CHAPTER 8 · IESTI05 EDGE AI ENGINEERING

Object Detection on the UNO Q

What is in the image, and where

Edge Impulse · FOMO · YOLO-Pro · App Lab Bricks · Bridge

Marcelo Rovai · UNIFEI



THE ARC SO FAR

From one label per image to one label per object

Generative AI
Chapters 2 to 6
one small language model
text · eyes · sensors · tools

→

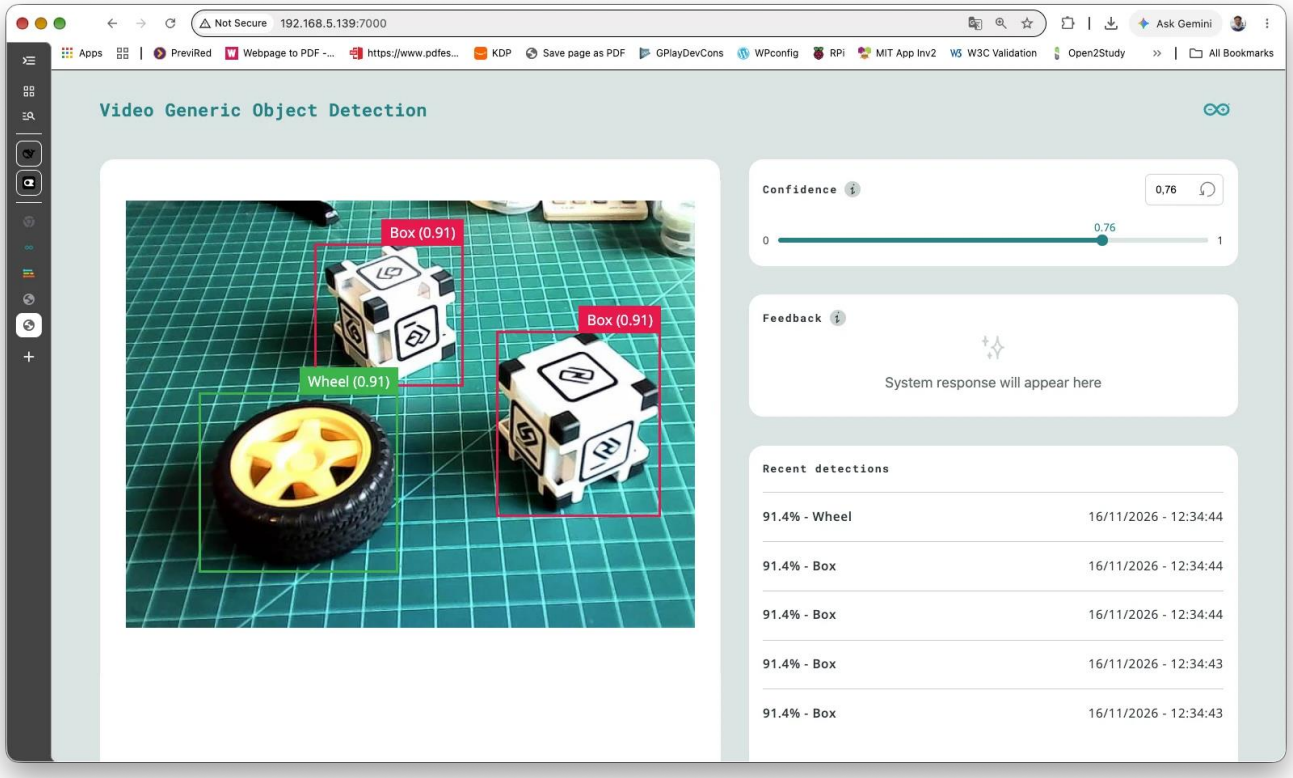
Image classification
Chapter 7
one label for the
whole image

→

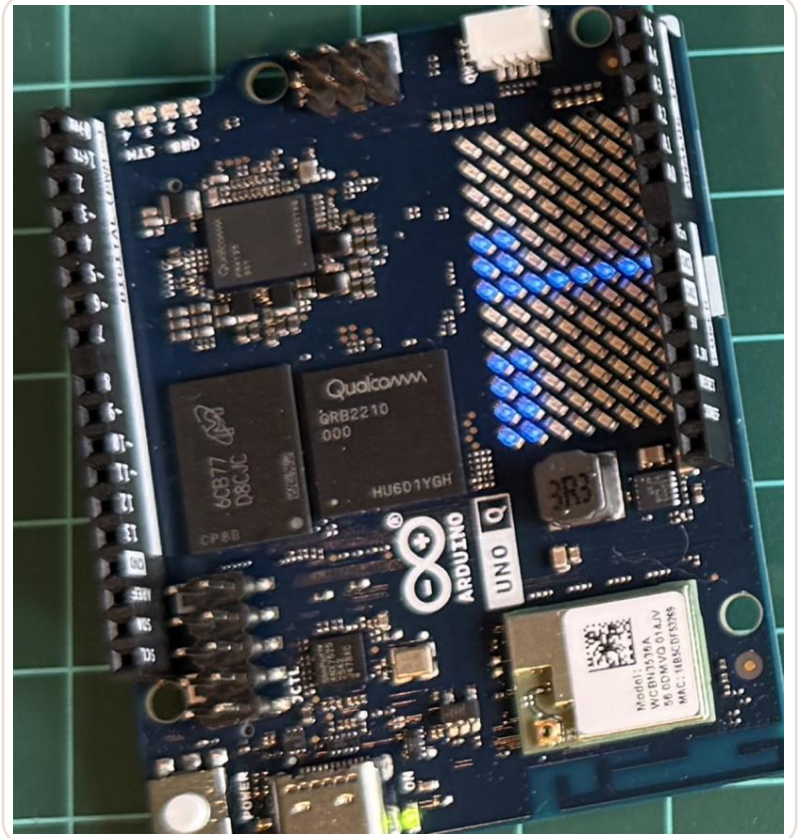
Object detection
Chapter 8
a label and a location
for every object

> Same board, same Edge Impulse workflow, same App Lab Bricks. The model's answer is what changes.

Count boxes and wheels, live, on the LED matrix



Live detection in the browser



The counts on the MCU's matrix

THE ROUTE

Eight stages, one live counter at the end

1	**Detection**	Classification versus detection, IoU, mAP, NMS, FOMO and YOLO.
2	**Pre-installed**	The two detectors that ship with the board, on images and on camera.
3	**Dataset**	One hundred fifty-three labeled images of boxes and wheels.
4	**Training**	FOMO and YOLO-Pro, side by side in one Edge Impulse project.
5	**Path A**	Our model on static images, and loading the .eim file directly.
6	**Path B**	The same model on the live camera, without changing a line of Python.
7	**Counting**	Counts per class, sent over Bridge to the LED matrix.
8	**Performance**	Latency and frame rate, measured on the board.

