CHAPTER 7 · IESTI05 EDGE AI ENGINEERING

Image Classification on the UNO Q

Teaching the board to name what it sees

Edge Impulse · App Lab Bricks · Bridge · LED matrix

Marcelo Rovai · UNIFEI



THE ARC SO FAR

From a model that talks to a model that sees

Generative AI
Chapters 2 to 6
one small language model
answers almost anything

→

Image classification
Chapter 7
one label for the
whole image

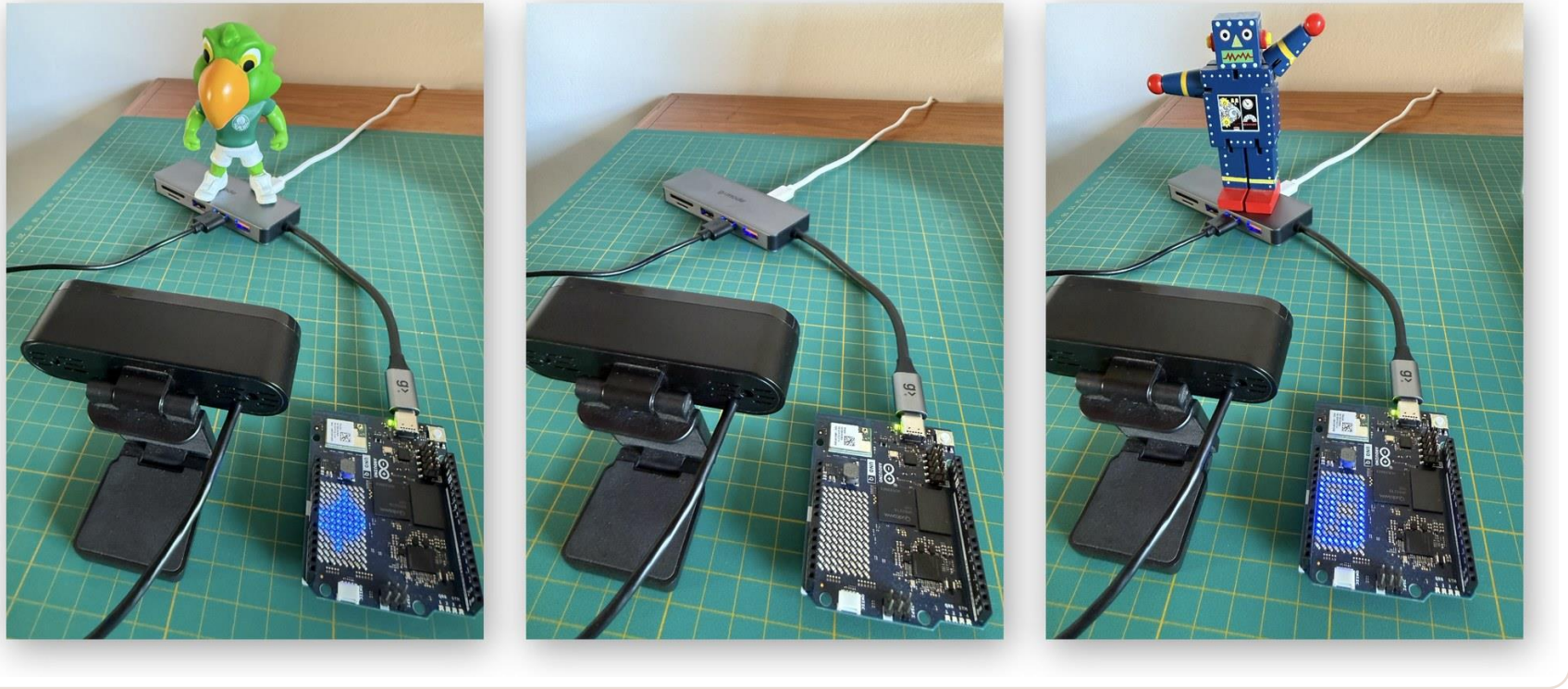
→

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

Fixed-function AI: trained for one job, small, fast, and predictable.

