

The logo for the Embedded Vision Summit is a circular emblem. It features a central white circle containing the text "embedded VISION summit". The word "embedded" is in a teal color, "VISION" is in a dark blue color, and "summit" is in a dark blue color. The letter "O" in "VISION" is replaced by a small, colorful circular graphic composed of several segments in shades of orange, yellow, and teal. Surrounding this central circle is a larger, multi-layered ring made of various colored segments in shades of teal, orange, yellow, and light blue, creating a gear-like or sunburst effect.

embedded
VISION
summit

FOMO: Real-Time Object Detection on Microcontrollers

Jan Jongboom
CTO
Edge Impulse

Wait... What? Edge Impulse?



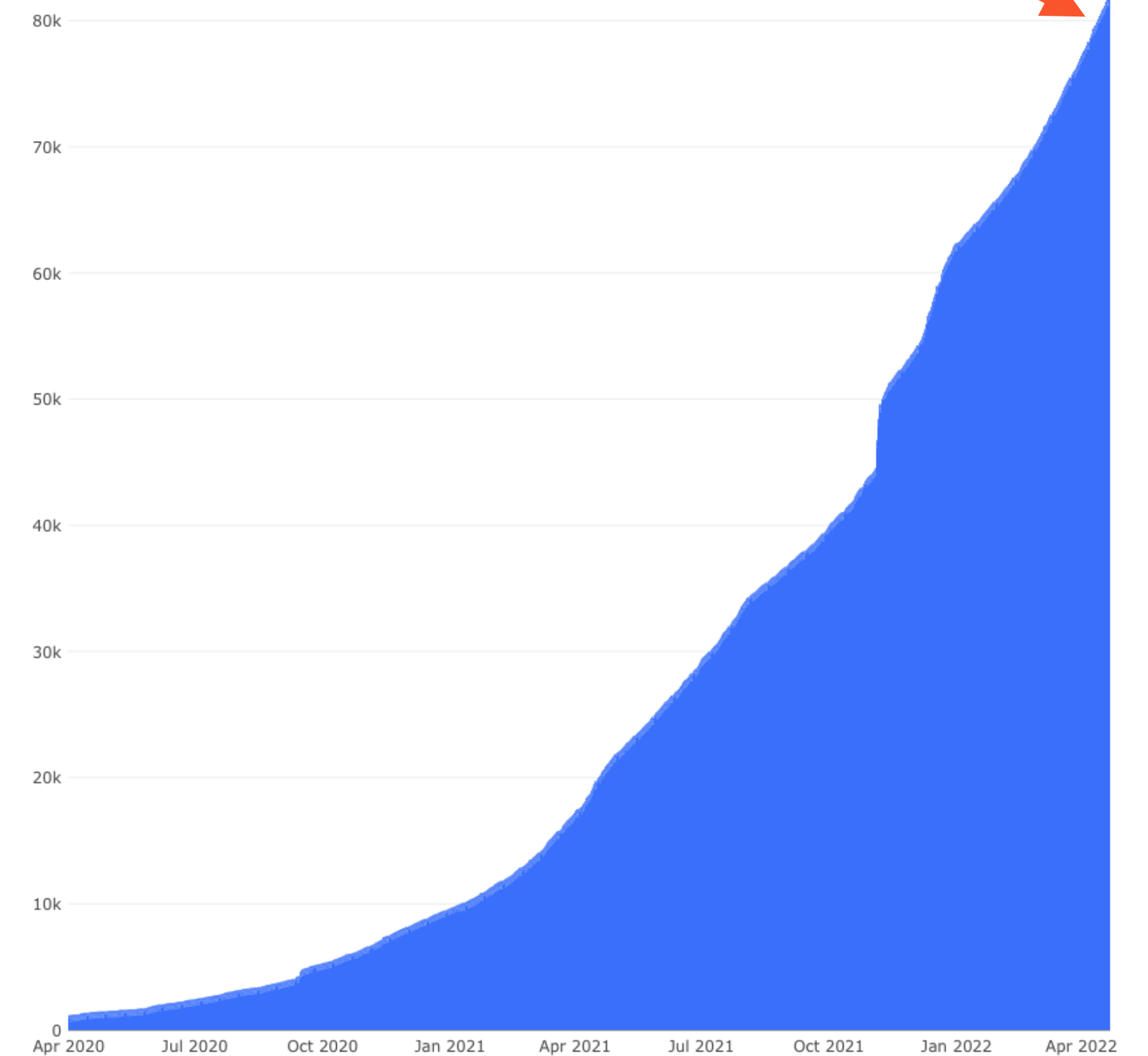
Leading development platform for machine learning on edge devices

Launched two years ago... Now 240 new projects daily (!)

40% of these are vision projects

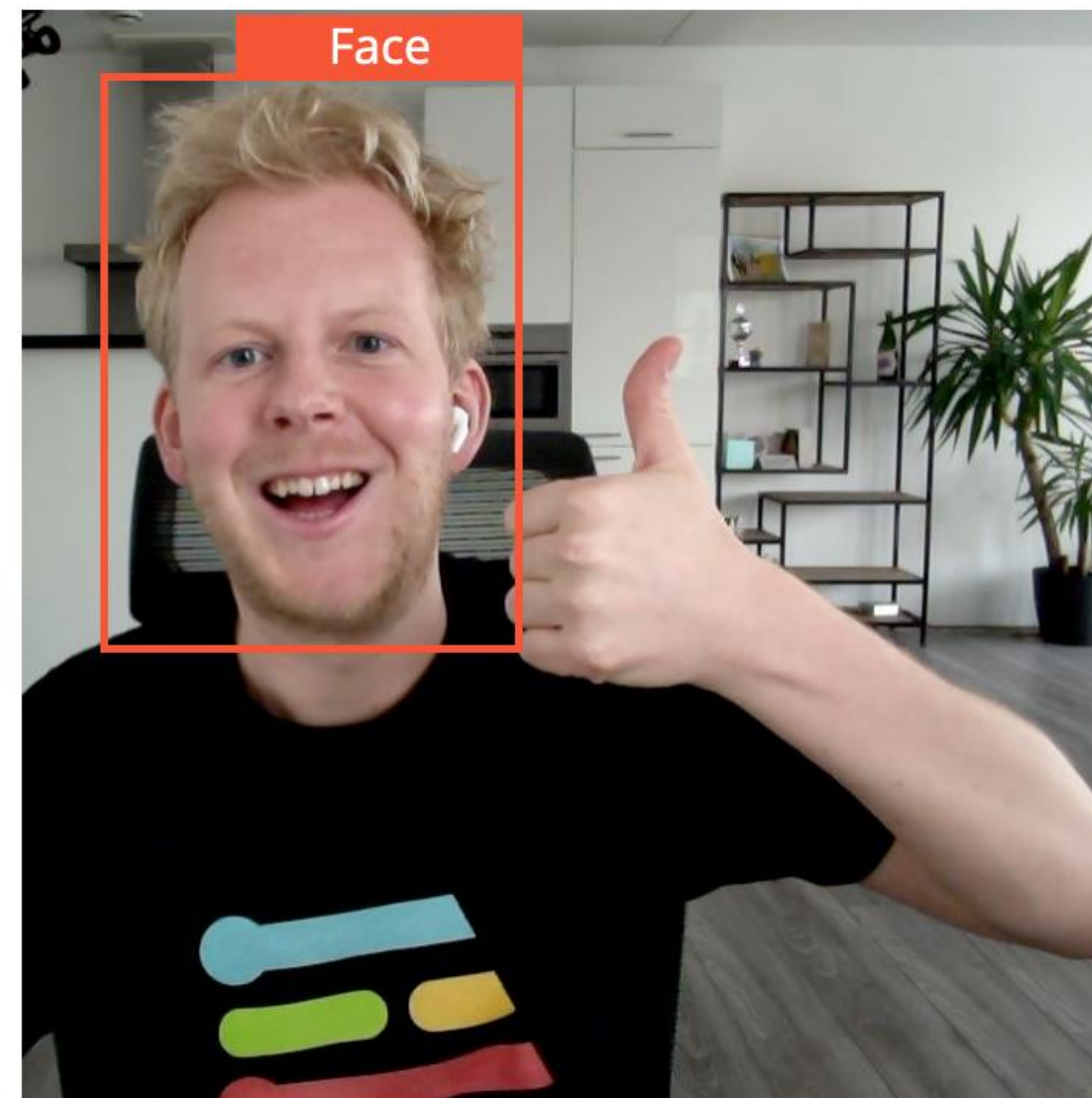
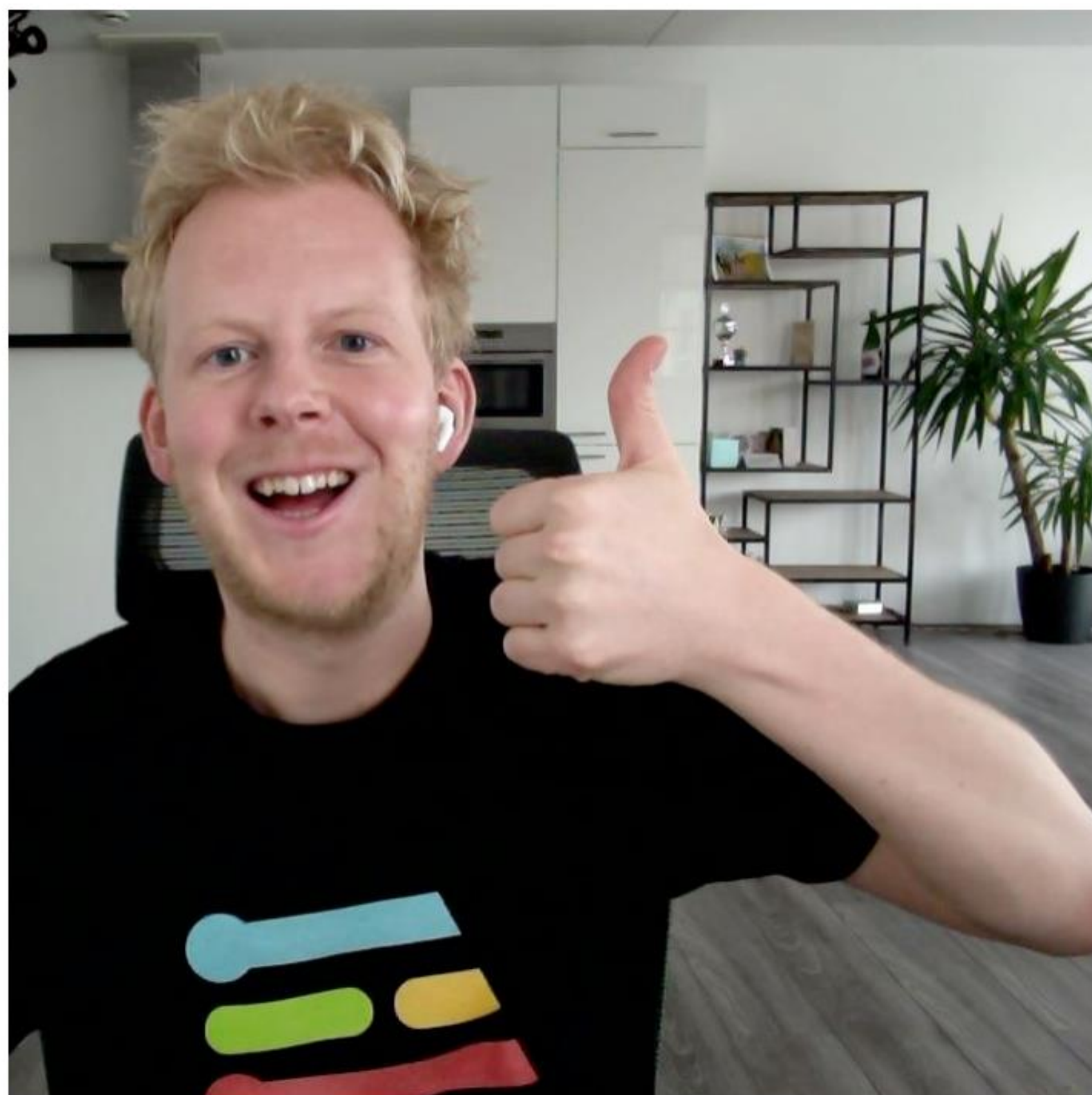
Two years ago
(1,163 projects)

