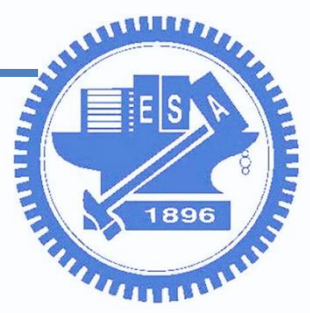


7. Object detection

National Chiao Tung University

Outline

- 網路攝影機
  - 觀看Raspberry Pi Camera的圖片
  - 影像辨識 (opencv + facial detection)
  - **物件辨識 (Object detection)**

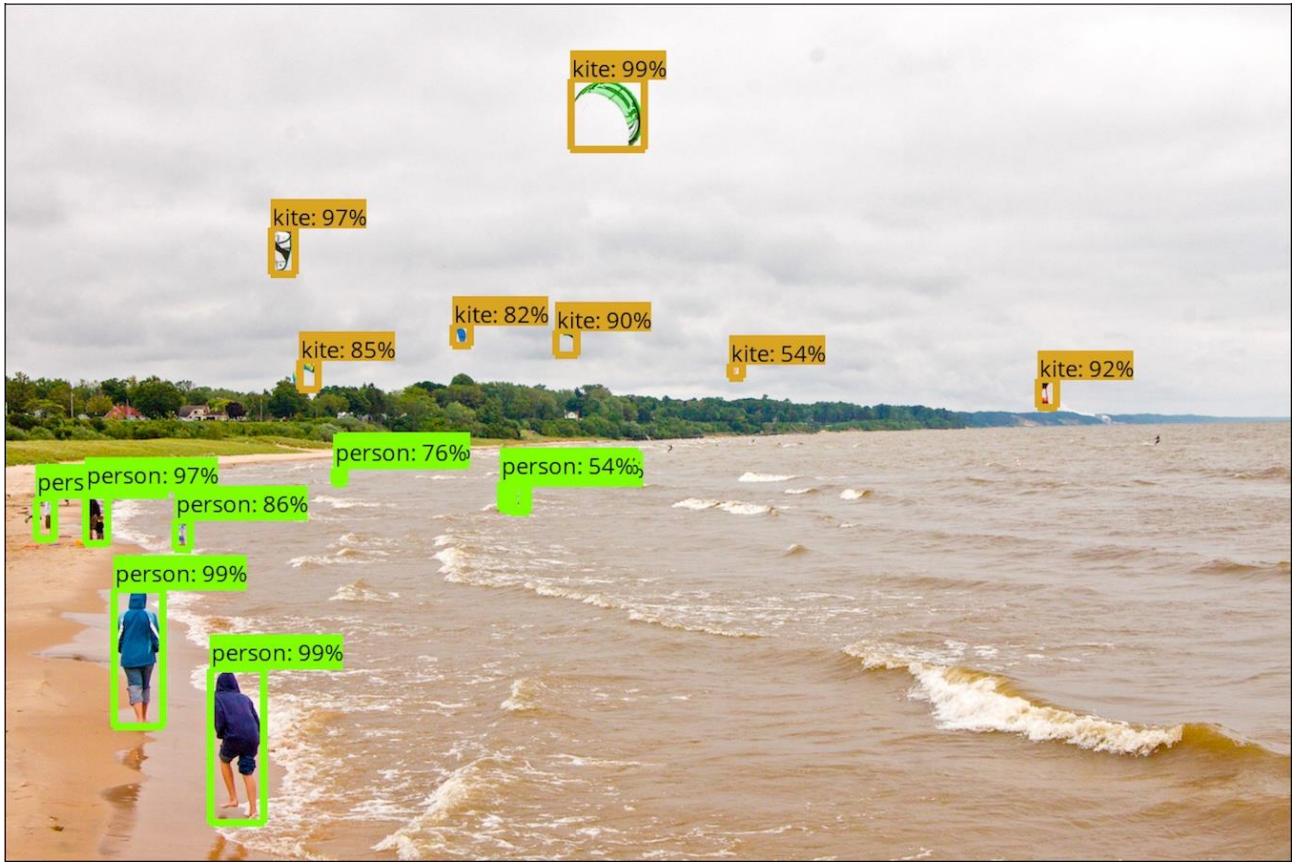
https://www.youtube.com/watch?v=gGqVNuYol6o