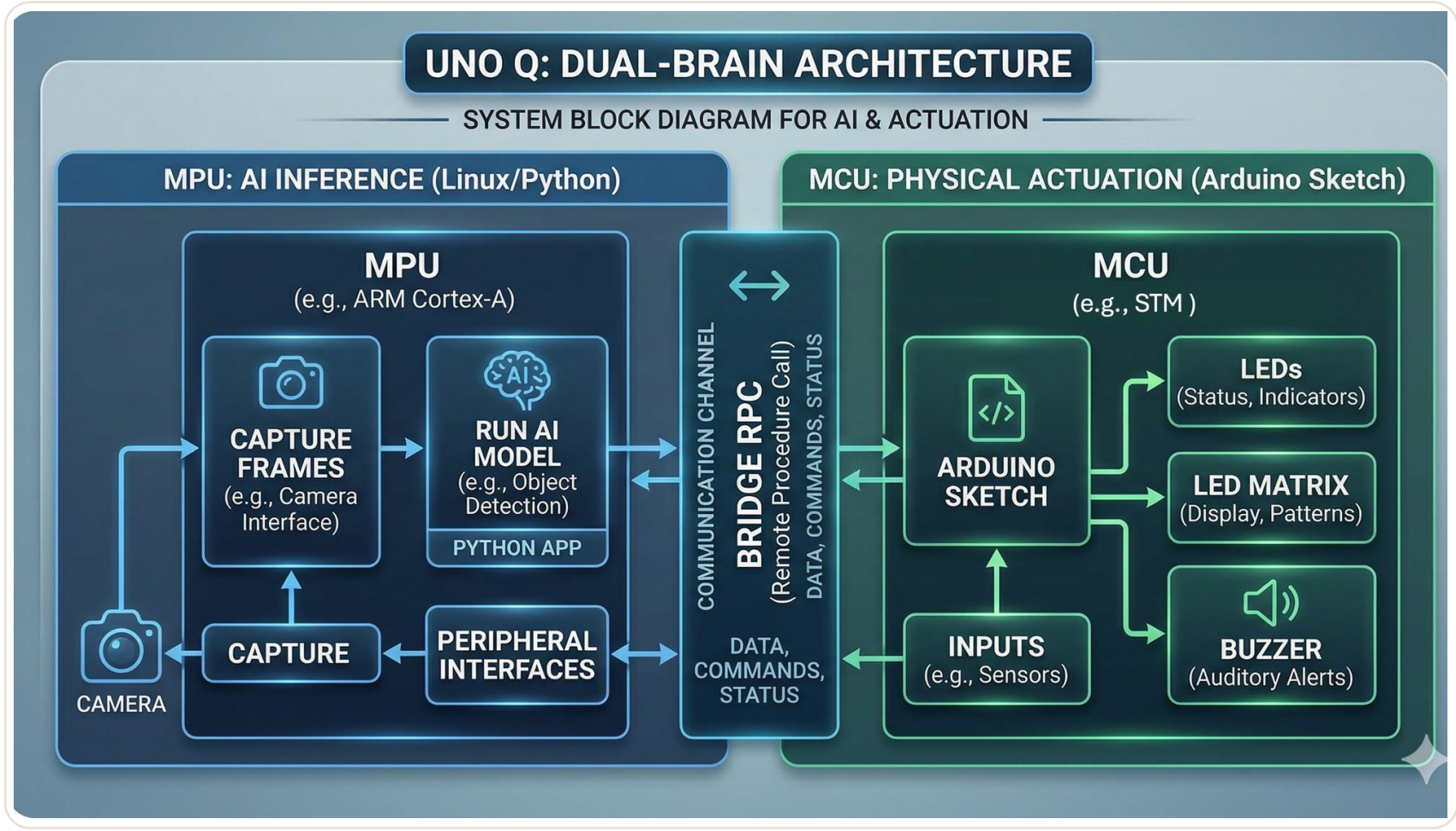
THE DESTINATION

A camera, a classifier, and an icon on the matrix



Python on Linux runs the model. The sketch on the MCU draws the result.

Python on the MPU, the sketch on the MCU



THE ROUTE

Eight stages, one live classifier at the end

1	**Vision**	Classification versus detection, and Nicla Vision versus UNO Q.
2	**Pre-installed**	The classifier that ships with the board, in the browser.
3	**Data**	Robot, periquito, and background: about sixty images per class.
4	**Training**	MobileNetV2 transfer learning in Edge Impulse Studio.
5	**Path A**	Our model in the static image app.
6	**Path B**	The same model on the live camera.
7	**MCU**	Bridge and the LED matrix.
8	**Results**	Latency, the two paths compared, and what to take away.

STAGE 1 OF 8

Computer vision at the edge

1

Classification gives one label per image.

TWO TASKS

Classification first, detection next

THIS CHAPTER

Image classification

One label for the whole image: this is a robot, this is a periquito.

NEXT CHAPTER

Object detection

A label and a bounding box for every object in the image.

TWO TASKS, SIDE BY SIDE

One label, or a box for every object

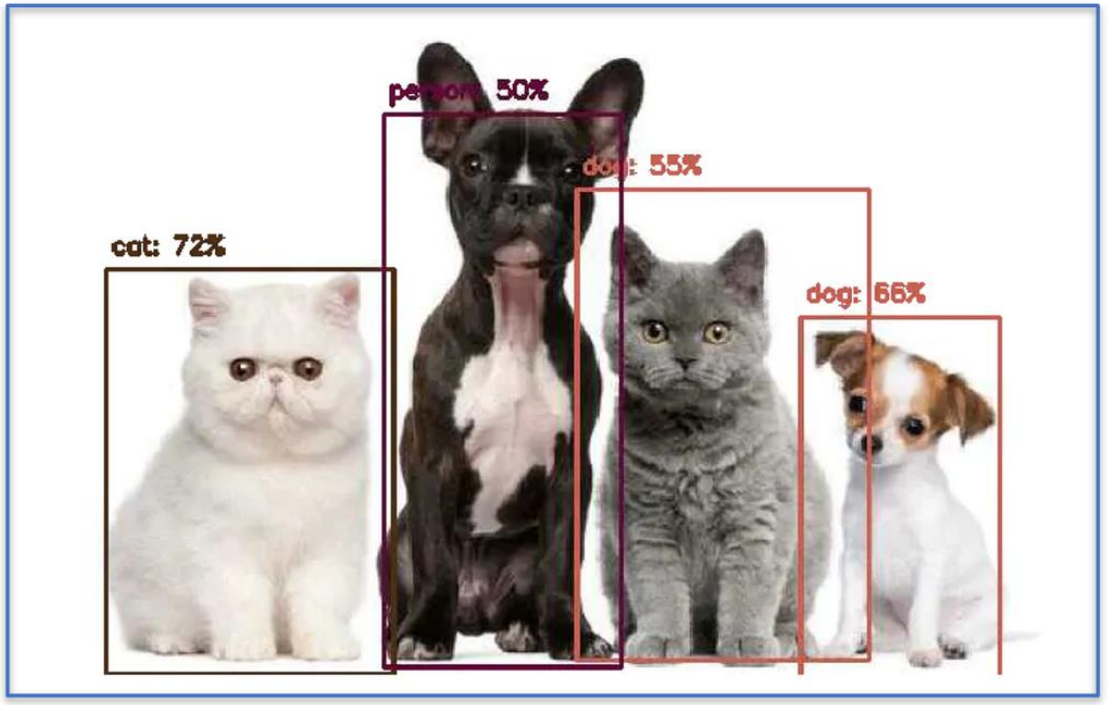
Image Classification

(**Multi-Class** Classification)



Object Detection

Multi-Label Classification + Object **Localization**



FROM NICLA VISION TO UNO Q

Same task, a very different board

	Nicla Vision (MCU)	Arduino UNO Q (MPU + MCU)
Camera	Built-in, 320 × 240	External USB webcam
Model	MobileNetV2 0.1, 96 × 96, int8	MobileNetV2 0.35 to 1.0, up to 160 × 160
Runtime	TF Lite Micro, C++	Edge Impulse model in an App Lab Brick, Python
Memory	About 256 KB of SRAM	2 or 4 GB of RAM
Actuation	GPIO, directly	MCU, through Bridge

STAGE 2 OF 8

The pre-installed classifier

2

Upload an image in the browser and read the label.

