SASTA OLIPHANT SCIENCE AWARDS

SOUTH AUSTRALIAN SCIENCE TEACHERS ASSOCIATION

Highly Commended

Computer Programming, Apps & Robotics

Year 3-4

Patrick Deng
Monica Deng

Highgate School

Government of South Australia
Department for Education

SASTA South Australian Science Teachers Association

ROWE SCIENTIFIC PTY. LTD. www.rowe.com.au

RASPBERRY PI

OBJECT DETECTION

Introduction:

Pikachu is a popular Pokémon, and it will be much fun if the computer can learn, recognise, and speak out Pikachu and other Pokémon's name, which can be now achieved by TensorFlow and Raspberry Pi. The use cases and possibilities of the TensorFlow Python library are almost limitless. It could be trained to detect people in an image, cats, cars and many more. In this document, it is mainly about object detection steps of Pokémon for entertainment purpose. Furthermore, the Python script can be applied into the real-life usage and scientific study, which will be covered in this document as proof concept.

Hardware:

Raspberry Pi 3

Windows 10 high end laptop

Power Bank

Speaker

USB Web Camera

Main Steps of Model Training:

The process of training the custom object detection model follows most of the instructions from EdjeElectronics in the github. (https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10), but with a better solution for bulk composited image dataset auto generation by Python script, which makes multiple classes detection model more efficient. The major steps are summarised below.

Step 1:

The first step will be about training the custom TensorFlow Lite object detection model on a window 10 high end laptop, because the laptop has much better graphic card, memory and CPU power than the Raspberry Pi, it will save significant time to train the object detection model, and then apply the model to Raspberry Pi.

- Install Anaconda from https://www.anaconda.com/products/individual , and set up an Anaconda virtual environment for training

- Activate new created Tensorflow Python environment, and then install the following Python libraries at least, which will be used in the training and object detection Python script, i.e. tensorflow (version 1.15), protobuf, pillow, lxml, Cython, contextlib2, pandas and opencv-python.

Step 2:

Taking as many as possible of photos is one of the most important step required to train the model successfully. What's more, the training images should have detected objects in different backgrounds, with various size, rotation, and lighting conditions. There should be some images where the desired object is partially obscured, overlapped with something else, or only halfway in the picture. Therefore, my sister and I took around 3 hours to photograph 421 desired photos for Pikachu and Charmander.

Fig 1: My sister taking a photo of Pikachu

Our photo file size was too big to be processed efficiently during the model training, therefore we had to use the snipping tool to resize all the photos one by one, which was very boring and time consuming. In addition, we need to use one graphical image annotation tool called LabelImg (https://github.com/tzutalin/labelImg), to do the object labelling, which can not only record the photo dimension and also the detected object, its bounding box relative location in the photo, as the machine needs to know what exactly the object is. While the total time we spent in resizing and labelling was nearly the whole day for only 421 photos. However, TensorFlow requires thousands of various conditions of photos to train into a good model, the number of current photos to train and test the model seems a bit shortage.

