Electrical and Computer Engineering
Erik Jonsson School of Engineering & Computer Science
The University of Texas at Dallas
Dr. Tooraj Nikoubin

Lab 7- Part 1:
TinyML – Image Classification

EEDG/CE 6302 Embedded Systems

Contents

TinyML – Adding sight to your sensors

Project Objectives

- Understand the concept of TinyML.
- A brief understanding of ML algorithms.
- A brief understanding about Espressif ESP-EYE (ESP32)
- Use machine learning to build a system that can recognize objects in your house through a camera task known as *image classification*

Introduction

Machine learning is a subfield of artificial intelligence, which is broadly defined as the capability of a machine to imitate intelligent human behavior. Artificial intelligence systems are used to perform complex tasks in a way that is similar to how humans solve problems.

Tiny machine learning, or TinyML, is an emerging field that is at the intersection of machine learning and embedded systems. An embedded system is a computing device that usually is small, or tiny, that operates with low power, extremely low power. So much so that some of these devices can run for days, weeks, months, sometimes even years on something like a coin cell battery. TinyML is a type of machine learning that shrinks deep learning networks to fit on tiny hardware. It brings together Artificial Intelligence and intelligent devices.

Edge computing brings computation and data storage closer to the origin of data. Majority of the edge devices that are integrated with IoT-based ecosystems are initially designed to collect sensor data and transmission of the data to neighborhood or remote cloud.

In this tutorial, you'll use machine learning to build a system that can recognize objects in your house through a camera - a task known as *image classification* - connected to a microcontroller. Adding sight to your embedded devices can make them see the difference between poachers and elephants, do quality control on factory lines, or let your RC cars drive themselves. In this tutorial you'll learn how to collect images for a well-balanced dataset, how to apply transfer learning to train a neural network and deploy the system to an embedded device.

Espressif ESP-EYE (ESP32)

Espressif ESP-EYE (ESP32) is a compact development board based on Espressif's ESP32 chip, equipped with a 2-Megapixel camera and a microphone. ESP-EYE also offers plenty of storage, with 8 MB PSRAM and 4 MB SPI flash - and it's fully supported by Edge Impulse.

There are plenty of other boards built with ESP32 chip - and of course there are custom designs utilizing ESP32 SoM. Edge Impulse firmware was tested with ESP-EYE and ESP FireBeetle boards, but there is a possibility to modify the firmware to use it with other ESP32 designs.

Use this link (for windows select the CP210x Windows Drivers) to install CP210x USB to UART Bridge Virtual COM Port (VCP) drivers to connect ESP-EYE to your device.

Edge Impulse

It is a cloud service for developing machine learning models in the TinyML targeted edge devices. This supports AutoML processing for edge platforms. It also supports several boards including smart phones to deploy learning models in such devices. Training is done on the cloud platform and the trained model can be exported to an edge device by following a data forwarder enabled path. The impulse can be run in local machine by the help from the in-built C++, Node.js, Python, and Go SDKs. Impulses are also deployable as a WebAssembly library.

Installing Procedure

1. Installing Dependencies

To set this device up in Edge Impulse, you will need to install the following software:

- **Edge Impulse CLI**

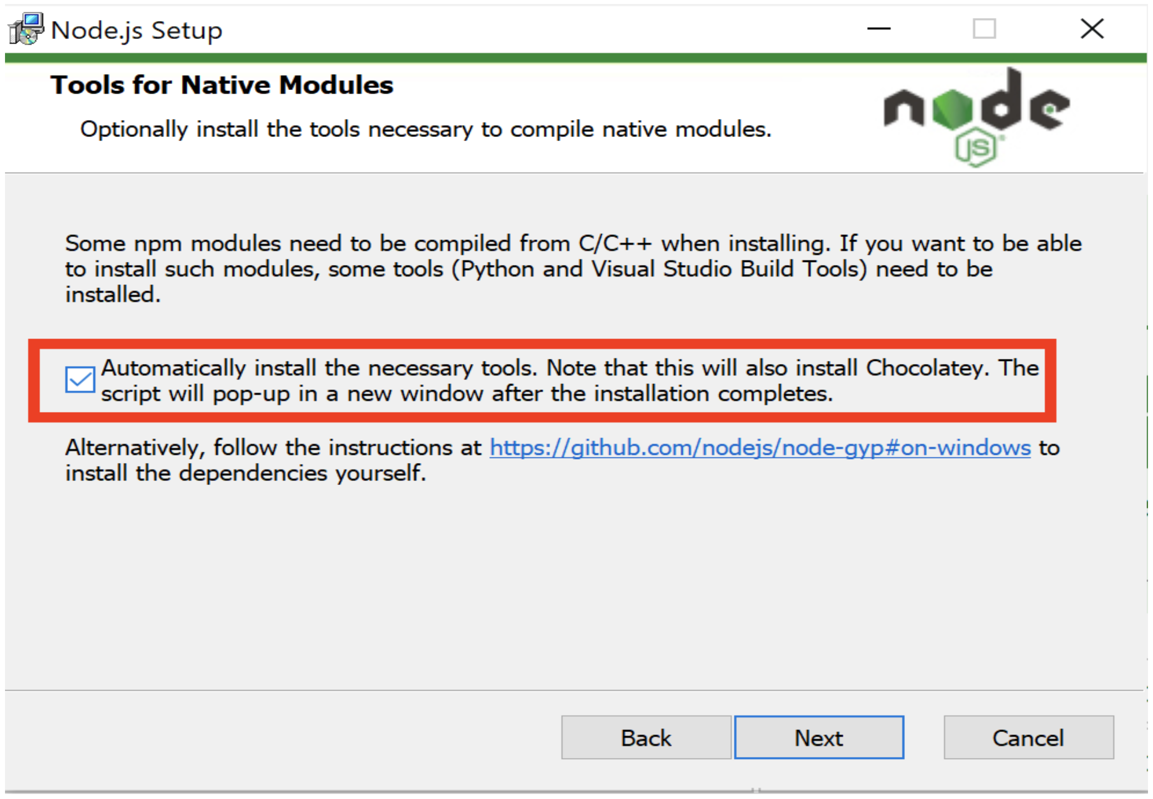
This Edge Impulse CLI is used to control local devices, act as a proxy to synchronize data for devices that don't have an internet connection, and to upload and convert local files. The CLI consists of the following tools:

- **edge-impulse-daemon** - configures devices over serial and acts as a proxy for devices that do not have an IP connection.
- **edge-impulse-uploader** - allows uploading and signing local files.