Today
(82,069 projects)



Edge Impulse project count

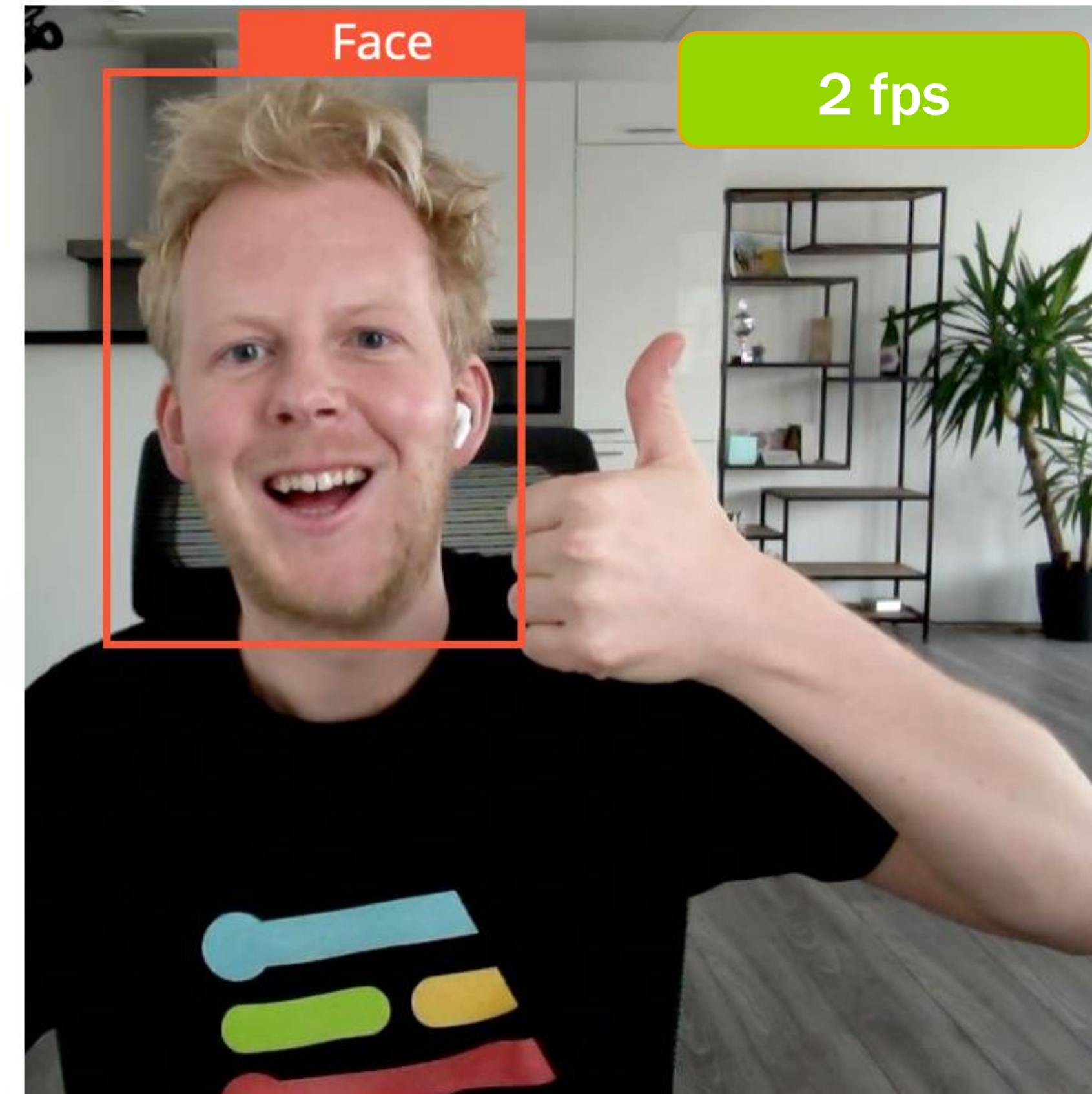
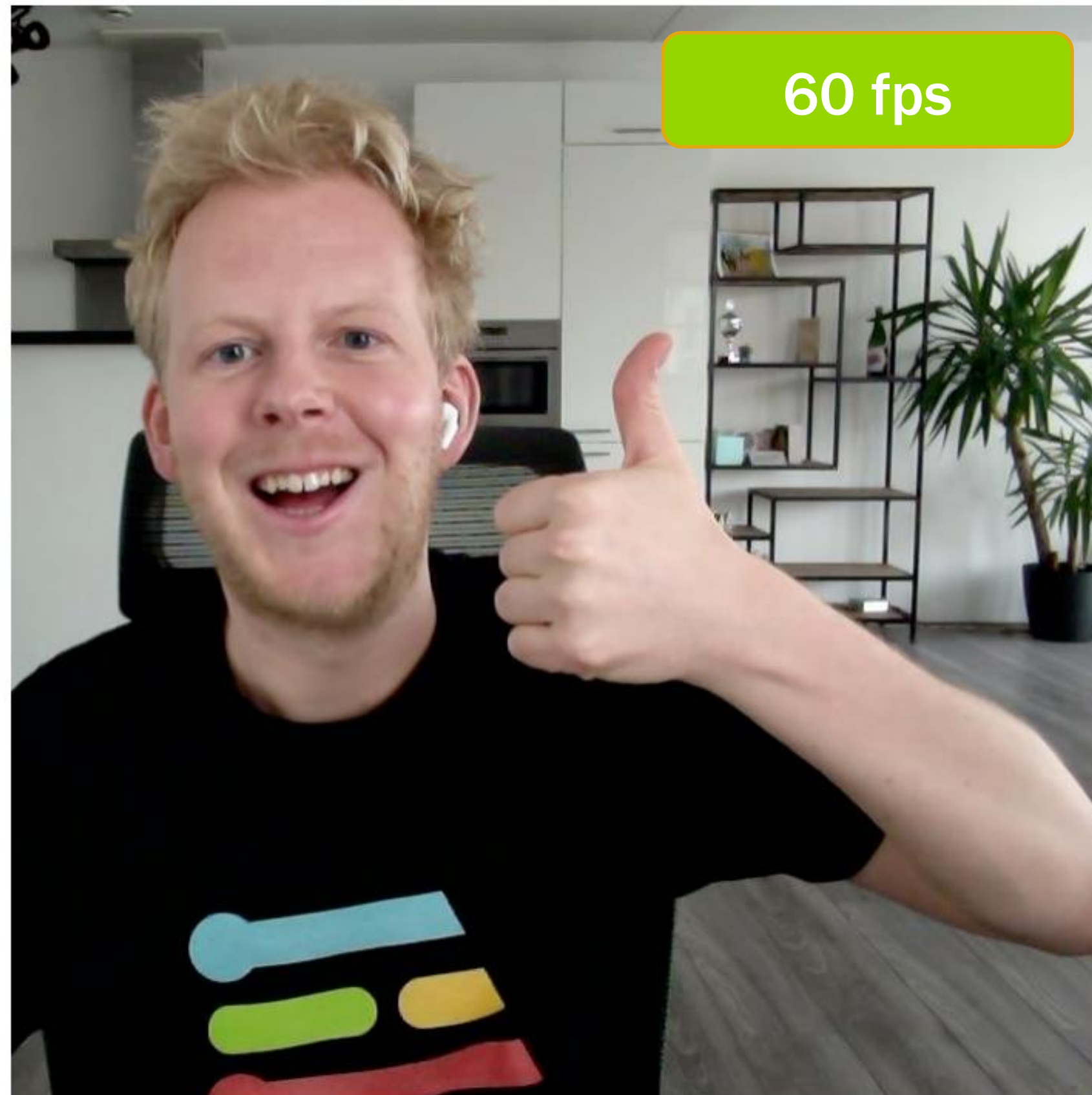
Image classification vs. object detection



