Edge Computing Lab

Class: TY-AIEC

School of Computing, MIT Art Design Technology University

Academic Year: 2024-25

Experiment No. 10

TASHU DHOTE

TY AIEC

2223307

Introduction

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

Introduction

Edge Impulse is a development platform for machine learning on edge devices, targeted at developers who want to create intelligent device solutions. The " Camera "sensor reading equivalent in Edge Impulse would typically involve creating a simple machine learning model that can run on an edge device, like classifying sensor data or recognizing a basic pattern.

Materials Required

- Nano BLE Sense Board

Theory

GPIO (General Purpose Input/Output) pins on the Raspberry Pi are used for interfacing with other electronic components. BCM numbering refers to the pin numbers in the Broadcom SOC channel, which is a more consistent way to refer to the GPIO pins across different versions of the

Here's a high-level overview of steps you'd follow to create a "Hello World" project on Edge Impulse:

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:

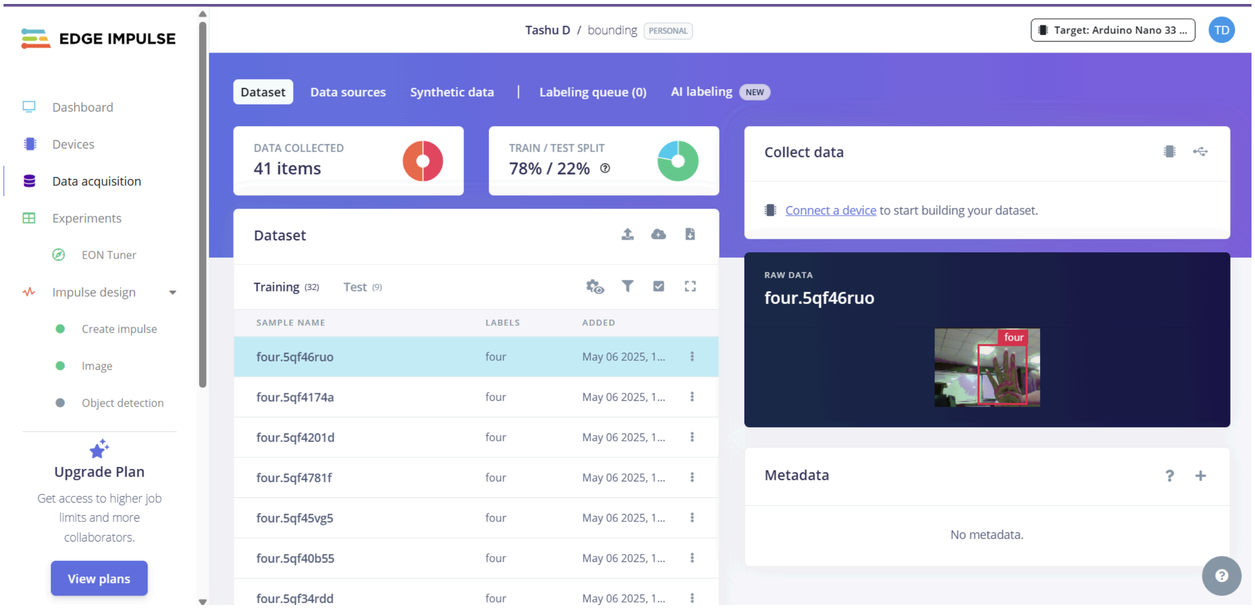
   - With the model deployed, run inference on the edge device to see it classifying data in real-time.

10. Monitor:

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

Paste your Edge Impulse project's Results:

1) Dataset Image



Get access to higher job limits and more collaborators.

DATA COLLECTED 41 items

TRAIN / TEST SPLIT 78% / 22%

Connect a device to start building your dataset.

Dataset

Training (32) Test (9)

