

The Beginner-Friendly Guide To

# Esp32 Camera Object Detection (FOMO)

The world-best guide to run Edge Impulse FOMO Object Detection on the Esp32 Camera with little effort. Even if you're a beginner



## # Introduction

Object detection is the task of detecting an object of interest inside an image. Until a couple years ago, this task was a matter of powerful computers due to the complexity of models and the prohibitive number of math operations to perform.

Thanks to platforms like [Edge Impulse](#), however, the entry barrier for beginners has become much lower and it is now possible to:

1. easily train an object detection model in the cloud
2. (not so easily) deploy this model to the Esp32 camera

Sadly, the Esp32 camera seems not to be a first-class citizen on the Edge Impulse platform and the (few) documentation and tutorials available online miss a lot of fundamental pieces.

The purpose of this post is to **help you develop and deploy your very own object detection model to your Esp32 camera with detailed, easy to follow steps**, even if you're a beginner with Edge Impulse or Arduino programming.

Let's start!

## # Setting up the Environment

To follow this tutorial, you will need the following hardware / software:

1. an Esp32 camera. The model doesn't matter, but it needs at least 4 Mb of PSRAM.  
(AiThinker, M5Stack, EspEye all work fine)
2. a free account on [Edge Impulse](#)
3. the [EloquentEsp32Cam library](#) version 1.1.1 (install from the Arduino IDE Library Manager)
4. ~30 minutes of your time

If you've never used Edge Impulse, I suggest you watch a couple video tutorials online before we start, otherwise you may get lost later on (there exists an [official tutorial playlist on YouTube](#)).

If you never used the EloquentEsp32Cam library, it's a library I wrote to make it painfully easy to get the most out of the Esp32 camera boards with very little effort. This very post is an extract of the eBook I've been writing around this library. You can check [the eBook dedicated page here](#).





## # Object Detection using ESP32 Camera

To perform object detection on our Esp32 camera board, we will follow these steps:

1. Use EloquentEsp32Cam to collect images
2. Use Edge Impulse to label the images
3. Use Edge Impulse to train the model
4. Use Edge Impulse to export the model into an Arduino library
5. Use EloquentEsp32Cam to run the model

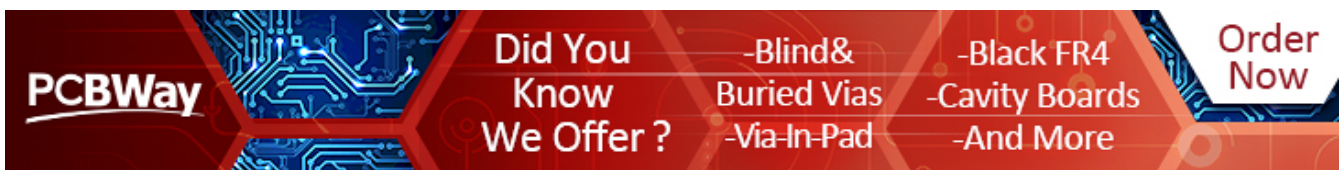
Before this post existed, steps 1. and 5. were not as easy as they should've been.

I invite you to stop reading this post for a minute and go search on Google *Esp32 -cam object detection* : read the first 4-5 tutorials on the list and tell me if you're able to deploy such a project.

I was not.

And I bet you neither.

Enough words, it's time to start tinkering.



### # 1. Collect images from the Esp32 camera

This is the first part where things are weird in the other tutorials. Many of them suggest that you use images from Google to train your ML model. Some others suggests to collect data using your smartphone.

How using random images from the web or from a 40 MP camera can help train a good model that will run on a 2\$ camera hardware is out of my knowledge.

That said, I believe that to get good results, we need to collect images from our own Esp32 camera board.

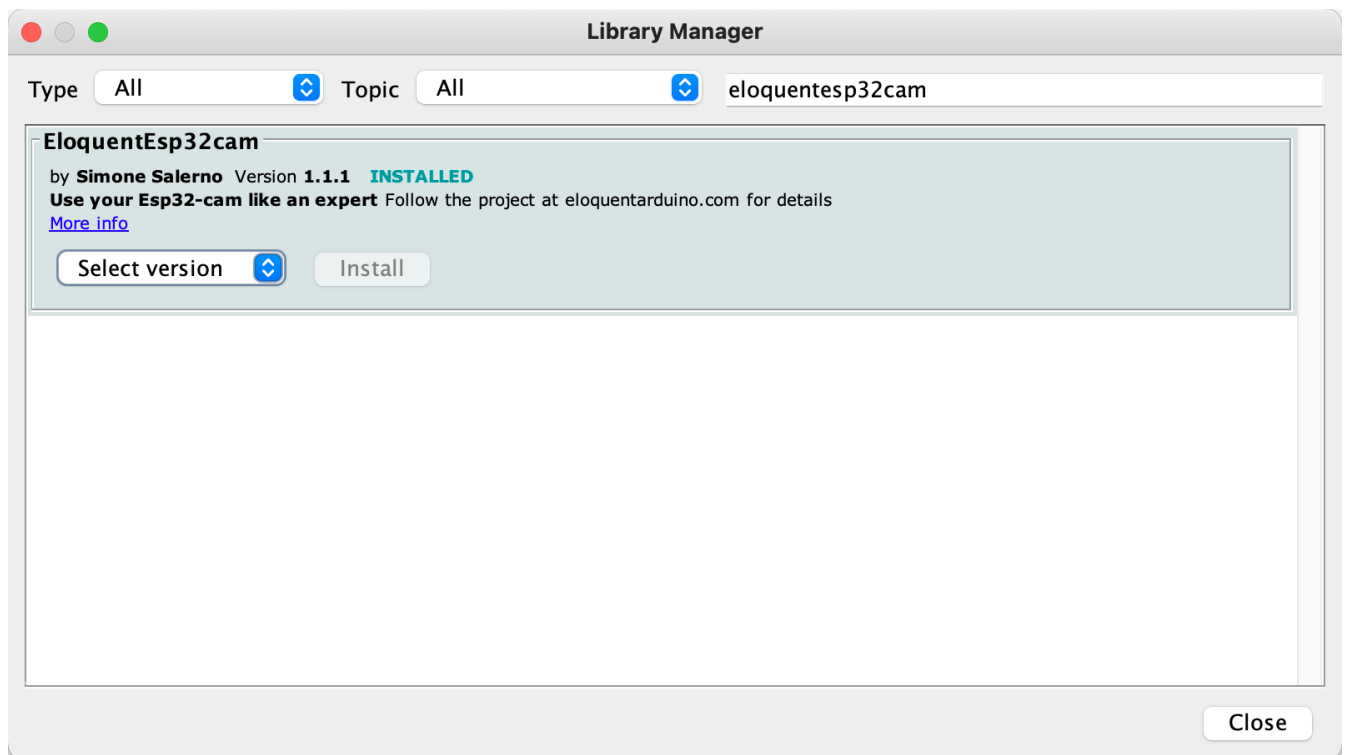
**This used to be hard**, but fear no more. I promise you will complete this task in a matter of minutes, thanks to the tools I created for you.

