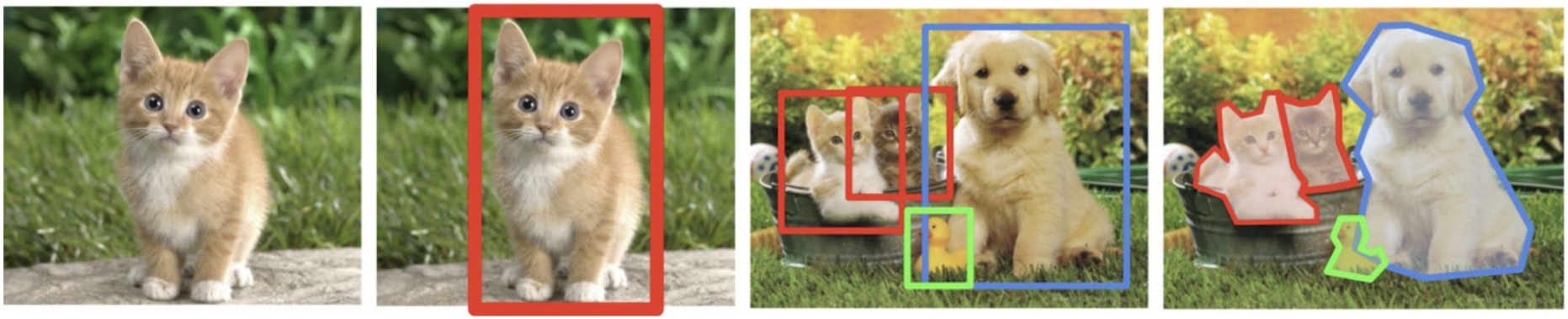
Computer Vision for Beginners (Raspberry Pi) v8



SINGAPORE WORKFORCE SKILLS QUALIFICATIONS

Trainer: Man Guo Chang

TERTIARY INFOTECH

Website:www.tertiarycourses.com.sg
Email: enquiry@tertiaryinfotech.com

About the Trainer

Mr. Man Guo Chang graduated from Nanyang Technological University, School of Electrical and Electronic Engineering, major in Computer Engineering.

Mr. Man has more than 25 years of working experience in the Semiconductor field, specialized in IC Testing, Product Engineering, Data Analysis, and Software Development.

Mr. Man is an ACTA certified trainer. His skill set includes Website Development, Software Development, OpenCV, Machine Vision, Internet of Things, ROS, Cyber Security, etc.

Let's Know Each Other...

Say a bit about yourself
- What is your Name
- What Industry you are from?
- Do you have any prior knowledge in Python and Computer Vision?
- Why you want to learn Computer Vision?
- What do you expect to learn from this course?

Ground Rules

- Set your mobile phone to silent mode
- Participate actively in the class. No question is stupid.
- Mutual respect. Agree to disagree.
- One conversation at one time.
- Be punctual. Back from breaks on time.
- Exit the class silently if you need to step out for phone call, toilet break etc.
- 75% attendance is required

Ground Rules for Virtual Training

- Upon entering, mute your mic and turn on the video. Use a headset if you can
- Use the 'raise hand' function to indicate when you want to speak
- Participant actively. Feel free to ask questions on the chat whenever.
- Facilitators can use breakout rooms for private sessions.

WSQ and SSG TG Application Form

Please fill up the following WSQ and SSG TG Application Form for TRACOM survey, e-cert generation and WSQ funding

https://forms.gle/pJ2WxHZ3fyRbDLVu6

Digital Attendance

- You need to take digital attendance in the AM and PM.
- Please download the mySkillsFuture apps below to take your digital attendance for WSQ and SFC courses.

Guidelines for Facilitators

1. Once all the participants are in and introduce themselves
2. Goto gallery mode, take a snapshot of the class photo - makes sure capture the date and time
3. Start the video recording (only for WSQ courses)
4. Continue the class
5. Before the class end on that day, take another snapshot of the class photo - makes sure capture the date and time
6. For NRIC verification, facilitator to create breakout room for individual participant to check (only for WSQ courses)
7. Before the assessment start, take another snapshot of the class photo - makes sure capture the date and time (only for WSQ courses)
8. For Oral Questioning assessment, facilitator to create breakout room for individual participant to OQ (only for WSQ courses)
9. End the video recording and upload to cloud (only for WSQ courses)
10. Assessor to send all the assessment records, assessment plan and photo and video to the staff (only for WSQ courses).

Prerequisite

This course assumes the following knowledge

- Basic Python

Learning Outcomes

By end of the course, learners should be able to

- understand basic vision systems concepts and applications
- apply basic computer vision techniques
- implement image classification
- implement object detection and video analytics
- evaluate edge and cloud-based computer vision system

Agenda

Topic 1 Overview of Computer Vision with Deep Learning

- Introduction to Computer Vision
- Computer Vision Business Applications
- Classical vs Deep Learning Computer Vision

Topic 2 Basic Computer Vision Techniques

- Setup Portable Mini Computer Board
- Setup Camera
- Setup OpenCV
- Image Processing with OpenCV

Topic 3 Image Classification

- Introduction to Deep Learning and Neural Network
- Overview of Convolutional Neural Network (CNN)
- Pre-Trained Vision Models
- Image Classification with Pre-Trained CNN

Agenda

Topic 4 Object Detection

- Overview of Single Shot Detector (SSD) Object Detection Technique
- Object Detection and Video Tracking with SSD
- Overview of YOLO Object Detection Technique
- Object Detection and Video Tracking using YOLO
- Basic Video Analytics

Topic 5 Computer Vision Systems

- Edge Based Vision System
- Cloud Based Vision System

Final Assessment

- Written Assessment(Q&A)
- Practicum Performance

Download Course Material

- You can download the course material from Google classroom https://classroom.google.com
- Enter the class code below to join the class on the top right.
- Goto Classwork>> Course material
- If you cannot access the Google Classroom, please inform trainer or staff.

h4nrjjy

CERTIFICATE

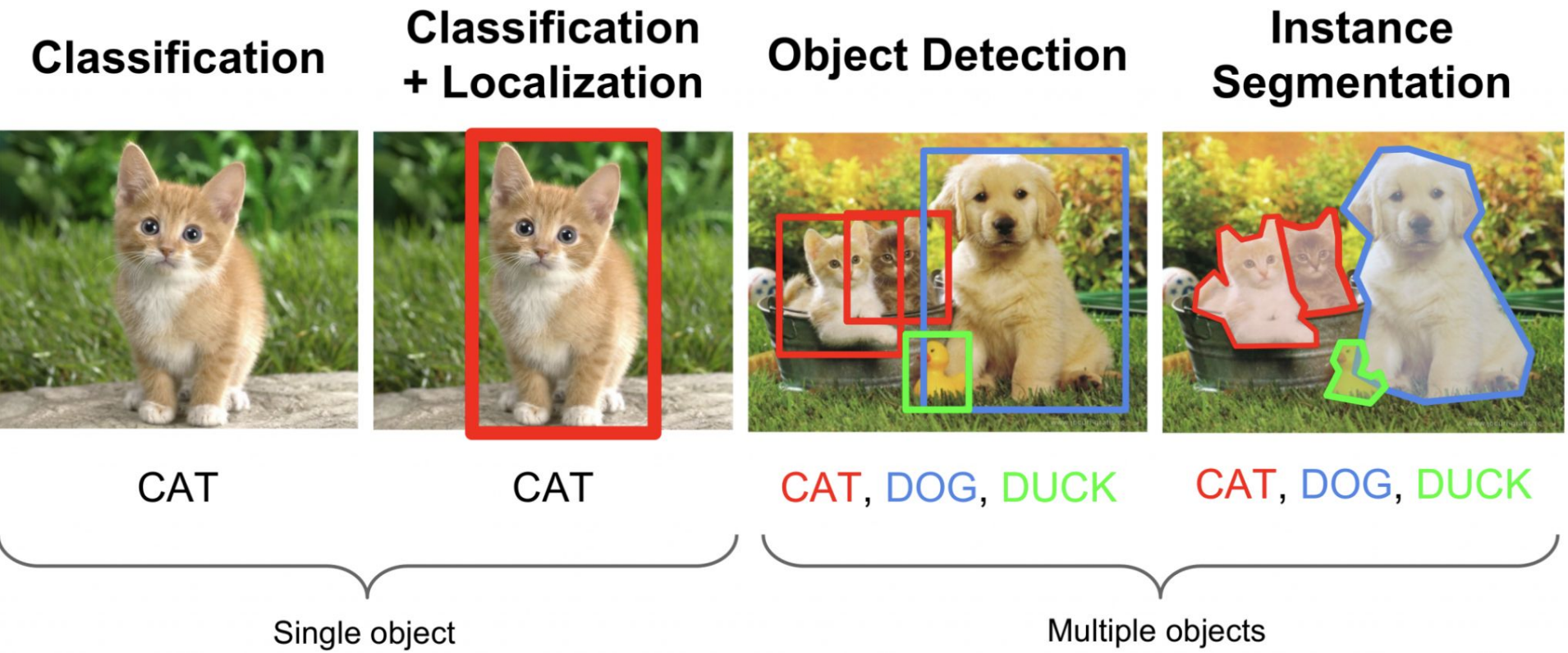
Two e-certificates will be awarded to trainees who have demonstrated competency in the WSQ assessment and achieved at least 75% attendance.

- A SkillsFuture WSQ Statement of Attainment (SOA) issued by WSG. Typically take 4 weeks
- Certification of Completion issued by Tertiary Infotech Pte Ltd, immediately after the course

Topic 1
Overview of Computer Vision with Deep Learning

Overview of Computer Vision

- Computer vision is a field of artificial intelligence (AI) enabling computers to derive information from images, videos and other inputs
- It consists of image classification, object detection and image segmentation.



Applications of Computer Vision

Public Safety and Home Security

Computer vision with image and facial recognition helps quickly identify unlawful entries or persons of interest, resulting in safer communities and a more effective way of deterring crimes.

Authentication and Enhanced Computer-human interaction

Enhanced human-computer interaction improves customer satisfaction such as offering products based on customer sentiment analysis in retail outlets or faster banking services with quick authentication based on customer identity and preferences.

Content Management and Analysis

With millions of images added every day to media and social channels. The use of computer vision technologies such as metadata extraction and image classification greatly improves efficiency and revenue opportunities.

Autonomous Driving

Using computer vision technologies. Auto manufacturers can provide improved and safer self-driving car navigation realizing the goal of making autonomous driving a reality and a reliable transportation option.

Medical Imaging

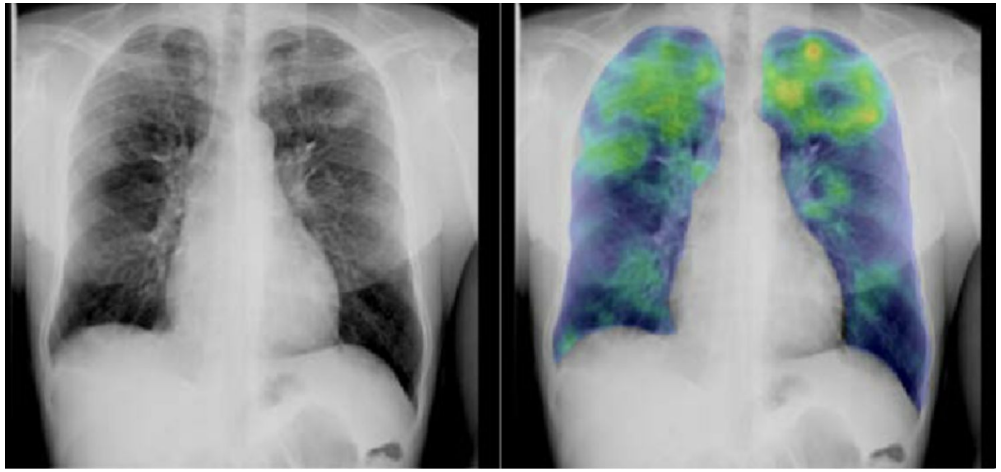
Medical image analysis with computer vision can greatly improve the accuracy and speed of a patient's medical diagnosis, resulting in better treatment outcomes and life expectancy.

Manufacturing Process Control

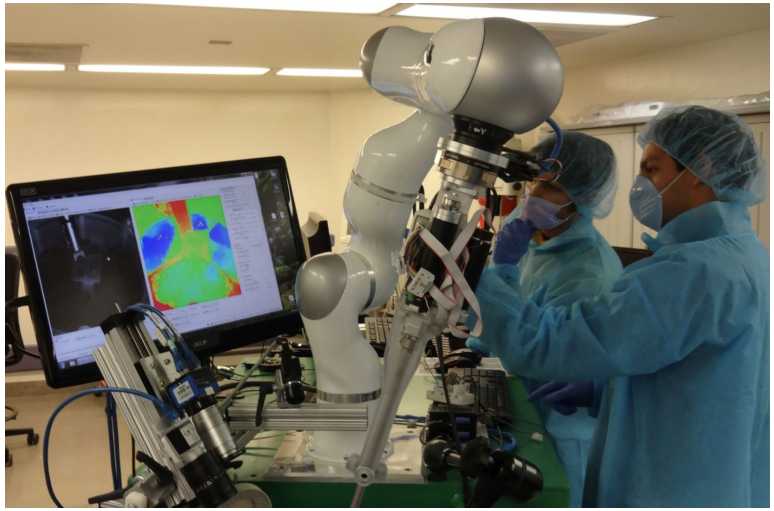
Well-trained computer vision incorporated in robotics improves quality assurance and operational efficiencies in manufacturing applications, resulting in more reliable and cost effective products.

CV Applications on Health Care

Identifying Tuberculosis

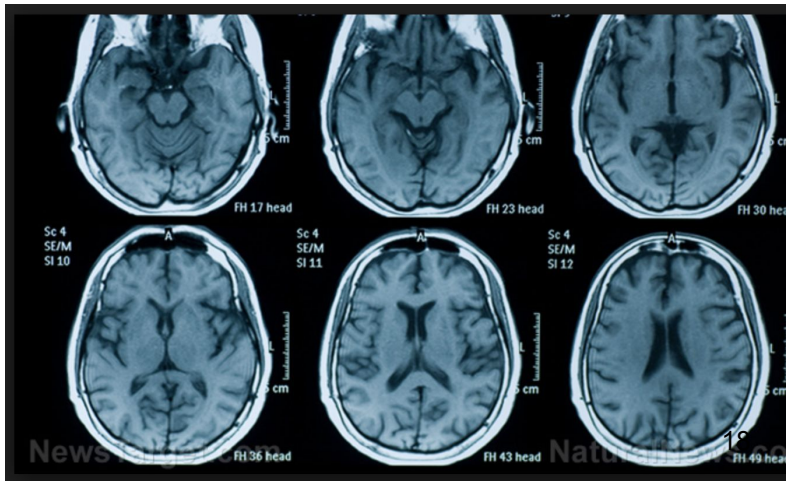


Robotics-Assisted Surgery



Detecting Brain Bleeds

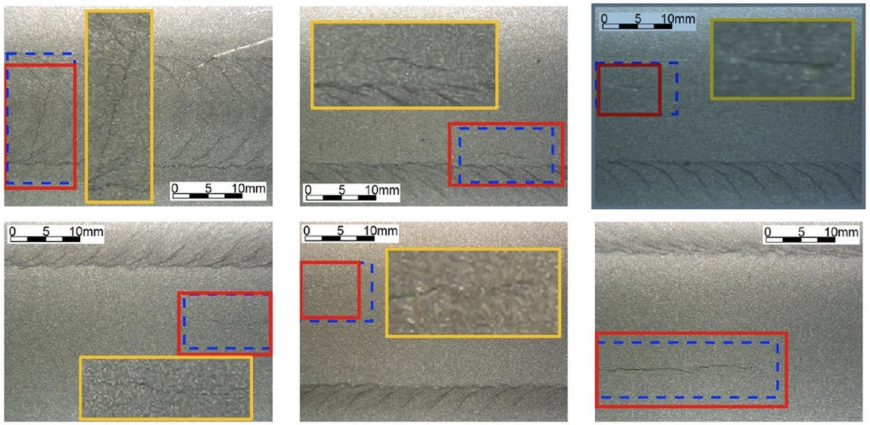
Detecting Alzheimer



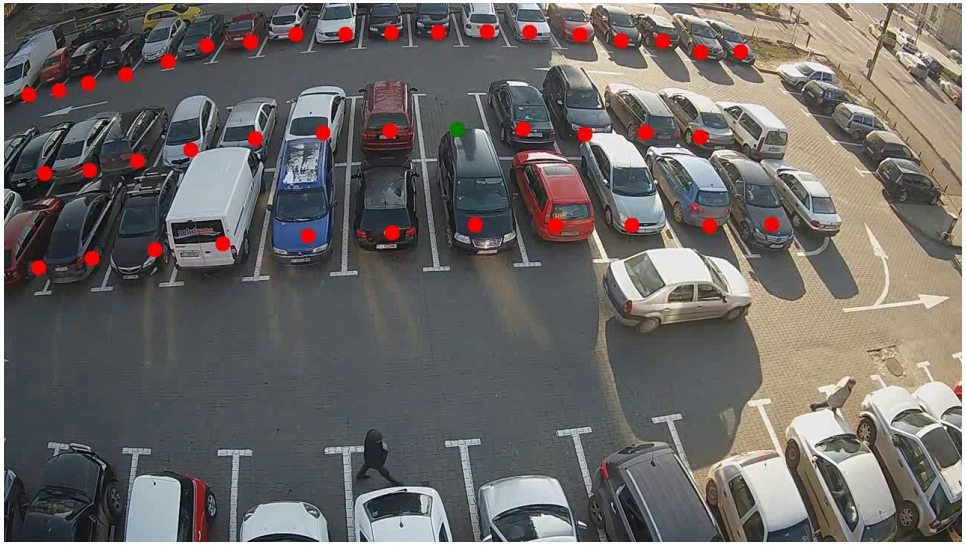
CV Applications on Security/Mfg

Face recognition

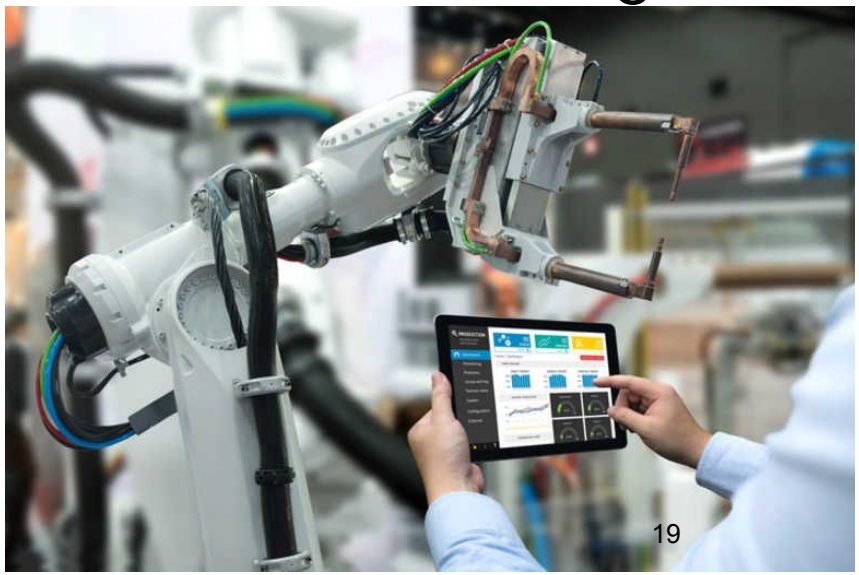
Defect Detection



Parking Spotter



Robots in Mfg

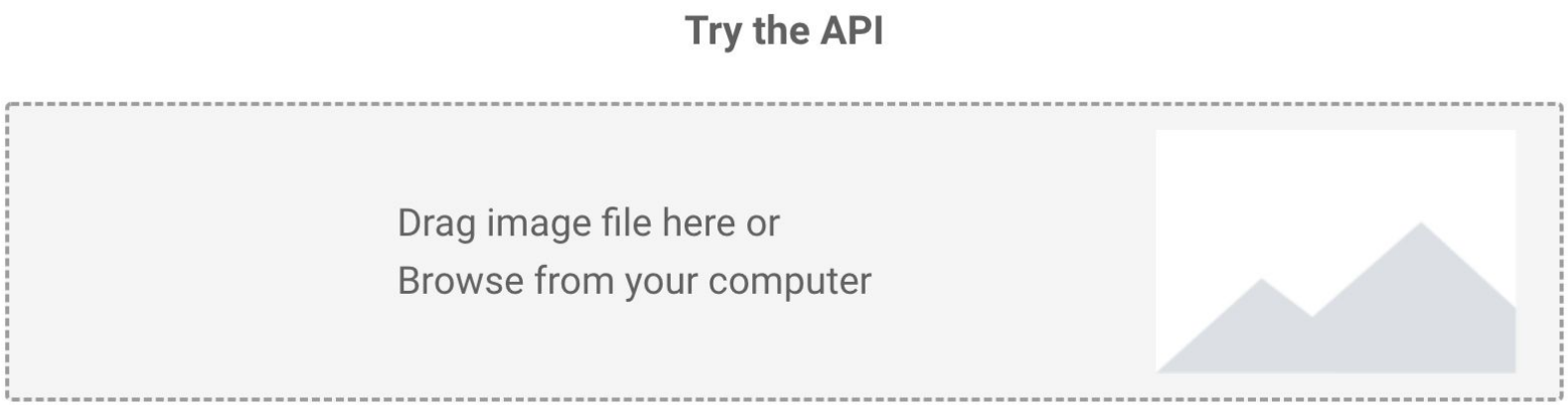


Activity: Computer Vision Demo

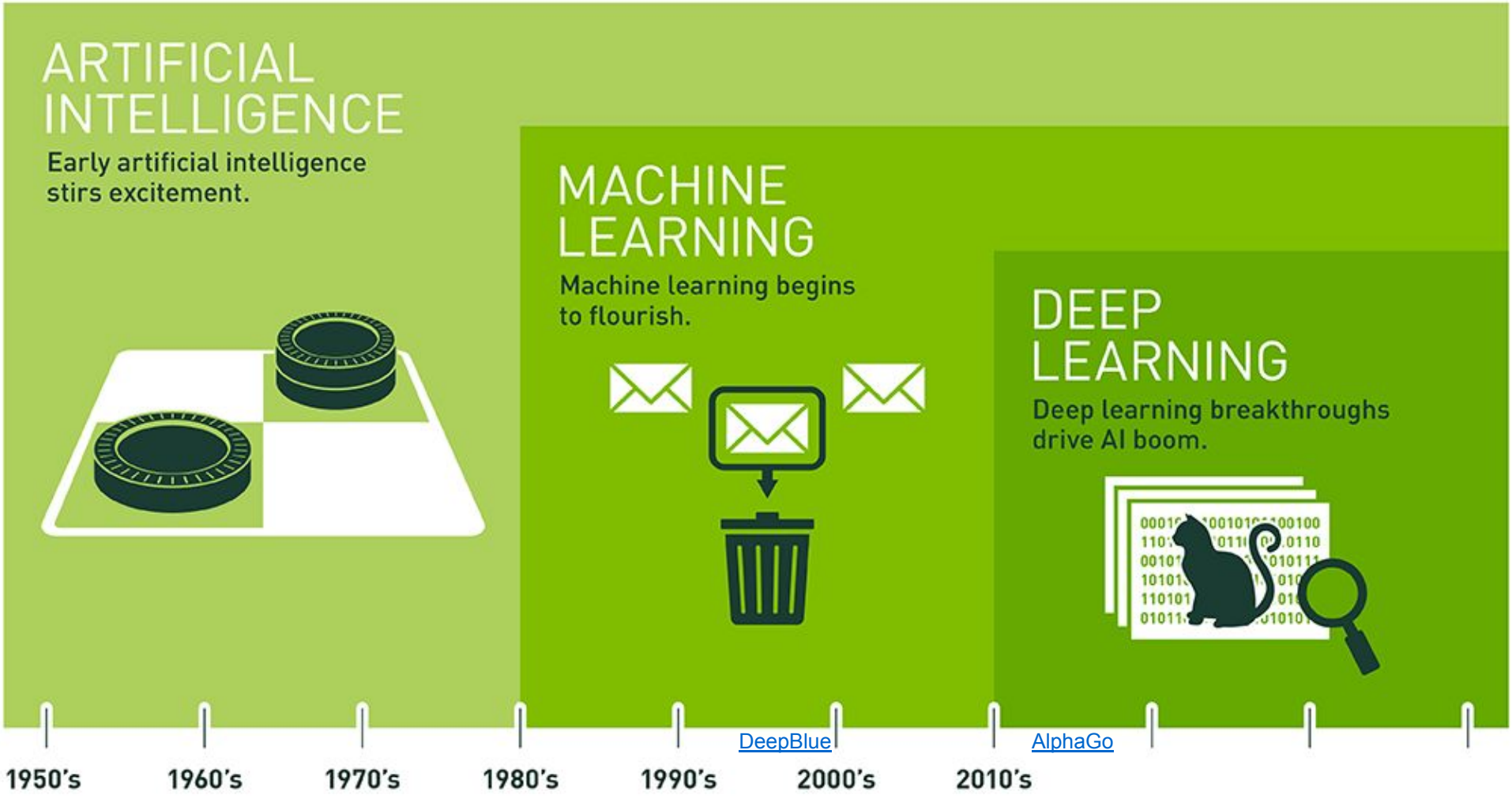
- Upload some images from internet
- Try https://cloud.google.com/vision

DEMO

Try the API



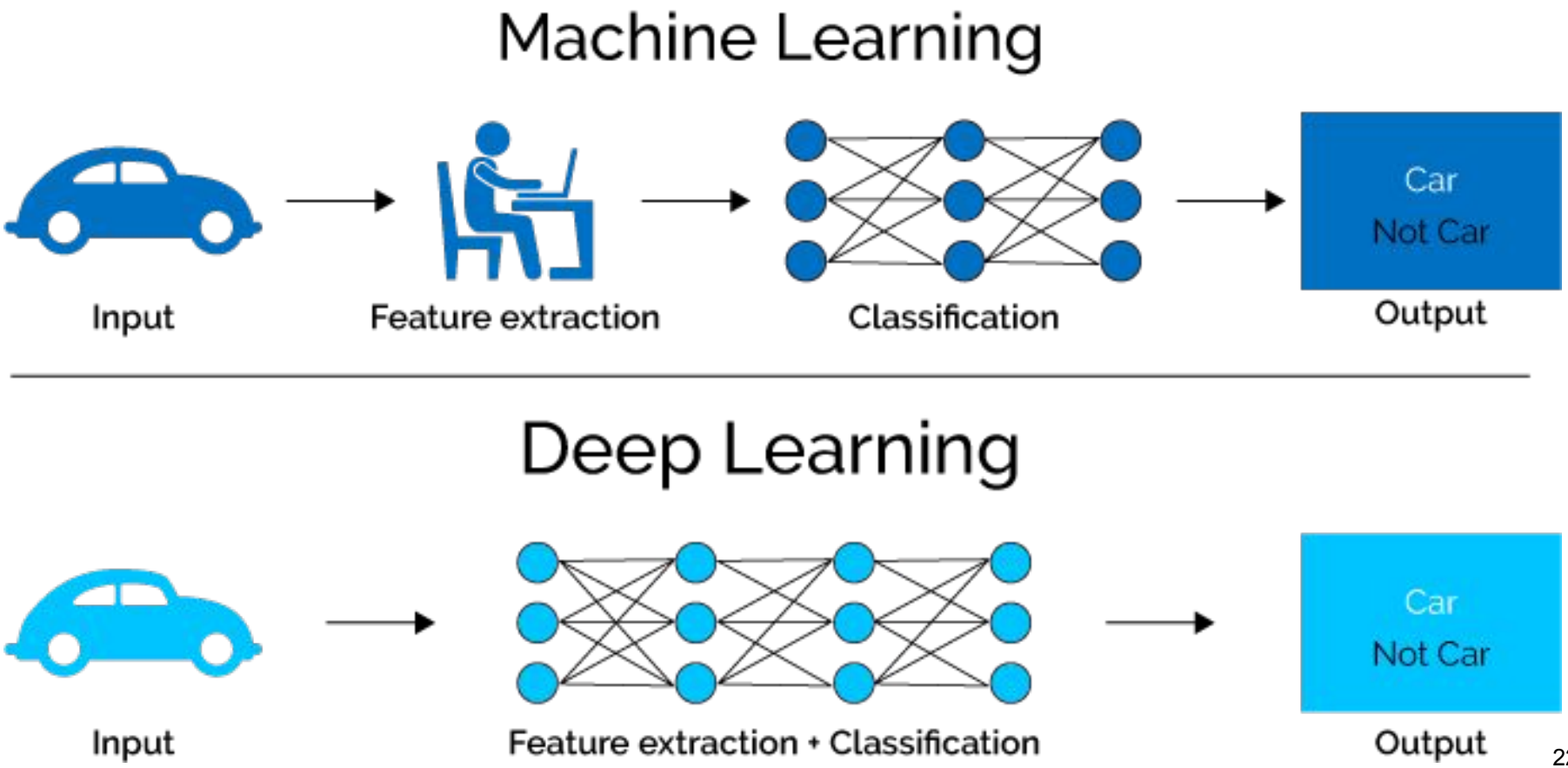
The Evolution of AI



Deep learning is a subset of AI and machine learning that uses multi-layered artificial neural networks to deliver state-of-the-art accuracy in tasks such as object detection, speech recognition, language translation, and others.

