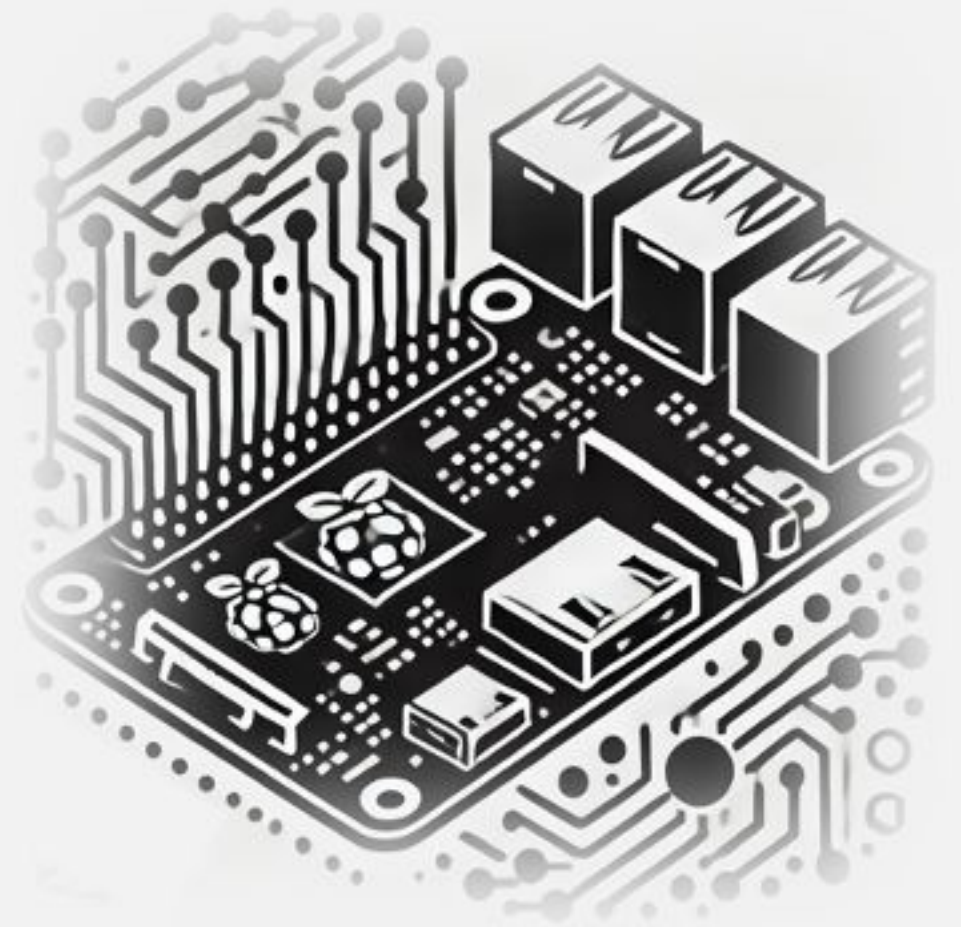


IESTI05 – Edge AI

Machine Learning System Engineering

8. Object Detection: Fundamentals



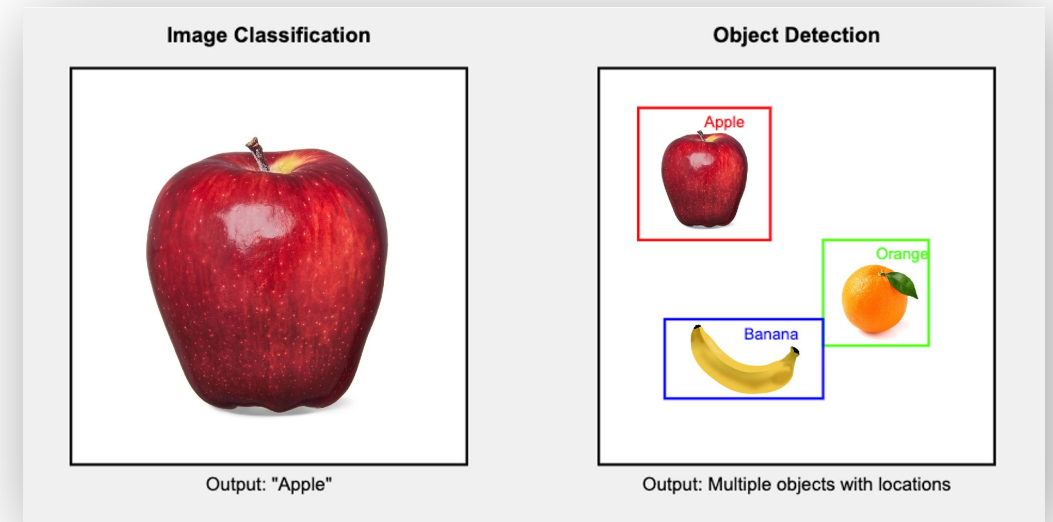
Object Detection

Fundamentals & SSD-Mobilenet Model

What is Object Detection?

Object detection is a computer vision task that **identifies and locates objects** within an image or video. Unlike simple image classification, it doesn't just say "there is an apple in this image," but rather "**here is an apple at these specific coordinates.**"

- **Identifies**: Determines the class of an object (e.g., car, person).
- **Localizes**: Provides a bounding box to indicate precisely where the object is located.



Can identify multiple objects with locations in the image

Key Challenges in Detection

Common Hurdles in Detection



Varying Scales

Objects can be large or small, near or far.



Occlusion

Objects can be partially hidden by others.



Background Clutter

Complex backgrounds can confuse the model.