THE ARCHITECTURE

Two Bricks, no microcontroller

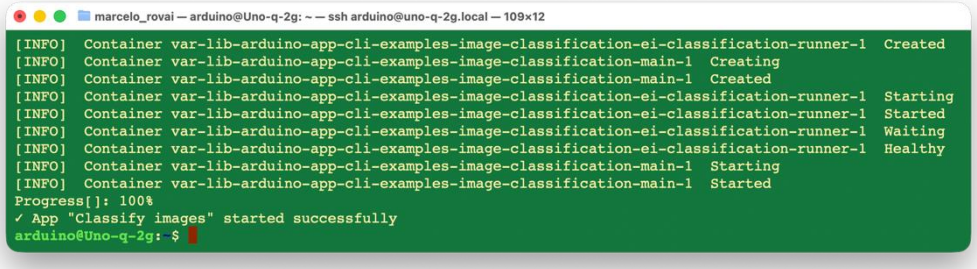


No webcam, no Bridge, no MCU. The model runs entirely inside the Brick.

RUN IT ON THE BOARD

One command, then open the browser

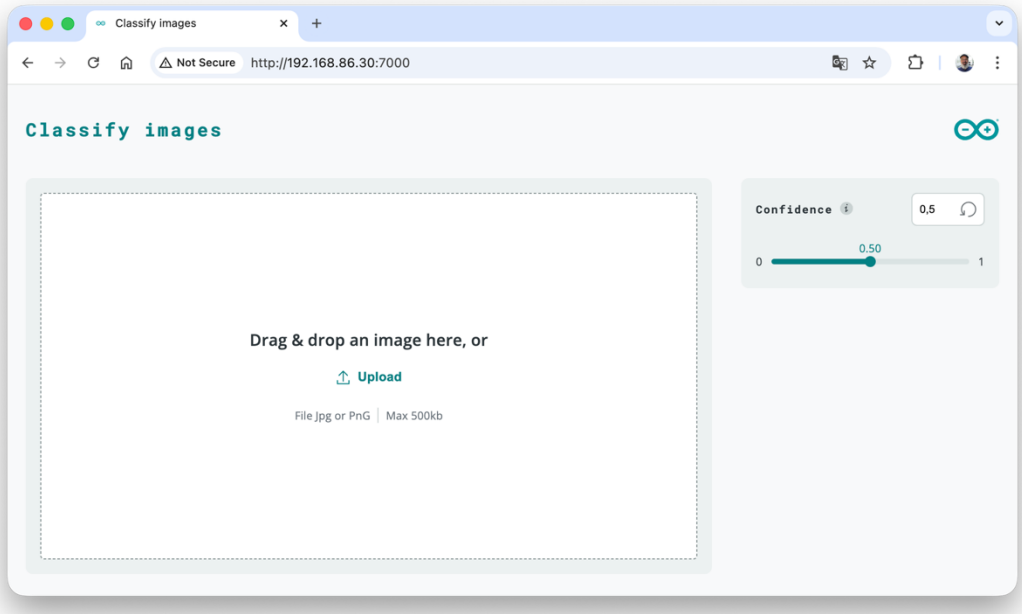
```
arduino-app-cli app start \
  examples:inspirational/image-classification
# then open http://<UNO_Q_IP>:7000
```



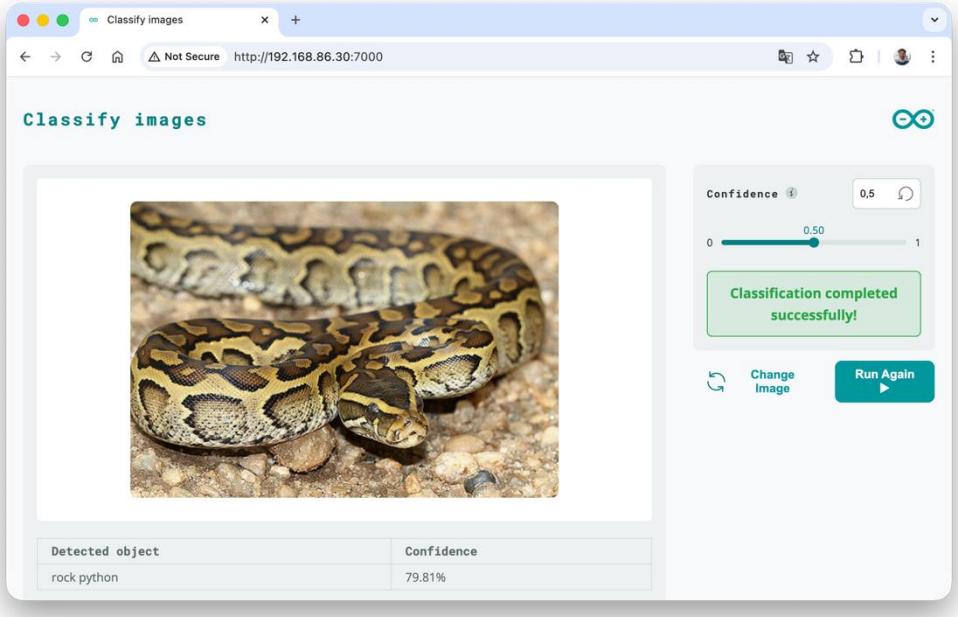
The app starts

CLASSIFY AN IMAGE

Upload, and the label comes back



The upload page



Labels and confidence scores

The results go to the browser over WebSocket, not to the terminal log.

