Simple Object Detection in Tensorflow

This lab will walk you through how to use object detection models available in Tensorflow Hub. In the following sections, you will:

- explore the Tensorflow Hub for object detection models
- load the models in your workspace
- preprocess an image for inference
- run inference on the models and inspect the output

Let's get started!

Imports

```
[1] import tensorflow as tf
    import tensorflow_hub as hub
    from PIL import Image
    from PIL import ImageOps
    import tempfile
    from six.moves.urllib.request import urlopen
    from six import BytesIO
```

Download the model from Tensorflow Hub

Tensorflow Hub is a repository of trained machine learning models which you can reuse in your own projects.

- You can see the domains covered here and its subcategories.
- For this lab, you will want to look at the image object detection subcategory.
- You can select a model to see more information about it and copy the URL so you can download it to your workspace.
- We selected a inception resnet version 2
- You can also modify this following cell to choose the other model that we selected, ssd mobilenet version 2

```
[2] # you can switch the commented lines here to pick the other model

    # inception resnet version 2
    module_handle = "https://tfhub.dev/google/faster_rcnn/openimages_v4/inception_resnet_v2/1"

    # You can choose ssd mobilenet version 2 instead and compare the results
    #module_handle = "https://tfhub.dev/google/openimages_v4/ssd/mobilenet_v2/1"
```

Load the model

Next, you'll load the model specified by the `module_handle`.

- This will take a few minutes to load the model.

```
[3] model = hub.load(module_handle)
```

```
INFO:tensorflow:Saver not created because there are no variables in the graph to restore
INFO:tensorflow:Saver not created because there are no variables in the graph to restore
```

Choose the default signature

Some models in the Tensorflow hub can be used for different tasks. So each model's documentation should show what *signature* to use when running the model.

- If you want to see if a model has more than one signature then you can do something like `print(hub.load(module_handle).signatures.keys())`. In your case, the models you will be using only have the `default` signature so you don't have to worry about other types.

```
[4] # take a look at the available signatures for this particular model
    model.signatures.keys()
```

```
KeysView(_SignatureMap({'default': <ConcreteFunction pruned(images) at 0x7F837605E310>}))
```

Please choose the 'default' signature for your object detector.

- For object detection models, its 'default' signature will accept a batch of image tensors and output a dictionary describing the objects detected, which is what you'll want here.

```
[5] detector = model.signatures['default']
```

download_and_resize_image

This function downloads an image specified by a given "url", pre-processes it, and then saves it to disk.

```
[6] def download_and_resize_image(url, new_width=256, new_height=256):
        '''
        Fetches an image online, resizes it and saves it locally.

        Args:
            url (string) -- link to the image
            new_width (int) -- size in pixels used for resizing the width of the image
            new_height (int) -- size in pixels used for resizing the length of the image

        Returns:
            (string) -- path to the saved image
        '''
```

```
    # create a temporary file ending with ".jpg"
    _, filename = tempfile.mkstemp(suffix=".jpg")

    # opens the given URL
    response = urlopen(url)

    # reads the image fetched from the URL
    image_data = response.read()

    # puts the image data in memory buffer
    image_data = BytesIO(image_data)

    # opens the image
    pil_image = Image.open(image_data)

    # resizes the image. will crop if aspect ratio is different.
    pil_image = ImageOps.fit(pil_image, (new_width, new_height), Image.ANTIALIAS)

    # converts to the RGB colorspace
    pil_image_rgb = pil_image.convert("RGB")

    # saves the image to the temporary file created earlier
    pil_image_rgb.save(filename, format="JPEG", quality=90)

    print("Image downloaded to %s." % filename)

    return filename
```

Download and preprocess an image

Now, using `download_and_resize_image` you can get a sample image online and save it locally.

