Name :- Tanmay Khedekar
Class :- TY AIEC-1 **Batch:-** A
Sub :- Edge Computing Lab
Roll No.:- 2223122
Er. No.:- MITU22BTCS0906

Experiment No. 10

Title: Study of Transfer Learning (Images) on Edge Computing Devices

Objective:

Build a project to apply Transfer Learning of MobileNetV1 & V2 architectures trained on an ImageNet dataset

Tasks:

- Understand Transfer learning
- Understanding of MobileNetV1 & V2 Architectures
- Configure Edge Impulse for Object Detection
- Apply a pre-trained network for you to fine-tune your specific application
- Building and Training a Model
- Deploy on Edge Computing Devices

Materials Required:

- Arduino Nano 33 BLE Sense
- OV7675 camera module
- Camera adapter (e.g., Arducam Mini or custom wiring)
- USB cable (Micro USB)
- Power source (USB or portable battery, optional)

Steps to Configure the Edge Impulse:

1. **Create an Account and New Project:**
   Sign up for an Edge Impulse account.
   Create a new project from the dashboard.
2. **Connect a Device:**
   You can use a supported development board or your smartphone as a sensor device.Follow the instructions to connect your device to your Edge Impulse project.
3. **Collect Data**:
   Use the Edge Impulse mobile app or the Web interface to collect data from the onboard sensors.
   For a "Hello World" project, you could collect accelerometer data, for instance.
4. **Create an Impulse:**

Go to the 'Create impulse' page.

Add a processing block (e.g., time-series data) and a learning block (e.g., classification). Save the impulse, which defines the machine learning pipeline.

5. Design a Neural Network:

Navigate to the 'NN Classifier' under the 'Learning blocks'.

Design a simple neural network. Edge Impulse provides a default architecture that works well for most basic tasks.

6. Train the Model:

Click on the 'Start training' button to train your machine learning model with the collected data.

7. Test the Model:

Once the model is trained, you can test its performance with new data in the 'Model Testing' tab.

8. Deploy the Model:

Go to the 'Deployment' tab.

Select the deployment method that suits your edge device (e.g., Arduino library, WebAssembly, container, etc.).

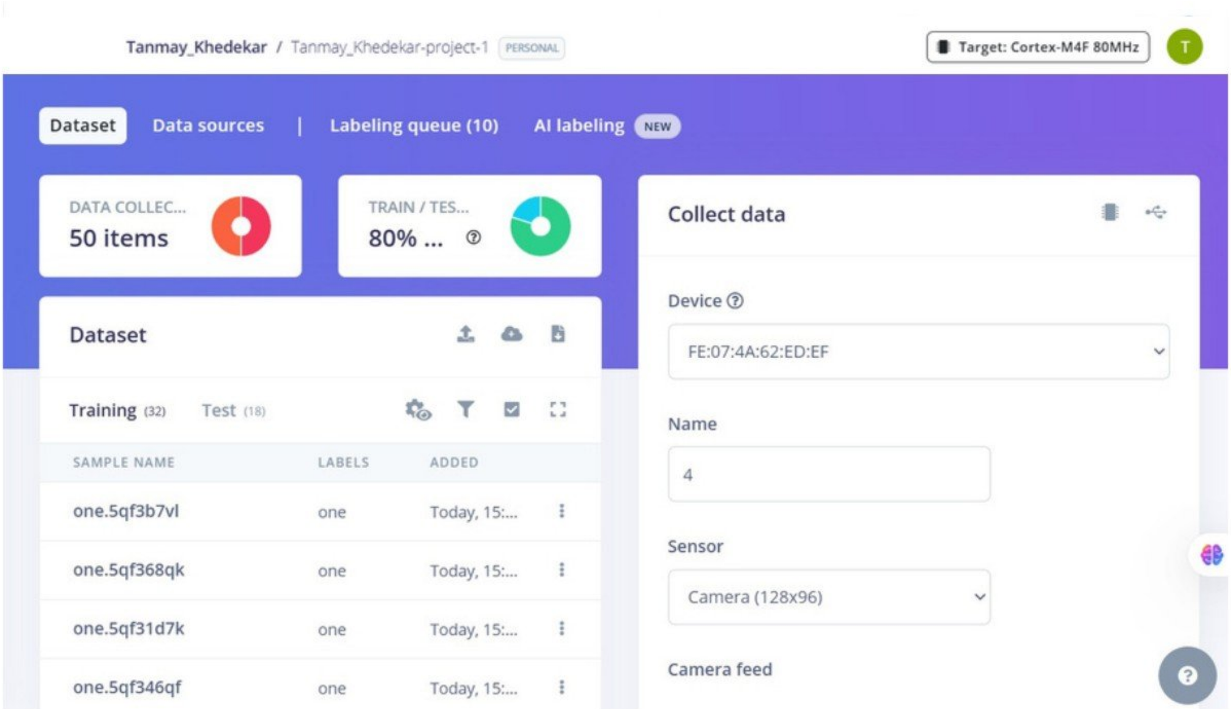
Follow the instructions to deploy the model to your device.