Significantly Different Performance (Rpi4)



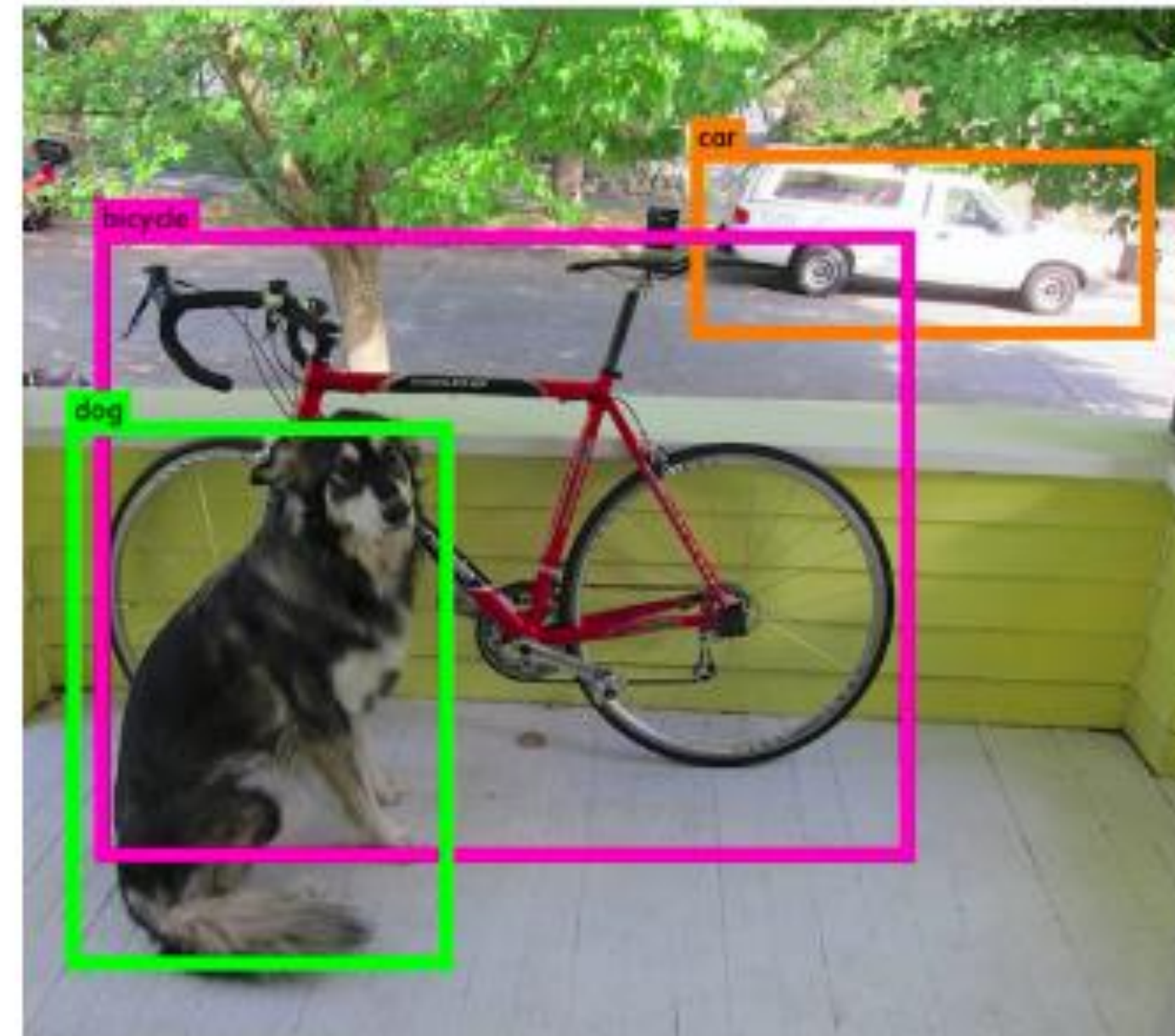
Image classification vs. object detection



But Why?



Multiple Bounding Boxes

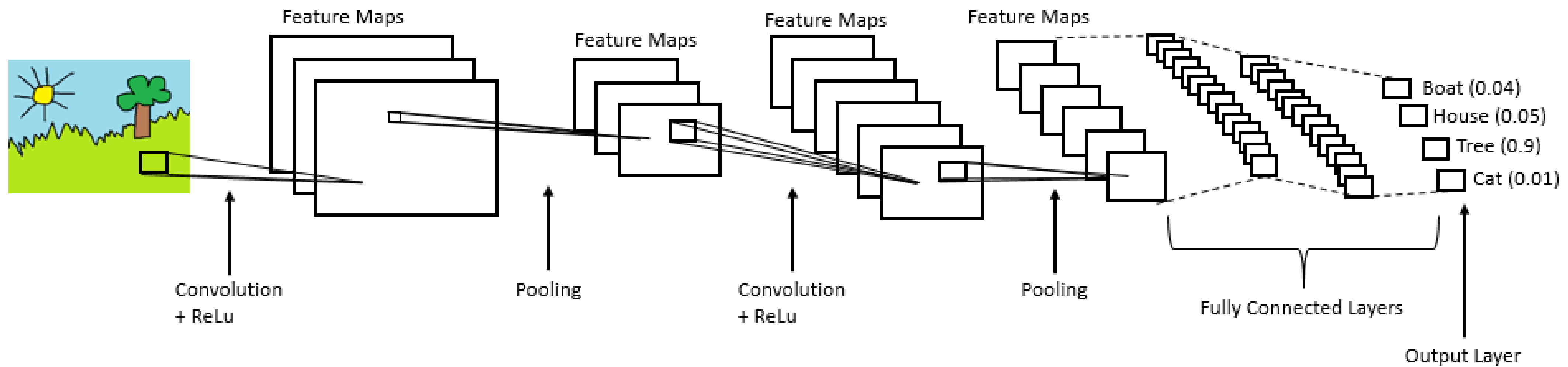


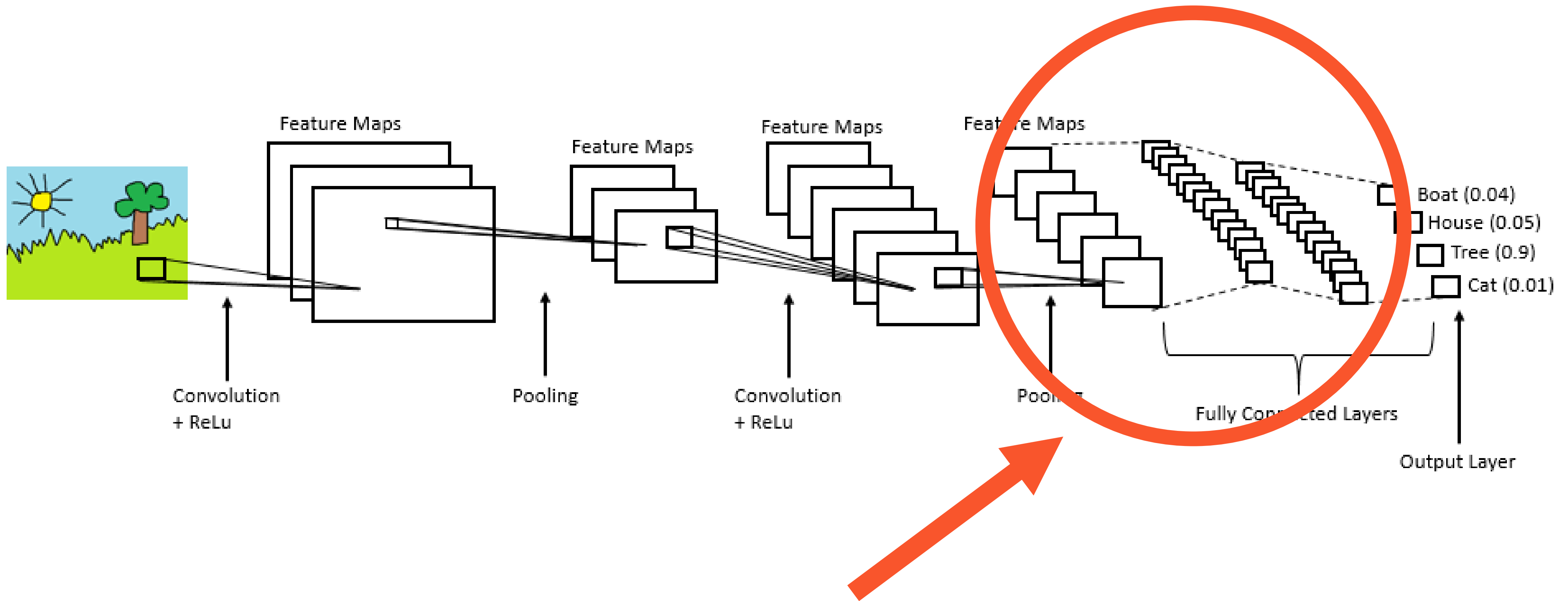
Final Bounding Boxes

Source: <https://pjreddie.com/darknet/yolov1/>

"The tiny version of YOLO only uses 516 MB of GPU memory"

Object Detection (MobileNet SSD, YOLO)