## # 1.A Upload the Image Collection Sketch to your Board

---

You need to install the `EloquentEsp32Cam` library from the Library Manager. **It is mandatory that you install version 1.1.1.**

*(later versions may work, but is not guaranteed)*



*Install the EloquentEsp32Cam library*

After you installed the EloquentEsp32Cam library, navigate to `File > Examples > EloquentEsp32Cam > 26_Collect_Images` and upload the sketch to your board.

*(yes, this is the 26th example of the aforementioned eBook I wrote. There are a lot of other interesting examples to checkout)*

In case you want to know what's inside, here's the code.

Copy



```
1 // 26_Collect_Images.ino
2 #define MAX_RESOLUTION_XGA 1
3
4 /**
5  * Run a development HTTP server to capture images
6  * for TinyML tasks
7  */
8
9 #include "esp32cam.h"
10 #include "esp32cam/http/FomoImageCollectionServer.h"
11
12 using namespace Eloquent::Esp32cam;
13
14
15 Cam cam;
16 Http::FOM0::CollectImagesServer http(cam);
17
18
19 void setup() {
20     Serial.begin(115200);
21     delay(3000);
22     Serial.println("Init");
23
24     /**
25      * Replace with your camera model.
26      * Available: aithinker, m5, m5wide, wrover, eye, ttgoLCD
27      */
28     cam.aithinker();
29     cam.highQuality();
30     cam.highestSaturation();
31     cam.xga();
32
33     while (!cam.begin())
34         Serial.println(cam.getErrorMessage());
35
36     // replace with your SSID and PASSWORD
37     while (!cam.connect("SSID", "PASSWORD"))
38         Serial.println(cam.getErrorMessage());
39
40     while (!http.begin())
41         Serial.println(http.getErrorMessage());
42
43     Serial.println(http.getWelcomeMessage());
44     cam.mDNS("esp32cam");
45 }
46
47
48 void loop() {
```

```
49 | http.handle();  
50 | }
```

**Don't forget to replace SSID and PASSWORD with your WiFi credentials!**

Once the upload is done, open the Serial Monitor and take note of the IP address of the camera.

## # 1.B Create your Shooting Setup

---

Since this is our first project on Esp32 camera object detection, I want you to succeed and get good results easy. Even though this *could be* not strictly necessary, I suggest you create a "fixed shooting setup": either fix your board to a table with some tape, or use a cheap camera mounting. Then, point it to a flat, monochrome surface (a wall, a closet...): the TinyML model will learn better the objects features.

Also try to keep a good illumination in the room: Esp32 camera performs poorly in low light conditions.

For each object you want to recognize (start with 1 or 2 the first time), you will put that object in front of the camera and, while the capturing is going on, you will move it a bit, rotate...

This ensure the model will be able to learn to detect the objects in different positions and orientations.

## # 1.C Collect Images from the Browser

---

Here's where things start to become exciting. You will collect the images for the ML model from the browser. No additional software is required.