Raspberry Pi Pet Detector Camera Using Python, TensorFlow, and Twilio

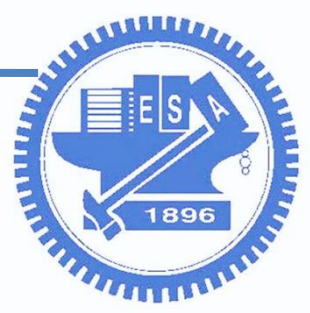
https://github.com/tensorflow/models/tree/master/research/object_detection

Tensorflow Object Detection API

- An open source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models.



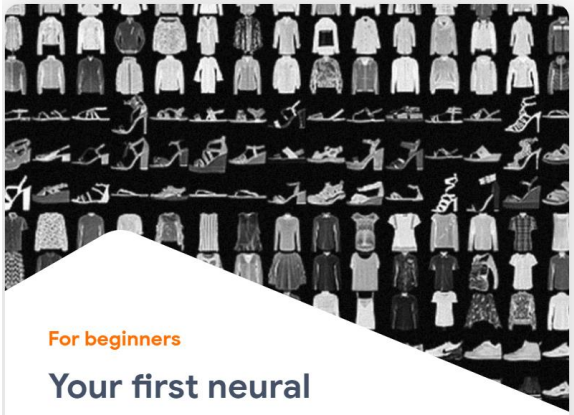
https://www.tensorflow.org/



TensorFlow

Solutions to common ML problems

Simple step-by-step walkthroughs to solve common ML problems with TensorFlow.



For beginners

Your first neural network

Train a neural network to classify images of clothing, like sneakers and shirts, in this fast-paced overview of a complete TensorFlow program.

For experts

Generative adversarial networks

Train a generative adversarial network to generate images of handwritten digits, using the Keras Subclassing API.

For experts

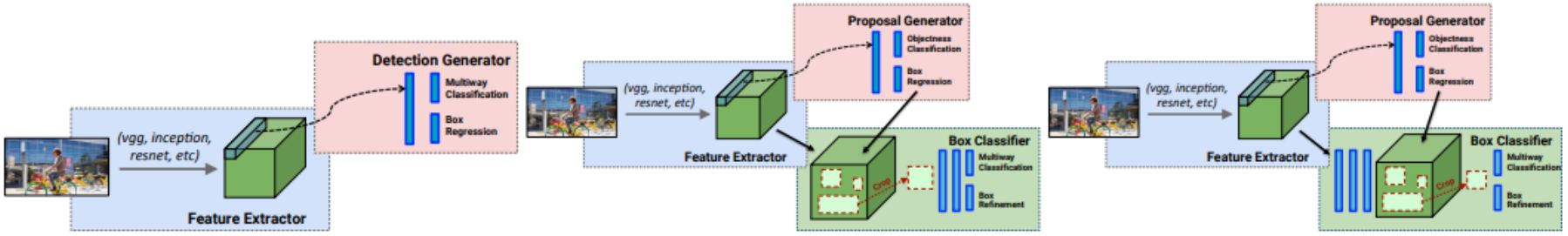
Neural machine translation with attention

Train a sequence-to-sequence model for Spanish to English translation using the Keras Subclassing API.

