet D0 512x512' : 'https://tfhub.dev/tensorflow/efficientdet/d0/1',
  'EfficientDet D1 640x640' : 'https://tfhub.dev/tensorflow/efficientdet/d1/1',
  'EfficientDet D2 768x768' : 'https://tfhub.dev/tensorflow/efficientdet/d2/1',
  'EfficientDet D3 896x896' : 'https://tfhub.dev/tensorflow/efficientdet/d3/1',
  'EfficientDet D4 1024x1024' : 'https://tfhub.dev/tensorflow/efficientdet/d4/1',
  'EfficientDet D5 1280x1280' : 'https://tfhub.dev/tensorflow/efficientdet/d5/1',
  'EfficientDet D6 1280x1280' : 'https://tfhub.dev/tensorflow/efficientdet/d6/1',
  'EfficientDet D7 1536x1536' : 'https://tfhub.dev/tensorflow/efficientdet/d7/1',
  'SSD MobileNet v2 320x320' : 'https://tfhub.dev/tensorflow/ssd_mobilenet_v2/2',
  'SSD MobileNet V1 FPN 640x640' : 'https://tfhub.dev/tensorflow/ssd_mobilenet_v1/fpn_640x640
  'SSD MobileNet V2 FPNLite 320x320' : 'https://tfhub.dev/tensorflow/ssd_mobilenet_v2/fpnlite
  'SSD MobileNet V2 FPNLite 640x640' : 'https://tfhub.dev/tensorflow/ssd_mobilenet_v2/fpnlite
  'SSD ResNet50 V1 FPN 640x640 (RetinaNet50)' : 'https://tfhub.dev/tensorflow/retinanet/resne
  'SSD ResNet50 V1 FPN 1024x1024 (RetinaNet50)' : 'https://tfhub.dev/tensorflow/retinanet/res
  'SSD ResNet101 V1 FPN 640x640 (RetinaNet101)' : 'https://tfhub.dev/tensorflow/retinanet/res
  'SSD ResNet101 V1 FPN 1024x1024 (RetinaNet101)' : 'https://tfhub.dev/tensorflow/retinanet/r
  'SSD ResNet152 V1 FPN 640x640 (RetinaNet152)' : 'https://tfhub.dev/tensorflow/retinanet/res
  'SSD ResNet152 V1 FPN 1024x1024 (RetinaNet152)' : 'https://tfhub.dev/tensorflow/retinanet/r
  'Faster R-CNN ResNet50 V1 640x640' : 'https://tfhub.dev/tensorflow/faster_rcnn/resnet50_v1_
  'Faster R-CNN ResNet50 V1 1024x1024' : 'https://tfhub.dev/tensorflow/faster_rcnn/resnet50_v
  'Faster R-CNN ResNet50 V1 800x1333' : 'https://tfhub.dev/tensorflow/faster_rcnn/resnet50_v1
  'Faster R-CNN ResNet101 V1 640x640' : 'https://tfhub.dev/tensorflow/faster_rcnn/resnet101_v
  'Faster R-CNN ResNet101 V1 1024x1024' : 'https://tfhub.dev/tensorflow/faster_rcnn/resnet101
  'Faster R-CNN ResNet101 V1 800x1333' : 'https://tfhub.dev/tensorflow/faster_rcnn/resnet101_
  'Faster R-CNN ResNet152 V1 640x640' : 'https://tfhub.dev/tensorflow/faster_rcnn/resnet152_v
  'Faster R-CNN ResNet152 V1 1024x1024' : 'https://tfhub.dev/tensorflow/faster_rcnn/resnet152
  'Faster R-CNN ResNet152 V1 800x1333' : 'https://tfhub.dev/tensorflow/faster_rcnn/resnet152_
  'Faster R-CNN Inception ResNet V2 640x640' : 'https://tfhub.dev/tensorflow/faster_rcnn/ince
  'Faster R-CNN Inception ResNet V2 1024x1024' : 'https://tfhub.dev/tensorflow/faster_rcnn/ir
  'Mask R-CNN Inception ResNet V2 1024x1024' : 'https://tfhub.dev/tensorflow/mask_rcnn/incept
}