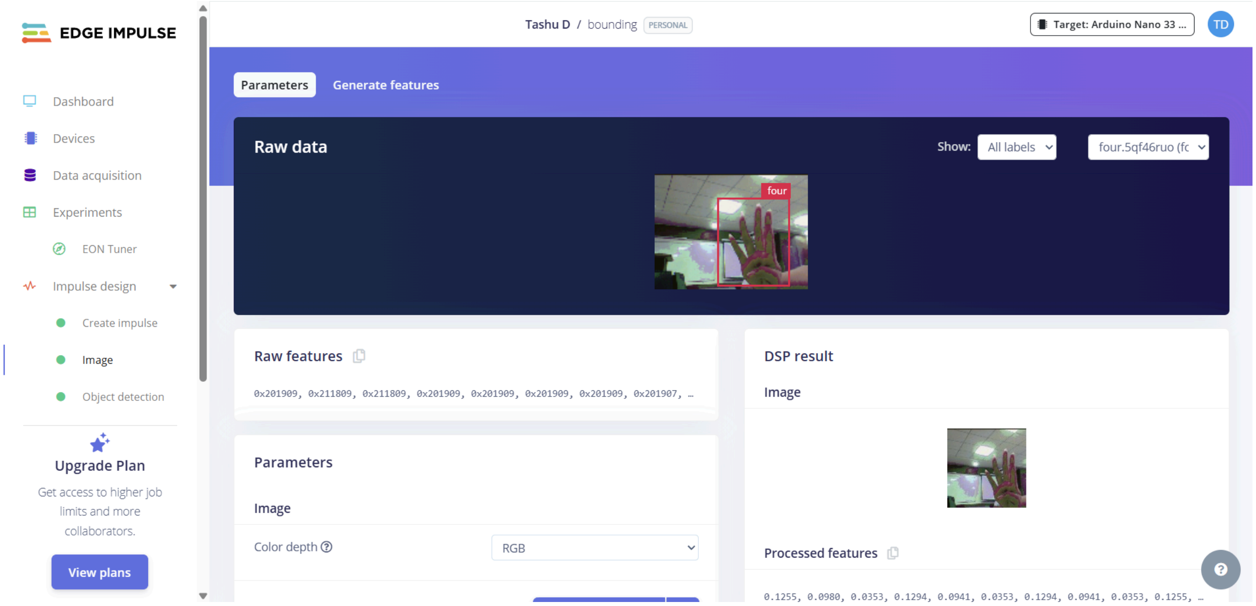
SAMPLE NAME	LABELS	ADDED
four.5qf46ruo	four	May 06 2025, 1...
four.5qf4174a	four	May 06 2025, 1...
four.5qf4201d	four	May 06 2025, 1...
four.5qf4781f	four	May 06 2025, 1...
four.5qf45vg5	four	May 06 2025, 1...
four.5qf40b55	four	May 06 2025, 1...
four.5qf34rdd	four	May 06 2025, 1...

RAW DATA

four.5qf46ruo

No metadata.

2) Feature extraction - Image



Raw data

Show: All labels · four.5qf46ruo (fc

Raw features

```
0x201909, 0x211809, 0x211809, 0x201909, 0x201909, 0x201909, 0x201909, 0x201907, …
```