Real-time Performance

Processing needs to be fast for live video.

- **Multiple Objects:** Images often contain multiple objects of different classes, sizes, and positions, making it challenging to detect and classify each one accurately.
- **Real-Time Performance:** Many applications require fast inference times, especially on edge devices with limited computational resources; striking a balance between accuracy and speed is essential.

Two-Stage vs Single-Stage

Two-Stage Detectors

Models like **Faster R-CNN** first propose regions of interest and then classify each region. This approach is accurate but computationally intensive.

Single-Stage Detectors

Models such as **SSD-MobileNet**, **YOLO**, and **EfficientDet** predict bounding boxes and class probabilities in one forward pass, making them faster and more suitable for edge devices.

Trade-offs

While two-stage detectors offer higher accuracy, single-stage detectors provide a better balance between speed and accuracy, crucial for real-time applications.

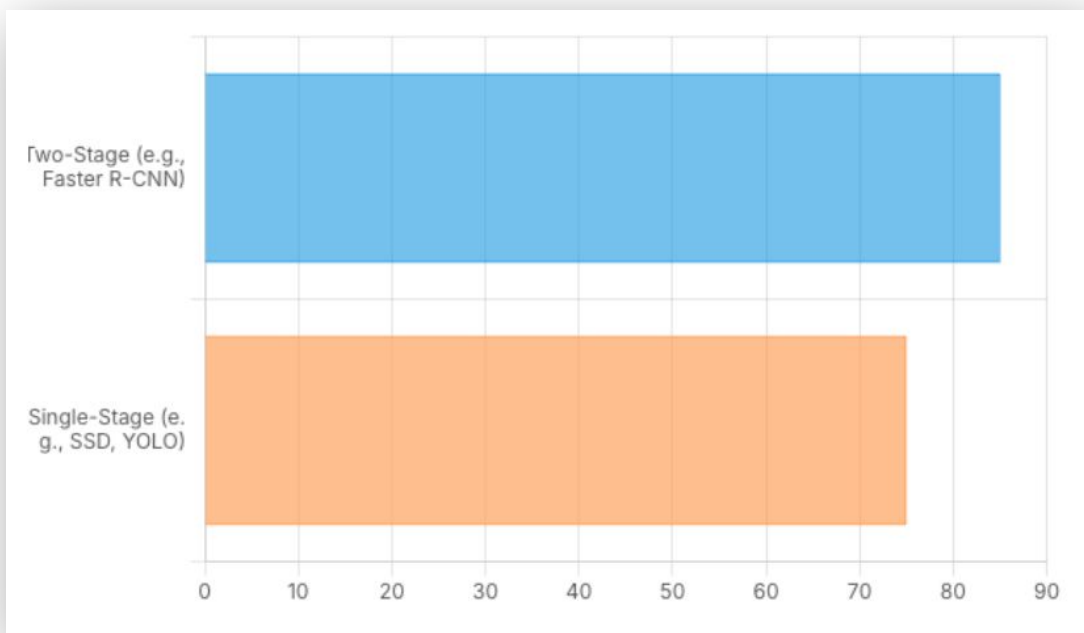
Examples

SSD MobileNet is a popular single-stage detector that combines the efficiency of SSD with the lightweight MobileNet backbone, ideal for edge deployment.

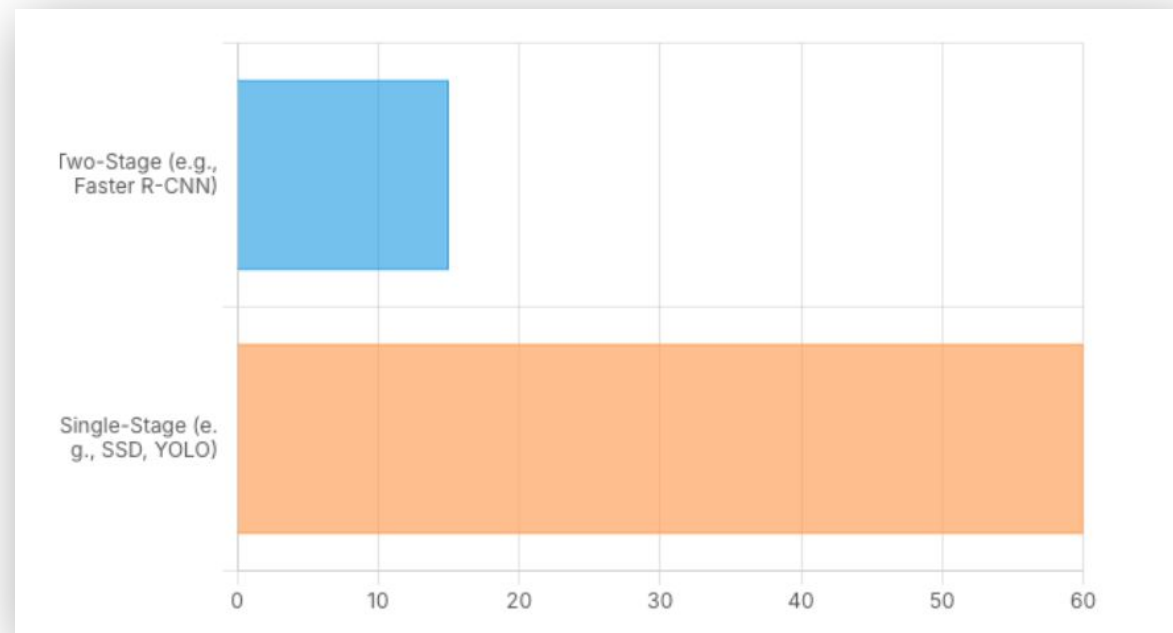
Detector Architectures: A Trade-Off

Models generally fall into two categories, each balancing accuracy and speed in different ways.

