

EEDG6302: Microprocessor and Embedded Systems

Wednesday Lab Report

Project 3 Lab 3: Image Classification

Aim:

- A brief understanding about ML algorithms
- A brief understanding about TI Launchpad (CC1352P) and Booster Sensors
- 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 branch of artificial intelligence that involves machines imitating intelligent human behavior to perform complex tasks. Tiny machine learning, or TinyML, is a newly emerging field that combines machine learning and embedded systems. Embedded systems are small computing devices that operate with extremely low power and can run for extended periods on small batteries. TinyML involves shrinking deep learning networks to fit on tiny hardware, making it possible to bring artificial intelligence to intelligent devices.

Edge computing, on the other hand, involves bringing computation and data storage closer to the source of data. Most edge devices in IoT ecosystems are designed to collect sensor data and transmit it to local or remote cloud platforms.

Espressif ESP-EYE (ESP32):

The ESP-EYE (ESP32) is a small development board that comes with a 2-Megapixel camera and a microphone. It also has ample storage space, with 8 MB PSRAM and 4 MB SPI flash, and is fully compatible with Edge Impulse. While there are other ESP32-based boards available, and some custom designs that utilize the ESP32 SoM, Edge Impulse firmware has been tested with ESP-EYE and ESP FireBeetle boards. However, it may be possible to modify the firmware to work with other ESP32 designs.



Edge Impulse

Edge Impulse is a cloud-based platform that enables the development of machine learning models for TinyML devices. The platform supports automated machine learning (AutoML) processing for edge devices and offers support for various boards, including smartphones, to deploy machine learning models on these devices. With Edge Impulse, developers can create and train machine learning models using data from sensors, audio, and other sources, and then deploy those models on edge devices for real-time processing.

Sampling New Data

To achieve optimal results in machine learning, a large dataset is essential. Therefore, relying on a single sample won't suffice. It's recommended to begin creating your own dataset at this point. The device will capture various images and transmit them to Edge Impulse

- 20 images of an Apple
- 20 images of an Mango
- 20 images of neither an apple nor an Mango (Robot)

Designing an Impulse

Once the training set is established, an impulse can be designed. An impulse processes the raw data by adjusting the image size, applying a preprocessing block to manipulate the image, and then using a learning block to classify new data. Preprocessing blocks are deterministic and will always produce the same outputs given the same inputs, while learning blocks adapt and learn from previous experiences. In this case, we will use the Images preprocessing block, which takes in a color image and can optionally convert it to grayscale before turning it into a features array. Then, we will employ a Transfer Learning learning block, which learns to distinguish between the three classes by analyzing all the images.

The screenshot displays the Edge Impulse web interface for designing an impulse. It consists of four main blocks arranged horizontally:

- Image data** (Red block): Contains input axes (image), image width (96), image height (96), and a resize mode dropdown set to 'Squash'. A note at the bottom states: "For optimal accuracy with transfer learning blocks, use a 96x96 or 160x160 image size."
- Image** (White block): Contains a name field (Image) and an input axes (1) dropdown set to 'image'.
- Transfer Learning (Images)** (Purple block): Contains a name field (Transfer learning), an input features dropdown set to 'Image', and an output features field (3 (Apple, Mango, Robot)).
- Output features** (Green block): Contains a checkmark icon and the text '3 (Apple, Mango, Robot)'. A 'Save Impulse' button is located below this block.

Configuring the processing block

After configuring the processing block. This will show the raw data on top of the screen, and the results of the processing step on the right. In the Feature generation screen, we can :

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

Afterwards the Feature explorer will load. This is a plot of all the data in our dataset. Because images have a lot of dimensions (here: $96 \times 96 \times 3 = 27,648$ 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 we can already see some clusters forming and can click on the dots to see which image belongs to which dot.

The screenshot displays the 'Feature Explorer' interface. At the top, a 'Raw data' section shows a selected item 'Apple' with a dropdown menu and a file path 'Apple.3unbi6c6 (Apple)'. Below this, a small image of a red apple is visible. The interface is divided into two main panels. The left panel, titled 'Raw features', shows a list of hexadecimal values: '0xffeb7e, 0xffeb7e, 0xffea7e, 0xffe87e, 0xffe57c, 0xffde79, 0xffd977...'. Below this is a 'Parameters' section with a 'Color depth' dropdown set to 'RGB' and a 'Save parameters' button. The right panel, titled 'DSP result', shows an 'Image' section with the same apple image and a 'Processed features' section displaying a list of numerical values: '1.0000, 0.9216, 0.4941, 1.0000, 0.9216, 0.4941, 1.0000, 0.9176, 0.49...'. The interface has a dark blue header and a light blue sidebar.