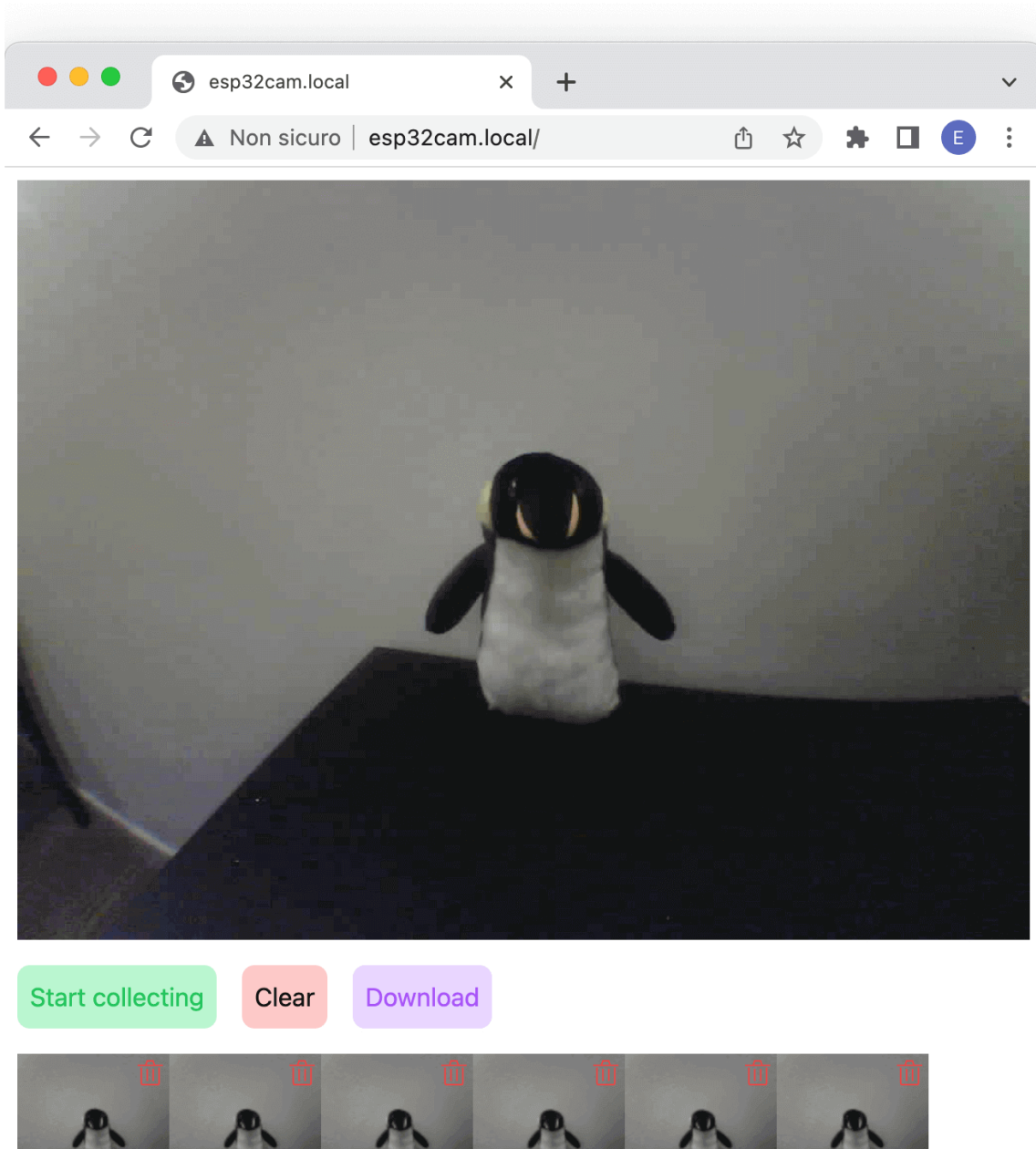


Pro tip: if your router supports mDNS (most do), you can enter `esp32cam.local` as address instead of the IP address. This way you don't have to open the Serial Monitor every time you power up the board (IP address may change, mDNS name stays the same).

Open a new tab in the browser and enter the IP address of your Esp32 camera (or navigate to <http://esp32cam.local> if you used mDNS). You can preview the real-time video stream from the camera. Use this preview to setup the correct position and orientation.

When you're ready to start, click **Start collecting** and the camera frames will start appearing below on the page. (continue reading before doing anything).

*You may experience a bit of lag: don't worry, just keep collecting.*





### *Image collection demo in the browser*

Images will be 96x96 because that's a recommended size for Edge Impulse image models. We're going to actually use 48x48 images during training since 96x96 is too large to fit into our Esp32-cam board, but it's better to have a larger resolution available if we decide to move to a more capable board later on.

Now follow these *EXACT* steps (they guarantee you won't have troubles later on):

1. put nothing in front of the camera. Collect 15-20 images. Pause, download and clear
2. put the first object in front of the camera. Collect 30-40 images while moving the object around. Pause, download and clear
3. repeat 2. for each object you have

After you finish, you will have one zip of images for each object + one for "no object / background".

Extract the zips and move to the next step.

## # 2. Use Edge Impulse to Label the Images

For object detection to work, we need to label the objects we want to recognize.

There are a few tools online, but Edge Impulse has one integrated that works good enough for our project. Register a free account on [edgeimpulse.com](https://edgeimpulse.com) if you don't have one already and create a new project.

Create a new project, name it something like `esp32-cam-object-detection`, then choose `Images > Classify multiple objects > Import existing data`.