IMAGES_FOR_TEST = {
  'Beach' : 'models/research/object_detection/test_images/image2.jpg',
  'Dogs' : 'models/research/object_detection/test_images/image1.jpg',
  # By Heiko Gorski, Source: https://commons.wikimedia.org/wiki/File:Naxos_Taverna.jpg
  'Naxos Taverna' : 'https://upload.wikimedia.org/wikipedia/commons/6/60/Naxos_Taverna.jpg',
  # Source: https://commons.wikimedia.org/wiki/File:The_Coleoptera_of_the_British_islands_(
  'Beatles' : 'https://upload.wikimedia.org/wikipedia/commons/1/1b/The_Coleoptera_of_the_Br
  # By Américo Toledano, Source: https://commons.wikimedia.org/wiki/File:Biblioteca_Maim%C3
  'Phones' : 'https://upload.wikimedia.org/wikipedia/commons/thumb/0/0d/Biblioteca_Maim%C3%
  # Source: https://commons.wikimedia.org/wiki/File:The_smaller_British_birds_(8053836633).
  'Birds' : 'https://upload.wikimedia.org/wikipedia/commons/0/09/The_smaller_British_birds_
}

COCO17_HUMAN_POSE_KEYPOINTS = [(0, 1),
 (0, 2),
 (1, 3),
 (2, 4),
 (0, 5),
 (0, 6),
 (5, 7),
 (7, 9),
 (6, 8),
 (8, 10),
 (5, 6),
 (5, 11),
 (6, 12),
 (11, 12),
 (11, 13),
 (13, 15),
 (12, 14),
 (14, 16)]
```

Visualization tools

To visualize the images with the proper detected boxes, keypoints and segmentation, we will use the TensorFlow Object Detection API. To install it we will clone the repo.

```
[4] # Clone the tensorflow models repository
    !git clone --depth 1 https://github.com/tensorflow/models
```

```
Cloning into 'models'...
remote: Enumerating objects: 2824, done.
remote: Counting objects: 100% (2824/2824), done.
remote: Compressing objects: 100% (2354/2354), done.
remote: Total 2824 (delta 722), reused 1273 (delta 435), pack-reused 0
Receiving objects: 100% (2824/2824), 32.82 MiB | 23.80 MiB/s, done.
Resolving deltas: 100% (722/722), done.
```

Intalling the Object Detection API

```
[5] %%bash
    sudo apt install -y protobuf-compiler
    cd models/research/
    protoc object_detection/protos/*.proto --python_out=.
    cp object_detection/packages/tf2/setup.py .
    python -m pip install .
```

```
Requirement already satisfied: typeguard>=2.7 in /usr/local/lib/python3.7/dist-packages (from tensorflow-addons->tf-models-official>=2.5.1->object-detection==0.1) (2.7.1)
Requirement already satisfied: importlib-resources in /usr/local/lib/python3.7/dist-packages (from tensorflow-datasets->tf-models-official>=2.5.1->object-detection==0.1) (5.2.0)
Requirement already satisfied: tensorflow-metadata in /usr/local/lib/python3.7/dist-packages (from tensorflow-datasets->tf-models-official>=2.5.1->object-detection==0.1) (1.1.0)
Requirement already satisfied: promise in /usr/local/lib/python3.7/dist-packages (from tensorflow-datasets->tf-models-official>=2.5.1->object-detection==0.1) (2.3)
Requirement already satisfied: attrs>=18.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow-datasets->tf-models-official>=2.5.1->object-detection==0.1) (21.2.0)
Building wheels for collected packages: object-detection, py-cpuinfo, avro-python3, dill, future, seqeval
  Building wheel for object-detection (setup.py): started
  Building wheel for object-detection (setup.py): finished with status 'done'
  Created wheel for object-detection: filename=object_detection-0.1-py3-none-any.whl size=1660307 sha256=fe32849a1906c842d31b709b4d27ae6e53d46066171c94071216f99eaa34c5b5
  Stored in directory: /tmp/pip-ephem-wheel-cache-_6gcvya7/wheels/fa/a4/d2/e9a5057e414fd46c8e543d2706cd836d64e1fcd9eccceb2329
  Building wheel for py-cpuinfo (setup.py): started
  Building wheel for py-cpuinfo (setup.py): finished with status 'done'
```