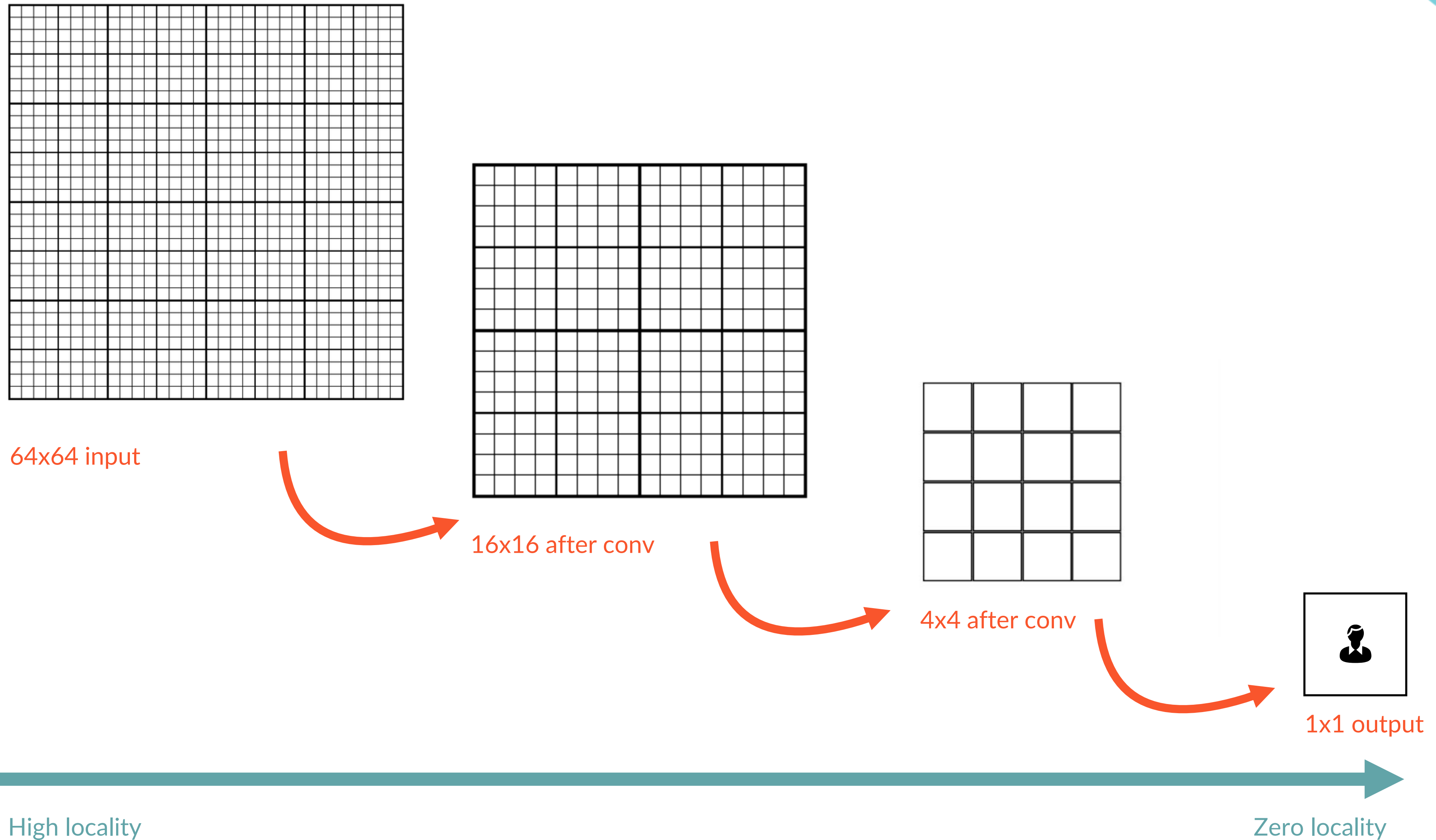


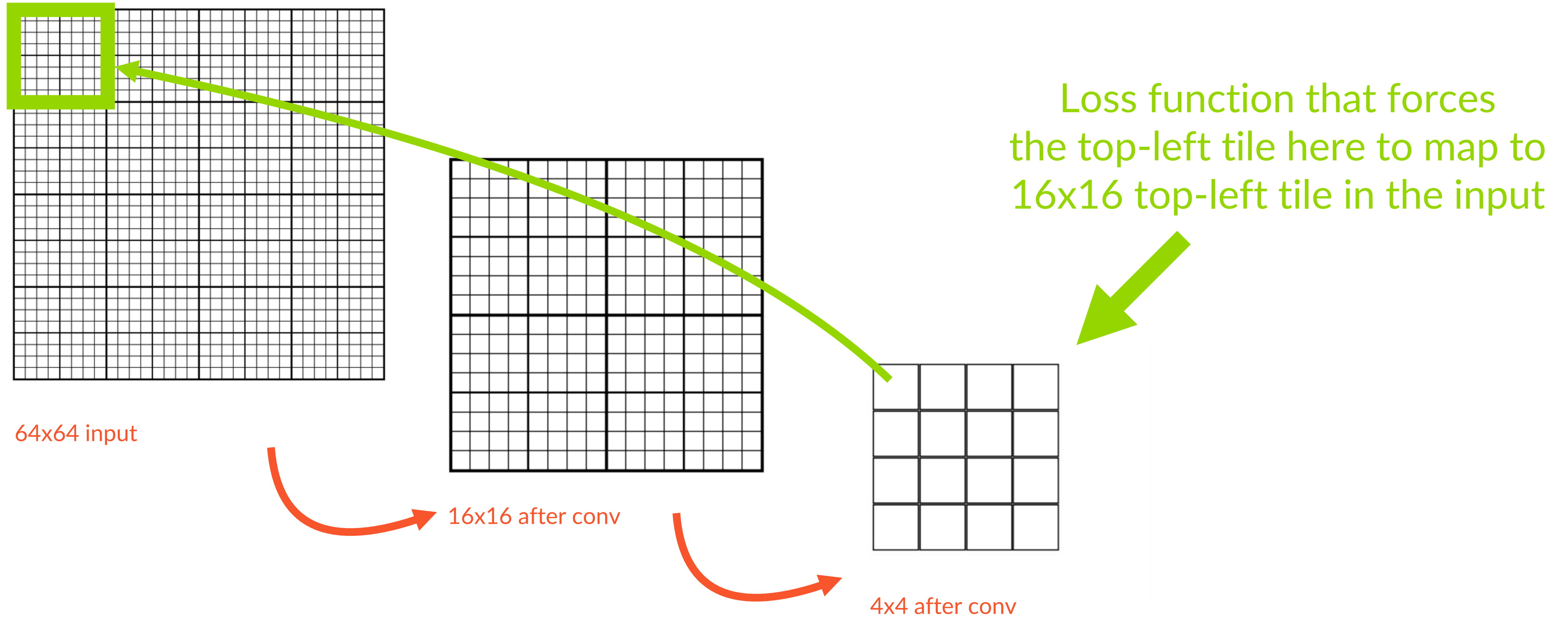


Replace with single per-region class probability map

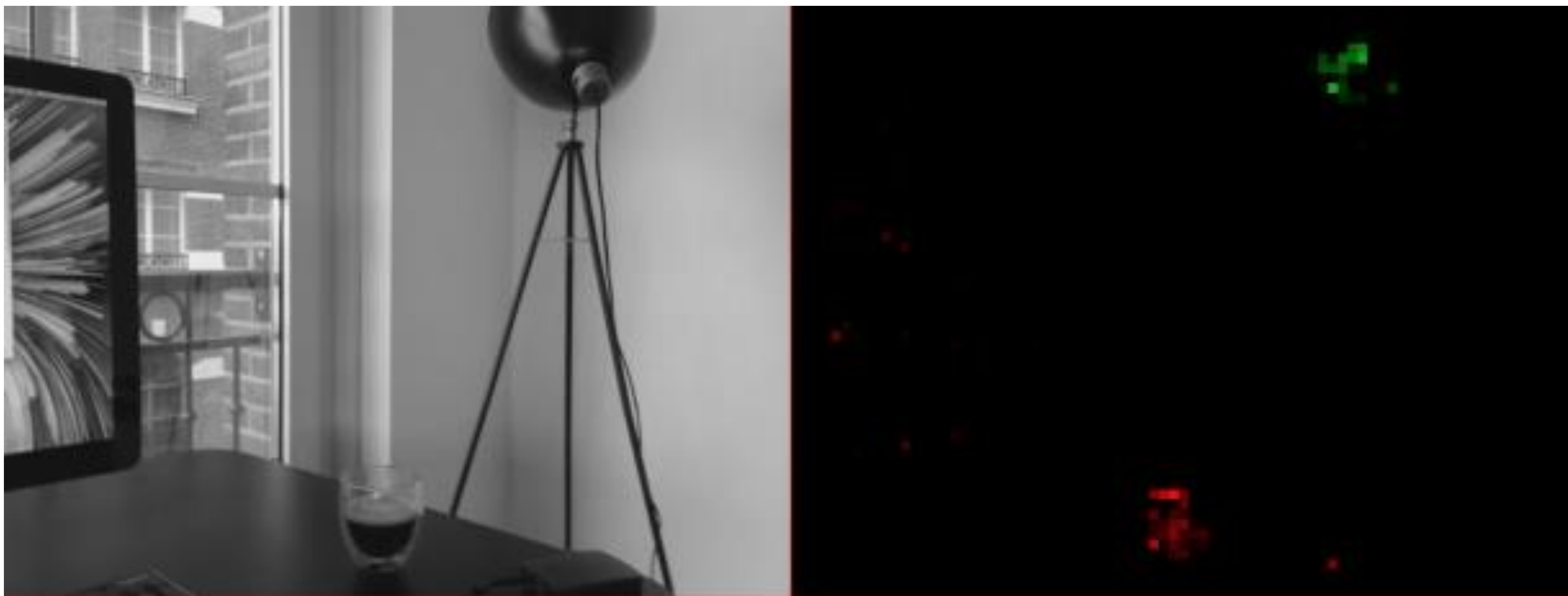
Single Per-Region Class Probability What?