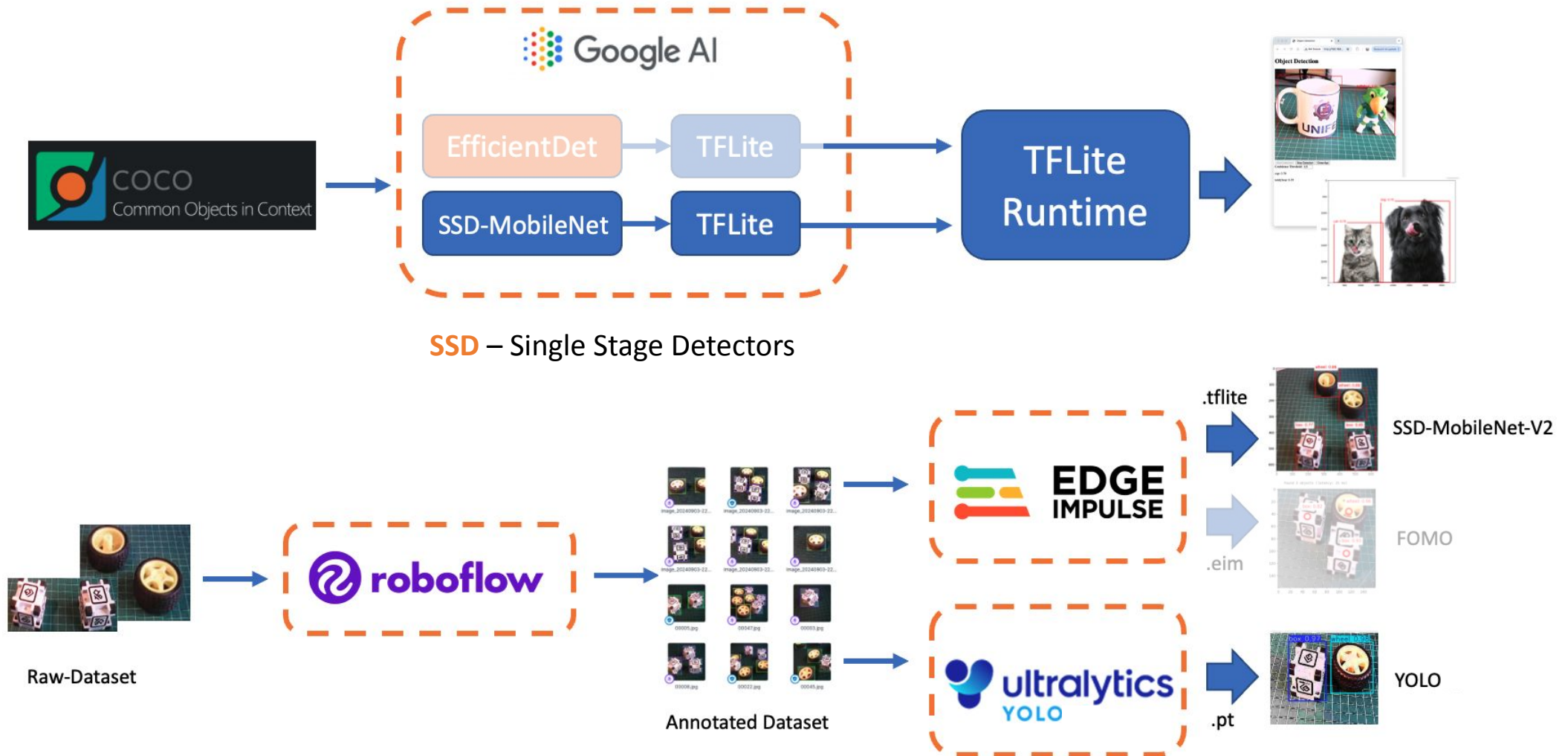
Accuracy



Speed



Models covered in the course



SSD-MobileNet: The best of two worlds

SSD (Single Shot Detector)

A fast, single-stage detection framework that finds objects in one pass.

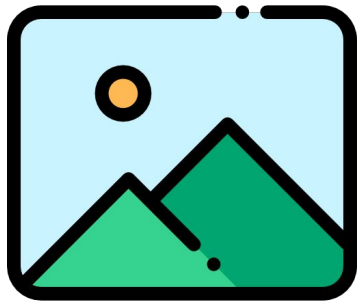


MobileNet

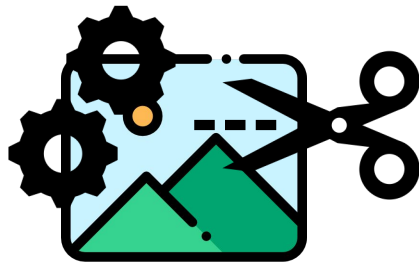
A lightweight backbone network designed for efficiency on mobile/edge devices.



