Eager Few Shot Object Detection Colab

Welcome to the Eager Few Shot Object Detection Colab --- in this colab we demonstrate fine tuning of a (TF2 friendly) RetinaNet architecture on very few examples of a novel class after initializing from a pre-trained COCO checkpoint. Training runs in eager mode.

Estimated time to run through this colab (with GPU): < 5 minutes.

Imports

[1] `!pip install -U --pre tensorflow=="2.2.0"`

```
Requirement already satisfied: wrapt>=1.11.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0) (1.12.1)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0) (1.1.0)
Requirement already satisfied: google-pasta>=0.1.8 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0) (0.2.0)
Requirement already satisfied: grpcio>=1.8.6 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0) (1.34.1)
Collecting h5py<2.11.0,>=2.10.0
  Downloading h5py-2.10.0-cp37-cp37m-manylinux1_x86_64.whl (2.9 MB)
     |████████████████████████████████| 2.9 MB 32.9 MB/s
Collecting tensorboard<2.3.0,>=2.2.0
  Downloading tensorboard-2.2.2-py3-none-any.whl (3.0 MB)
     |████████████████████████████████| 3.0 MB 41.1 MB/s
Collecting tensorflow-estimator<2.3.0,>=2.2.0
  Downloading tensorflow_estimator-2.2.0-py2.py3-none-any.whl (454 kB)
     |████████████████████████████████| 454 kB 55.1 MB/s
Requirement already satisfied: absl-py>=0.7.0 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0) (0.12.0)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0) (3.3.0)
Requirement already satisfied: scipy==1.4.1 in /usr/local/lib/python3.7/dist-packages (from tensorflow==2.2.0) (1.4.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (1.8.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (3.3.4)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (0.4.4)
Requirement already satisfied: setuptools>=41.0.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (57.2.0)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (2.23.0)
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (1.0.1)
Requirement already satisfied: google-auth<2,>=1.6.3 in /usr/local/lib/python3.7/dist-packages (from tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (1.32.1)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.7/dist-packages (from google-auth<2,>=1.6.3->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-packages (from google-auth<2,>=1.6.3->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (4.7.2)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from google-auth<2,>=1.6.3->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (4.2.2)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.7/dist-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (1.
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from markdown>=2.6.8->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (4.6.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.7/dist-packages (from pyasn1-modules>=0.2.1->google-auth<2,>=1.6.3->tensorboard<2.3.0,>=2.2.0->tensorflow==2.
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (2021.5.30)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests<3,>=2.21.0->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (3.0.4)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.3.0,>=2.2.0->tensc
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->markdown>=2.6.8->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0) (3.5.0)
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->markdown>=2.6.8->tensorboard<2.3.0,>=2.2.0->tensorflow==2.2.0)
Installing collected packages: tensorflow-estimator, tensorboard, h5py, gast, tensorflow
  Attempting uninstall: tensorflow-estimator
    Found existing installation: tensorflow-estimator 2.5.0
    Uninstalling tensorflow-estimator-2.5.0:
      Successfully uninstalled tensorflow-estimator-2.5.0
  Attempting uninstall: tensorboard
    Found existing installation: tensorboard 2.5.0
    Uninstalling tensorboard-2.5.0:
      Successfully uninstalled tensorboard-2.5.0
  Attempting uninstall: h5py
    Found existing installation: h5py 3.1.0
    Uninstalling h5py-3.1.0:
      Successfully uninstalled h5py-3.1.0
  Attempting uninstall: gast
    Found existing installation: gast 0.4.0
    Uninstalling gast-0.4.0:
      Successfully uninstalled gast-0.4.0
  Attempting uninstall: tensorflow
    Found existing installation: tensorflow 2.5.0
    Uninstalling tensorflow-2.5.0:
      Successfully uninstalled tensorflow-2.5.0
Successfully installed gast-0.3.3 h5py-2.10.0 tensorboard-2.2.2 tensorflow-2.2.0 tensorflow-estimator-2.2.0
```

