Project Report 5
Deep Learning on Raspberry Pi

Orion Peter Fernandes - 65047854

1. Project Description (2 pts)

- The project deals with learning to make a pre-trained deep learning network on the ImageNet/PASCAL VOC dataset and apply it to input images using OpenCV on Python3
- Using OpenCV 3.3 in this project, which brings deep learning (dnn) module that supports a variety of deep learning frameworks, including Caffe, Tensorflow, Torch/Pytorch, and Darknet.
- The previous OpenCV library was too old for DL applications. Thus, we're using the newer OpenCV Python3 packages from Pypl repositories.
- This project is divided into 2 parts –
  - i) Deep Learning with OpenCV and Image Classification using OpenCV and various pre-trained Caffe models
  - ii) Real-Time Object Detection using OpenCV and the pre-trained Caffe model
- Images shall be classified with OpenCV and can also be used in real-time object detection using OpenCV and pre-trained Caffe model

2. Experimental Setup (4 pts)

- Upgrade pip before installing OpenCV

pip install --upgrade pip

- Install OpenCV

pip install opencv-python==4.5.3.56

- If this error pops up

  File "/home/pi/.local/lib/python3.7/site-packages/cv2/__init__.py", line 5, in <module> from .cv2 import * ImportError: numpy.core.multiarray failed to import

  Use

  python3 -m pip install --upgrade numpy

- Check if it works with

  $ python3

  >>>import cv2

  >>>cv2.__version__

  CTRL+D to leave Python3

- Download required files from Google Drive Link through Canvas Page – **p5_classification.py**, **p5_object_detection.py**, **synset_words.txt**, **image_classification**, **images**, **object_detection**, **and videos**.

- Extract the downloaded files before transferring to Raspberry Pi under homework folder.

- Modify **p5_classififcation.py** to calculate timing operation of program with

  import time

  start=time.time()

  time.sleep(1)

  end=time.time()

- For Image Classification, run the GoogleNet Image classifier

$ python3 deep_learning_with_opencv.py -i eagle.png -p bvlc_googlenet.prototxt.txt -m bvlc_googlenet.caffemodel -l synset_words.txt

- The following flags used refer to

-i or --image: *The path to the input image.*

-p or --pretext: *The path to the Caffe "deploy" prototxt file.*

-m or --model: *The pre-trained Caffe model (i.e,. the network weight files).*

-l or --labels: *The path to ImageNet labels.*

- Record observations for each image classified.
- Similarly, modify **p5_object_detection.py** to calculate timing operation of program with

```
import time
start=time.time()
time.sleep(1)
end=time.time()
```

- For Object Detection, run the Python code

$ python3 real_time_object_detection.py --video_in ground_crew_480p.mp4 --prototxt MobileNetSSD_deploy.prototxt.txt --model MobileNetSSD_deploy.caffemodel --confidence 0.2

- The following flags used refer to

  -i or --video_in: *Path to input video. If not specified, the program uses the camera, /dev/video0.*

  -o or --video_out: *Path/Name to output video, e.g. output.mp4. If not specified, there will be std output to the console. Press q to terminate.*

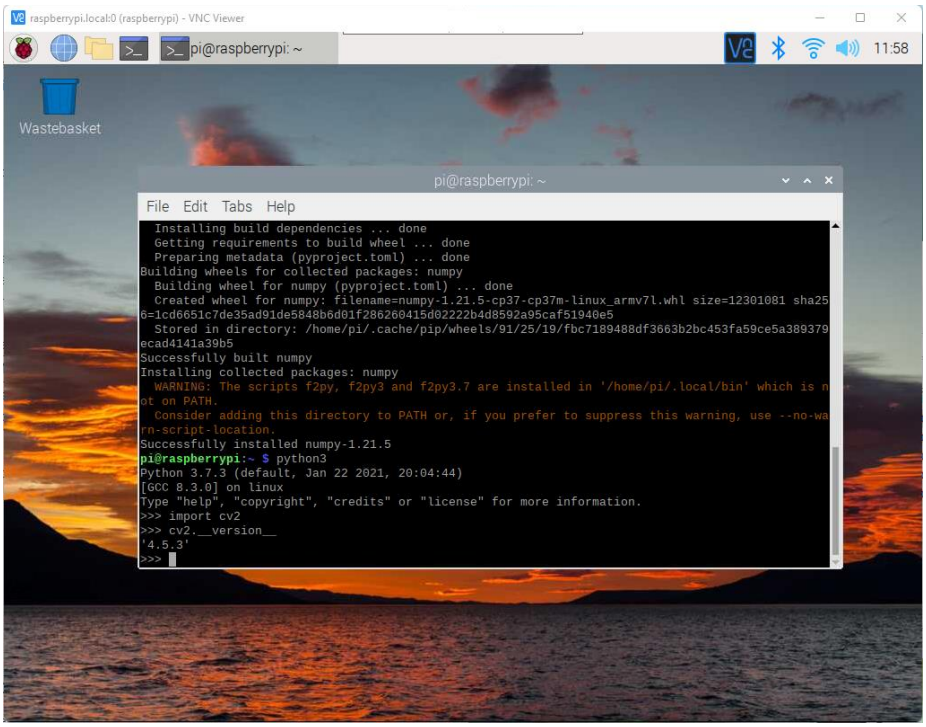
  -p or --prototxt: *The path to the Caffe "deploy" prototxt file.*