Object Detection: Inference Pipeline



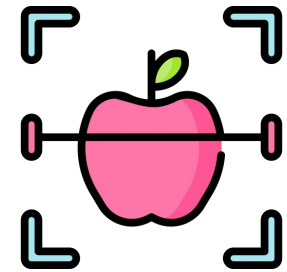
Input Image



Pre-Process

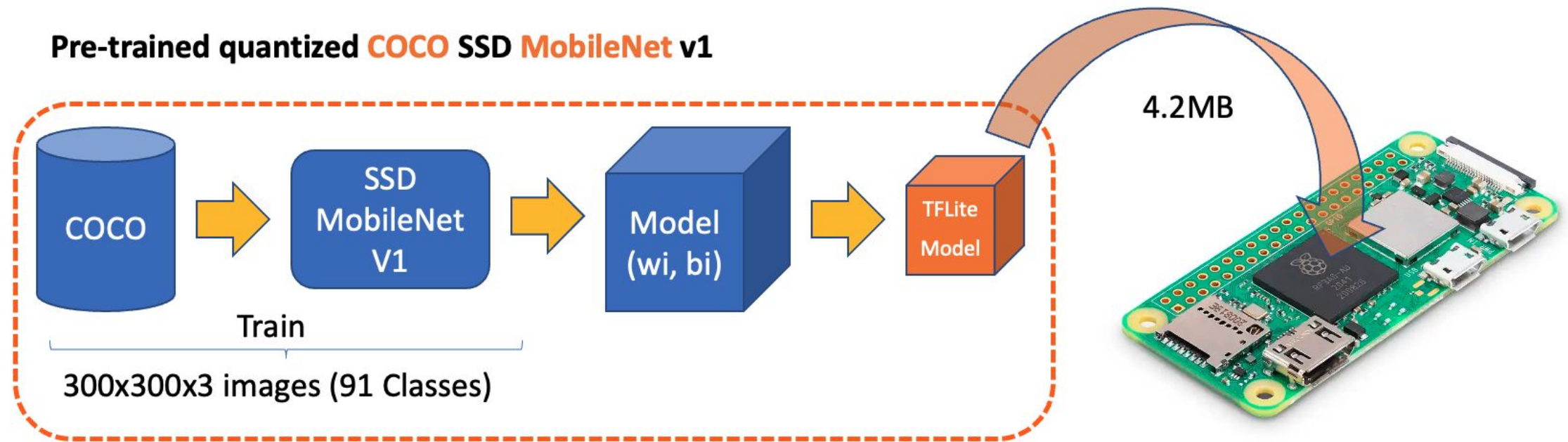


Detection
Model



Predict Labels
& Bounding Boxes

The SSD-MobileNet V1 model



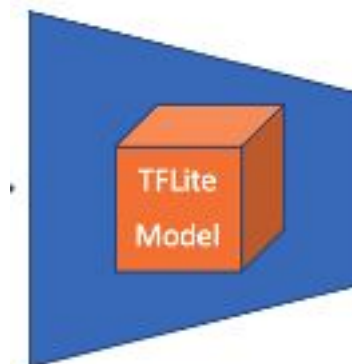
It outputs up **to ten detections per image**, including bounding boxes, class IDs, and confidence scores

Model Input

input_details

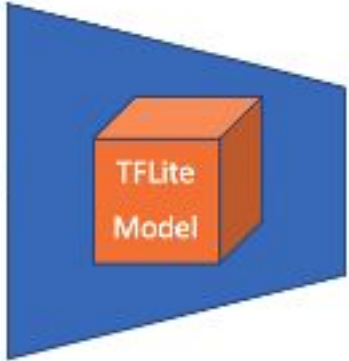
```
[{'name': 'normalized_input_image_tensor',  
  'index': 175,  
  'shape': array([ 1, 300, 300,  3], dtype=int32),  
  'shape_signature': array([ 1, 300, 300,  3], dtype=int32),  
  'dtype': numpy.uint8,  
  'quantization': (0.0078125, 128),  
  'quantization_parameters': {'scales': array([0.0078125], dtype=float32),  
    'zero_points': array([128], dtype=int32),  
    'quantized_dimension': 0},  
  'sparsity_parameters': {}}]
```

TFLite Interpreter



Model Output

TFLite Interpreter



- 1.Boxes** [1,10,4]: Location (normalized coordinates)
- 2.Scores** [1,10]: Confidence (0.0-1.0)
- 3.Classes** [1,10]: Class (COCO class IDs 0-90)
- 4.Count** [1]: How many valid detections (0-10)

output_details

- ```
[{'name': 'TFLite_Detection_PostProcess',
 'index': 167,
 'shape': array([1, 10, 4], dtype=int32),
 'shape_signature': array([1, 10, 4], dtype=int32),
 'dtype': numpy.float32,
 'quantization': (0.0, 0),
 'quantization_parameters': {'scales': array([], dtype=float32),
 'zero_points': array([], dtype=int32),
 'quantized_dimension': 0},
 'sparsity_parameters': {}},
```