Convolutional Image Classification Model

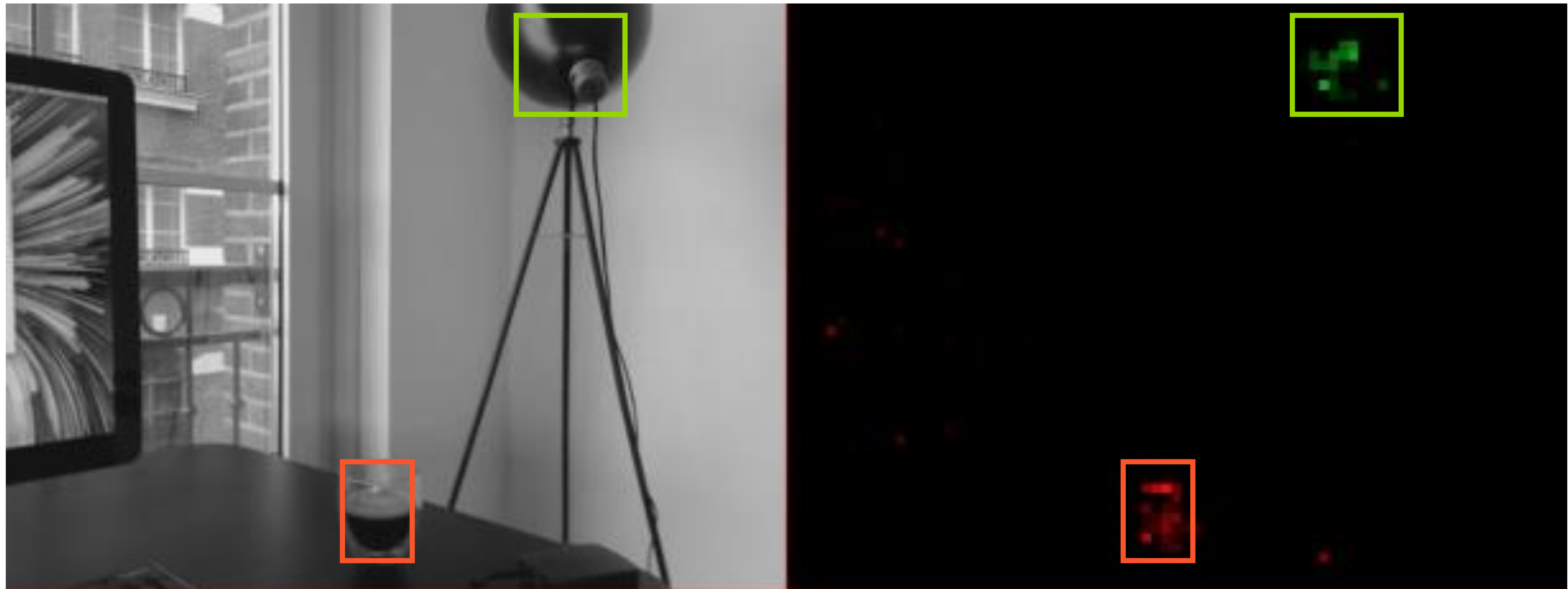




Output = Heatmap



Postprocessing in Software

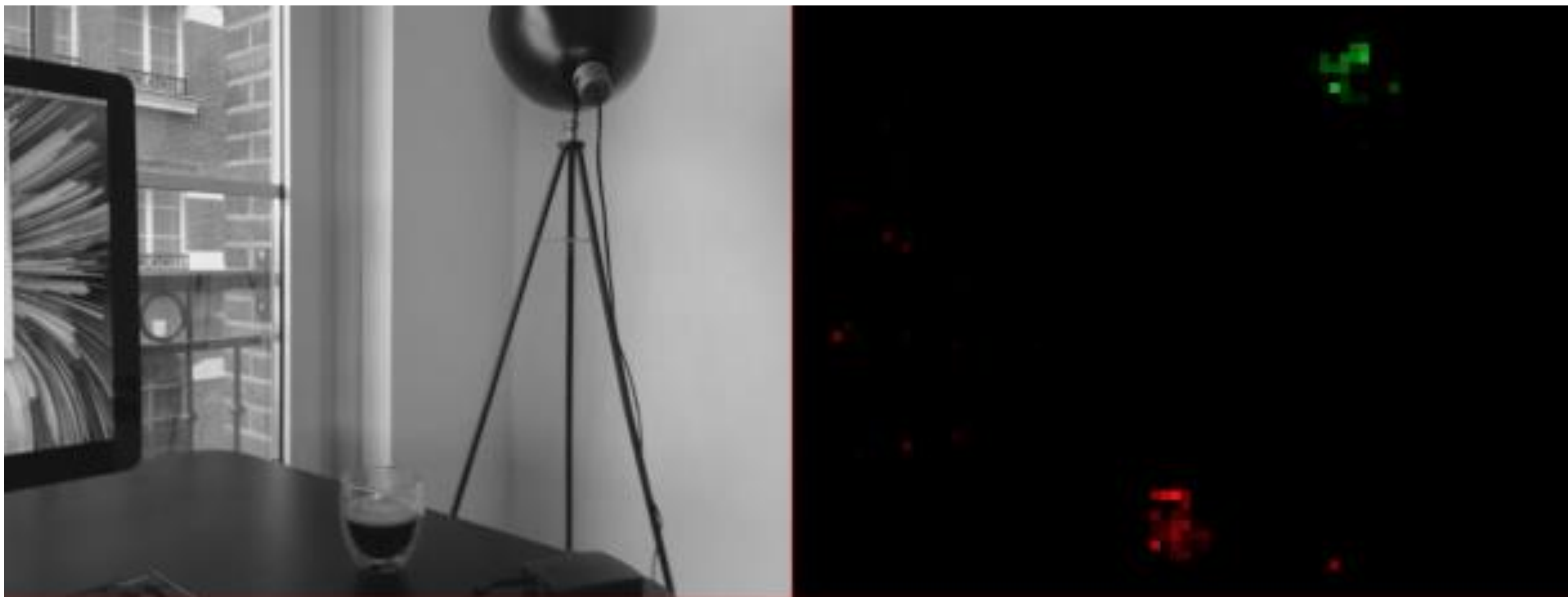


Not constrained to just bounding boxes!