THE SOURCE, IN TEN LINES

The Brick does the heavy lifting

```
image_classification = ImageClassification()
def on_classify_image(client_id, data):
    pil_image = Image.open(io.BytesIO(
        base64.b64decode(data['image'])))
    results = image_classification.classify(pil_image)
    ui.send_message('classification_result', results)
ui = WebUI()
ui.on_message('classify_image', on_classify_image)
App.run()
```

STAGE 3 OF 8

Data

3

Robot, periquito, and background: about sixty images per class.

THE PROJECT

Three classes, a live camera, an icon on the matrix



Robot · Periquito · Background (nothing in front of the camera)

COLLECTING THE IMAGES

Three ways to build the dataset

OPTION A

Your smartphone

Connect the phone to Edge Impulse Studio with a QR code and capture directly into the project.

OPTION B

The UNO Q's webcam

A small OpenCV script saves 320 × 240 frames on the board; copy them with scp and upload.

OPTION C

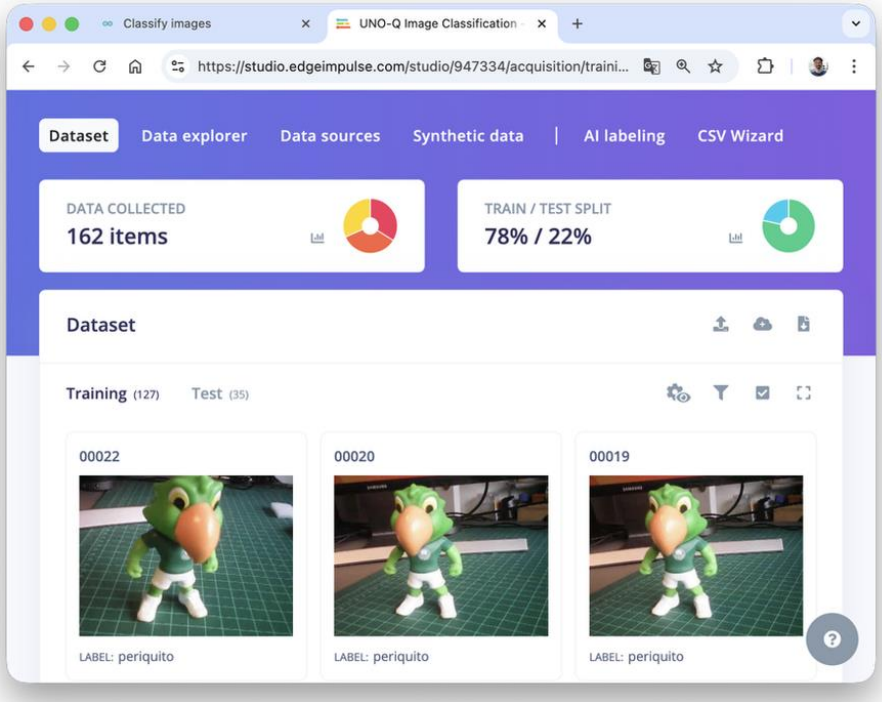
The Nicla images

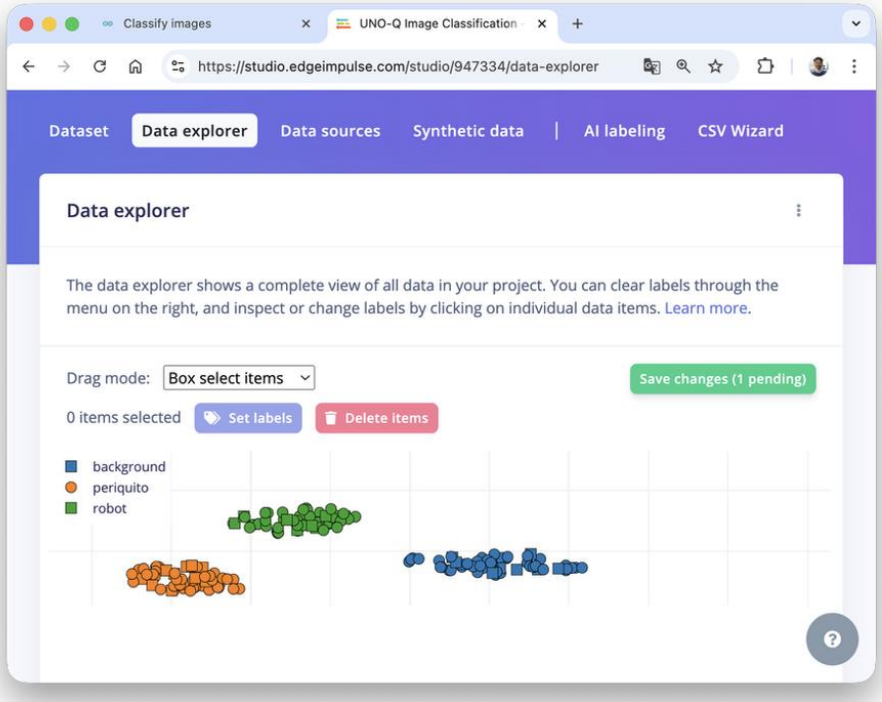
Reuse the dataset from the TinyML book, uploaded with the right label for each batch.

About 60 images per class, from varied angles, lighting, and backgrounds.

THE DATASET IN EDGE IMPULSE

Check the labels and the balance





STAGE 4 OF 8

Training

4

Transfer learning with MobileNetV2, in Edge Impulse Studio.