[2]
```
import os
import pathlib

# Clone the tensorflow models repository if it doesn't already exist
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
```

```
Cloning into 'models'...
remote: Enumerating objects: 2824, done.
remote: Counting objects: 100% (2824/2824), done.
remote: Compressing objects: 100% (2357/2357), done.
remote: Total 2824 (delta 721), reused 1262 (delta 432), pack-reused 0
Receiving objects: 100% (2824/2824), 32.82 MiB | 29.35 MiB/s, done.
Resolving deltas: 100% (721/721), done.
```

[3]
```
# Install the Object Detection API
%%bash
cd models/research/
protoc object_detection/protos/*.proto --python_out=.
cp object_detection/packages/tf2/setup.py .
python -m pip install .
```

```
  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=06f40987bf143dec24b4b1d668a41ab0c5175770f331c664b47014eeb1e7d849
  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=47c89aa6223de5259340619aaa9700cb8a1c300d76ed1f9c885f43127444b8f7
```

```
  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=4ad18db03475710d95fb9e387c0c94801e89951a373e6104426c3bfe1376dd36
  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, tensorflow-estimator, tensorboard, portalocker, h5py, gast, future, dill, tf-slim, tensorflow-model-optimization, tensorflow-addons, tensorflow, s
  Attempting uninstall: requests
    Found existing installation: requests 2.23.0
    Uninstalling requests-2.23.0:
      Successfully uninstalled requests-2.23.0
  Attempting uninstall: tensorflow-estimator
    Found existing installation: tensorflow-estimator 2.2.0
    Uninstalling tensorflow-estimator-2.2.0:
      Successfully uninstalled tensorflow-estimator-2.2.0
  Attempting uninstall: tensorboard
    Found existing installation: tensorboard 2.2.2
    Uninstalling tensorboard-2.2.2:
      Successfully uninstalled tensorboard-2.2.2
  Attempting uninstall: h5py
    Found existing installation: h5py 2.10.0
    Uninstalling h5py-2.10.0:
      Successfully uninstalled h5py-2.10.0
  Attempting uninstall: gast
    Found existing installation: gast 0.3.3
    Uninstalling gast-0.3.3:
      Successfully uninstalled gast-0.3.3
  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: tensorflow
    Found existing installation: tensorflow 2.2.0
    Uninstalling tensorflow-2.2.0:
      Successfully uninstalled tensorflow-2.2.0
  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 gast-0.4.0 h5py-3.1.0 hdfs-2.6.0 lvis-0.5.3 object-detection-0.1 opencv-python-hea
  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.
```

[4]
```
import matplotlib
import matplotlib.pyplot as plt

import os
import random
import io
import imageio
import glob
import scipy.misc
import numpy as np
from six import BytesIO
from PIL import Image, ImageDraw, ImageFont
from IPython.display import display, Javascript
from IPython.display import Image as IPyImage

import tensorflow as tf

from object_detection.utils import label_map_util
from object_detection.utils import config_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.utils import colab_utils
from object_detection.builders import model_builder

%matplotlib inline
```