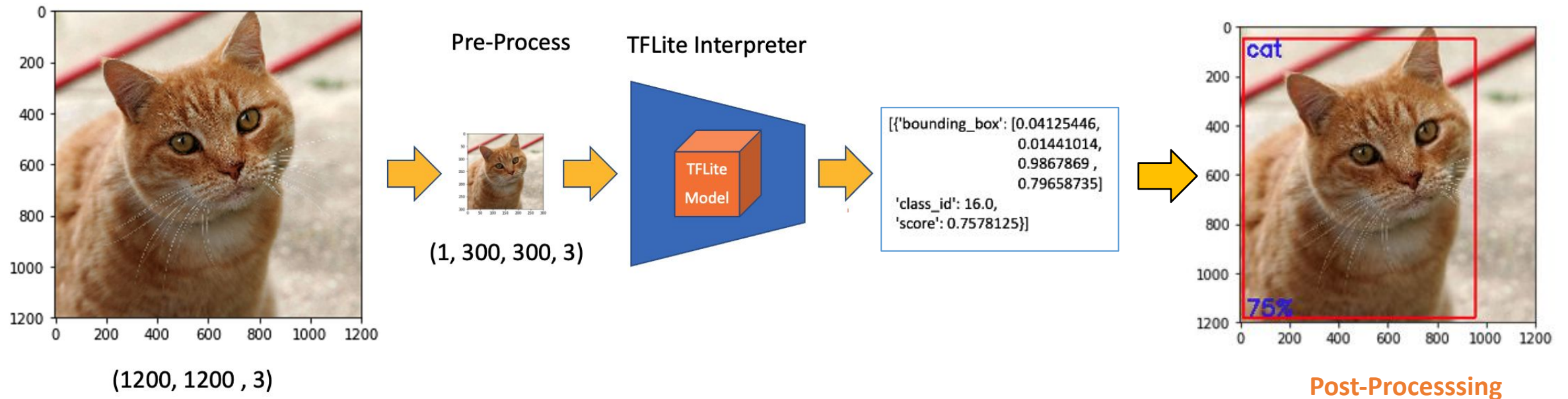
**1. Boxes**
- ```
{'name': 'TFLite_Detection_PostProcess:1',  
  'index': 168,  
  'shape': array([ 1, 10], dtype=int32),  
  'shape_signature': array([ 1, 10], dtype=int32),  
  'dtype': numpy.float32,  
  'quantization': (0.0, 0),  
  'quantization_parameters': {'scales': array([], dtype=float32),  
    'zero_points': array([], dtype=int32),  
    'quantized_dimension': 0},  
  'sparsity_parameters': {}},
```

2. Scores
- ```
{'name': 'TFLite_Detection_PostProcess:2',
 'index': 169,
 'shape': array([1, 10], dtype=int32),
 'shape_signature': array([1, 10], dtype=int32),
 'dtype': numpy.float32,
 'quantization': (0.0, 0),
 'quantization_parameters': {'scales': array([], dtype=float32),
 'zero_points': array([], dtype=int32),
 'quantized_dimension': 0},
 'sparsity_parameters': {}},
```

**3. Classes**
- ```
{'name': 'TFLite_Detection_PostProcess:3',  
  'index': 170,  
  'shape': array([1], dtype=int32),  
  'shape_signature': array([1], dtype=int32),  
  'dtype': numpy.float32,  
  'quantization': (0.0, 0),  
  'quantization_parameters': {'scales': array([], dtype=float32),  
    'zero_points': array([], dtype=int32),  
    'quantized_dimension': 0},  
  'sparsity_parameters': {}}
```

4. Count

Making **inferences** with the SSD-MobileNet V1



Input Image



Pre-Process



Detection Model



Predict Labels & Bounding Boxes

Bounding Boxes

bounding box:

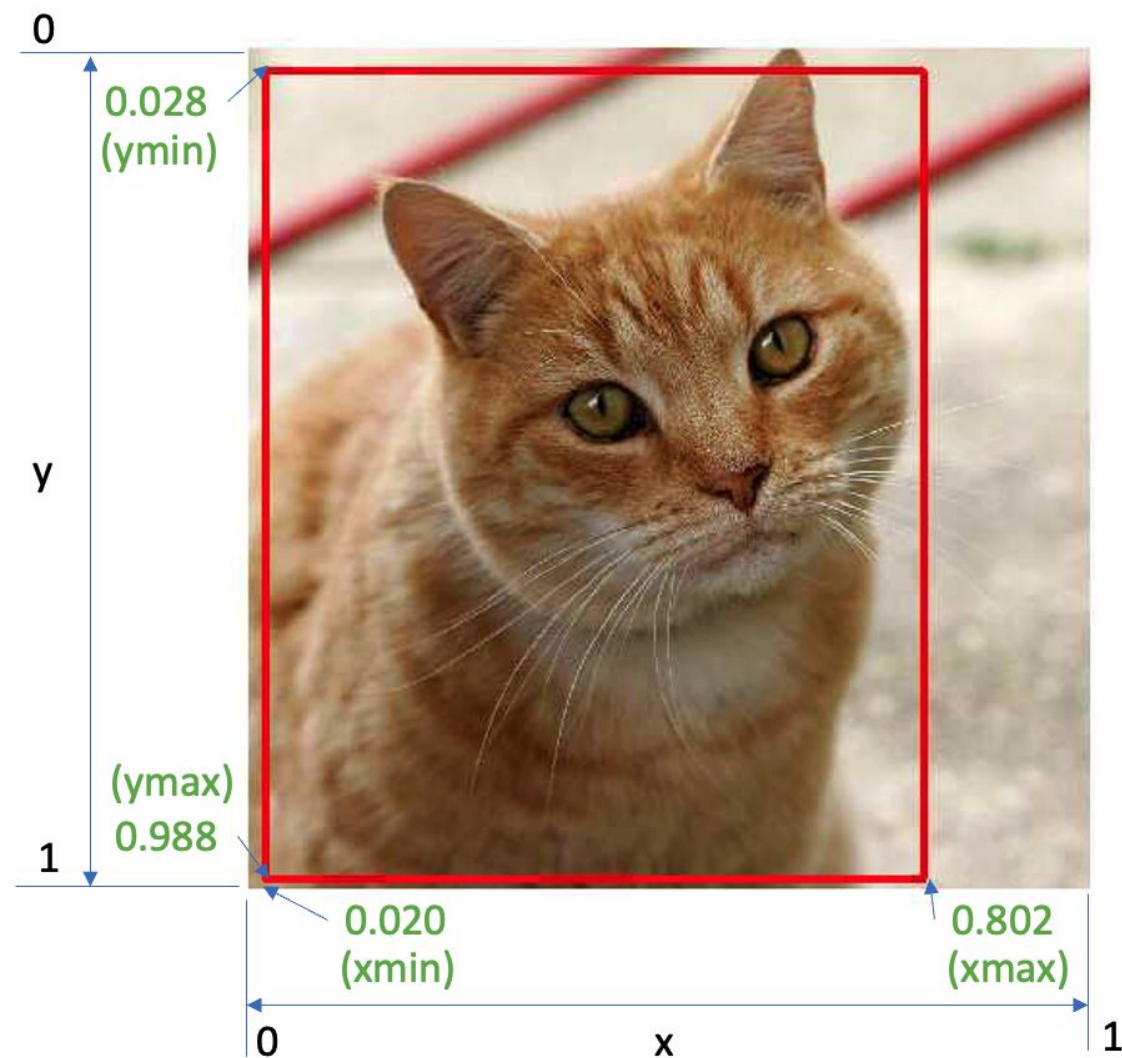
[0.028011084, 0.020121813, 0.9886069, 0.802299]