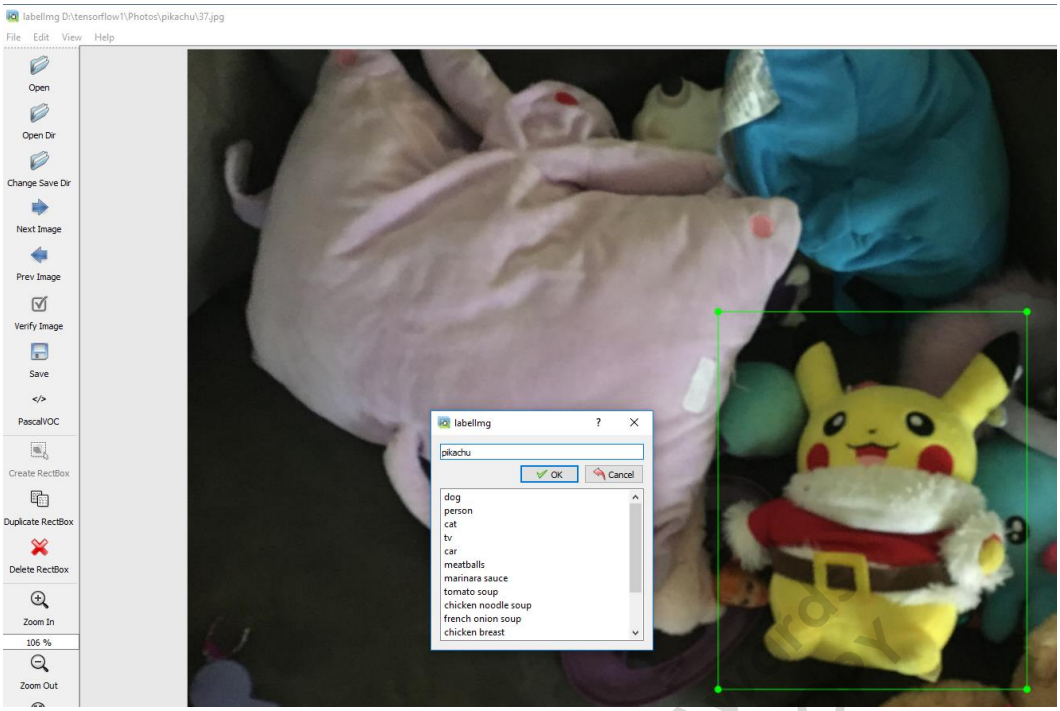

Fig 2: Using LabelImg

Step 3:

In order to generate more photos for the model training, we had to change our way smartly. We took a few PNG format photos of Pikachu and Charmander, as well as thousands of different scenes of downloaded photos as a background. Then, we can utilise a FOR loop in Python to compose an object on top of the background image and generate a bounding box (smallest and largest X and Y relative location values).

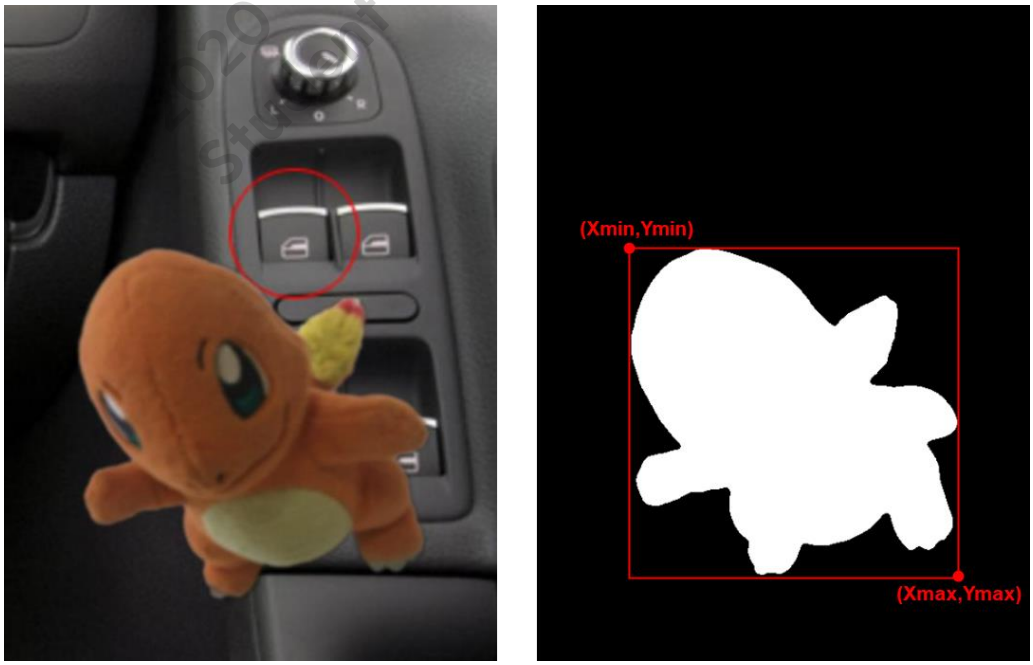

Fig 3: Composited image with bounding box

In addition, we can rescale, rotate the front object photo, and add random gaussian blur for a better training model. As a result, this Python script can generate thousands of synthetic image dataset with additional annotation information within hours, which makes the job a lot more productive and easier.

```
 # Rotate the foreground
angle_degrees = random.randint(0, 359)
foreground = foreground.rotate(angle_degrees, resample=Image.BICUBIC, expand=True)

# Scale the foreground
scale = random.random() * .5 + .5 # Pick something between .5 and 1
new_size = (int(foreground.size[0] * scale), int(foreground.size[1] * scale))
foreground = foreground.resize(new_size, resample=Image.BICUBIC)

# Add Gausssian Blur
foreground = foreground.filter(ImageFilter.GaussianBlur(radius = random.randint(0, 4)))
```