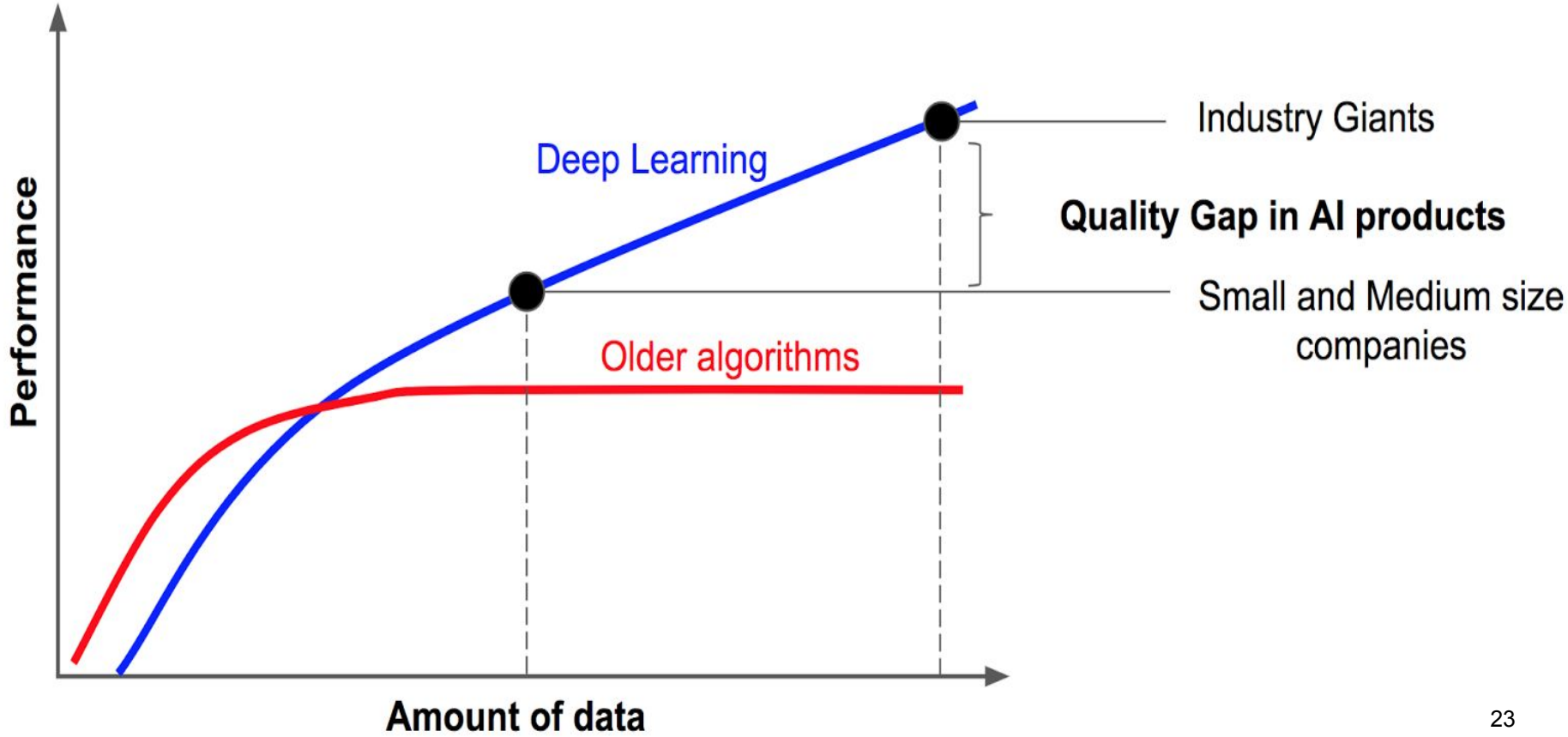
Machine Learning vs Deep Learning

Deep Learning are neural network based and consists of millions of parameters to train.

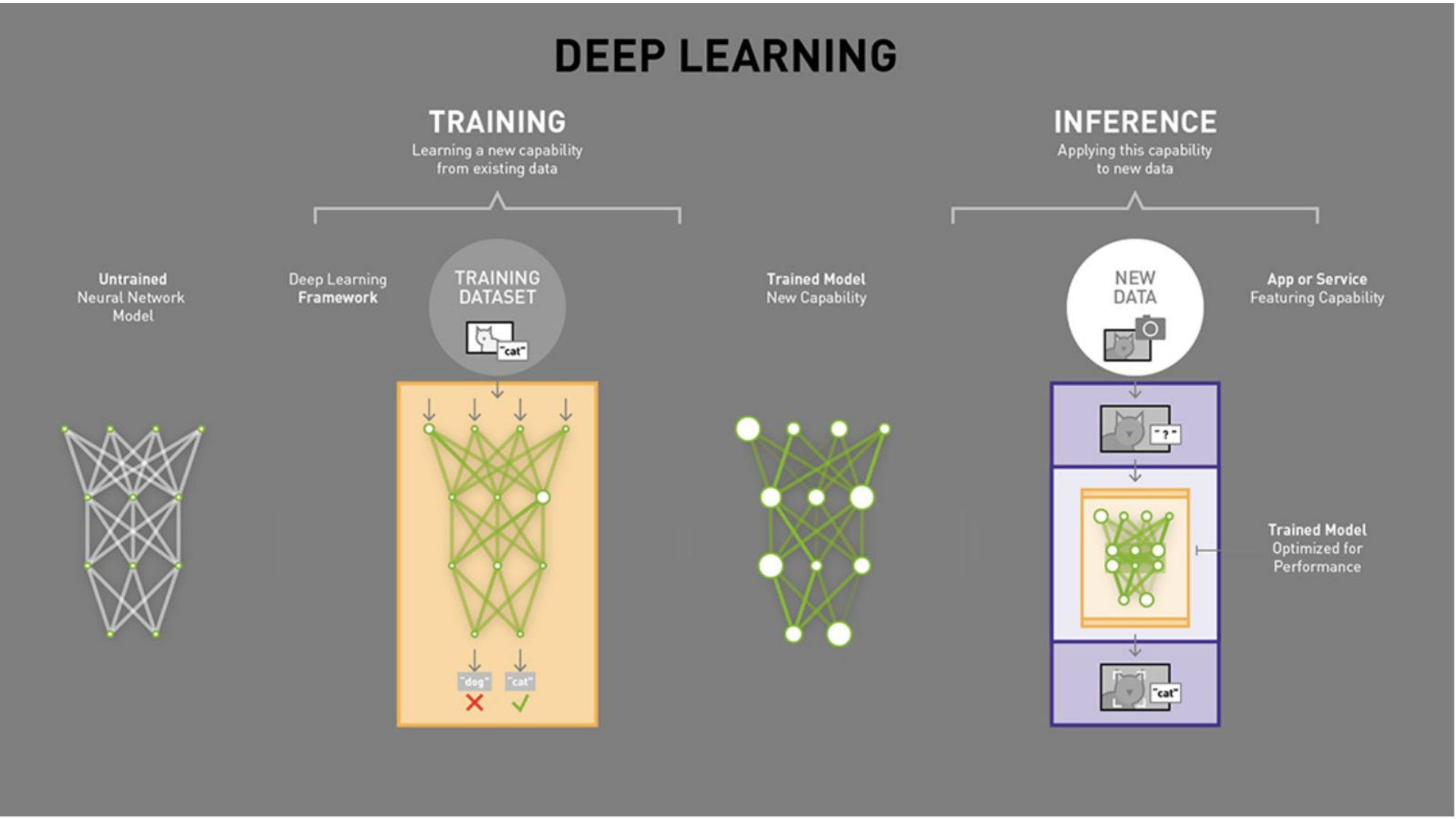


Why Deep Learning?

The performance of Deep Learning surpasses classical machine learning with millions and millions of data.



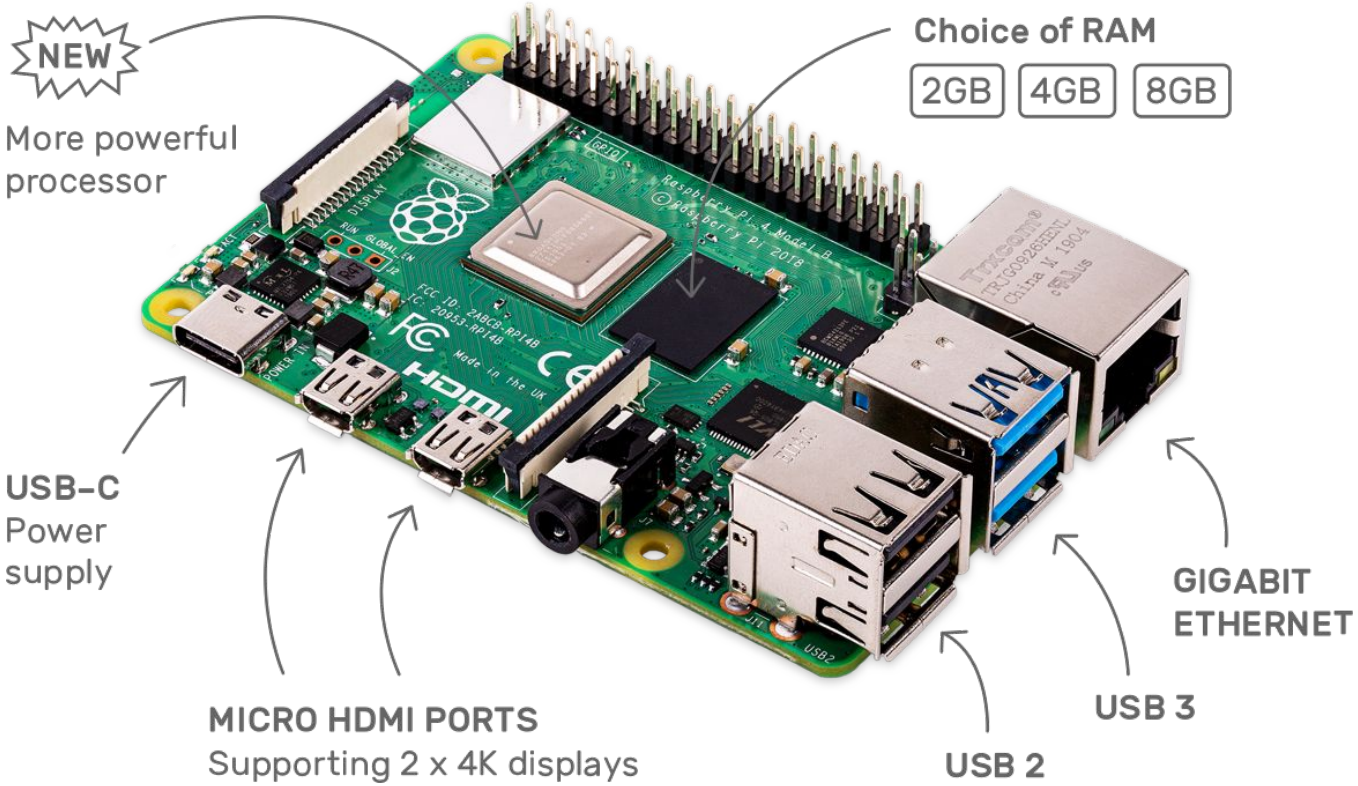
Deep Learning Methodology



Topic 2
Basic Computer Vision Techniques

Overview of Raspberry Pi

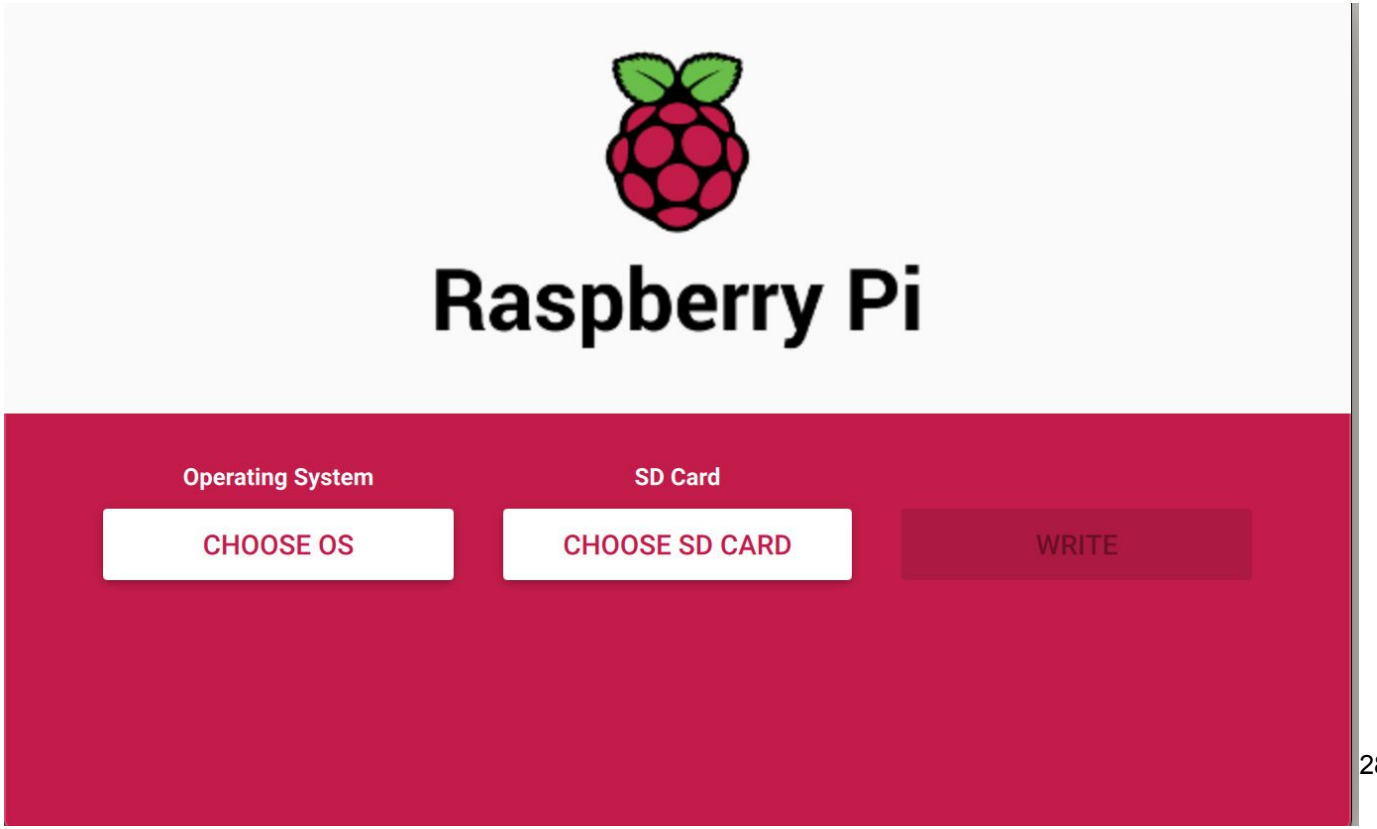
- The Raspberry Pi is a low cost, credit-card sized computer.
- It enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python



Overview of Raspberry Pi

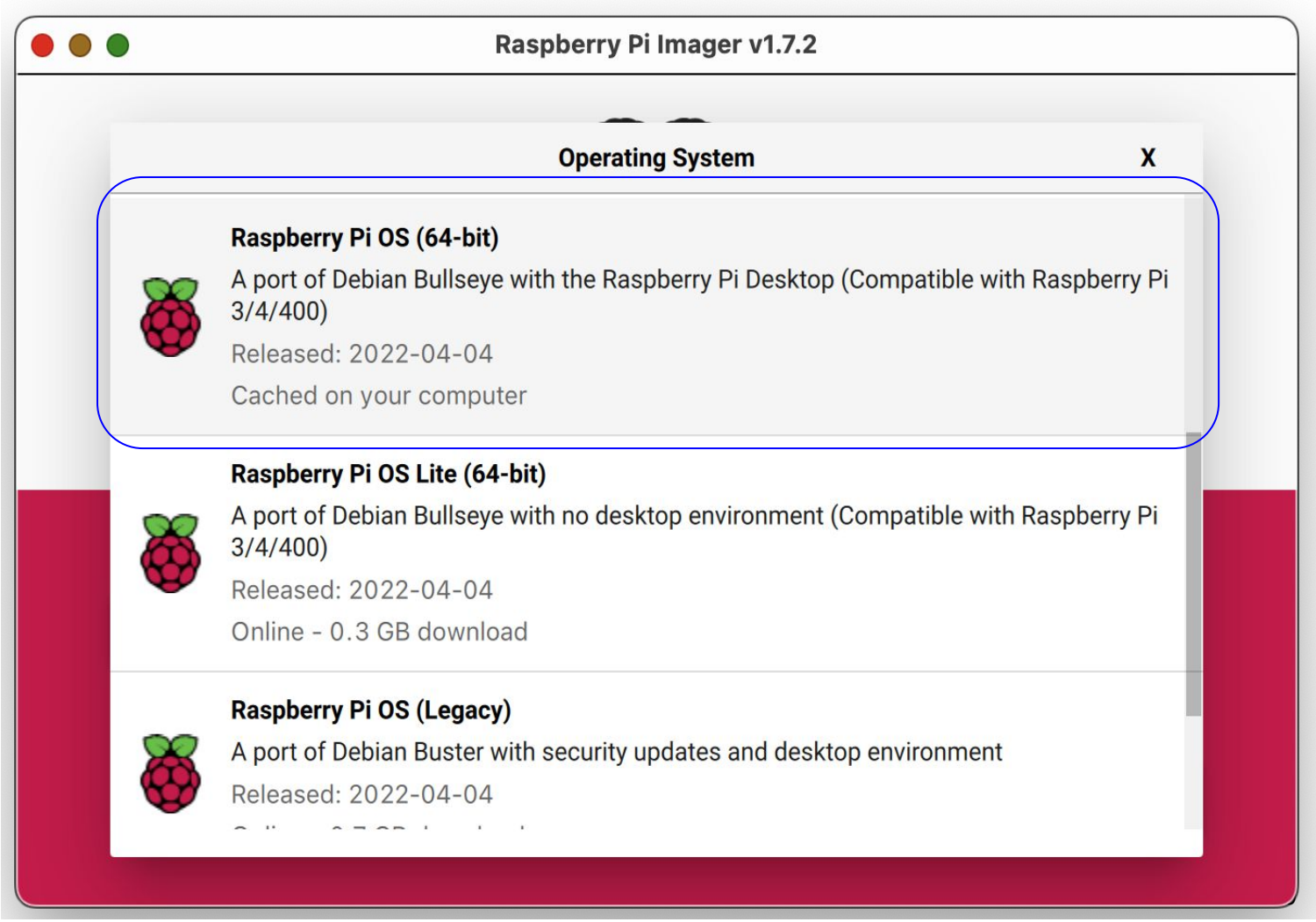
Burn Raspberry Pi Image

- Use Raspberry Pi Imager for an easy way to install Raspberry Pi OS and other operating systems to an SD card ready to use with your Raspberry Pi
- You can get the imager software from https://www.raspberrypi.org/downloads/



Select Raspberry Pi OS

- Select "Raspberry Pi OS (64-bit) ... Desktop"

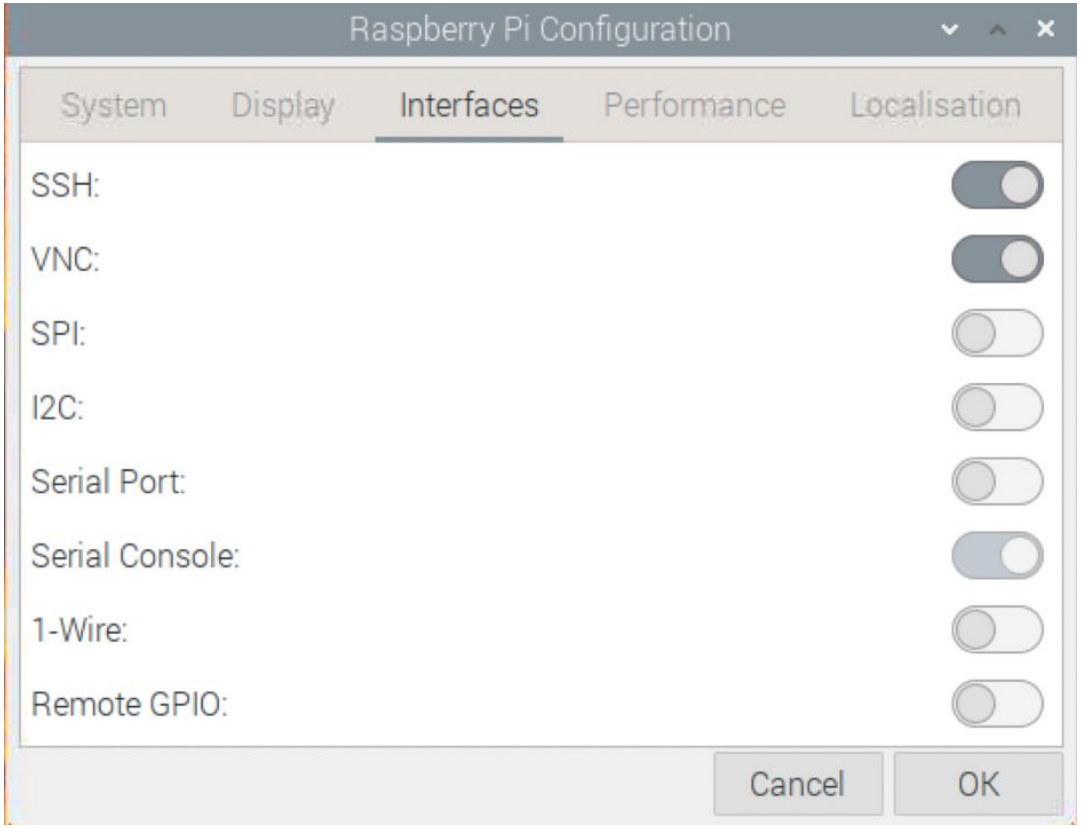


Boot Your New OS

- You can now insert the SD card into the Raspberry Pi and power it up.
- For the official Raspberry Pi OS, the default user name is pi, with password raspberry. You can change the default password straight away to ensure your Raspberry Pi is secure.
- The default keyboard layout is set to UK. Change to US International
- The default timezone is UK. You can change to Singapore.
- Setup your WiFi credential
- Download and the OS update
- Reboot the OS

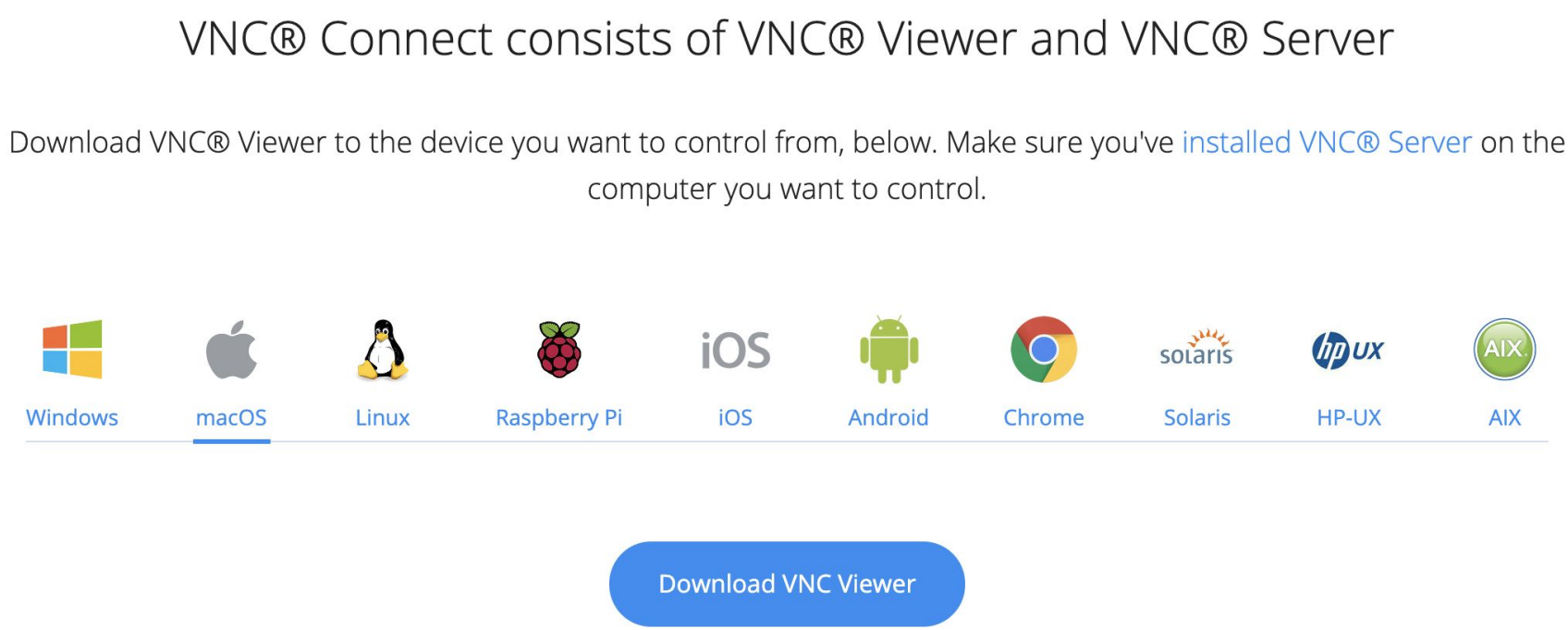
Enable VNC

- The VNC server is already installed in your Raspbian Operating System, you just need to activate it.
- First of all, you need to enable VNC on your Raspberry Pi. Choose Menu-> Preferences-> Raspberry Pi Configuration. Enable SSH & VMC.



