This is a detailed description of the project, mainly about how I trained the 2 models with my own dataset and the outcome, a comparison between 2 models.

1. Image downloading

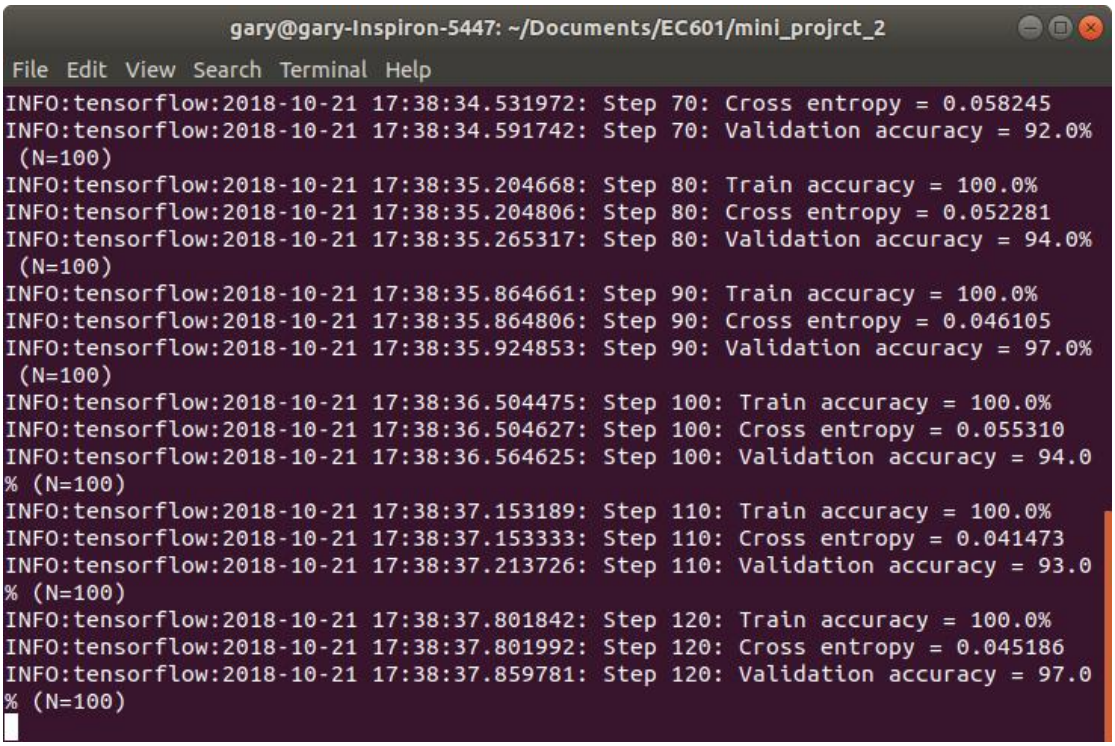
My objects are cups and bottles. I used the tool ‘googleimagesdownload’ to download images from google. The tool can be installed easily on ubuntu by apt-get install command. Type in the key word and the number of images then it will start to run. Be aware that not all of the images downloaded can be used. Some of them are broken and some of them are in strange formats. I downloaded 100 images for each object and manually deleted those useless ones. Therefore the number of images left in the dataset is around 80 each.

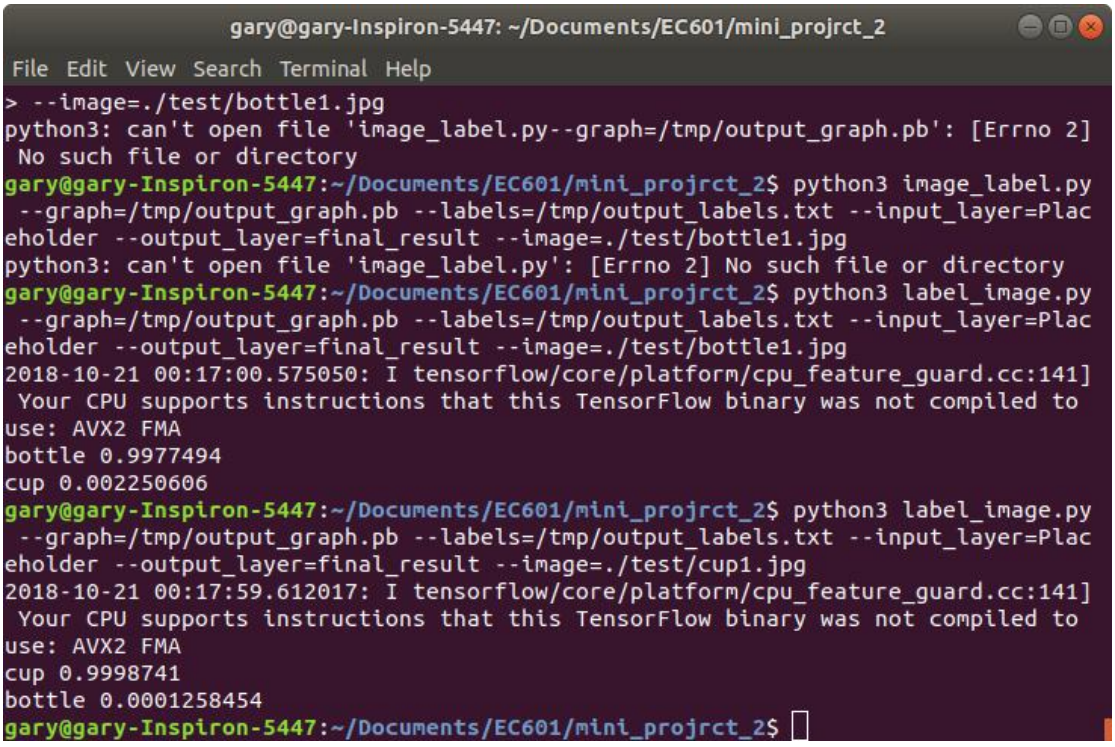
2. Classification model based on tensorflow