- **edge-impulse-data-forwarder** - a very easy way to collect data from any device over a serial connection and forward the data to Edge Impulse.
- **edge-impulse-run-impulse** - show the impulse running on your device.
- **edge-impulse-blocks** - create organizational transformation blocks.

Installation – for Windows

1. Install Python 3 on your host computer.
2. Install Node.js v14 or higher on your host computer.
   - For Windows users, install the **Additional Node.js tools** (called **Tools for Native Modules** on newer versions) when prompted.



Tools for Native Modules

Optionally install the tools necessary to compile native modules.

Some npm modules need to be compiled from C/C++ when installing. If you want to be able to install such modules, some tools (Python and Visual Studio Build Tools) need to be installed.

[✓] Automatically install the necessary tools. Note that this will also install Chocolatey. The script will pop-up in a new window after the installation completes.

Alternatively, follow the instructions at https://github.com/nodejs/node-gyp#on-windows to install the dependencies yourself.

Fig 1: Install the additional Node.js tools

3. Install the CLI tools via:

   npm install -g edge-impulse-cli --force
   You should now have the tools available in your PATH.

4. Install the CLI tools via:
5. ESP Tool.
   The ESP documentation website has instructions for macOS and Linux.
6. If you haven't already, create an edge impulse account. Many of our CLI tools require the user to log in to connect with the Edge Impulse Studio.

Connecting To Edge impulse

Connect the development board to your computer

Use a micro-USB cable to connect the development board to your computer.

Update the firmware

The development board does not come with the right firmware yet. To update

1. Download the latest Edge Impulse firmware, and unzip the file.
2. Open the flash script for your operating system (flash_windows.bat, flash_mac.command or flash_linux.sh) to flash the firmware.
3. Wait until flashing is complete and press the RESET button once to launch the new firmware.

Setting the Keys

From a command prompt or terminal, run:

edge-impulse-daemon

This will start a wizard which will ask you to log in and choose an Edge Impulse project. If you want to switch projects run the command with **--clean**.

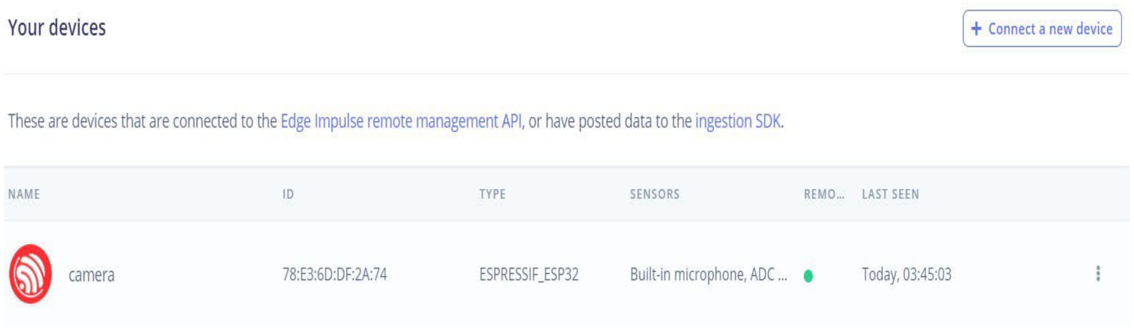
Connection to which device

The Launchpad enumerates two serial ports. The first is the

Application/User UART, which the edge-impulse firmware communications through. The other is an Auxiliary Data Port, which is unused.
When running the edge-impulse-daemon you will be prompted on which serial port to connect to.
Generally, select the lower numbered serial port. This usually corresponds with the Application/User UART. On windows, the serial port may also be verified in the device manager.

Verifying that the device is connected

That's all! Your device is now connected to Edge Impulse. To verify this, go to your Edge Impulse project, and click **Devices**. The device will be listed here.



These are devices that are connected to the Edge Impulse remote management API, or have posted data to the ingestion SDK.

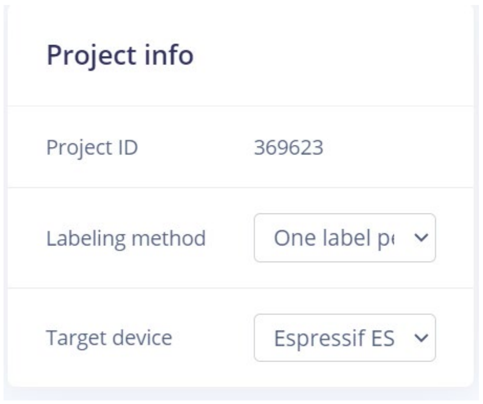
NAME	ID	TYPE	SENSORS	REMO...	LAST SEEN
camera	78:E3:6D:DF:2A:74	ESPRESSIF_ESP32	Built-in microphone, ADC ...	●	Today, 03:45:03

PART 1

Project Procedure

1. Building a dataset

First make sure that you have selected one label per data item as labeling method from the Edge Impulse dashboard.