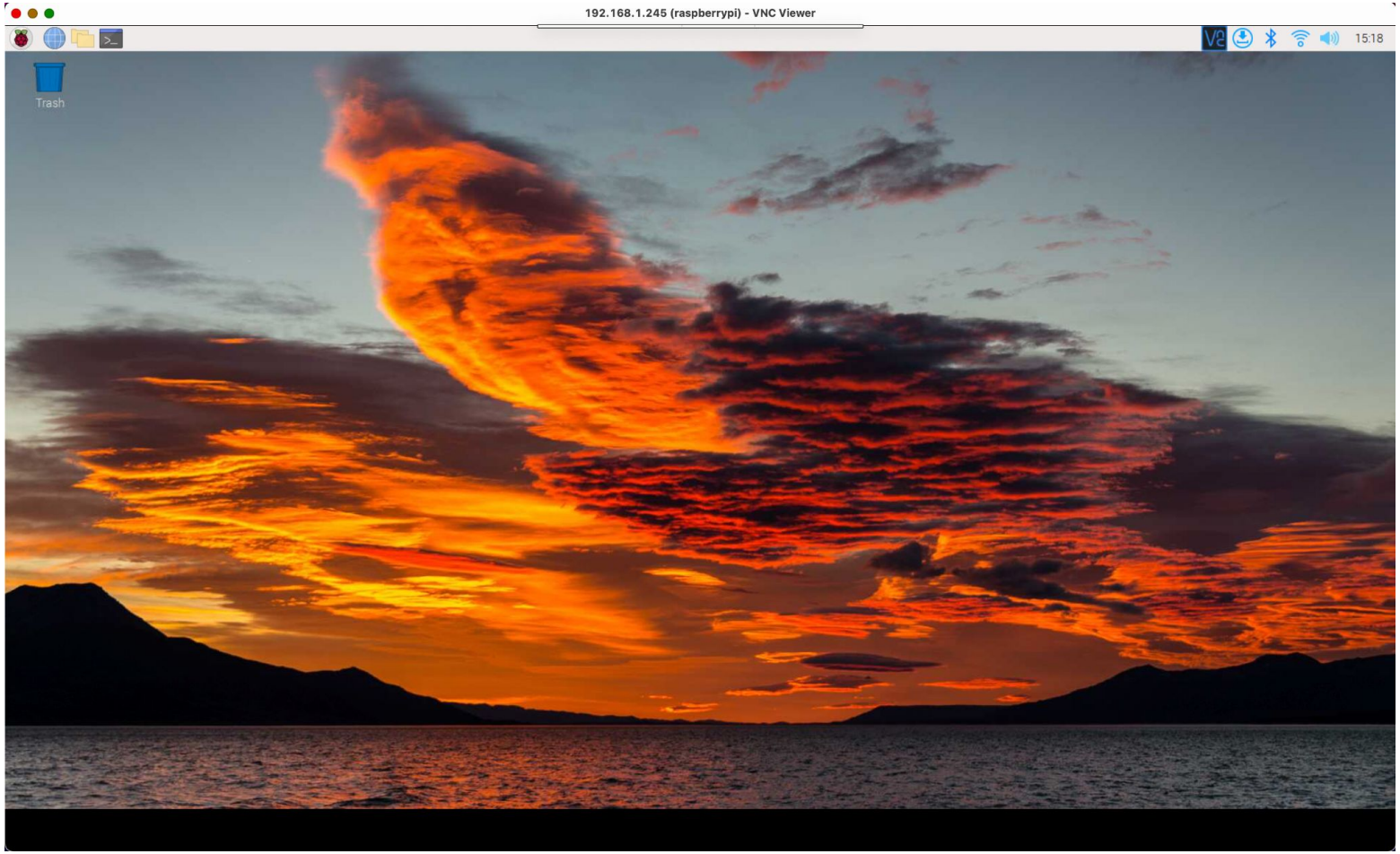
Download VNC Viewer

Download VNC Viewer on your local laptop
https://www.realvnc.com/en/connect/download/viewer/



Remote Access Raspberry Pi via VNC

Enter the IP address to your VNC viewer and you can now remote access your Raspberry Pi from laptop



Setup Camera

You can capture a video from the camera by plugging the USB camera to Raspberry Pi.

OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.

Open a Terminal (Ctrl-Alt-T) and execute the following commands:

```
sudo apt update
sudo apt upgrade -y
pip install opencv-python
```

Capture Video from Camera

```
import numpy as np
import cv2 as cv

cap = cv.VideoCapture(0)
while True:
    ret, frame = cap.read()
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break
    gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
    cv.imshow('frame', gray)
    if cv.waitKey(1) == ord('q'):
        break
cap.release()
cv.destroyAllWindows()
```

Code Explanation

- To capture a video, you need to create a VideoCapture object. Its argument can be either the device index or the name of a video file.

- A device index is just the number to specify which camera.

- cap.read() returns a bool (True/False). If the frame is read correctly, it will be True

- Press the 'q' key to quit

Saving Video

```
import numpy as np
import cv2 as cv
cap = cv.VideoCapture(0)
fourcc = cv.VideoWriter_fourcc(*'XVID')
out = cv.VideoWriter('output.avi', fourcc, 20.0, (640,  480))

while cap.isOpened():
    ret, frame = cap.read()
    frame = cv.flip(frame, 0)

    out.write(frame)
    cv.imshow('frame', frame)
    if cv.waitKey(1) == ord('q'):
        break

cap.release()
out.release()
cv.destroyAllWindows()
```

Code Explanation

- This time we create a VideoWriter object.
- We should specify the output file name (eg: output.avi).
- Then we should specify the FourCC code. FourCC is a 4-byte code used to specify the video codec. T
- Then number of frames per second (fps) and frame size should be passed.
- And the last one is the isColor flag. If it is True, the encoder expect color frame, otherwise it works with grayscale fram

Activity: Video Playback

Play back the video output.avi using OpenCV

Topic 3
Image Classification

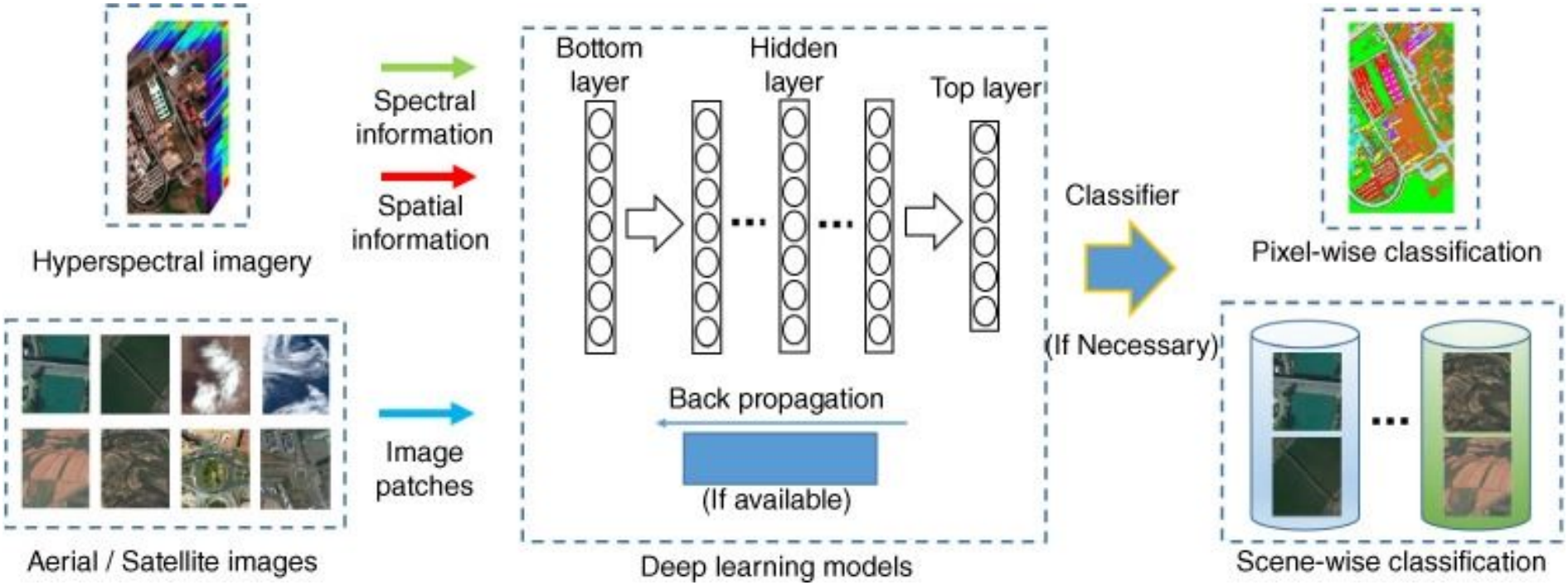
Image Classification

Image recognition output class probabilities representing the entire input image.

Image Recognition
Classification

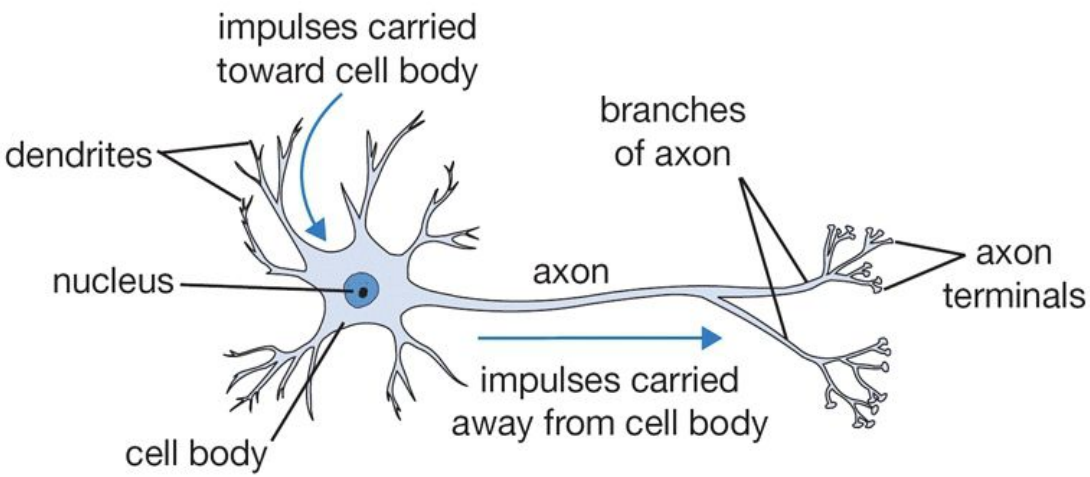
DNN Image Classification

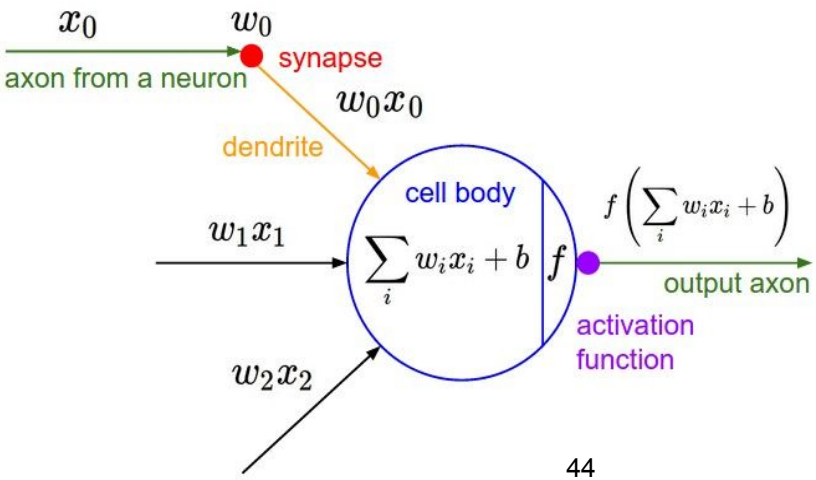
Deep Neural Networks (DNN) are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns



What is a Neuron?

- A biological neuron has dendrites to receive signals, a cell body to process them, and an axon to send signals out to other neurons
- An artificial neuron has a number of input channels, a processing stage, and one output that can fan out to multiple other artificial neurons.
- The weighted sum of inputs + bias is feed through a non-linear activation function to output a value to next neurons.



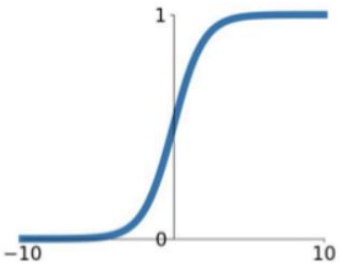


Activation Functions

Activation functions allow Neural Networks to learn complicated and Non-linear complex functional mappings between the inputs and outputs.

Sigmoid

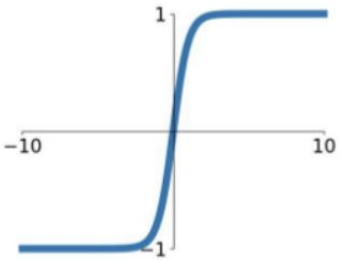
$\sigma(x) = \frac{1}{1+e^{-x}}$



$$\sigma(z) = \frac{1}{1+e^{-z}}$$

tanh

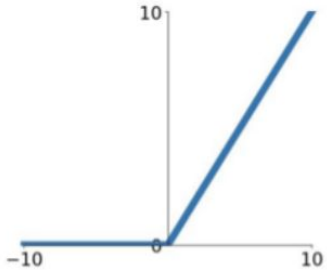
$\tanh(x)$



$$\sigma(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

ReLU

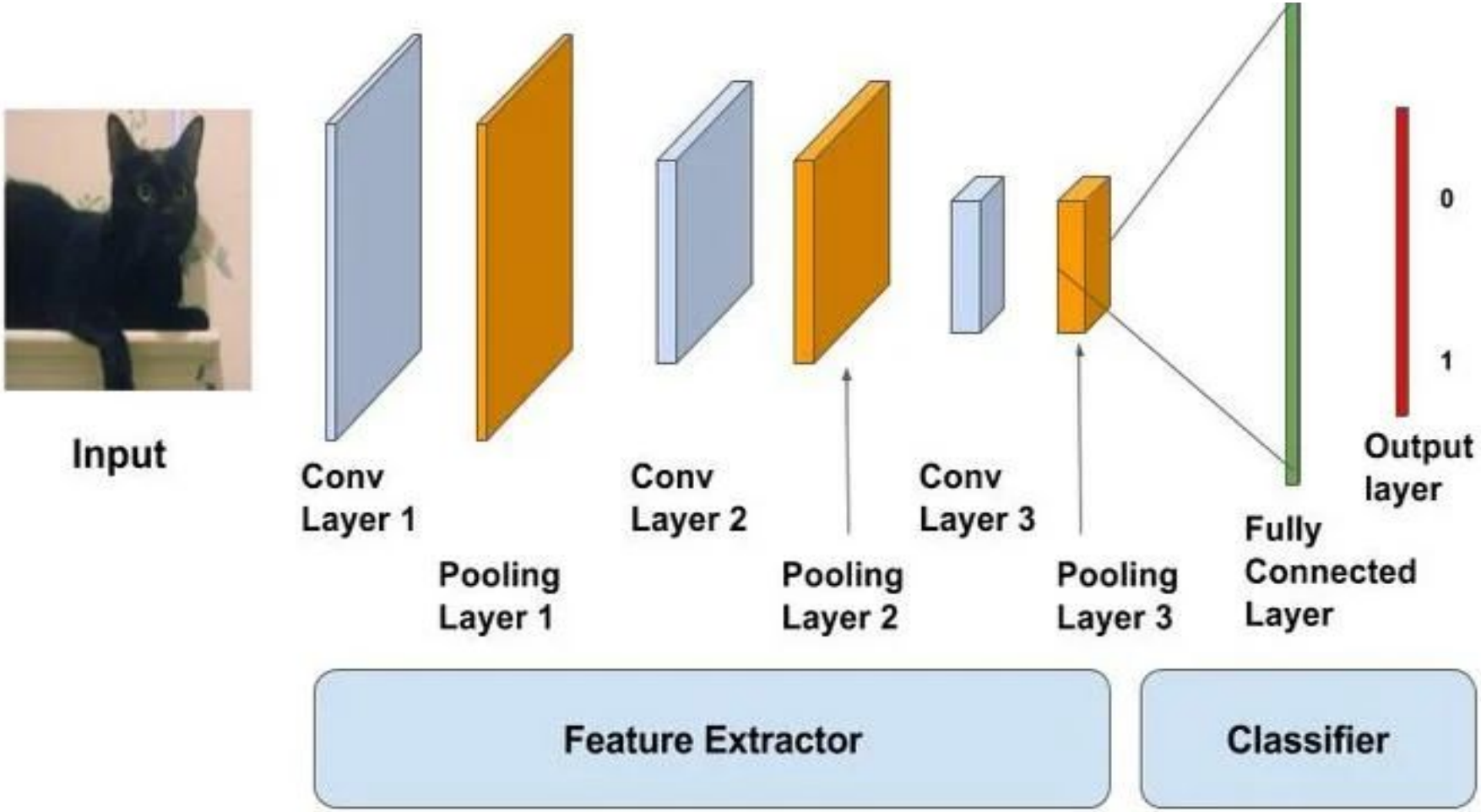
$\max(0, x)$



$$\sigma(z) = max(0, z)$$

Convolutional Neural Network (CNN)

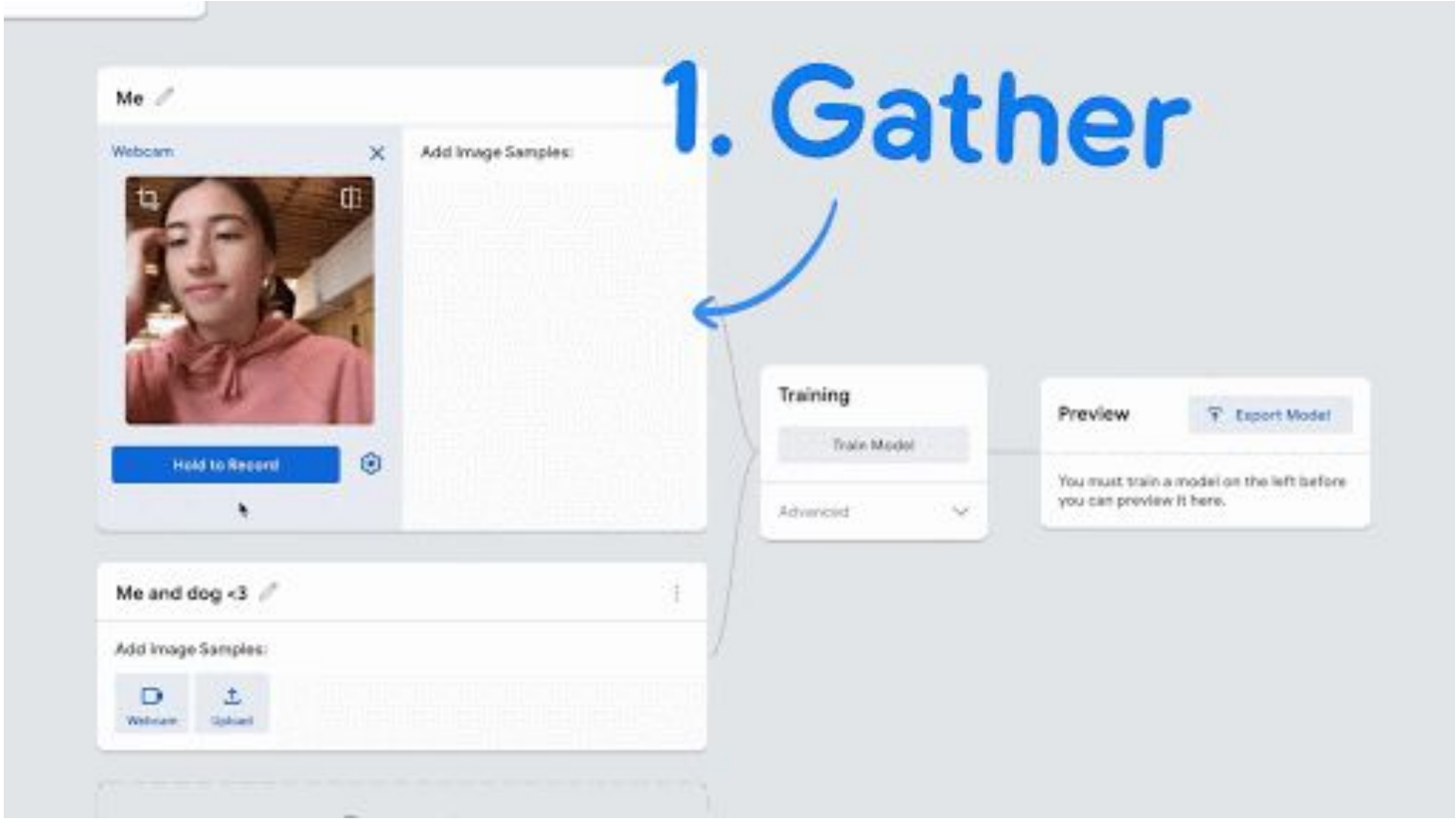
The key technology behind image classification is convolutional neural network (CNN)



Teachable Machine

Teachable Machine is a web-based tool that makes creating machine learning models fast, easy, and accessible to everyone.