Heatmap Ratio is Configurable

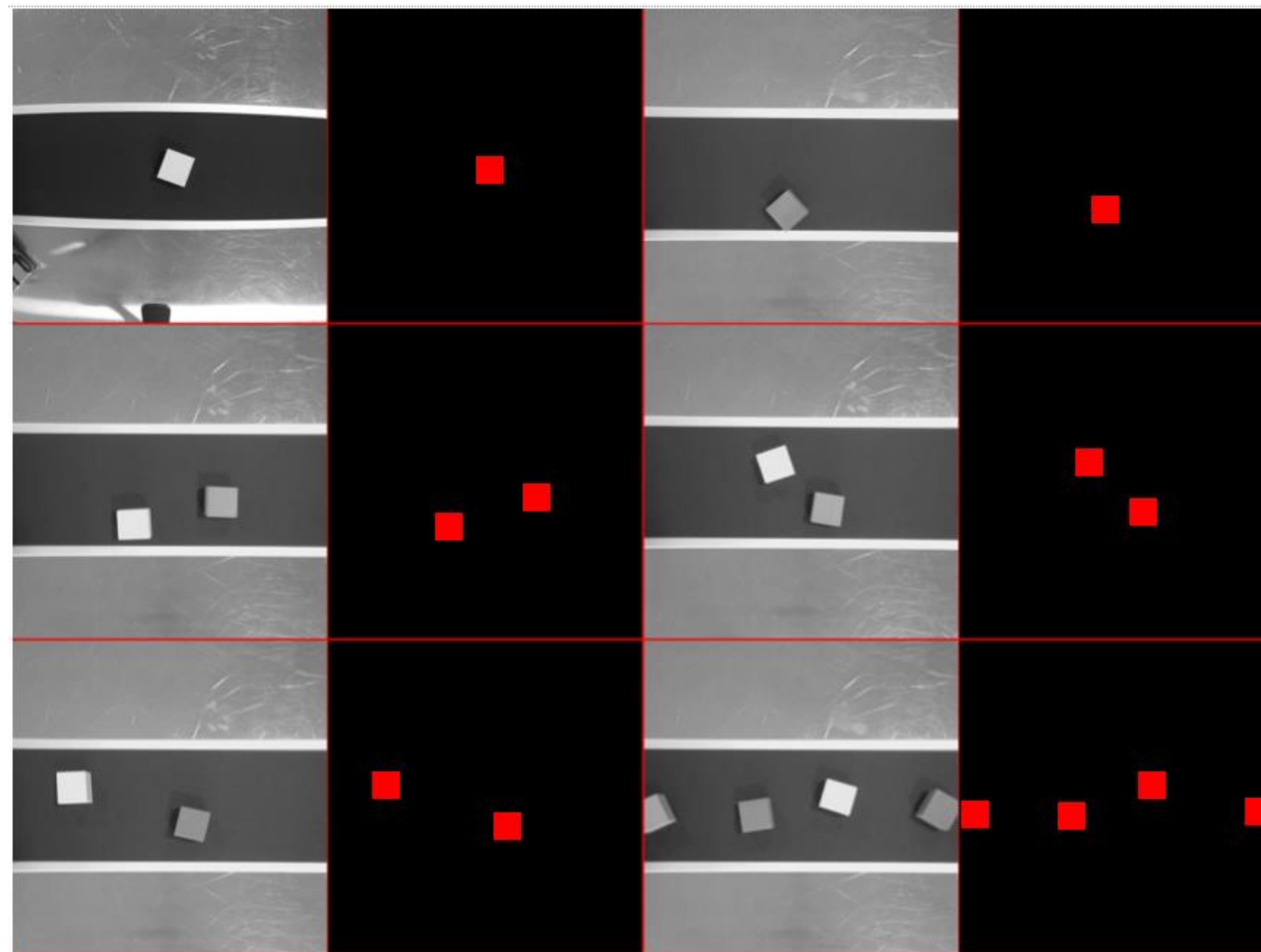


Flexibility in Input / Output Resolution



320 x 320 input => 80 x 80 output

Flexibility in Input / Output Resolution



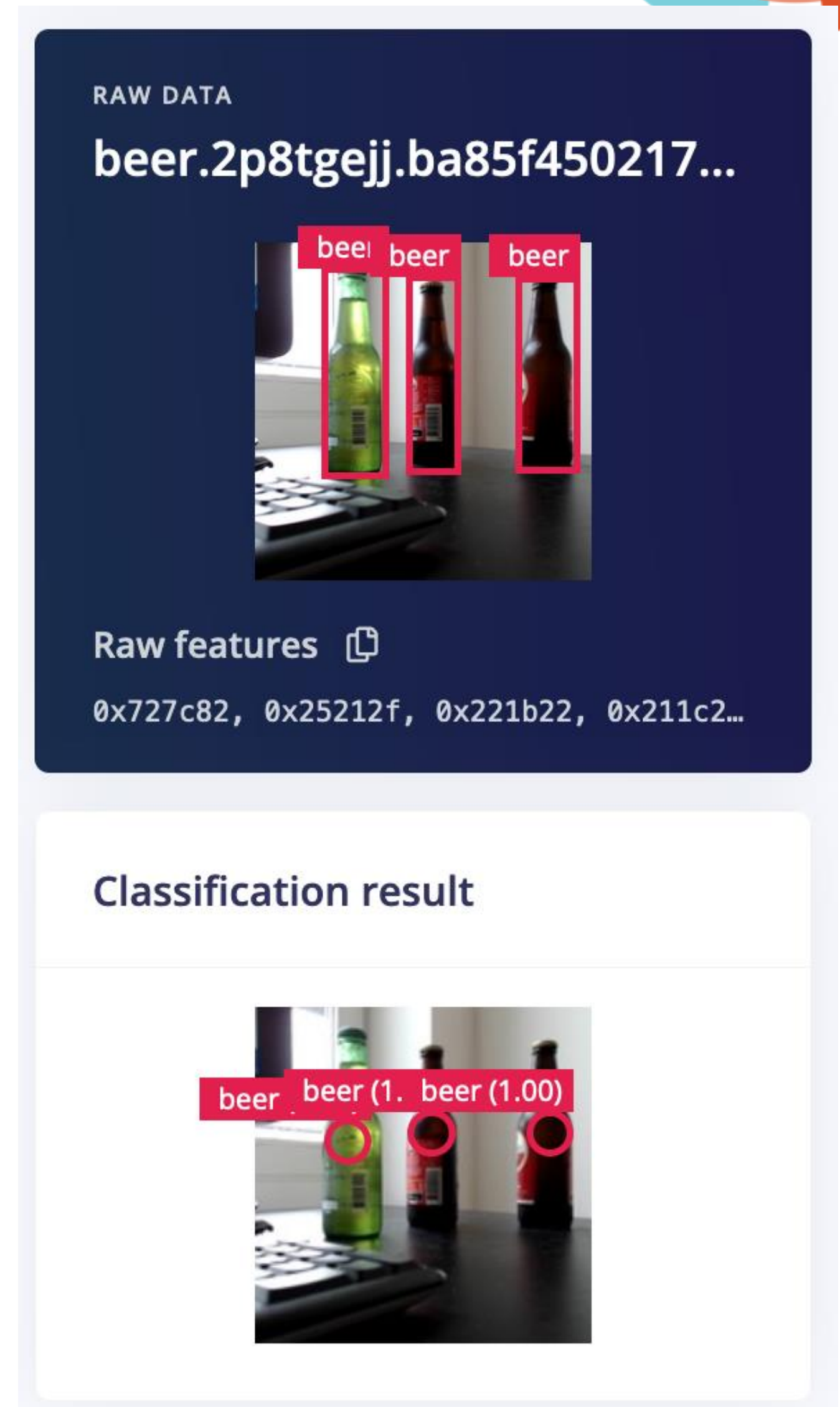
96 x 96 input => 12 x 12 output

Bounding boxes are an implementation detail

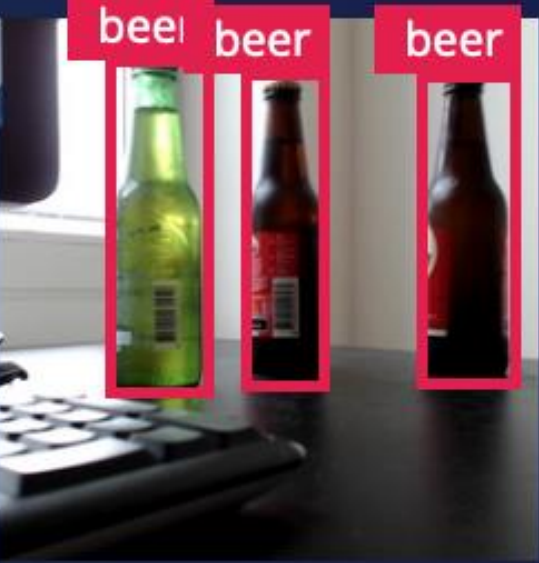
Most of the time you just want to know **where** and **how many** objects there are


FOMO trains on the centroid of an object, w/ loss function allowing some error

Convolutional network, so will still look around the object, but lot less error prone against background




RAW DATA
beer.2p8tgejj.ba85f450217...



Raw features 
0x727c82, 0x25212f, 0x221b22, 0x211c2...

Classification result



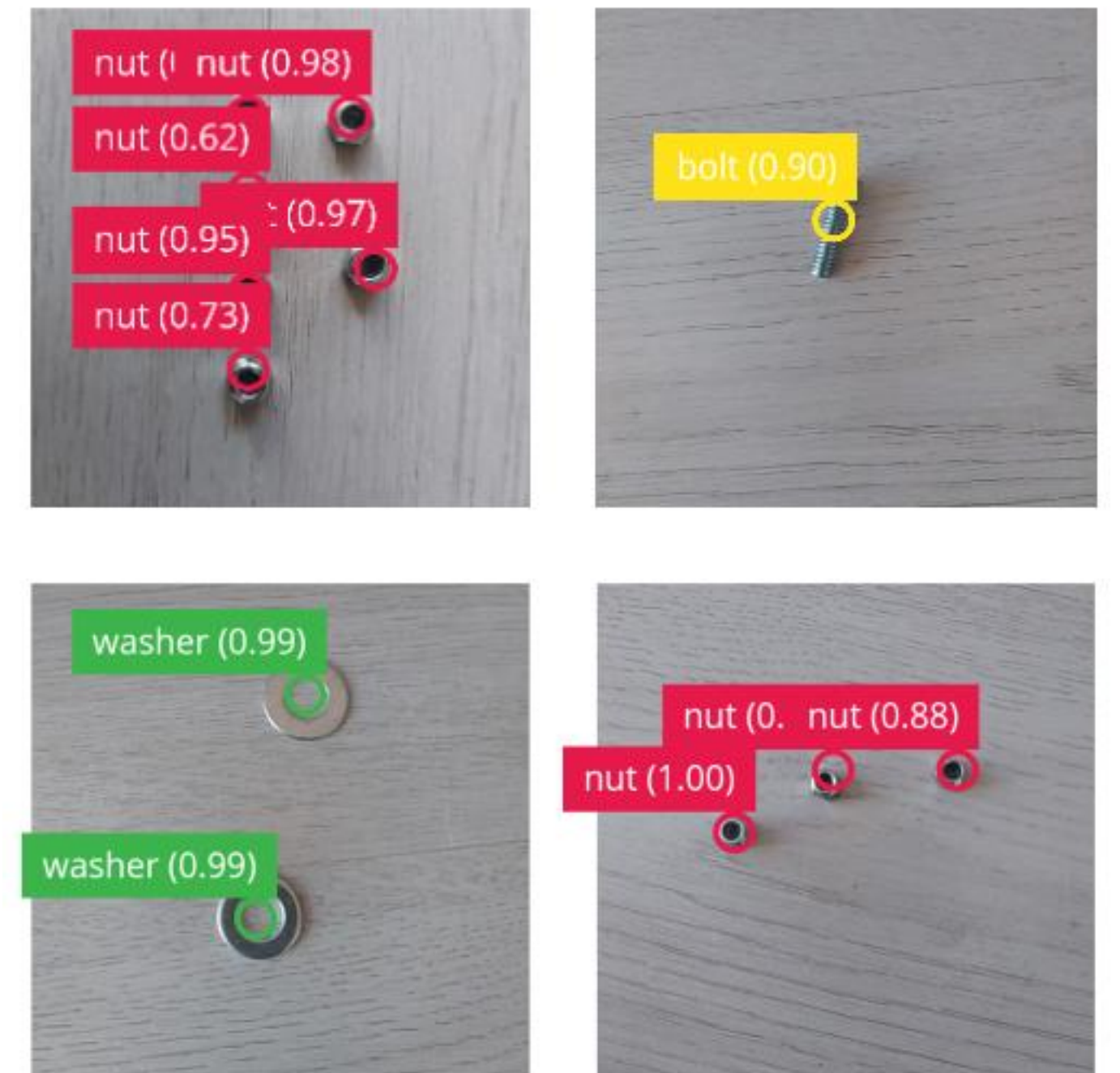
"Equal to an object detector, but bounding boxes are fixed and always square"