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

Object Detection

- 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

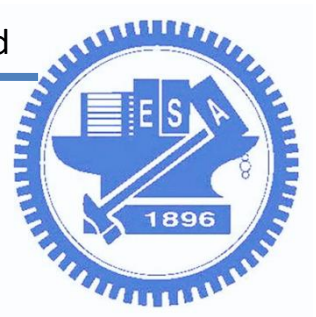
Object detection

- Install dependency packages…
  - Tensorflow
  - Dependencies
  - Protocol Buffers
  - Object Detection API

Install TensorFlow

- Install TensorFlow, it also needs the LibAtlas package.
- libatlas = Automatically Tuned Linear Algebra Software

  - mkdir tf
  - cd tf
  - wget https://github.com/lhelontra/tensorflow-on-arm/releases/download/v1.8.0/tensorflow-1.8.0-cp35-none-linux_armv7l.whl
  - sudo pip3 install /home/pi/tf/tensorflow-1.8.0-cp35-none-linux_armv7l.whl
  - sudo apt-get install libatlas-base-dev



Install dependencies

- Install dependencies that will be used by the TensorFlow Object Detection API.
  - sudo apt-get install libxml2-dev libxslt1-dev
  - sudo pip3 install lxml
  - sudo pip3 install pillow matplotlib cython
  - sudo apt-get install python-tk

Install dependencies

- The object detection scripts in this guide's GitHub repository use OpenCV
- a few dependencies that need to be installed through apt-get.
  - sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev -y
  - sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev -y
  - sudo apt-get install libxvidcore-dev libx264-dev -y
  - sudo apt-get install qt4-dev-tools -y
  - pip3 install opencv-python

Install Protobuf

- Install Protobuf
  - sudo apt-get install autoconf automake libtool curl -y
  - wget https://github.com/protocolbuffers/protobuf/releases/download/v3.5.1/protobuf-all-3.5.1.tar.gz
  - tar -zxvf protobuf-all-3.5.1.tar.gz
  - cd protobuf-3.5.1
  - ./configure
  - make // "make" might cost 60 min to execute
  - sudo make install

The TensorFlow object detection API uses Protobuf
(Google's Protocol Buffer data format)

Install Protobuf

- Install Protobuf – part 2
  - cd python
  - export LD_LIBRARY_PATH=../src/.libs
  - python3 setup.py build --cpp_implementation
  - python3 setup.py test --cpp_implementation
  - sudo python3 setup.py install --cpp_implementation
  - export PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=cpp
  - export PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION_VERSION=3
  - sudo ldconfig

Install Protobuf

- protoc // after install, it prints the help text (default)

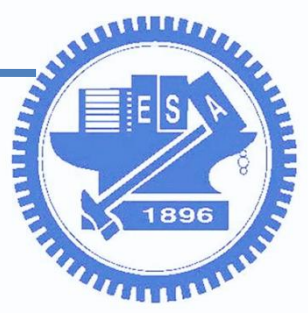
(COM8) [80x26]

連線(C) 編輯(E) 檢視(V) 視窗(W) 選項(O) 說明(H)

```
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
```

- sudo reboot

https://developers.google.com/protocol-buffers/



Protocol Buffers

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

What are protocol buffers?

Protocol buffers are Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data – think XML, but smaller, faster, and simpler. You define how you want your data to be structured once, then you can use special generated source code to easily write and read your structured data to and from a variety of data streams and using a variety of languages.

LEARN MORE

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

Pick your favorite language

Protocol buffers currently support generated code in Java, Python, Objective-C, and C++. With our new proto3 language version, you can also work with Dart, Go, Ruby, and C#, with more languages to come.

C++ C# DART GO JAVA PYTHON

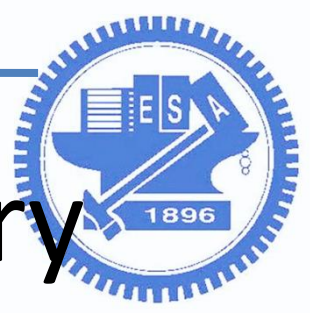
```
Person john;
fstream input(argv[1],
    ios::in | ios::binary);
john.ParseFromIstream(&input);
id = john.id();
name = john.name();
email = john.email();
```

How do I start?

1. Download and install the protocol buffer compiler.
2. Read the overview.
3. Try the tutorial for your chosen language.

Protocol buffers are a language-neutral, platform-neutral extensible mechanism for serializing structured data.