Project info

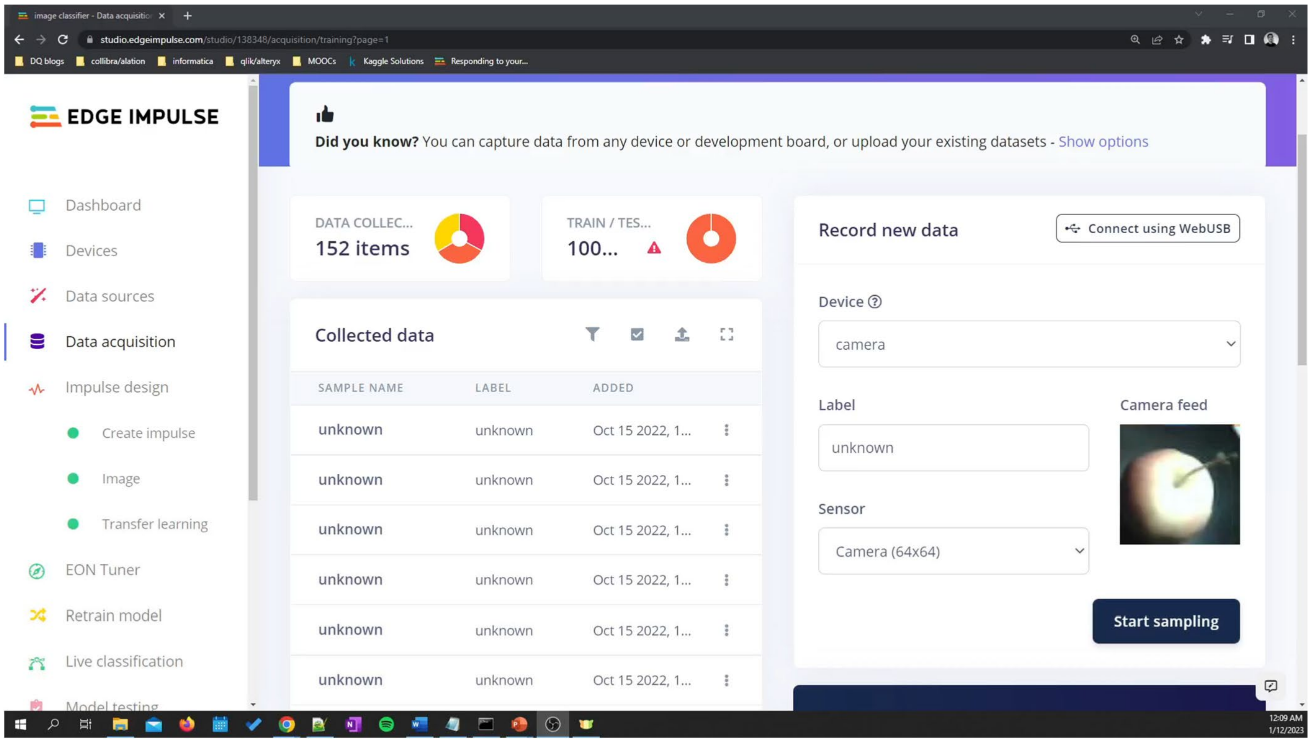
Project ID	369623
Labeling method	One label pı
Target device	Espressif ES

In this tutorial we'll build a model that can distinguish between two objects -we've used a Apple and a Banana, but feel free to pick two other objects. To make your machine learning model see it's important that you capture a lot of example images of these objects. When training the model these example images are used to let the model distinguish between them. Because there are (hopefully) a lot more fruits than just Apple or Banana, you also need to capture images that are *neither* a Banana or a Apple to make the model work well. **Make sure that the sensor selected is "Camera 64x64**".

Capture the following amount of data - make sure you capture a wide variety of angles and zoom levels:

- 50 images of a Banana.
- 50 images of a Apple.
- 50 images of neither a Banana nor a Apple - make sure to capture a wide variation ofrandom objects in the same room as your lamp or plant.

You should have a well-balanced dataset listed under **Data acquisition** in your Edge Impulse project. You can switch between your training and testing data with the two buttons above the 'Data collected' widget.



Did you know? You can capture data from any device or development board, or upload your existing datasets - Show options

DATA COLLEC... 152 items

TRAIN / TES... 100...

Collected data

SAMPLE NAME	LABEL	ADDED
unknown	unknown	Oct 15 2022, 1...
unknown	unknown	Oct 15 2022, 1...
unknown	unknown	Oct 15 2022, 1...
unknown	unknown	Oct 15 2022, 1...
unknown	unknown	Oct 15 2022, 1...
unknown	unknown	Oct 15 2022, 1...

Record new data

Device: camera

Label: unknown

Sensor: Camera (64x64)

Camera feed

2.Designing an Impulse

With the training set in place, you can design an impulse. An impulse takes the raw data, adjusts the image size, uses a preprocessing block to manipulate the image, and then uses a learning block to classify new data. Preprocessing blocks always return the same values for the same input (e.g., convert a color image into a grayscale one), while learning blocks learn from past experiences.
For this tutorial we'll use the 'Images' preprocessing block. This block takes in the color image, optionally makes the image grayscale, and then turns the data into a features array. If you want to do more interesting preprocessing steps - like finding faces in a photo before feeding the image into the network -, see the Building custom processing blocks tutorial. Then we'll use a 'Transfer Learning' learning block, which takes all the images in and learns to distinguish between the three (Apple, Banana, 'unknown') classes.
In the studio go to **Create impulse**, set the image width and image height to 96, and add the 'Images' and 'Transfer Learning (Images)' blocks. Then click Save impulse.