```
  Created wheel for py-cpuinfo: filename=py_cpuinfo-8.0.0-py3-none-any.whl size=22257 sha256=9929665fbc95ab0ce098490a41db654e88df5ae10757a1b34d47345196009ebe
  Stored in directory: /root/.cache/pip/wheels/d2/f1/1f/041add21dc9c4220157f1bd2bd6afe1f1a49524c3396b94401
  Building wheel for avro-python3 (setup.py): started
  Building wheel for avro-python3 (setup.py): finished with status 'done'
  Created wheel for avro-python3: filename=avro_python3-1.9.2.1-py3-none-any.whl size=43512 sha256=1274f40b3918da04185b5b525ad275d6dc3524456048f6d19611f5aed9c7dbd0
  Stored in directory: /root/.cache/pip/wheels/bc/49/5f/fdb5b9d85055c478213e0158ac122b596816149a02d82e0ab1
  Building wheel for dill (setup.py): started
  Building wheel for dill (setup.py): finished with status 'done'
  Created wheel for dill: filename=dill-0.3.1.1-py3-none-any.whl size=78544 sha256=c49048bce9f07c986e219726603c623b2cd51584ab0ff62f12dc797283ee5b85
  Stored in directory: /root/.cache/pip/wheels/a4/61/fd/c57e374e580aa78a45ed78d5859b3a44436af17e22ca53284f
  Building wheel for future (setup.py): started
  Building wheel for future (setup.py): finished with status 'done'
  Created wheel for future: filename=future-0.18.2-py3-none-any.whl size=491070 sha256=70641466a2f6a60fce2db7d2497d86710e1bd64a3be40fafda8ee27c7622690d
  Stored in directory: /root/.cache/pip/wheels/56/b0/fe/4410d17b32f1f0c3cf54cdfb2bc04d7b4b8f4ae377e2229ba0
  Building wheel for seqeval (setup.py): started
  Building wheel for seqeval (setup.py): finished with status 'done'
  Created wheel for seqeval: filename=seqeval-1.2.2-py3-none-any.whl size=16181 sha256=baa7b1900490dd9e0466808db6b23c76df1aa7ddc7d31880fae32edf533549a3
  Stored in directory: /root/.cache/pip/wheels/05/96/ee/7cac4e74f3b19e3158dce26a20a1c86b3533c43ec72a549fd7
Successfully built object-detection py-cpuinfo avro-python3 dill future seqeval
Installing collected packages: requests, portalocker, future, dill, tf-slim, tensorflow-model-optimization, tensorflow-addons, seqeval, sentencepiece, sacrebleu, pyyaml, py-cpuinfo, openc
  Attempting uninstall: requests
    Found existing installation: requests 2.23.0
    Uninstalling requests-2.23.0:
      Successfully uninstalled requests-2.23.0
  Attempting uninstall: future
    Found existing installation: future 0.16.0
    Uninstalling future-0.16.0:
      Successfully uninstalled future-0.16.0
  Attempting uninstall: dill
    Found existing installation: dill 0.3.4
    Uninstalling dill-0.3.4:
      Successfully uninstalled dill-0.3.4
  Attempting uninstall: pyyaml
    Found existing installation: PyYAML 3.13
    Uninstalling PyYAML-3.13:
      Successfully uninstalled PyYAML-3.13
Successfully installed apache-beam-2.31.0 avro-python3-1.9.2.1 dill-0.3.1.1 fastavro-1.4.4 future-0.18.2 hdfs-2.6.0 lvis-0.5.3 object-detection-0.1 opencv-python-headless-4.5.3.56 portalc

