物聯網與微處理機系統設計
Internet of Things and Microprocessor System Design

Lecture 08 – CV on RPi

Lecturer: 陳彥安 Chen, Yan-Ann

YZU CSE

Modified from: NCTU IoT Course
https://github.com/coldwufish/RaspPI

Outline

- Facial recognition
- Object detection

Outline

- Facial recognition
- Object detection

Preparation (1/2)

- Install OpenCV
- Load Linux V4L2 camera driver

```
$ sudo modprobe bcm2835-v4l2
```

Preparation (2/2)

- Download sample codes and unzip it.

$ wget https://github.com/yachentw/yzucseiot/raw/main/lec08/face_detection.tar.gz

$ tar -zxvf face_detection.tar.gz

```
pi@rpi4-A00:~/iot $ tar -zxvf face_detection.tar.gz
face_detection/
face_detection/image_face_detect.py
face_detection/haarcascade_eye_tree_eyeglasses.xml
face_detection/camera_face_detect.py
face_detection/haarcascade_frontalface_default.xml
```

$ cd face_detection

```
pi@rpi4-A00:~/iot $ cd face_detection/
pi@rpi4-A00:~/iot/face_detection $
```

- Sample codes
  - Images from files: image_face_detect.py
  - Images from camera: camera_face_detect.py

Facial Detection

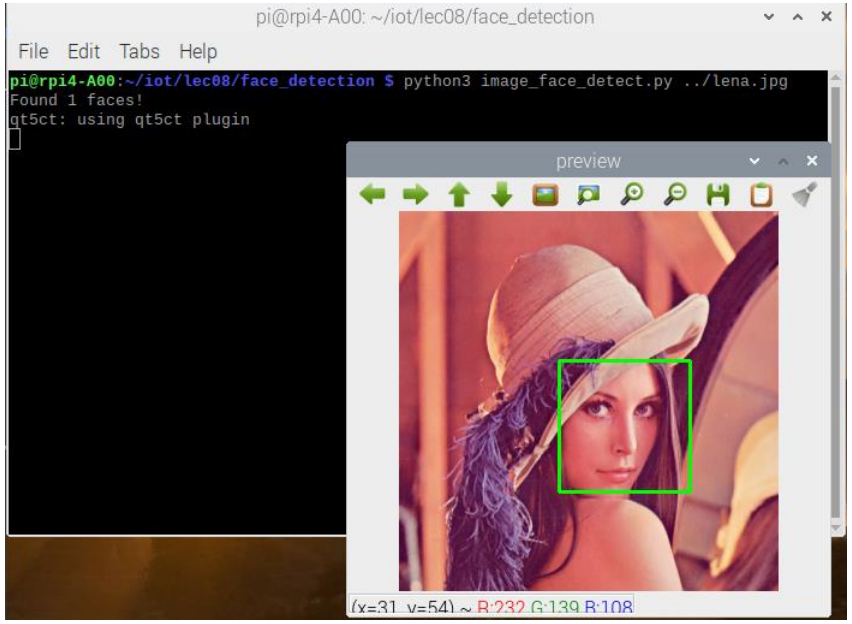
- Detect face of an image in the file system.

$ wget https://upload.wikimedia.org/wikipedia/zh/3/34/Lenna.jpg

$ mv Lenna.jpg ~/lena.jpg

$ python3 image_face_detect.py ../lena.jpg

- Specify the path to “lena.jpg” according to your environment.



image_face_detect.py

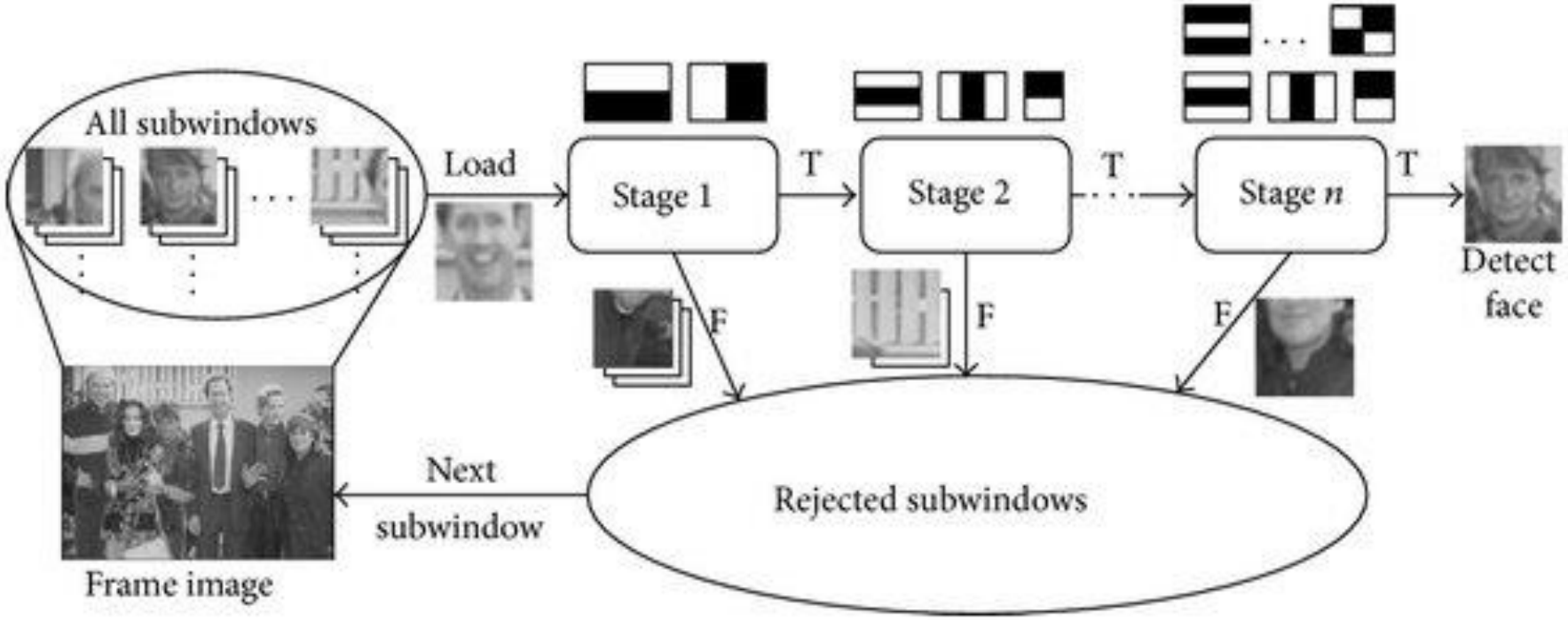
```
import sys
import cv2

# Get user supplied values
cascPath = "haarcascade_frontalface_default.xml"
imagePath = sys.argv[1]
if len(sys.argv) > 2:
    cascPath = sys.argv[2]

# Create the haar cascade
faceCascade = cv2.CascadeClassifier(cascPath)

if cv2.__version__.startswith('2'):
    PROP_FRAME_WIDTH = cv2.cv.CV_CAP_PROP_FRAME_WIDTH
    PROP_FRAME_HEIGHT = cv2.cv.CV_CAP_PROP_FRAME_HEIGHT
    HAAR_FLAGS = cv2.cv.CV_HAAR_SCALE_IMAGE
elif cv2.__version__.startswith('3') or cv2.__version__.startswith('4'):
    PROP_FRAME_WIDTH = cv2.CAP_PROP_FRAME_WIDTH
    PROP_FRAME_HEIGHT = cv2.CAP_PROP_FRAME_HEIGHT
    HAAR_FLAGS = cv2.CV_FEATURE_PARAMS_HAAR
# Read the image
image = cv2.imread(imagePath)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

```
# Detect faces in the image
faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor=1.1,
    minNeighbors=5,
    minSize=(30, 30),
    flags=HAAR_FLAGS
)

print ("Found {0} faces!".format(len(faces)))

# Draw a rectangle around the faces
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)

cv2.imshow("preview", image)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