ymin

xmin,

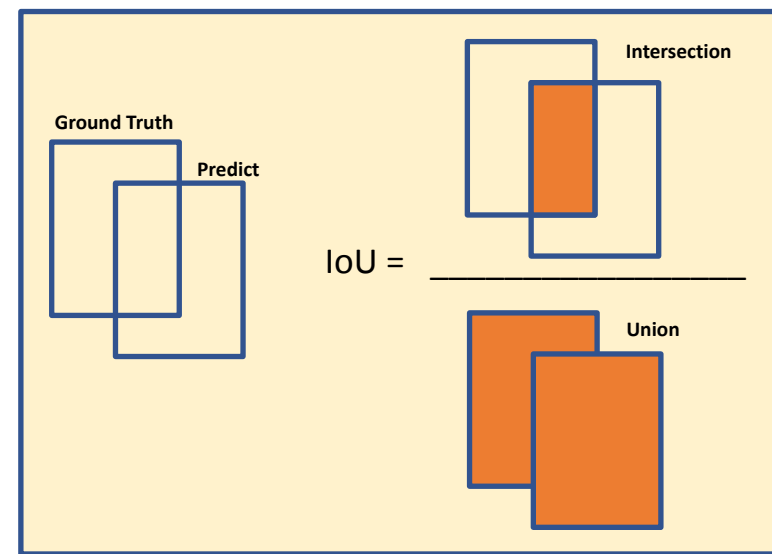
ymax

xmax

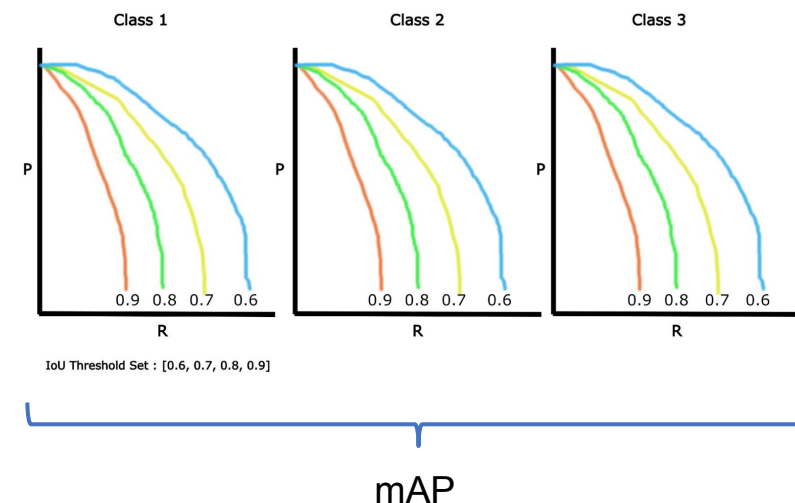


Evaluation Metrics

Intersection over Union (IoU) is a metric used to evaluate the **accuracy** of an object detector. It measures the overlap between two bounding boxes: the **Ground Truth** box (the manually labeled correct box) and the **Predicted** box (the box generated by the object detection model). The IoU value is calculated by dividing the area of the **Intersection** (the overlapping area) by the area of the **Union** (the total area covered by both boxes). A higher IoU value indicates a better prediction.



Mean Average Precision (mAP) is a widely used metric for evaluating the **performance** of object detection models. It provides a single number that reflects a model's ability to accurately both **classify** and **localize** objects. The "mean" in mAP refers to the average taken over all object classes in the dataset. The "average precision" (AP) is calculated for each class, and then these AP values are averaged to get the final mAP score. A high mAP score indicates that the model is excellent at identifying all objects and placing a tight-fitting, accurate bounding box around them.