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

  DEPRECATION: A future pip version will change local packages to be built in-place without first copying to a temporary directory. We recommend you use --use-feature=in-tree-build to tes
    pip 21.3 will remove support for this functionality. You can find discussion regarding this at https://github.com/pypa/pip/issues/7555.
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
multiprocess 0.70.12.2 requires dill>=0.3.4, but you have dill 0.3.1.1 which is incompatible.
google-colab 1.0.0 requires requests~=2.23.0, but you have requests 2.26.0 which is incompatible.
datascience 0.10.6 requires folium==0.2.1, but you have folium 0.8.3 which is incompatible.
```

Now we can import the dependencies we will need later

[6]
```
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.utils import ops as utils_ops

%matplotlib inline
```

Load label map data (for plotting).

Label maps correspond index numbers to category names, so that when our convolution network predicts `5`, we know that this corresponds to `airplane`. Here we use internal utility functions, but anything that returns a dictionary mapping integers to appropriate string labels would be fine.

We are going, for simplicity, to load from the repository that we loaded the Object Detection API code

[7]
```
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)
```

Build a detection model and load pre-trained model weights

Here we will choose which Object Detection model we will use. Select the architecture and it will be loaded automatically. If you want to change the model to try other architectures later, just change the next cell and execute following ones.

Tip: if you want to read more details about the selected model, you can follow the link (model handle) and read additional documentation on TF Hub. After you select a model, we will print the handle to make it easier.

[8] Model Selection

model_display_name: CenterNet HourGlass104 Keypoints 512x512

Show code

```
Selected model:CenterNet HourGlass104 Keypoints 512x512
Model Handle at TensorFlow Hub: https://tfhub.dev/tensorflow/centernet/hourglass_512x512_kpts/1
```

Loading the selected model from TensorFlow Hub

Here we just need the model handle that was selected and use the Tensorflow Hub library to load it to memory.

[9]
```
print('loading model...')
hub_model = hub.load(model_handle)
print('model loaded!')
```

```
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_207976) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_53564) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_26561) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_223696) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_48396) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_48841) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_221176) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_residual_block_69_layer_call_and_return_conditional_losses_67266) with ops with custom gradients. Will likely fail if a gradient is requeste
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_50176) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_50850) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_218176) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_223096) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_50405) with ops with custom gradients. Will likely fail if a gradient is requested.
```

```
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_37272) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_210016) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_209536) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_219016) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_47519) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_54918) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_219976) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_31888) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_210136) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_54009) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_211816) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_46629) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_23675) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_convolutional_block_71_layer_call_and_return_conditional_losses_203998) with ops with custom gradients. Will likely fail if a gradient is re
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_43527) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_40412) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_205576) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_214336) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_218776) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_22569) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_207376) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_45294) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_221416) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_212056) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_32994) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_213736) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_center_net_hourglass_feature_extractor_layer_call_and_return_conditional_losses_139752) with ops with custom gradients. Will likely fail if
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_213856) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_207736) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_43082) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_216376) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_43743) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_43298) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_25226) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_33884) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_26790) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_211336) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_47735) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_30108) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_219256) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_213976) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_218896) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_55827) with ops with custom gradients. Will likely fail if a gradient is requested.
WARNING:absl:Importing a function (__inference_batchnorm_layer_call_and_return_conditional_losses_56488) with ops with custom gradients. Will likely fail if a gradient is requested.
model loaded!
```

Loading an image

Let's try the model on a simple image. To help with this, we provide a list of test images.

Here are some simple things to try out if you are curious:

- Try running inference on your own images, just upload them to colab and load the same way it's done in the cell below.
- Modify some of the input images and see if detection still works. Some simple things to try out here include flipping the image horizontally, or converting to grayscale (note that we still expect the input image to have 3 channels).

Be careful: when using images with an alpha channel, the model expect 3 channels images and the alpha will count as a 4th.

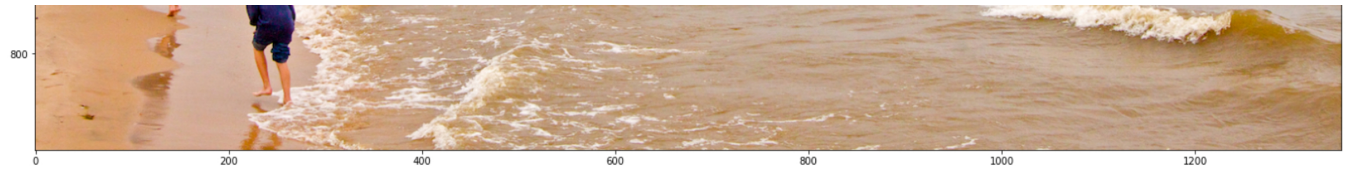
[10] Image Selection (don't forget to execute the cell!)

selected_image: Beach

flip_image_horizontally:

convert_image_to_grayscale:

Show code



Doing the inference

To do the inference we just need to call our TF Hub loaded model.

Things you can try:

- Print out `result['detection_boxes']` and try to match the box locations to the boxes in the image. Notice that coordinates are given in normalized form (i.e., in the interval [0, 1]).
- inspect other output keys present in the result. A full documentation can be seen on the models documentation page (pointing your browser to the model handle printed earlier)