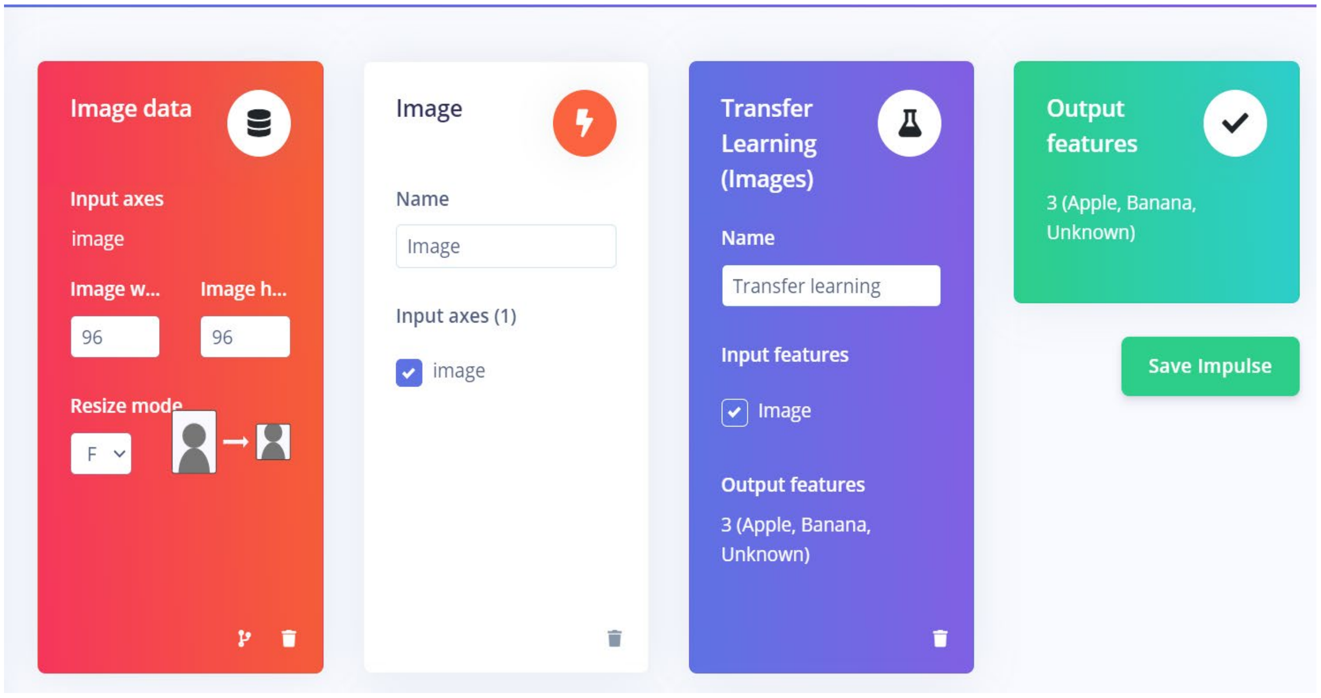


Image data
Input axes: image
Image w...: 96
Image h...: 96
Resize mode: F

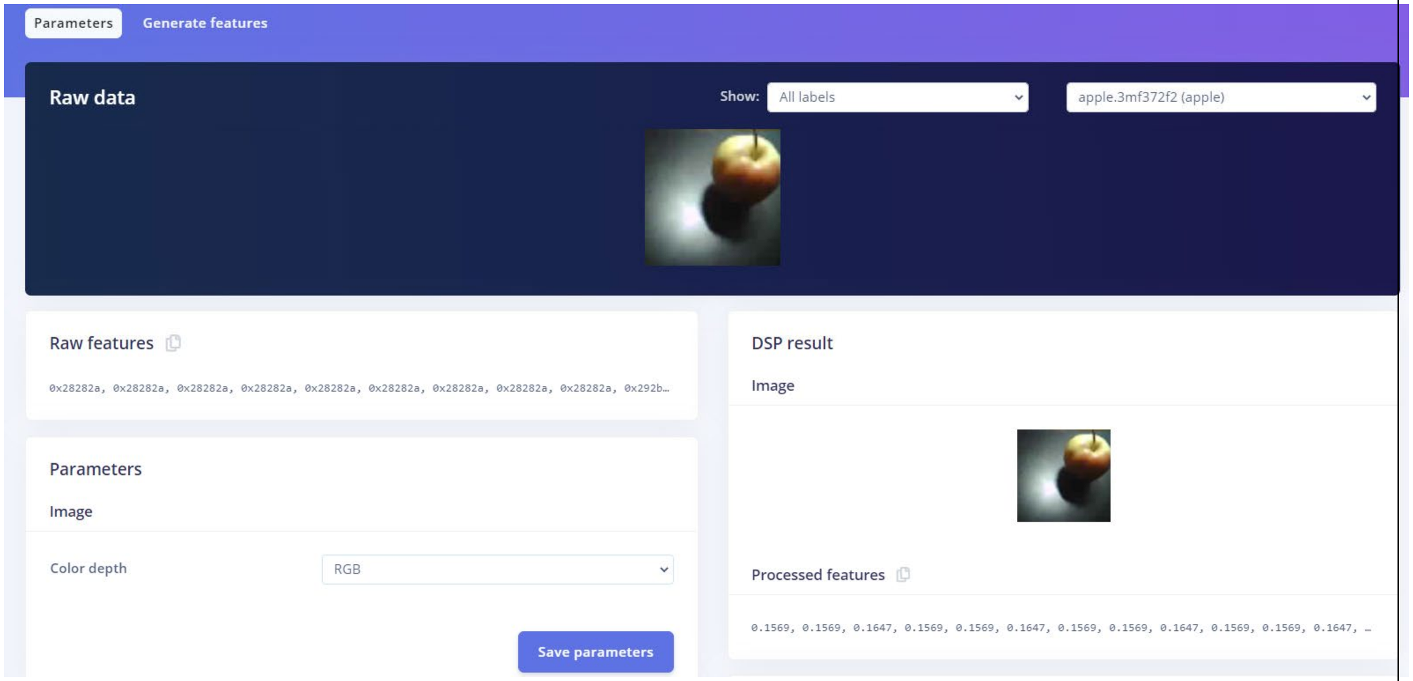
Image
Name: Image
Input axes (1)
✓ image