Now follow these *EXACT* steps to speed up the labelling:

1. click `Select files` and select all the images in the "no object / background" folder;

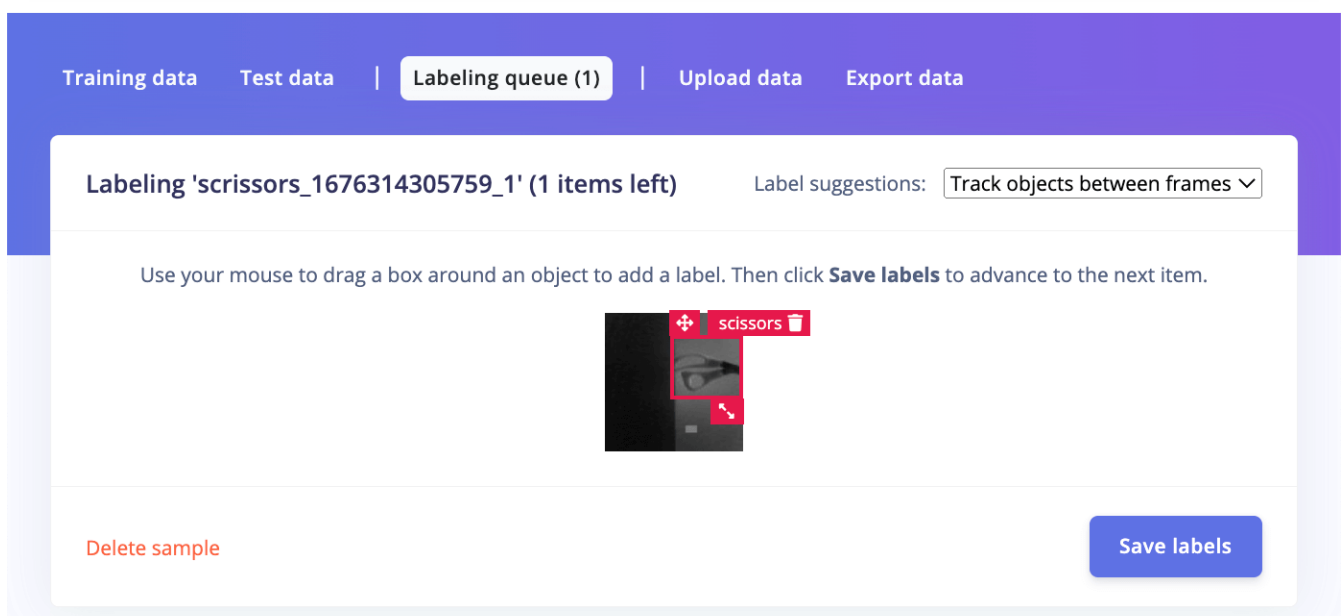


check *Automatically split between training and testing*; then hit *Begin upload*

2. click *Labelling queue* in the bar on top
3. always click *Save labels* **without doing anything!** (since there's no object in these images, we'll use them for background modelling)
4. once done for all the images, go back to *Upload data* in the bar on top
5. click *Select files* and select all the images in the first object folder. **Only upload the images of a single folder at a time!!**. Check *Automatically split between training and testing*; then hit *Begin upload*
6. go to *Labelling queue* in the top bar and draw the box around the object you want to recognize. On the right, make sure *Label suggestions: Track objects between frames* is selected
7. label all the images. Make sure to fix the bounding box to fit the object while leaving a few pixels of padding
8. repeat 4-7 for each object



If you upload all the images at once, the labelling queue will mix the different objects and you will lose a lot more time to draw the bounding box. Be smart!



### Edge Impulse Labelling Queue example

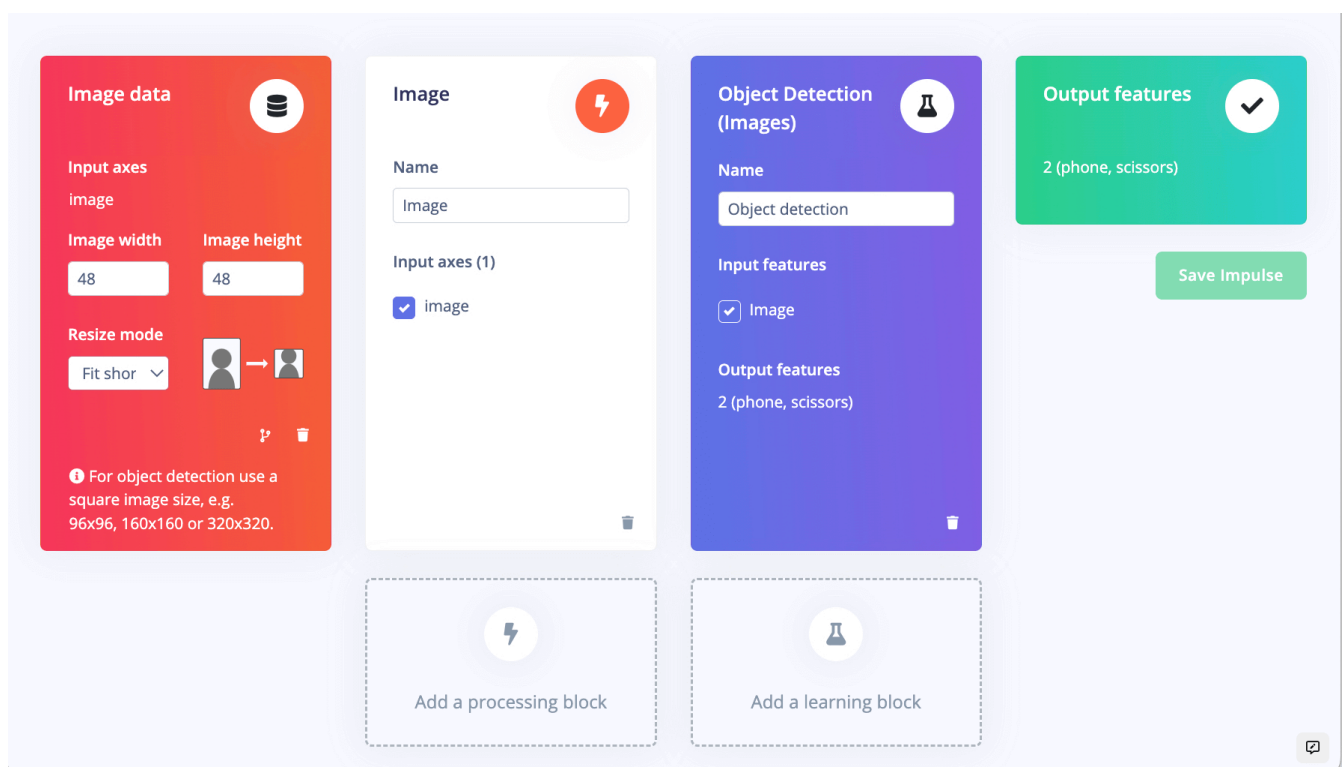
At this point, you will have all the data you need to train the model.