Set up TensorFlow Directory



- Set up TensorFlow Directory Structure and PYTHONPATH Variable
- Download the tensorflow repository from GitHub
  - mkdir tensorflow1
  - cd tensorflow1
  - git clone --recurse-submodules https://github.com/tensorflow/models.git

Set up TensorFlow Directory

- sudo nano ~/.bashrc

- Put the following parameter to .bashrc
- export PYTHONPATH=$PYTHONPATH:/home/pi/tensorflow1/models/research:/home/pi/tensorflow1/models/research/slim
- save and exit

Compile the Protocol Buffer

- Use Protoc to compile the Protocol Buffer (.proto) files used by the Object Detection API
- The .proto files are located in /research/object_detection/protos, but we need to execute the command from the /research directory.

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

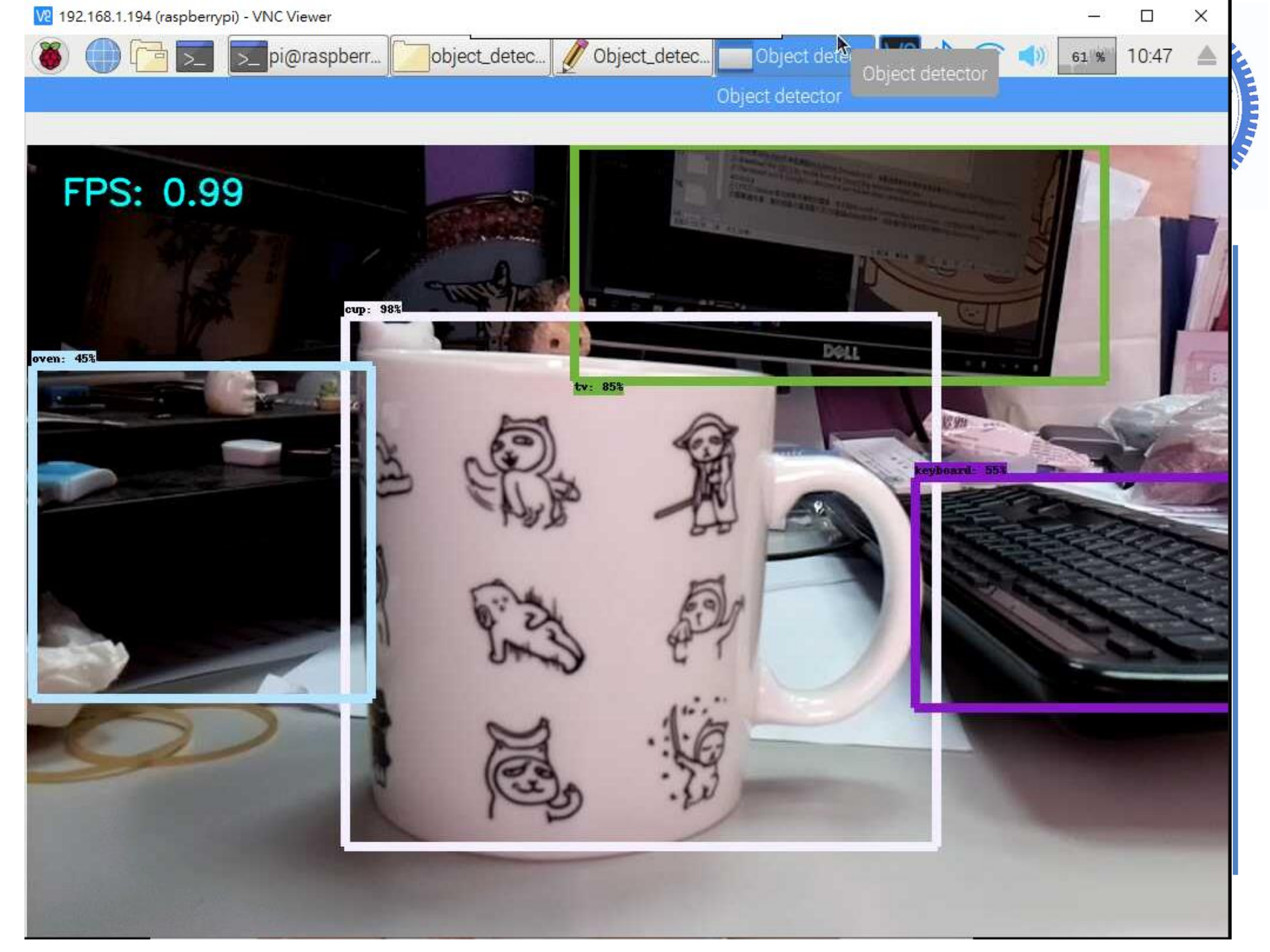
Object detection

- Move into the object_detection directory
  - 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 the Object_detection_picamera.py file into the object_detection directory