Utilities

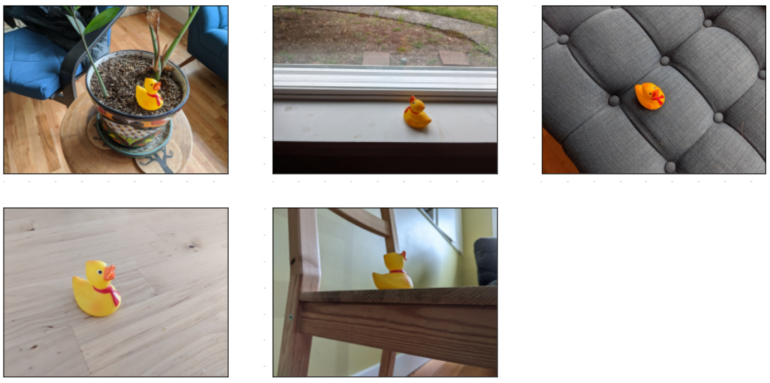
[5]
```
def load_image_into_numpy_array(path):
  """Load an image from file into a numpy array.

  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: a file path.

  Returns:
    uint8 numpy array with shape (img_height, img_width, 3)
  """
  img_data = tf.io.gfile.GFile(path, 'rb').read()
  image = Image.open(BytesIO(img_data))
  (im_width, im_height) = image.size
  return np.array(image.getdata()).reshape(
      (im_height, im_width, 3)).astype(np.uint8)

def plot_detections(image_np,
                    boxes,
                    classes,
                    scores,
                    category_index,
                    figsize=(12, 16),
                    image_name=None):
  """Wrapper function to visualize detections.

  Args:
    image_np: uint8 numpy array with shape (img_height, img_width, 3)
```

```
    boxes: a numpy array of shape [N, 4]
    classes: a numpy array of shape [N]. Note that class indices are 1-based,
      and match the keys in the label map.
    scores: a numpy array of shape [N] or None.  If scores=None, then
      this function assumes that the boxes to be plotted are groundtruth
      boxes and plot all boxes as black with no classes or scores.
    category_index: a dict containing category dictionaries (each holding
      category index `id` and category name `name`) keyed by category indices.
    figsize: size for the figure.
    image_name: a name for the image file.
  """
  image_np_with_annotations = image_np.copy()
  viz_utils.visualize_boxes_and_labels_on_image_array(
      image_np_with_annotations,
      boxes,
      classes,
      scores,
      category_index,
      use_normalized_coordinates=True,
      min_score_thresh=0.8)
  if image_name:
    plt.imsave(image_name, image_np_with_annotations)
  else:
    plt.imshow(image_np_with_annotations)
```

Rubber Ducky data

We will start with some toy (literally) data consisting of 5 images of a rubber ducky. Note that the coco dataset contains a number of animals, but notably, it does *not* contain rubber duckies (or even ducks for that matter), so this is a novel class.

```
[6]  # Load images and visualize
     train_image_dir = 'models/research/object_detection/test_images/ducky/train/'
     train_images_np = []
     for i in range(1, 6):
       image_path = os.path.join(train_image_dir, 'robertducky' + str(i) + '.jpg')
       train_images_np.append(load_image_into_numpy_array(image_path))

     plt.rcParams['axes.grid'] = False
     plt.rcParams['xtick.labelsize'] = False
     plt.rcParams['ytick.labelsize'] = False
     plt.rcParams['xtick.top'] = False
     plt.rcParams['xtick.bottom'] = False
     plt.rcParams['ytick.left'] = False
     plt.rcParams['ytick.right'] = False
     plt.rcParams['figure.figsize'] = [14, 7]

     for idx, train_image_np in enumerate(train_images_np):
       plt.subplot(2, 3, idx+1)
       plt.imshow(train_image_np)
     plt.show()
```

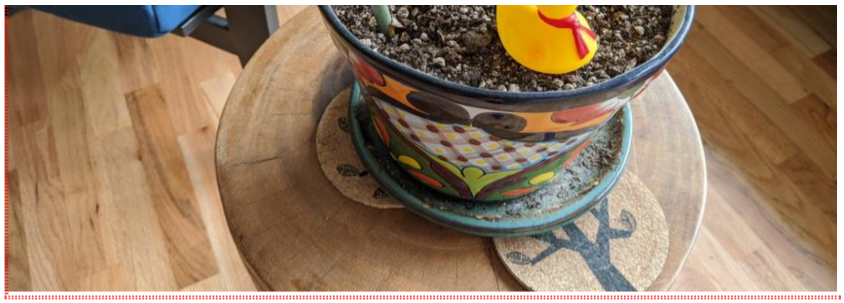


Annotate images with bounding boxes

In this cell you will annotate the rubber duckies --- draw a box around the rubber ducky in each image; click `next image` to go to the next image and `submit` when there are no more images.

If you'd like to skip the manual annotation step, we totally understand. In this case, simply skip this cell and run the next cell instead, where we've prepopulated the groundtruth with pre-annotated bounding boxes.

```
[7]  gt_boxes = []
     colab_utils.annotate(train_images_np, box_storage_pointer=gt_boxes)
```



prev image | next image | undo bbox | delete all

submit

In case you didn't want to label...