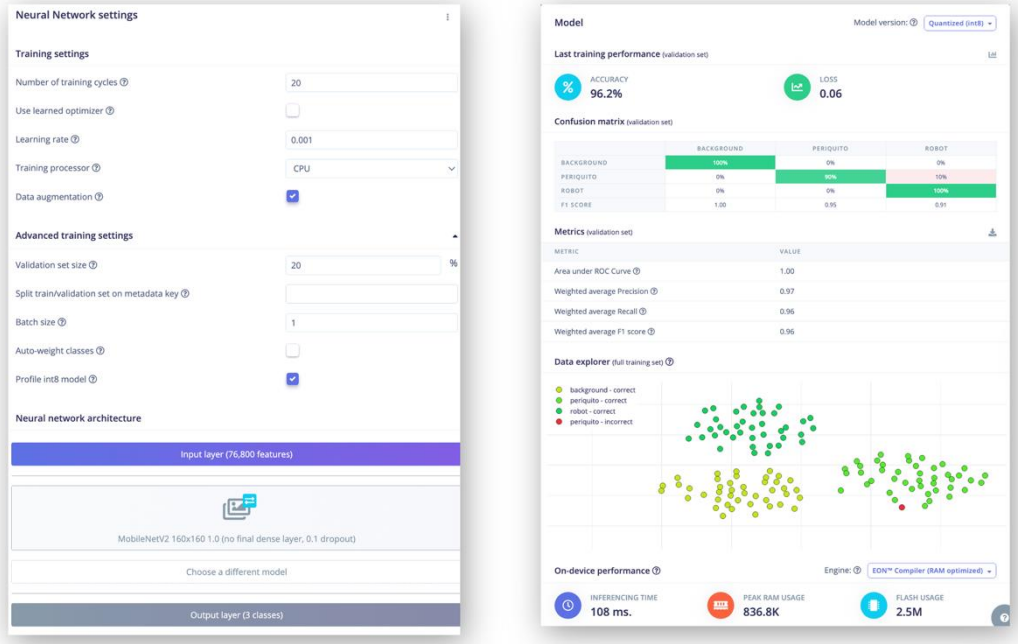
CREATE THE IMPULSE

Five settings, in order

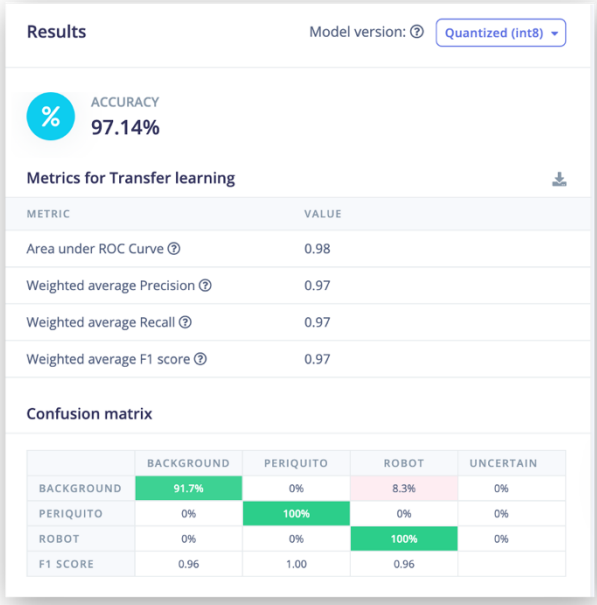
1. **Input: 160 × 160**
   larger than on the Nicla, for higher accuracy
2. **Processing block: Image, RGB**
   save parameters, then generate features
3. **Learning block: Transfer Learning**
   MobileNetV2 160 × 160, alpha 1.0
4. **Training: 20 cycles**
   with data augmentation
5. **Output: robot, periquito, background**
   three classes

TRAINING RESULTS

High accuracy, and an estimate to check later



Training: accuracy and confusion matrix



Model testing, int8

> Edge Impulse estimates about 100 ms per inference. The board will tell a different story.

STAGE 5 OF 8

Path A: static images

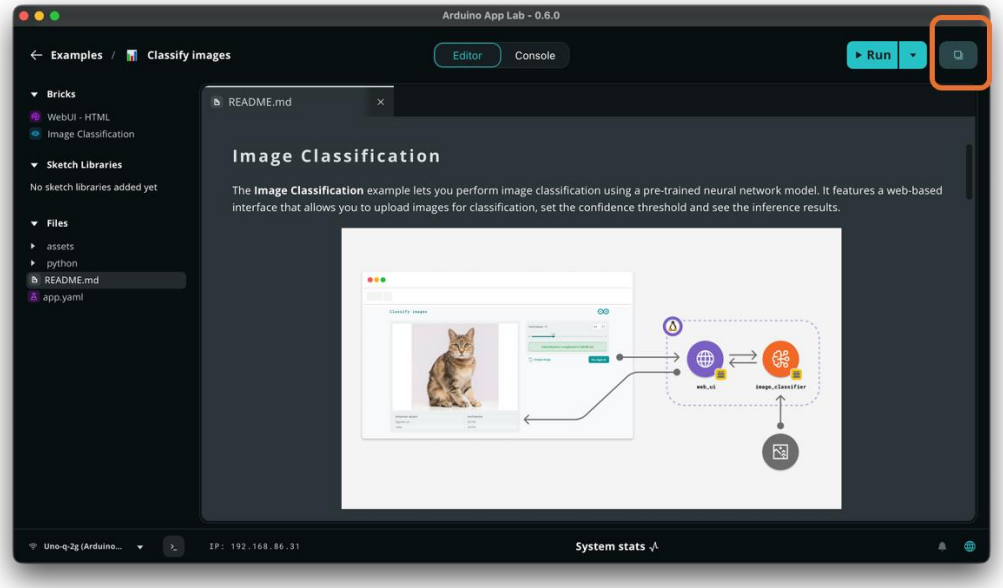
5

Our model in the pre-installed app.