```
[11] # running inference
     results = hub_model(image_np)

     # different object detection models have additional results
     # all of them are explained in the documentation
     result = {key:value.numpy() for key,value in results.items()}
     print(result.keys())
```

```
dict_keys(['num_detections', 'detection_keypoint_scores', 'detection_keypoints', 'detection_classes', 'detection_boxes', 'detection_scores'])
```

Visualizing the results

Here is where we will need the TensorFlow Object Detection API to show the squares from the inference step (and the keypoints when available).

the full documentation of this method can be seen here

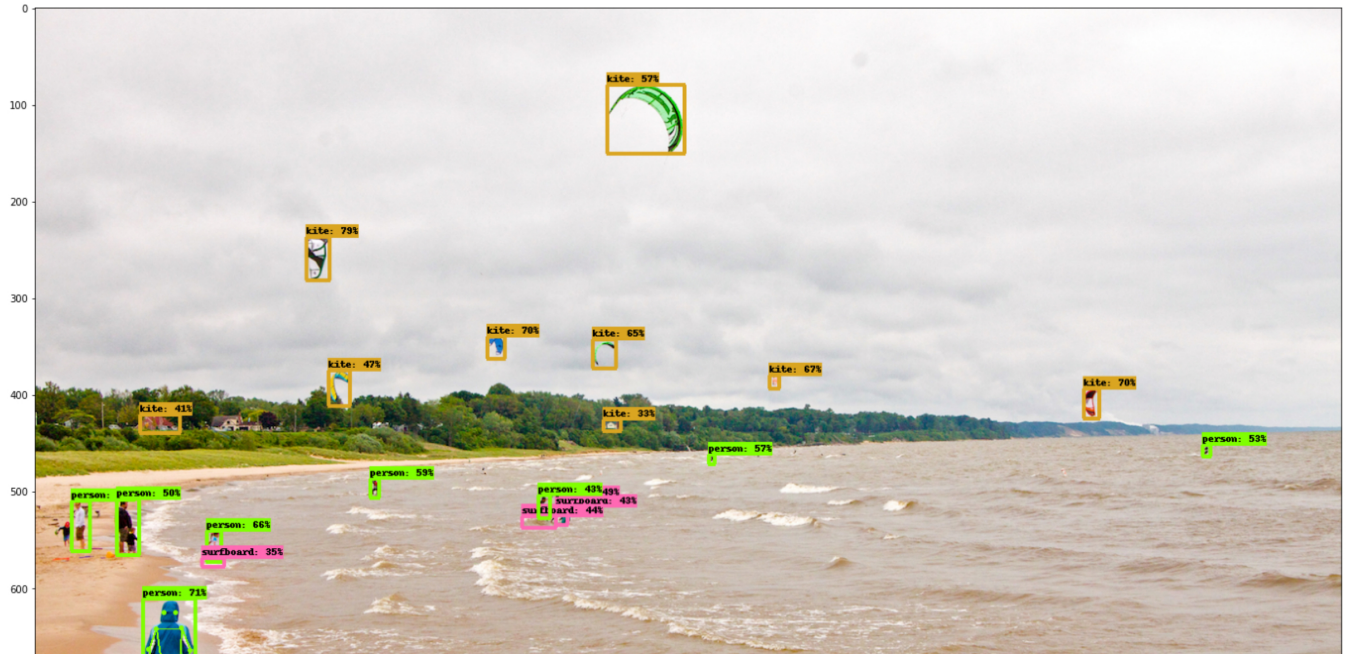
Here you can, for example, set `min_score_thresh` to other values (between 0 and 1) to allow more detections in or to filter out more detections.

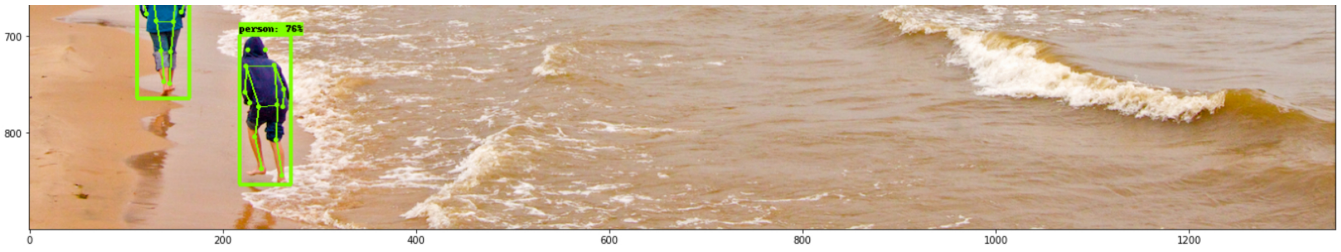
```
[12] label_id_offset = 0
     image_np_with_detections = image_np.copy()