- We've provided a URL for you, but feel free to choose another image to run through the object detector.
- You can use the original width and height of the image but feel free to modify it and see what results you get.

```
[7]  # You can choose a different URL that points to an image of your choice
     image_url = "https://upload.wikimedia.org/wikipedia/commons/f/fb/20130807_dublin014.JPG"

     # download the image and use the original height and width
     downloaded_image_path = download_and_resize_image(image_url, 3872, 2592)
```

```
Image downloaded to /tmp/tmp4n8kn7xa.jpg.
```

run_detector

This function will take in the object detection model `detector` and the path to a sample image, then use this model to detect objects and display its predicted class categories and detection boxes.

- run_detector uses `load_image` to convert the image into a tensor.

```
[8]  def load_img(path):
         '''
         Loads a JPEG image and converts it to a tensor.

         Args:
             path (string) -- path to a locally saved JPEG image

         Returns:
             (tensor) -- an image tensor
         '''

         # read the file
         img = tf.io.read_file(path)

         # convert to a tensor
         img = tf.image.decode_jpeg(img, channels=3)

         return img


     def run_detector(detector, path):
         '''
         Runs inference on a local file using an object detection model.

         Args:
             detector (model) -- an object detection model loaded from TF Hub
             path (string) -- path to an image saved locally
         '''

         # load an image tensor from a local file path
         img = load_img(path)

         # add a batch dimension in front of the tensor
         converted_img  = tf.image.convert_image_dtype(img, tf.float32)[tf.newaxis, ...]

         # run inference using the model
         result = detector(converted_img)

         # save the results in a dictionary
         result = {key:value.numpy() for key,value in result.items()}

         # print results
         print("Found %d objects." % len(result["detection_scores"]))

         print(result["detection_scores"])
         print(result["detection_class_entities"])
         print(result["detection_boxes"])
```

Run inference on the image

You can run your detector by calling the `run_detector` function. This will print the number of objects found followed by three lists:

- The detection scores of each object found (i.e. how confident the model is),
- The classes of each object found,
- The bounding boxes of each object