  -m or --model: *The pre-trained Caffe model (i.e,. the network weight files).*

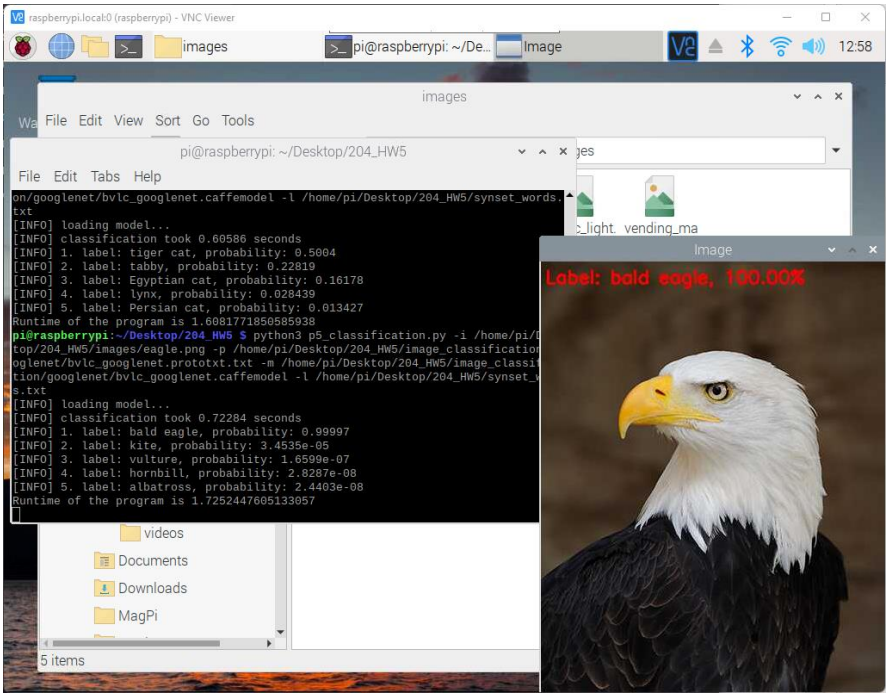
  -c or --confidence: *minimum probability to filter weak detections*

- Record observations for each object detection.

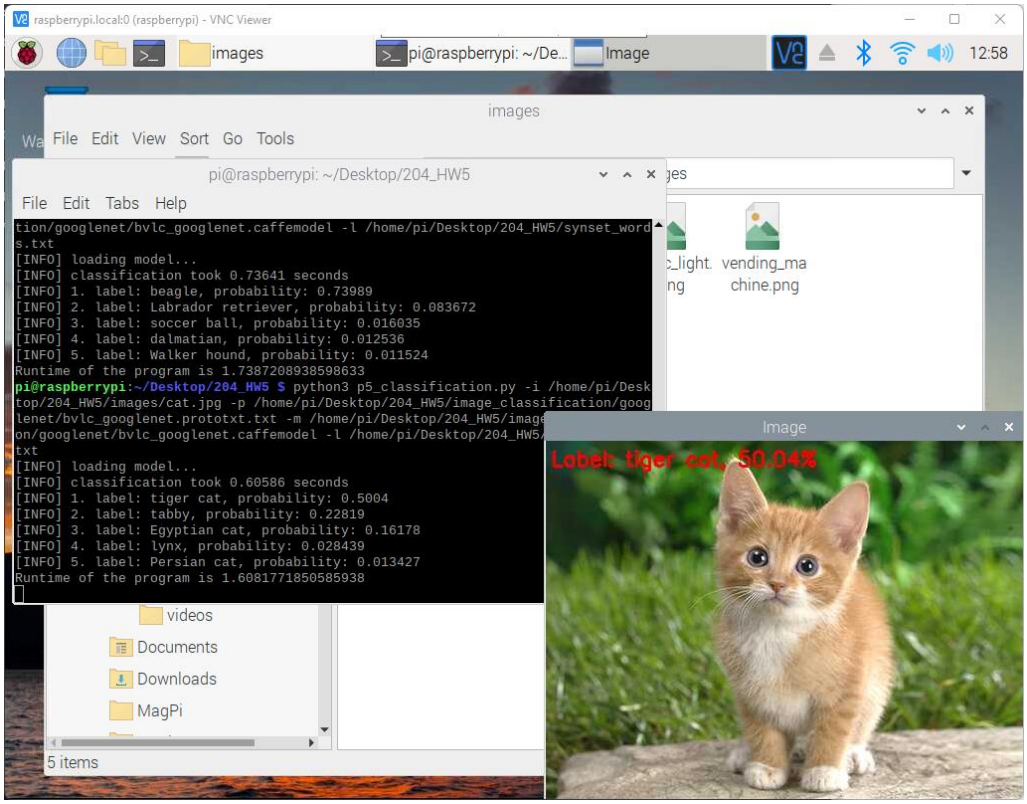
3. Results (6 pts)