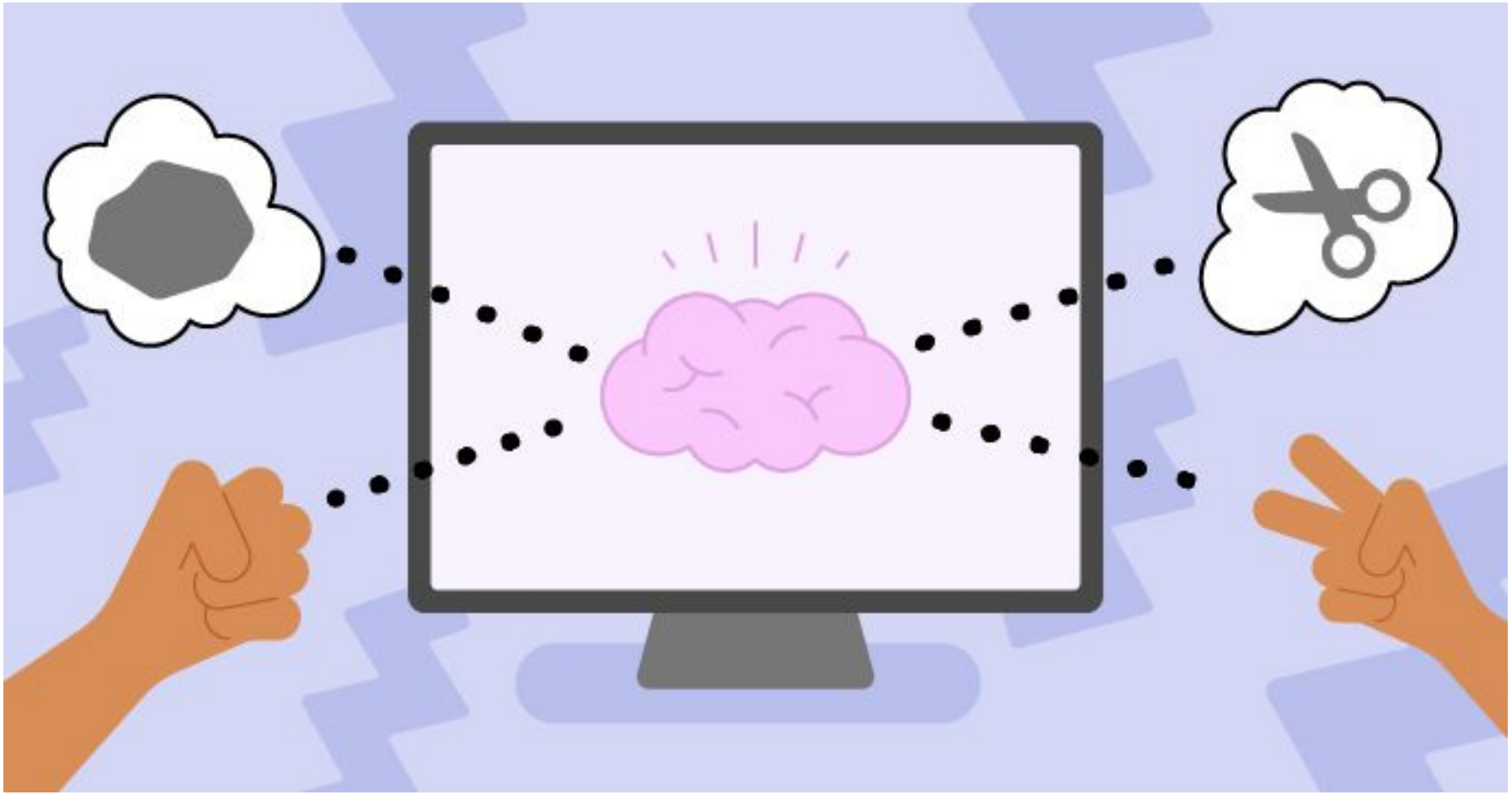
https://teachablemachine.withgoogle.com/train/image



Activity : Teachable Machine

Create an image classification model for scissor-paper stone computer vision app on Teachable Machine

https://teachablemachine.withgoogle.com/train/image



Install TFLite Raspberry Pi Examples

# Update packages on your Raspberry Pi OS.
sudo apt-get update

Check your Python version. You should have Python 3.7 or later.
python -V

upgrade pip
python -m pip install --user --upgrade pip

Clone the TensorFlow example repository with the TFLite Raspberry Pi samples.
git clone https://github.com/tensorflow/examples.git

Setup Classification Examples

```
# Change directory to image_classification for rpi
cd examples/lite/examples/image_classification/raspberry_pi/

# Add the following line to requirements.txt
protobuf==3.20.1

# Install dependencies required by the sample
sh setup.sh

# Install some additional libraries
sudo apt-get install libportaudio2

# Run the classification sample
python classify.py
```

Image Classification Demo

- If you see an error running the sample:
- ImportError: libcblas.so.3: cannot open shared object file: No such file or directory
- you can fix it by installing an OpenCV dependency that is missing on your Raspberry Pi.
- sudo apt-get install libatlas-base-dev

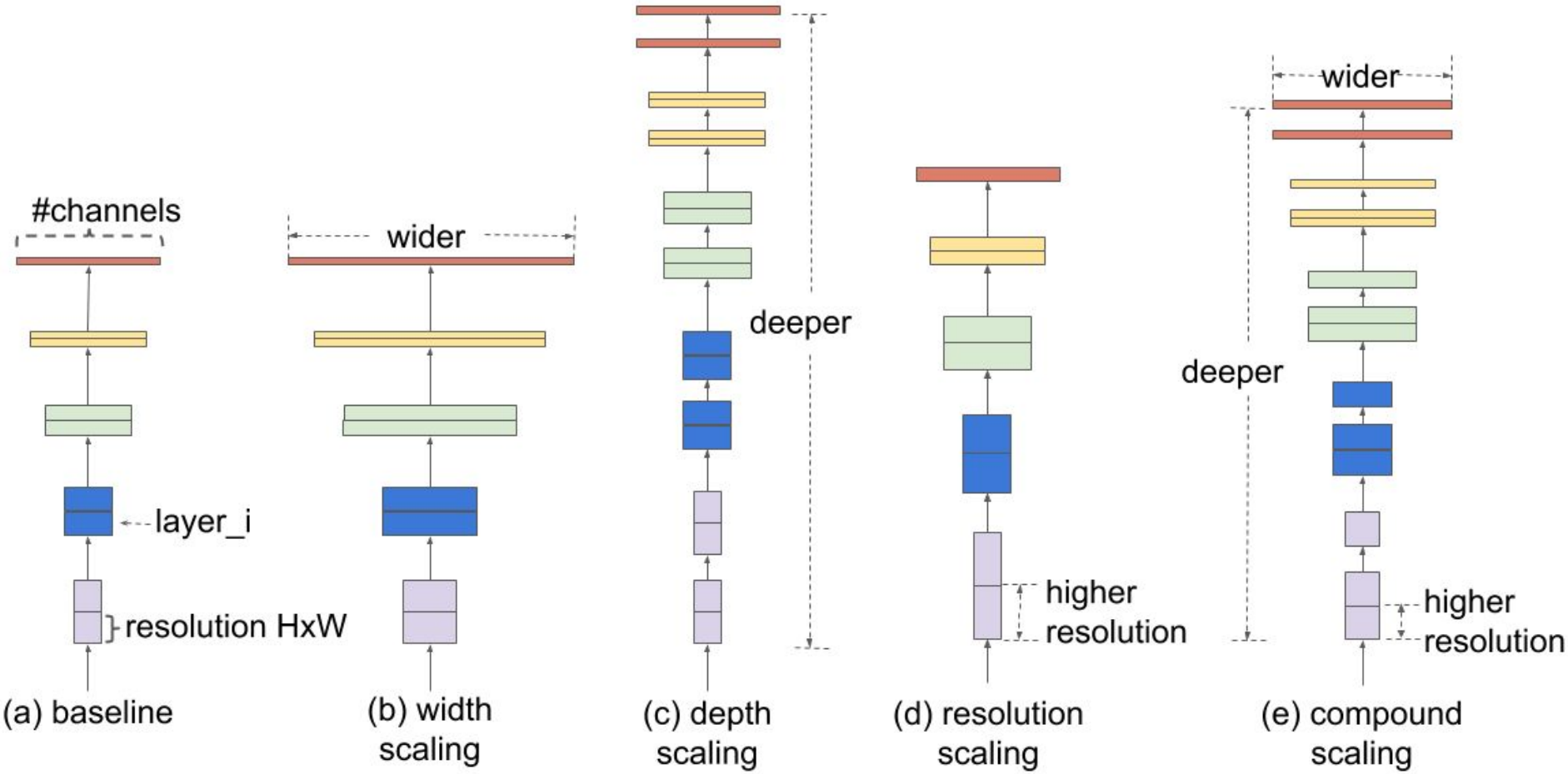
classify.py

The default classify.py uses EfficientNet Tensorflow Lite model for the classification.

```
parser.add_argument(
    '--model',
    help='Name of image classification model.',
    required=False,
    default='efficientnet_lite0.tflite')
```

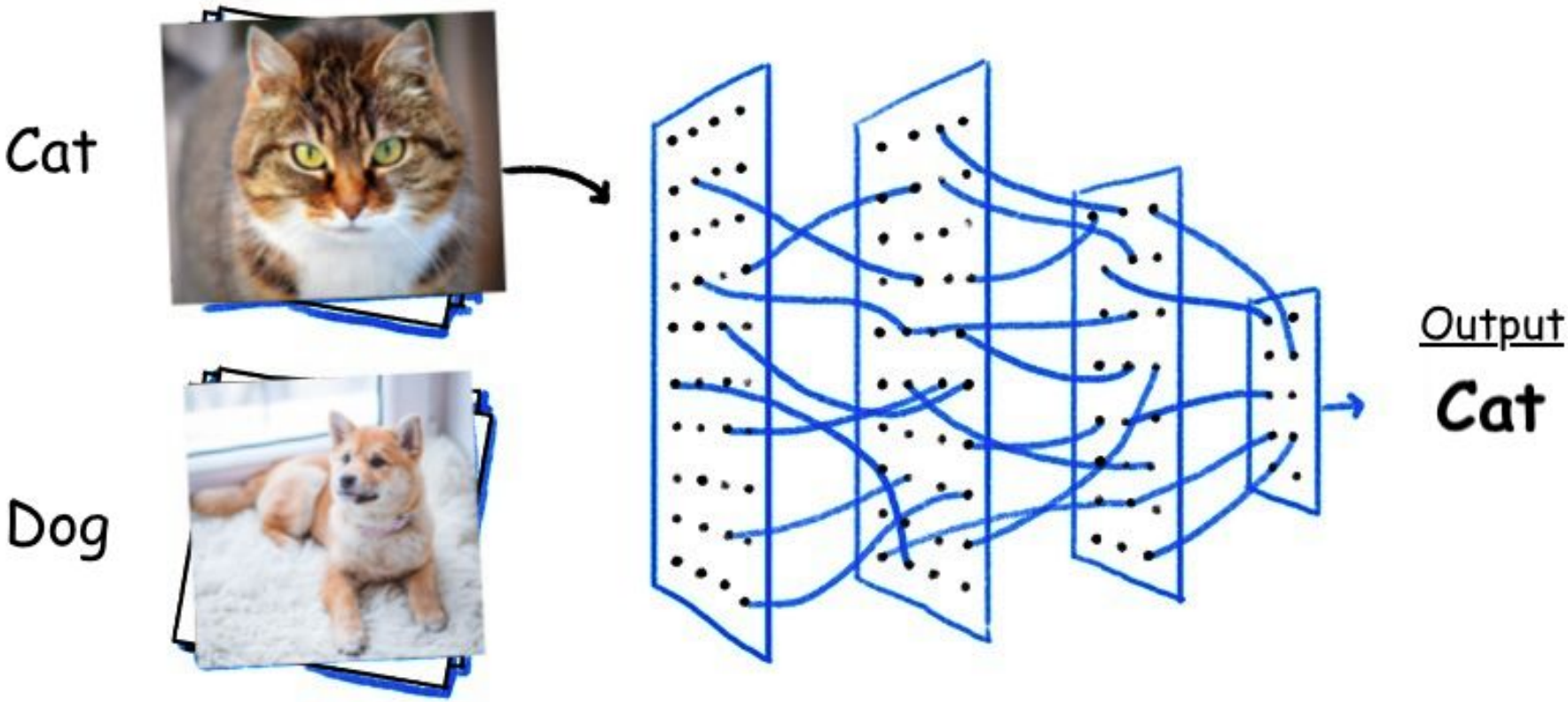
EfficientNET Architecture

EfficientNet is a convolutional neural network architecture and scaling method that uniformly scales all dimensions of depth/width/resolution using a compound coefficient.



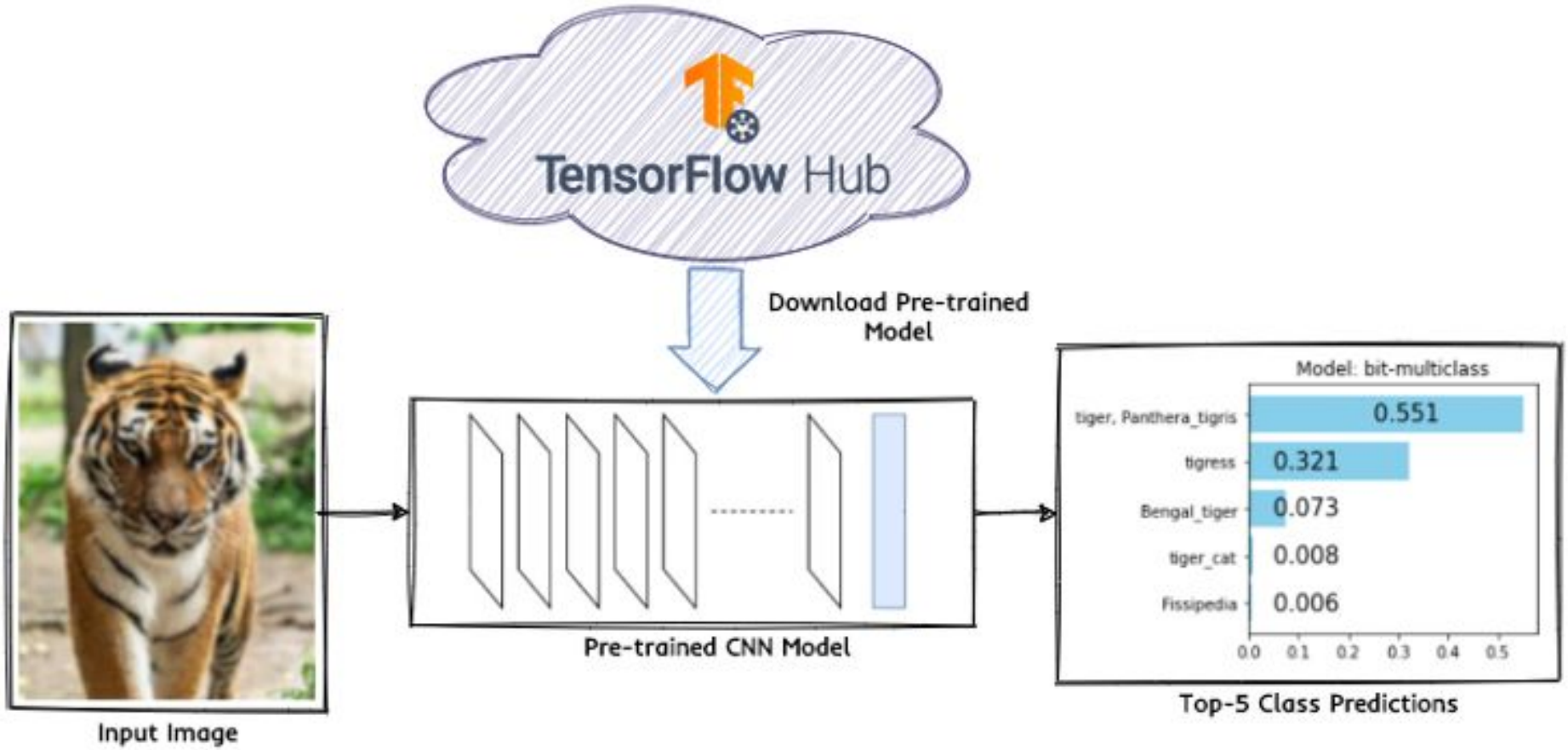
Activity: Image Classification

Try using classify.py to classify your own objects.



Tensorflow Hub

You can get more pre-trained image classification model from TensorFlow Hub (https://www.tensorflow.org/hub)



Get TFlite Models for Raspberry Pi

https://tfhub.dev/s?deployment-format=lite

TensorFlow Hub

Search for models, collections & publishers

Send feedback

Filters

Clear all

Problem domain

Model format

TF.js | TFLite | Coral

TF Version

TF1 | TF2

Fine tunable

Architecture

Publisher

Dataset

Language

Image classification — TFLite

disease-classification

Publisher: AgriPredict 2.0k

Fine-grained image classification model for crop disease classification.

Architecture: MobileNet | Dataset: Internal AgriPredict Dataset

Image super resolution — TFLite

esrgan-tf2

Publisher: Adrish Dey (@captain-pool). 2.7k

Enhanced Super Resolution GAN (Wang et. al.)[1] for image super resolution. Produces x4 Super Resolution Image from images of {Height, Width} >=64. Works best on Bicubically downsampled images.\ (*This is because, the model is originally trained on Bicubically…

Architecture: GAN | Dataset: DIV2K

Audio event classification — TFLite | .JS

yamnet

Publisher: Google 52.9k

An audio event classifier trained on the AudioSet dataset to predict audio events from the AudioSet ontology.

Architecture: MobileNet V1 | Dataset: AudioSet

Image segmentation — TFLite

edgetpu/vision/autoseg-edgetpu/default_argmax/xs

Publisher: Google 73

Autoseg-EdgeTPU is a set of semantic segmentation models that are optimized for Pixel 6 leveraging neural architecture search algorithms.

Dataset: ADE20K

Image classification — TFLite

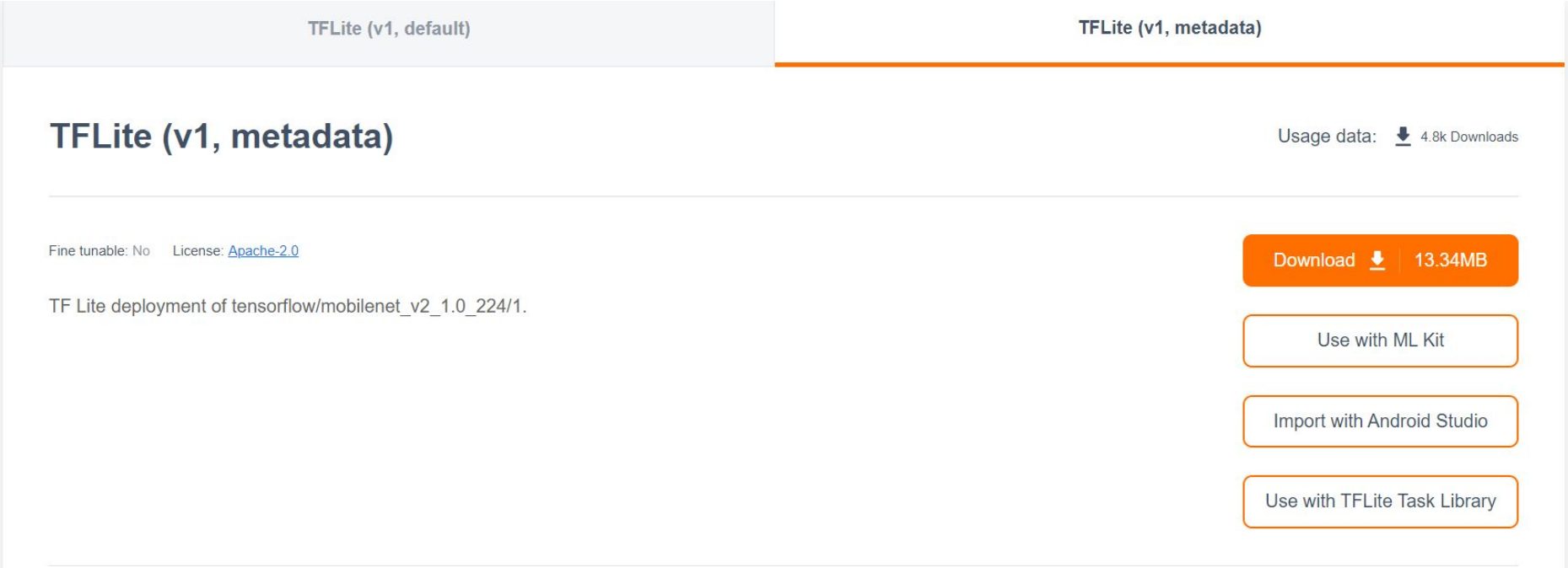
on_device_vision/classifier/landmarks_classifier_europ

Image classification — TFLite

object_detection/mobile_object_labeler_v1

MobileNet TFLite Model

Download MobileNet tflite file
https://tfhub.dev/tensorflow/lite-model/mobilenet_v2_1.0_224/1/metadata/1



Run MobileNet TFLite Model

Copy file to image classification folder

```
cp ~/Downloads/mobilenet_v2_1.0_224_1_metadata_1.tflite
~/examples/lite/examples/image_classification/raspberry_pi
```

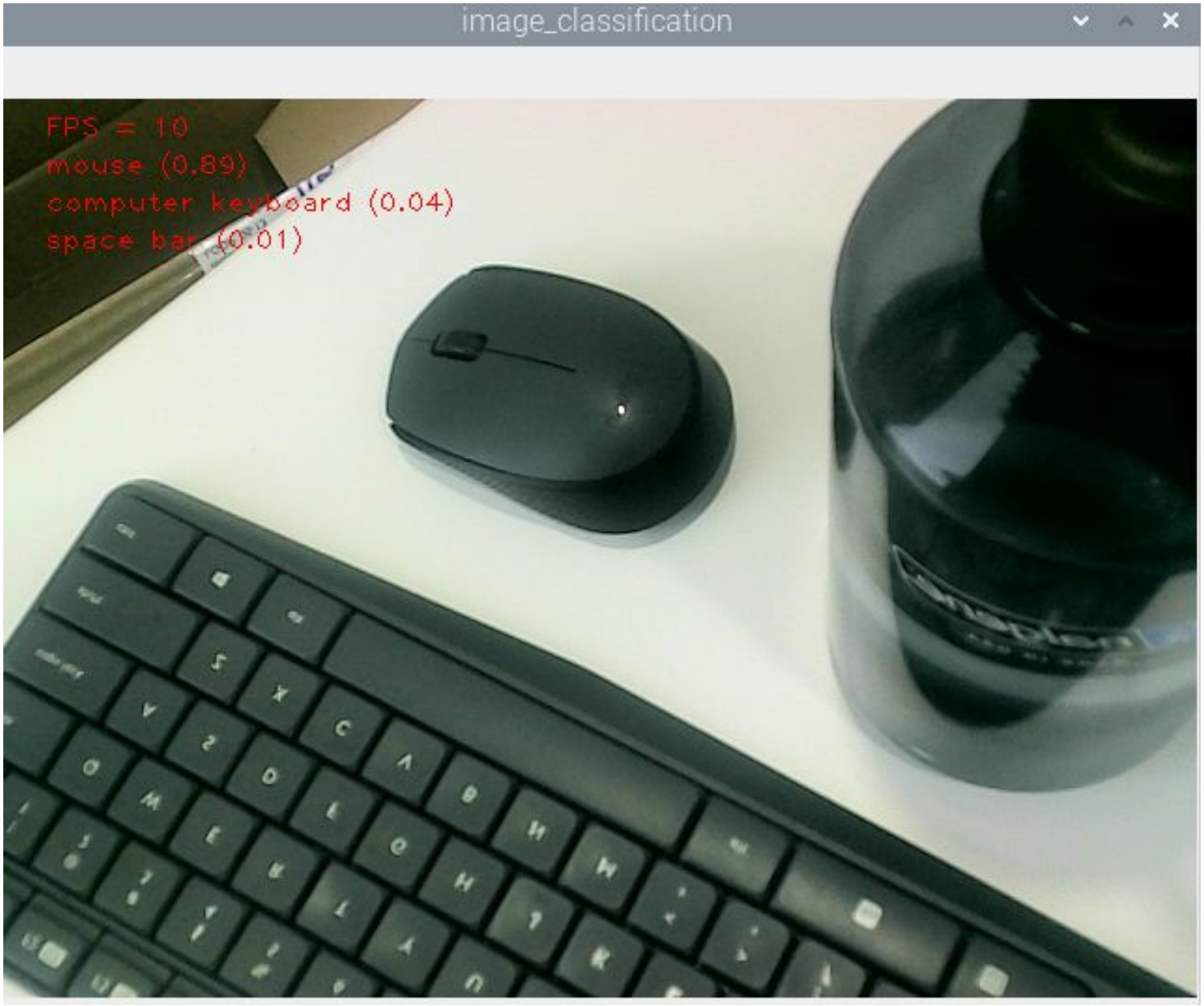
Change directory

```
cd examples/lite/examples/image_classification/raspberry_pi
```

#Running mobilenetv2 model

```
python classify.py --model
mobilenet_v2_1.0_224_1_metadata_1.tflite
```

MobileNet TFLite Model Result



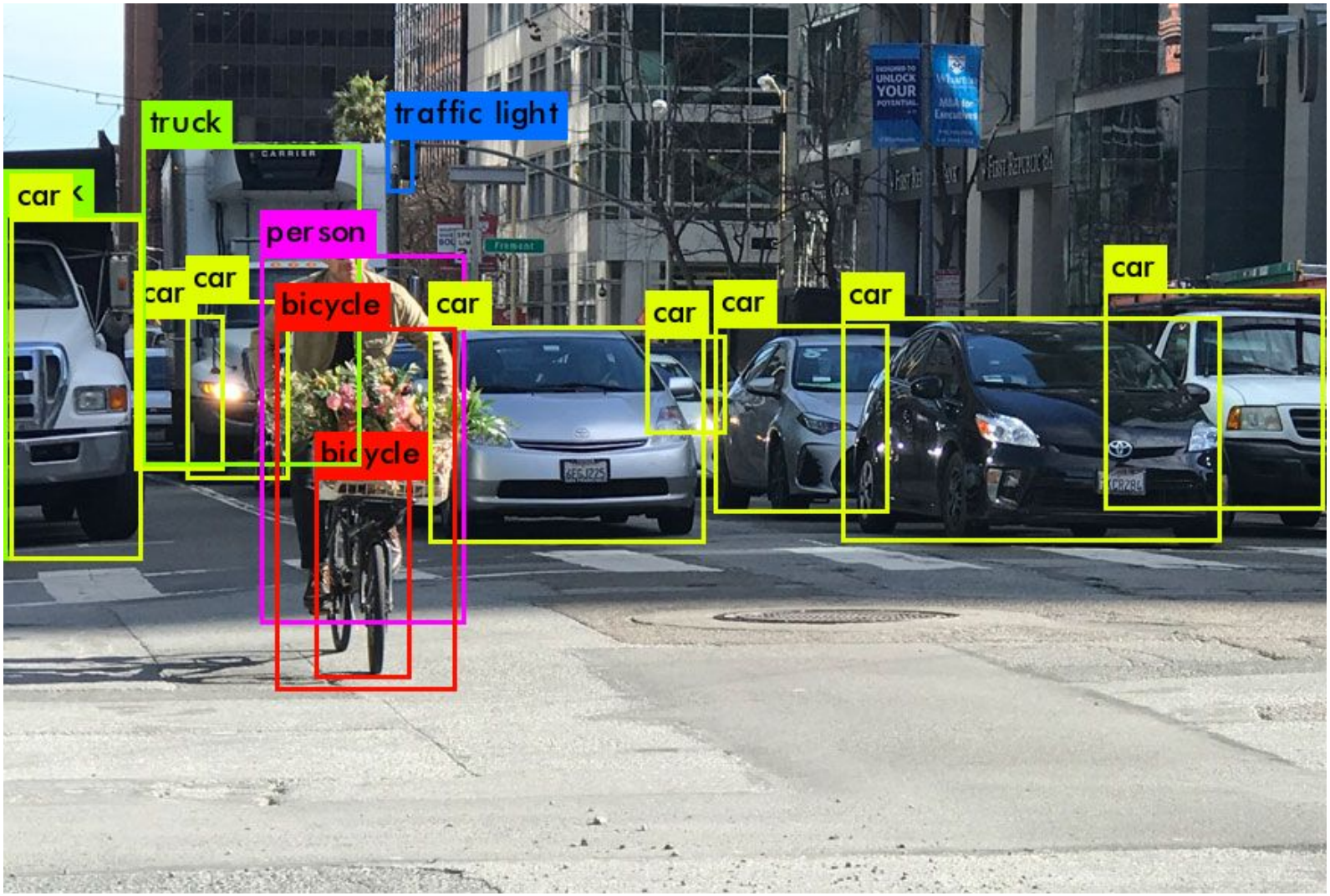
Activity: Image Classification

Run the mobilenet_v1_1.0_224_quant.tflite on Raspberry Pi

Topic 4
Object Detection

Object Detection

Object detection is able to find various objects located in a frame by extracting their bounding boxes.