## # Use Edge Impulse to Train the Model

If you've ever used Edge Impulse, you know this part is going to be pretty easy.

Navigate to *Impulse design* on the left menu; enter *48* as both *image width* and *image height*, then select *Fit shortest axis* as resize mode.

Add the *Image* processing block, the *Object detection* learning block and save the impulse.



### Edge Impulse Impulse design

Now navigate to *Impulse design > Image* on the left, select *Grayscale* as color depth and *Save parameters*. Next click on *Generate features*. It should take less than a minute to complete, depending on the number of images.

Now navigate to *Impulse design > Object detection*, set the number of training

cycles to 30 , training learning rate to 0.005 , click on *Choose a different model* right below the FOMO block and select *FOMO (Faster Objects, More Objects) MobileNetV2 0.1* . This is the smallest model we can train and the only one that fits fine on the Esp32-camera.

### Neural Network settings

Training settings

Number of training cycles ?

15

Learning rate ?

0.01

Validation set size ?

20


%

Data augmentation ?

☒

Neural network architecture

Input layer (2,304 features)



FOMO (Faster Objects, More Objects) MobileNetV2 0.1

Choose a different model

Output layer (2 classes)

Start training

*Edge Impulse training parameters*

Hit *Start training* and wait until it completes. It can take 2-3 minutes depending on the number of images.

Model

Model version: ?

Quantized (int8) ▾

### Last training performance (validation set)

 **F1 SCORE**  
**100.0%**

### Confusion matrix (validation set)

	BACKGROUND	PHONE	SCISSORS
BACKGROUND	100%	0%	0%
PHONE	0%	100%	0%
SCISSORS	0%	0%	100%
F1 SCORE	1.00	1.00	1.00

### On-device performance

 **INFERRING TIME**  
**241 ms.**

 **PEAK RAM USAGE**  
**75,9K**

 **FLASH USAGE**  
**59,4K**

*Edge Impulse object detection confusion matrix*



To get accurate estimates of inferencing time and memory usage as the above image shows, be sure to select "Espressif ESP-EYE" as target board in the top-right corner of the Edge Impulse page.

If you're satisfied with the results, move to the next step. If you're not, you have to:

1. collect more / better images. Good input data and labelling is a critical step of Machine Learning
2. increase the number of training cycles (don't go over **50**, it's almost useless)
3. decrease the learning rate to **0.001**. It may help, or not

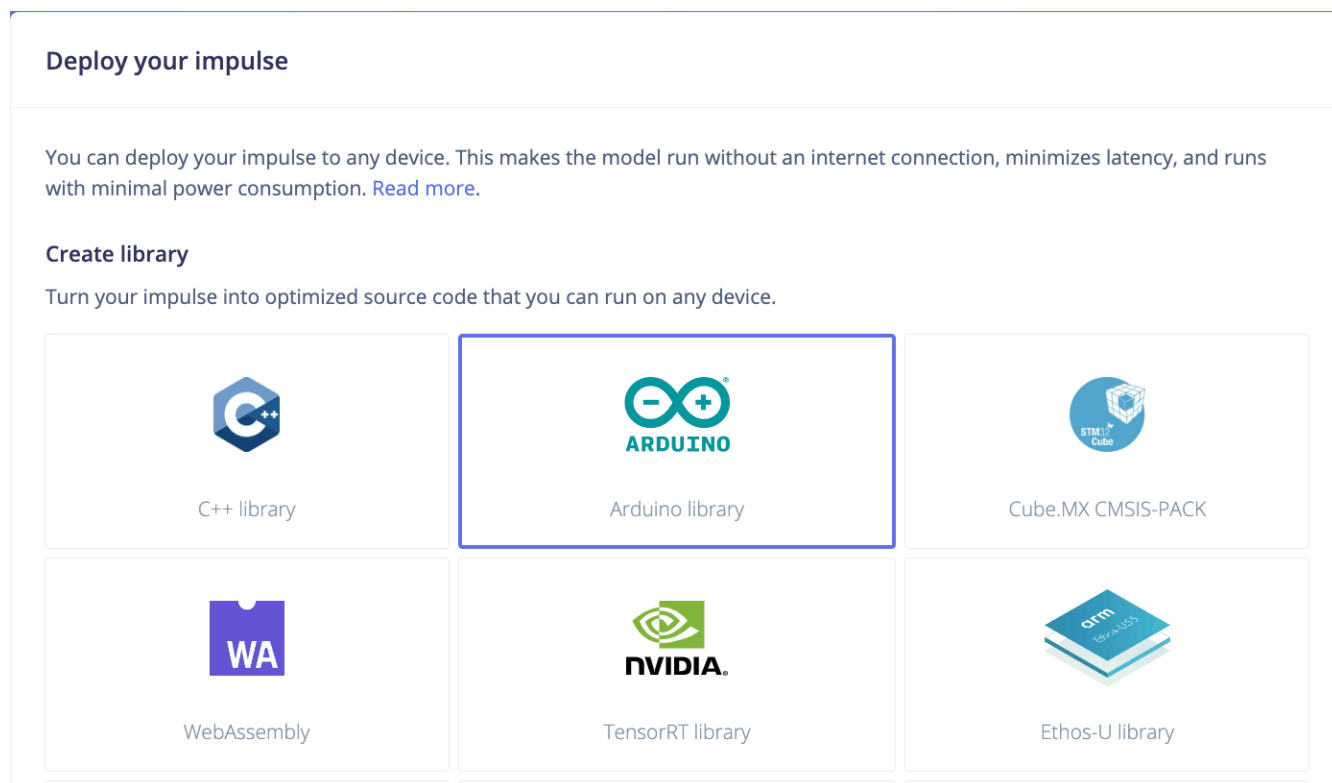
Now that all looks good, it's time to export the model.