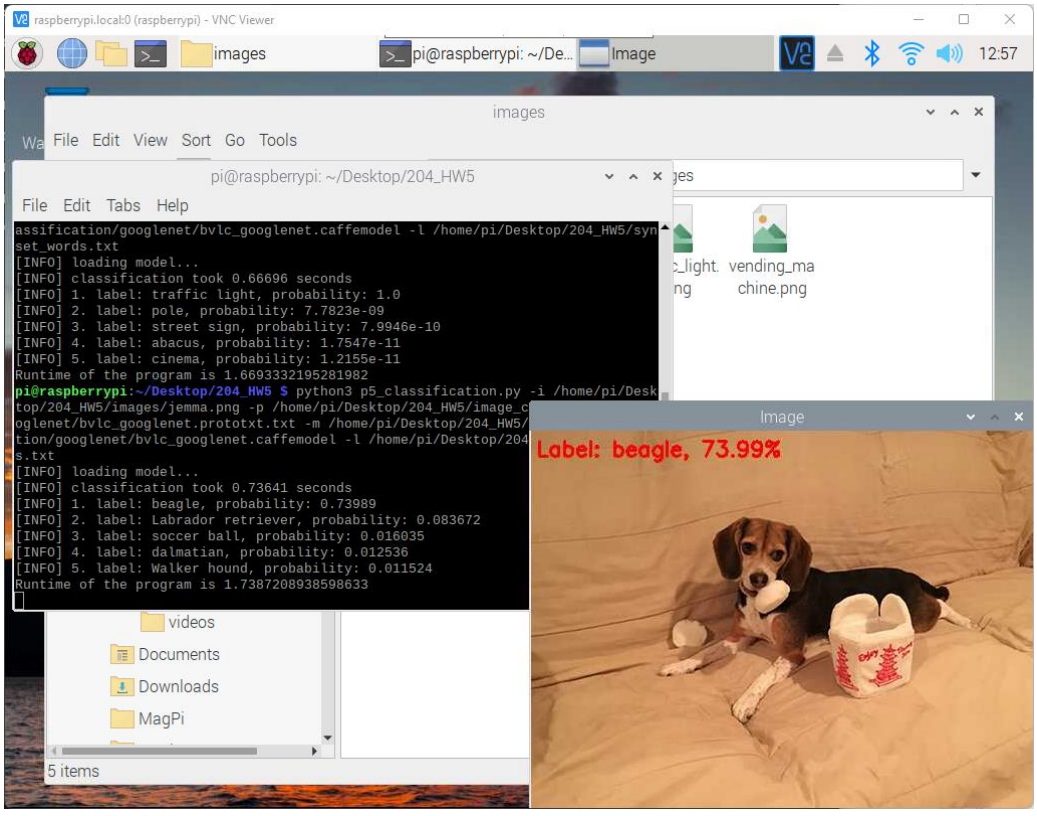
OpenCV Installed



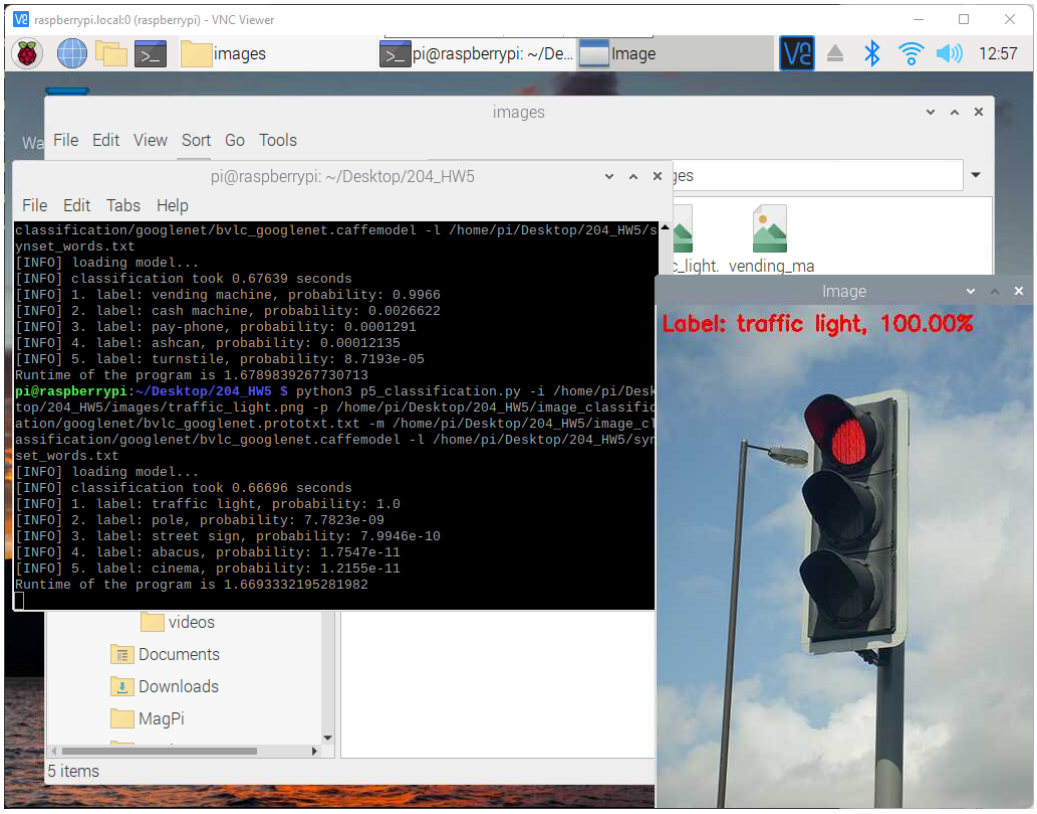
Eagle model classified



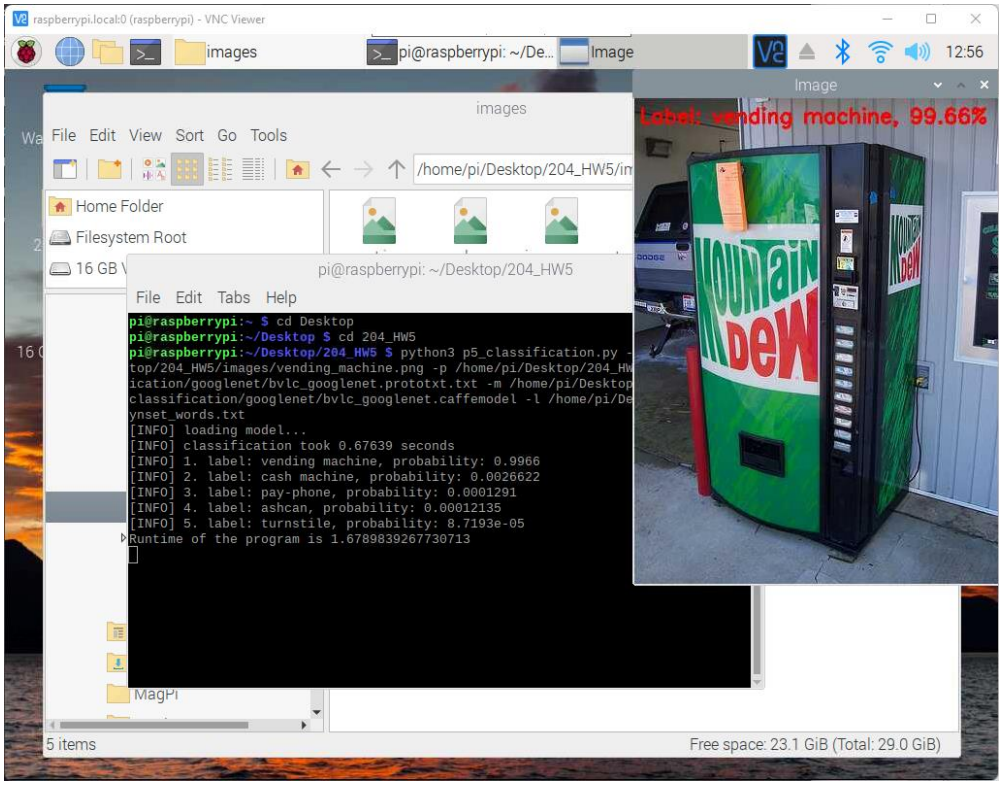
Cat model classified



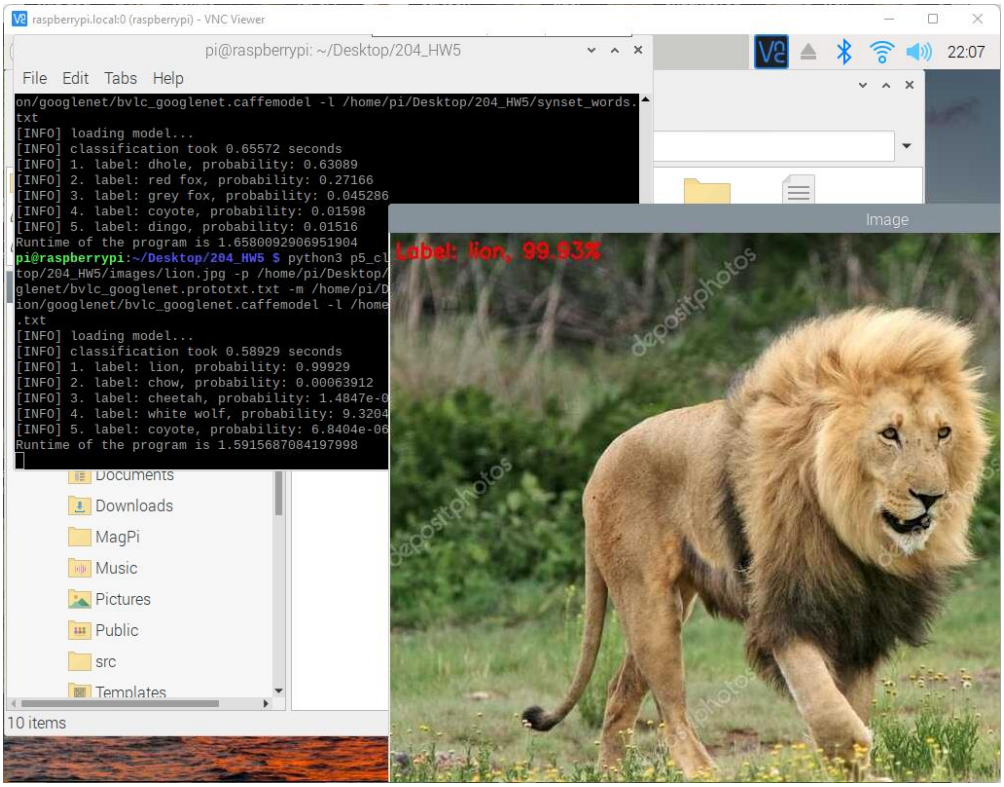
Beagle model classified



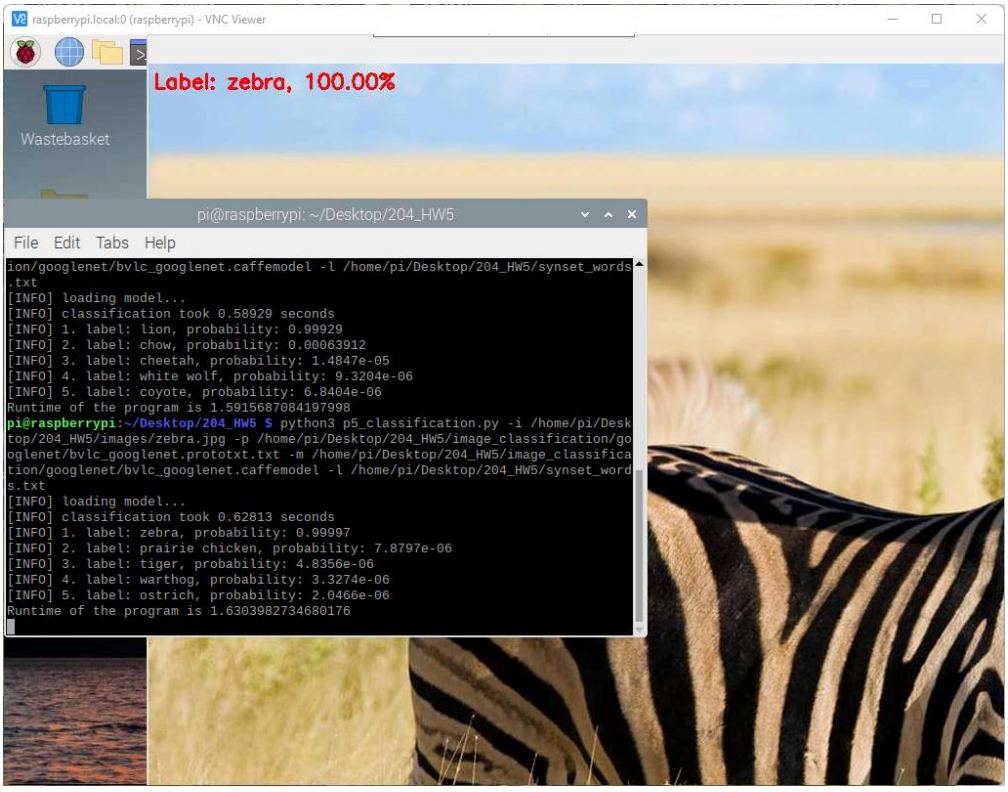
Traffic Light classified



Vending Machine classified



Lion Classified



Zebra Classified

Image Classification Observations

Image	Runtime
Eagle	**1.75244**
Cat	**1.608177**
Beagle	**1.7387208**
Traffic Light	**1.6693332**
Vending Machine	**1.6789839**
Lion	**1.5915687**
Zebra	**1.630398**

- Image classification depends on the background and other parameters as well.
- Gflops was found to be 3.190