STAGE 1 OF 8

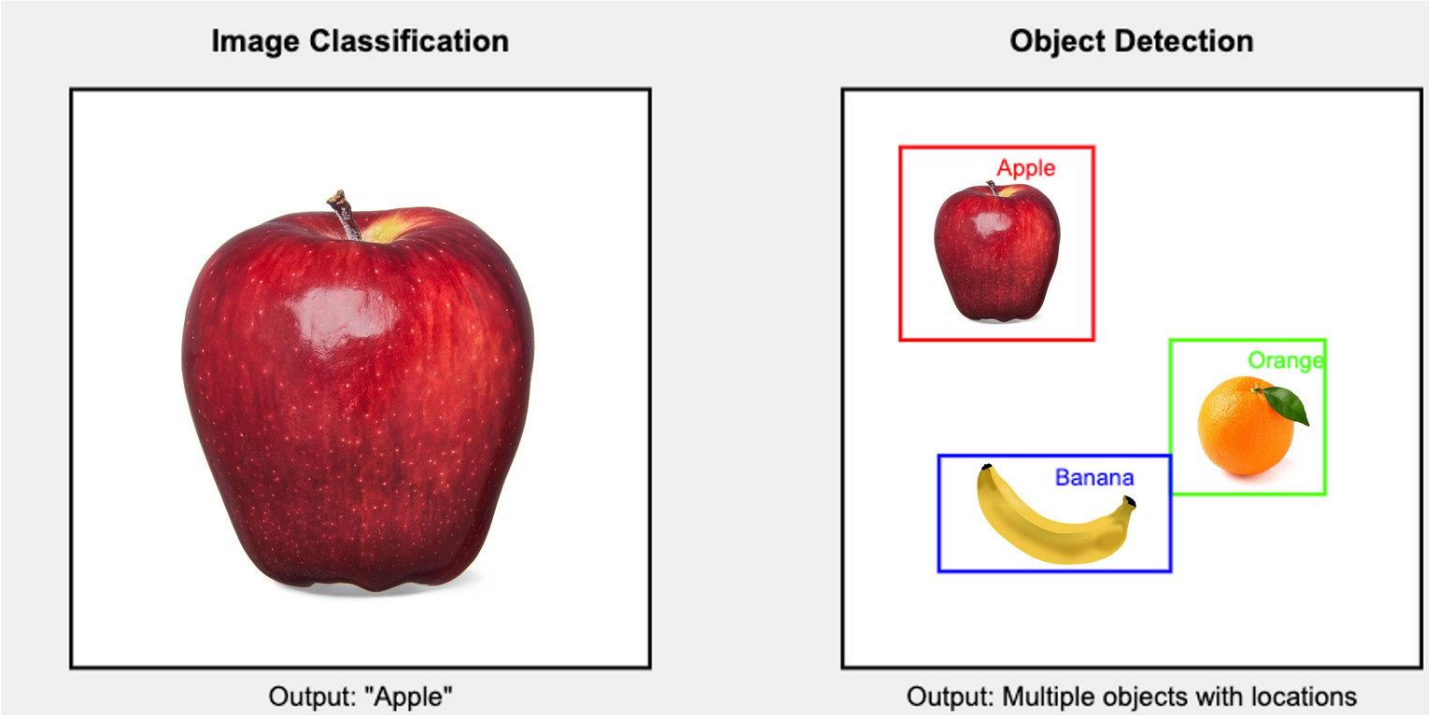
What detection adds

1

One label per image becomes a label, a score, and a box per object.

CLASSIFICATION VERSUS DETECTION

One answer per image, or one per object



A detector does two jobs at once: localization (where) and classification (what), per region.

JUDGING A DETECTOR

Three numbers matter

IOU

Overlap

Intersection over union of the predicted and the labeled box. A detection usually counts when IoU is at least 0.5.

MAP

Accuracy

Precision averaged over recall levels and classes. mAP@0.5 uses the 0.5 IoU rule.

FPS

Speed

Frames per second, or latency per frame. On the edge, three percent more accuracy at three times the latency is often the wrong trade.

NMS keeps the most confident of overlapping boxes. On the UNO Q it is already inside the .eim model.

TWO SINGLE-STAGE FAMILIES

FOMO finds centers. YOLO fits boxes.

FOMO

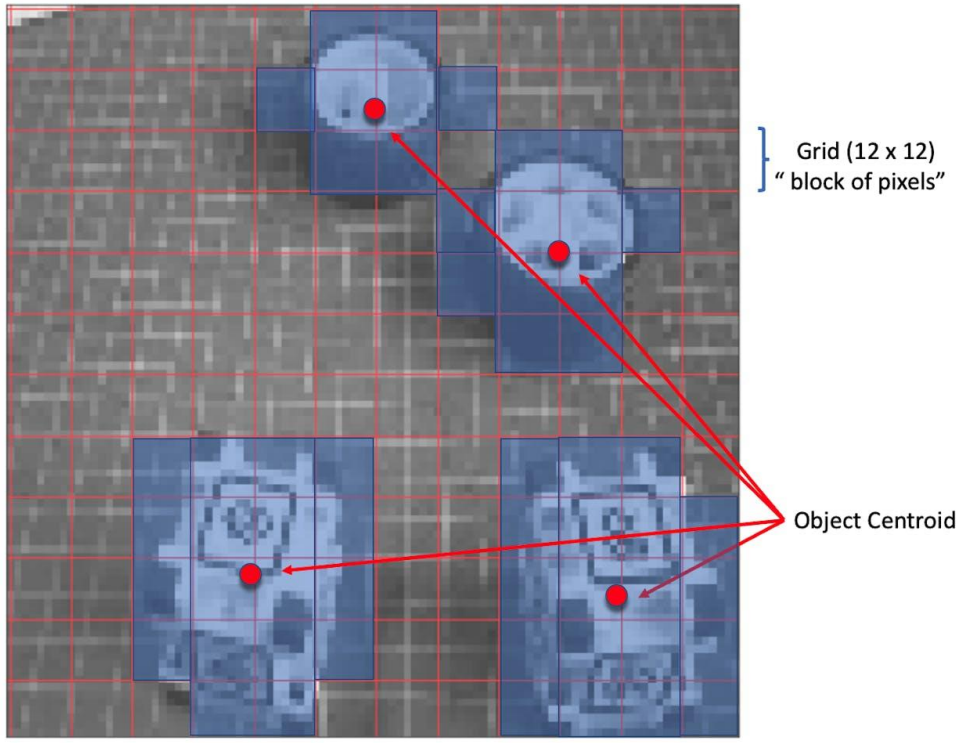
A centroid per object

Classifies each cell of a coarse grid. Very small and fast. Reports where an object is, not how big it is.

YOLO

A box per object

Predicts boxes and classes in one pass. Gives the object's extent, at a higher cost. The pre-installed model is a YOLOX.



FOMO: grid cells merge into one centroid

STAGE 2 OF 8

The pre-installed detectors

2

YOLOX-Nano, trained on COCO's eighty classes, on images and on camera.

TWO EXAMPLES, ONE MODEL

YOLOX-Nano, trained on 80 COCO classes

App ID	Bricks	Input
`examples:inspirational/object-detection`	object_detection + web_ui	Static image, uploaded in the browser
`examples:inspirational/video-generic-object-detection`	video_object_detection + web_ui	Live USB webcam