Cascade Classification

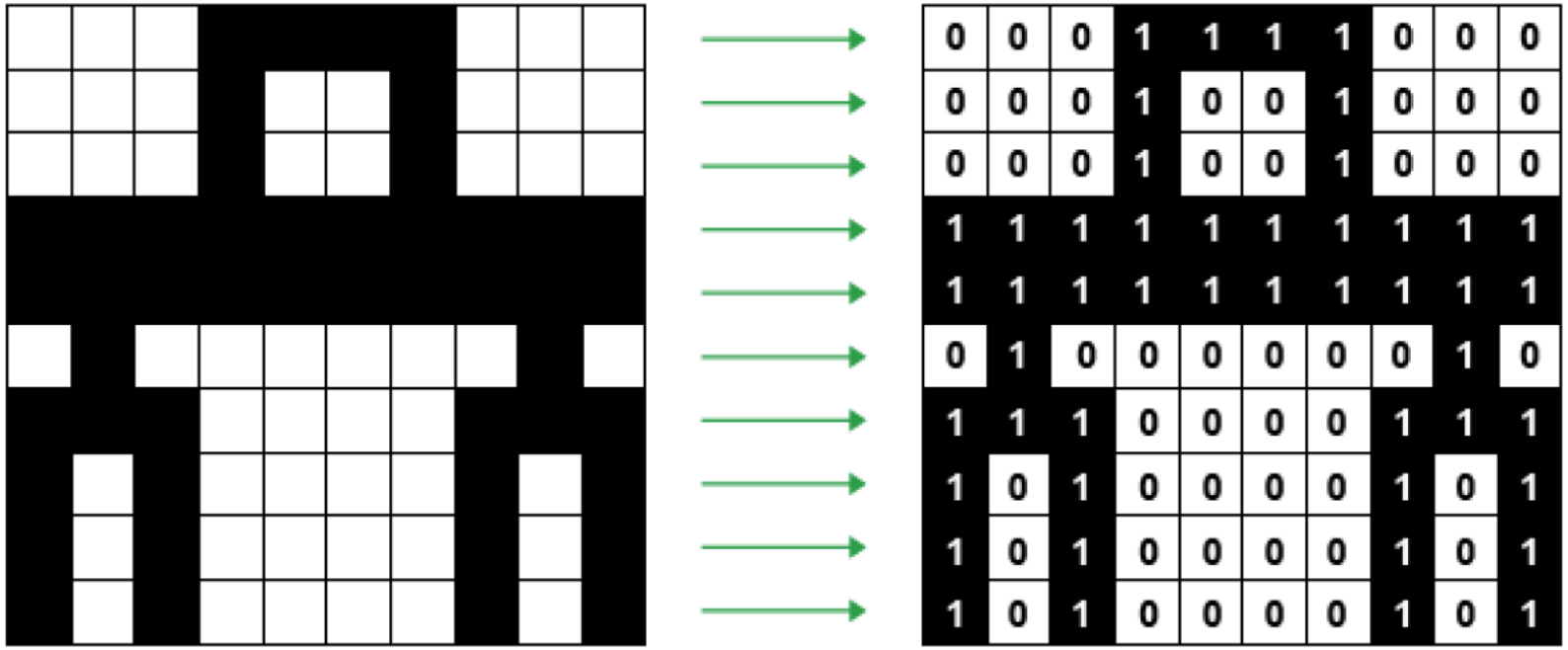
- https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html
- Haar feature-based cascade classifier for object detection
- The object detector described below has been initially proposed by Paul Viola [Viola01] and improved by Rainer Lienhart [Lienhart02].
- A classifier is trained with a few hundred sample views of a particular object (i.e., a face or a car), called positive examples.
  - Output 1: the region is likely to show the object (i.e., face/car)
  - Output 0: otherwise

Concept


All subwindows
Load
Stage 1
T
Stage 2
T
Stage n
T
Detect face
F
F
F
Frame image
Next subwindow
Rejected subwindows

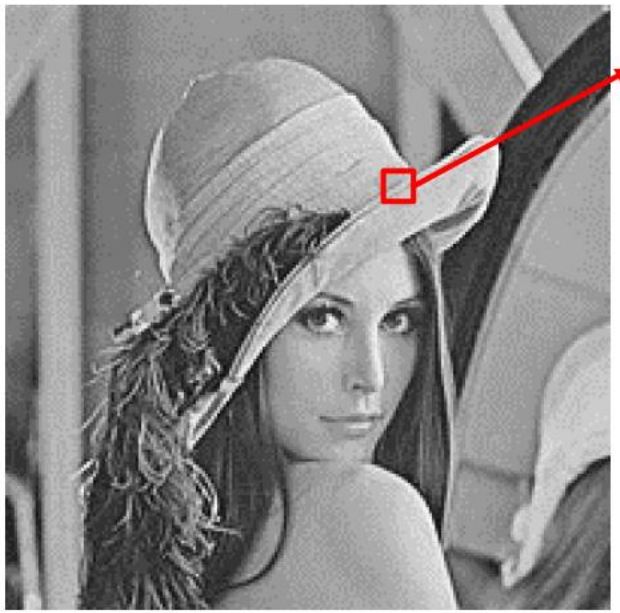
Bitmap Images

- Example: black-and-white image



Bitmap Images

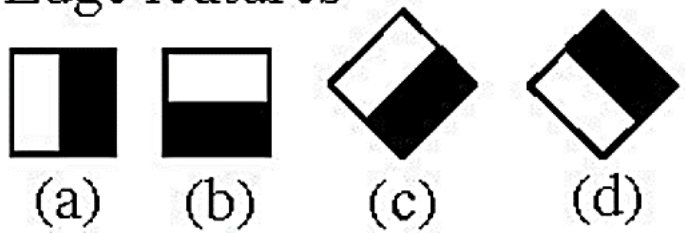
- Example: grayscale picture
  - 8 bits per pixel
  - This pixel depth allows 256 different intensities



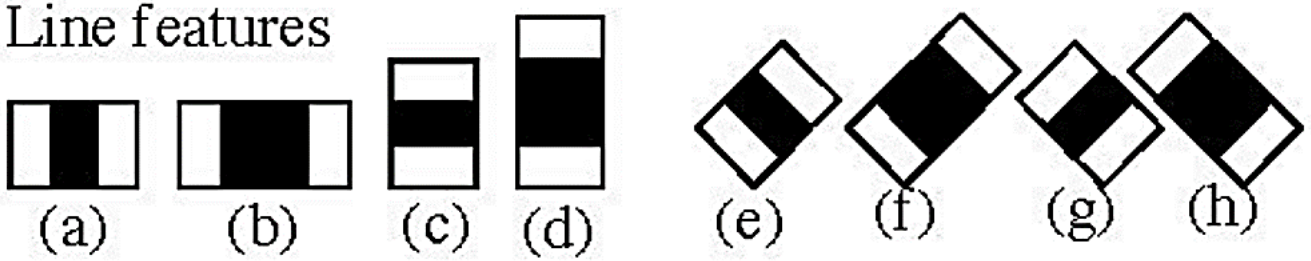
154	108	198	216	52
61	168	148	52	45
72	80	55	134	39
89	129	232	204	155
156	99	118	125	83

Haar-Like Features

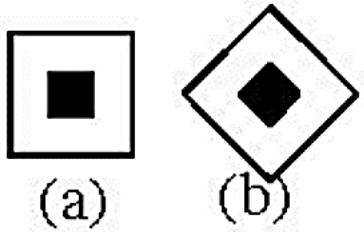
1. Edge features



2. Line features



3. Center-surround features

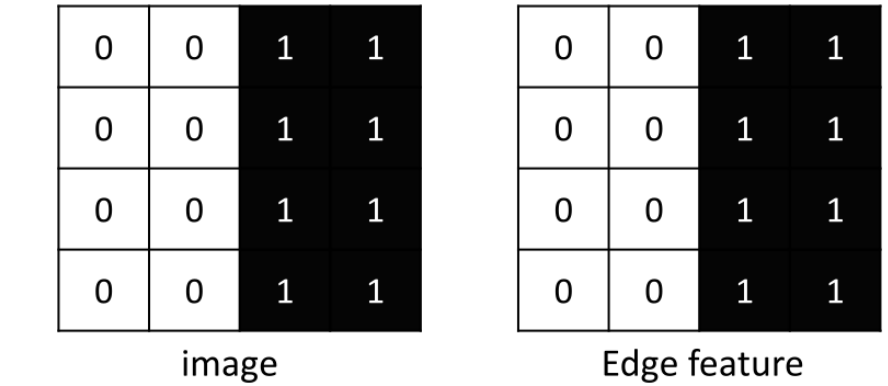


Features (1/2)

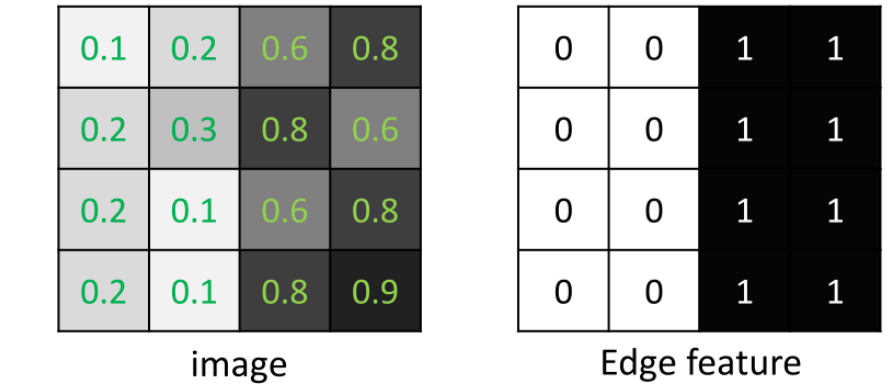
- Pick a scale (ex: 24x24 pixels) for the feature
- Slide it across the image
- Compute the average pixel values under the white area and the black area.
- If the difference between the areas is above some threshold, the feature matches.