- Image is loaded from disk using

image = cv2.imread(args["image"])

- .caffemodel consists of pre-trained models from Caffe which is a deep learning framework.
- protxt.txt is a file prototype machine learning model created for use with Caffe.
- synset_words.txt consists of trained labels for images that have been classified during training.
- Image is fed into DNN with

image = cv2.imread(args["image"])

Class labels loaded from disk using

```
rows = open(args["labels"]).read().strip().split("\n")
classes = [r[r.find(" ") + 1:].split(",")[0] for r in rows]
```

Image resized to blob to meet CNN requirements of shape (1, 3, 224, 224)

```
blob = cv2.dnn.blobFromImage(image, 1, (224, 224), (104, 117, 123))
```

Serialized model loaded from disk using

```
print("[INFO] loading model...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
```

Blob set as input to network and perform a forward-pass to obtain output classification

```
net.setInput(blob)
start = time.time()
preds = net.forward()
end = time.time()
print("[INFO] classification took {:.5} seconds".format(end - start))
```

- To predict the classification, indexes of probabilities provided are sorted in descending order (higher probability first) and the first 5 are listed.
- Output image is displayed with predicted label with highest value.

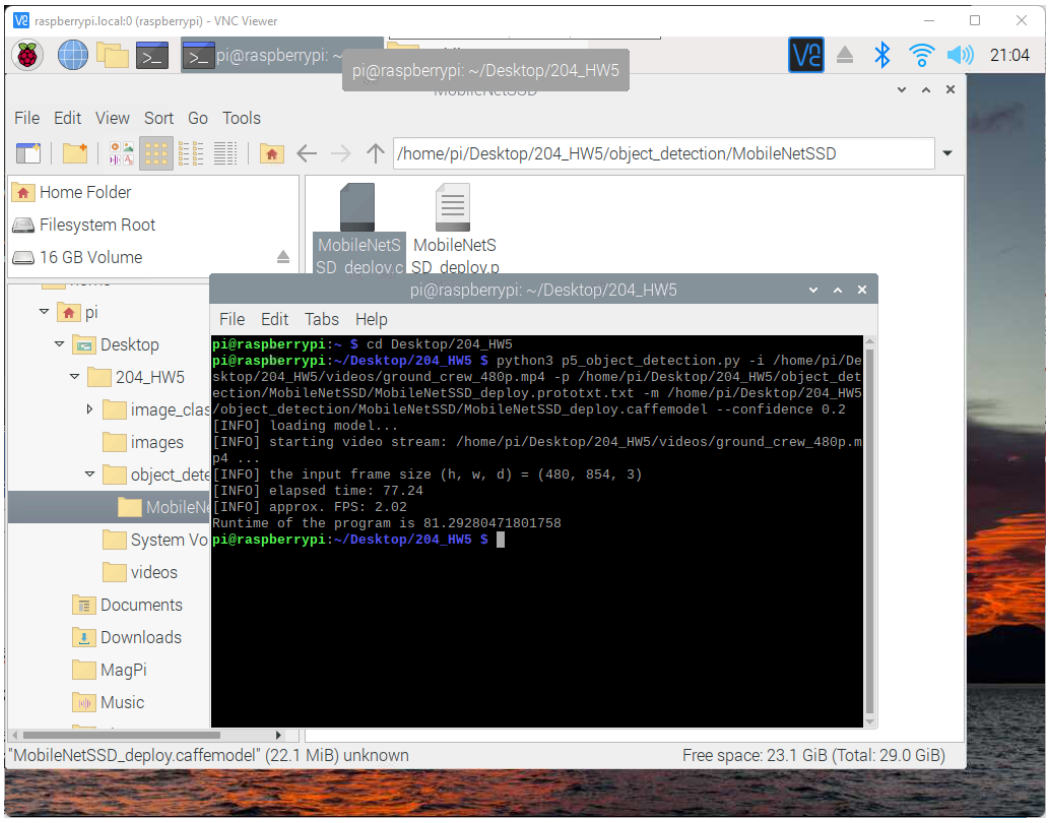
- To calculate the computational complexity, code used –