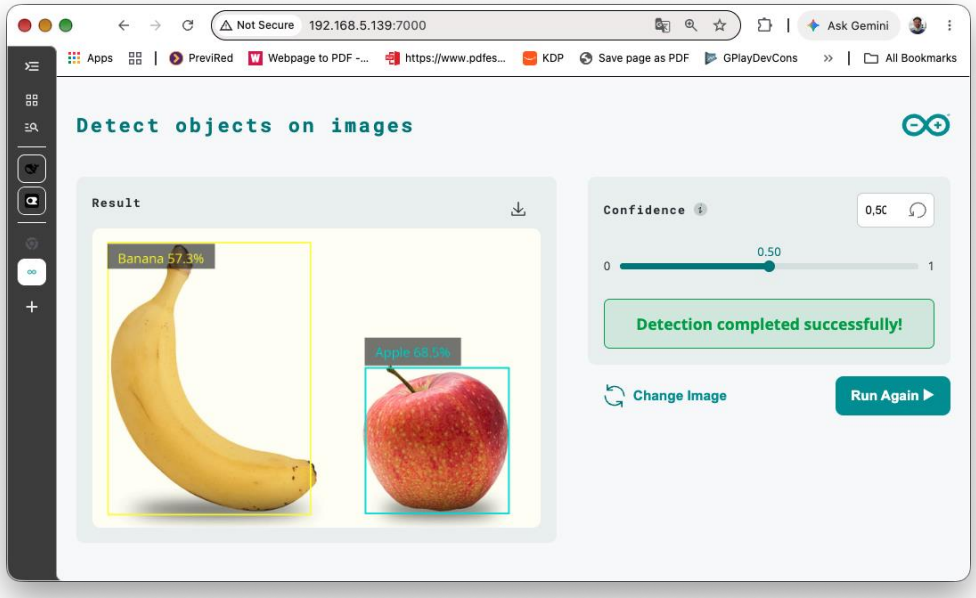
> Check the IDs on your board: arduino-app-cli app list --examples | grep -i detect

RUN IT ON THE BOARD

Static images first. No webcam needed.

```
arduino-app-cli app start \
  examples:inspirational/object-detection
# then open http://<UNO_Q_IP>:7000
```

Move the threshold slider: at 0.7 only the sure objects remain; at 0.2 false positives appear.



Boxes, labels, and scores, drawn by the Brick

THE RESULT FORMAT

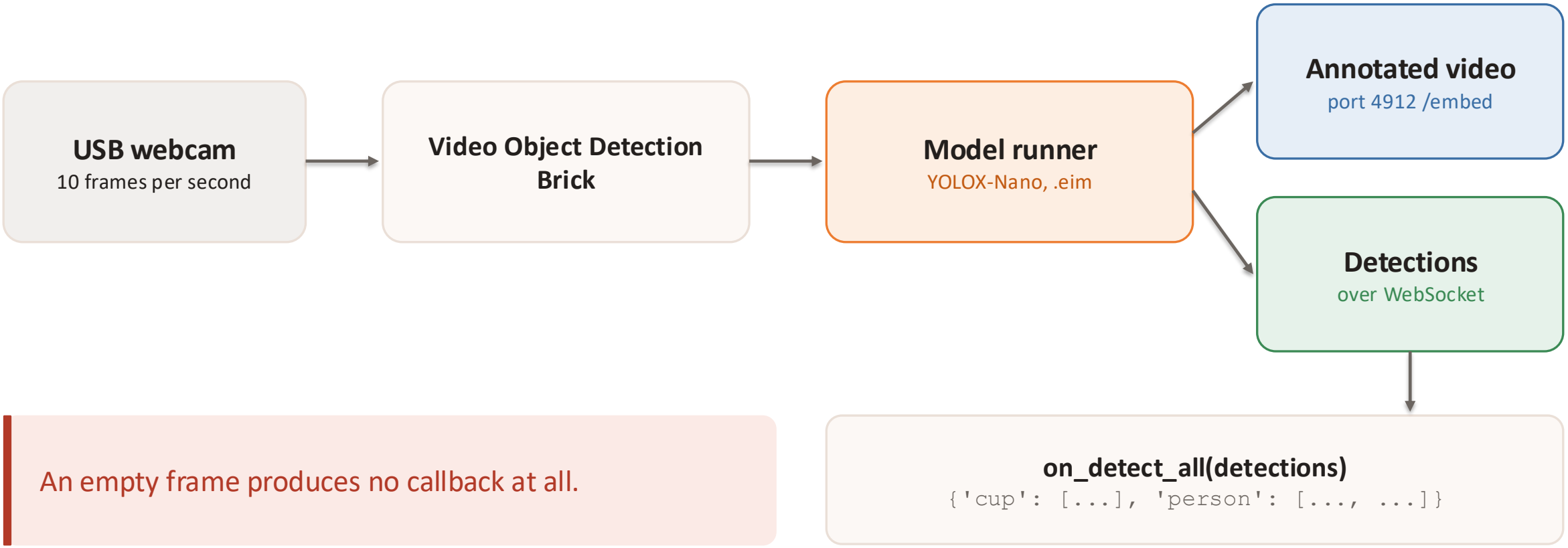
A list of objects, each with a class, a score, and a box

```
{"detection": [
  {"class_name": "cup",
   "confidence": "87.50",
   "bounding_box_xyxy": [41, 112, 98, 180]}
]}
```

> Watch the scales: the result's confidence is a string from 0 to 100; the parameter you pass to detect() is a float from 0 to 1.

THE LIVE EXAMPLE

Python never touches a frame



An empty frame produces no callback at all.

STAGE 3 OF 8

The dataset

3

Boxes and wheels, on a factory bench. Every object needs a box and a label.

THE PROJECT

Sort and count wheels and special boxes



Any number of boxes and wheels per image, or none at all. Background is implicit.

OPTION A: THE READY-MADE DATASET

Labeled once, for the Raspberry Pi book

153
images, labeled with bounding boxes

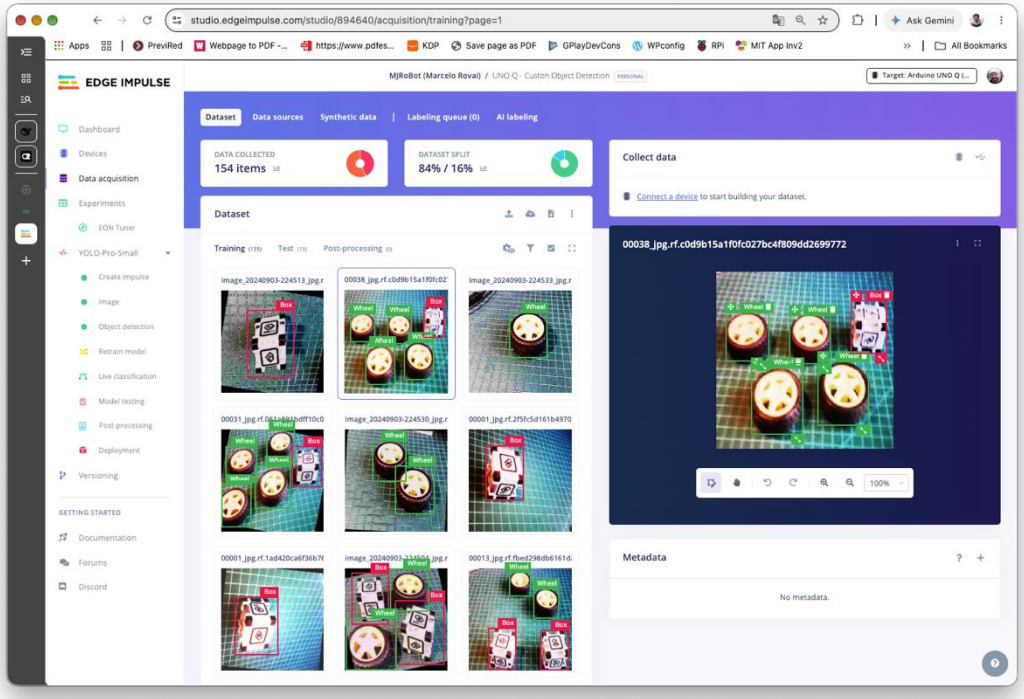
320×320
pixels, already augmented

2
classes: box and wheel

Roboflow Universe: box-versus-wheel-auto-dataset, exported in YOLOv8 format.

INTO EDGE IMPULSE

Upload the YOLO folders, then check the label names



If objects show up as 0 and 1, rename them now. Your Python code will match these strings.