  - wget https://raw.githubusercontent.com/EdjeElectronics/TensorFlow-Object-Detection-on-the-Raspberry-Pi/master/Object_detection_picamera.py
  - python3 Object_detection_picamera.py
    - You have to wait for a few minutes, then a new window will pop up
    - Press 'q' to quit

Path location: /home/pi/tensorflow1/models/research/object_detection


192.168.1.194 (raspberrypi) - VNC Viewer
pi@raspberr...
object_detec...
Object_detec...
Object detector
61 %
10:47
Object detector
FPS: 0.99
oven: 45%
cup: 98%
tv: 85%
keyboard: 55%

COCO-trained models

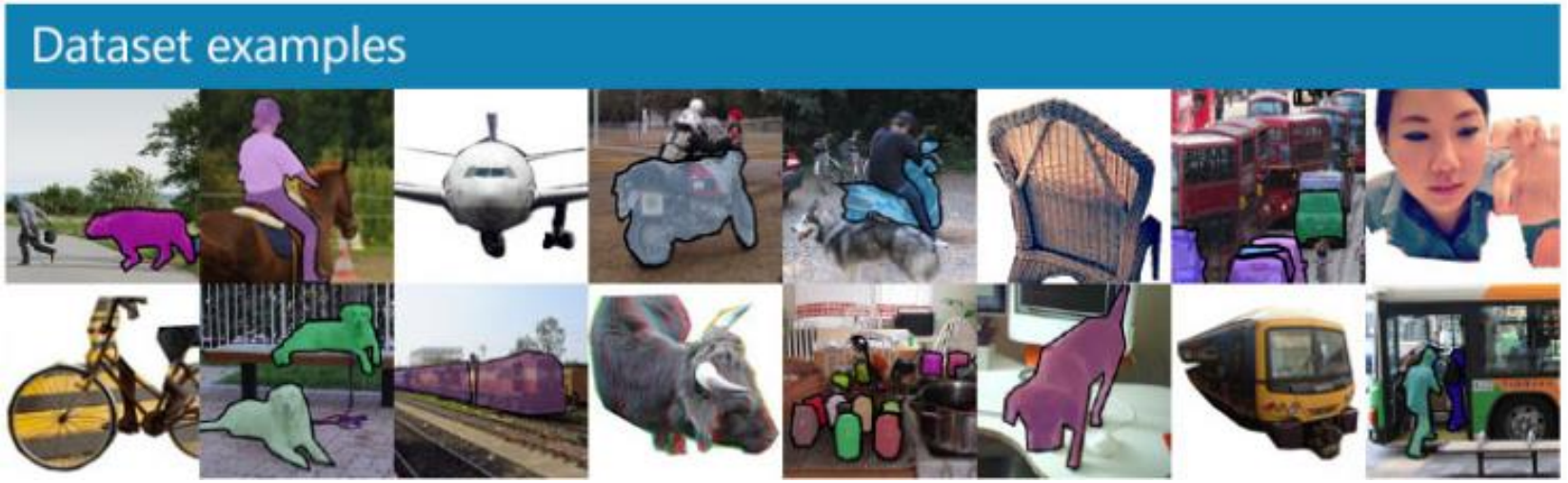
Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v1_0.75_depth_coco ☆	26	18	Boxes
ssd_mobilenet_v1_quantized_coco ☆	29	18	Boxes
ssd_mobilenet_v1_0.75_depth_quantized_coco ☆	29	16	Boxes
ssd_mobilenet_v1_ppn_coco ☆	26	20	Boxes
ssd_mobilenet_v1_fpn_coco ☆	56	32	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

