**Multi class vehicle classifier**

--------------------------------------------------------------------------------------------------------------------------------------

**Requirements**

**1. Python 3.5.x and Pip**

**2. TensorFlow**

**Note:** For Windows users, TensorFlow only supports version 3.5.x of Python.

--------------------------------------------------------------------------------------------------------------------------------------

**How to create your own image classifier**

**1. Install Python 3.5.x and Pip**

Install for Ubuntu

Run "sudo apt-get install python-pip python-dev" in terminal.

**2. Install TensorFlow**

Install for windows

Run "pip3 install --upgrade tensorflow" in terminal.

Install for Ubuntu

Run "pip3 install tensorflow" in terminal.

**3. Validate Your Installation**

For both users, invoke python from your shell as follows:

$ python # Ubuntu

C:\> python # Windows

Enter the following short program inside the python interactive shell:

>>> import tensorflow as tf

>>> hello = tf.constant('Hello, TensorFlow!')

>>> sess = tf.Session()

>>> print(sess.run(hello))

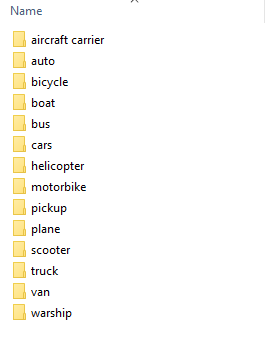
If the system outputs the following, then Tensorflow has been successfully installed:

Hello, TensorFlow!

**4. Prepare your training data**

Training data is what you want your computer to learn, to recognize the objects in the images. There is a super convenient Google plug-in called “Fatkun Batch Download Image” may help us downloading amount of images at once.

**5. Place the images into different folder for each different type as they are arranged in data folder.**



**6. Start Retraining**

At first, get the latest sample code from git TensorFlow repository. The sample code will be in "/tensorflow/tensorflow/examples/image\_retraining/retrain.py"

For training type the following command in terminal

python {$your-working\_directory}/retrain.py

--bottleneck\_dir=/{$your-working\_directory}/bottlenecks

--how\_many\_training\_steps 500

--model\_dir=/{$your-working\_directory}/inception

--output\_graph=/{$your-working\_directory}/retrained\_graph.pb

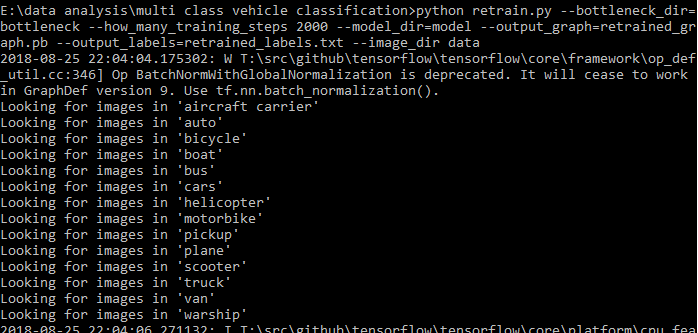
--output\_labels=/{$your-working\_directory}/retrained\_labels.txt

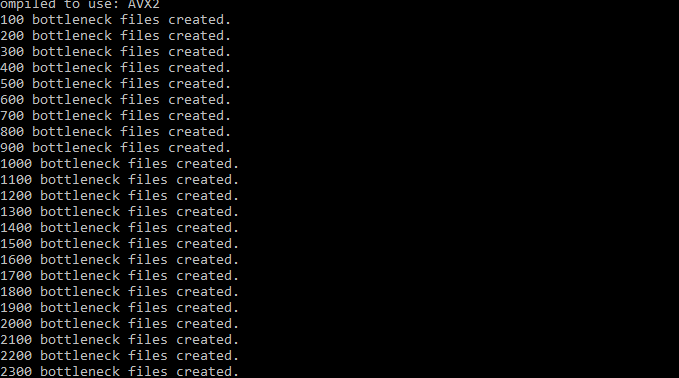
--image\_dir /{$your-working\_directory}/${your\_training\_data\_path}