## # Use Edge Impulse to Export the Model into

# an Arduino Library

This is the shortest step.

Navigate to **Deployment** on the left menu, select **Arduino Library**, scroll down and hit **Build**. A zip containing the model library will download.



*Edge Impulse deployment to Arduino library*

Keep this ready for the next step.

## # 5. Use EloquentEsp32Cam to Run the Object Detection Model

This last part is the one that struggled me the most.

As a beginner, **you won't find any good tutorial on how to deploy an Edge Impulse FOMO model to the Esp32-camera on the web.**

Until now, of course!

I'll detail these steps as much as possible so you will have a hard time failing.

## # 5.A Create a new Sketch

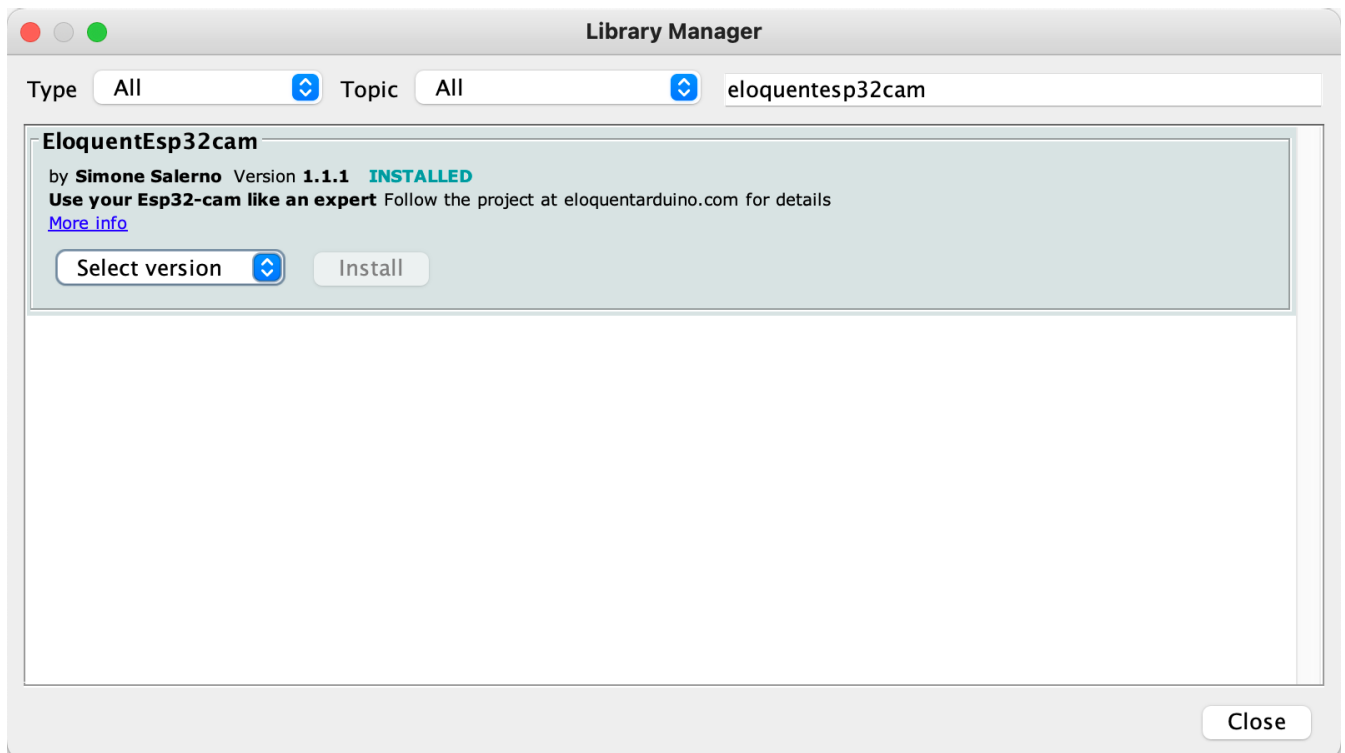
---

Open the Arduino IDE and create a new sketch. Select `Esp32 Dev Module` as board and enable external PSRAM, if available on your board.

## # 5.B Install the Libraries

---

You need to install the `EloquentEsp32Cam` library from the Library Manager. **It is mandatory that you install version 1.1.1.**



*Install the EloquentEsp32Cam library*

Next navigate to `Sketch > Include library > Add .zip library` and select the zip downloaded from Edge Impulse.