Features (2/2)

1. Calculate the average of white/black pixel
2. Calculate the difference



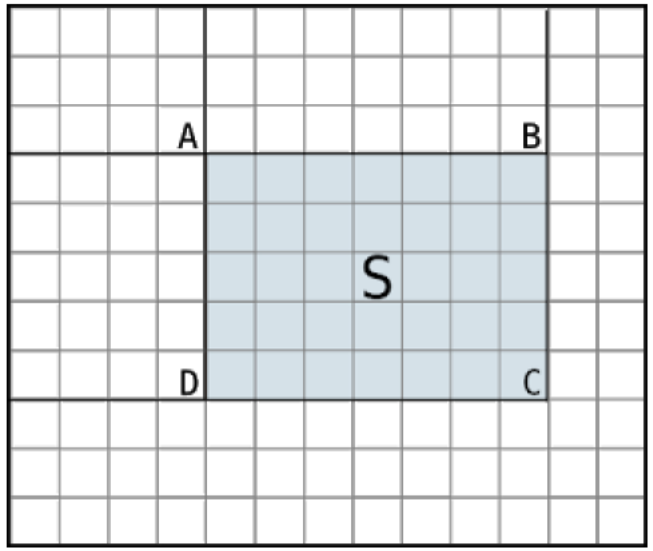
$$\Delta = black - whilte = 1$$



$$\Delta = \frac{0.6 + 0.8 + \cdots}{8} - \frac{0.1 + 0.2 + \cdots}{8}$$
$$= 0.7375 - 0.175 = 0.56$$

Integral Image (1/3)

- A quick and effective way of calculating the sum of values (pixel values) of a rectangular subset of a grid
- It can also used for calculating the average intensity within a given image.
- https://docs.opencv.org/2.4.13.7/modules/imgproc/doc/miscellaneous_transformations.html



Sum = Value(C) - Value(B) - Value(D) + Value(A)

Integral Image (2/3)

0.1	0.2	0.6	0.8
0.2	0.3	0.8	0.6
0.2	0.1	0.6	0.8
0.2	0.1	0.8	0.9

Original image

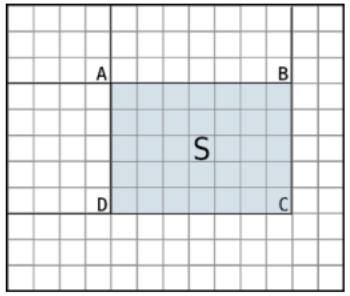
0.1	0.3	0.9	1.7
0.3	0.8	2.2	3.6
0.5	1.1	3.1	5.3
0.7	1.4	4.2	7.3

integral image

Integral Image (3/3)

0.1	0.2	0.6	0.8
0.2	0.3	0.8	0.6
0.2	0.1	0.6	0.8
0.2	0.1	0.8	0.9

0.1 A	0.3	**0.9** B	1.7
0.3	0.8	2.2	3.6
0.5 D	1.1	**3.1** C	5.3
0.7	1.4	4.2	7.3



Calculate the are summation

0.1	0.3	0.9	1.7
0.3	0.8	2.2	3.6
0.5	1.1	3.1 C	5.3
0.7	1.4	4.2	7.3

0.1	0.3	0.9 B	1.7
0.3	0.8	2.2	3.6
0.5	1.1	3.1	5.3
0.7	1.4	4.2	7.3

0.1 A	0.3	0.9	1.7
0.3	0.8	2.2	3.6
0.5	1.1	3.1	5.3
0.7	1.4	4.2	7.3

0.1	0.3	0.9	1.7
0.3	0.8	2.2	3.6
0.5 D	1.1	3.1	5.3
0.7	1.4	4.2	7.3

Sum = Value(C) - Value(B) + Value(A) - Value(D)

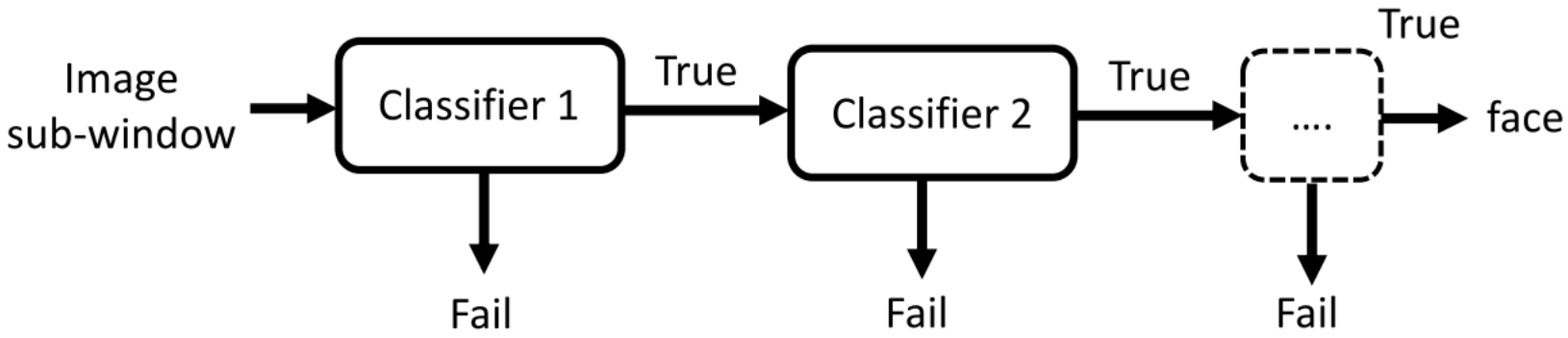
AdaBoost

- Adaptive Boosting
  - Try out multiple weak classifiers over several rounds.
  - Select the best weak classifier in each round and combining the best weak classifiers to create a strong classifier.

Data point	Classifier 1	Classifier 2	Classifier 3	…
P_1	Pass	Fail	Fail	…
P_2	Pass	Pass	Pass	…
P_3	Fail	Pass	Pass	…
…	…	…	…	…

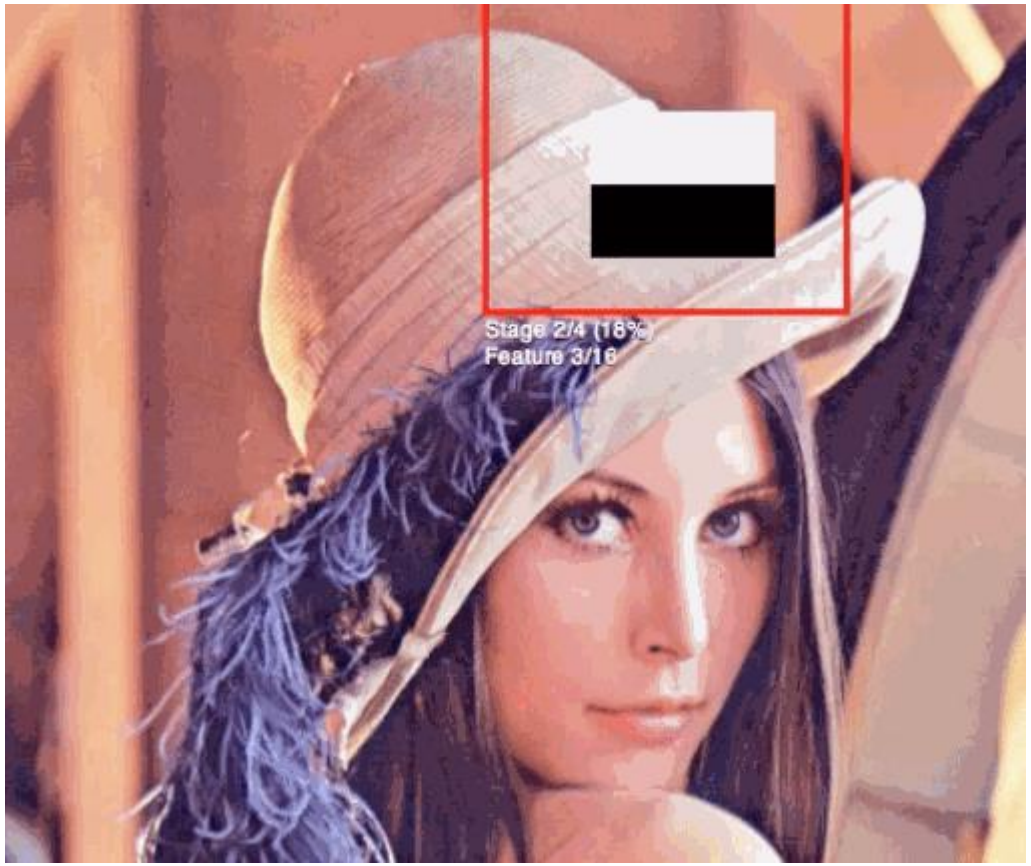
Cascades

- Haar cascades consists of a series of weak classifiers
  - barely better than 50% correct
  - If an area passes a single classifier, go to the next classifier; otherwise, the area doesn't match.



Visualization

- https://vimeo.com/12774628



Recall

```
while True:
    # Capture frame-by-frame
    ret, frame = cap.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(30, 30),
        flags=HAAR_FLAGS
    )
```