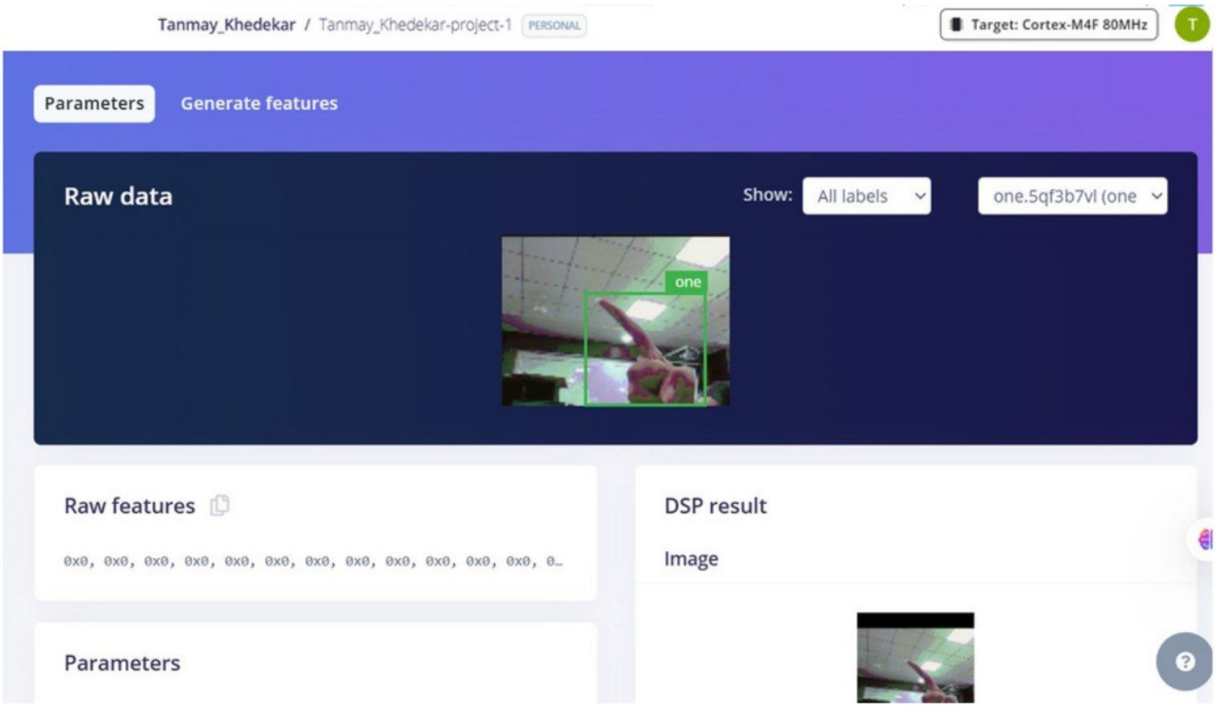
9. Run Inference:

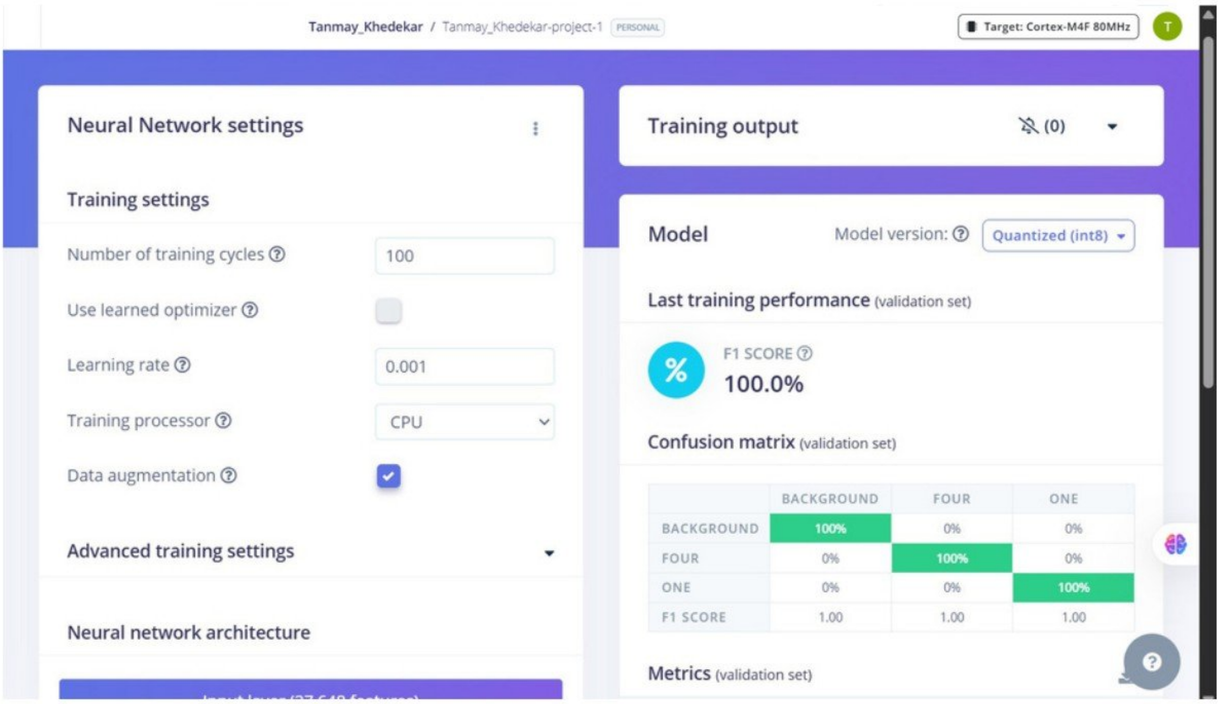
With the model deployed, run inference on the edge device to see it classifying data in real-time.