Transfer Learning (Images)
Name: Transfer learning
Input features
✓ Image
Output features: 3 (Apple, Banana, Unknown)

Output features
3 (Apple, Banana, Unknown)

a. Configuring the processing block

To configure your processing block, click **Images** in the menu on the left. This will show you the raw data on top of the screen (you can select other files via the drop-down menu), and the results of the processing step on the right. You can use the options to switch between 'RGB' and 'Grayscale' mode, but for now leave the color depth on 'RGB' and click **Save parameters**.



Raw features

```
0x28282a, 0x28282a, 0x28282a, 0x28282a, 0x28282a, 0x28282a, 0x28282a, 0x28282a, 0x28282a, 0x292b…
```

Parameters

Image

Color depth: RGB

DSP result

Image

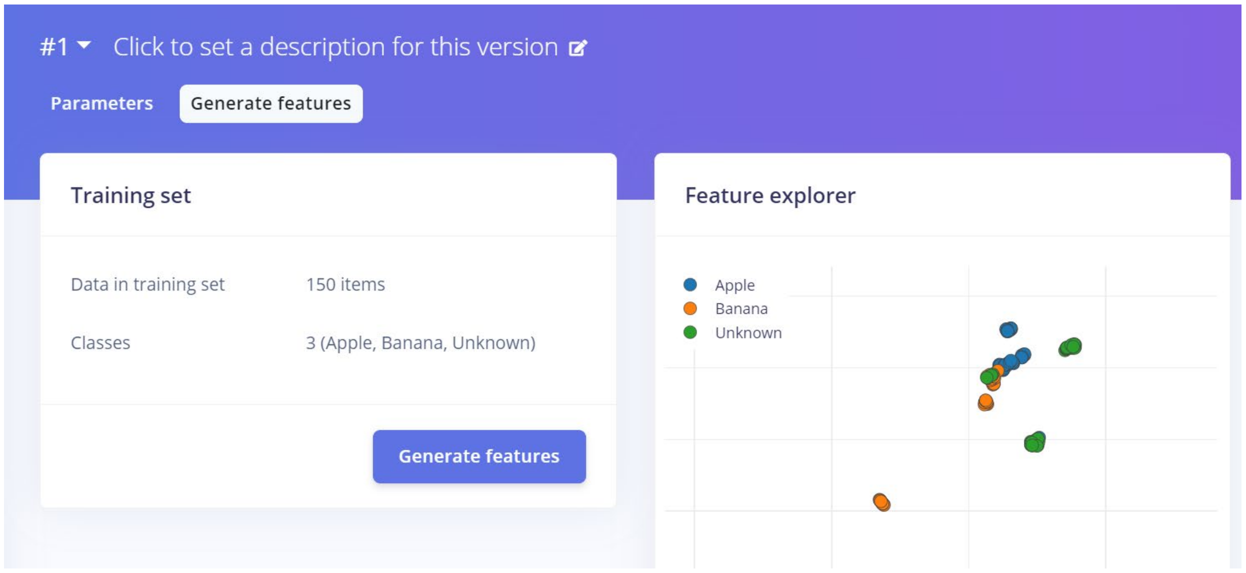
Processed features

```
0.1569, 0.1569, 0.1647, 0.1569, 0.1569, 0.1647, 0.1569, 0.1569, 0.1647, 0.1569, 0.1569, 0.1647, …
```

This will send you to the 'Feature generation' screen. In here you'll:

- Resize all the data.
- Apply the processing block on all this data.
- Create a 3D visualization of your complete dataset.

Click **Generate features** to start the process.
Afterwards the 'Feature explorer' will load. This is a plot of all the data in your dataset. Because images have a lot of dimensions (here: 96x96x3=27,648 features) we run a process called 'dimensionality reduction' on the dataset before visualizing this. Here the 27,648 features are compressed down to just 3, and then clustered based on similarity. Even though we have little data you can already see some clusters forming (lamp images are all on the right), and can click on the dots to see which image belongs to which dot.



Data in training set: 150 items

Classes: 3 (Apple, Banana, Unknown)