Run this cell only if you didn't annotate anything above and would prefer to just use our preannotated boxes. Don't forget to uncomment.

```
[8]  # gt_boxes = [
     #             np.array([[0.436, 0.591, 0.629, 0.712]], dtype=np.float32),
     #             np.array([[0.539, 0.583, 0.73, 0.71]], dtype=np.float32),
     #             np.array([[0.464, 0.414, 0.626, 0.548]], dtype=np.float32),
     #             np.array([[0.313, 0.308, 0.648, 0.526]], dtype=np.float32),
     #             np.array([[0.256, 0.444, 0.484, 0.629]], dtype=np.float32)
     # ]
```

Prepare data for training

Below we add the class annotations (for simplicity, we assume a single class in this colab; though it should be straightforward to extend this to handle multiple classes). We also convert everything to the format that the training loop below expects (e.g., everything converted to tensors, classes converted to one-hot representations, etc.).

```
[9]
     # By convention, our non-background classes start counting at 1.  Given
     # that we will be predicting just one class, we will therefore assign it a
     # `class id` of 1.
     duck_class_id = 1
     num_classes = 1

     category_index = {duck_class_id: {'id': duck_class_id, 'name': 'rubber_ducky'}}

     # Convert class labels to one-hot; convert everything to tensors.
     # The `label_id_offset` here shifts all classes by a certain number of indices;
     # we do this here so that the model receives one-hot labels where non-background
     # classes start counting at the zeroth index.  This is ordinarily just handled
     # automatically in our training binaries, but we need to reproduce it here.
     label_id_offset = 1
     train_image_tensors = []
     gt_classes_one_hot_tensors = []
     gt_box_tensors = []
     for (train_image_np, gt_box_np) in zip(
         train_images_np, gt_boxes):
       train_image_tensors.append(tf.expand_dims(tf.convert_to_tensor(
           train_image_np, dtype=tf.float32), axis=0))
       gt_box_tensors.append(tf.convert_to_tensor(gt_box_np, dtype=tf.float32))
       zero_indexed_groundtruth_classes = tf.convert_to_tensor(
           np.ones(shape=[gt_box_np.shape[0]], dtype=np.int32) - label_id_offset)
       gt_classes_one_hot_tensors.append(tf.one_hot(
           zero_indexed_groundtruth_classes, num_classes))
     print('Done prepping data.')
```

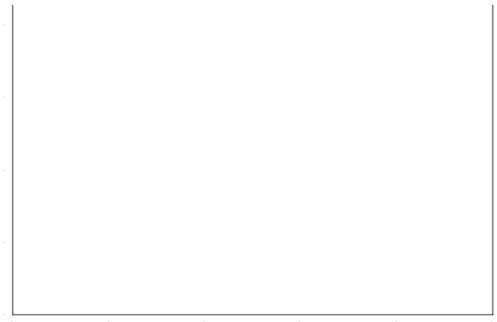
```
Done prepping data.
```

Let's just visualize the rubber duckies as a sanity check

```
[10] dummy_scores = np.array([1.0], dtype=np.float32)  # give boxes a score of 100%

     plt.figure(figsize=(30, 15))
     for idx in range(5):
       plt.subplot(2, 3, idx+1)
       plot_detections(
           train_images_np[idx],
           gt_boxes[idx],
           np.ones(shape=[gt_boxes[idx].shape[0]], dtype=np.int32),
           dummy_scores, category_index)
     plt.show()
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-10-e1b5d25f5b0f> in <module>()
      6   plot_detections(
      7       train_images_np[idx],
----> 8       gt_boxes[idx],
      9       np.ones(shape=[gt_boxes[idx].shape[0]], dtype=np.int32),
     10       dummy_scores, category_index)

IndexError: list index out of range
```

SEARCH STACK OVERFLOW



Create model and restore weights for all but last layer

In this cell we build a single stage detection architecture (RetinaNet) and restore all but the classification layer at the top (which will be automatically randomly initialized).