print('gflops:', net.getFLOPS((1,3,224,224))*1e-9)

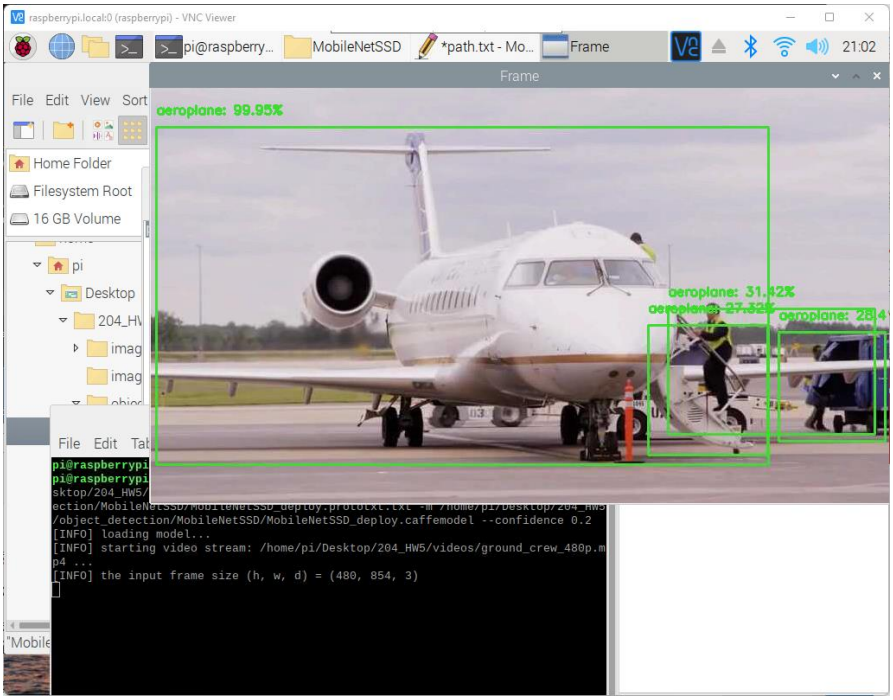
```
import cv2
weights = self.find_dnn_file("dnn/bvlc_googlenet.caffemodel", required=False)
config = self.find_dnn_file("dnn/bvlc_googlenet.prototxt", required=False)
if weights is None or config is None:
    raise Exception("Missing DNN test files (dnn/caffemodel name.{prototxt/caffemodel}). Verify OPENCV_DNN_TEST_DATA_PATH configuration parameter.")
# Use OpenCV to Instantiate Caffe2 Model
model = cv2.dnn_DetectionModel(weights, config)
model.setInputParams(size=(224, 224), mean=(0, 0, 0), scale=1.0)

# set output layer for forward pass
outputLayer = "detection_out"

# Evaluate Memory Consumption
weightsMemory, blobsMemory = model.getMemoryConsumption((1,3,224,224))
# calculate FLOPS
flops = model.getFLOPS((1,3,224,224))
model.forward(outputLayer)
print("Memory consumption:")
print("    Weights(parameters): ", (weightsMemory + (1<<20) - 1) / (1<<20), " Mb")
print("    Blobs: ", (blobsMemory + (1<<20) - 1) / (1<<20), " Mb")
print("Calculation complexity: ", flops * 1e-9, " GFlops")
```