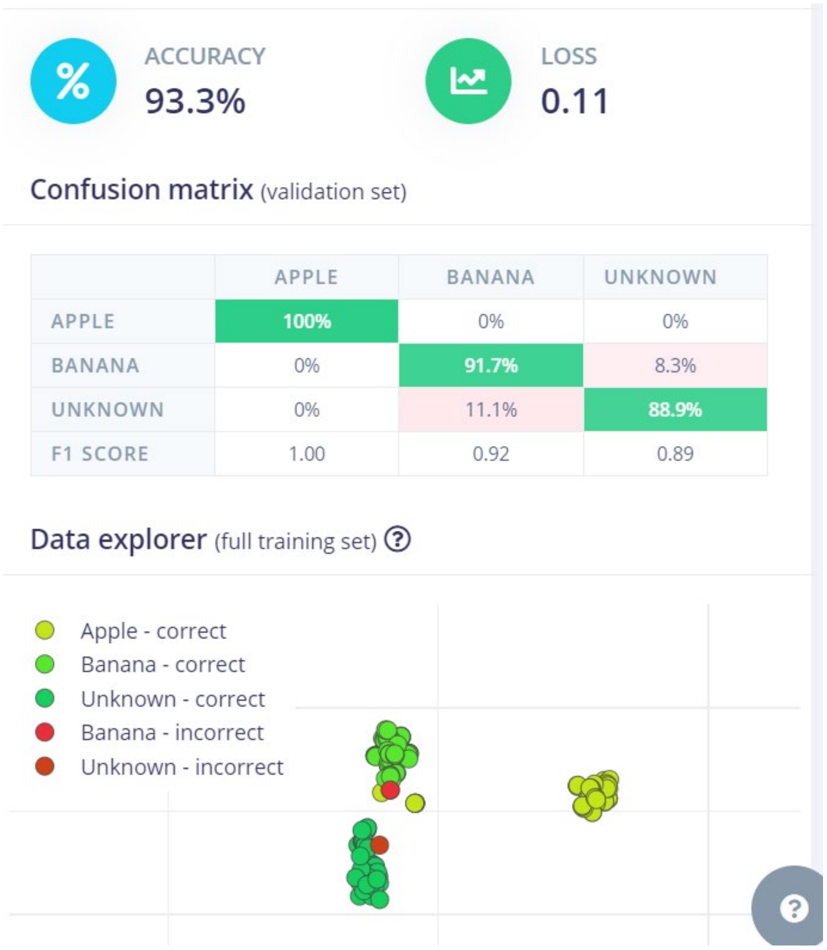
b. Configuring the transfer learning model

With all data processed it's time to start training a neural network. Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. The network that we're training here will take the image data as an input and try to map this to one of the three classes.
It's very hard to build a good working computer vision model from scratch, as you need a wide variety of input data to make the model generalize well, and training such models can take days on a GPU. To make this easier and faster we are using transfer learning. This lets you piggyback on a well-trained model, only retraining the upper layers of a neural network, leading to much more reliable models that train in a fraction of the time and work with substantially smaller datasets.
To configure the transfer learning model, click **Transfer learning** in the menu on the left. Here you can select the base model (the one selected by default will work, but you can change this based on your size requirements), optionally enable data augmentation (images are randomly manipulated to make the model perform better in the real world), and the rate at which the network learns.
Set:

- Number of training cycles to 100.
- Learning rate to 0.005.
- Data augmentation, Profile int8 model: enabled.

- Validation set size: 20%.

And click **Start training**. After the model is done you'll see accuracy numbers, a confusion matrix and some predicted on-device performance on the bottom. You have now trained your model!



ACCURACY
93.3%

LOSS
0.11

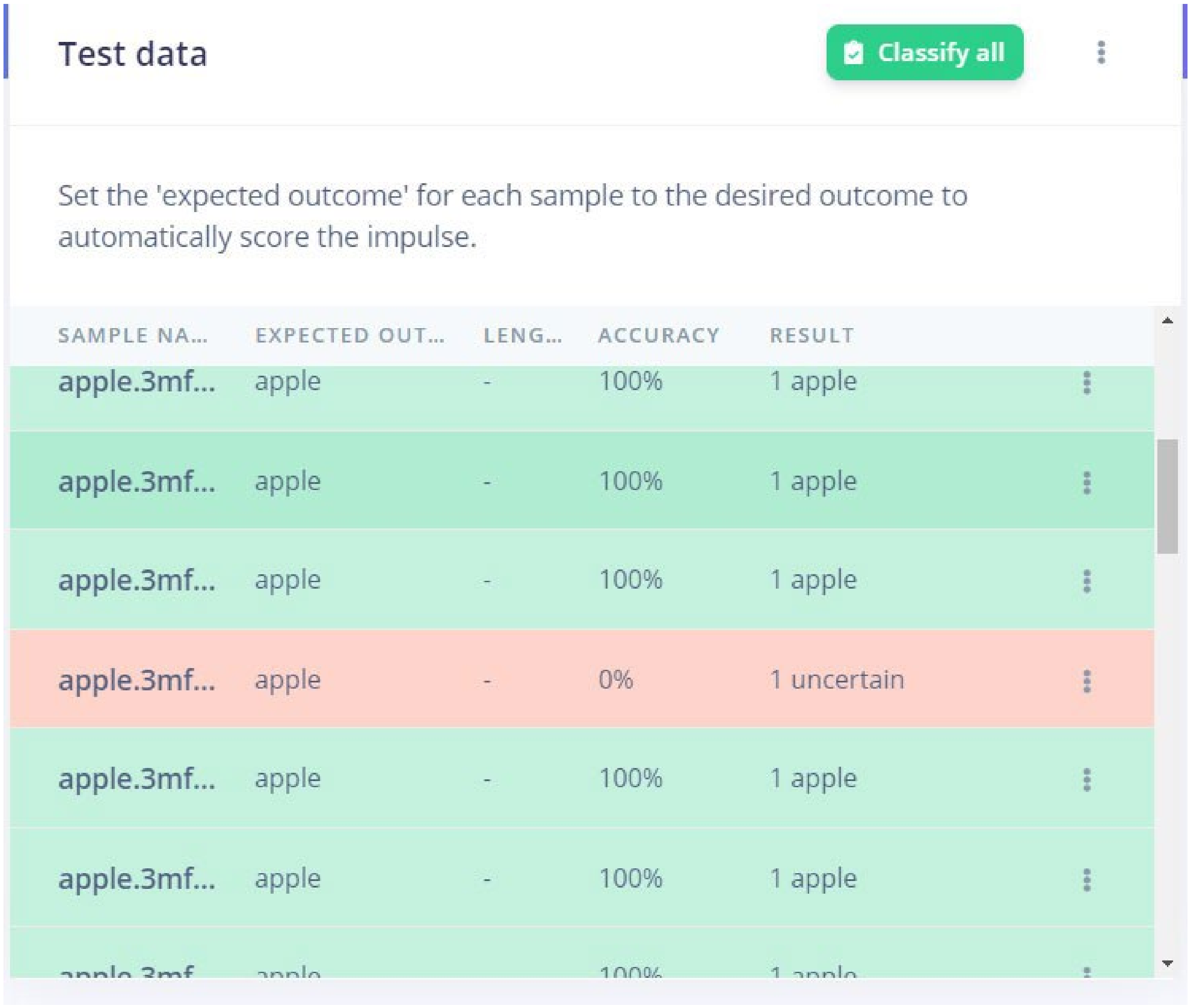
Confusion matrix (validation set)