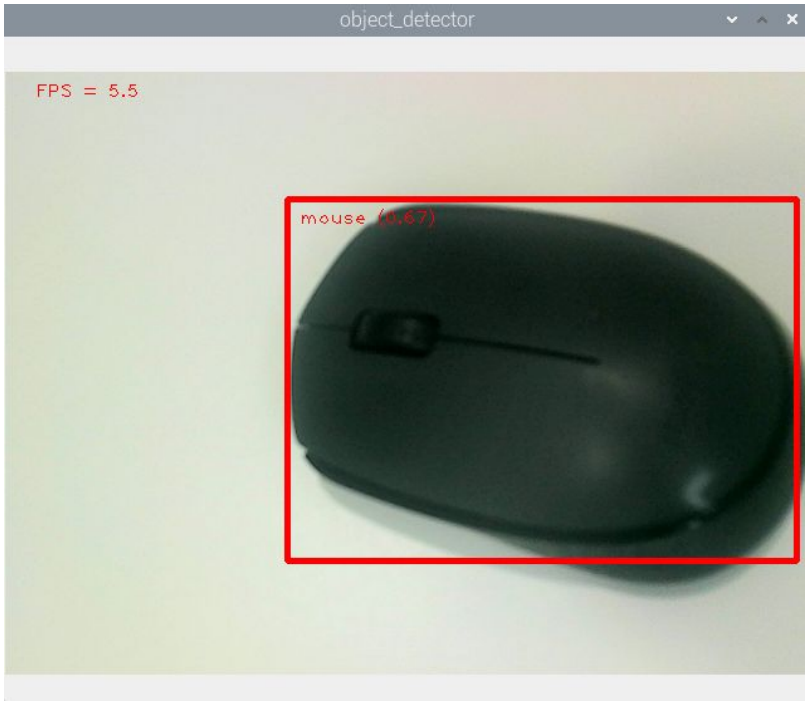


Object Detection Examples

Change directory to object_detection for rpi
cd examples/lite/examples/object_detection/raspberry_pi

Install dependencies required by the sample
sh setup.sh

Run the object detection sample
python detect.py



detect.py

The default detect.py also uses EfficientNet Tensorflow Lite model for the object detection.

```
parser.add_argument(
    '--model',
    help='Path of the object detection model.',
    required=False,
    default='efficientdet_lite0.tflite')
```

Activity: Object Detection

Try the object detection on your own objects

Object Detection TFLite Models

https://tfhub.dev/s?deployment-format=lite&module-type=image-object-detection

Image object detection TFLite

ssd_mobilenet_v1_100_320/uint8

Publisher: IREE 46

Mobilenet V1 with SSDLite head trained on COCO 2017, in `uint8`.

Architecture: SSD Mobilenet V1 | Dataset: COCO 2017

Image object detection TFLite

ssd_mobilenet_v2_100/fp32

Publisher: IREE 42

Mobilenet V2 with SSDLite head trained on COCO 2017, in `fp32`.

Architecture: SSD Mobilenet V1 | Dataset: COCO 2017

Image object detection TFLite

ssd_mobilenet_v2_100/int8

Publisher: IREE 62

Mobilenet V2 with SSDLite head trained on COCO 2017, in `int8`.

Architecture: SSD Mobilenet V1 | Dataset: COCO 2017

Image object detection TFLite

ssd_mobilenet_v2_fpn_100/fp32

Publisher: IREE 19

Mobilenet V2 with FPNLite head trained on COCO 2017, in `fp32`.

Architecture: SSD Mobilenet V2 FPN | Dataset: COCO 2017

Image object detection TFLite

ssd_mobilenet_v2_fpn_100/uint8

Publisher: IREE 279

Mobilenet V2 with FPNLite head trained on COCO 2017, in `uint8`.

Architecture: SSD Mobilenet V2 FPN | Dataset: COCO 2017

Image object detection TFLite

ssd_spaghettinet_edgetpu_large_320/fp32

Publisher: IREE 8

Spaghettinet EdgeTPU Large trained on COCO 2017, in `fp32`.

Architecture: SSD Spaghettinet EdgeTPU Large | Dataset: COCO 2017

Image object detection TFLite

yolo-v5-tflite

Publisher: Neha Soni 2.5k

[YOLOv5](https://docs.ultralytics.com/) is a family of object detection architectures and models pretrained on the [COCO dataset](https://cocodataset.org/#home).

Architecture: YOLOv5 | Dataset: COCO 2017

Image object detection TFLite

yolo-cppe5-lite

Publisher: Rishit Dagli.

The YOLO model (with DarkNet 53 backbone) trained on the CPPE - 5 (Medical Personal Protective Equipment) dataset in TF Lite Format.

Architecture: YOLO | Dataset: CPPE 5

Image object detection TFLite

faster-rcnn-cppe5-lite

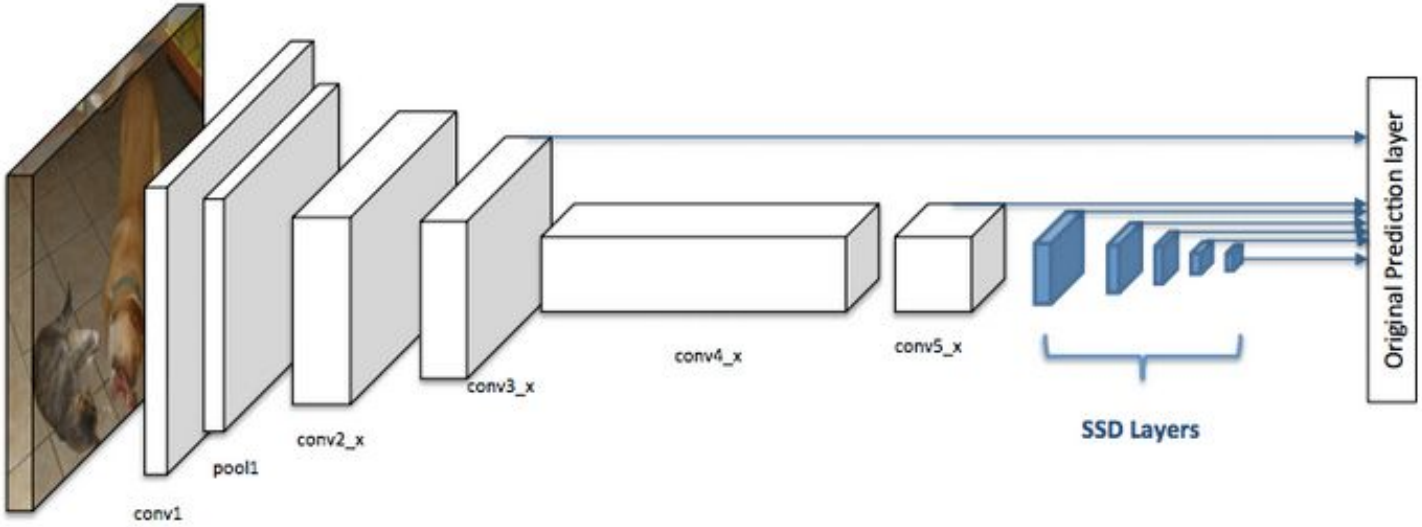
Publisher: Rishit Dagli.

The Faster RCNN model (with ResNet 101 backbone) trained on the CPPE - 5 (Medical Personal Protective Equipment) dataset in TF Lite Format.

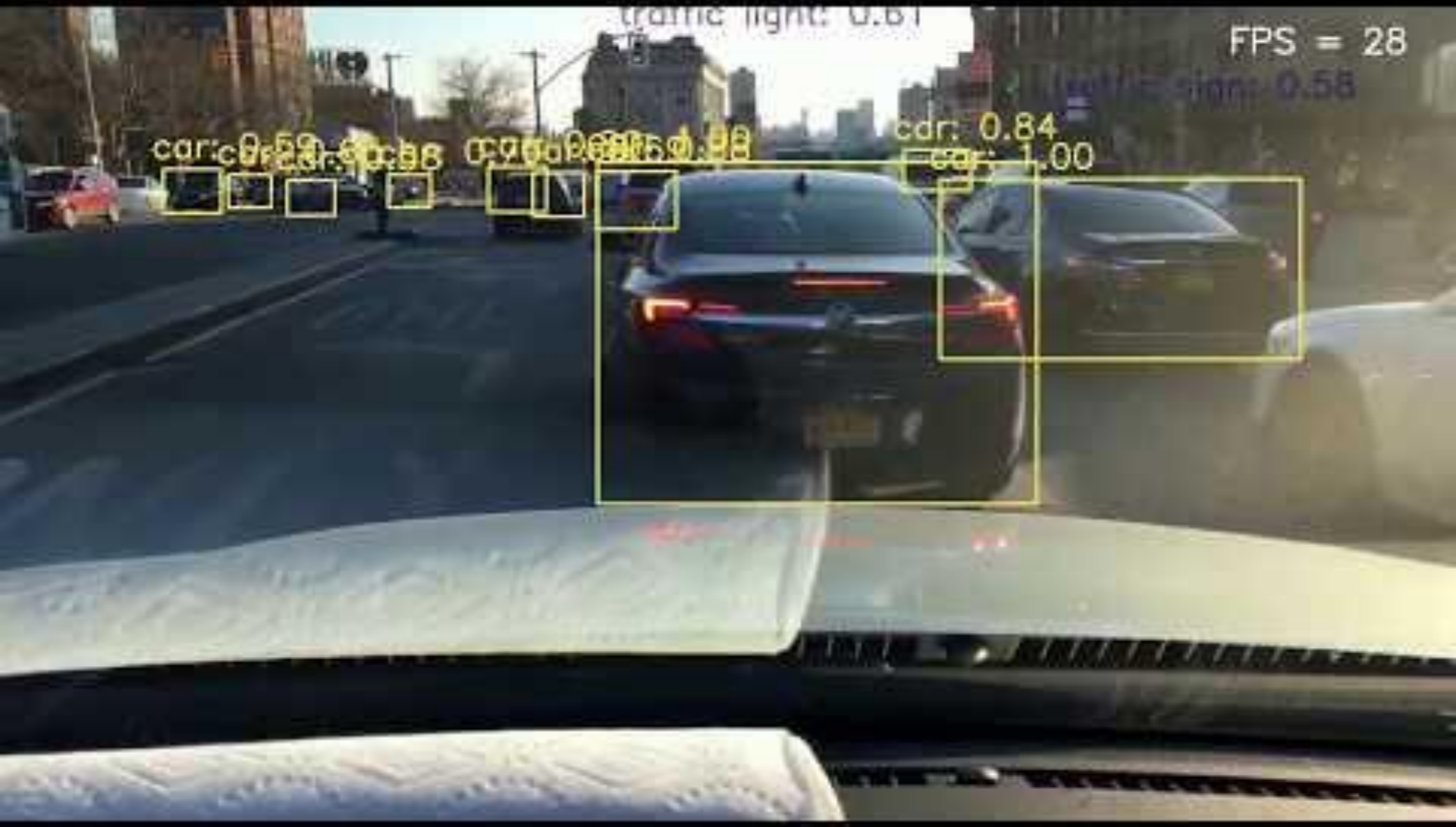
Architecture: Faster R-CNN | Dataset: CPPE 5

SSD Model

- SSD has two components: a backbone model and SSD head.
- Backbone model usually is a pre-trained image classification network as a feature extractor. This is typically a network like ResNet trained on ImageNet

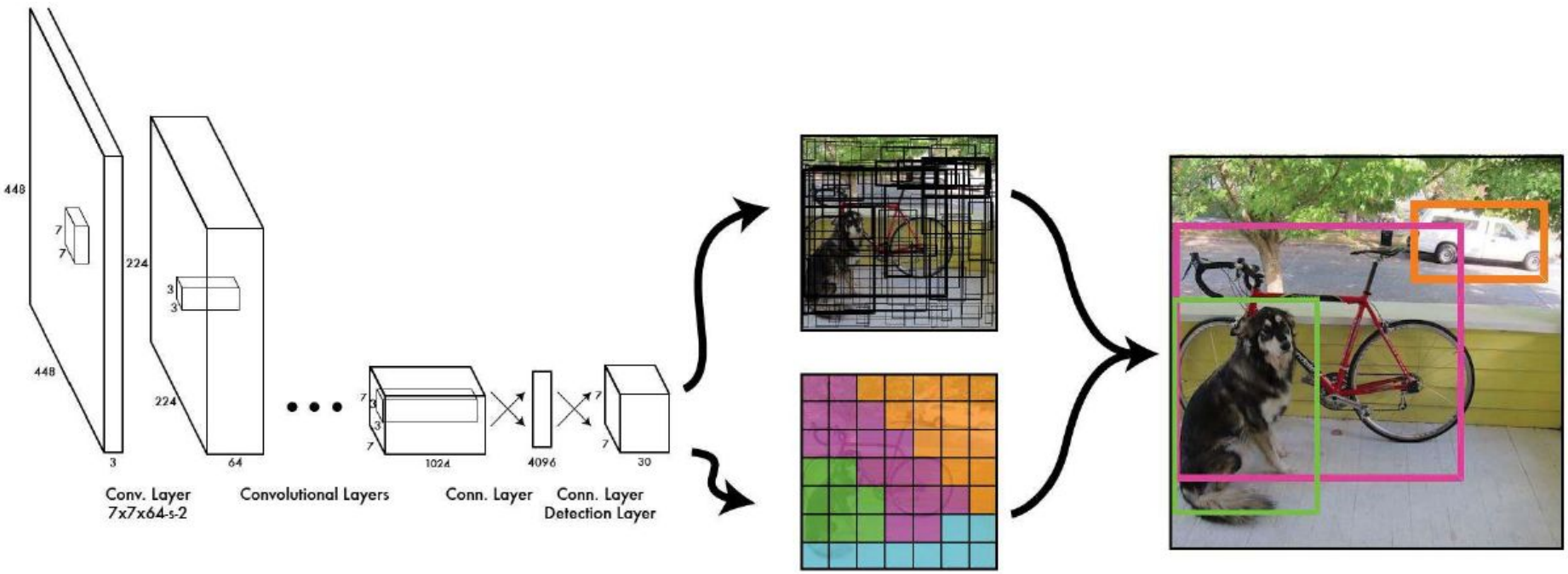


SSD Demo



YOLO Model

YOLO stands for "You Only Look Once". It is a highly optimized machine-learning model to recognize objects in videos and images.



YOLO v5 Demo

Setup YOLO on Raspberry Pi

```
#installation
git clone https://github.com/ultralytics/yolov5

cd yolov5
pip install -r requirements.txt

#inference with detect.py
python detect.py --source 0

### if there is a numpy error
pip3 install --upgrade numpy  #need to upgrade the numpy
```

Download YOLO v5 TFLite Model

#installation yolov5lite
https://tfhub.dev/neso613/lite-model/yolo-v5-tflite/tflite_model/1

Setup YOLO on Raspberry Pi

```
#go to yolov5 directory
cd yolov5

#copy the model from downloads folder to yolov5 directory
cp ~/Downloads/lite-model_yolo-v5-tflite_tflite_model_1.tflite .

# Make a copy of detect.py
cp detect.py detect-lite.py

#open detect-lite.py file
#change line 216
parser.add_argument('--imgsz', '--img', '--img-size', nargs='+', type=int, default=[320],
help='inference size h,w')

# Install tensorflow and tflite-runtime
python -m pip install --upgrade
https://storage.googleapis.com/tensorflow/mac/cpu/tensorflow-1.12.0-py3-none-
any.whl

pip install tflite-runtime

#run yolov5lite model
python detect-lite.py --source 0 --weights
lite-model_yolo-v5-tflite_tflite_model_1.tflite
```

Activity: Object Detection

Download one of the tflite object detection model from TF Hub Eg
https://tfhub.dev/tensorflow/lite-model/efficientdet/lite4/detection/metadata/2

Change directory Eg
```
cd examples/lite/examples/object_detection/raspberry_pi
cp ~/Downloads/lite-model_efficientdet_lite4_detection_metadata_2.tflite .
```

#Run object detection model Eg
```
python detect.py --model
lite-model_efficientdet_lite4_detection_metadata_2.tflite
```

Semantic Segmentation

Semantic segmentation is based on image recognition, except the classifications occur at the pixel level as opposed to the entire image.

Segmentation
Free Space

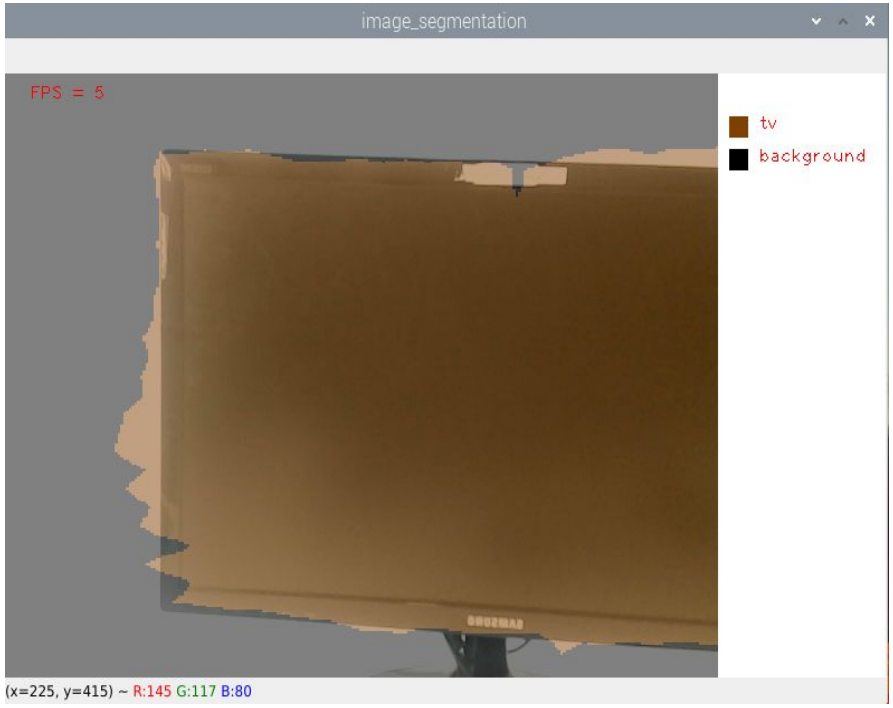
Setup Image Segmentation

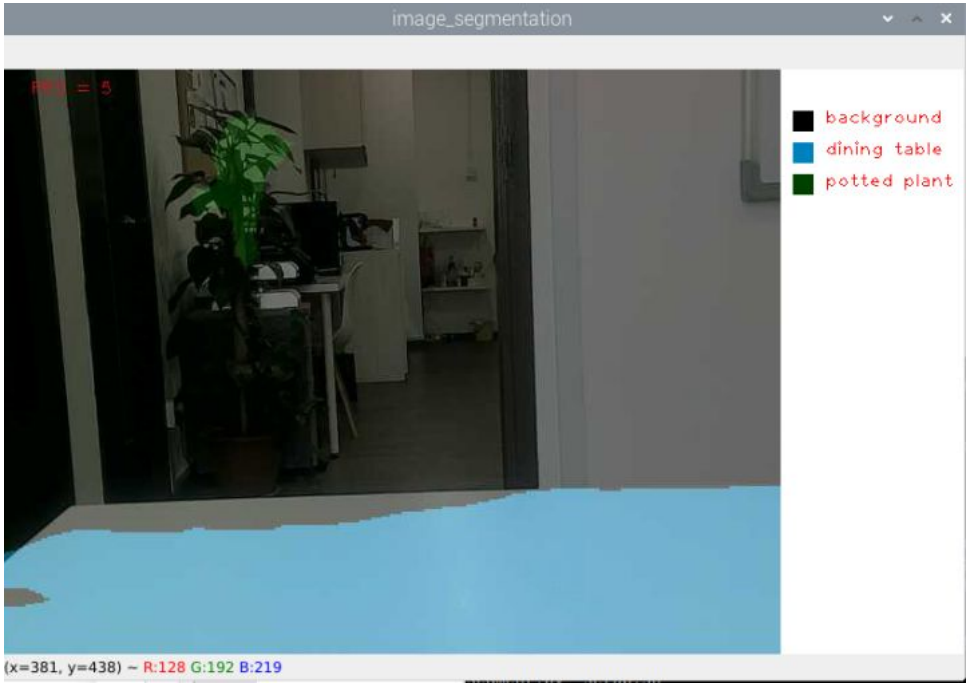
```
# change directory to image_segmentation for rpi
cd
examples/lite/examples/image_segmentation/raspberry_pi/

# Install dependencies required by the sample
sh setup.sh
```

Image Segmentation Demo

Run the object detection sample

python segment.py





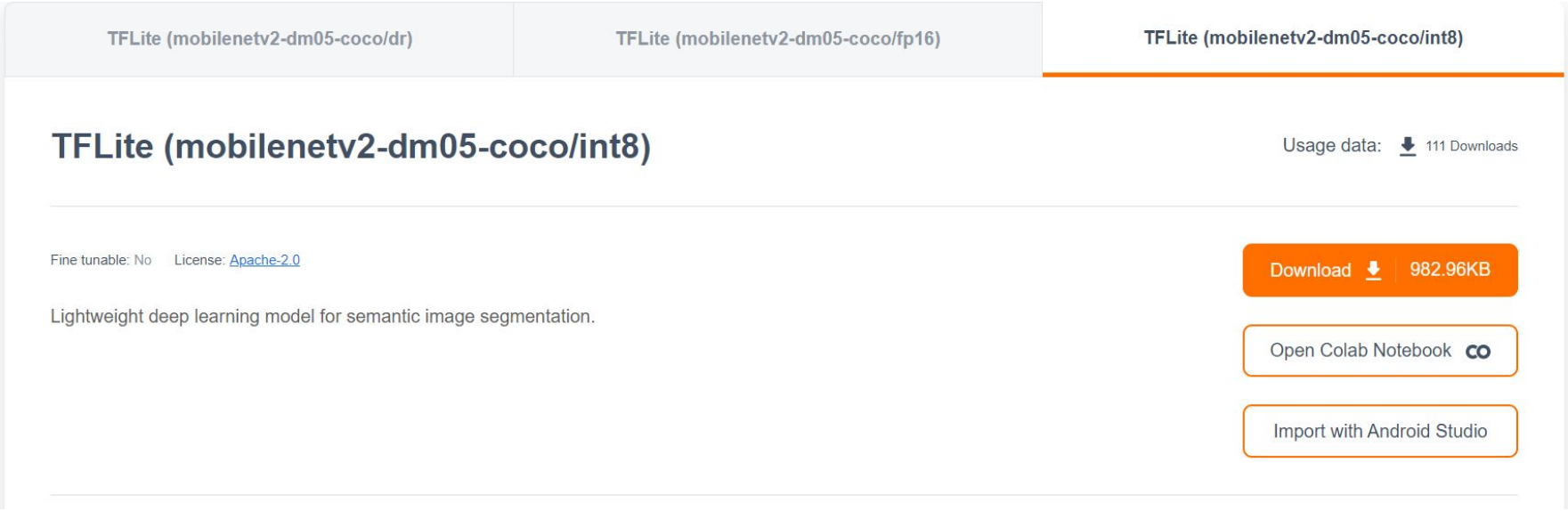
segment.py

The default segement.py also uses DeepLab Tensorflow Lite model for the image segmentation.

```
parser.add_argument(
    '--model',
    help='Name of image segmentation model.',
    required=False,
    default='deeplabv3.tflite')
```

Image Segmentation TFLite Models

Download model
https://tfhub.dev/sayakpaul/lite-model/mobilenetv2-dm05-coco/int8/1