Frames Per Second (FPS) for real-time performance

TFLite Runtime and Model Setup

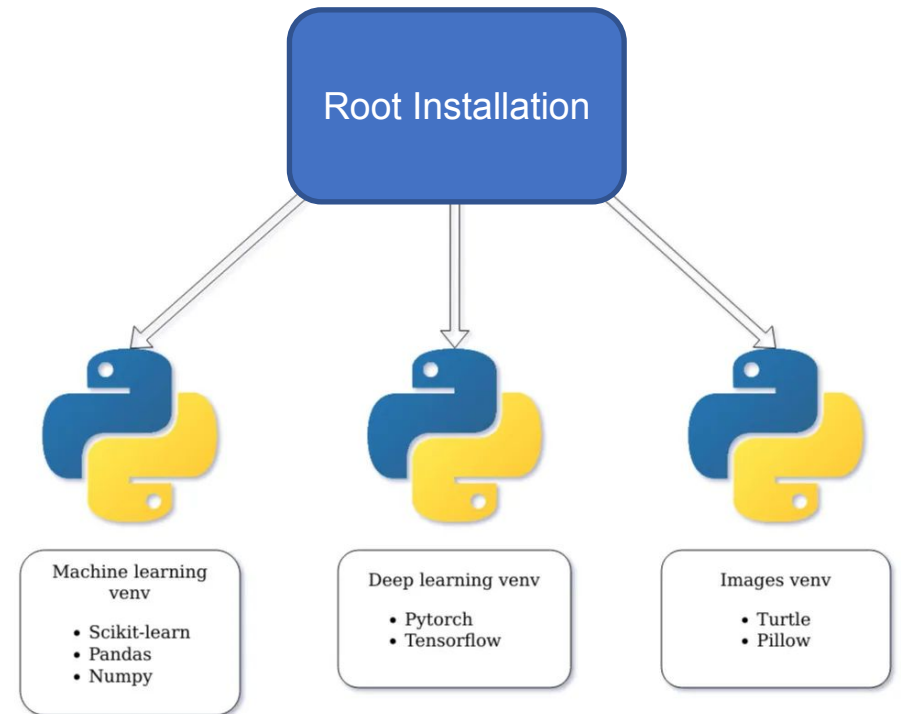
Setting up a Virtual Environment

Activate the environment:

```
source ~/tflite_env/bin/activate
```

To **exit** the virtual environment, use:

```
deactivate
```



Creating a **working directory** & get the **model**

```
marcelo_rovai — mjrovai@raspi-zero: ~/Documents/TFLITE/OBJ_DETECT/models — ssh mjrovai@192.168.4.210 — 81x10
(tflite_env) mjrovai@raspi-zero:~ $
(tflite_env) mjrovai@raspi-zero:~ $ cd Documents/TFLITE/
(tflite_env) mjrovai@raspi-zero:~/Documents/TFLITE $ mkdir OBJ_DETECT
(tflite_env) mjrovai@raspi-zero:~/Documents/TFLITE $ cd OBJ_DETECT
(tflite_env) mjrovai@raspi-zero:~/Documents/TFLITE/OBJ_DETECT $ mkdir models
(tflite_env) mjrovai@raspi-zero:~/Documents/TFLITE/OBJ_DETECT $ mkdir images
(tflite_env) mjrovai@raspi-zero:~/Documents/TFLITE/OBJ_DETECT $ ls
images  models
(tflite_env) mjrovai@raspi-zero:~/Documents/TFLITE/OBJ_DETECT $ cd models
(tflite_env) mjrovai@raspi-zero:~/Documents/TFLITE/OBJ_DETECT/models $
```

```
wget
https://github.com/Mjrovai/EdgeML-with-Raspberry-Pi/raw/refs/heads/main/OBJ_DETECT/models/ssd-mobilenet-v1-tflite-default-v1.tar.gz
```

```
tar -xzf ssd-mobilenet-v1-tflite-default-v1.tar.gz
mv 1.tflite ssd-mobilenet-v1-tflite-default-v1.tflite
```

```
wget
https://raw.githubusercontent.com/Mjrovai/EdgeML-with-Raspberry-Pi/refs/heads/main/OBJ_DETECT/models/coco_labels.txt
```

Running up Jupyter Notebook

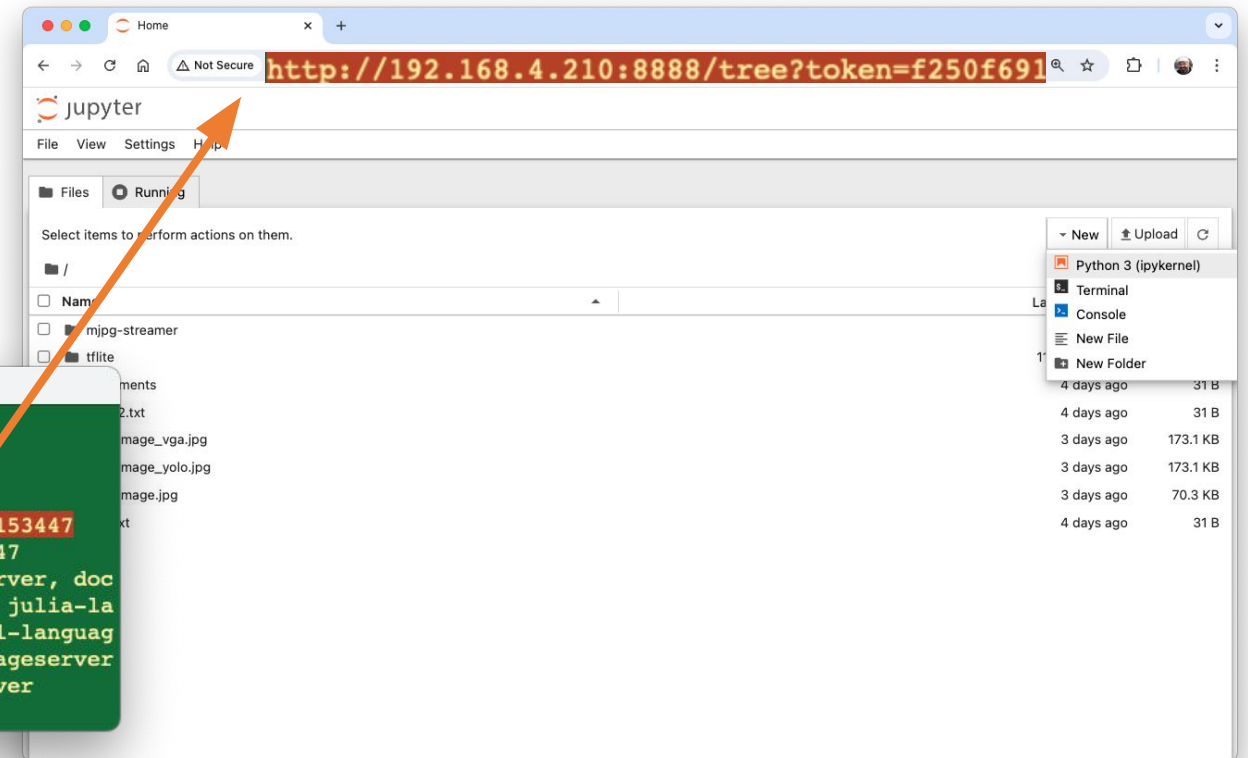


```
jupyter notebook --ip=[YOUR IP ADDRESS] --no-browser
```