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

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



Discussion

- In Object_detection_picamera.py:

```
# Perform the actual detection by running the model with the image as input
(boxes, scores, classes, num) = sess.run(
    [detection_boxes, detection_scores, detection_classes, num_detections],
    feed_dict={image_tensor: frame_expanded})

# Draw the results of the detection (aka 'visulaize the results')
vis_util.visualize_boxes_and_labels_on_image_array(
    frame,
    np.squeeze(boxes),
    np.squeeze(classes).astype(np.int32),
    np.squeeze(scores),
    category_index,
    use_normalized_coordinates=True,
    line_thickness=8,
    min_score_thresh=0.40)
```

Discussion

- # Draw the results of the detection

```
print (np.squeeze(classes))
[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. 1. 1. 1. 1. 1.
 1. 1. 1. 1.]

print (np.squeeze(scores))
[0.7613534 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.        0.        0.
 0.        0.       ]
```

Discussion

- # Draw the results of the detection

print (category_index)

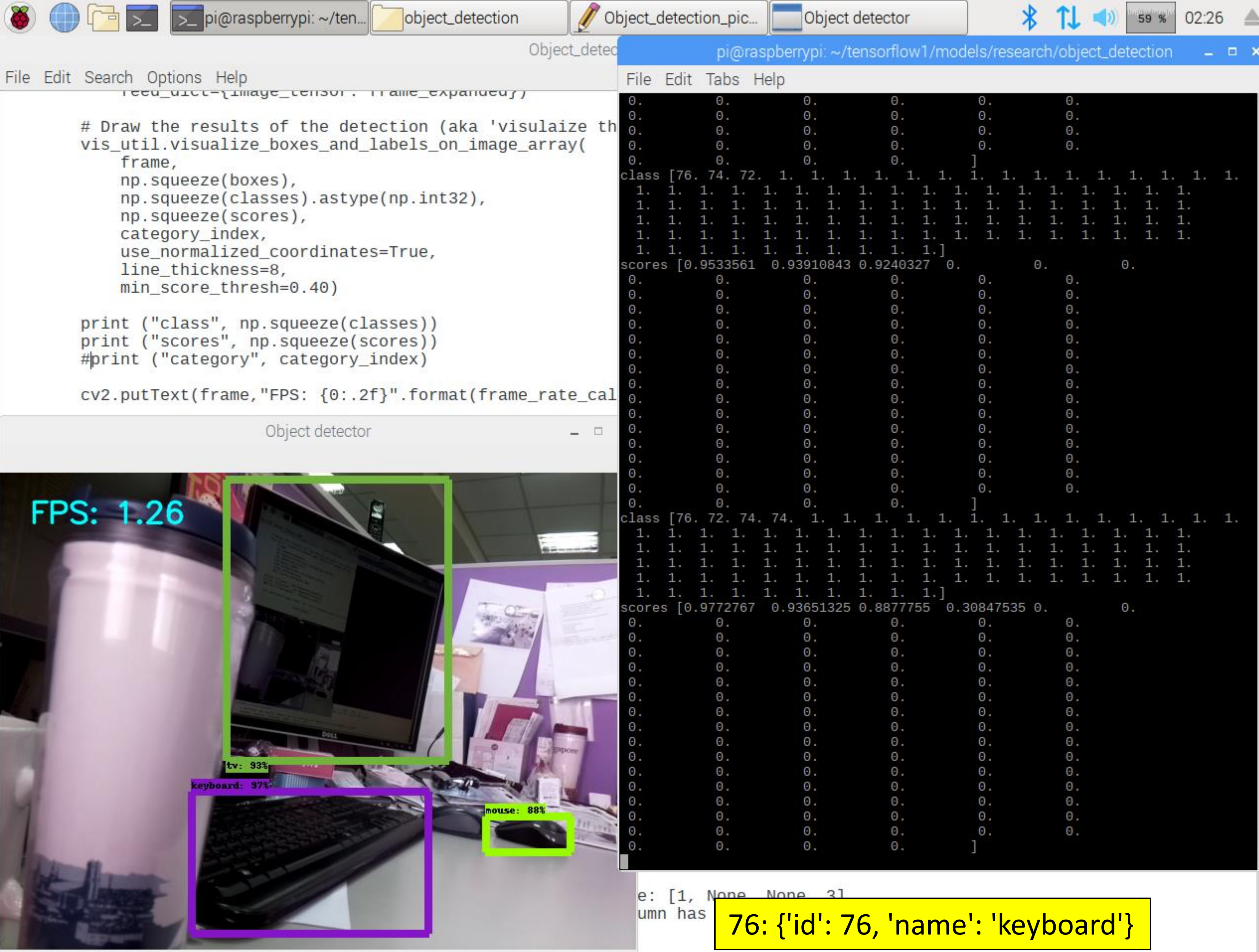
{1: {'id': 1, 'name': 'person'}, 2: {'id': 2, 'name': 'bicycle'}, 3: {'id': 3, 'name': 'car'}, 4: {'id': 4, 'name': 'motorcycle'}, 5: {'id': 5, 'name': 'airplane'}, 6: {'id': 6, 'name': 'bus'}, 7: {'id': 7, 'name': 'train'}, 8: {'id': 8, 'name': 'truck'}, 9: {'id': 9, 'name': 'boat'}, 10: {'id': 10, 'name': 'traffic light'}, 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, 'name': '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': 'giraffe'}, 27: {'id': 27, 'name': 'backpack'}, 28: {'id': 28, 'name': 'umbrella'}, 31: {'id': 31, 'name': 'handbag'}, 32: {'id': 32, 'name': 'tie'}, 33: {'id': 33, 'name': 'suitcase'}, 34: {'id': 34, 'name': 'frisbee'}, 35: {'id': 35, 'name': 'skis'}, 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': 'sandwich'}, 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': 'cake'}, 62: {'id': 62, 'name': 'chair'}, 63: {'id': 63, 'name': 'couch'}, 64: {'id': 64, 'name': 'potted plant'}, 65: {'id': 65, 'name': 'bed'}, 67: {'id': 67, 'name': 'dining table'}, 70: {'id': 70, 'name': 'toilet'}, 72: {'id': 72, 'name': 'tv'}, 73: {'id': 73, 'name': 'laptop'}, 74: {'id': 74, 'name': 'mouse'}, 75: {'id': 75, 'name': 'remote'}, 76: {'id': 76, 'name': 'keyboard'}, 77: {'id': 77, 'name': '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': 'scissors'}, 88: {'id': 88, 'name': 'teddy bear'}, 89: {'id': 89, 'name': 'hair drier'}, 90: {'id': 90, 'name': 'toothbrush'}}