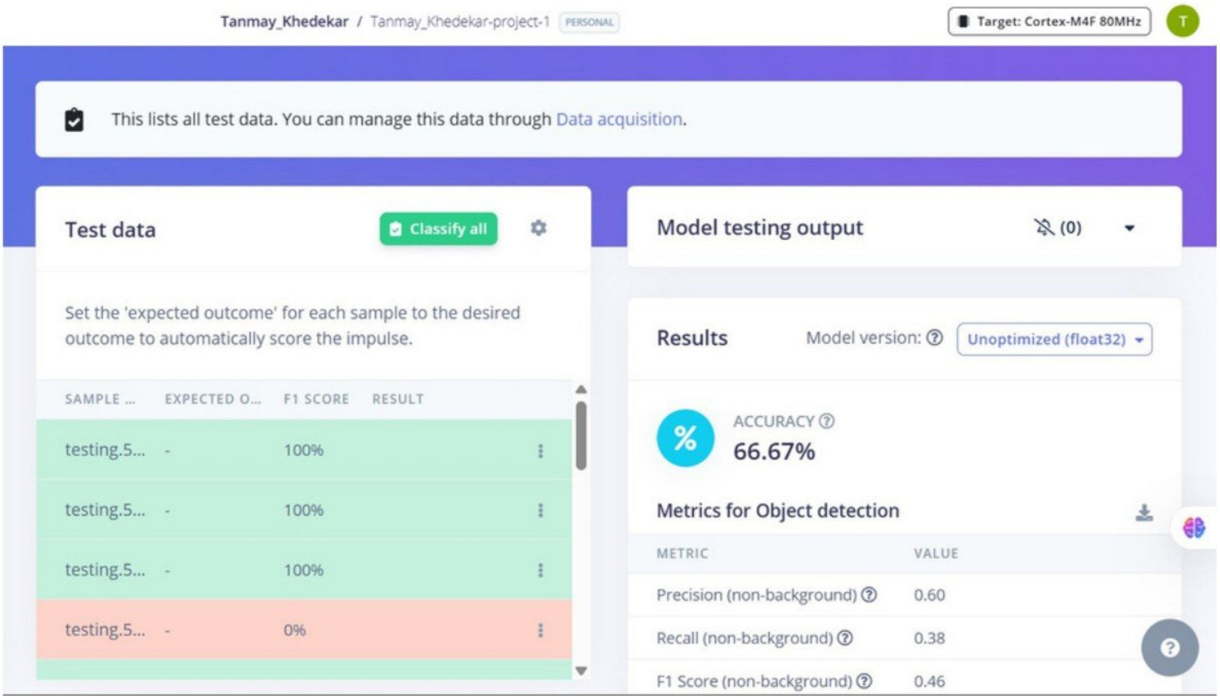
10. Monitor:

You can monitor the performance of your device through the Edge Impulse studio.




Tanmay_Khedekar / Tanmay_Khedekar-project-1 PERSONAL
Target: Cortex-M4F 80MHz
Parameters
Generate features
Raw data
Show: All labels
one.5qf3b7vl (one
one
Raw features
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0…
DSP result
Image
Parameters


Tanmay_Khedekar / Tanmay_Khedekar-project-1 PERSONAL
Target: Cortex-M4F 80MHz
Neural Network settings
Training settings
Number of training cycles
100
Use learned optimizer
Learning rate
0.001
Training processor
CPU
Data augmentation
Advanced training settings
Neural network architecture
Training output
(0)
Model
Model version:
Quantized (int8)
Last training performance (validation set)
F1 SCORE
100.0%
Confusion matrix (validation set)
BACKGROUND FOUR ONE
BACKGROUND 100% 0% 0%
FOUR 0% 100% 0%
ONE 0% 0% 100%
F1 SCORE 1.00 1.00 1.00
Metrics (validation set)



Python Code :

```
#include <EdgeImpulseModel.h> // Replace with actual header from your Edge Impulse
export
#include <Arduino_LSM9DS1.h> // If you use IMU or other sensors

void setup() {
  Serial.begin(115200);
  if (!IMU.begin()) {
    Serial.println("Failed to initialize IMU!");
    while (1);
  }
}

void loop() {
  float buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE];

  // Fill buffer with your sensor/image data (depends on your model)
  for (int i = 0; i < EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE; i++) {
    buffer[i] = analogRead(A0); // example input, replace with camera/image data
  }

  // Run inference
  ei_impulse_result_t result = { 0 };
  signal_t signal;
  numpy::signal_from_buffer(buffer, EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE,
&signal);
```

```
 EI_IMPULSE_ERROR res = run_classifier(&signal, &result, false);

  if (res != EI_IMPULSE_OK) {
    Serial.print("ERR: Failed to run classifier ");
    Serial.println(res);
    return;
  }

  // Print results
  for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
    Serial.print(result.classification[ix].label);
    Serial.print(": ");
    Serial.println(result.classification[ix].value);
  }

  delay(1000);
}
```

Conclusion :-

This project demonstrates how **Transfer Learning** using **MobileNetV1/V2**, trained on the **ImageNet** dataset, can be effectively applied for custom image classification on **Edge Computing Devices**. By fine-tuning pre-trained models and deploying them through **Edge Impulse**, we achieve accurate, low-latency inference on devices like the **Arduino Nano 33 BLE Sense**. This approach enables efficient real-time object detection while optimizing for limited computational resources.