COPY THE EXAMPLE

Our own app, from the pre-installed one

```
cp -r /var/lib/arduino-app-cli/examples/\
inspirational/common/image-classification/ \
  ~/ArduinoApps/my-classifier-bricks
```



Or copy it in App Lab

FROM EDGE IMPULSE TO APP LAB

The model travels in four clicks

1. **Brick › AI models › Train new AI model**
   in the copied app, in App Lab

2. **Log in and open your project**
   App Lab sends you to Edge Impulse Studio

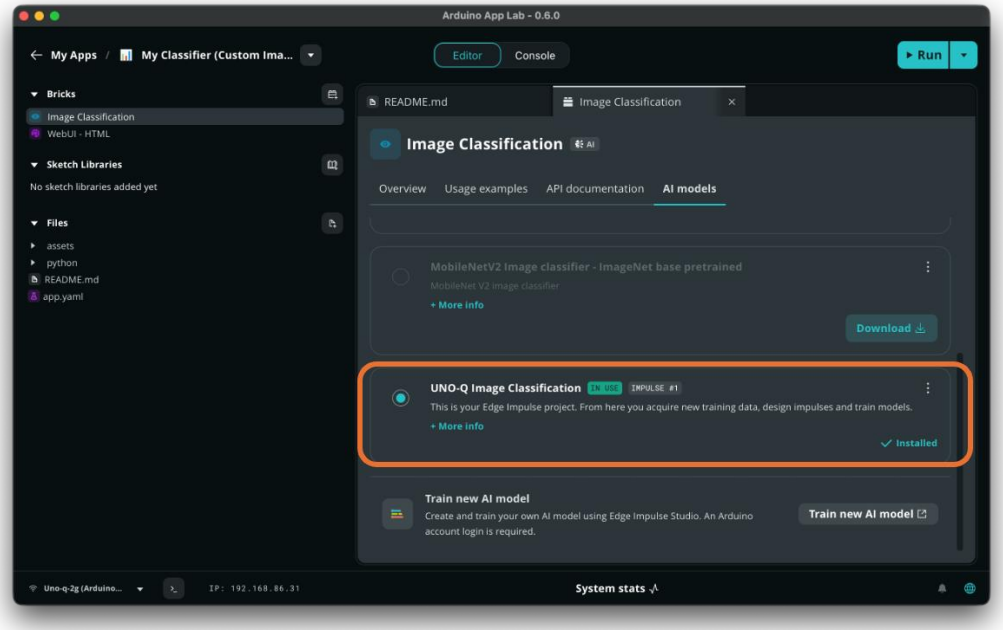
3. **Deployment › Arduino UNO Q › Build**
   the .eim file is built for the board

4. **Go to Arduino**
   the model appears in the app's AI models list

SELECT THE MODEL

App Lab writes it into app.yaml

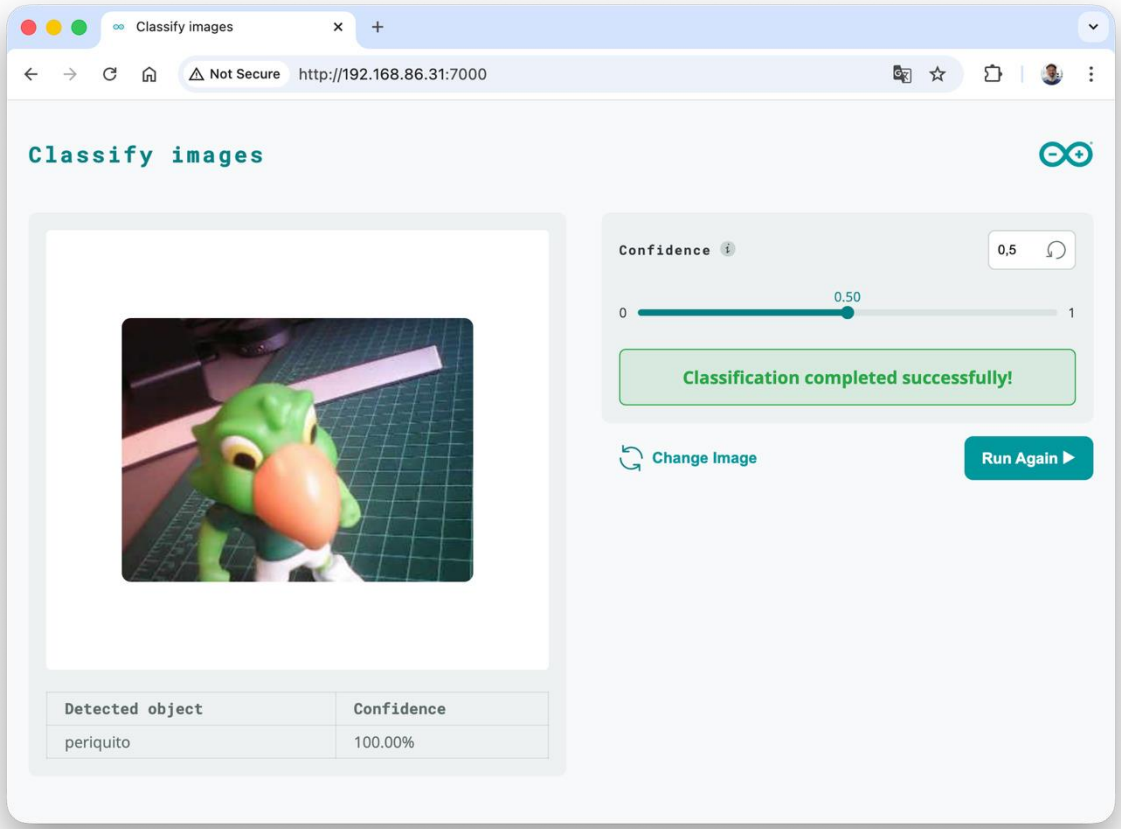
```
name: My Classifier (Custom Images)
bricks:
- arduino:image_classification:
    model: ei-model-947334-1
- arduino:web_ui: {}
```