Parameters

Image

Color depth: RGB

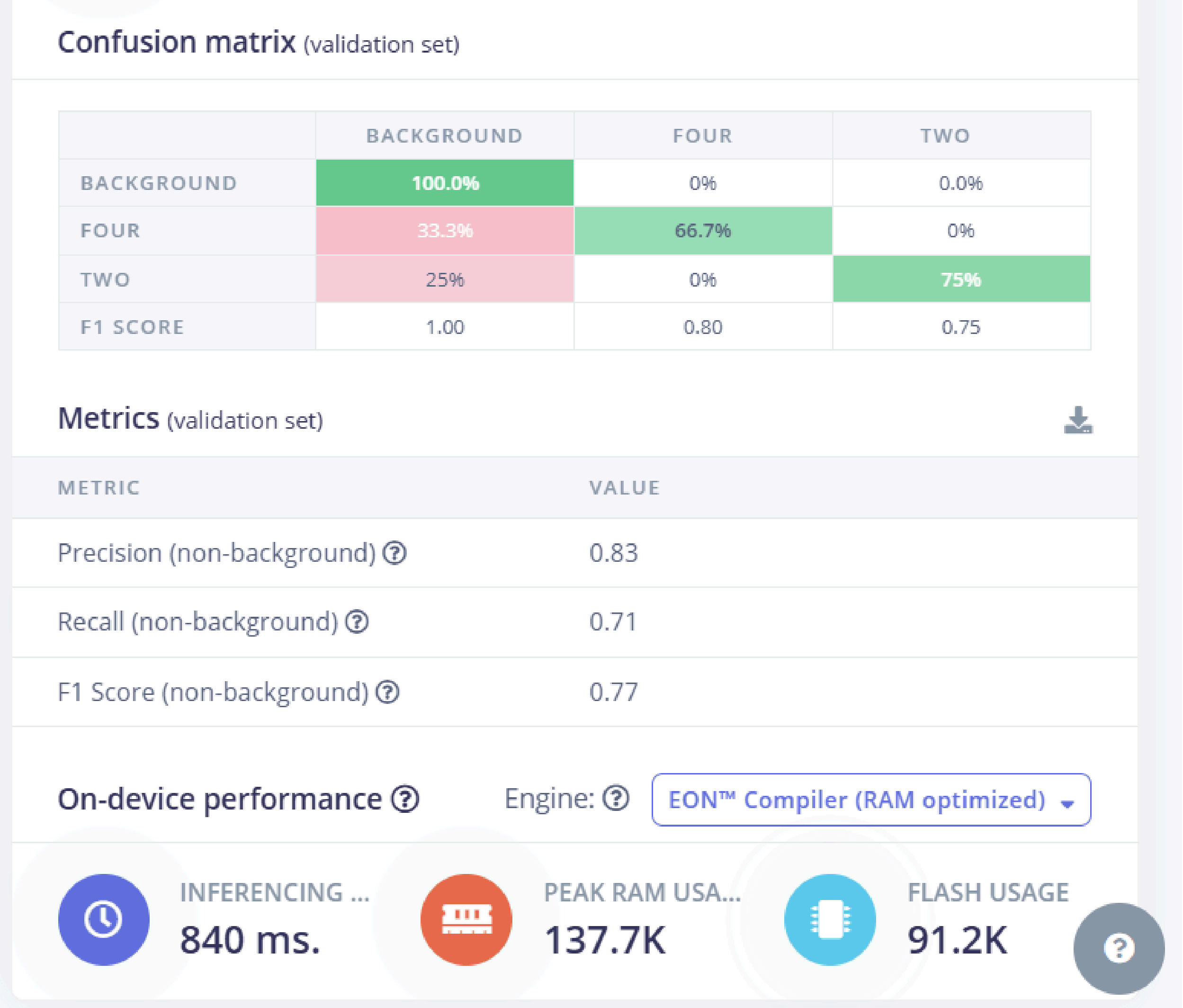
DSP result

Image

Processed features

```
0.1255, 0.0980, 0.0353, 0.1294, 0.0941, 0.0353, 0.1294, 0.0941, 0.0353, 0.1255, …
```

3) Accuracy / Loss - Confusion Matrix – image



Confusion matrix (validation set)

	BACKGROUND	FOUR	TWO
BACKGROUND	100.0%	0%	0.0%
FOUR	33.3%	66.7%	0%
TWO	25%	0%	75%
F1 SCORE	1.00	0.80	0.75

Metrics (validation set)

METRIC	VALUE
Precision (non-background) ⓘ	0.83
Recall (non-background) ⓘ	0.71
F1 Score (non-background) ⓘ	0.77

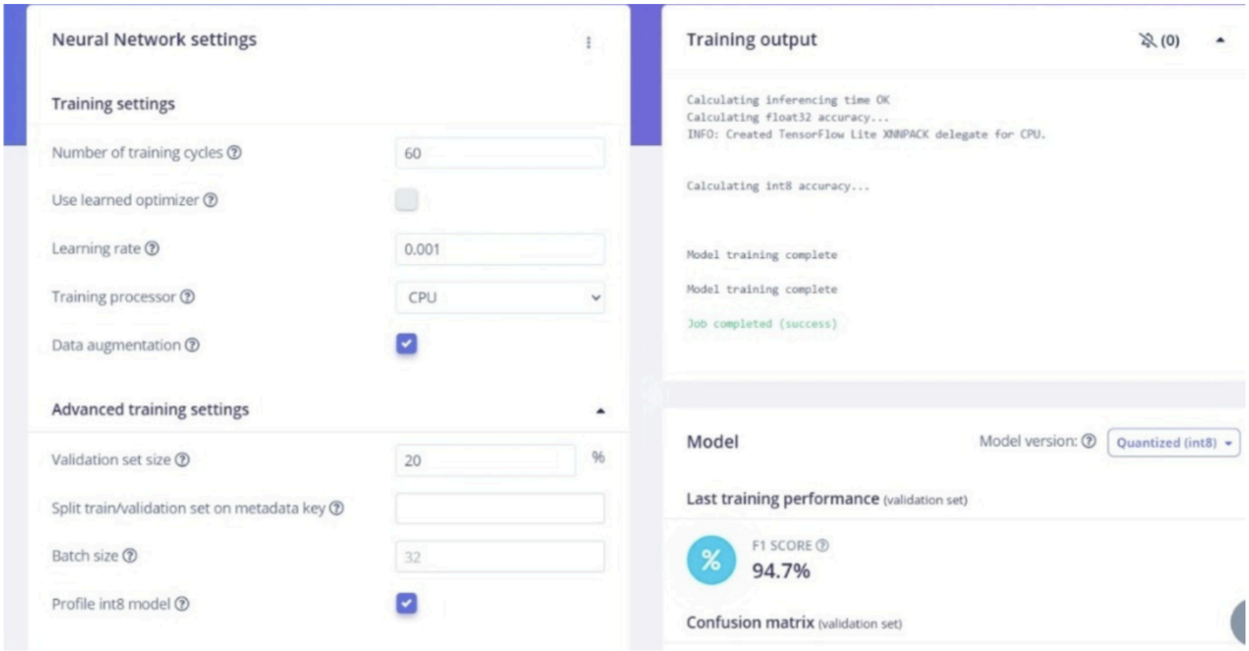
On-device performance ⓘ Engine: ⓘ EON™ Compiler (RAM optimized)

INFERENCING ... 840 ms.

PEAK RAM USA... 137.7K

FLASH USAGE 91.2K

4) Validation Result – Image



Neural Network settings

Training settings

Number of training cycles ⓘ 60

Use learned optimizer ⓘ □

Learning rate ⓘ 0.001

Training processor ⓘ CPU

Data augmentation ⓘ ☑

Advanced training settings

Validation set size ⓘ 20 %

Split train/validation set on metadata key ⓘ

Batch size ⓘ 32

Profile int8 model ⓘ ☑

Training output (0)

```
Calculating inferencing time OK
Calculating float32 accuracy...
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.


Calculating int8 accuracy...



Model training complete

Model training complete

Job completed (success)
```

Model

Model version: ⓘ Quantized (int8)

Last training performance (validation set)

F1 SCORE ⓘ 94.7%

Confusion matrix (validation set)