STAGE 4 OF 8

Two models, one project

4

FOMO for speed, YOLO-Pro for fitted boxes. Edge Impulse Experiments holds both.

TWO IMPULSES

Same data, very different models

IMPULSE 1

FOMO MobileNetV2 0.35

160 × 160 input · 60 epochs

float32: the int8 version dropped below 0.70 F1

F1 = 0.82

IMPULSE 2

YOLO-Pro Small

320 × 320 input · 30 epochs

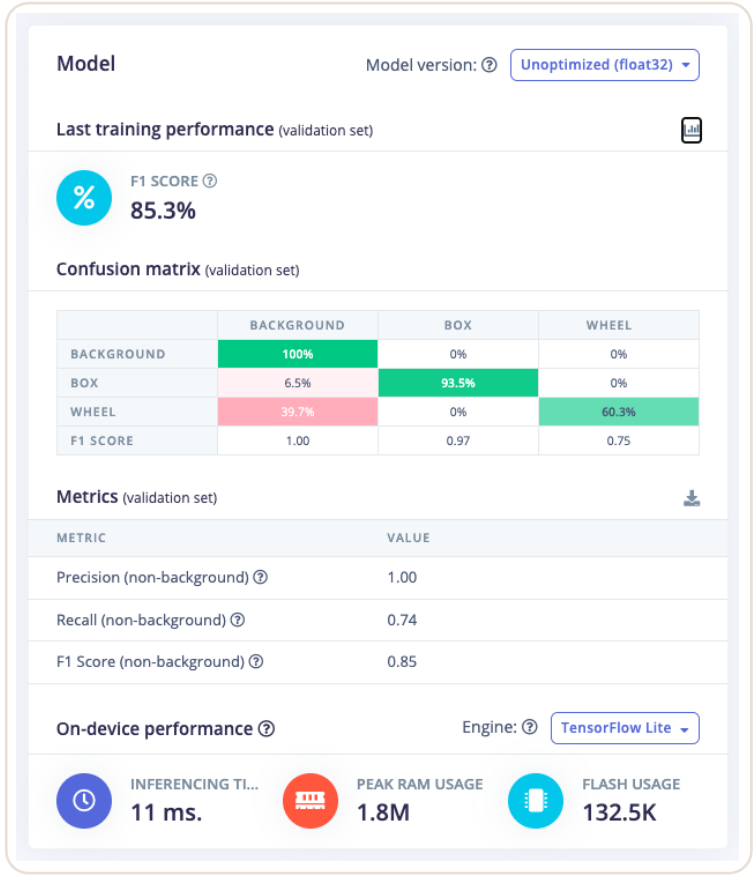
6.9 million parameters · int8

mAP@0.5 = 0.99

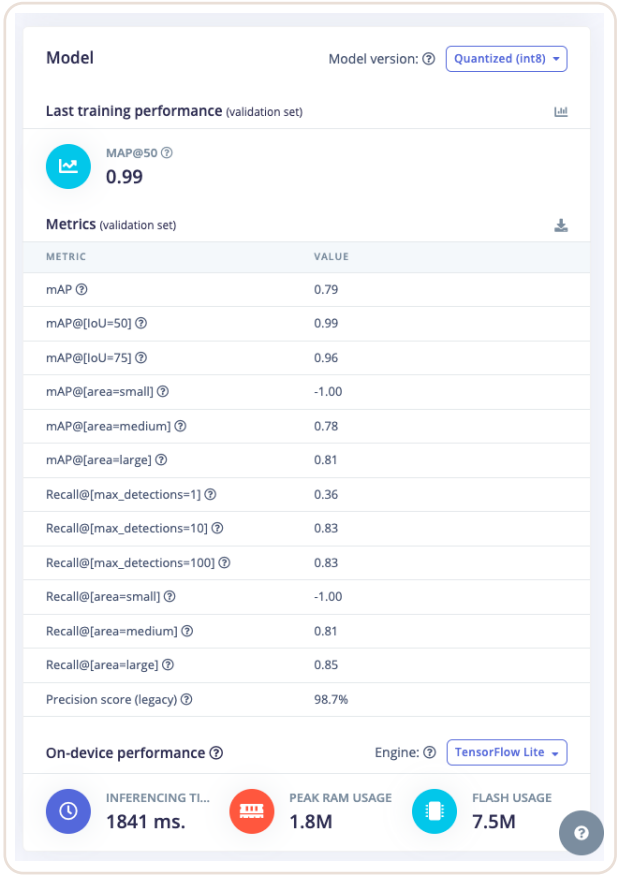
F1 and mAP are not comparable: mAP also requires the box to overlap the label. Judge by the counts.

TRAINING RESULTS

FOMO reports F1. YOLO reports mAP.



FOMO: F1 score and confusion matrix



YOLO-Pro Small: mAP and on-device estimate

STAGE 5 OF 8

Path A: static images

5

Our model in the pre-installed app, and a shorter way to load it.

COPY THE EXAMPLE

Our own app, with its own name

```
cp -r /var/lib/arduino-app-cli/examples/\
inspirational/common/object-detection/ \
  ~/ArduinoApps/my-detector-bricks

cd ~/ArduinoApps/my-detector-bricks
sed -i 's/^name: .*/name: My Detector/' app.yaml
```

App Lab shows the name: field from app.yaml, not the folder name. Rename it, or two apps look identical.

TWO WAYS TO LOAD YOUR MODEL

Through App Lab, or straight from the .eim file

APP LAB

Train new AI model

Log in from the Brick, deploy for Arduino UNO Q, and App Lab downloads the model and writes a model: line in app.yaml.

DIRECT

Point the Brick at the file

Download the .eim from Edge Impulse, copy it to the board, and set one variable in app.yaml. No App Lab round trip.

LOAD THE .EIM FILE DIRECTLY

Copy, make it executable, and point the Brick at it

```
# on your computer
scp box-wheel-yolo.eim arduino@<UNO_Q_IP>:~
# on the board
mkdir -p ~/.arduino-bricks/ei-models
mv ~/box-wheel-yolo.eim ~/.arduino-bricks/ei-models/
chmod +x ~/.arduino-bricks/ei-models/box-wheel-yolo.eim
```