Discussion

- Try to detect an object
  - Observe the **name** and **probability** on bounding box
  - Try to find **the corresponding value** from log
    - Extend the code, print the log
    - Hint: boxes, scores, classes

```
print (np.squeeze(classes))
[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. 1. 1. 1. 1. 1. 1. 1. 1. 1.
 1. 1. 1. 1.]

print (np.squeeze(scores))
[0.7613534 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.        0.        0.
 0.        0.       ]
```



76: {'id': 76, 'name': 'keyboard'}

Summary

- Write down the answer for discussion
  - **Discussion (**Deadline: Before 5/3, 12:00)
    - **Try to detect an object, understand the value from frame and code**
    - **Put your student ID on frame**
    - **Upload your observation to e3**

- Next week is midterm!
- Next week is midterm!
- Next week is midterm!

Reference for PI camera

- Raspberry Pi Camera + Python
  - https://www.slideshare.net/raspberrypi-tw/raspberry-pi-camera-python
- Raspberry Pi Camera + Python + OpenCV (Day1)
  - https://www.slideshare.net/raspberrypi-tw/raspberry-pi-camera-python-opencv-day1
- Raspberry Pi Camera + Python + OpenCV (Day2)
  - https://www.slideshare.net/raspberrypi-tw/raspberry-pi-camera-and-opencv-day2