For simplicity, we have hardcoded a number of things in this colab for the specific RetinaNet architecture at hand (including assuming that the image size will always be 640x640), however it is not difficult to generalize to other model configurations.

```
[11] # Download the checkpoint and put it into models/research/object_detection/test_data/

     !wget http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_resnet50_v1_fpn_640x640_coco17_tpu-8.tar.gz
     !tar -xf ssd_resnet50_v1_fpn_640x640_coco17_tpu-8.tar.gz
     !mv ssd_resnet50_v1_fpn_640x640_coco17_tpu-8/checkpoint models/research/object_detection/test_data/
```

```
--2021-08-06 20:17:31--  http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_resnet50_v1_fpn_640x640_coco17_tpu-8.tar.gz
Resolving download.tensorflow.org (download.tensorflow.org)... 142.250.188.48, 2607:f8b0:4004:835::2010
Connecting to download.tensorflow.org (download.tensorflow.org)|142.250.188.48|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 244817203 (233M) [application/x-tar]
Saving to: ‘ssd_resnet50_v1_fpn_640x640_coco17_tpu-8.tar.gz’

ssd_resnet50_v1_fpn 100%[===================>] 233.48M   198MB/s    in 1.2s

2021-08-06 20:17:32 (198 MB/s) - ‘ssd_resnet50_v1_fpn_640x640_coco17_tpu-8.tar.gz’ saved [244817203/244817203]
```

```
[12] tf.keras.backend.clear_session()

     print('Building model and restoring weights for fine-tuning...', flush=True)
     num_classes = 1
     pipeline_config = 'models/research/object_detection/configs/tf2/ssd_resnet50_v1_fpn_640x640_coco17_tpu-8.config'
     checkpoint_path = 'models/research/object_detection/test_data/checkpoint/ckpt-0'

     # Load pipeline config and build a detection model.
     #
     # Since we are working off of a COCO architecture which predicts 90
     # class slots by default, we override the `num_classes` field here to be just
     # one (for our new rubber ducky class).
     configs = config_util.get_configs_from_pipeline_file(pipeline_config)
     model_config = configs['model']
     model_config.ssd.num_classes = num_classes
     model_config.ssd.freeze_batchnorm = True
     detection_model = model_builder.build(
           model_config=model_config, is_training=True)

     # Set up object-based checkpoint restore --- RetinaNet has two prediction
     # `heads` --- one for classification, the other for box regression.  We will
     # restore the box regression head but initialize the classification head
     # from scratch (we show the omission below by commenting out the line that
     # we would add if we wanted to restore both heads)
     fake_box_predictor = tf.compat.v2.train.Checkpoint(
         _base_tower_layers_for_heads=detection_model._box_predictor._base_tower_layers_for_heads,
         # _prediction_heads=detection_model._box_predictor._prediction_heads,
         #    (i.e., the classification head that we *will not* restore)
         _box_prediction_head=detection_model._box_predictor._box_prediction_head,
         )
     fake_model = tf.compat.v2.train.Checkpoint(
               _feature_extractor=detection_model._feature_extractor,
               _box_predictor=fake_box_predictor)
     ckpt = tf.compat.v2.train.Checkpoint(model=fake_model)
     ckpt.restore(checkpoint_path).expect_partial()

     # Run model through a dummy image so that variables are created
     image, shapes = detection_model.preprocess(tf.zeros([1, 640, 640, 3]))
     prediction_dict = detection_model.predict(image, shapes)
     _ = detection_model.postprocess(prediction_dict, shapes)
     print('Weights restored!')
```

```
Building model and restoring weights for fine-tuning...
Weights restored!
```