APP.YAML

One variable selects the model

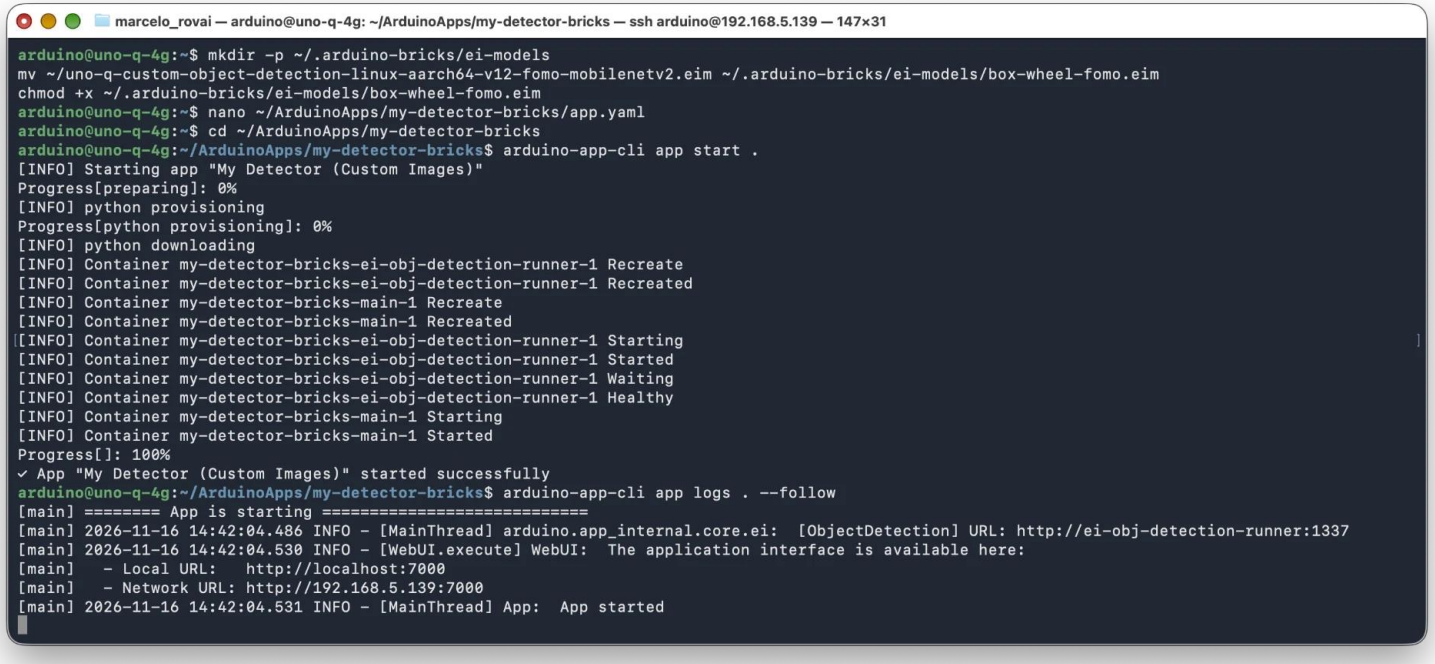
```
name: My Detector
bricks:
- arduino:web_ui: {}
- arduino:object_detection:
    variables:
      EI_OBJ_DETECTION_MODEL: \
        /home/arduino/.arduino-bricks/\
        ei-models/box-wheel-yolo.eim
```

Delete any model: line. With it, the runner cannot see ei-models/ and the app fails with ENOENT.

For the live camera Brick, the variable is EI_V_OBJ_DETECTION_MODEL.

IT RUNS

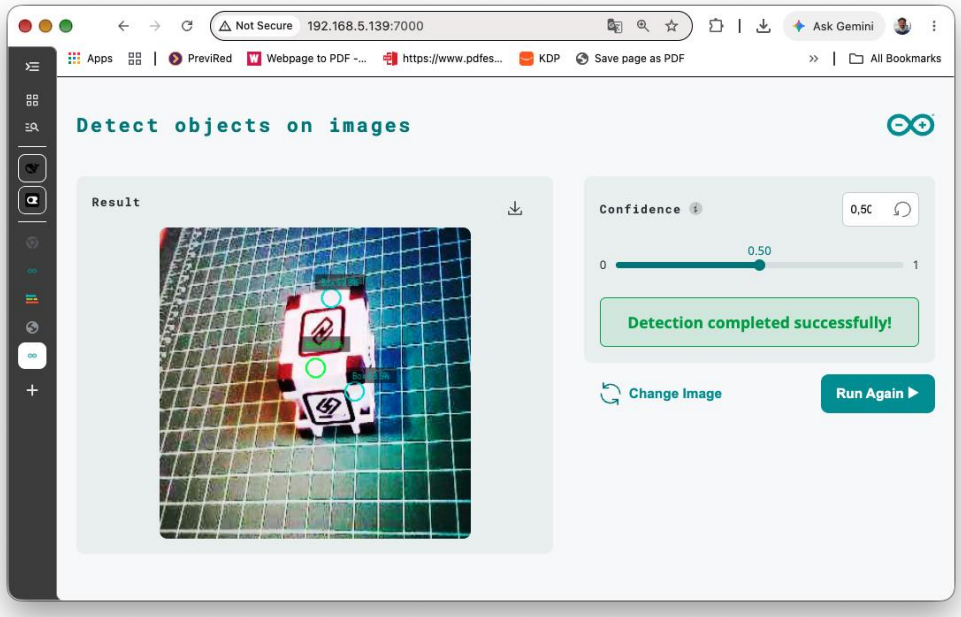
Our labels, from our file



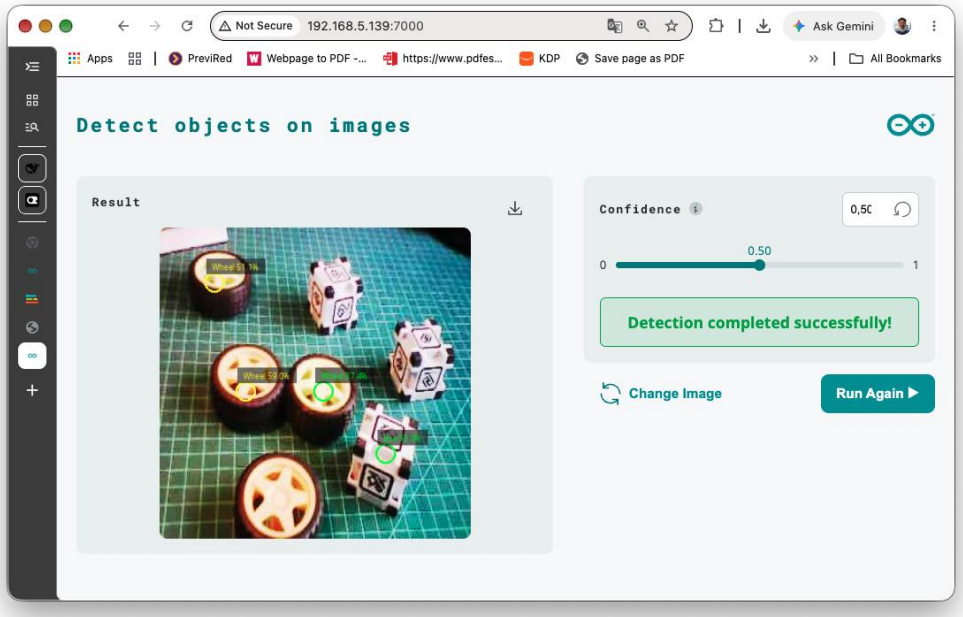
If the runner is unhealthy, read its own log: docker logs --tail 20 <app>-ei-obj-detection-runner-1

