**Deploying retrained model on Google Vision Kit**

***Step 1***

It is a good idea to use Google Compute Cloud to train and compile the model before you can deploy it Google vision kit. This tutorial proved to be most useful

<https://towardsdatascience.com/running-jupyter-notebook-in-google-cloud-platform-in-15-min-61e16da34d52>

Train the model from scratch or in this case use transfer learning (MobileNet) to retrain on the custom set of pictures. It should be done on Ubuntu 16.04

*Setup:*

The package I have installed on my machine is Anaconda, very useful for machine learning purposes

**Run this command to get it:**

wget:<http://repo.continuum.io/archive/Anaconda3-4.0.0-Linux-x86_64.sh>

**And this command will install it:**

bash Anaconda3-4.0.0-Linux-x86\_64.sh

The “TensorFlow For Poet” tutorial I followed to retrain MobilNet to classify flowers is here:

<https://codelabs.developers.google.com/codelabs/tensorflow-for-poets/#0>

While you are on it be attentive to the resolution you specify for your retrained model architecture

input size: 160x160, depth multiplier = 0.5

*So this is basically going through tutorial*

git clone https://github.com/googlecodelabs/tensorflow-for-poets-2

This repo has a script that will enable us to train MobileNet on our pictures and extract bottlenecks for our case. We just have to put our pictures in folders with names of what is inside to tf\_files.

This time I am creating a model with higher resolution(but I tried 160 before that too)

Run this to set environment variables in terminal

IMAGE\_SIZE=160  
ARCHITECTURE="mobilenet\_0.50\_${IMAGE\_SIZE}"

Run the script in your terminal to start retraining, all scripts are in repo already:

python -m scripts.retrain \  
 --bottleneck\_dir=tf\_files/bottlenecks \  
 --model\_dir=tf\_files/models/ \  
 --summaries\_dir=tf\_files/training\_summaries/"${ARCHITECTURE}" \  
 --output\_graph=tf\_files/retrained\_graph.pb \  
 --output\_labels=tf\_files/retrained\_labels.txt \  
 --architecture="${ARCHITECTURE}" \  
 --image\_dir=tf\_files/**robotrack** \

Additionally, you can specify the name of output tensor (add one more parameter: --final\_tensor\_name=”your\_name”), you will need it later when you are compiling to .binaryproto file!!!

This runs for 4000 steps by default and it was pretty fast

The graph (.pb file) will be in the *tf\_files* folder

***Step 2***

Next, we need to compile frozen model in order to use it on Vision Bonnet. That should be done on Ubuntu 14.04 os.

Need to download the compiler for Bonnet

<https://dl.google.com/dl/aiyprojects/vision/bonnet_model_compiler_latest.tgz>

Unzip it with tar -xzf bonnet\_model\_compiler\_2018\_04\_26.tgz

and need to chmod u+x bonnet\_model\_compiler.par to run it.

Transfer retrained\_graph.pb to this 14.04 Ubuntu machine

Now run this script in terminal and you should get retrained\_graph in the current directory if all went well

./bonnet\_model\_compiler.par --debug \  
 --frozen\_graph\_path=./dumb.pb \  
 --output\_graph\_path=./dumb.binaryproto \  
 --input\_tensor\_name="input" \  
 --output\_tensor\_names="final\_result" \  
 --input\_tensor\_size=160

--output\_tensor\_names - that's where you put the name you have chosen earlier while retraining or by default it is “final\_result”

***Step 3***

Transfer compiled <\*.binaryproto> file to the PI.

name@local scp file.binaryproto pi@192.168.2.44:/home/pi/

And then ssh to the pi:

ssh pi@192.168.2.44

Password: raspberry

On the Pi:

sudo cp file.binaryproto /opt/aiy/models/

We need a script to infer compiled model. It can be real-time classification from a video stream or we can feed an image and receive classification.

This script takes one image as input and classifies it.

#!/usr/bin/env python3

import argparse

from PIL import Image

from aiy.vision import inference

from aiy.vision.models import utils

def process(result, labels):

"""Processes inference result and returns labels sorted by confidence."""

assert len(result.tensors) == 1

tensor = result.tensors['final\_result']

probs, shape = tensor.data, tensor.shape

assert shape.depth == len(labels)

#0.1 is a threshold, if the score is less then that confidence level is to low

pairs = [pair for pair in enumerate(probs) if pair[1] > 0.1]

pairs = sorted(pairs, key=lambda pair: pair[1], reverse=True)

pairs = pairs[0:3]

if (pairs):

print ( "detected: ".join([' %s (%.2f)' % (labels[index], prob) for index, prob in pairs]))

else:

print("nothing was detected")

def main():

parser = argparse.ArgumentParser()

parser.add\_argument('--input', '-i', dest='input', required=True)

args = parser.parse\_args()

model = inference.ModelDescriptor(

name='mobilenet\_224',

input\_shape=(1, 224, 224, 3),

input\_normalizer=(128.0, 128.0),

compute\_graph=utils.load\_compute\_graph('test.binaryproto'))

labels = ['daisy', 'dandelion', 'roses', 'sunflower', 'tulips']

print ("give me a sec...")

with inference.ImageInference(model) as inf:

img = Image.open(args.input)

process(inf.run(img),labels)

if \_\_name\_\_ == '\_\_main\_\_':

main()