Download it to the board, then select it

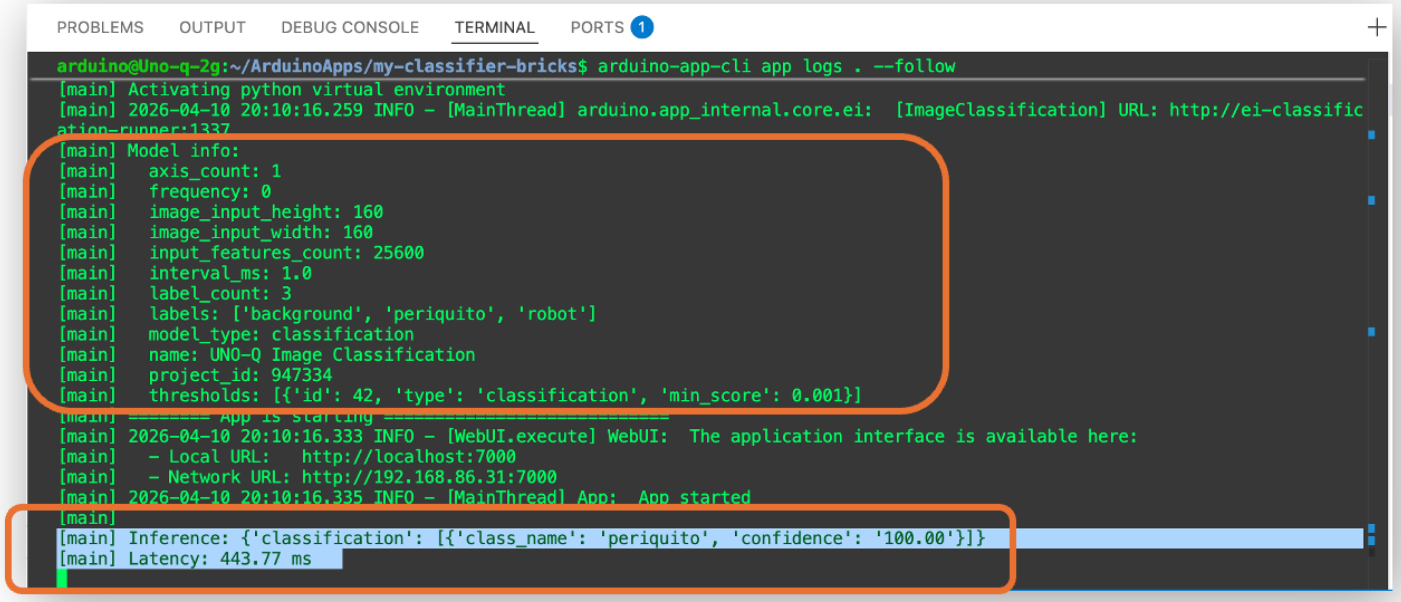
OUR CLASSES, ON THE BOARD

Same app, our model



WHAT THE LOG TELLS US

160 × 160 input, three labels, about 444 ms



About 444 ms per image, a little over 2 FPS. The estimate said about 100 ms.