## # 5.C Copy the Object Detection Sketch

---

Copy the following sketch contents.

```
1 // 27_EdgeImpulse_FOM0.ino
2 #define MAX_RESOLUTION_VGA 1
3
```

Copy



```
4  /**
5   * Run Edge Impulse FOMO model on the Esp32 camera
6   */
7
8  // replace with the name of your library
9  #include <esp32-cam-object-detection_inferencing.h>
10 #include "esp32cam.h"
11 #include "esp32cam/tinyml/edgeimpulse/FOMO.h"
12
13
14 using namespace Eloquent::Esp32cam;
15
16 Cam cam;
17 TinyML::EdgeImpulse::FOMO fomo;
18
19
20 void setup() {
21     Serial.begin(115200);
22     delay(3000);
23     Serial.println("Init");
24
25     cam.aithinker();
26     cam.highQuality();
27     cam.highestSaturation();
28     cam.vga();
29
30     while (!cam.begin())
31         Serial.println(cam.getErrorMessage());
32 }
33
34 void loop() {
35     if (!cam.capture()) {
36         Serial.println(cam.getErrorMessage());
37         delay(1000);
38         return;
39     }
40
41     // run FOMO model
42     if (!fomo.detectObjects(cam)) {
43         Serial.println(fomo.getErrorMessage());
44         delay(1000);
45         return;
46     }
47
48     // print found bounding boxes
49     if (fomo.hasObjects()) {
50         Serial.printf("Found %d objects in %d millis\n", fomo.count(), fo
51
```

```

52         fomo.forEach([](size_t ix, ei_impulse_result_bounding_box_t bbox)
53             Serial.print(" > BBox of label ");
54             Serial.print(bbox.label);
55             Serial.print(" at (");
56             Serial.print(bbox.x);
57             Serial.print(", ");
58             Serial.print(bbox.y);
59             Serial.print("), size ");
60             Serial.print(bbox.width);
61             Serial.print(" x ");
62             Serial.print(bbox.height);
63             Serial.println();
64         });
65     }
66     else {
67         Serial.println("No objects detected");
68     }
69
70 }

```

Even if you've never used the `EloquentEsp32Cam` library and you're a beginner, you should still be able to get what's going on in the sketch: at each loop, we take a photo, run the object detection model and for each object found we print its label, position and size.

## # 5.D (Optionally) Fix the Edge Impulse library

---

After a few days I wrote this tutorial, the Edge Impulse downloaded library stopped compiling out of the box.

The compiler complains about `std::fmin` and `std::fmax`.

If you get a similar error, here's how to fix:

1. navigate to `Arduino root > libraries > esp32-cam-object-detection-inferencing > src > edge-impulse-sdk > tensorflow > lite > micro > tensor_utils_common.cpp`
2. search for `const double rmin`. For me, it is at line 84
3. replace `const double rmin = std::fmin(0, *minmax.first);` with `const double rmin = (*minmax.first) > 0 ? 0 : (*minmax.first);`



4. on the next line, replace `const double rmax = std::fmax(0, *minmax.second);` with `const double rmax = (*minmax.second) < 0 ? 0 : (*minmax.second);`

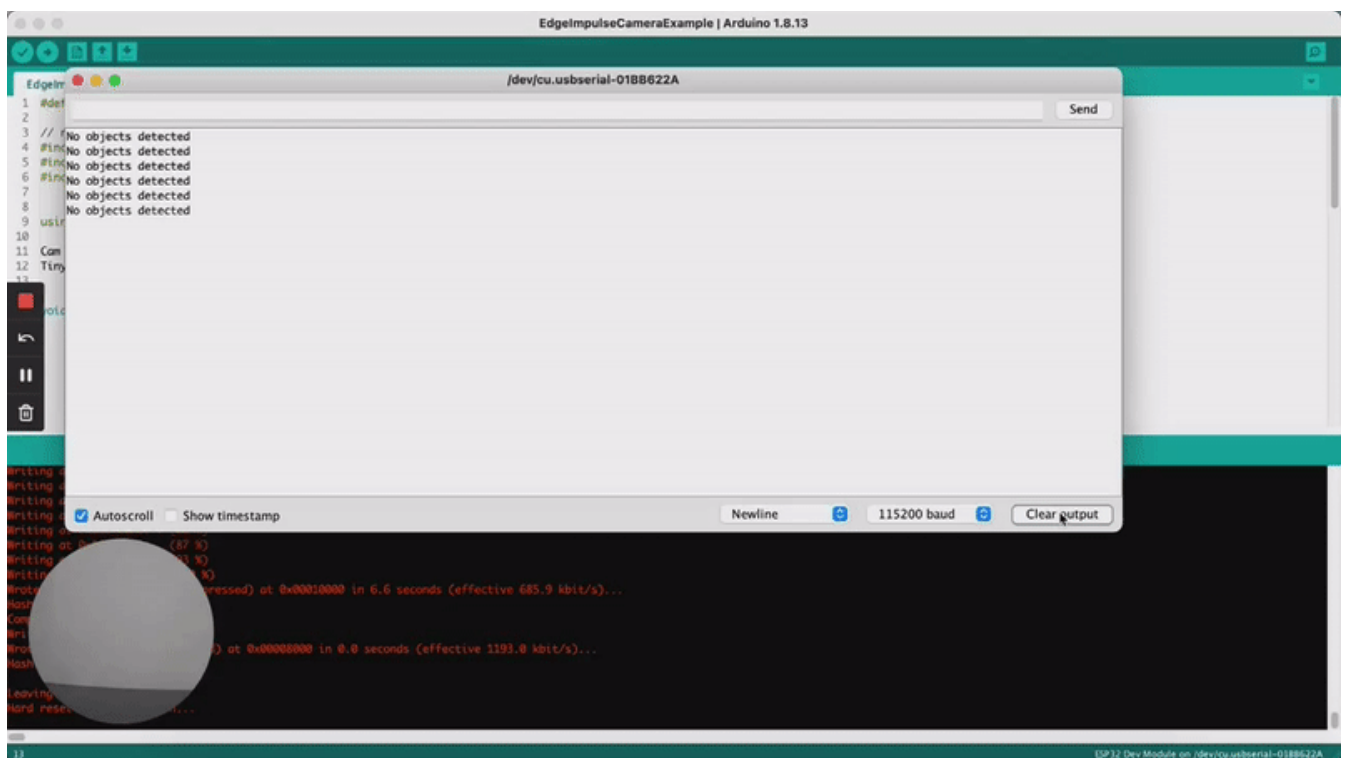
Save, go back to the Arduino IDE and compile again. Now it should succeed.