P5_object_detection.py Execution

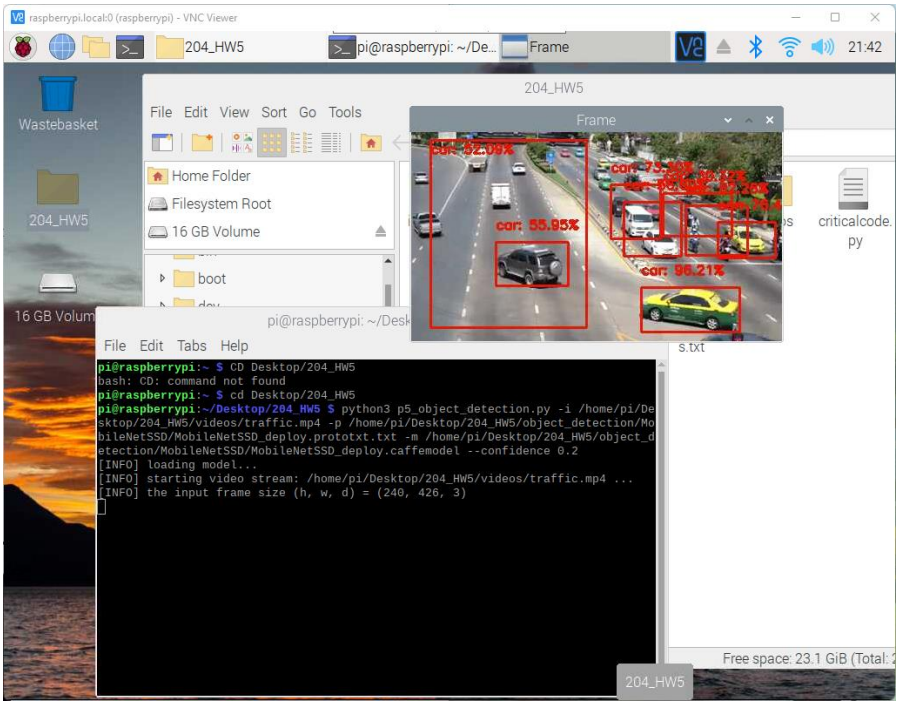


ground_crew.h264 Object Detection

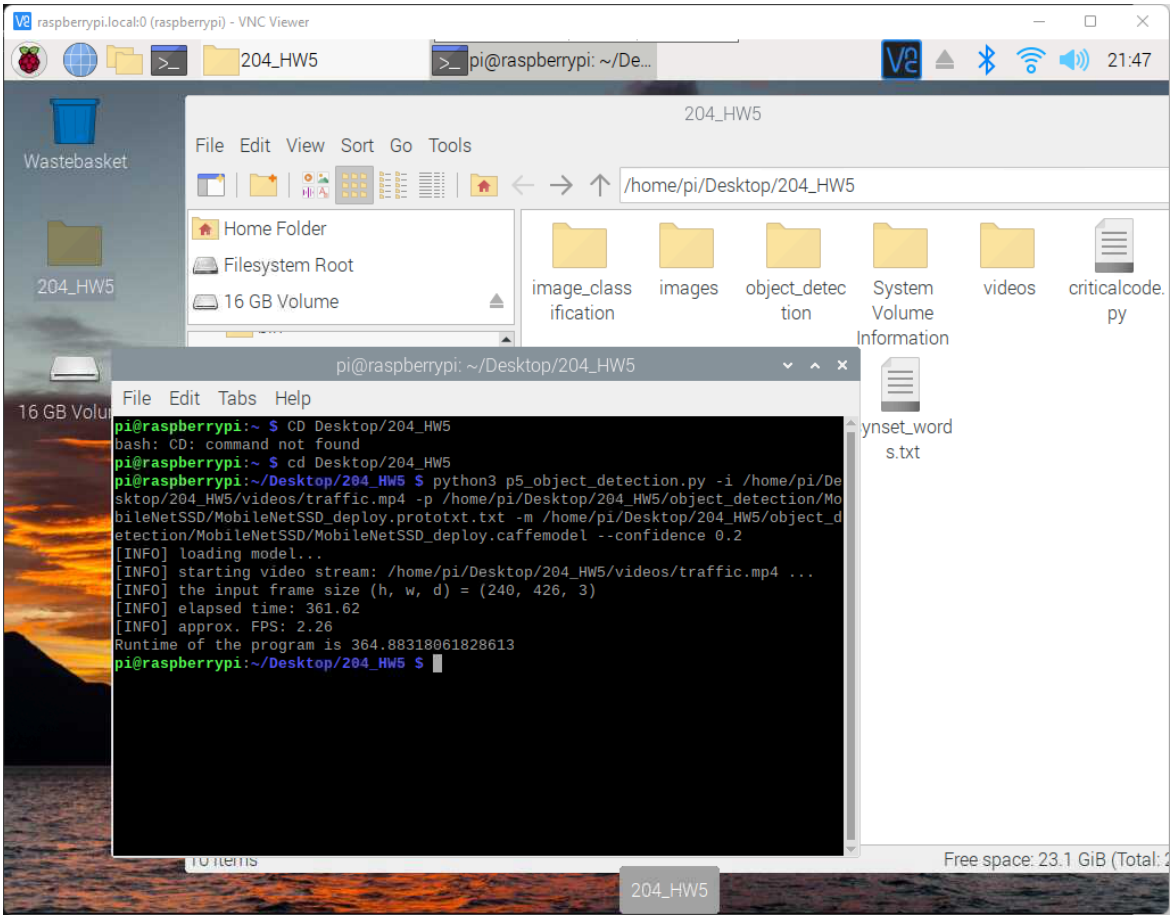
Observations

ground_crew.h264	Results
Input Frame Size – (h, w, d)	**(480, 854, 3)**
Elapsed Time	**77.24**
FPS	**2.02**
Runtime	**81.29280**

- The runtime is clearly dependent on the length of the video file that's input.
- FPS is considered low, considering multiple objects are being detected.
- Input frame size is dependent on resolution of input video.
- Objects are identified using the detection algorithm upon which index of the class label is extracted to compute (x, y) coordinates of bounding box of the object.
- MobileNET is a neural network used for classification and recognition whereas SSD (Single Shot Detector) is a framework that's used to realize the multibox detector. MobileNET+SSD can perform object detection.
- Object detection may be performed using MobileNET+SSD or GoogleNet .