FOMO ON THE BOARD

Fast, but it miscounts



One large box becomes several centroids

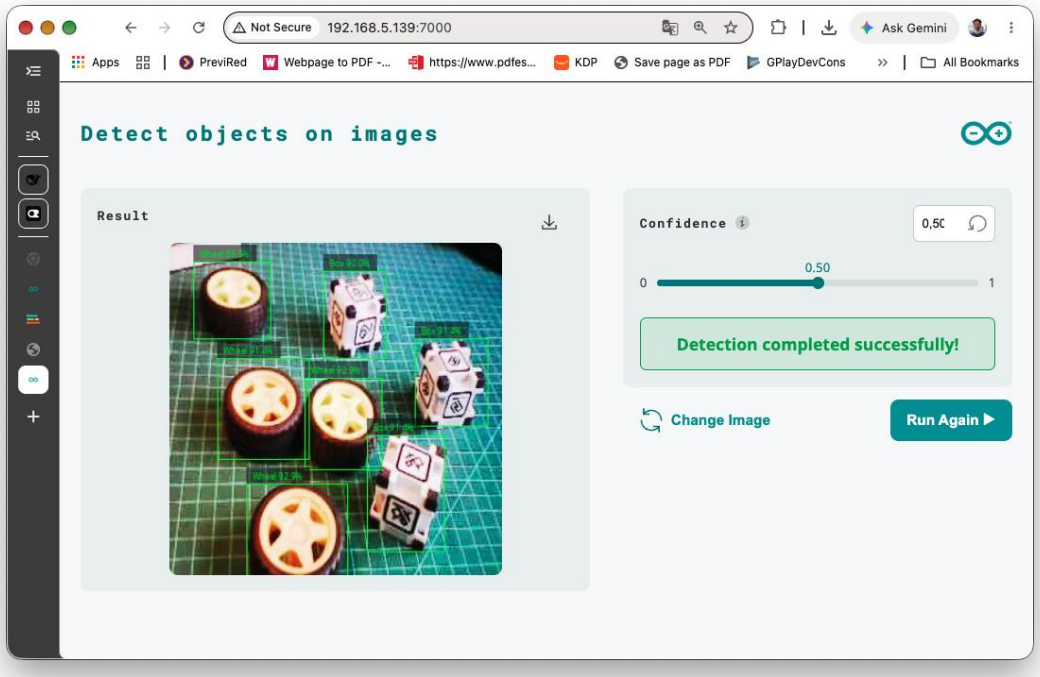


Crowded scene: objects merge or vanish

FOMO counted 2 of 6 test images correctly.

YOLO-PRO SMALL ON THE BOARD

Fitted rectangles, every count right



Four wheels and three boxes: 6 of 6 test images counted correctly, at about 331 ms per image.

ESTIMATE VERSUS BOARD

Edge Impulse's estimates were wrong in both directions

11 → 160 ms

FOMO: estimated versus measured

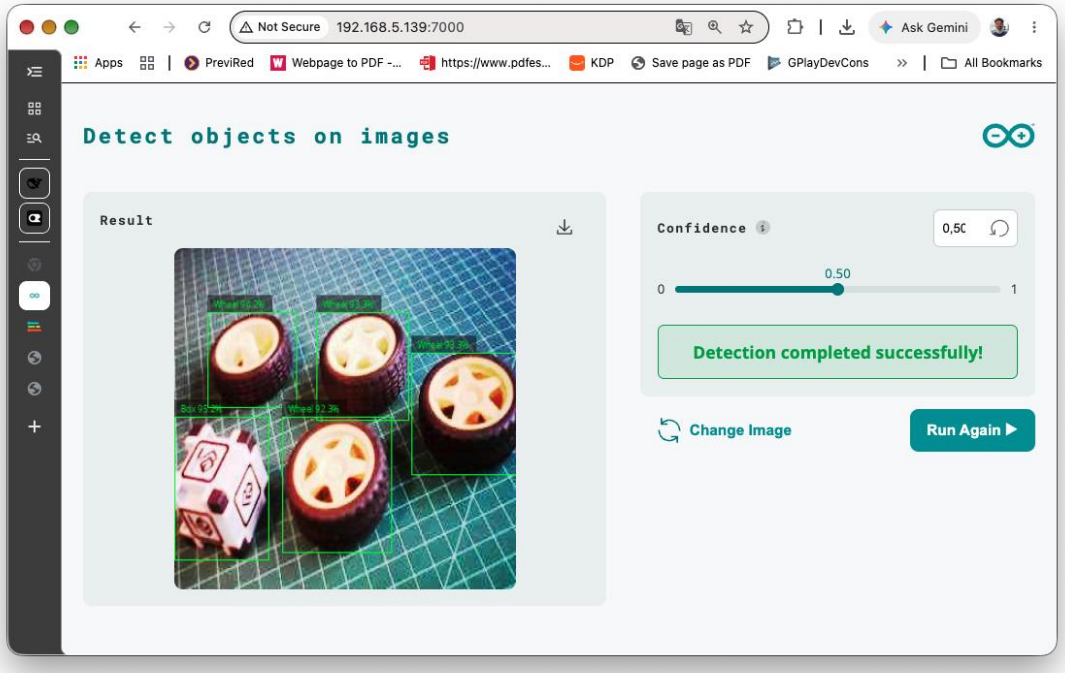
1,841 → 331 ms

YOLO-Pro Small: estimated versus measured

> The time around detect() includes moving the image to the runner. Always measure on the board.

A SMALLER YOLO

YOLO-Pro Nano: same counts, more speed



2.4 million parameters, int8: 6 of 6 correct, about 221 ms per image.

STAGE 6 OF 8

Path B: the live camera

6

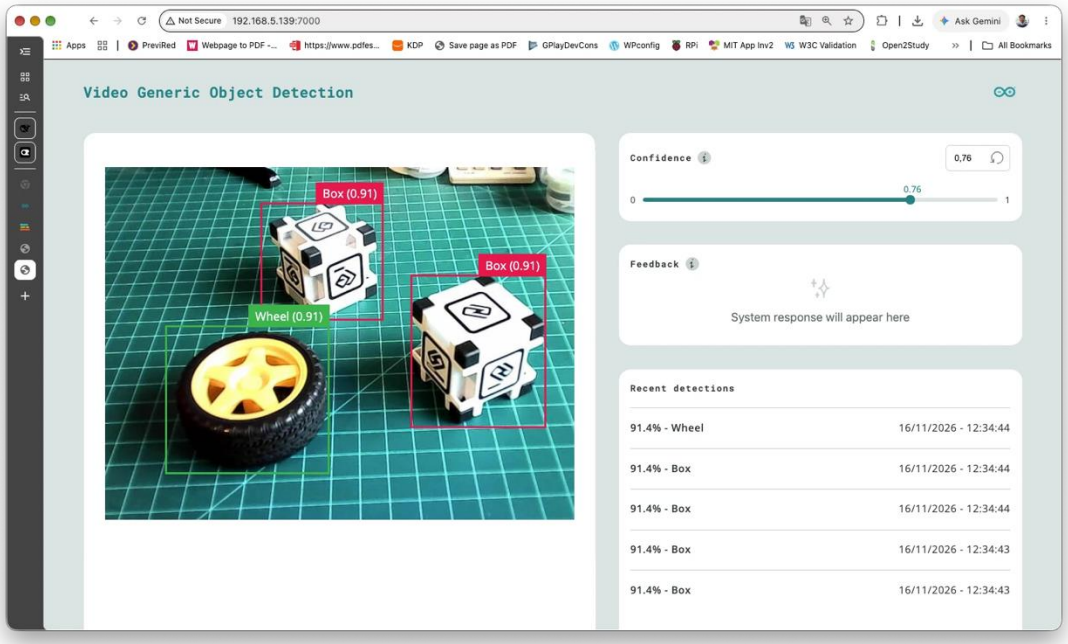
Same .eim file, the video Brick, and no change to the Python code.

LIVE CAMERA

Same file, the video Brick

```
- arduino:video_object_detection:
    variables:
      EI_V_OBJ_DETECTION_MODEL: \
        .../box-wheel-yolo-nano.eim
- arduino:web_ui: {}
```

The example's main.py is generic: it runs our model unchanged.



Box and Wheel, live

STAGE 7 OF 8

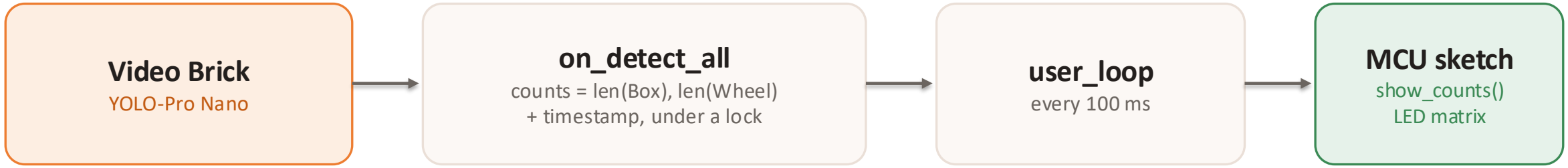
Counting, on the MCU

7

Counts per class in Python. Blocks on the LED matrix, drawn by the sketch.