Related Parameters

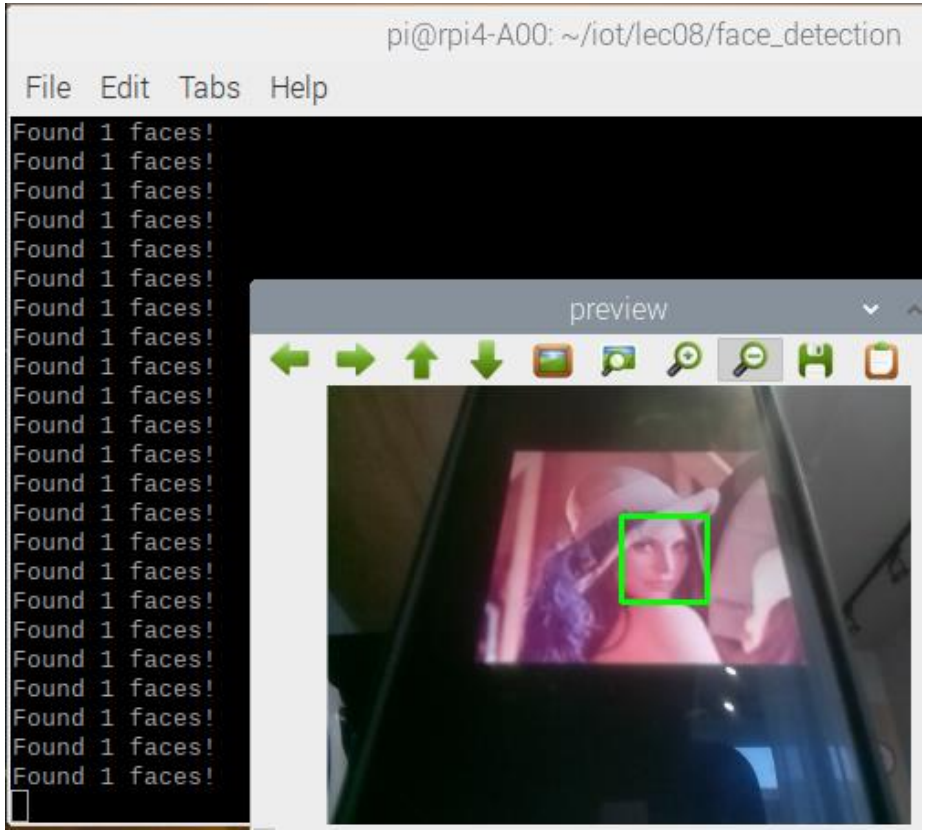
- https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html#cascadeclassifier-detectmultiscale
- detectMultiScale
  - cascade – Haar classifier cascade (OpenCV 1.x API only).It can be loaded from XML or YAML file using Load(). When the cascade is not needed anymore, release it using cvReleaseHaarClassifierCascade(&cascade).
  - image – Matrix of the type CV_8U containing an image where objects are detected.
  - objects – Vector of rectangles where each rectangle contains the detected object.
  - scaleFactor – Parameter specifying how much the image size is reduced at each image scale.
  - minNeighbors – Parameter specifying how many neighbors each candidate rectangle should have to retain it.
  - flags – Parameter with the same meaning for an old cascade as in the function cvHaarDetectObjects. It is not used for a new cascade.
  - minSize – Minimum possible object size. Objects smaller than that are ignored.
  - maxSize – Maximum possible object size. Objects larger than that are ignored.

Try to use different parameters, you will get different results.

Facial Detection

- Get images from camera

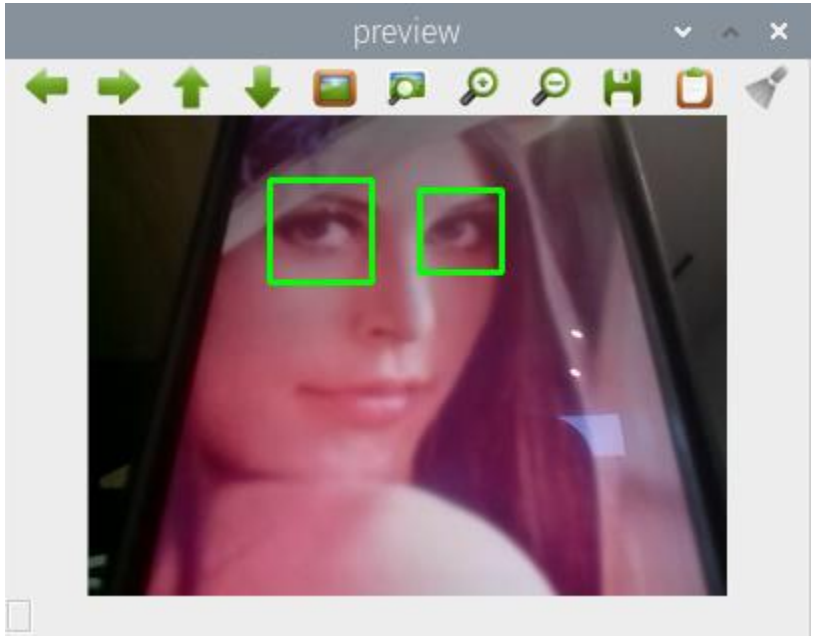
$ python3 camera_face_detect.py



Eyes Detection

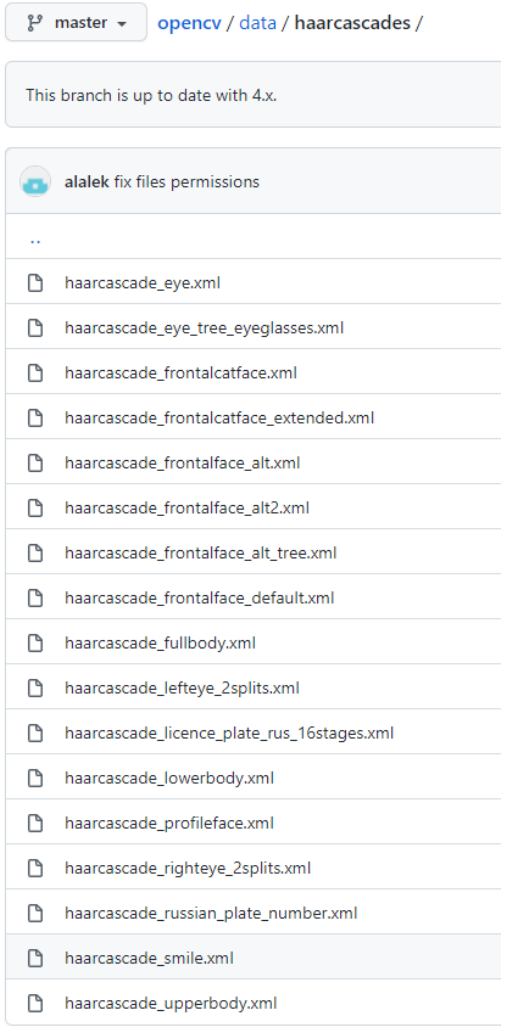
- https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html
- Eyes detection
  - Adjust detectMultiScale()

$ python3 camera_face_detect.py haarcascade_eye_tree_eyeglasses.xml



More Classifiers

- https://github.com/opencv/opencv/tree/master/data/haarcascades



Reference

- Online resource
  - Facial Detection
    - https://www.youtube.com/watch?v=sWTvK72-SPU
  - Computer Vision - Haar-Features
    - https://www.youtube.com/watch?v=F5rysk51txQ
  - Computer Vision - Integral Images
    - https://www.youtube.com/watch?v=x41KFOFGnUE
  - Recognition Part II: Face Detection via AdaBoost
    - https://courses.cs.washington.edu/courses/cse455/16wi/notes/15_FaceDetection.pdf

FAQ

- Q: Program error with “Address in use” or “Camera problem”?
- A: Check if you stop/suspend your program by “ctrl+z”.

```
pi@rpi4-A00:~/iot/lec08/mjpg $ python3 app-camera.py
 * Serving Flask app "app-camera" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:8000/ (Press CTRL+C to quit)
 * Restarting with stat
^Z
[1]+  Stopped                 python3 app-camera.py
```

```
pi@rpi4-A00:~/iot/lec08/mjpg $ python3
Python 3.7.3 (default, Jul 25 2020, 13:03:44)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
[2]+  Stopped                 python3
```

$ jobs

```
pi@rpi4-A00:~/iot/lec08/mjpg $ jobs
[1]-  Stopped                 python3 app-camera.py
[2]+  Stopped                 python3
```

- Resume the program

$ fg <#>

```
pi@rpi4-A00:~/iot/lec08/mjpg $ fg 1
python3 app-camera.py
 * Debugger is active!
 * Debugger PIN: 308-914-434
```

```
pi@rpi4-A00:~/iot/lec08/mjpg $ fg 2
python3
```

- Use “ctrl+c” to terminate the program.
- Use “exit()” to exit python shell.

```
pi@rpi4-A00:~/iot/lec08/mjpg $ fg 2
python3


>>> exit()
```

- Find the program number

$ ps -aux | grep python3

```
pi@rpi4-A00:~/iot/lec08/mjpg $ ps -aux | grep python3
pi        5007  0.5  0.1  14960  7456 pts/0    T    00:02   0:00 python3
pi        5009 16.8  1.6 301500 64460 pts/0    Tl   00:02   0:02 python3 app-cam
era.py
pi        5013 13.3  1.0 297456 40640 pts/0    Tl   00:02   0:01 /usr/bin/python
3 app-camera.py
pi        5019  0.0  0.0   7348   564 pts/0    S+   00:02   0:00 grep --color=au
to python3
```