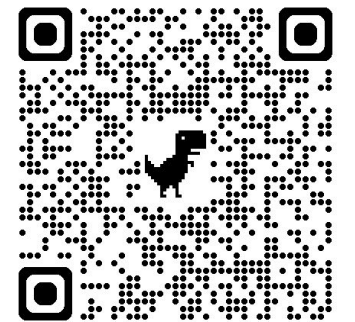
Copy the **Raspberry Pi's IP address** and the provided **token** in a web browser.

```
marcelo_rovai — mjrovai@raspi-zero: ~ — ssh mjrovai@192.168.4.210 — 96x12

To access the server, open this file in a browser:
  file:///home/mjrovai/.local/share/jupyter/runtime/jpserver-7399-open.html
Or copy and paste one of these URLs:
  http://192.168.4.210:8888/tree?token=f250f69117df5adf9d1aff01dd3ede657cb47b0ba8153447
  http://127.0.0.1:8888/tree?token=f250f69117df5adf9d1aff01dd3ede657cb47b0ba8153447
[I 2024-08-23 19:40:59.979 ServerApp] Skipped non-installed server(s): bash-language-server, doc
kerfile-language-server-nodejs, javascript-typescript-langserver, jedi-language-server, julia-la
nguage-server, pyright, python-language-server, python-lsp-server, r-languageserver, sql-languag
e-server, texlab, typescript-language-server, unified-language-server, vscode-css-languageserver
-bin, vscode-html-languageserver-bin, vscode-json-languageserver-bin, yaml-language-server
```



Raspberry Pi Inference: SSD_MobileNetV1.ipynb



Live Object Detection

object_detection_app.py

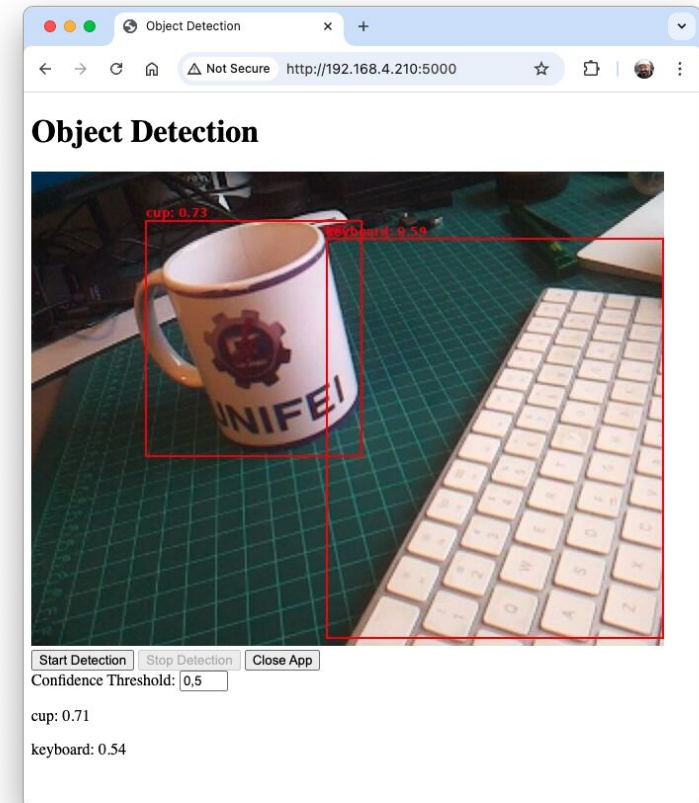
* Running on <http://192.168.4.210:5000>
Press CTRL+C to quit

```
marcelo_rovai — mjrovai@raspi-zero: ~/Documents/TFLITE/OBJ_DETECT — ssh mjrovai@192.168.4.210 — 98x22
GNU nano 7.2 object_detection_app.py
picam2 = None
frame = None
frame_lock = threading.Lock()
is_detecting = False
confidence_threshold = 0.5
model_path = "./models/ssd-mobilenet-v1-tflite-default-v1.tflite"
labels_path = "./models/coco_labels.txt"
interpreter = None
detection_queue = Queue(maxsize=1)
latest_detections = []
detections_lock = threading.Lock()

def load_labels(path):
    with open(path, 'r') as f:
        return {i: line.strip() for i, line in enumerate(f.readlines())}

labels = load_labels(labels_path)

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line
```



Questions?



Prof. Marcelo J. Rovai

rovai@unifei.edu.br



UNIFEI