Name: Sudip Satish Konde
Class: TY-15 AIEC
Roll No: 2223118
SUB: ECL

Experiment No.9

Title: The Object Detection using Camera on Edge Computing Devices

Objective:

Build a project to detect an object using Edge Computing

Tasks:

- Generate the dataset for customized object
- Configure Edge Impulse for Object Detection
- Building and Training a Model
- Deploy on Edge Computing Device

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.

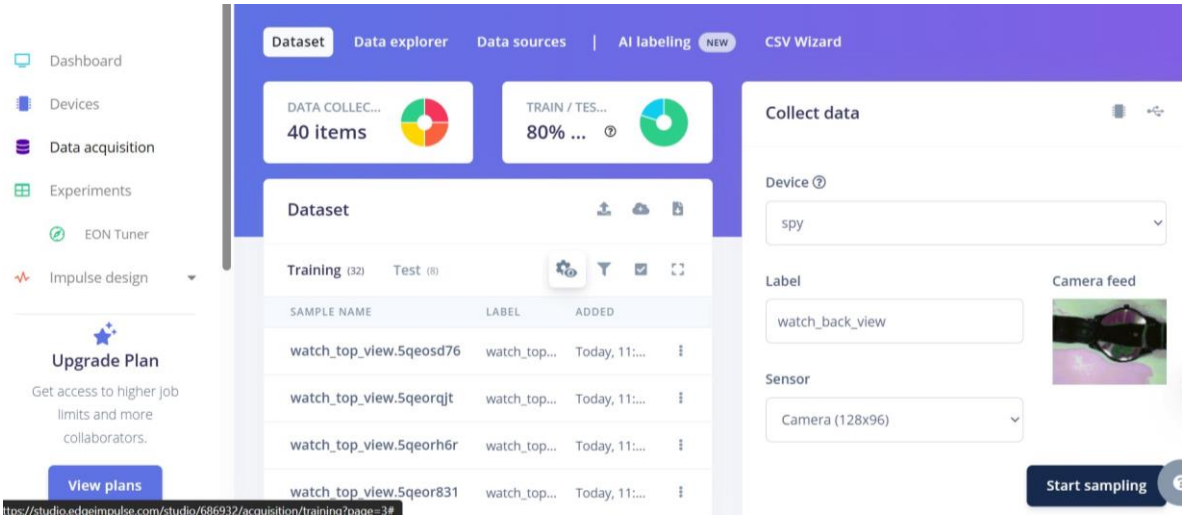
Select the deployment method that suits your edge device (e.g., Arduino library, WebAssembly, container, etc.).

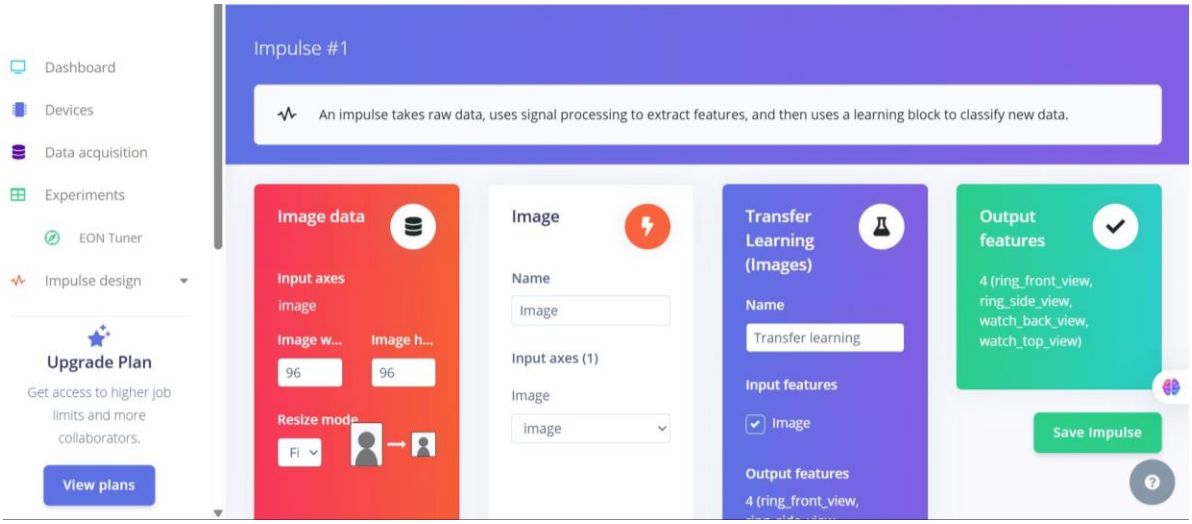
9.Run Inference:

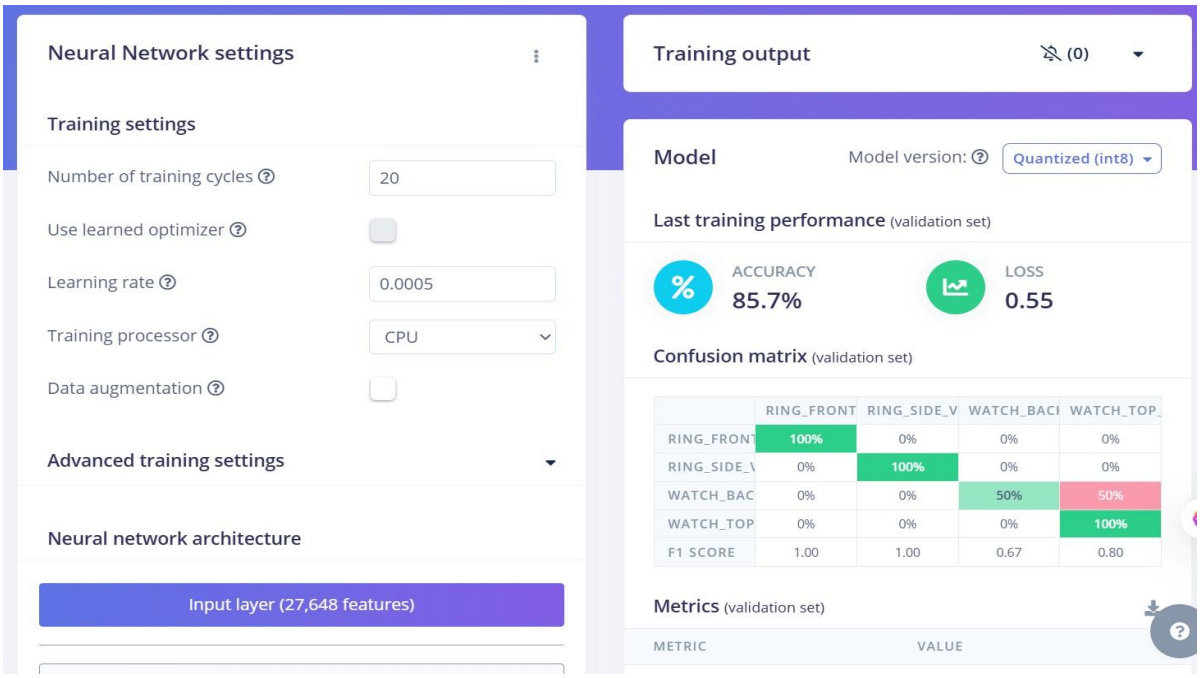
With the model deployed, run inference on the edge device to see it classifying data in real-time.

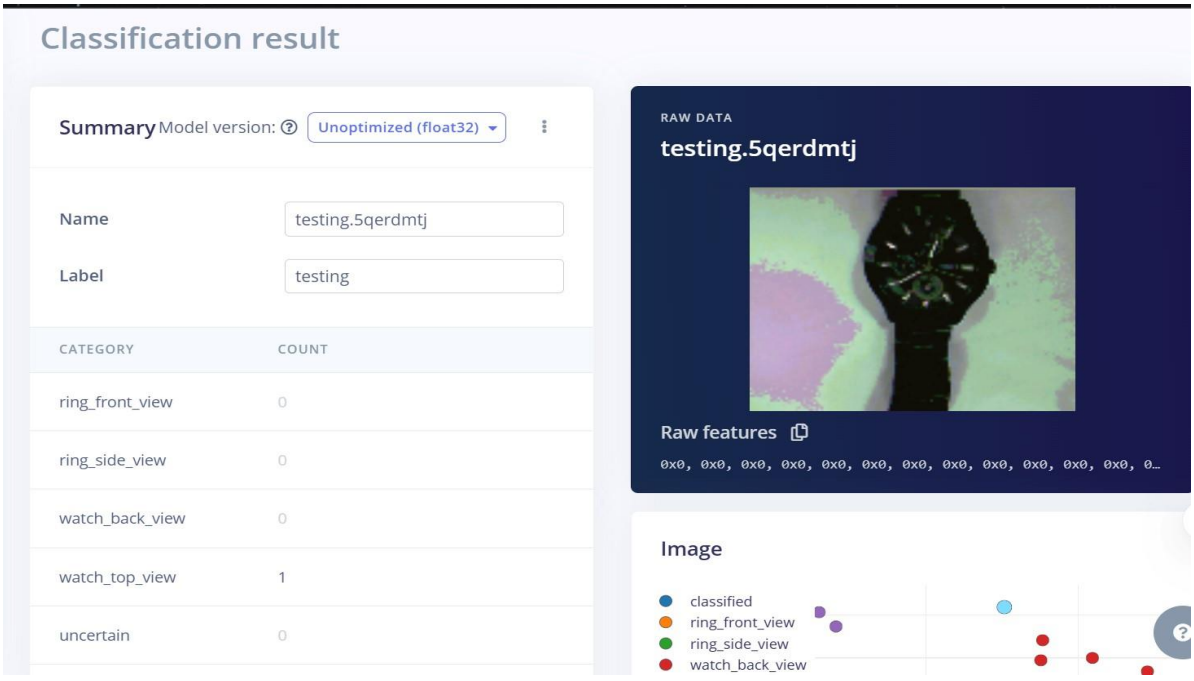
10.Monitor:

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




Dashboard
Devices
Data acquisition
Experiments
EON Tuner
Impulse design
Upgrade Plan
Get access to higher job limits and more collaborators.
View plans
Impulse #1
An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.
Image data
Input axes
image
Image w...
Image h...
96
96
Resize mode
Image
Name
Image
Input axes (1)
Image
image
Transfer Learning (Images)
Name
Transfer learning
Input features
Image
Output features
4 (ring_front_view,
Output features
4 (ring_front_view, ring_side_view, watch_back_view, watch_top_view)
Save Impulse


Neural Network settings
Training settings
Number of training cycles
20
Use learned optimizer
Learning rate
0.0005
Training processor
CPU
Data augmentation
Advanced training settings
Neural network architecture
Input layer (27,648 features)
Training output
(0)
Model
Model version:
Quantized (int8)
Last training performance (validation set)
ACCURACY
85.7%
LOSS
0.55
Confusion matrix (validation set)
RING_FRONT
RING_SIDE_V
WATCH_BACI
WATCH_TOP_
RING_FRONT 100% 0% 0% 0%
RING_SIDE_V 0% 100% 0% 0%
WATCH_BAC 0% 0% 50% 50%
WATCH_TOP 0% 0% 0% 100%
F1 SCORE 1.00 1.00 0.67 0.80
Metrics (validation set)
METRIC
VALUE


Classification result
Summary
Model version:
Unoptimized (float32)
Name
testing.5qerdmtj
Label
testing
CATEGORY
COUNT
ring_front_view 0
ring_side_view 0
watch_back_view 0
watch_top_view 1
uncertain 0
RAW DATA
testing.5qerdmtj
Raw features
0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0x0, 0…
Image
classified
ring_front_view
ring_side_view
watch_back_view

Python code:

```
import edgeimpulse_linux
import cv2
import numpy as np

# Load the trained Edge Impulse model
model = edgeimpulse_linux.EimModel("models/object_detector.eim")  # Example path

# Initialize the camera (use 0 for default webcam)
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Error: Could not access the camera.")
    exit()

print("Running Inference...")

# Inference loop (run inference on live video feed)
while True:
    ret, frame = cap.read()
    if not ret:
        print("Error: Failed to capture image.")
        break

    # Resize and preprocess image to match model input shape
    resized_frame = cv2.resize(frame, (model.input_shape[1], model.input_shape[0]))  # (width, height)
    normalized_frame = resized_frame.astype(np.float32) / 255.0  # Normalize to [0,1] range

    # Run inference using the Edge Impulse model
    result = model.classify(normalized_frame)

    # Extract label and confidence
    label = result['label']
    confidence = result['confidence']

    # Print the result
    print(f"Predicted Class: {label} with Confidence: {confidence:.2f}")
```

```
    # Display the result on the video frame
    cv2.putText(frame, f"Class: {label} ({confidence:.2f})",
                (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2)
    cv2.imshow("Live Object Detection", frame)

    # Exit the loop when 'q' is pressed
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Clean up
cap.release()
cv2.destroyAllWindows()
```

Conclusion

This project successfully demonstrates real-time object detection using an Edge Impulse-trained model deployed on an edge device. By integrating OpenCV for video capture and preprocessing with the Edge Impulse Linux SDK for inference, we achieved low-latency, on-device classification without relying on cloud services. The system accurately identified objects in live video feeds, making it suitable for applications like smart surveillance, industrial automation, and embedded AI. This approach highlights the potential of edge AI in building responsive, energy-efficient, and privacy-preserving machine learning solutions.