Eager mode custom training loop

```
[13] tf.keras.backend.set_learning_phase(True)

     # These parameters can be tuned; since our training set has 5 images
     # it doesn't make sense to have a much larger batch size, though we could
     # fit more examples in memory if we wanted to.
     batch_size = 4
     learning_rate = 0.01
     num_batches = 100

     # Select variables in top layers to fine-tune.
     trainable_variables = detection_model.trainable_variables
```

```
to_fine_tune = []
prefixes_to_train = [
  'WeightSharedConvolutionalBoxPredictor/WeightSharedConvolutionalBoxHead',
  'WeightSharedConvolutionalBoxPredictor/WeightSharedConvolutionalClassHead']
for var in trainable_variables:
  if any([var.name.startswith(prefix) for prefix in prefixes_to_train]):
    to_fine_tune.append(var)

# Set up forward + backward pass for a single train step.
def get_model_train_step_function(model, optimizer, vars_to_fine_tune):
  """Get a tf.function for training step."""

  # Use tf.function for a bit of speed.
  # Comment out the tf.function decorator if you want the inside of the
  # function to run eagerly.
  @tf.function
  def train_step_fn(image_tensors,
                    groundtruth_boxes_list,
                    groundtruth_classes_list):
    """A single training iteration.

    Args:
      image_tensors: A list of [1, height, width, 3] Tensor of type tf.float32.
        Note that the height and width can vary across images, as they are
        reshaped within this function to be 640x640.
      groundtruth_boxes_list: A list of Tensors of shape [N_i, 4] with type
        tf.float32 representing groundtruth boxes for each image in the batch.
      groundtruth_classes_list: A list of Tensors of shape [N_i, num_classes]
        with type tf.float32 representing groundtruth boxes for each image in
        the batch.

    Returns:
      A scalar tensor representing the total loss for the input batch.
    """
    shapes = tf.constant(batch_size * [[640, 640, 3]], dtype=tf.int32)
    model.provide_groundtruth(
        groundtruth_boxes_list=groundtruth_boxes_list,
        groundtruth_classes_list=groundtruth_classes_list)
    with tf.GradientTape() as tape:
      preprocessed_images = tf.concat(
          [detection_model.preprocess(image_tensor)[0]
           for image_tensor in image_tensors], axis=0)
      prediction_dict = model.predict(preprocessed_images, shapes)
      losses_dict = model.loss(prediction_dict, shapes)
      total_loss = losses_dict['Loss/localization_loss'] + losses_dict['Loss/classification_loss']
      gradients = tape.gradient(total_loss, vars_to_fine_tune)
      optimizer.apply_gradients(zip(gradients, vars_to_fine_tune))
    return total_loss

  return train_step_fn

optimizer = tf.keras.optimizers.SGD(learning_rate=learning_rate, momentum=0.9)
train_step_fn = get_model_train_step_function(
    detection_model, optimizer, to_fine_tune)

print('Start fine-tuning!', flush=True)
for idx in range(num_batches):
  # Grab keys for a random subset of examples
  all_keys = list(range(len(train_images_np)))
  random.shuffle(all_keys)
  example_keys = all_keys[:batch_size]

  # Note that we do not do data augmentation in this demo.  If you want a
  # a fun exercise, we recommend experimenting with random horizontal flipping
  # and random cropping :)
  gt_boxes_list = [gt_box_tensors[key] for key in example_keys]
  gt_classes_list = [gt_classes_one_hot_tensors[key] for key in example_keys]
  image_tensors = [train_image_tensors[key] for key in example_keys]

  # Training step (forward pass + backwards pass)
  total_loss = train_step_fn(image_tensors, gt_boxes_list, gt_classes_list)

  if idx % 10 == 0:
    print('batch ' + str(idx) + ' of ' + str(num_batches)
    + ', loss=' +  str(total_loss.numpy()), flush=True)

print('Done fine-tuning!')
```

```
Start fine-tuning!
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/backend.py:435: UserWarning: `tf.keras.backend.set_learning_phase` is deprecated and will be removed after 2020-10-11. To upda
  warnings.warn('`tf.keras.backend.set_learning_phase` is deprecated and '
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-13-b5207157937d> in <module>()
     75   # a fun exercise, we recommend experimenting with random horizontal flipping
     76   # and random cropping :)
---> 77   gt_boxes_list = [gt_box_tensors[key] for key in example_keys]
     78   gt_classes_list = [gt_classes_one_hot_tensors[key] for key in example_keys]
     79   image_tensors = [train_image_tensors[key] for key in example_keys]

<ipython-input-13-b5207157937d> in <listcomp>(.0)
     75   # a fun exercise, we recommend experimenting with random horizontal flipping
     76   # and random cropping :)
---> 77   gt_boxes_list = [gt_box_tensors[key] for key in example_keys]
     78   gt_classes_list = [gt_classes_one_hot_tensors[key] for key in example_keys]
     79   image_tensors = [train_image_tensors[key] for key in example_keys]

IndexError: list index out of range
```

SEARCH STACK OVERFLOW

Load test images and run inference with new model!