$ sudo kill -9 <PID#>

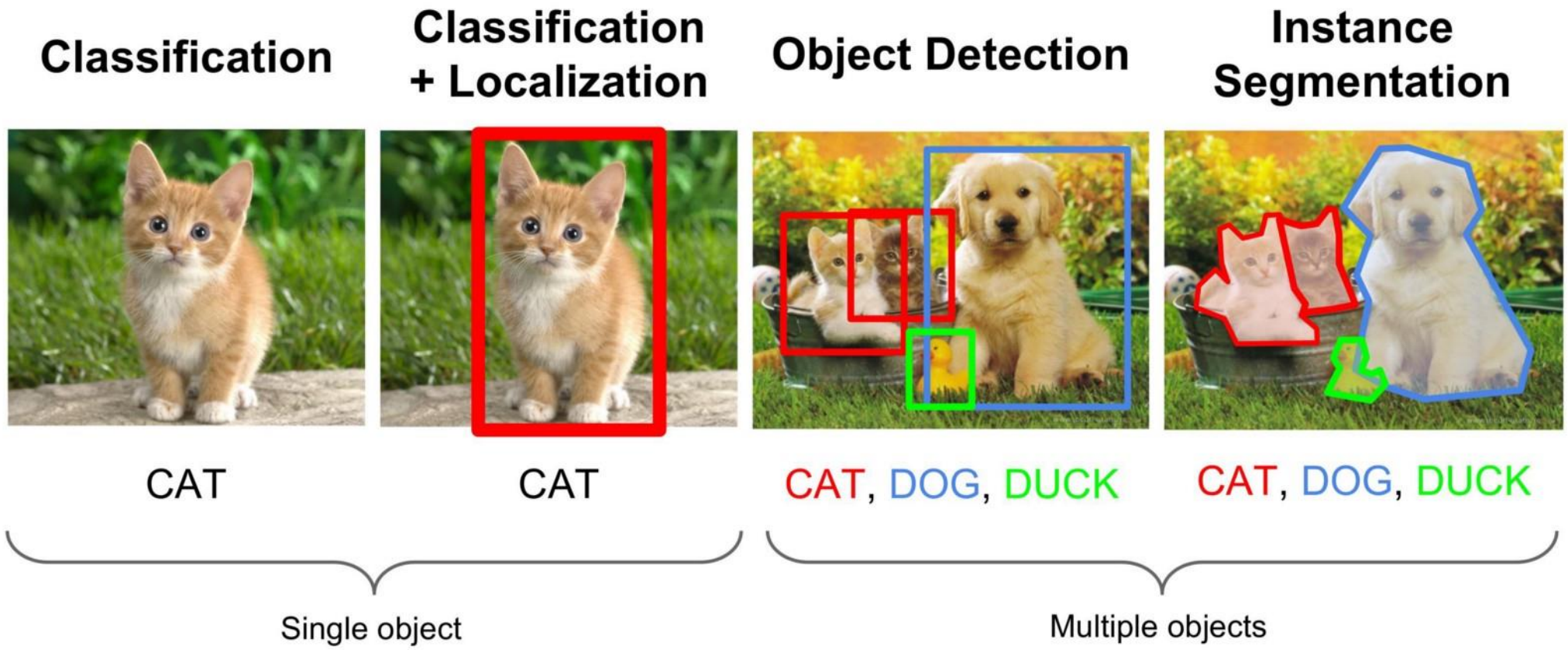
```
pi@rpi4-A00:~/iot/lec08/mjpg $ sudo kill -9 5009
[2]+  Killed                  python3 app-camera.py
```

```
pi@rpi4-A00:~/iot/lec08/mjpg $ ps -aux | grep python3
pi        5007  0.0  0.1  14960  7456 pts/0    T    00:02   0:00 python3
pi        5040  0.0  0.0   7348   568 pts/0    S+   00:04   0:00 grep --color=au
to python3
```

Outline

- Facial recognition
- Object detection

Object Detection



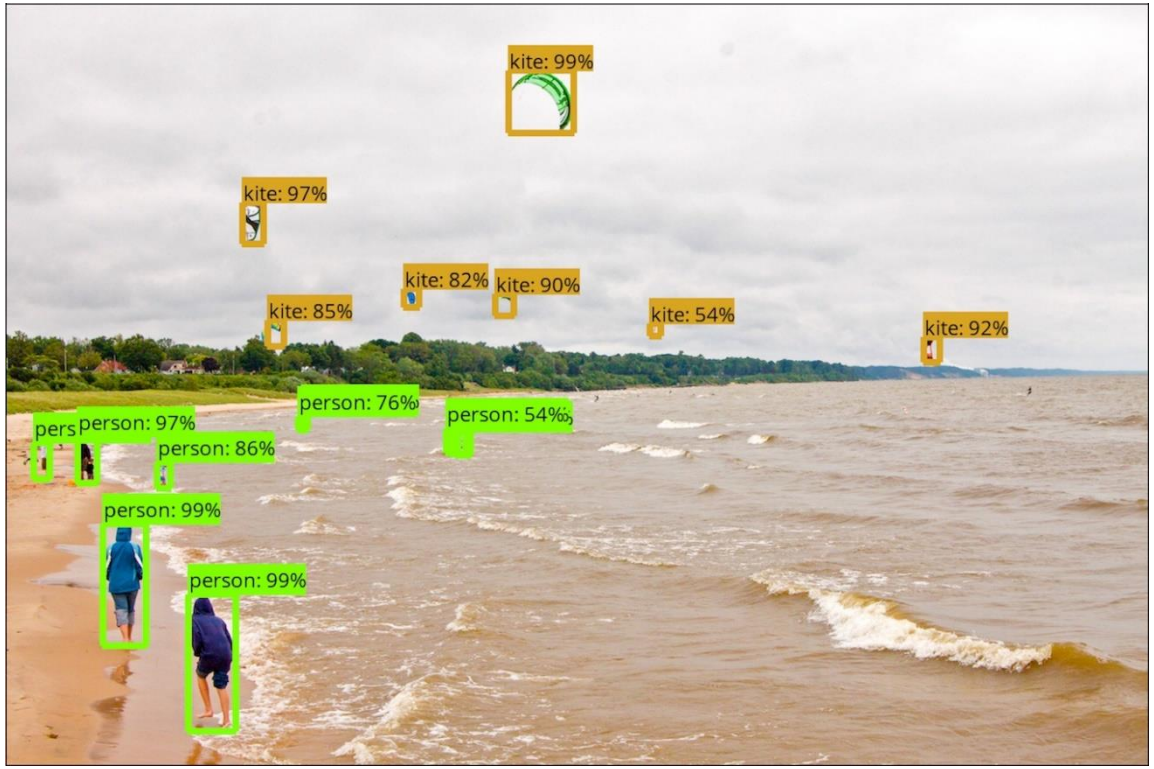
src: https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852

Object Detection

- https://youtu.be/gGqVNuYol6o
- https://github.com/EdjeElectronics/TensorFlow-Object-Detection-on-the-Raspberry-Pi/blob/master/Pet_detector.py

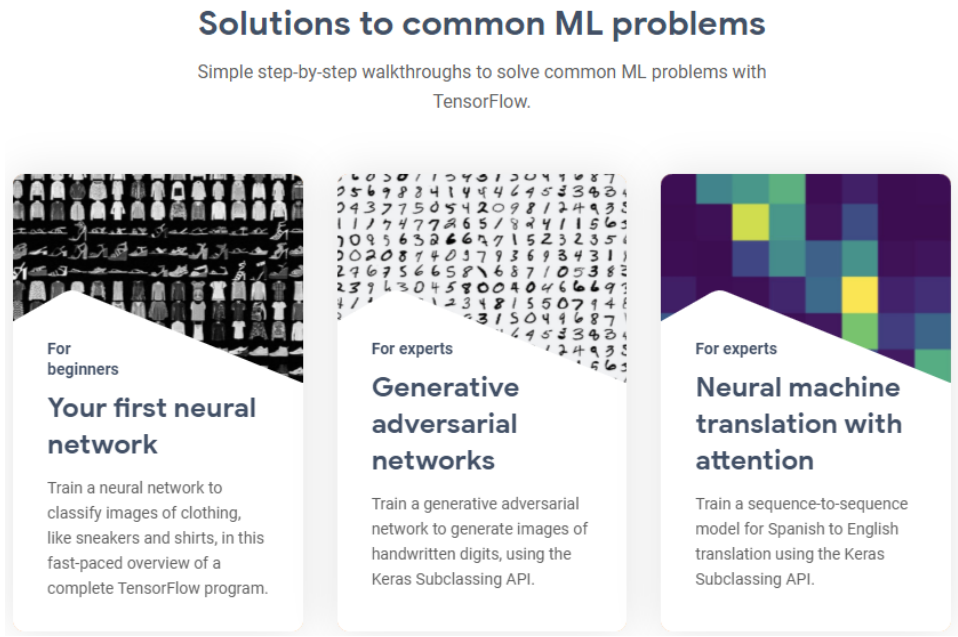
Object Detection of TF

- https://github.com/tensorflow/models/tree/master/research/object_detection
- An open source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models.



TensorFlow

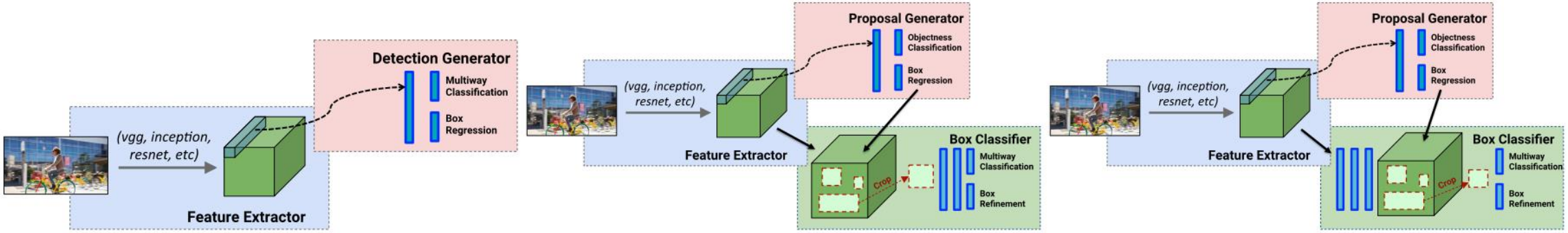
- TensorFlow is an end-to-end open source platform for machine learning.
- It has a comprehensive, flexible ecosystem of tools, libraries, and community resources
  - Let researchers push the state-of-the-art in ML
  - Give developers the ability to easily build and deploy ML-powered applications.



ref: https://tensorflow.org/

Object Detectors

- Speed/accuracy trade-offs for modern convolutional object detectors.



(a) SSD. (b) Faster RCNN. (c) R-FCN.

Figure 1: High level diagrams of the detection meta-architectures compared in this paper.