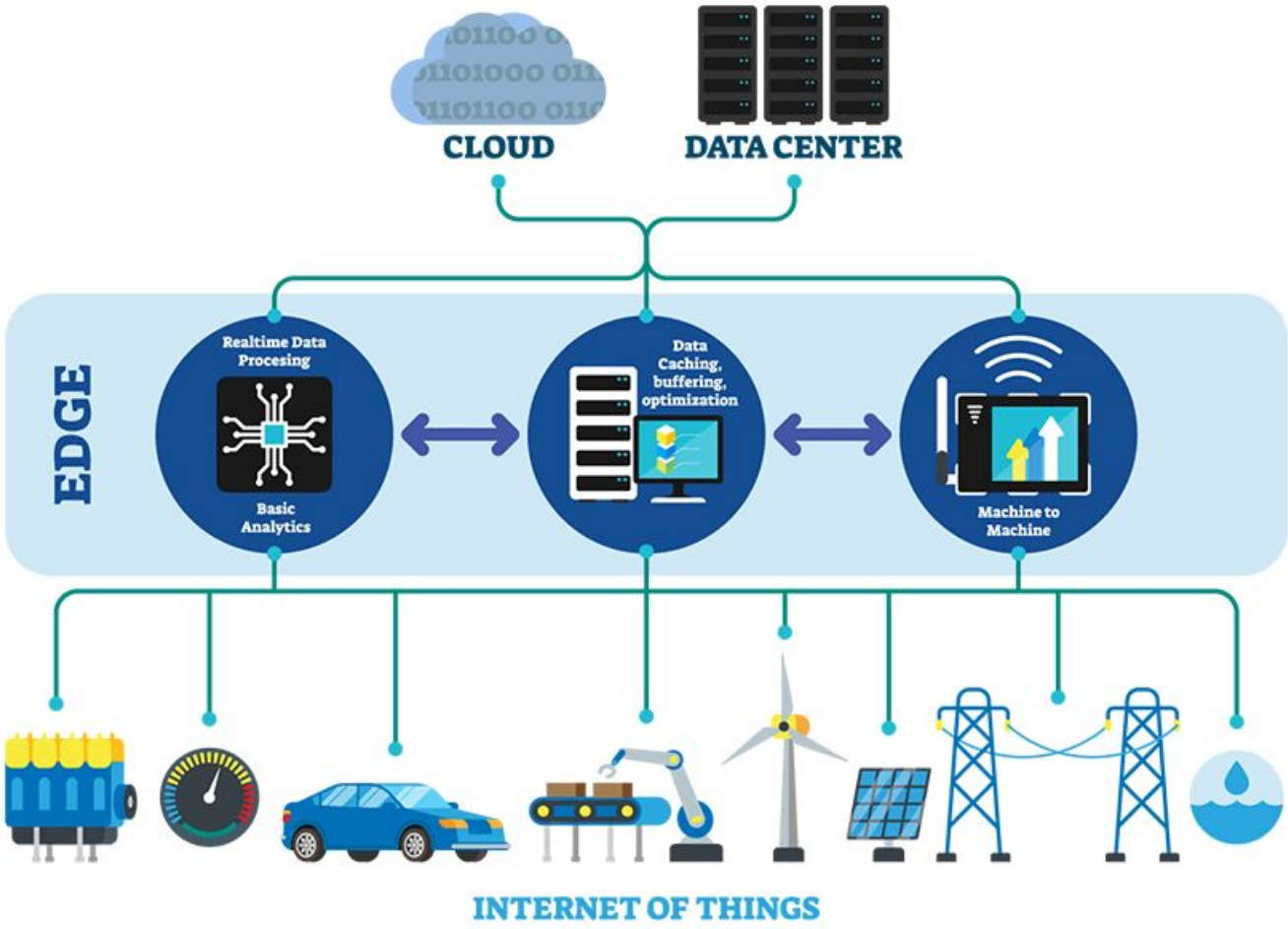
Activity: Image Segmentation

```
# Change directory
cd examples/lite/examples/image_segmentation/raspberry_pi/

# Copy file to Image Segmentation folder
cp ~/Downloads/lite-model_mobilenetv2-dm05-coco_int8_1.tflite .

# Running mobilenetv2 model
python segment.py --model
lite-model_mobilenetv2-dm05-coco_int8_1.tflite
```

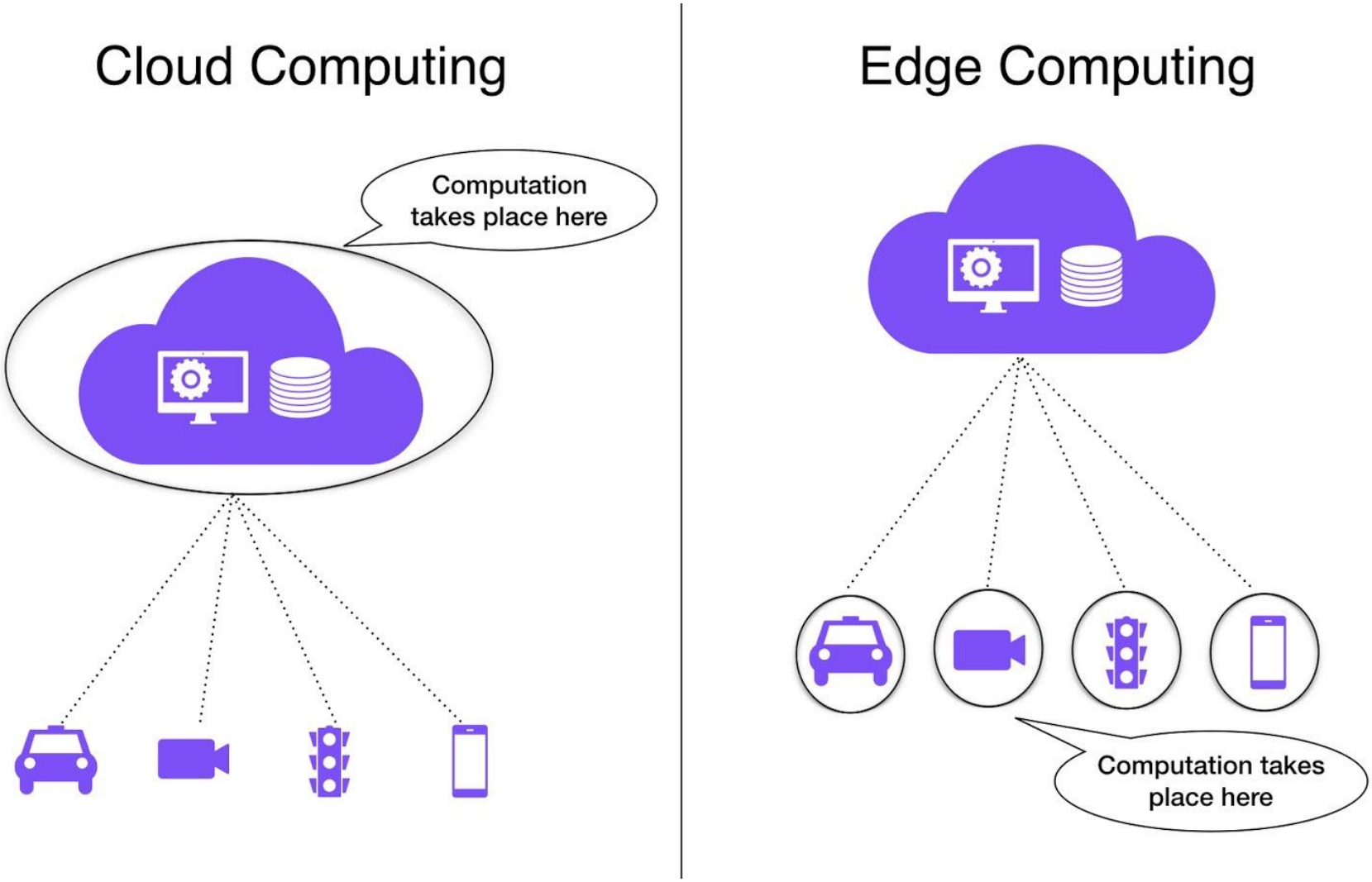
Topic 5
Computer Vision Systems

Edge vs Cloud Computing

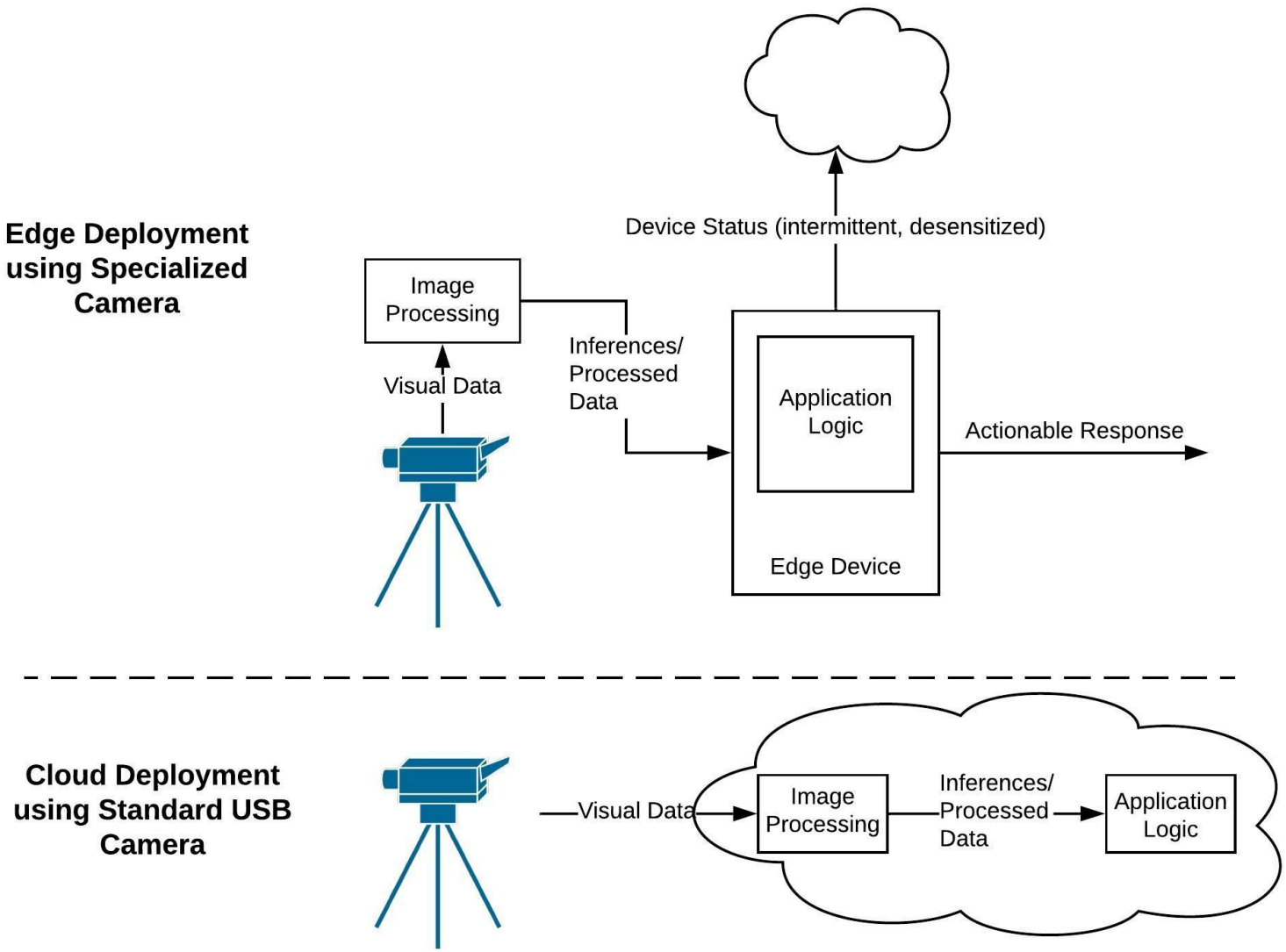
Edge computing is computing that's done at or near the source of the data, instead of relying on the cloud at one of a dozen data centers to do all the work.



Edge vs Cloud Computing



Edge vs Cloud Computer Vision Systems Deployment



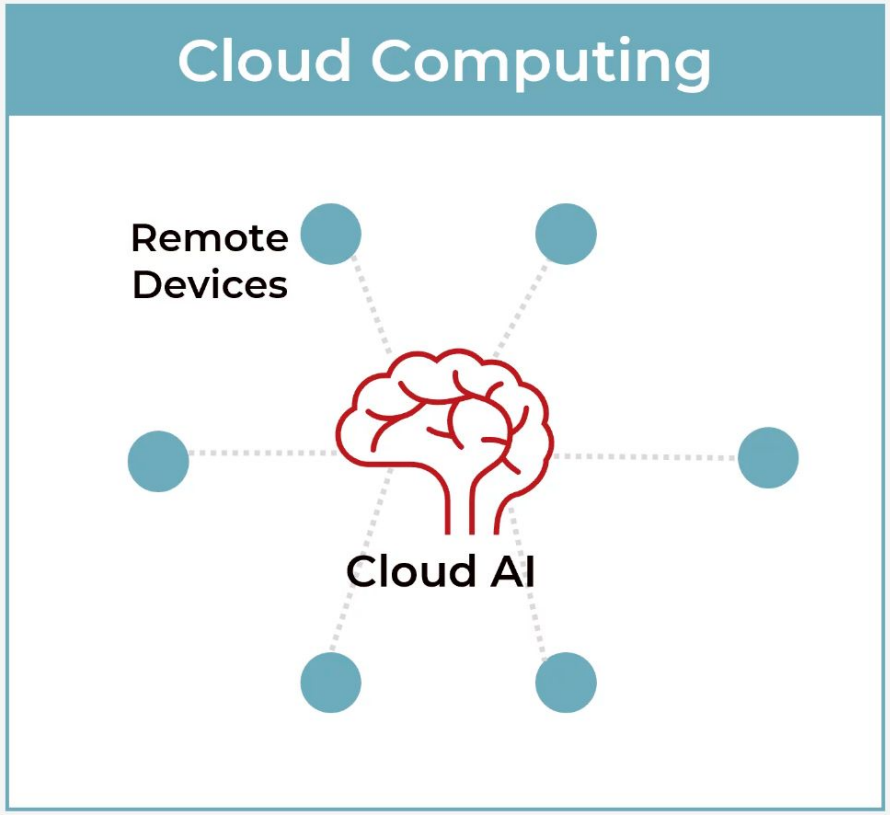
Why Edge Computing?

Edge Computing Use Case

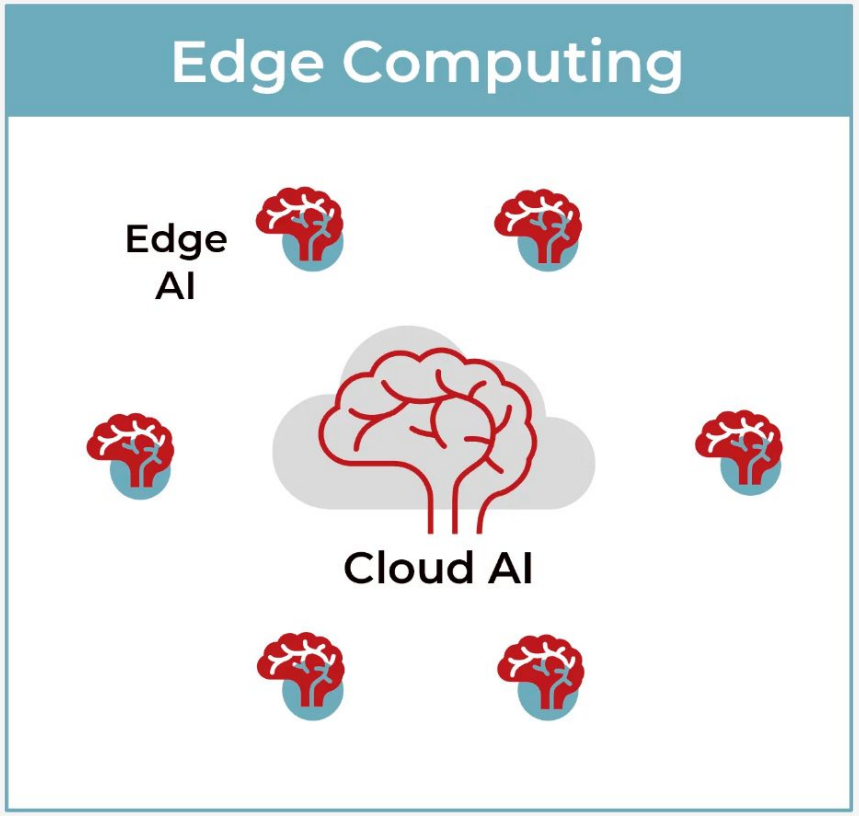
- A self-driving car is a perfect example of Edge Computing.
- In order for cars to drive down any road safely, it must observe the road in real-time and stop if a person walks in front of the car.
- In such a case, processing visual information and making a decision is done at the Edge, using Edge Computing.

The Emergence of Edge AI

Edge AI happens when AI techniques are embedded in Internet of Things (IoT) endpoints, gateways, and other devices at the point of use



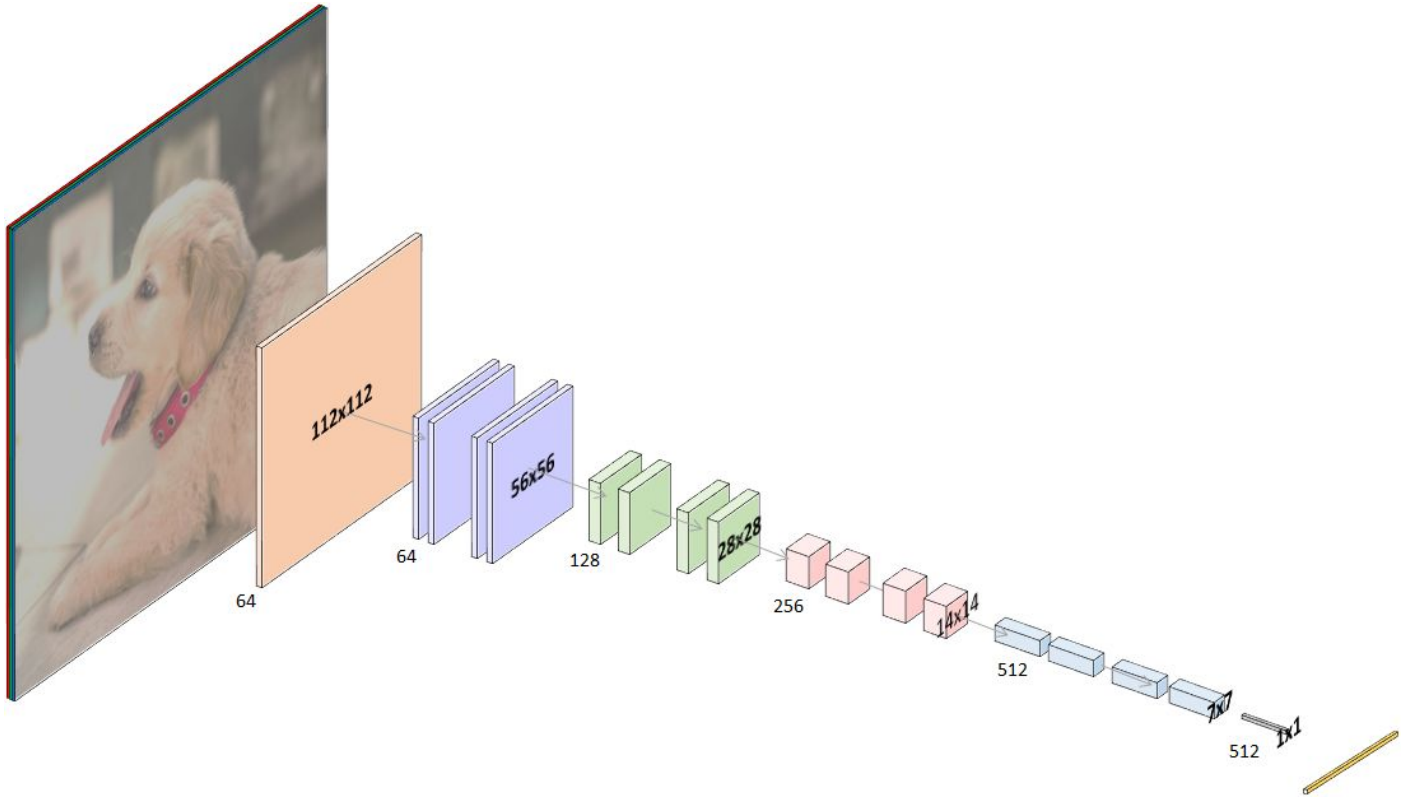
Remote devices connect to an AI in the cloud, where data processing occurs.



With increasing power and smaller size, AI processors perform computation and inference on the device.

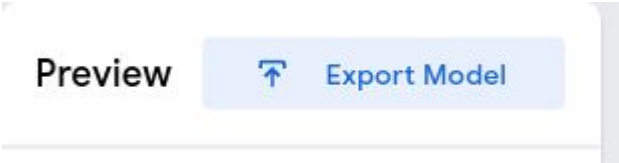
Demo of Edge AI - Transfer Learning

- Transfer learning is a technique for re-training a DNN model on a new dataset, which takes less time than training a network from scratch.
- With transfer learning, the weights of a pre-trained model are fine-tuned to classify a customized dataset.

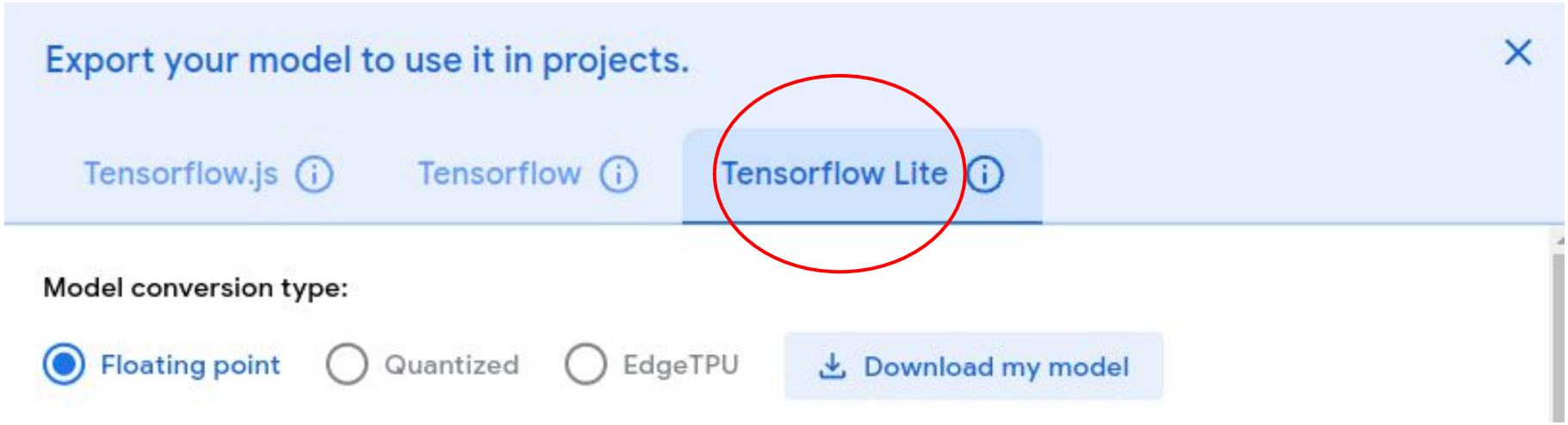


Step 1: Exporting TFlite Model

- The first demo is Transfer Learning on Image Classification.
- From previous scissor-paper-stone activity, export tflite model after training and testing.