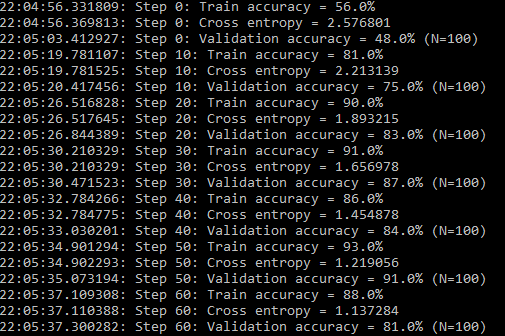
The command I used was:

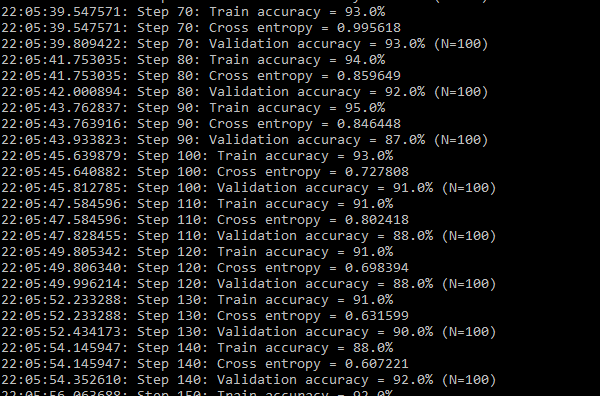
python retrain.py --bottleneck\_dir=bottleneck --how\_many\_training\_steps 2000 --model\_dir=model --output\_graph=retrained\_graph.pb --output\_labels=retrained\_labels.txt --image\_dir data

**Note:** Retraining will take at least 20 minutes.



****

****

****

**7. Test your image classifier**

Go to your work directory and create a file called "label\_image.py".

Type the following lines of code and save it.

import tensorflow as tf, sys

image\_path = sys.argv[1]

# Read in the image\_data

image\_data = tf.gfile.FastGFile(image\_path, 'rb').read()

# Loads label file, strips off carriage return

label\_lines = [line.rstrip() for line

in tf.gfile.GFile("D:/tensorflow\_work/retrained\_labels.txt")]

# Unpersists graph from file

with tf.gfile.FastGFile("D:/tensorflow\_work/retrained\_graph.pb", 'rb') as f:

graph\_def = tf.GraphDef()

graph\_def.ParseFromString(f.read())

\_ = tf.import\_graph\_def(graph\_def, name='')

# Feed the image\_data as input to the graph and get first prediction

with tf.Session() as sess:

softmax\_tensor = sess.graph.get\_tensor\_by\_name('final\_result:0')

predictions = sess.run(softmax\_tensor,

{'DecodeJpeg/contents:0': image\_data})

# Sort to show labels of first prediction in order of confidence

top\_k = predictions[0].argsort()[-len(predictions[0]):][::-1]

for node\_id in top\_k:

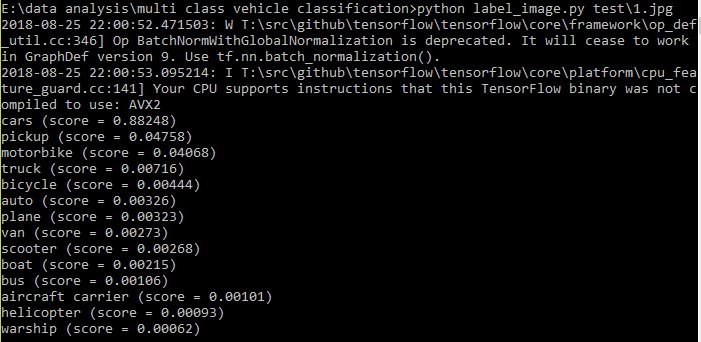
human\_string = label\_lines[node\_id]

score = predictions[0][node\_id]

print('%s (score = %.5f)' % (human\_string, score))

Run the command in terminal

"python label\_image.py Image\_to\_test\_path/Image\_name.extension"



--------------------------------------------------------------------------------------------------------------------------------------

**Notes:**

1. There should be at least 50 images of each type in the dataset for proper training of the model.

2. The model may take 20 minutes to 2 hours depending on the amount of data and configuration of the machine.

3. The above is made and tested on Python 3.5.2 only.