Constrained object detection for constrained problems

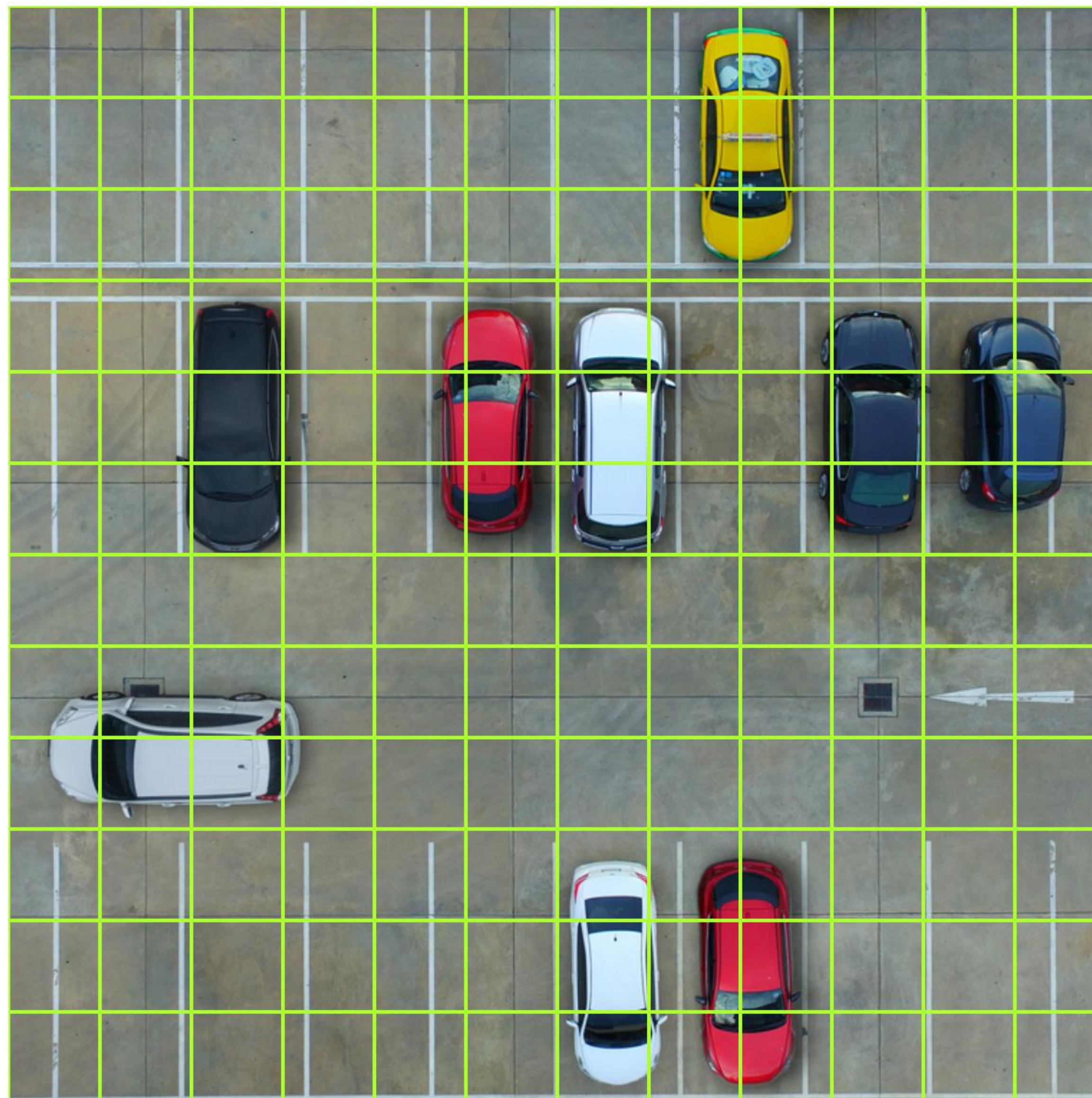
Works best with similar sized objects, at similar distances

One activation per heat map cell, so objects overlapping each other might be fused together

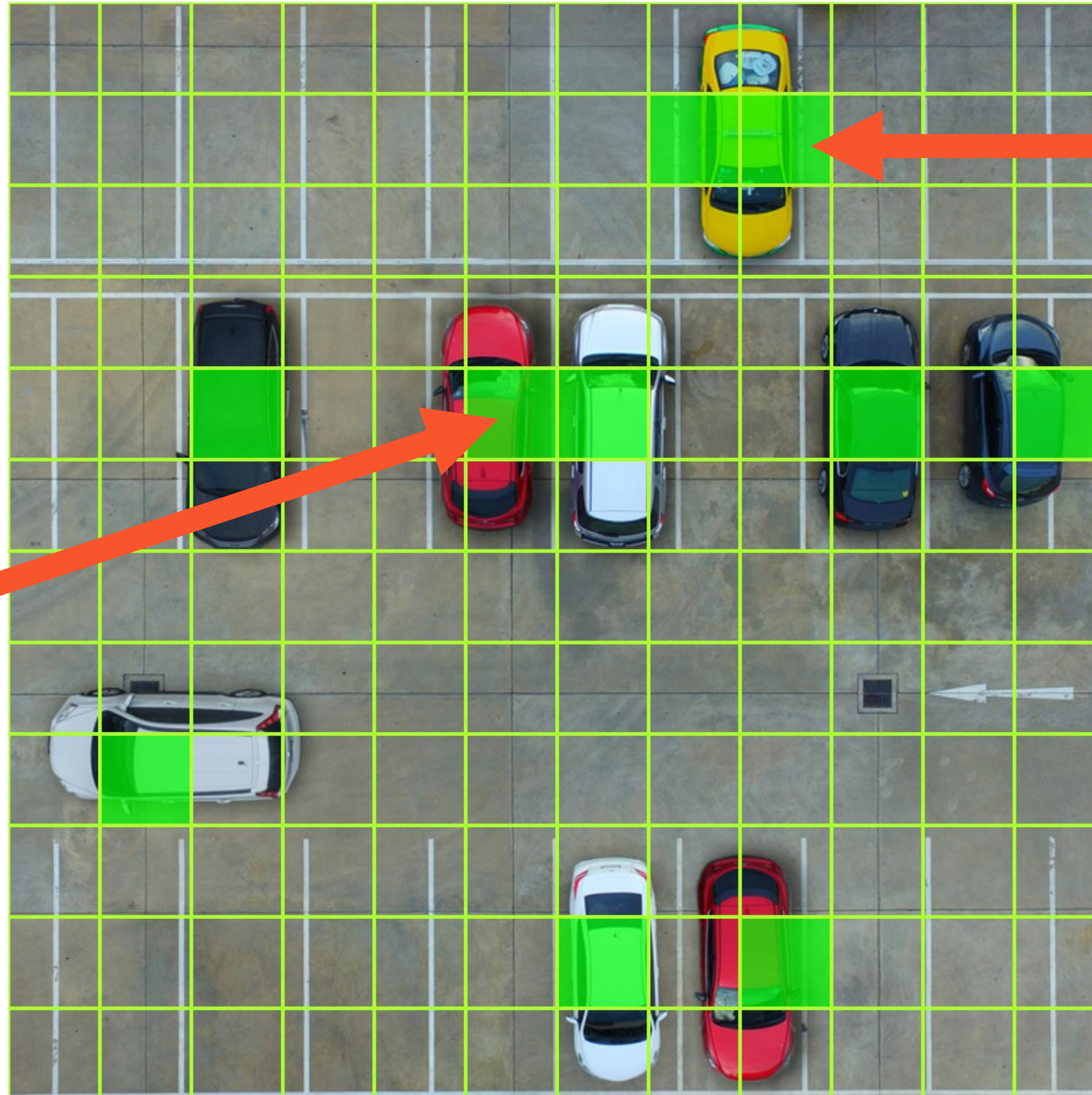
Few classes? Fine. 100s of objects across 60 classes? Use YOLOv5.



Overlapping Objects?