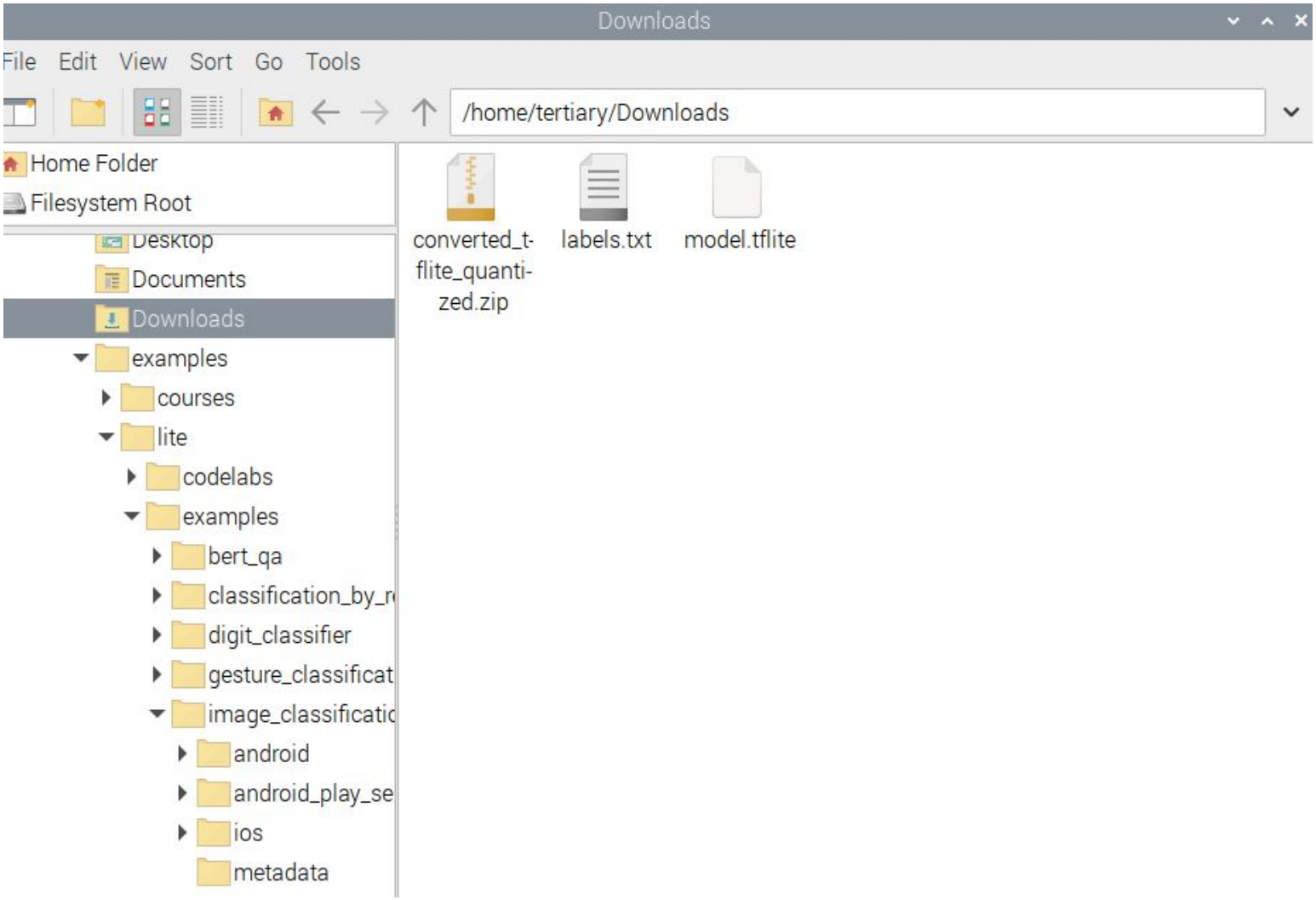


- Download the quantized tflite model



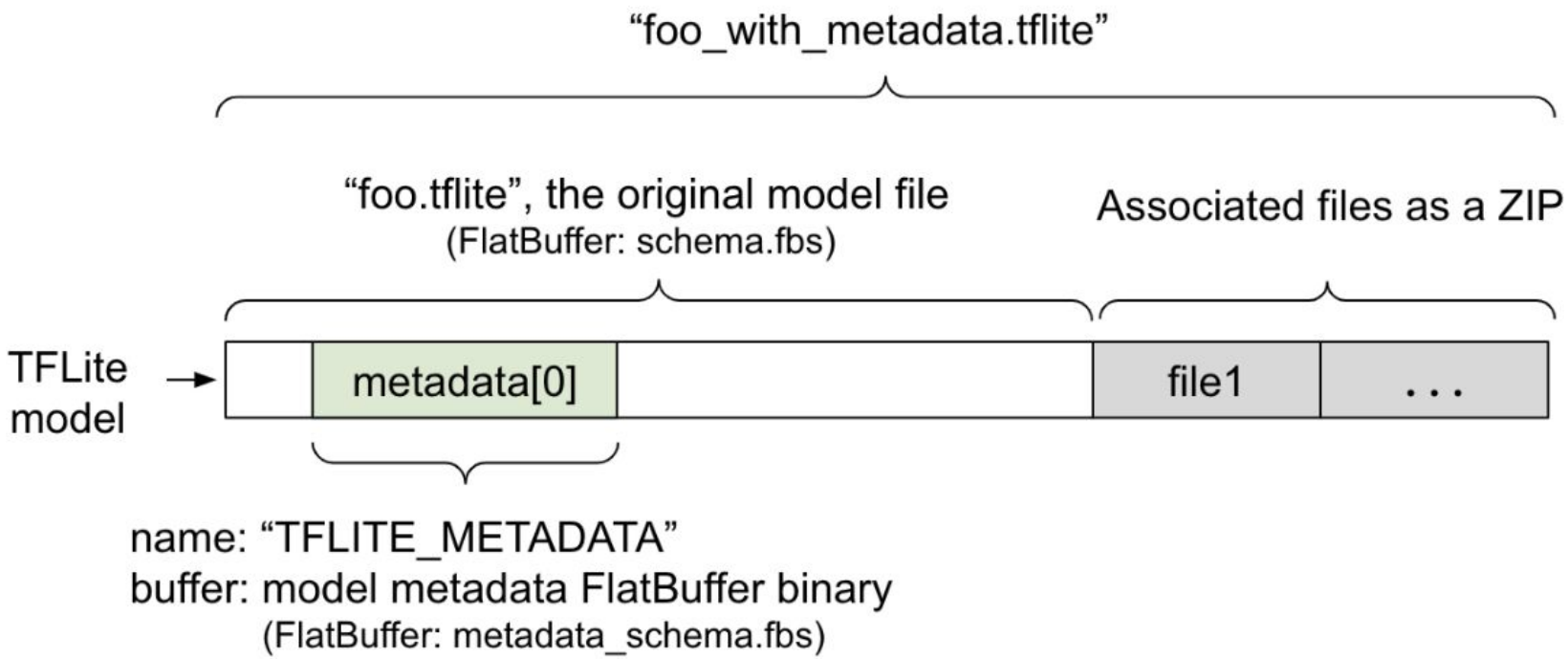
Step 2: Extract the Zip File

Extract the file from Download folder



TFLite Metadata

- TensorFlow Lite metadata provides a standard for model descriptions.
- The metadata is an important source of knowledge about what the model does and its input / output information

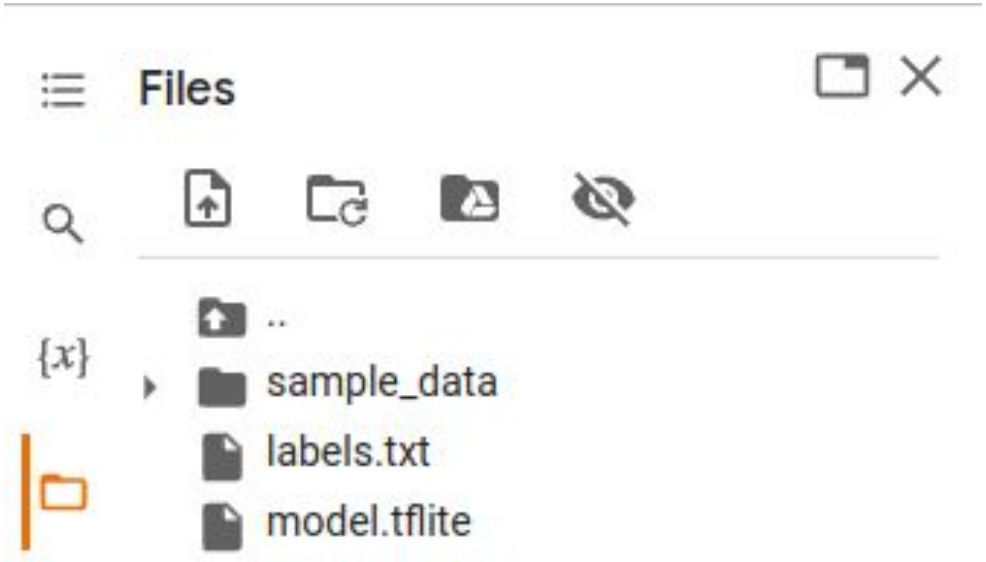


Step 3-1: Convert to TFlite Metadata Model

- You need to regenerate a tflite model with metadata.
- Follow the step here

https://colab.research.google.com/drive/1olvUDSQ5nFrLgusOhf78HpHsWWZ0j3Iw?usp=sharing

- Drag the label.txt and model.tflite file to the file folder on the top left



Step 3-2: Install TFLite-Support Package

- Run the code below to install TensorFlow Lite Support Pypi package

```
!pip install tflite-support-nightly
```

- Import the required packages

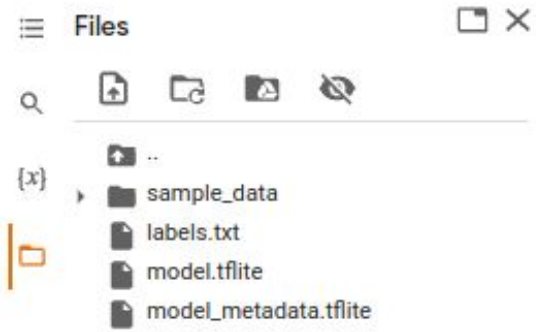
```
[9] from tflite_support.metadata_writers import image_classifier
    from tflite_support.metadata_writers import writer_utils
```

Step 3-3: Create Model Metadata

- Run the code below to get the metadata file

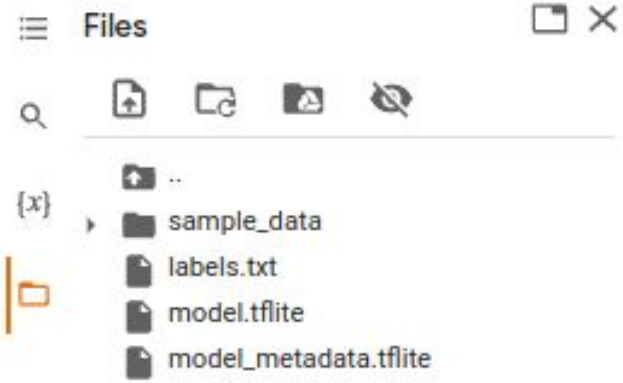
```
ImageClassifierWriter = image_classifier.MetadataWriter
_MODEL_PATH = "model.tflite"
# Task Library expects label files that are in the same format as the one below.
_LABEL_FILE = "labels.txt"
_SAVE_TO_PATH = "model_metadata.tflite"
# Normalization parameters is required when reprocessing the image. It is
# optional if the image pixel values are in range of [0, 255] and the input
# tensor is quantized to uint8. See the introduction for normalization and
# quantization parameters below for more details.
# https://www.tensorflow.org/lite/convert/metadata#normalization_and_quantization_parameters)
_INPUT_NORM_MEAN = 127.5
_INPUT_NORM_STD = 127.5

# Create the metadata writer.
writer = ImageClassifierWriter.create_for_inference(
    writer_utils.load_file(_MODEL_PATH), [_INPUT_NORM_MEAN], [_INPUT_NORM_STD],
    [_LABEL_FILE])

# Verify the metadata generated by metadata writer.
print(writer.get_metadata_json())

# Populate the metadata into the model.
writer_utils.save_file(writer.populate(), _SAVE_TO_PATH)
```

- After running the metadata file will be at the file folder

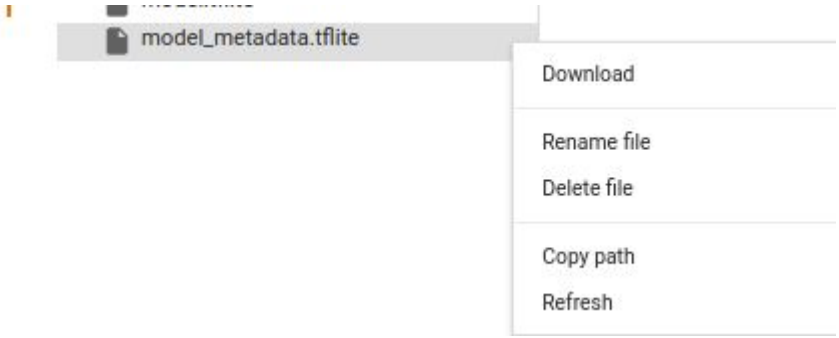


Step 3-4: Downloading TFlite Metadata Model

- After running the metadata file will be at the file folder

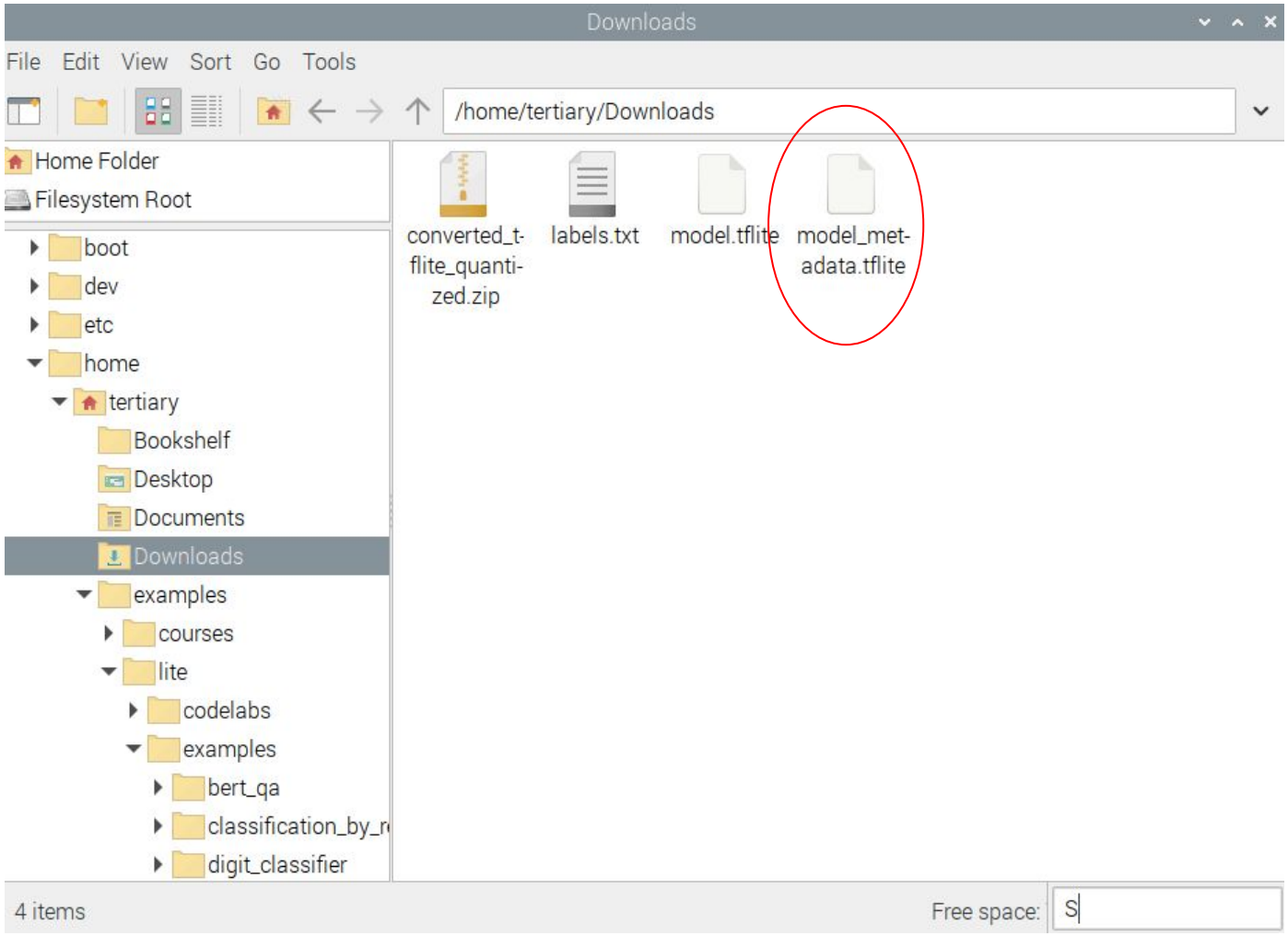


- Download the metadata.tflite file



Step 4: Download TFLite Metadata Model to Raspberry Pi

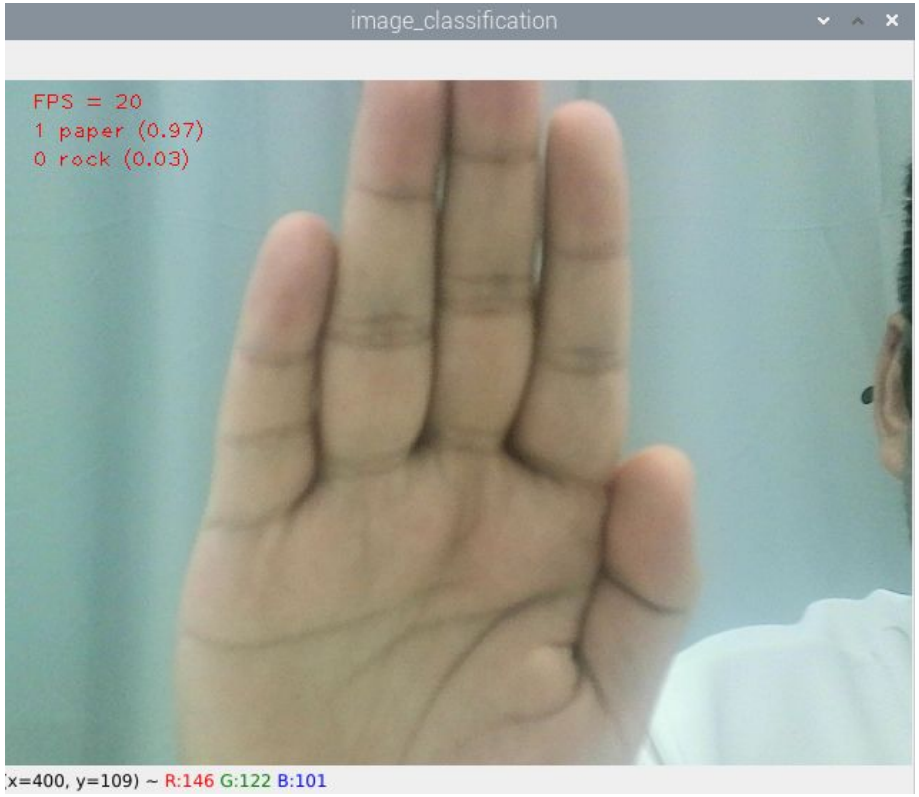
The metadata.tflite file will be shown in the Download folder of Raspberry Pi

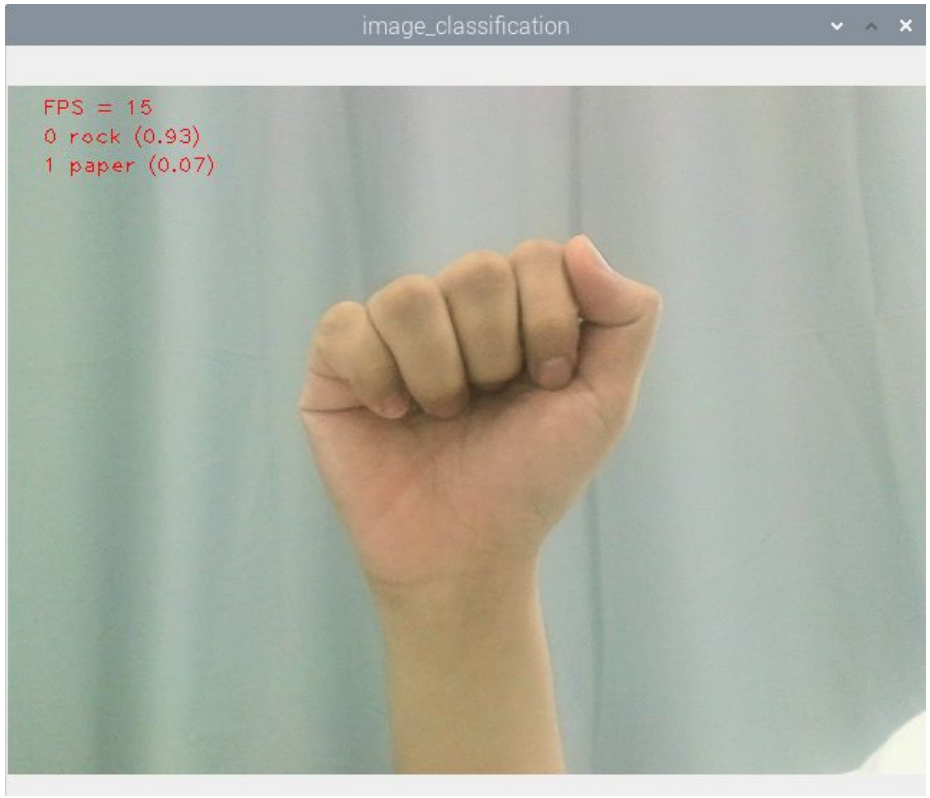


Step 5: Image Classifier with Custom TFLite Model

```
# Open terminal
# Change directory
cd examples/lite/examples/image_classification/raspberry_pi

# Copy the file to
cp ~/Downloads/model_metadata.tflite .

# Run the custom trained model
python classify.py --model model_metadata.tflite
```

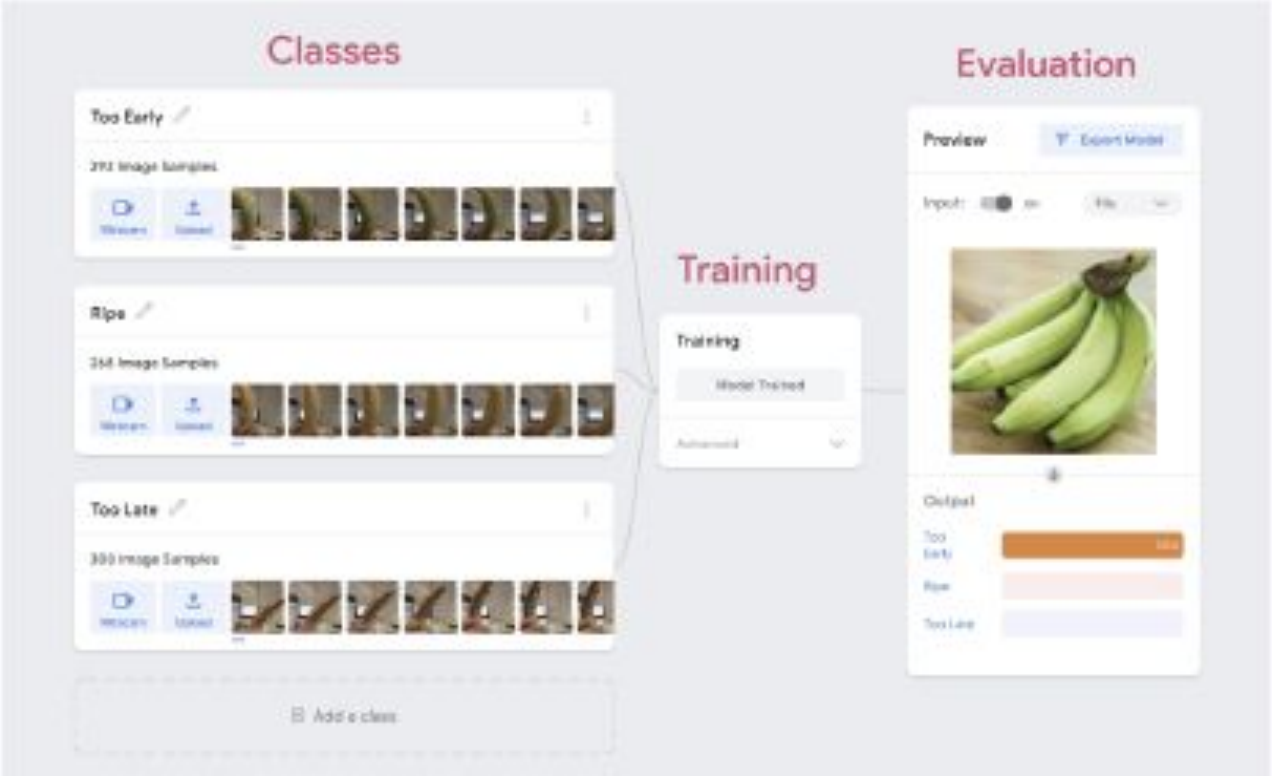
Custom TFLite Image Classification Model Demo on Raspberry Pi



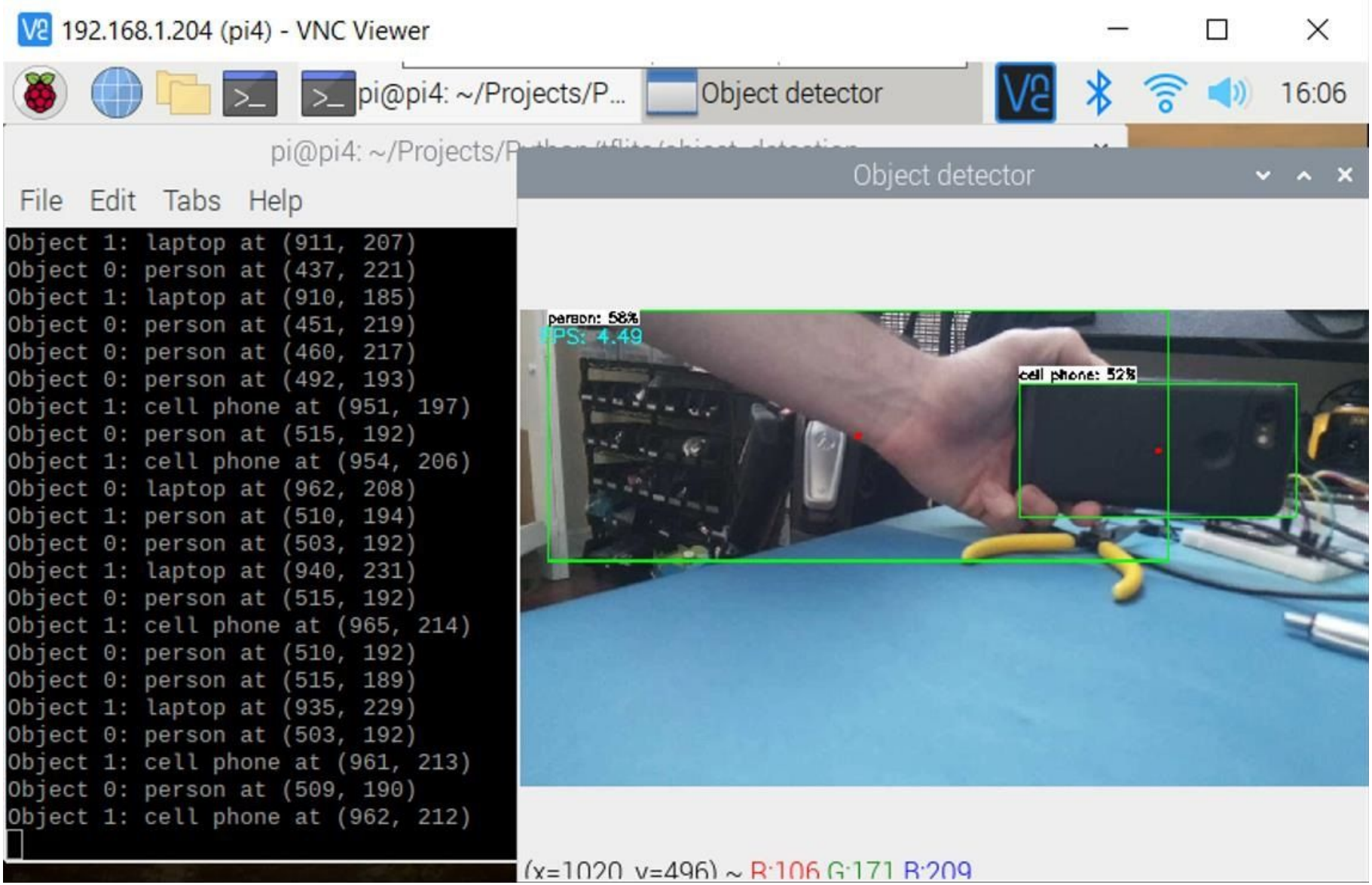


Activity: Custom Image Classification

- Train an image classifier using teachable machine
- Export the TFLite model
- Convert the model to TFLite with metadata
- Upload the custom image classifier model to Raspberry Pi
- Test the image classification on Raspberry Pi