ARCHITECTURE

Two threads, one Bridge call



The callback only records. It runs on worker threads, several times a second, so direct Bridge calls could overlap.

The loop is the only code that talks to the MCU, and only when the counts change.

THE EMPTY-FRAME TRAP

Silence has to mean zero

THE PROBLEM

No callback, no update

The Brick is silent on an empty frame. Remove every object and the matrix would freeze on the last count.

THE FIX

A one-second timeout

No detections for one second means zero objects. That is about three times the gap between frames, for all our models.

THE PYTHON SIDE

Record in the callback, act in the loop

```
def on_all_detections(detections):
    counts = (len(detections.get('Box', [])),
              len(detections.get('Wheel', [])))
    with state_lock:
        latest_counts, last_seen = counts, time.monotonic()
def loop():
    if time.monotonic() - last_seen > EMPTY_AFTER_SEC:
        counts = (0, 0)
    if counts != shown_counts:
        Bridge.call('show_counts', *counts)
```

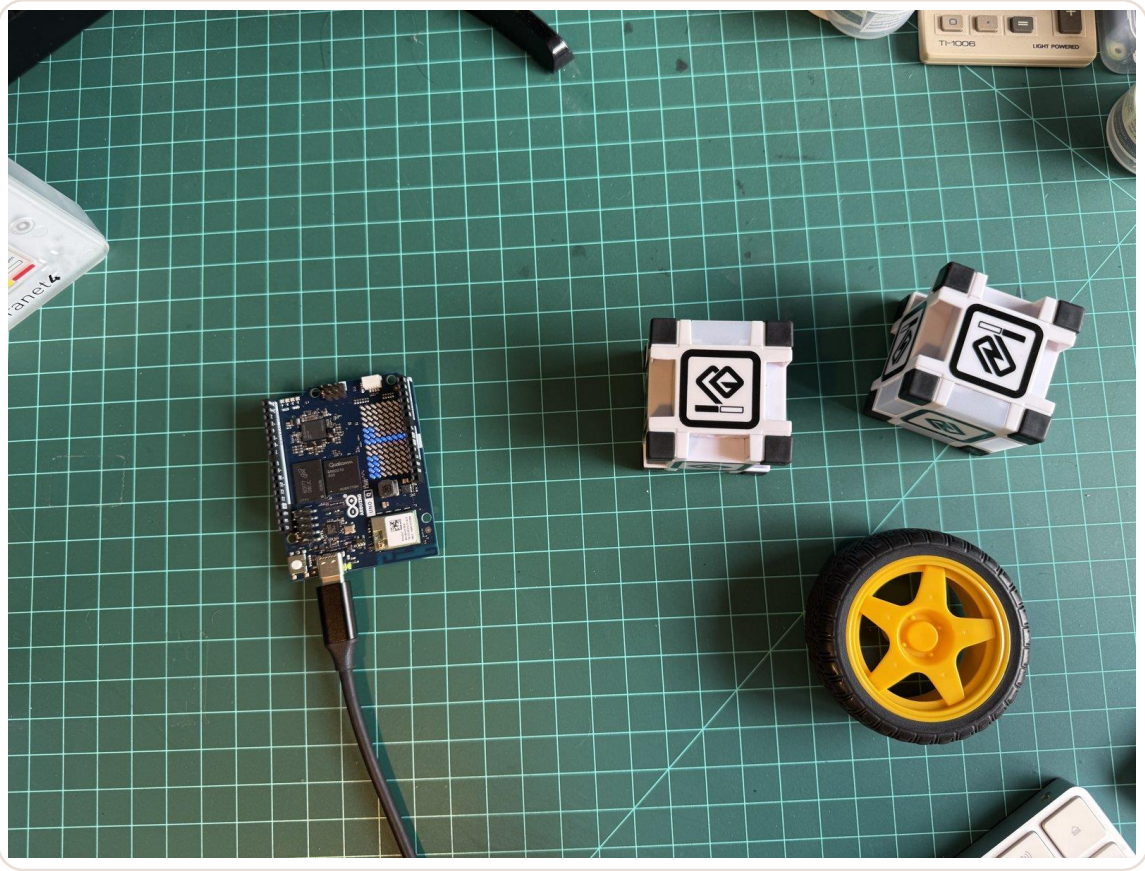
THE SKETCH

A 2×2 block per object, a divider in the middle

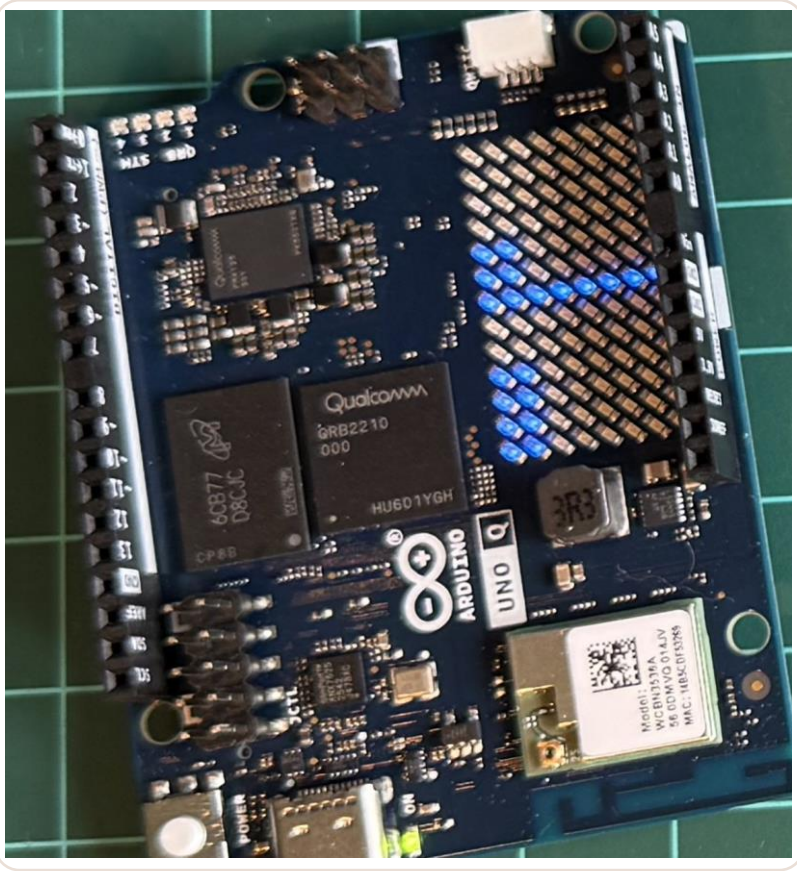
```
void show_counts(int boxes, int wheels) {
  memset(frame, 0, sizeof(frame));
  for (int r = 0; r < 8; r++) frame[r][6] = 1;
  for (int k = 0; k < boxes; k++)  drawBlock(k, 0);
  for (int k = 0; k < wheels; k++) drawBlock(k, 7);
  matrix.renderBitmap(frame, 8, 13);
}

Bridge.provide_safe("show_counts", show_counts);
```

IT WORKS

The camera sees, Python counts, the MCU draws



Two boxes and one wheel in front of the camera



Detail: two blocks, the divider, one block

EXTENDING THE ACTUATION

Numbers, not a label, reach the MCU

ALARM

A batch is complete

Beep when the wheel count reaches four.