## # 6.D Run

---

This is going to be the most rewarding step of all this tutorial.

Save the sketch, hit upload, open the Serial Monitor and watch the predictions scrolling while you put your objects in front of the camera.



*Live demo of Esp32 camera object detection (1.5x speed)*

Text may not be easily readable, but the object detection takes ~180 ms to run. This is pretty fast, if you ask me.

## # To Summarize

I walked you step by step on how to **deploy a FOMO object detection model on your Esp32 camera even if you are a beginner**. The steps we implemented are:

1. Collect images using the tool from the EloquentEsp32Cam library

2. Label fast the images in Edge Impulse
3. Train a FOMO object detection model in the cloud
4. Deploy the model back to our Esp32 camera and integrate it into our own sketch

Before this tutorial came out, collecting images and deploying the model back to the Esp32 camera were very hard to implement for a beginner.

All the existing tutorials were very skimmy on these two fundamental steps. No one (and I challenge you to prove me wrong) ever showed how to integrate the FOMO code into an existing sketch without using the (messy) default example.

If you wanted to e.g. blink a LED every time an object was detected, you had to figure it out by yourself by reading throughout all the (verbose) code of the default Edge Impulse example sketch.

Thanks to the `EloquentEsp32Cam` library, you can now easily do this and a lot more!

## Get the video walkthrough in your inbox!

You prefer video over text? No problem, enter your email address and I will send you the entire recorded screencast

**Sign Up**

We use Mailchimp as our marketing platform. By submitting this form, you acknowledge that the information you provided will be transferred to Mailchimp for processing in accordance with their [terms of use](#). We will use your email to send you updates relevant to this website.

## # What's next?

Now that you have the FOMO tool at your disposal, you can create whichever project you like that leverages machine vision on the cheap Esp32 camera board.

You may also want to learn all the other magic tricks that are built inside the `EloquentEsp32Cam` library, such as:


- motion detection
- Telegram messaging
- face detection
- person detection
- line crossing detection
- color blob detection

I wrote an entire eBook dedicated to this.



If you appreciated this tutorial, I will appreciate that you share it on social media.

If you have comments, don't hesitate to use the form below.

**Having troubles?**  
**Ask a question** 

## Related posts



TINYML

Color classification with



LIBRARIES

## Eloquent Edge Impulse for Arduino

An Arduino library to make Edge Impulse neural networks more accessible and easy to use

## TinyML

Get familiar with data collection and pre-processing for Machine Learning tasks



TINYML

## Iris flower classification with TinyML

An introduction to Machine Learning development for Arduino



TINYML

## Attiny Machine Learning

Can an Attiny85 really run Machine Learning? Let's find out in this article!



TINYML

## Separable Conv2D kernels: a benchmark

Real world numbers on 2D convolution optimization strategies for embedded microcontrollers


## Get the video walkthrough in your inbox!

You prefer video over text? No problem, enter your email address and I will send you the entire recorded screencast

Email Address

Sign Up

We use Mailchimp as our marketing platform. By submitting this form, you acknowledge that the information you provided will be transferred to Mailchimp for processing in accordance with their terms of use. We will use your email to send you updates relevant to this website.

**Having troubles?**  
**Ask a question** 

## Eloquent Arduino

Level up your TinyML skills

### TINYML

Arduino WiFi Indoor Positioning

Get started with Arduino Machine Learning

Attiny Machine Learning

Color classification with TinyML

Gesture classification

Iris flower classification with TinyML

Separable Conv2D kernels: a benchmark

TensorFlow Lite on Esp32

### ESP32-CAM

Esp32 and Arduino Person Detection

Esp32-cam Color Tracking

Real-Time image recognition

Motion detection without PIR

Esp32 Camera Object Detection

Mastering the Esp32 Camera

## LIBRARIES

[Eloquent Edge Impulse for Arduino](#)

[EloquentTinyML for Arduino](#)

[MicroMLGen for Python](#)

[TinyMLgen for Python](#)

## PYTHON

[Machine Learning for MicroPython](#)

[Write Arduino sketches from Python](#)

## CONTACTS

[Twitter](#)

[support@eloquentarduino.com](mailto:support@eloquentarduino.com)

© Copyright 2023 Eloquent Arduino. All Rights Reserved.

[Terms](#)   [Privacy](#)