	APPLE	BANANA	UNKNOWN
APPLE	100%	0%	0%
BANANA	0%	91.7%	8.3%
UNKNOWN	0%	11.1%	88.9%
F1 SCORE	1.00	0.92	0.89

Data explorer (full training set)

4. Validating your model

With the model trained let's try it out on some test data. When collecting the data we split the data up between a training and a testing dataset. The model was trained only on the training data, and thus we can use the data in the testing dataset to validate how well the model will work in the real world. This will help us ensure the model has not learned to overfit the training data, which is a common occurrence.
To validate your model, go to **Model testing**, select the checkbox next to 'Sample name' and click **Classify selected**. Here we hit 89% accuracy, which is great for a model with so little data.

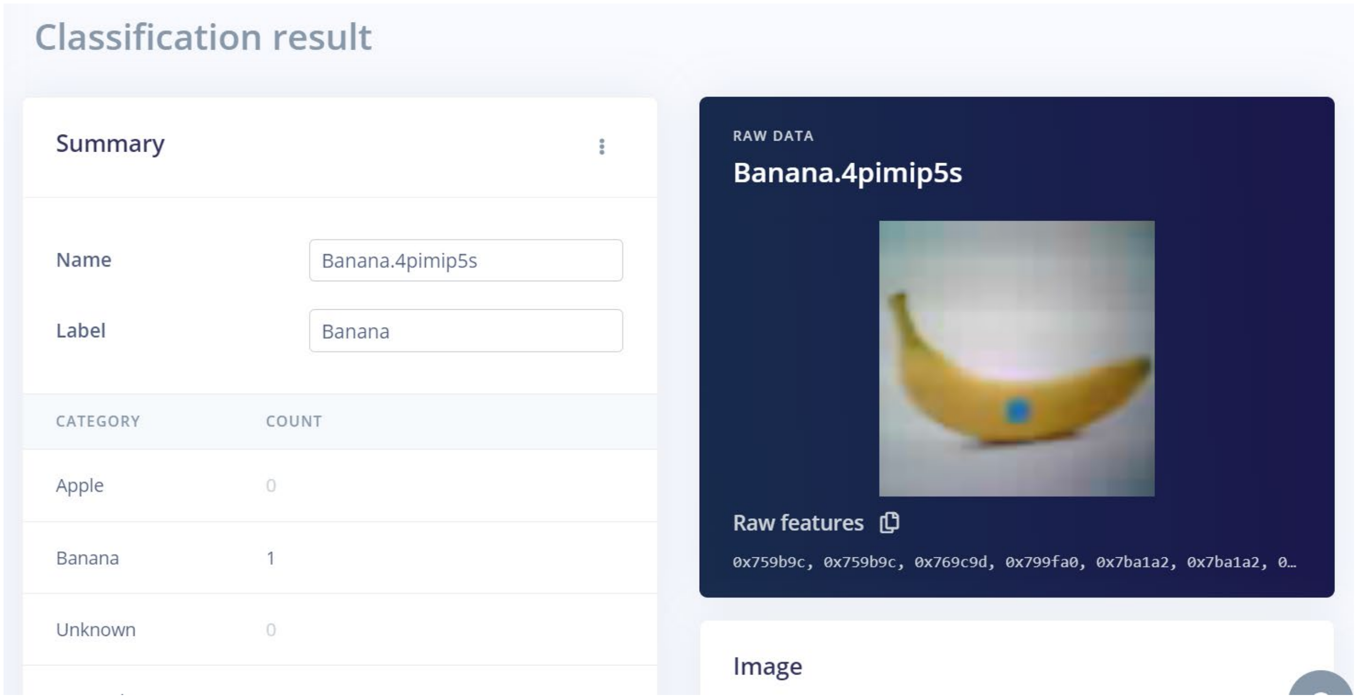


Test data

Set the 'expected outcome' for each sample to the desired outcome to automatically score the impulse.

SAMPLE NA...	EXPECTED OUT...	LENG...	ACCURACY	RESULT
apple.3mf...	apple	-	100%	1 apple
apple.3mf...	apple	-	100%	1 apple
apple.3mf...	apple	-	100%	1 apple
apple.3mf...	apple	-	0%	1 uncertain
apple.3mf...	apple	-	100%	1 apple
apple.3mf...	apple	-	100%	1 apple
apple.3mf	apple	-	100%	1 apple

To see a classification in detail, click the three dots next to an item, and select **Show classification**. This brings you to the Live classification screen with much more details on the file (if you collected data with your mobile phone, you can also capture new testing data directly from here). This screen can help you determine why items were misclassified.