- "Speed/accuracy trade-offs for modern convolutional object detectors." Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z, Song Y, Guadarrama S, Murphy K, CVPR 2017

Required Packages

- Install dependency packages…
  - Tensorflow
  - Dependencies
  - Protocol Buffers
  - Object Detection API
- Installation reference
  - https://github.com/EdjeElectronics/TensorFlow-Object-Detection-on-the-Raspberry-Pi
  - https://github.com/PINTO0309/Tensorflow-bin/

Installations of TF

- Log into your RPi and open a console to execute the following commands.

```
$ cd ~
$ wget "https://raw.githubusercontent.com/PINTO0309/Tensorflow-bin/master/tensorflow-1.15.0-cp37-cp37m-linux_armv7l_download.sh"
$ sh tensorflow-1.15.0-cp37-cp37m-linux_armv7l_download.sh
$ sudo pip3 install tensorflow-1.15.0-cp37-cp37m-linux_armv7l.whl
```

Installation of Protoc

- Protocol buffer
  - A package that implements Google's Protocol Buffer data format.

$ sudo apt-get install protobuf-compiler

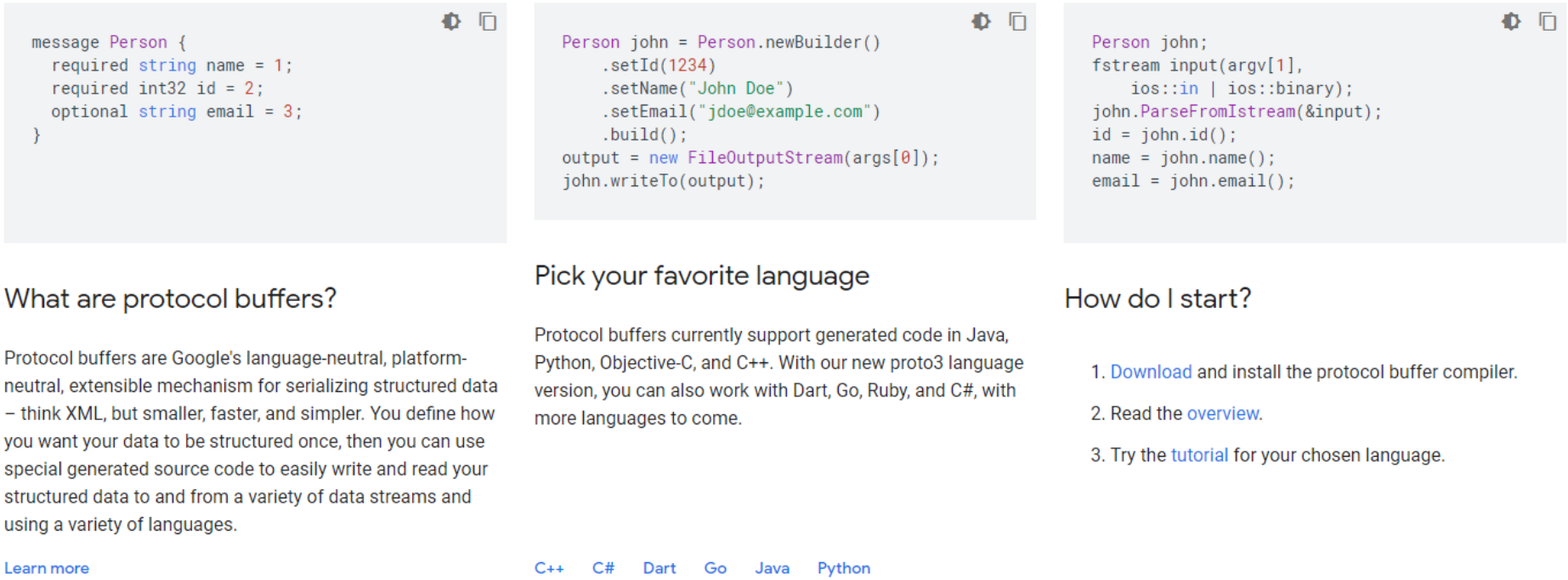
- Confirm if protocol buffer is installed.

$ protoc

```
pi@raspberrypi:~/pocketsphinx-python$ protoc
Usage: protoc [OPTION] PROTO_FILES
Parse PROTO_FILES and generate output based on the options given:
  -IPATH, --proto_path=PATH   Specify the directory in which to search for
                              imports.  May be specified multiple times;
                              directories will be searched in order.  If not
                              given, the current working directory is used.
  --version                   Show version info and exit.
  -h, --help                  Show this text and exit.
  --encode=MESSAGE_TYPE       Read a text-format message of the given type
                              from standard input and write it in binary
                              to standard output.  The message type must
```

Protocol Buffers

- https://developers.google.com/protocol-buffers/
  - Protocol buffers are a language-neutral, platform-neutral extensible mechanism for serializing structured data.



```
message Person {
  required string name = 1;
  required int32 id = 2;
  optional string email = 3;
}
```

```
Person john = Person.newBuilder()
    .setId(1234)
    .setName("John Doe")
    .setEmail("jdoe@example.com")
    .build();
output = new FileOutputStream(args[0]);
john.writeTo(output);
```

```
Person john;
fstream input(argv[1],
    ios::in | ios::binary);
john.ParseFromIstream(&input);
id = john.id();
name = john.name();
email = john.email();
```

TF Models

- Setup TensorFlow directory structure and environment variable.
- Download models and examples built with TensorFlow.

```
$ cd ~
$ mkdir tensorflow1
$ cd tensorflow1
$ git clone --depth 1 https://github.com/tensorflow/models.git
```

Configurations

- Compile the Protocol Buffer files (.proto) used by the Object Detection API.
- The .proto files are located in /research/object_detection/protos.

```
$ cd /home/pi/tensorflow1/models/research
$ protoc object_detection/protos/*.proto --python_out=.
```

- Reboot the system

```
$ sudo reboot
```

Download Models

- Connect to your RPi with VNC Viewer and open a terminal.
- Change directory to the “object detection”

```
$ cd /home/pi/tensorflow1/models/research/object_detection
```

- Download the SSD_Lite model from the TensorFlow detection model zoo.
  - The model zoo is Google’s collection of pre-trained object detection models that have various levels of speed and accuracy.
  - SSD: Single Shot Multibox Detector
  - MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications

```
$ wget http://download.tensorflow.org/models/object_detection/ssdlite_mobilenet_v2_coco_2018_05_09.tar.gz
```

```
$ tar -xzvf ssdlite_mobilenet_v2_coco_2018_05_09.tar.gz
```

Object Detection

- Download a sample program “Object_detection_picamera.py”.

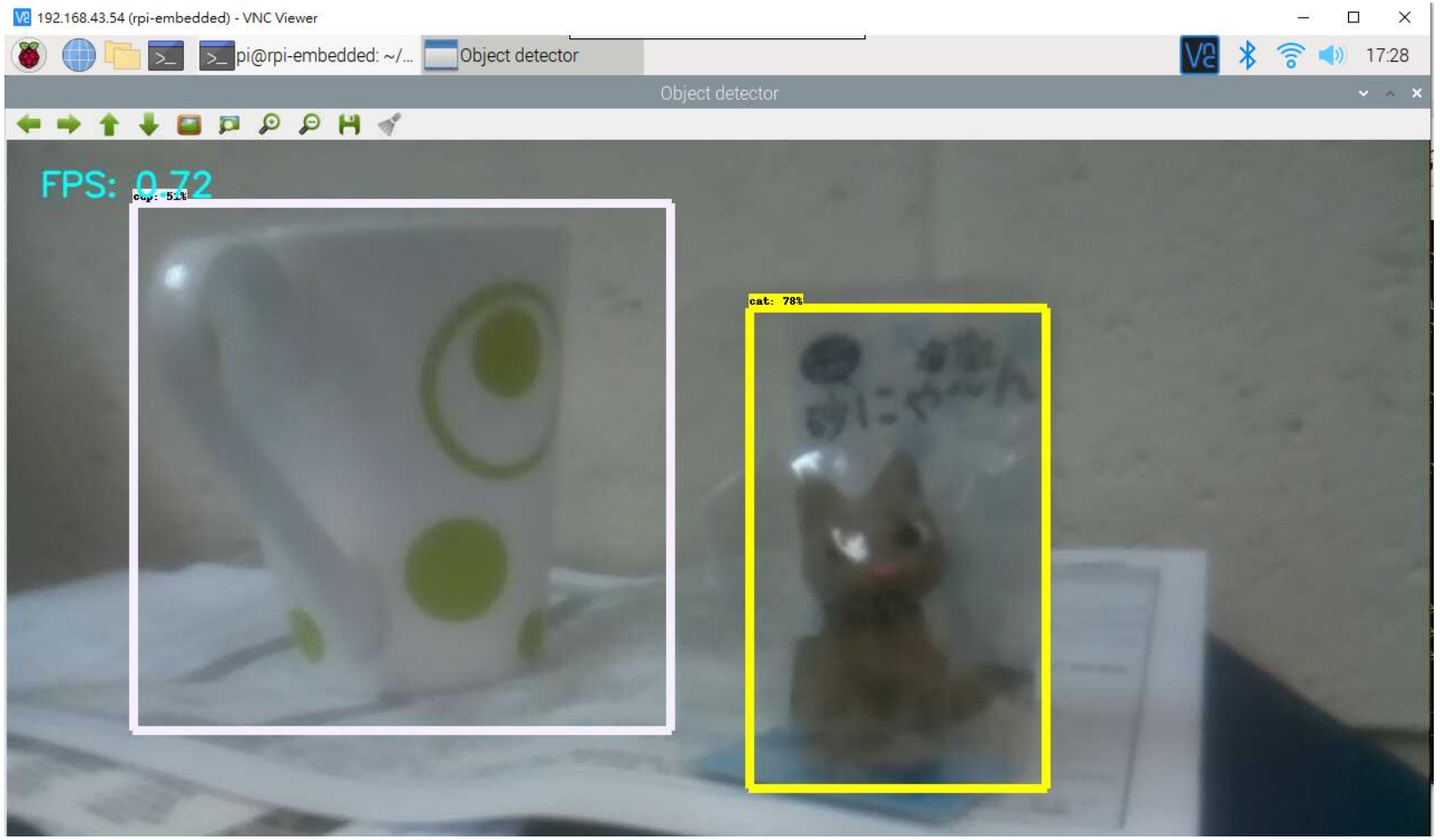
$ https://raw.githubusercontent.com/yachentw/yzucseiot/main/lec08/Object_detection_picamera.py

- Run the sample program

$ python3 Object_detection_picamera.py

  - You have to wait for a while, then a new window will pop up.
  - Press 'q' to quit.