Fig 4: The coding of rescale, rotate and blurring

Step 4:

Once all the photos have been generated and labelled, then create two folders named “train” and “test”. 70% of the images go to the folder “train”, while the rest go to the folder “test”. After that, I used provided TensorFlow Python script to transfer the images and annotation information into “tfrecord” format, which is the array dataset format for TensorFlow in machine learning. The last thing before the training is to create a label map, which tells the machine what the object is by defining a mapping of class names to class ID numbers. The example below is the label map that we used in the Pokémon object detection.

```
item {
  id: 1
  name: 'pikachu'
}
item {
  id: 2
  name: 'charmander'
}
```

Fig 5: Label Map

Step 5:

We downloaded SSD-MobileNet-V2-Quantized-COCO model from Google detection model zoo, as quantized models use 8-bit integer values instead of 32-bit floating values within the neural network, allowing them to run much more efficiently. With couple of modifications in the training pipeline config file, i.e. Change num_classes to the number of different objects, which is 2 in this case, num_examples variable to be the number of images in the “test” folder, also the file path of the label map and “tfrecord” files. Finally, the training job is ready to run by typing this command, ***python train.py --logtostderr –train_dir=training/ --pipeline_config_path=training/ssd_mobilenet_v2_quantized_300x300_coco.config***

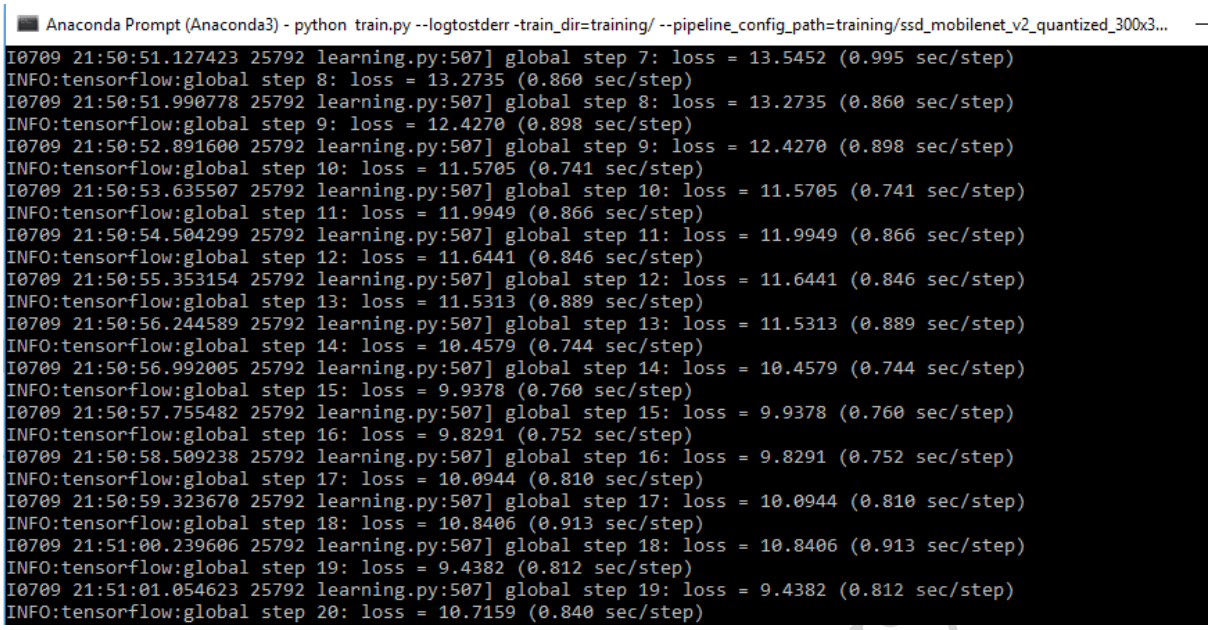

Fig 6: Training job steps

During the training, there is chance that the laptop may go to the blue screen, no response or error return, due to the running out of resource, thus we have to restart the laptop to continue, however we won't lose a lot of steps, since the latest every 500 steps will be saved. After a whole day of calculation, the total steps that were running was 100785. The loss rate dropped from more than ten at the beginning, to constantly under two, which can be observed from TensorFlow dashboard shown below.