Name: Banana.4pimip5s
Label: Banana

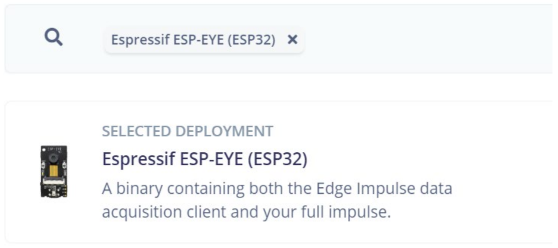
CATEGORY	COUNT
Apple	0
Banana	1
Unknown	0

```
0x759b9c, 0x759b9c, 0x769c9d, 0x799fa0, 0x7ba1a2, 0x7ba1a2, 0…
```

5. Running the model on your device

With the impulse designed, trained and verified you can deploy this model back to your device. This makes the model run without an internet connection, minimizes latency, and runs with minimum power consumption. Edge Impulse can package up the complete impulse - including the preprocessing steps, neural network weights, and classification code - in a single C++ library that you can include in your embedded software.
To run your impulse on either the OpenMV camera or your phone, follow these steps:

Click on **Deployment** in the menu. Then under 'Build firmware' select your development board and click **Build**.



SELECTED DEPLOYMENT
Espressif ESP-EYE (ESP32)
A binary containing both the Edge Impulse data acquisition client and your full impulse.

This will export the impulse, and build a binary that will run on your development board in a single step. After building is completed you'll get promptedto download a binary. Save this on your computer.

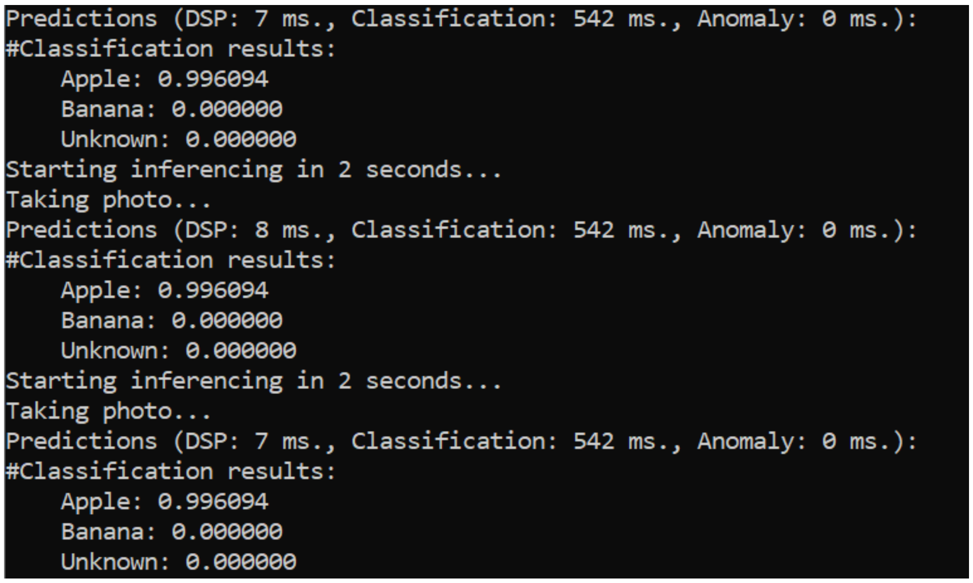
a. **Flashing the device**

When you click the **Build** button, you'll see a pop-up with text and video instructions on how to deploy the binary to your particular device. Follow these instructions. Once you are done, we are ready to test your impulse out.

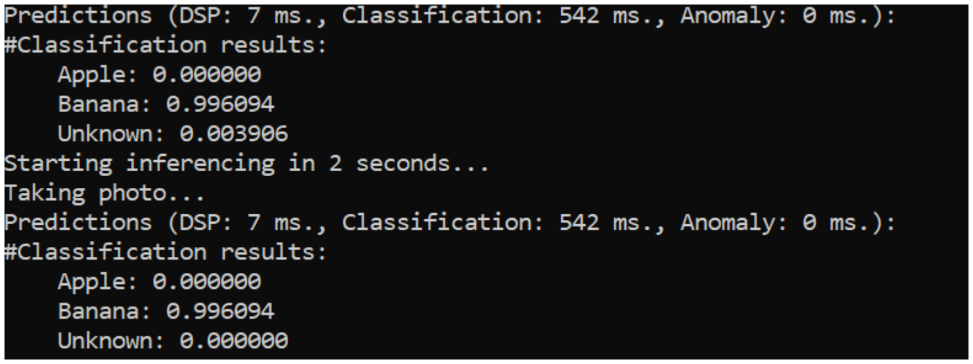
a. **Running the model on the device**

We can connect to the board's newly flashed firmware over serial. Open a terminal and run:

$ edge-impulse-run-impulse



```
Predictions (DSP: 7 ms., Classification: 542 ms., Anomaly: 0 ms.):
#Classification results:
    Apple: 0.996094
    Banana: 0.000000
    Unknown: 0.000000
Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 8 ms., Classification: 542 ms., Anomaly: 0 ms.):
#Classification results:
    Apple: 0.996094
    Banana: 0.000000
    Unknown: 0.000000
Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 7 ms., Classification: 542 ms., Anomaly: 0 ms.):
#Classification results:
    Apple: 0.996094
    Banana: 0.000000
    Unknown: 0.000000
```



```
Predictions (DSP: 7 ms., Classification: 542 ms., Anomaly: 0 ms.):
#Classification results:
    Apple: 0.000000
    Banana: 0.996094
    Unknown: 0.003906
Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 7 ms., Classification: 542 ms., Anomaly: 0 ms.):
#Classification results:
    Apple: 0.000000
    Banana: 0.996094
    Unknown: 0.000000
```

The machine learning model running in real-time on device, classifying fruits. Congratulations! You've added sight to your sensors.