The instructions and python scripts of training this model are from the tensorflow tutorial. It is based on a pretrained model. First, download the python scripts ‘retrain.py’ and ‘label_image.py’. Then divide the dataset into 2 folders named ‘bottle’ and ‘cup’, and put them both into a folder(mine is ‘downloads’). Them run the command ‘python3 retrain.py --image_dir ./downloads’. This will identify the path of the dataset and it will take the names of the folders in ‘downloads’ as labels automatically. The training programme takes 80% of the images as training set, 10% as validation set and 10% as testing set for each class. It will generate a weight file named ‘output_graph.pb’ and a txt file ‘output_labels.txt’ to store the labels of classes. This model does not need the labeling process because it defines a random range from the center of images and consider them as the object.

After the training is over, we can use the command ‘python3 label_image.py --graph=/tmp/output_graph.pb --labels=/tmp/output_labels.txt --input_layer=Placeholder --output_layer=final_result --image=./test/cup1.jpg’ to see the result of the training. The command separately defines the path of the weight file and the label file, the image input and output layer and the path of the image to test.

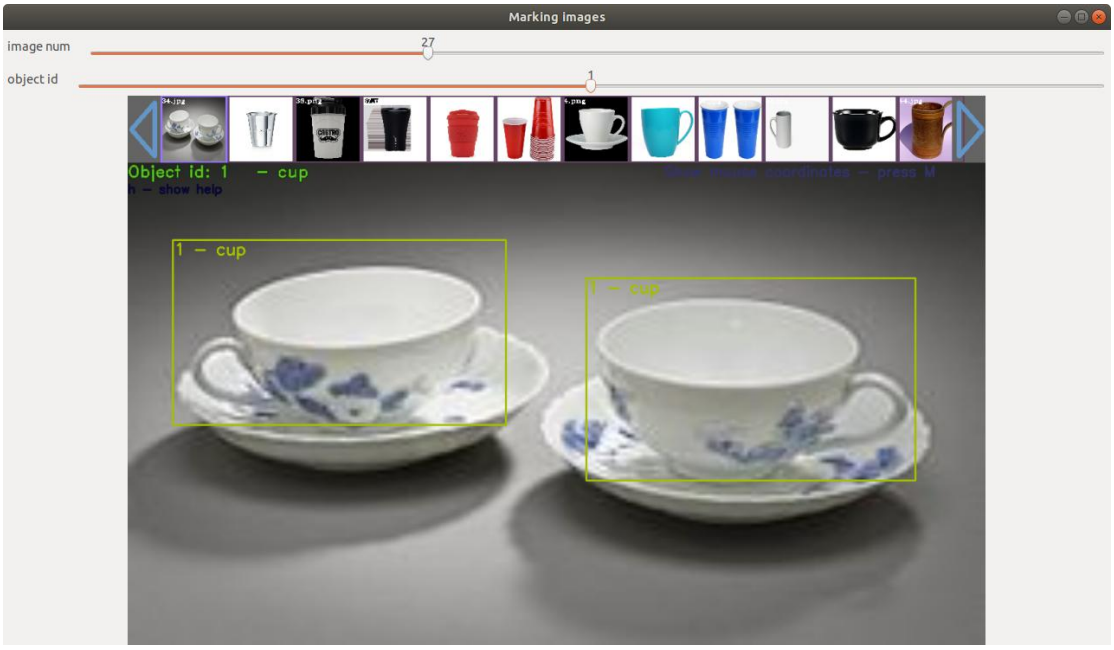
The 2 pictures attached below are screenshots of the training process and the outcome, which indicates a good object classification result.