Traffic Example Object Detection



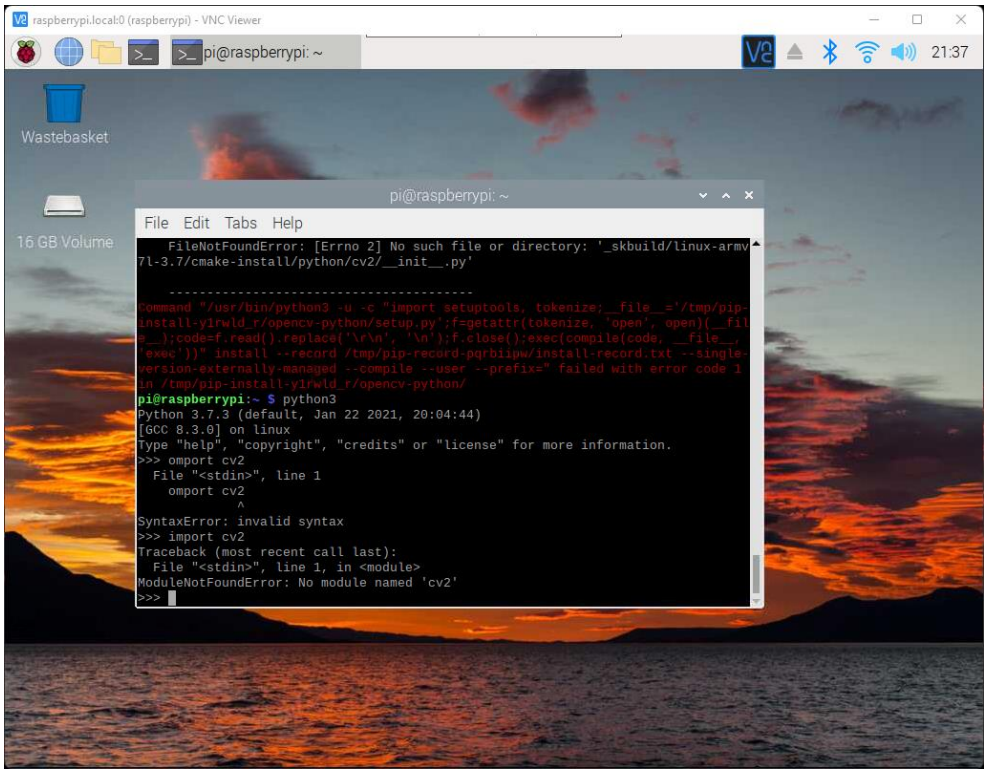
Traffic Example Execution

traffic.mp4	Results
Input Frame Size – (h, w, d)	**(240, 426, 3)**
Elapsed Time	**361.62**
FPS	**2.26**
Runtime	**364.883**

- The traffic video was around 27 seconds long and was a sample for object detection
- There's barely any change in FPS between both examples.
- Gflops was found to be 9.774.

4. Problems and Discussion (6 pts)

- OpenCV wouldn't install with several methods.




```
>>> cv2.__version__
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'cv2' is not defined
>>>
```

- OpenCV finally installed using

pip install --upgrade pip

pip install opencv-python==4.5.3.56

python3 -m pip install --upgrade numpy

- Extraction of downloaded files takes longer on the Raspberry Pi, thus extracting them on PC before transferring it to the Raspberry Pi is much more efficient
- In the traffic video example taken for object detection, buses were occasionally labeled as cars. This was perhaps of poor labelling of the training set. Similarly, passengers boarding the airplane were labeled as airplane.

- The results matched my expectations such that images were correctly classified and objects detected in the video sample.
- Methodology can be involved by installing a later version of OpenCV – 4.3.56 and providing more samples to test the algorithm on. Moreover, files downloaded from the Caffe Model drive link should be zipped beforehand for a shorter download period and extracted later.
- Had to resort to using github regarding OpenCV installation since I even installed it on a virtual environment using a partition on my Raspberry Pi and it didn't install through.

5. Conclusion (2 pts)

- Image classification refers to classify what is contained in an image. It largely depends on the training set since it's a predictive model and other elements such as background, brightness, etc. may affect its prediction.
- Object detection specifies location of multiple objects in image, including –
    - Image Classification: classify what's contained in an image.
    - Image Localization: specify location of every single object in an image.

6. References

- https://www.youtube.com/watch?v=jjlBnrzSGjc&list=PLcQZGj9lFR7y5WikozDSrdk6UCtAnM9mB&ab_channel=PanasonicSecurity – **Traffic Video Sample**
- https://www.studytonight.com/post/calculate-time-taken-by-a-program-to-execute-in-python#:~:text=The%20timeit()%20method%20of,the%20default%20value%20is%20pass – **Calculate Script Execution Time**
- https://docs.opencv.org/4.2.0/db/d30/classcv_1_1dnn_1_1Net.html - **OpenCV Documentation**
- https://medium.com/nerd-for-tech/how-to-test-the-performance-of-the-caffe2-model-caffemodel-fbce2e4d9c68 - **Testing Performance of Caffe2 Model**
- https://intellipaat.com/community/16415/mobilenet-vs-ssd#:~:text=As%20far%20as%20I%20know,resnet%2C%20inception%20and%20so%20on – **MobileNET VS SSD**

  https://www.vlchelp.com/cut-trim-videos-with-vlc-media-player/ - **Video Editing to trim with VLC**