Object Detection



COCO-trained Models

- https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf1_detection_zoo.md

COCO-trained models

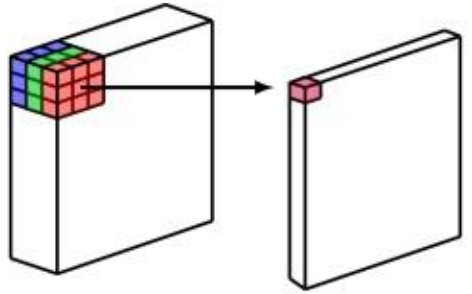
Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v1_0.75_depth_coco ☆	26	18	Boxes
ssd_mobilenet_v1_quantized_coco ☆	29	18	Boxes
ssd_mobilenet_v1_0.75_depth_quantized_coco ☆	29	16	Boxes
ssd_mobilenet_v1_ppn_coco ☆	26	20	Boxes
ssd_mobilenet_v1_fpn_coco ☆	56	32	Boxes
ssd_resnet_50_fpn_coco ☆	76	35	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssd_mobilenet_v2_quantized_coco	29	22	Boxes
ssdlite_mobilenet_v2_coco	27	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes
faster_rcnn_resnet50_lowproposals_coco	64		Boxes
rfcn_resnet101_coco	92	30	Boxes
faster_rcnn_resnet101_coco	106	32	Boxes
faster_rcnn_resnet101_lowproposals_coco	82		Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	620	37	Boxes
faster_rcnn_inception_resnet_v2_atrous_lowproposals_coco	241		Boxes
faster_rcnn_nas	1833	43	Boxes
faster_rcnn_nas_lowproposals_coco	540		Boxes
mask_rcnn_inception_resnet_v2_atrous_coco	771	36	Masks
mask_rcnn_inception_v2_coco	79	25	Masks
mask_rcnn_resnet101_atrous_coco	470	33	Masks
mask_rcnn_resnet50_atrous_coco	343	29	Masks

COCO (Common Objects in Context)

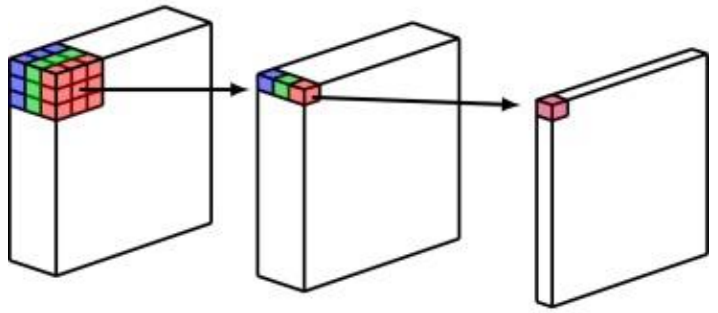
- COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features:
  - Object segmentation
  - Recognition in context
  - Superpixel stuff segmentation
  - 330K images (>200K labeled)
  - 1.5 million object instances
  - 80 object categories
  - 91 stuff categories
  - 5 captions per image
  - 250,000 people with keypoints
- http://cocodataset.org

MobileNet

- Utilize depthwise separable convolutional neural network to reduce the computation.
  - Smaller model size: Fewer number of parameters
  - Smaller complexity: Fewer Multiplications and Additions (Multi-Adds)
- Suitable for portable devices.



(a) Conventional Convolutional Neural Network

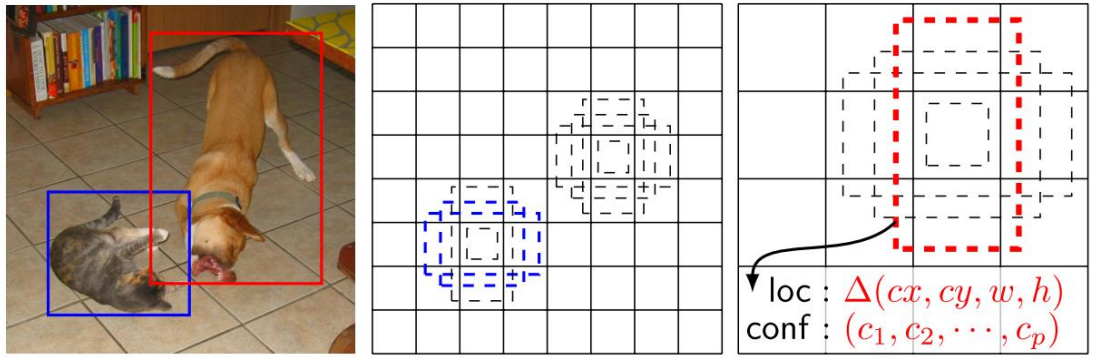


(b) Depthwise Separable Convolutional Neural Network

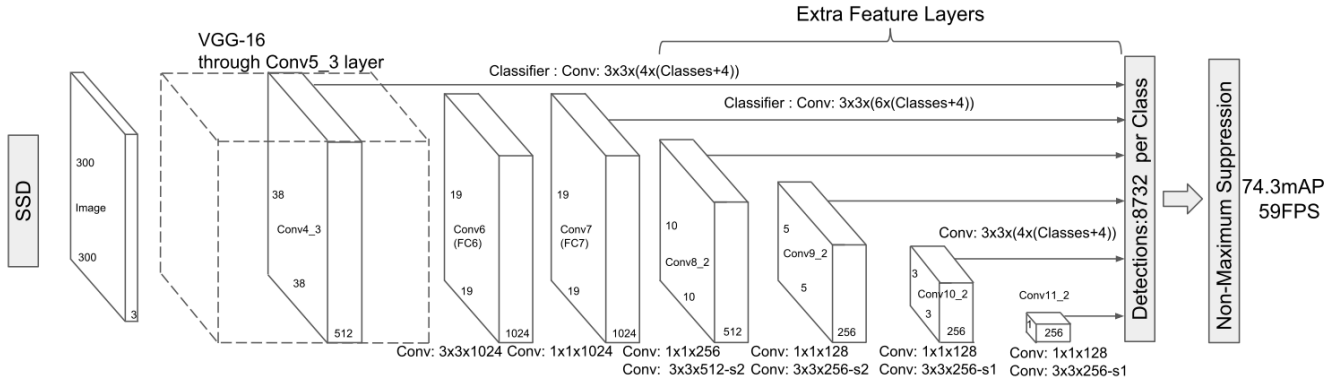
Table 4. Depthwise Separable vs Full Convolution MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

SSD



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map



Tracing Code

- In Object_detection_picamera.py:

```
191         ret, frame = camera.read()
192         frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
193         frame_expanded = np.expand_dims(frame_rgb, axis=0)
194 
195         # Perform the actual detection by running the model with the image as input
196         (boxes, scores, classes, num) = sess.run(
197             [detection_boxes, detection_scores, detection_classes, num_detections],
198             feed_dict={image_tensor: frame_expanded})
199 
200         # Draw the results of the detection (aka 'visulaize the results')
201         vis_util.visualize_boxes_and_labels_on_image_array(
202             frame,
203             np.squeeze(boxes),
204             np.squeeze(classes).astype(np.int32),
205             np.squeeze(scores),
206             category_index,
207             use_normalized_coordinates=True,
208             line_thickness=8,
209             min_score_thresh=0.4)
210         # print(boxes[0][:10])
211         # print(np.squeeze(classes))
212         # print(np.squeeze(scores))
213         # print(category_index)
```

Tracing Code

- print(np.squeeze(classes))

```
[5. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1.]
```

- print(np.squeeze(scores))

```
[0.96914864 0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.        ]
```