gary@gary-Inspiron-5447: ~/Documents/EC601/mini_projrct_2
File Edit View Search Terminal Help
INFO:tensorflow:2018-10-21 17:38:34.531972: Step 70: Cross entropy = 0.058245
INFO:tensorflow:2018-10-21 17:38:34.591742: Step 70: Validation accuracy = 92.0%
(N=100)
INFO:tensorflow:2018-10-21 17:38:35.204668: Step 80: Train accuracy = 100.0%
INFO:tensorflow:2018-10-21 17:38:35.204806: Step 80: Cross entropy = 0.052281
INFO:tensorflow:2018-10-21 17:38:35.265317: Step 80: Validation accuracy = 94.0%
(N=100)
INFO:tensorflow:2018-10-21 17:38:35.864661: Step 90: Train accuracy = 100.0%
INFO:tensorflow:2018-10-21 17:38:35.864806: Step 90: Cross entropy = 0.046105
INFO:tensorflow:2018-10-21 17:38:35.924853: Step 90: Validation accuracy = 97.0%
(N=100)
INFO:tensorflow:2018-10-21 17:38:36.504475: Step 100: Train accuracy = 100.0%
INFO:tensorflow:2018-10-21 17:38:36.504627: Step 100: Cross entropy = 0.055310
INFO:tensorflow:2018-10-21 17:38:36.564625: Step 100: Validation accuracy = 94.0
% (N=100)
INFO:tensorflow:2018-10-21 17:38:37.153189: Step 110: Train accuracy = 100.0%
INFO:tensorflow:2018-10-21 17:38:37.153333: Step 110: Cross entropy = 0.041473
INFO:tensorflow:2018-10-21 17:38:37.213726: Step 110: Validation accuracy = 93.0
% (N=100)
INFO:tensorflow:2018-10-21 17:38:37.801842: Step 120: Train accuracy = 100.0%
INFO:tensorflow:2018-10-21 17:38:37.801992: Step 120: Cross entropy = 0.045186
INFO:tensorflow:2018-10-21 17:38:37.859781: Step 120: Validation accuracy = 97.0
% (N=100)


gary@gary-Inspiron-5447: ~/Documents/EC601/mini_projrct_2
File Edit View Search Terminal Help
> --image=./test/bottle1.jpg
python3: can't open file 'image_label.py--graph=/tmp/output_graph.pb': [Errno 2]
No such file or directory
gary@gary-Inspiron-5447:~/Documents/EC601/mini_projrct_2$ python3 image_label.py
--graph=/tmp/output_graph.pb --labels=/tmp/output_labels.txt --input_layer=Plac
eholder --output_layer=final_result --image=./test/bottle1.jpg
python3: can't open file 'image_label.py': [Errno 2] No such file or directory
gary@gary-Inspiron-5447:~/Documents/EC601/mini_projrct_2$ python3 label_image.py
--graph=/tmp/output_graph.pb --labels=/tmp/output_labels.txt --input_layer=Plac
eholder --output_layer=final_result --image=./test/bottle1.jpg
2018-10-21 00:17:00.575050: I tensorflow/core/platform/cpu_feature_guard.cc:141]
Your CPU supports instructions that this TensorFlow binary was not compiled to
use: AVX2 FMA
bottle 0.9977494
cup 0.002250606
gary@gary-Inspiron-5447:~/Documents/EC601/mini_projrct_2$ python3 label_image.py
--graph=/tmp/output_graph.pb --labels=/tmp/output_labels.txt --input_layer=Plac
eholder --output_layer=final_result --image=./test/cup1.jpg
2018-10-21 00:17:59.612017: I tensorflow/core/platform/cpu_feature_guard.cc:141]
Your CPU supports instructions that this TensorFlow binary was not compiled to
use: AVX2 FMA
cup 0.9998741
bottle 0.0001258454
gary@gary-Inspiron-5447:~/Documents/EC601/mini_projrct_2$

3. Detection model based on yolov3

Yolov3 is a pretrained object detection model based on darknet. I used the same dataset of the classification model to retrain yolov3 to perform object detection of bottles and cups. Unlike the classification model, yolov3 requires labeled images as dataset. To label the images, I downloaded the yolomark tool (https://github.com/AlexeyAB/Yolo_mark). This tool can generate a txt file which has a same name of the image, containing the class and coordinates of the rectangle label. Both the image and the txt file can be read by yolov3 so that the content in the rectangle will be considered as bottle(or cup). Below are 2 screenshots of labeling.



Yolov3 dose not automatically divide images into different sets. So I picked 15 images as validation set and 5 as test set. The configuration files can be found in the github.
Pack all these images into proper folders and download the pre-trained weight file of darknet. Then run the command ’./darknet detector train cfg/obj.data cfg/yolov3-obj.cfg darknet53.conv.74’ to train the model.
Some of the outcome of object detection is attached as following.
PS: More details of yolov3 installing and other configurations can be found on https://pjreddie.com/darknet/yolo/

4. A comparison between 2 systems

The object detection system is more complex than classification system, since the former one’s training and detecting takes much more time than the latter one. However, detection system has a wide usage range while classification system seems a little bit useless at this point. It is really hard to say which is better, but it is clear that object detection system has a higher hardware requirement and better usage range.