Fixed and Always Square



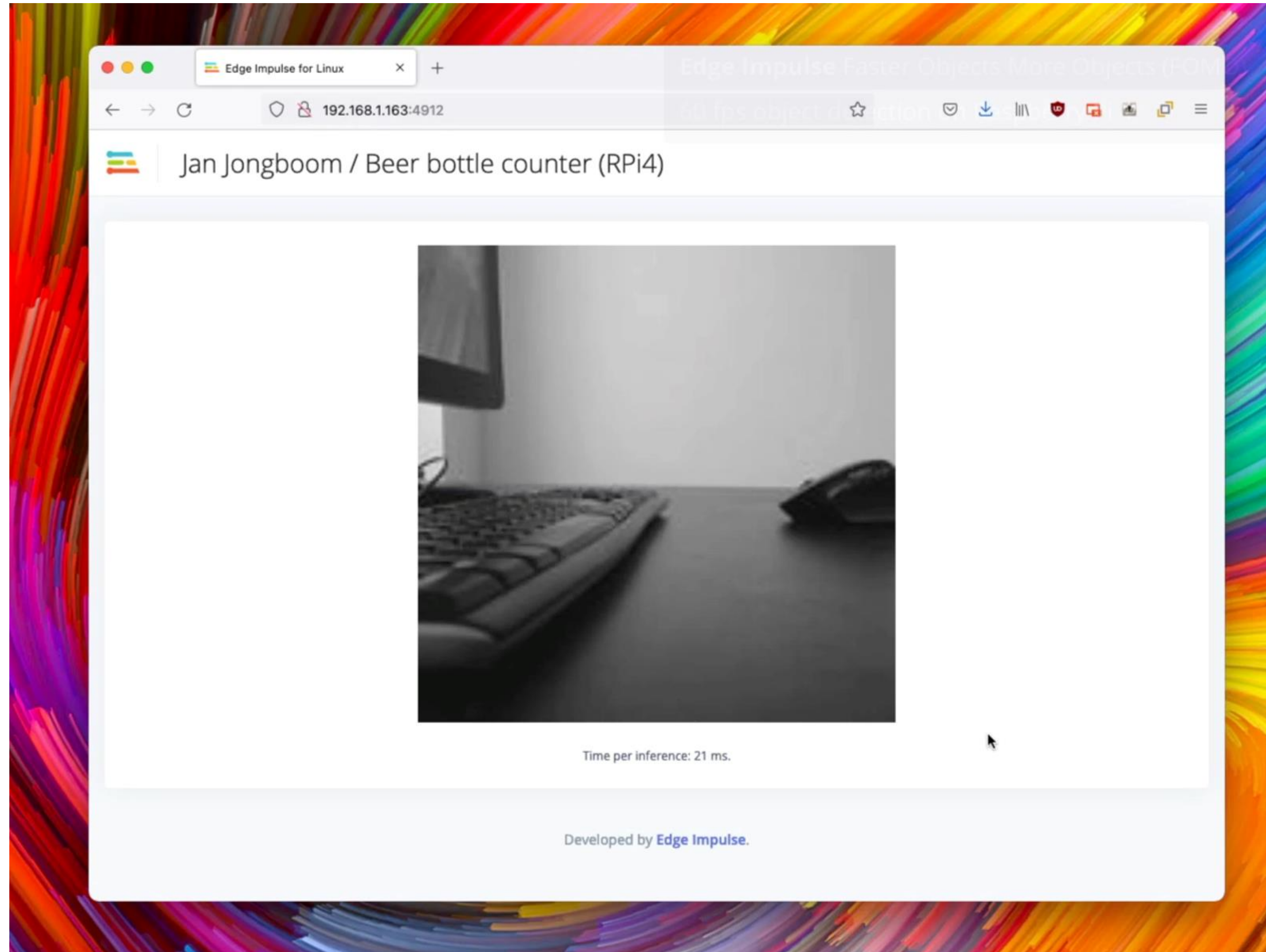
*Potential issue,
merge cells?*

*But... merging leads
to issue here*

Centroids Should Have Meaning



Demo (Rpi4 on CPU, 160x160 input, 60 fps)



Demo (Cortex-M7, 96x96 input, 28 fps)



The screenshot displays the OpenMV IDE interface. On the left, a code editor shows a Python script named 'helloworld_1.py' with the following content:

```
1 # Edge Impulse - OpenMV Object Detection Example
2
3 import sensor, image, time, os, tf, math, uos, gc, utime
4
5 sensor.reset() # Reset and initialize the sensor.
6 sensor.set_pixformat(sensor.RGB565) # Set pixel format to RGB565 (or GRAYSCALE)
7 sensor.set_framesize(sensor.QVGA) # Set frame size to QVGA (320x240)
8 sensor.set_windowing((240, 240)) # Set 240x240 window.
9 sensor.skip_frames(time=2000) # Let the camera adjust.
10 sensor.set_vflip(False)
11 sensor.set_hmirror(True)
12 sensor.set_transpose(True)
13
14
15 net = None
16 labels = None
17 min_confidence = 0.5
18
19 try:
20     # Load built in model
21     labels, net = tf.load_builtin_model('trained')
22 except Exception as e:
23     raise Exception(e)
24
25 colors = [ # Add more colors if you are detecting more than 7 types of classes
26     (255, 0, 0),
27     (0, 255, 0),
28     (255, 0, 255),
29     (0, 0, 255),
30     (255, 0, 255),
31     (0, 255, 255),
32     (255, 255, 255),
33 ]
34
35 clock = time.clock()
36 while(True):
37     clock.tick()
38
39     img = sensor.snapshot()
40
41     # detect() returns all objects found in the image (splitted out per class)
42     # we skip class index 0, as that is the background, and then draw circles
43     # of our objects
44
45     start = utime.ticks_ms()
46
47     for i, detection_list in enumerate(net.detect(img, thresholds=[(math.ceil(
48         if (i == 0): continue # background class
49         if (len(detection_list) == 0): continue # no detections for this class
50
51         print("***** %s *****" % labels[i])
52         for d in detection_list:
53             [x, y, w, h] = d.rect()
54             center_x = math.floor(x + (w / 2))
55             center_y = math.floor(y + (h / 2))
56             print("x %d y %d" % (center_x, center_y))
57             img.draw_circle((center_x, center_y), 0, color=colors[i], thickness
58
59     end = utime.ticks_ms()
60
61     print('time per inference', end-start)
62
63     print(clock.fps(), "fps", end="\n\n")
64
```

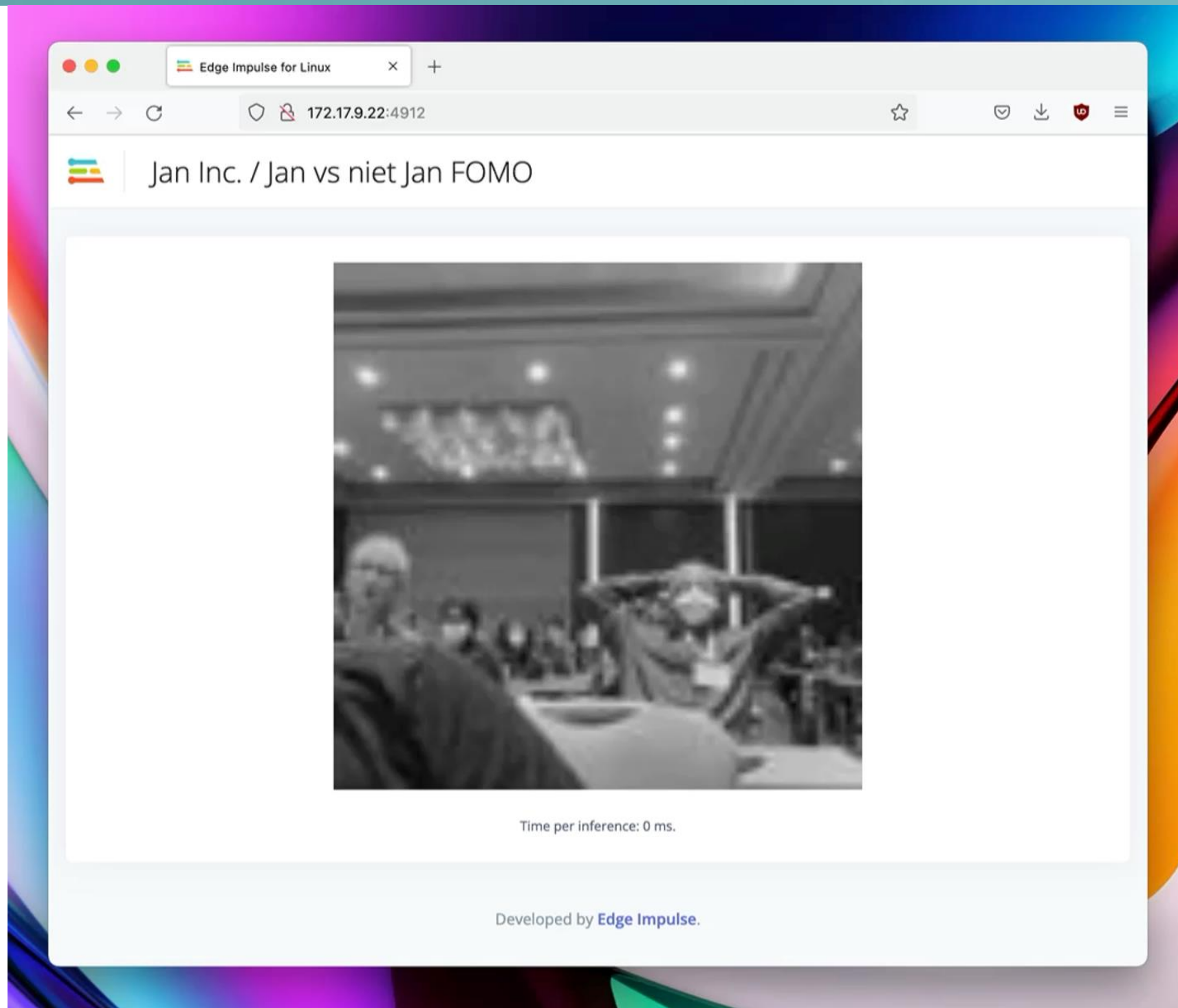
Below the code editor is a Serial Terminal window showing the following output:

```
28.6026 fps
time per inference 27
28.6026 fps
time per inference 27
28.6026 fps
time per inference 27
28.6026 fps
```

The main window of the IDE displays a live video feed from a camera. The feed shows an indoor scene with a large window and a person walking in the background. The video feed is overlaid with a dark overlay, and the text 'helloworld_1.py' is visible in the top right corner of the IDE window.

At the bottom of the IDE window, the status bar displays the following information: Board: N1CLAV, Sensor: GC2145, Firmware Version: 4.3.1 - [latest], Serial Port: cu.usbmodem396A356830311, Drive: /Volumes/NO NAME, FPS: 11,2

Demo (M1, 160x160, 10000 fps)



Other Cool Features of FOMO



Add-on to any convolutional image network (incl. transfer learning models), so highly configurable

Fully convolutional, just the ratio is set

Can count objects

Can be a segmentation model

Performs much better on small objects than YOLOv5 or MobileNet SSD





Cortex-M7 @ 480MHz: **30 fps** (96x96 MobileNetV2 a=0.1)

Raspberry Pi 4: **60 fps** (160x160 MobileNetV2 a=0.35)

Himax DSP @ 400MHz: **14 fps** (96x96 MobileNetV2 a=0.35)

Cortex-M4F @ 156MHz: **5 fps** (96x96 MobileNetV2 a=0.05)

My Macbook: **1000 fps :-)**

(Can be bolted on other CNNs, e.g. MobileNetV1)

Getting Started



- FOMO is available today for free: <https://edgeimpulse.com/fomo>
- Very wide range of dev boards, from Cortex-M4F to Jetson Nano
- Deploy to any device that has a C++ compiler
- Or use your phone!
- Be one of today's 194 new projects!



Vision is such a cool sense

Classification is cool, locality and count matters

Want a 30x increase in performance? Use FOMO!

edgeimpulse.com

Questions?



Full docs:

<https://docs.edgeimpulse.com>

FOMO

<https://docs.edgeimpulse.com/docs/fomo-object-detection-for-constrained-devices>

We're hiring!

<https://edgeimpulse.com/careers>

More questions:

forum.edgeimpulse.com / jan@edgeimpulse.com