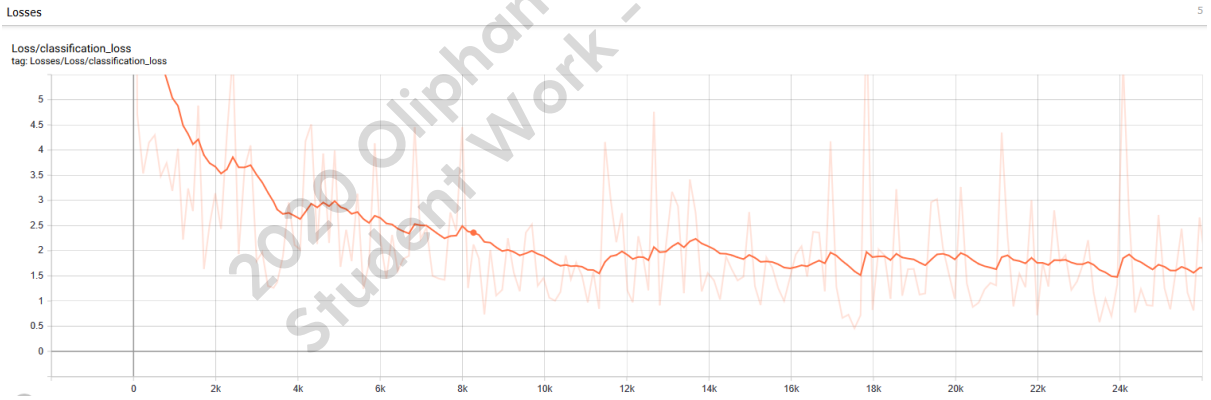

Fig 7: TensorFlow loss rate dashboard

Step 6:

Besides Python and OpenCV packages, it is also required to install TensorFlow Lite runtime environment in Raspberry Pi, which is particularly designed for Raspberry Pi or other mobile devices. Once the training completes, exporting the final frozen TensorFlow Lite graph file to Raspberry Pi, then in the command terminal of Raspberry Pi, issuing this command, ***Python3 Detection.py***, which will enable the object detection by using the USB web camera.

Pokémon Detection Result:

It was very exiting that the Raspberry Pi could recognise the Pikachu and Charmander correctly. In order to add more challenge, we decided to get Raspberry Pi to speak out the Pokémon's name, when it detects a Pokémon, which can be solved by adding one audio speak Python function shown below.

```
def audio_speak(audio_string):
    tts = gTTS(text=audio_string,lang='en')
    r = random.randint(1,10000000)
    audio_file = 'audio-'+str(r) + '.mp3'
    tts.save(audio_file)
    pygame.mixer.music.load(audio_file)
    pygame.mixer.music.play()
    os.remove(audio_file)
```

Fig 8: Audio speak Python function

Therefore, it will be a helpful education programme for children to get familiar with Pokémon and other things, such as animals or food products.

Fig 9: Detection result of Pikachu and Charmander

Real-life Example:

When the kids are at school and the parents are at work, and there might be an important parcel been delivered and left outdoor. For a long time, Raspberry Pi has been customised for home security system, but without object detection ability. We can easily modify this script that it can

detect the parcel, as well as sending a SMS message by using Twilio Python library, to inform the parents come home and pick the package up.

Fig 10: Detection result of a parcel

This experiment is not working quite well with our existing hardware and training model, because the parcel can be different shapes and packages, our model is only being trained to detect the Australian mailing box. In addition, the brightness condition of outside can change during the day due to the sun's direction and nearby object's shadow. It can be improved by taking more photos of different shapes of parcels in various of lighting condition for model training.

Scientific Study Example:

In the lab, there is a case that the scientists want to compare which food that the mouse likes the most, so they can create a solution to control the mouse reproduction. However, it will be time consuming to get people to do the counting 24/7, and can easily miss count, while using Raspberry Pi object detection, which is inexpensive and portable tool to achieve this requirement quite simply. For example, we modified the script to monitor the area of food A and food B, as well as trained the model to recognise the mouse. If the mouse, its centre point is inside the food A box area for more than certain time, the program will do the counting automatically. At the end of the observation, scientists will get the statistic result immediately.