SORTING

Steer a servo gate

Turn toward whichever class is in the majority.

LOGGING

Count over time

Keep a running total on the MCU, independent of Linux.

STAGE 8 OF 8

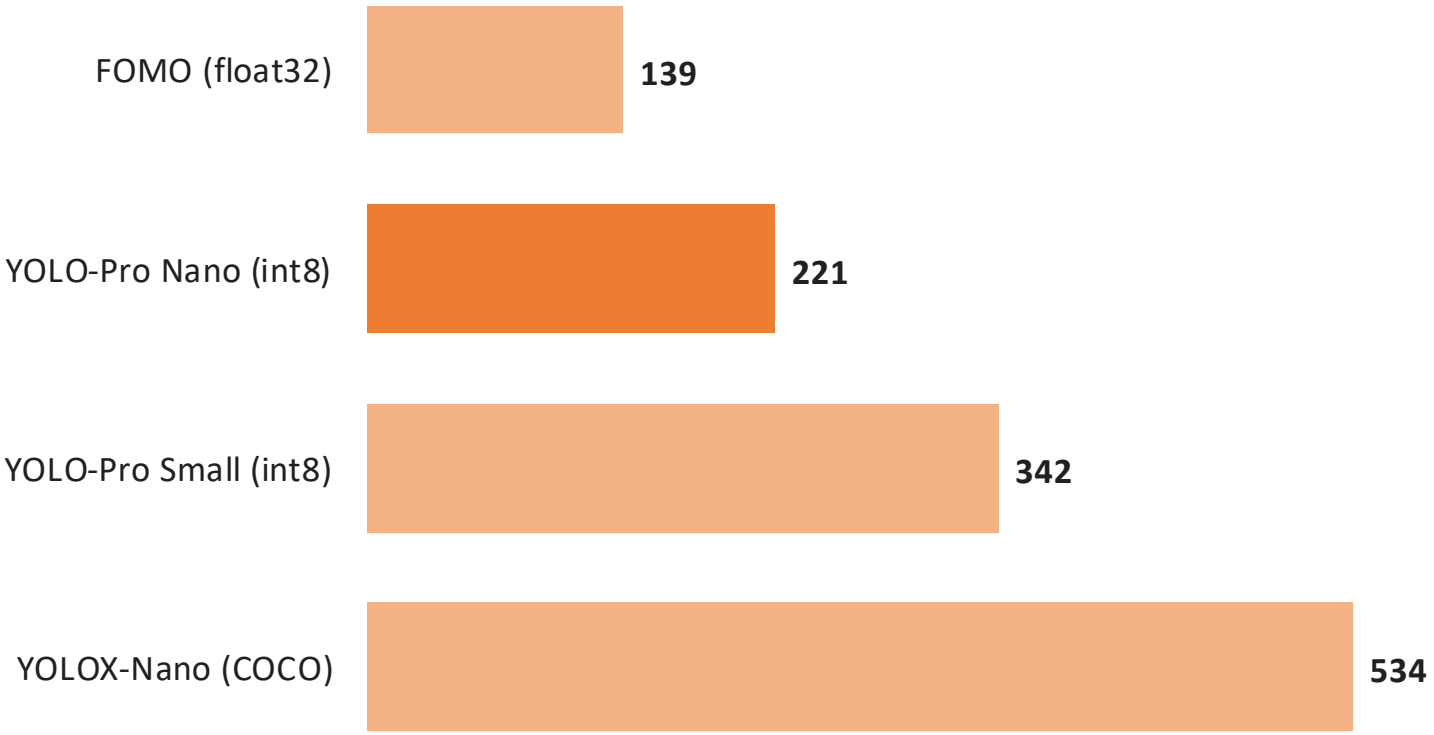
Measured on the board

8

Per-image latency and live frame rate, same board, same script.

PATH A: LATENCY PER IMAGE

Time around detect(), mean of 30 calls



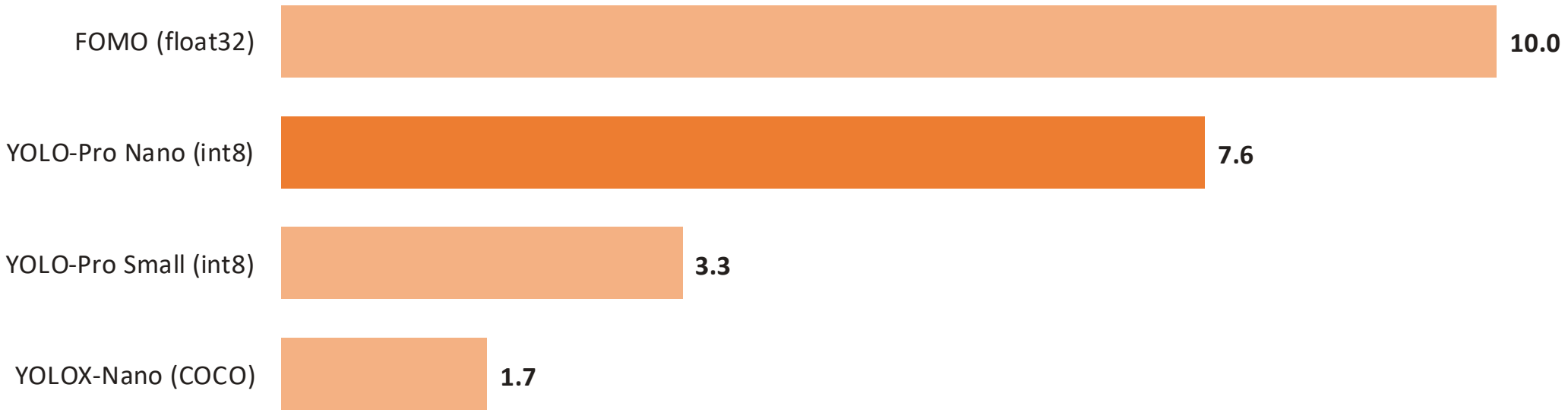
milliseconds per image, lower is better

~115 ms
fixed cost of moving the image to the runner

23 ms
FOMO's own processing inside the runner

PATH B: LIVE FRAME RATE

Results per second from the runner, over 30 seconds



frames per second, higher is better

FOMO hits the camera's 10 FPS limit. Nano more than doubles Small, with the same counts.

CHOOSING A MODEL

The measurements decide it

	FOMO	YOLO-Pro Nano	YOLO-Pro Small
Output	Centroid	Fitted box	Fitted box
Correct counts (6 test images)	2 of 6	6 of 6	6 of 6
Live frame rate	~10 FPS (camera)	~7.6 FPS	3.3 FPS
Edge Impulse estimate	11 ms	593 ms	1,841 ms
Best for	Small, separated objects	Counting, live	When every point of mAP matters

> Start from the accuracy you need, then check the latency measured on the board.

TAKE-AWAYS

Five things to remember

1. **Detection answers what and where**
   a label, a score, and a box per object
2. **Load the .eim file directly**
   one variable in app.yaml, and no model: line
3. **Silence is not zero**
   the video Brick says nothing on empty frames; use a timeout
4. **Record in the callback, act in the loop**
   one place talks to the MCU
5. **Measure on the board**
   the estimates missed by five times, in both directions

Questions?

Prof. Marcelo J. Rovai

rovai@unifei.edu.br

UNIFEI — Federal University of Itajubá, Brazil
AiEng4D — Academic Network Co-Chair
EdgeAIP — Academia-Industry Partnership Co-Chair