Custom TFLite Object Detection Model Demo on Raspberry Pi



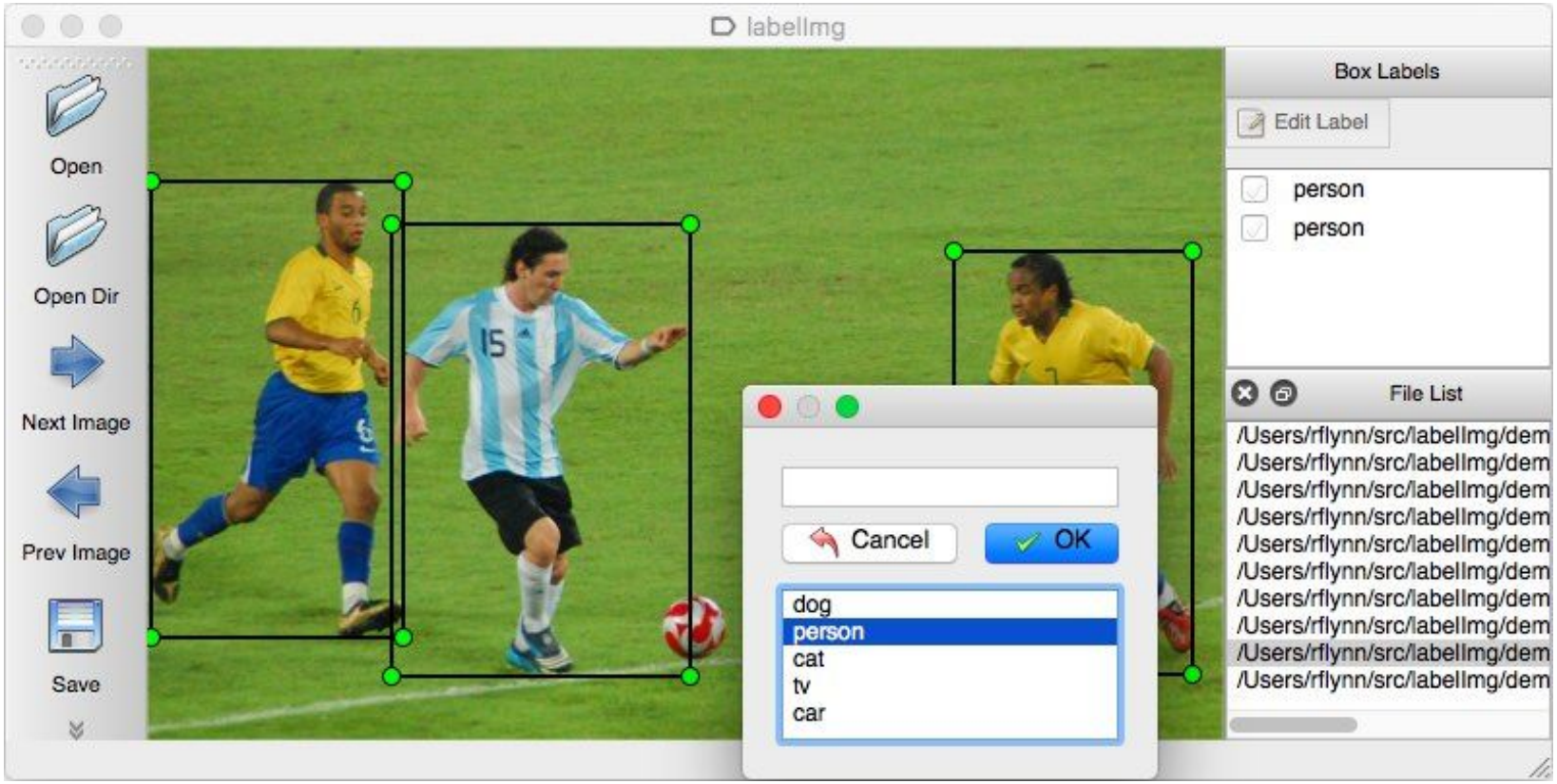
Step 1: Install Image Labeling Software

```
# Install labelImg  software from
https://tzutalin.github.io/labelImg/

# Download linux_v1.4.3
# Click on Linux_v1.4.3
# Unzip the Linux_v1.4.3 in Downloads

# Go to Linux_v1.4.3 directory
cd ~/Downloads/linux_v1.4.3
```

```
# Install devtool and labelImg
sudo apt-get install pyqt5-dev-tools -y
pip install labelImg

# Run labelImg
labelImg
```

Step 2: Label Your Own Dataset

- Create 2 folder name train and validate
- Put training image and validate image into the respective folder
- load the image into labelImg file and box out the object of interest

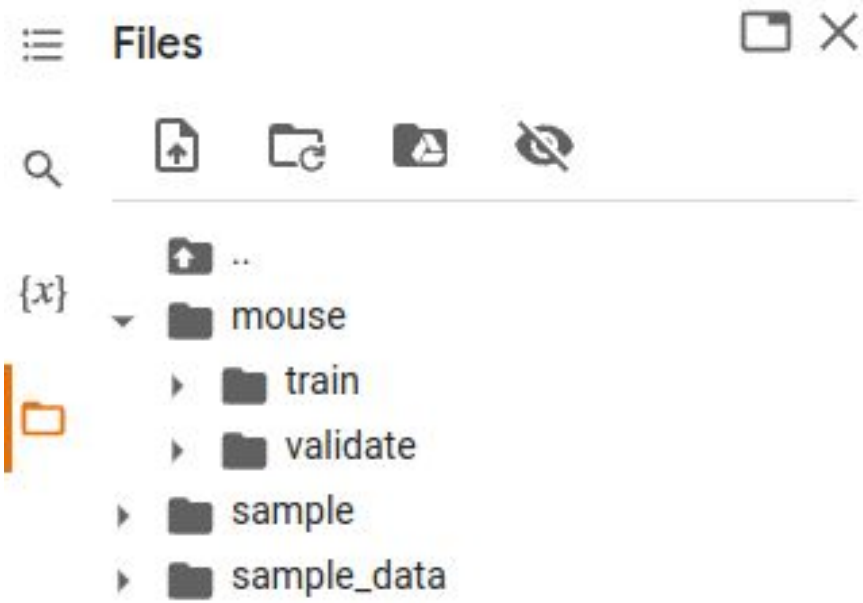


Step 3: Upload Dataset to Colab

Save the file into the same folder as the image. It will be save as an .xml file type

Go to link
https://colab.research.google.com/drive/1XJvj5J15-Ww09gNRYwvIh_ydADuKw8WC?usp=sharing

Upload your folder into the File folder on the top left of colab



Step 4-1: Install and Import Packages

- Run the code below to install required packages

```
!pip install -q tflite-model-maker
!pip install -q tflite-support
```

- Import the library

```
import numpy as np
import os

from tflite_model_maker.config import ExportFormat, QuantizationConfig
from tflite_model_maker import model_spec
from tflite_model_maker import object_detector

from tflite_support import metadata

import tensorflow as tf
assert tf.__version__.startswith('2')

tf.get_logger().setLevel('ERROR')
from absl import logging
logging.set_verbosity(logging.ERROR)
```

Step 4-2: Load the Dataset

- Modify your dataset directory and label accordingly below.

```
train_data = object_detector.DataLoader.from_pascal_voc(
    'mouse/train',
    'mouse/train',
    ['mouse']
)

val_data = object_detector.DataLoader.from_pascal_voc(
    'mouse/validate',
    'mouse/validate',
    ['mouse']
)
```

Step 4-3: Specify the Model Architecture

Specify your model below

```
spec = model_spec.get('efficientdet_lite0')
```

Model architecture	Size(MB)*	Latency(ms)**	Average Precision***
EfficientDet-Lite0	4.4	146	25.69%
EfficientDet-Lite1	5.8	259	30.55%
EfficientDet-Lite2	7.2	396	33.97%
EfficientDet-Lite3	11.4	716	37.70%
EfficientDet-Lite4	19.9	1886	41.96%

Step 4-4: Train the Model

Run the code below to start training your model

```
model = object_detector.create(train_data, model_spec=spec, batch_size=4, train_whole_model=True, epochs=20, validation_data=val_data)
```

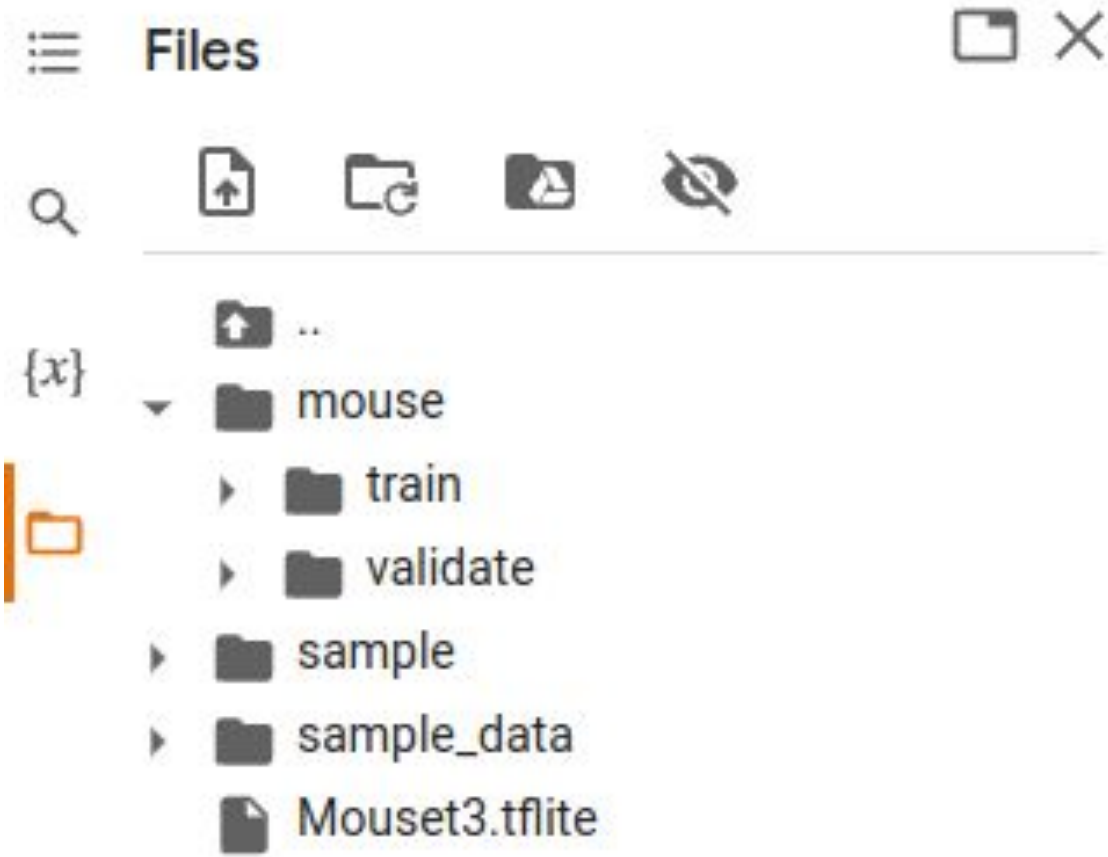
The training will typically take 5-15 minutes to complete

```
Epoch 1/20
4/4 [==============================] - 51s 3s/step - det_loss: 1.8197 - cls_loss: 1.1519 - box_loss: 0.0134 - reg_l2_loss: 0.0630 - loss: 1.8827 - learning_rate: 0.0067 - gradien
Epoch 2/20
4/4 [==============================] - 5s 1s/step - det_loss: 1.7492 - cls_loss: 1.1247 - box_loss: 0.0125 - reg_l2_loss: 0.0630 - loss: 1.8122 - learning_rate: 0.0049 - gradient
Epoch 3/20
4/4 [==============================] - 5s 1s/step - det_loss: 1.6677 - cls_loss: 1.0924 - box_loss: 0.0115 - reg_l2_loss: 0.0630 - loss: 1.7307 - learning_rate: 0.0048 - gradient
Epoch 4/20
4/4 [==============================] - 5s 2s/step - det_loss: 1.6055 - cls_loss: 1.0410 - box_loss: 0.0113 - reg_l2_loss: 0.0630 - loss: 1.6685 - learning_rate: 0.0046 - gradient
Epoch 5/20
4/4 [==============================] - 11s 4s/step - det_loss: 1.5397 - cls_loss: 0.9994 - box_loss: 0.0108 - reg_l2_loss: 0.0630 - loss: 1.6027 - learning_rate: 0.0044 - gradien
Epoch 6/20
4/4 [==============================] - 4s 1s/step - det_loss: 1.4733 - cls_loss: 0.9398 - box_loss: 0.0107 - reg_l2_loss: 0.0630 - loss: 1.5363 - learning_rate: 0.0040 - gradient
Epoch 7/20
4/4 [==============================] - 5s 1s/step - det_loss: 1.3866 - cls_loss: 0.8781 - box_loss: 0.0102 - reg_l2_loss: 0.0630 - loss: 1.4496 - learning_rate: 0.0037 - gradient
Epoch 8/20
4/4 [==============================] - 4s 1s/step - det_loss: 1.3229 - cls_loss: 0.8204 - box_loss: 0.0101 - reg_l2_loss: 0.0630 - loss: 1.3859 - learning_rate: 0.0033 - gradient
Epoch 9/20
4/4 [==============================] - 5s 1s/step - det_loss: 1.1468 - cls_loss: 0.6889 - box_loss: 0.0092 - reg_l2_loss: 0.0630 - loss: 1.2098 - learning_rate: 0.0029 - gradient
Epoch 10/20
4/4 [==============================] - 6s 2s/step - det_loss: 1.1880 - cls_loss: 0.7529 - box_loss: 0.0087 - reg_l2_loss: 0.0630 - loss: 1.2510 - learning_rate: 0.0025 - gradient
Epoch 11/20
4/4 [==============================] - 5s 1s/step - det_loss: 1.1007 - cls_loss: 0.6518 - box_loss: 0.0090 - reg_l2_loss: 0.0630 - loss: 1.1637 - learning_rate: 0.0021 - gradient
Epoch 12/20
4/4 [==============================] - 5s 1s/step - det_loss: 1.0363 - cls_loss: 0.5714 - box_loss: 0.0093 - reg_l2_loss: 0.0630 - loss: 1.0993 - learning_rate: 0.0017 - gradient
Epoch 13/20
4/4 [==============================] - 5s 1s/step - det_loss: 1.0796 - cls_loss: 0.6214 - box_loss: 0.0092 - reg_l2_loss: 0.0630 - loss: 1.1426 - learning_rate: 0.0013 - gradient
Epoch 14/20
4/4 [==============================] - 5s 1s/step - det_loss: 1.0815 - cls_loss: 0.5961 - box_loss: 0.0097 - reg_l2_loss: 0.0630 - loss: 1.1445 - learning_rate: 9.8258e-04 - grad
```

Step 4-5: Convert Model to TFlite

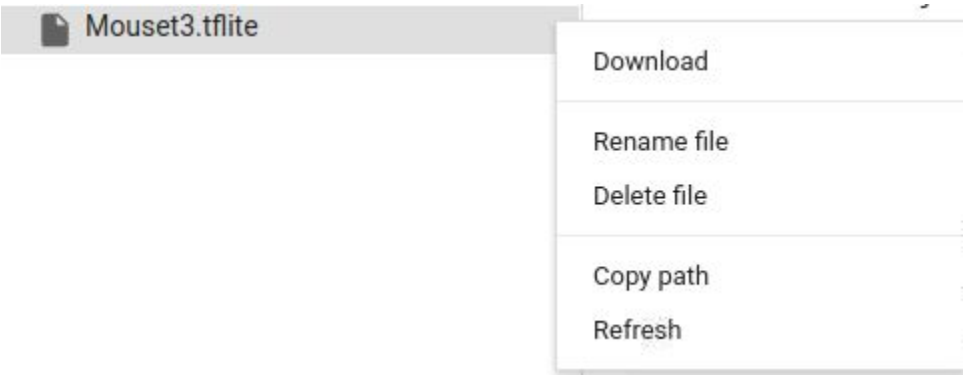
Convert your model to TFlite as follows

```
model.export(export_dir='.', tflite_filename='Mouset3.tflite')
```

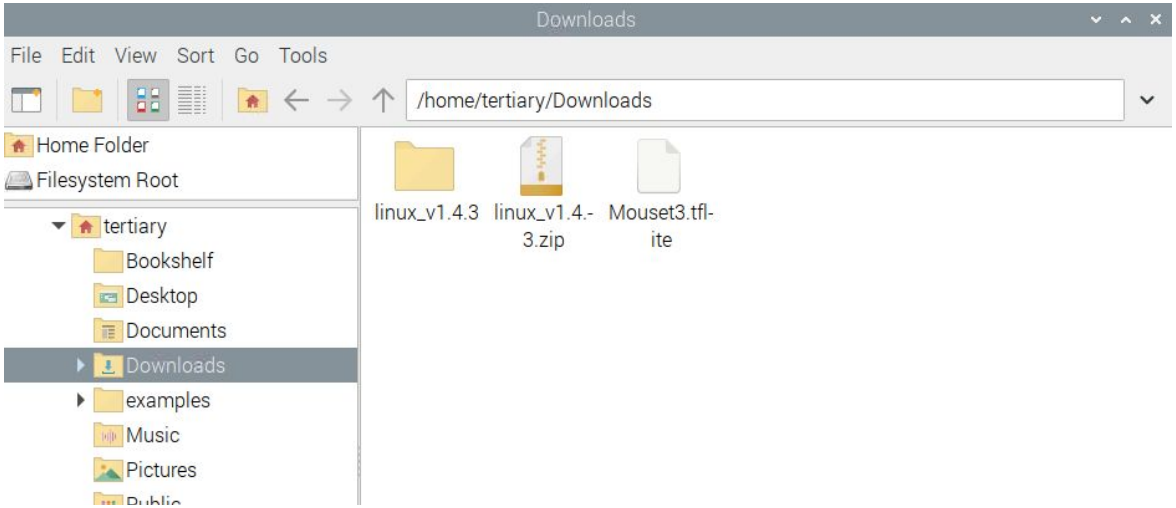


Step 4-6: Download TFlite Model

- Download the tflite from the File Folder to Raspberry pi

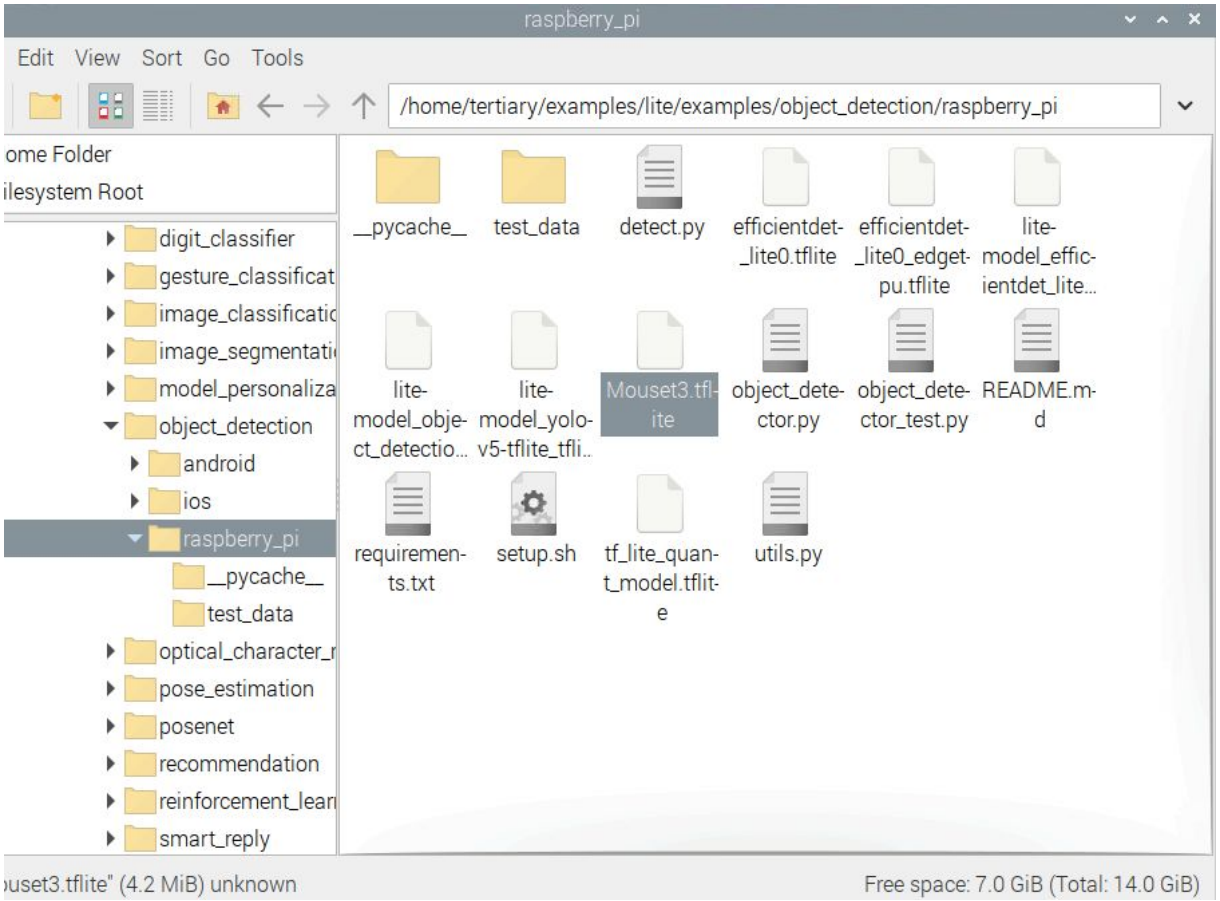


- The tflite model will be shown in the Download folder of Raspberry Pi



Step 4-7: Move the Model to Raspberry Pi

copy the file to /home/tertiary/examples/lite/examples/object_detection/raspberry_pi directory



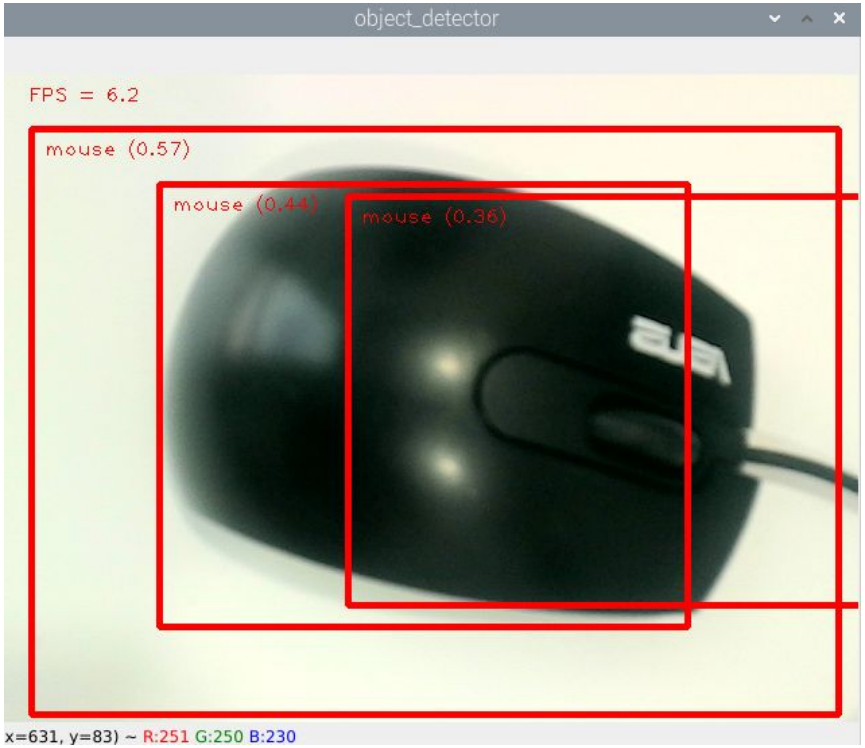
Step 5: Run the Custom Model on Raspberry Pi

Change directory to object detection folder

```
cd
~/examples/lite/examples/object_detection/raspberry_pi/
```

Run your custom trained model

```
python detect.py --model Mouset3.tflite
```



Activity: Custom Object Detection

- Extract your own dataset using LabelImg
- Train the EfficientNet using your dataset
- Convert the model to TFLite
- Upload to Raspberry Pi
- Test the object detection on Raspberry PI

Final Assessment

Written Assessment (Q&A)

This is the Written Assessment (Q&A)

Duration: 60 mins

1. The assessor will pass the questions in hardcopy to you. There are 13 questions. You need to answer all the questions.
2. This is an open book exam that must be completed individually.
3. You need to get all answers correct to be competent.

Practical Performance

This is a Written Assessment (PP)

Duration: 90 mins

1. The assessor will pass the practical problems in hardcopy to you.
2. This is an open book exam that must be completed individually.

Submission Procedure:

1. After completion, please pass the hardcopy to the assessor

Summary Q&A

Feedback

https://goo.gl/R2eumq

TRAQOM Survey

- You will receive a TRAQOM link after the class.
- Please submit TRAQOM feedback and survey after the class.

Thank You!

Man Guo Chang
gc.man.sg@gmail.com