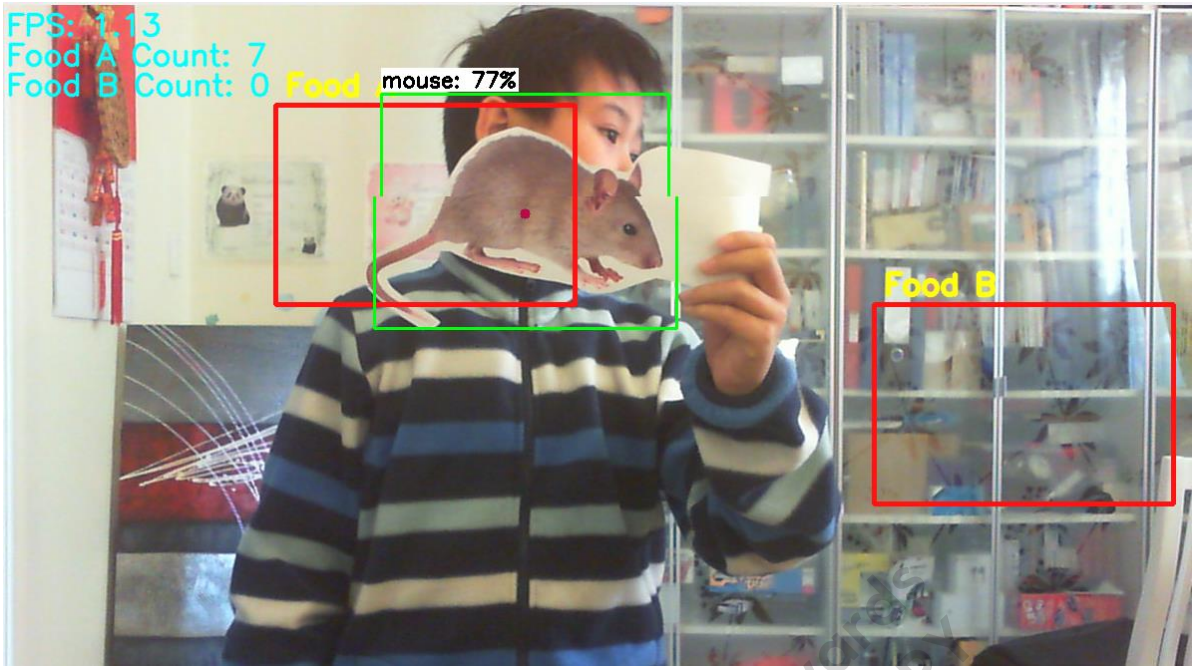

Fig 11: Scientific study of food choice by mouse

Conclusion:

In summary, this document covers the basic steps to train the custom object detection model, with an improvement to generate thousands of composited images automatically instead of manually taking photos and using LabelImg application. When the training model is being transferred into Raspberry Pi, people can utilise this low cost and credit card size machine to do object detection in different areas.

However, the video frame rate of this Raspberry Pi is between 0.9 to 1.4 FPS, which is not suitable for real time detection. Because it is Raspberry Pi 3, and if it is upgraded to Raspberry Pi 4, with USB image accelerator, there will be significant faster and better detection result.

The detection result video clip can be accessed without password from my Google drive, https://drive.google.com/drive/folders/1G51dcCYLdCAA5pvSsQ1YKPzKtJNNOt8b?usp=sharing

Acknowledgement:

I happened to see machine object detection tutorial videos from YouTube and thought it would be very cool if I can use my Raspberry Pi to recognise all my Pokémon toys, as my sister and I are a big fan of Pokémon, which was highly supported by my father. This project would not have been possible without his technical assistance in the Python coding when I was understanding the example object detection script from TensorFlow website and EdjeElectronics in the github, as well as doing modification for improvement. I am very grateful that my father sacrificed his time helping me during the report writing.

Bibliography:

Evan, How To Train an Object Detection Classifier for Multiple Objects Using TensorFlow (GPU) on Windows 10. https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10

Training Custom Object Detector. Copyright 2018, Lyudmil Vladimirov Revisio. https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/training.html