- print(category_index)

```
{1: {'id': 1, 'name': 'person'}, 2: {'id': 2, 'name': 'bicycle'}, 3: {'id': 3, '
name': 'car'}, 4: {'id': 4, 'name': 'motorcycle'}, 5: {'id': 5, 'name': 'airplan
e'}, 6: {'id': 6, 'name': 'bus'}, 7: {'id': 7, 'name': 'train'}, 8: {'id': 8, 'n
ame': 'truck'}, 9: {'id': 9, 'name': 'boat'}, 10: {'id': 10, 'name': 'traffic li
ght'}, 11: {'id': 11, 'name': 'fire hydrant'}, 13: {'id': 13, 'name': 'stop sign
'}, 14: {'id': 14, 'name': 'parking meter'}, 15: {'id': 15, 'name': 'bench'}, 16
: {'id': 16, 'name': 'bird'}, 17: {'id': 17, 'name': 'cat'}, 18: {'id': 18, 'nam
e': 'dog'}, 19: {'id': 19, 'name': 'horse'}, 20: {'id': 20, 'name': 'sheep'}, 21
: {'id': 21, 'name': 'cow'}, 22: {'id': 22, 'name': 'elephant'}, 23: {'id': 23,
'name': 'bear'}, 24: {'id': 24, 'name': 'zebra'}, 25: {'id': 25, 'name': 'giraff
e'}, 27: {'id': 27, 'name': 'backpack'}, 28: {'id': 28, 'name': 'umbrella'}, 31:
 {'id': 31, 'name': 'handbag'}, 32: {'id': 32, 'name': 'tie'}, 33: {'id': 33, 'n
ame': 'suitcase'}, 34: {'id': 34, 'name': 'frisbee'}, 35: {'id': 35, 'name': 'sk
is'}, 36: {'id': 36, 'name': 'snowboard'}, 37: {'id': 37, 'name': 'sports ball'}
, 38: {'id': 38, 'name': 'kite'}, 39: {'id': 39, 'name': 'baseball bat'}, 40: {'
id': 40, 'name': 'baseball glove'}, 41: {'id': 41, 'name': 'skateboard'}, 42: {'
id': 42, 'name': 'surfboard'}, 43: {'id': 43, 'name': 'tennis racket'}, 44: {'id
': 44, 'name': 'bottle'}, 46: {'id': 46, 'name': 'wine glass'}, 47: {'id': 47, '
name': 'cup'}, 48: {'id': 48, 'name': 'fork'}, 49: {'id': 49, 'name': 'knife'},
50: {'id': 50, 'name': 'spoon'}, 51: {'id': 51, 'name': 'bowl'}, 52: {'id': 52,
'name': 'banana'}, 53: {'id': 53, 'name': 'apple'}, 54: {'id': 54, 'name': 'sand
wich'}, 55: {'id': 55, 'name': 'orange'}, 56: {'id': 56, 'name': 'broccoli'}, 57
: {'id': 57, 'name': 'carrot'}, 58: {'id': 58, 'name': 'hot dog'}, 59: {'id': 59
, 'name': 'pizza'}, 60: {'id': 60, 'name': 'donut'}, 61: {'id': 61, 'name': 'cak
e'}, 62: {'id': 62, 'name': 'chair'}, 63: {'id': 63, 'name': 'couch'}, 64: {'id'
: 64, 'name': 'potted plant'}, 65: {'id': 65, 'name': 'bed'}, 67: {'id': 67, 'na
me': 'dining table'}, 70: {'id': 70, 'name': 'toilet'}, 72: {'id': 72, 'name': '
tv'}, 73: {'id': 73, 'name': 'laptop'}, 74: {'id': 74, 'name': 'mouse'}, 75: {'i
d': 75, 'name': 'remote'}, 76: {'id': 76, 'name': 'keyboard'}, 77: {'id': 77, 'n
ame': 'cell phone'}, 78: {'id': 78, 'name': 'microwave'}, 79: {'id': 79, 'name':
 'oven'}, 80: {'id': 80, 'name': 'toaster'}, 81: {'id': 81, 'name': 'sink'}, 82:
 {'id': 82, 'name': 'refrigerator'}, 84: {'id': 84, 'name': 'book'}, 85: {'id':
85, 'name': 'clock'}, 86: {'id': 86, 'name': 'vase'}, 87: {'id': 87, 'name': 'sc
issors'}, 88: {'id': 88, 'name': 'teddy bear'}, 89: {'id': 89, 'name': 'hair dri
er'}, 90: {'id': 90, 'name': 'toothbrush'}}
```

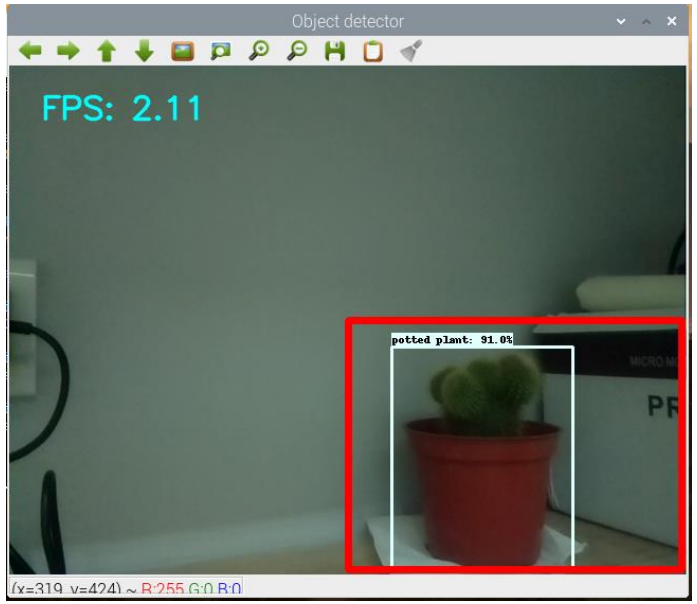
Observations

- Try to detect an object (ex: mouse)
  - Observe the name and probability on bounding box.
  - Check the corresponding value from program messages.
    - Extend the code to print some logs
    - Ex: boxes, scores, classes.
- Explain your observation to the TA.
  - What is the meanings of these 4 values in boxes?

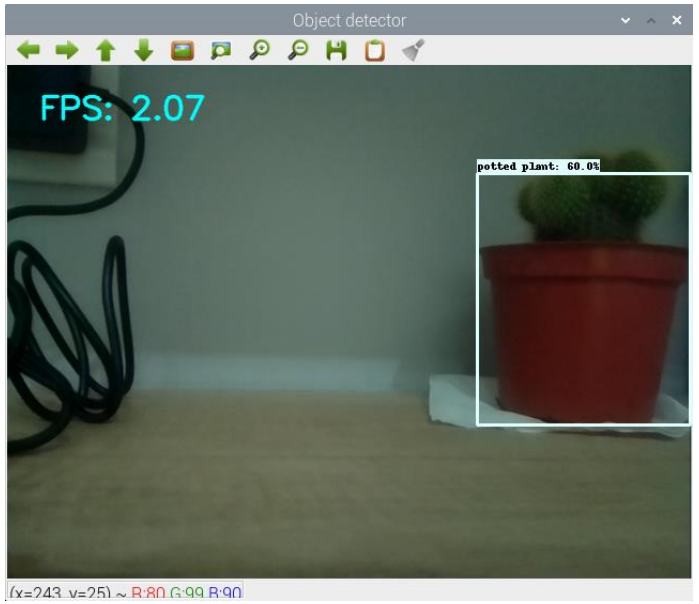
```
[0.05501831 0.06906126 0.44656342 0.5512208 ]
[0.04684409 0.0718818  0.44538224 0.5478414 ]
[0.16795996 0.09565087 0.5515163  0.55543965]
[0.41594383 0.13077407 0.79498076 0.5818809 ]
[0.5129771  0.16718942 0.8876519  0.6002387 ]
[0.50959575 0.1832101  0.8822446  0.61404395]
```

Lab

- Show a red rectangle while an object is in the 4th quadrant.
  - Plot a red rectangle from the center to the bottom right corner.
- "Detected" means that the bounding box of the object is fully included in the 4th quadrant.



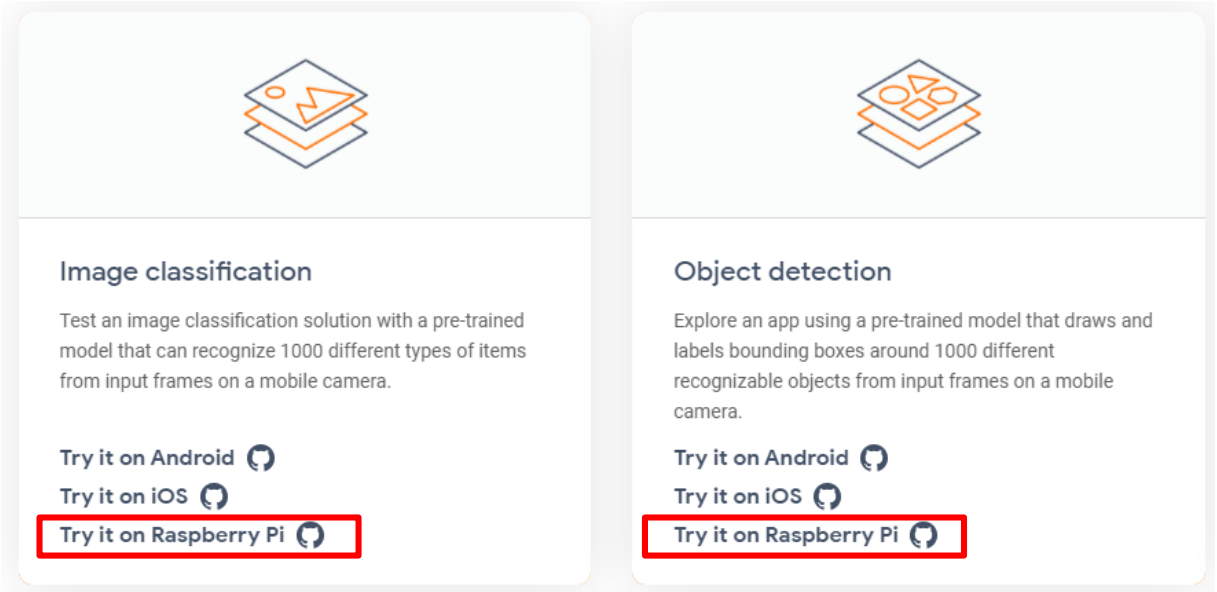
Detected!



Not Detected!

TensorFlow Lite

- TensorFlow Lite is an open source deep learning framework for on-device inference.
- https://www.tensorflow.org/lite/examples/



USB Accelerator

- A USB accessory that brings machine learning inferencing to existing systems. Works with Raspberry Pi and other Linux systems.