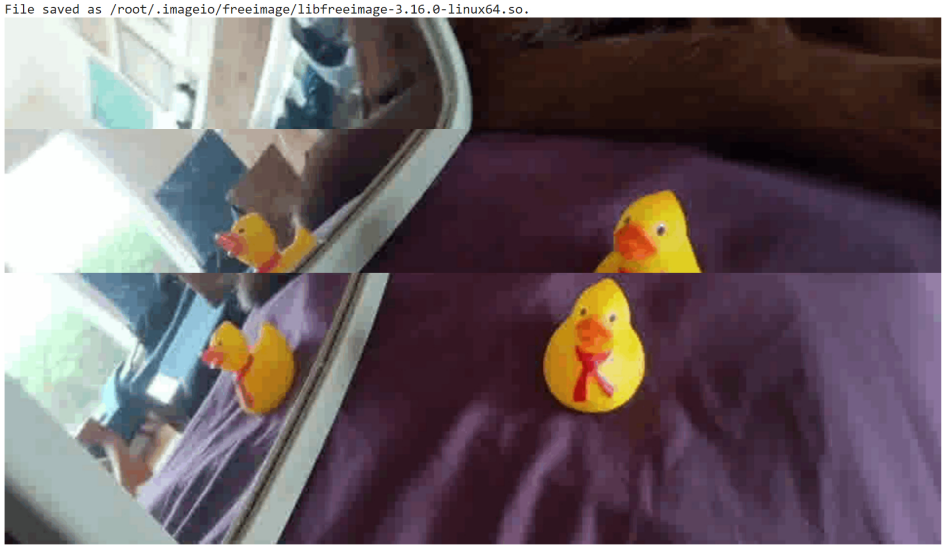
```
[14] test_image_dir = 'models/research/object_detection/test_images/ducky/test/'
     test_images_np = []
     for i in range(1, 50):
```

```
    image_path = os.path.join(test_image_dir, 'out' + str(i) + '.jpg')
    test_images_np.append(np.expand_dims(
        load_image_into_numpy_array(image_path), axis=0))

# Again, uncomment this decorator if you want to run inference eagerly
@tf.function
def detect(input_tensor):
  """Run detection on an input image.

  Args:
    input_tensor: A [1, height, width, 3] Tensor of type tf.float32.
      Note that height and width can be anything since the image will be
      immediately resized according to the needs of the model within this
      function.

  Returns:
    A dict containing 3 Tensors (`detection_boxes`, `detection_classes`,
      and `detection_scores`).
  """
  preprocessed_image, shapes = detection_model.preprocess(input_tensor)
  prediction_dict = detection_model.predict(preprocessed_image, shapes)
  return detection_model.postprocess(prediction_dict, shapes)

# Note that the first frame will trigger tracing of the tf.function, which will
# take some time, after which inference should be fast.

label_id_offset = 1
for i in range(len(test_images_np)):
  input_tensor = tf.convert_to_tensor(test_images_np[i], dtype=tf.float32)
  detections = detect(input_tensor)

  plot_detections(
      test_images_np[i][0],
      detections['detection_boxes'][0].numpy(),
      detections['detection_classes'][0].numpy().astype(np.uint32)
      + label_id_offset,
      detections['detection_scores'][0].numpy(),
      category_index, figsize=(15, 20), image_name="gif_frame_" + ('%02d' % i) + ".jpg")
```

13s

```
imageio.plugins.freeimage.download()

anim_file = 'duckies_test.gif'

filenames = glob.glob('gif_frame_*.jpg')
filenames = sorted(filenames)
last = -1
images = []
for filename in filenames:
  image = imageio.imread(filename)
  images.append(image)

imageio.mimsave(anim_file, images, 'GIF-FI', fps=5)

display(IPyImage(open(anim_file, 'rb').read()))
```

```
Imageio: 'libfreeimage-3.16.0-linux64.so' was not found on your computer; downloading it now.
Try 1. Download from https://github.com/imageio/imageio-binaries/raw/master/freeimage/libfreeimage-3.16.0-linux64.so (4.6 MB)
Downloading: 4830080/4830080 bytes (100.0%)
  Done
File saved as /root/.imageio/freeimage/libfreeimage-3.16.0-linux64.so.
```



✓ 13s completed at 3:24 PM