Configuring the Neural Network

Neural networks are a collection of algorithms that loosely mimic the workings of the human brain and are designed to identify patterns. The network we are currently training will take the image data as input and attempt to classify it into one of the three classes.

Creating an effective computer vision model from scratch can be extremely challenging, as it requires a vast amount of diverse input data to produce a model that can generalize well. Additionally, training such models can take days on a GPU. To simplify and accelerate the process, we are utilizing transfer learning. By only training the upper layers of a neural network, we can produce more reliable models that can be trained in a fraction of the time and work with substantially smaller datasets.

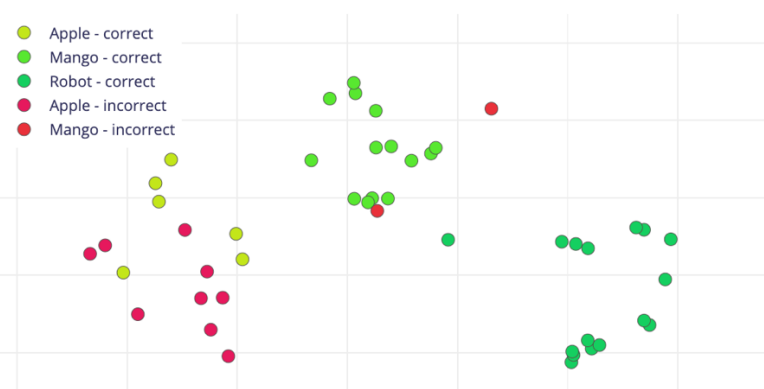
Last training performance (validation set)



Confusion matrix (validation set)

	APPLE	MANGO	ROBOT
APPLE	100%	0%	0%
MANGO	0%	75%	25%
ROBOT	0%	0%	100%
F1 SCORE	1.00	0.86	0.89

Data explorer (full training set) ⓘ

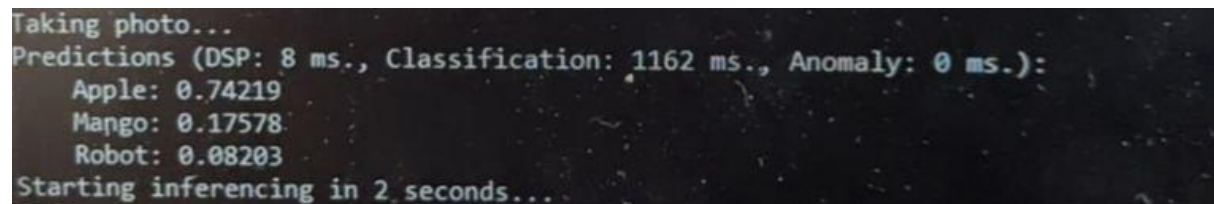


On-device performance ⓘ



Deploying Back to Device

Once the impulse has been designed, trained, and verified, we can deploy the model back to your device. This allows the model to run without an internet connection, reduces latency, and minimizes power consumption. Edge Impulse can bundle the entire impulse - including the signal processing code, neural network weights, and classification code - into a single C++ library, which can be incorporated into embedded software.



```
Taking photo...
Predictions (DSP: 8 ms., Classification: 1162 ms., Anomaly: 0 ms.):
  Apple: 0.74219
  Mango: 0.17578
  Robot: 0.08203
Starting inferencing in 2 seconds...
```

The image is a screenshot of a terminal window with a dark background and light-colored text. It displays the output of a classification process. The first line is 'Taking photo...'. The second line is 'Predictions (DSP: 8 ms., Classification: 1162 ms., Anomaly: 0 ms.):'. The third line shows 'Apple: 0.74219'. The fourth line shows 'Mango: 0.17578'. The fifth line shows 'Robot: 0.08203'. The sixth line is 'Starting inferencing in 2 seconds...'.

The output shows the prediction of the sample as the 80% probability of being Apple after mangoes are displayed by using already neural networks which are trained by classified data that are labelled earlier.