STAGE 6 OF 8

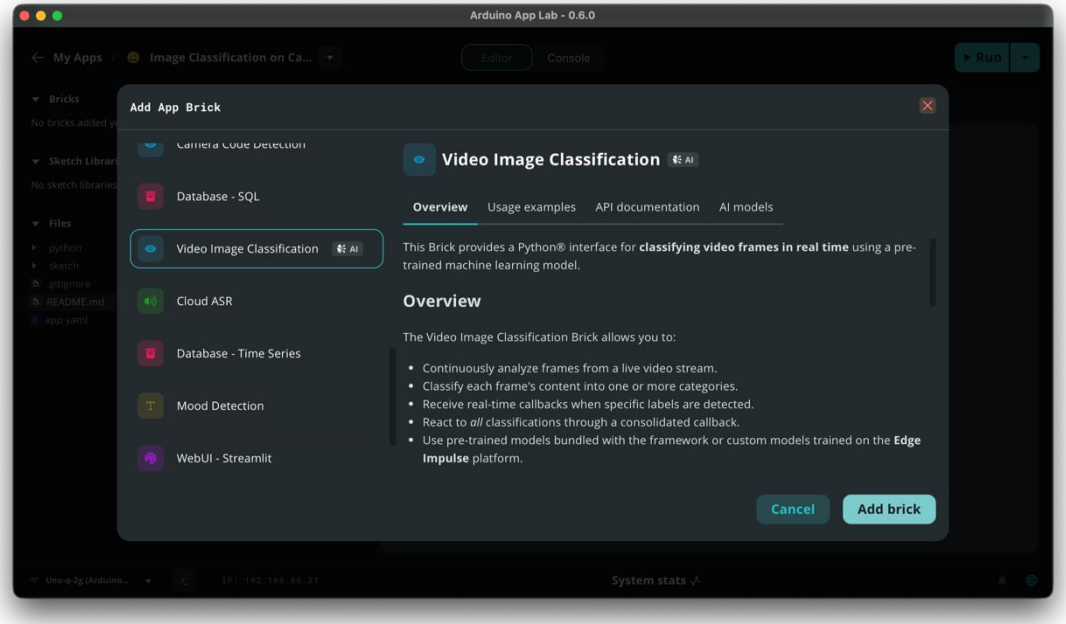
Path B: the live camera

6

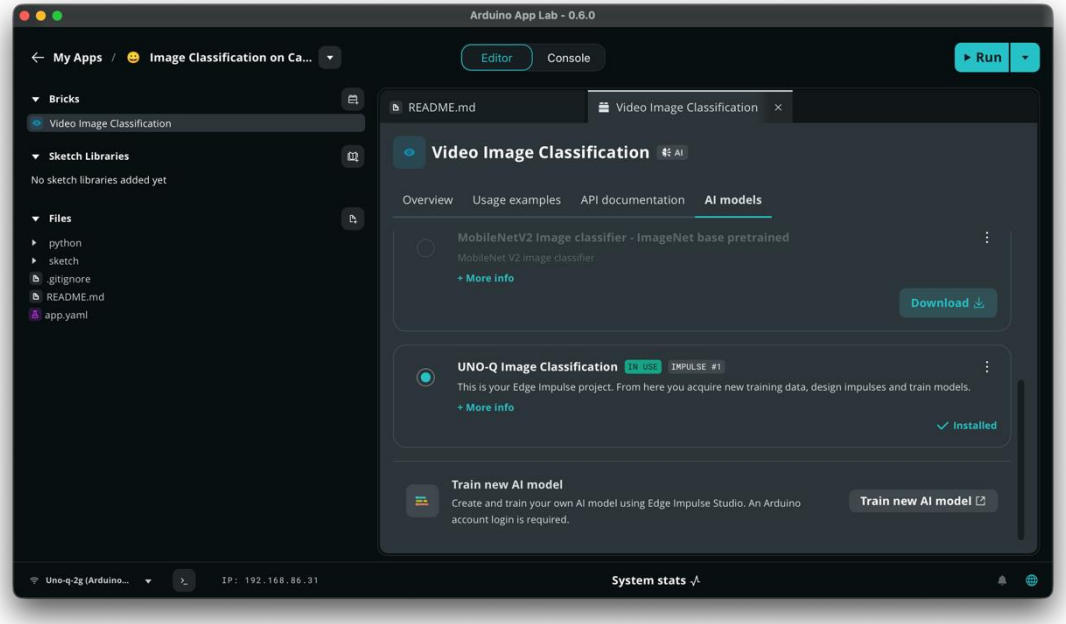
The Video Image Classification Brick, and our model.

A NEW APP WITH A VIDEO BRICK

Add the Brick, then pick our model



Add Brick › Video Image Classification



AI models › our project's model

CALLBACKS, NOT A LOOP

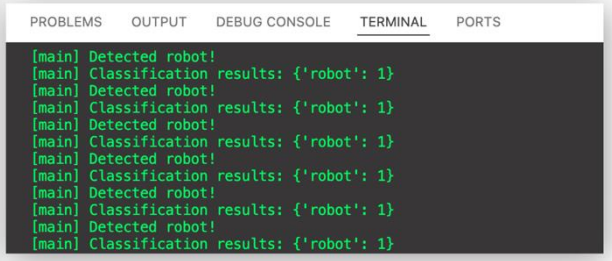
The Brick calls you when it sees a class

```
stream = VideoImageClassification(confidence=0.5)

def periquito_detected():
    print('Detected periquito!')

stream.on_detect('periquito', periquito_detected)
stream.on_detect_all(lambda r: print(r))
App.run()
```





The callbacks, in the log

STAGE 7 OF 8

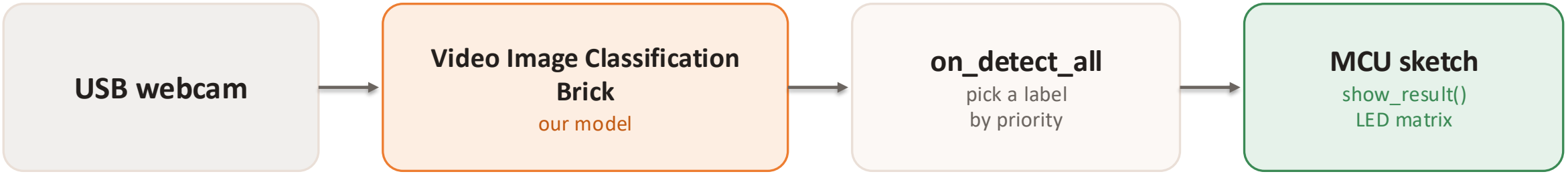
The MCU and the LED matrix

7

Bridge turns a label into an icon.

ARCHITECTURE

From a label to an icon



Bridge.call("show_result", label), only when the label changes.

robot
a face

periquito
a bird

background
empty

THE SKETCH

Three 8 × 13 bitmaps and one function

```
void show_result(String label) {
  if (label == "robot")
    matrix.renderBitmap(robot_frame, 8, 13);
  else if (label == "periquito")
    matrix.renderBitmap(bird_frame, 8, 13);
  else
    matrix.renderBitmap(empty_frame, 8, 13);
}
Bridge.provide("show_result", show_result);
```

THE PYTHON SIDE

Pick the top class, call only on change

```
PRIORITY = {'robot': 2, 'periquito': 1, 'background': 0}

def all_detected(results):
    label = max(results, key=lambda r: PRIORITY.get(r, -1)) \
            if results else 'background'
    if label != current_label:
        current_label = label
        Bridge.call('show_result', label)
```

Confidence lowered to 0.3, so that background results still come through.

IT WORKS

The camera sees, the MCU draws







EXTENDING THE ACTUATION

One label, many outputs

LEDS

A color per class

Blue for the robot, green for the periquito, red for background.

BUZZER

A tone per class

A short beep at a different pitch for each object.

SERVO

A position per class

Zero, ninety, or one hundred eighty degrees.

STAGE 8 OF 8

Results

8

What we measured, and the two paths side by side.

MEASURED ON THE BOARD

MobileNetV2 1.0 at 160 × 160, Path A

~444 ms
per image, end to end

~2 FPS
a little over two classifications a second

~100 ms
Edge Impulse's estimate

> The gap is the work around the model. Measure on the board before choosing a model.

PATH A VERSUS PATH B

Two Bricks, two jobs

	Path A: Image Classification Brick	Path B: Video Image Classification Brick
Input	Static images, uploaded in the browser	Live webcam, continuous
Trigger	One upload, one result	Callbacks, frame after frame
MCU actuation	Not included	Yes, through Bridge
Best for	Checking a model, demos	Real-time applications

TAKE-AWAYS

Four things to remember

1

Edge Impulse to App Lab to the board

collect, train, deploy, and run, guided by the UI

2

Two Bricks, two modes

static images to check, a video stream to deploy

3

Two brains

Python runs the model; the sketch drives the hardware; Bridge connects them

4

Measure on the board

about 444 ms measured, against an estimate of about 100

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