You will see how to overlay this information on the original image in the next sections and in this week's assignment!

```
# runs the object detection model and prints information about the objects found
run_detector(detector, downloaded_image_path)
```

```
[5.82192898e-01 3.64929765e-01 7.13880658e-01 4.84707862e-01]
[5.23547709e-01 7.49199331e-01 5.85378110e-01 7.65317559e-01]
[6.09156787e-01 4.26705897e-01 7.05165207e-01 4.87089008e-01]
[3.51368606e-01 9.74856079e-01 5.53130627e-01 9.98878717e-01]
[0.00000000e+00 8.11223328e-01 6.86410844e-01 9.97151256e-01]
[5.76297641e-01 3.57461751e-01 7.04812348e-01 4.40279901e-01]
[5.64892411e-01 3.63023102e-01 7.08650351e-01 4.16036338e-01]
[1.09374998e-02 2.33155154e-02 7.26522923e-01 4.21747833e-01]
[4.84686643e-01 4.10686046e-01 6.94686472e-01 4.63092834e-01]
[8.09777379e-02 3.84715289e-01 2.07808718e-01 4.11746383e-01]
[5.38284421e-01 6.03573740e-01 6.34776115e-01 6.34408653e-01]
[6.29844606e-01 6.14971519e-01 6.44933462e-01 6.25384450e-01]
[5.02758026e-01 3.82395953e-01 5.96146226e-01 4.12722319e-01]
[0.00000000e+00 1.24522885e-02 1.40193507e-01 2.47382112e-02]
[5.14441311e-01 7.47791588e-01 5.91985822e-01 7.66827345e-01]
[5.06182134e-01 5.00406921e-01 6.00681305e-01 5.23312032e-01]
[0.00000000e+00 2.11283579e-01 6.50794089e-01 4.34300780e-01]
[4.89451557e-01 4.54391301e-01 5.72340131e-01 4.76470768e-01]
[0.00000000e+00 7.06215978e-01 6.16998792e-01 8.66189659e-01]
[5.09172916e-01 4.16281193e-01 6.69304490e-01 4.59598720e-01]
[4.65173740e-03 8.03094208e-01 1.59853578e-01 8.40397060e-01]
[5.26151001e-01 5.68352938e-01 5.79440355e-01 5.82810223e-01]
[6.71924829e-01 9.40277696e-01 8.21276009e-01 9.89250779e-01]
[5.02770185e-01 3.73883098e-01 6.46991491e-01 4.12972301e-01]
[5.74243903e-01 2.67400861e-01 6.57769084e-01 3.20318550e-01]
[4.86056775e-01 4.44508791e-01 6.24788880e-01 4.73503351e-01]
[5.17248929e-01 7.56969213e-01 5.88517189e-01 7.71465480e-01]
[5.23374975e-01 5.57850122e-01 5.79139531e-01 5.73541582e-01]
[6.12461030e-01 4.27332461e-01 7.06080198e-01 4.88251865e-01]
[5.24124146e-01 5.61553180e-01 5.78385353e-01 5.80475152e-01]
[0.00000000e+00 2.44231746e-01 6.07754774e-02 2.93613434e-01]
[1.48921711e-02 2.14740215e-03 7.45441973e-01 2.59790719e-01]
[4.93236154e-01 9.23950195e-01 8.37110877e-01 9.97755051e-01]
[8.37683585e-03 2.42165715e-01 4.97285351e-02 2.83162564e-01]
[5.05334914e-01 3.60175252e-01 6.43561006e-01 3.91461760e-01]
[5.13099134e-01 5.23794115e-01 6.00504339e-01 5.42967975e-01]
[5.20421326e-01 6.00978673e-01 6.46124125e-01 6.34366393e-01]
[5.18224835e-01 5.03395557e-01 5.97548664e-01 5.22683859e-01]
[5.94199121e-01 3.61327976e-01 7.05465913e-01 4.15853351e-01]
[5.13256431e-01 6.79316938e-01 5.50533950e-01 6.92482173e-01]
[5.22302687e-01 5.36195457e-01 5.97564995e-01 5.53163290e-01]
[4.29876357e-01 8.28702271e-01 5.89928269e-01 8.64323139e-01]
[5.04884601e-01 3.89426976e-01 6.15080714e-01 4.19936091e-01]
[5.26588559e-01 6.27176881e-01 5.63299775e-01 6.53728902e-01]
[5.01304924e-01 3.64189029e-01 6.59964740e-01 4.03793275e-01]
[5.15171111e-01 6.24104798e-01 5.63795388e-01 6.58002079e-01]
[5.73137939e-01 2.66902655e-01 6.66162014e-01 3.18640232e-01]
[8.34235623e-02 4.07414347e-01 5.84092379e-01 5.58522940e-01]
[2.88196921e-01 4.77982452e-04 4.14364636e-01 3.65995839e-02]
[4.97272849e-01 4.55296665e-01 5.83817124e-01 4.77936029e-01]
[6.27168000e-01 3.61024052e-01 7.05996811e-01 4.09780174e-01]
[5.15861034e-01 3.80056977e-01 5.96893847e-01 4.11758274e-01]
[1.18098035e-02 3.08121800e-01 9.72859487e-02 3.25038970e-01]
[5.12501717e-01 6.23653352e-01 5.62422156e-01 6.57641888e-01]
[4.01003152e-01 8.85088801e-01 5.81281602e-01 9.39214468e-01]
[5.13853133e-01 5.29484570e-01 6.02009714e-01 5.52362800e-01]
[0.00000000e+00 1.00606047e-02 1.36156827e-01 3.16007212e-02]
[4.80426341e-01 6.20422781e-01 5.65284550e-01 6.60150290e-01]
[5.19355476e-01 3.61840397e-01 6.24995410e-01 3.84919673e-01]]
```

1m 1s completed at 1:59 PM