     # Use keypoints if available in detections
     keypoints, keypoint_scores = None, None
     if 'detection_keypoints' in result:
       keypoints = result['detection_keypoints'][0]
       keypoint_scores = result['detection_keypoint_scores'][0]

     viz_utils.visualize_boxes_and_labels_on_image_array(
           image_np_with_detections[0],
           result['detection_boxes'][0],
           (result['detection_classes'][0] + label_id_offset).astype(int),
           result['detection_scores'][0],
           category_index,
           use_normalized_coordinates=True,
           max_boxes_to_draw=200,
           min_score_thresh=.30,
           agnostic_mode=False,
           keypoints=keypoints,
           keypoint_scores=keypoint_scores,
           keypoint_edges=COCO17_HUMAN_POSE_KEYPOINTS)

     plt.figure(figsize=(24,32))
     plt.imshow(image_np_with_detections[0])
     plt.show()
```





[Optional]

Among the available object detection models there's Mask R-CNN and the output of this model allows instance segmentation.

To visualize it we will use the same method we did before but adding an aditional parameter:

`instance_masks=output_dict.get('detection_masks_reframed', None)`

```
# Handle models with masks:
image_np_with_mask = image_np.copy()

if 'detection_masks' in result:
  # we need to convert np.arrays to tensors
  detection_masks = tf.convert_to_tensor(result['detection_masks'][0])
  detection_boxes = tf.convert_to_tensor(result['detection_boxes'][0])

  # Reframe the the bbox mask to the image size.
  detection_masks_reframed = utils_ops.reframe_box_masks_to_image_masks(
            detection_masks, detection_boxes,
              image_np.shape[1], image_np.shape[2])
  detection_masks_reframed = tf.cast(detection_masks_reframed > 0.5,
                                     tf.uint8)
  result['detection_masks_reframed'] = detection_masks_reframed.numpy()

viz_utils.visualize_boxes_and_labels_on_image_array(
      image_np_with_mask[0],
      result['detection_boxes'][0],
      (result['detection_classes'][0] + label_id_offset).astype(int),
      result['detection_scores'][0],
      category_index,
      use_normalized_coordinates=True,
      max_boxes_to_draw=200,
      min_score_thresh=.30,
      agnostic_mode=False,
      instance_masks=result.get('detection_masks_reframed', None),
      line_thickness=8)

plt.figure(figsize=(24,32))
plt.imshow(image_np_with_mask[0])
plt.show()
```

✓ 6s completed at 2:44 PM