LAB TASK 2

K K ISHVITHA SHREE

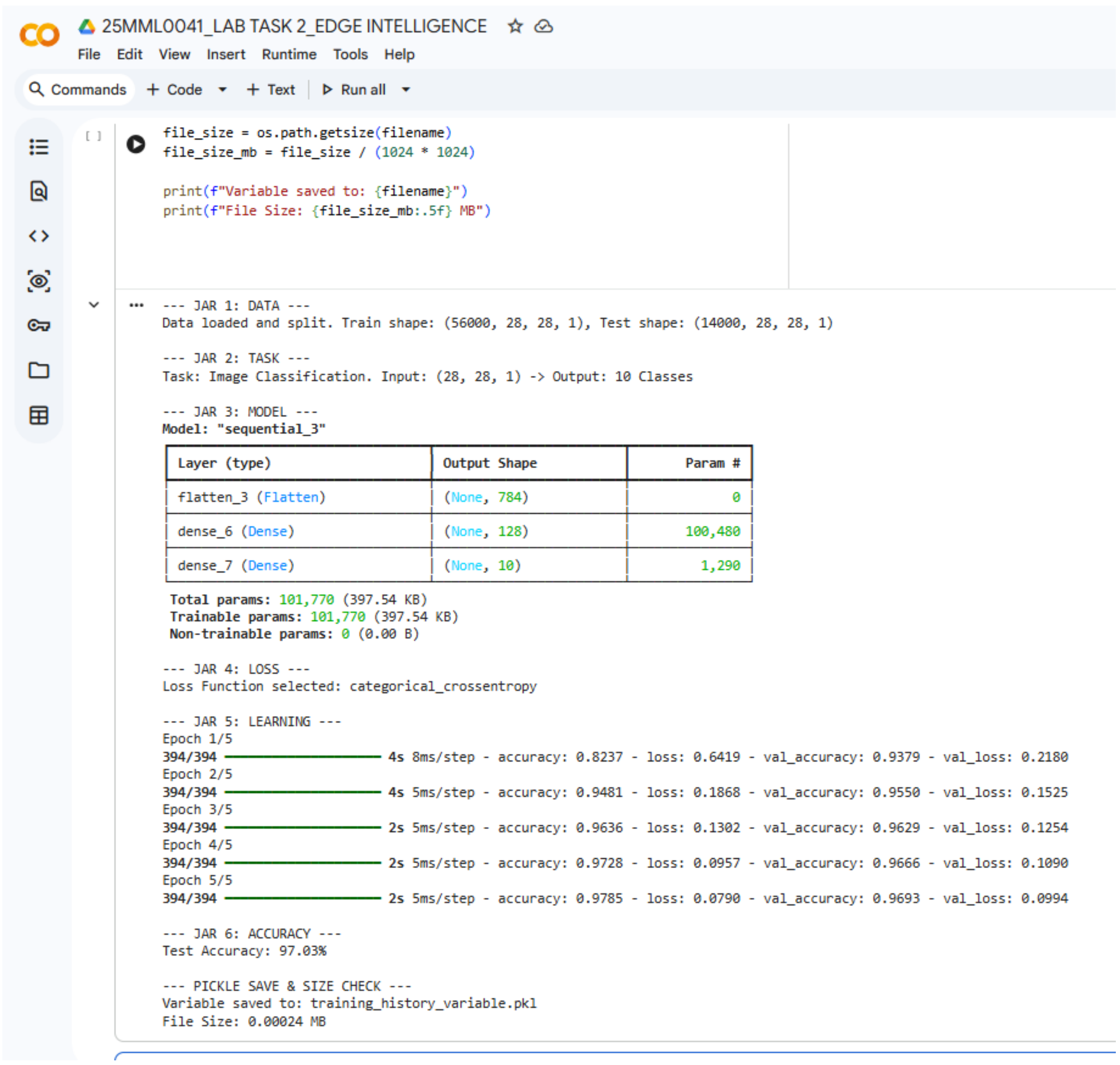
25MML0041

Q1. MNIST Dataset

Code with arguments:

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
import pickle
import os

# ==========================================
# JAR 1: DATA
# Purpose: Load, Preprocess, and Split Data
# ==========================================
print("--- JAR 1: DATA ---")
# Load existing MNIST dataset
(X_full, y_full), (X_test_orig, y_test_orig) =
keras.datasets.mnist.load_data()

# Combine to demonstrate custom splitting
X_total = np.concatenate((X_full, X_test_orig))
y_total = np.concatenate((y_full, y_test_orig))

# Preprocessing: Normalize images to [0, 1]
X_total = X_total.astype("float32") / 255.0
X_total = np.expand_dims(X_total, -1) # Shape: (70000, 28, 28, 1)

# ARGUMENT FOR SPLITTING:
# We use 'stratify=y_total' to ensure the Train and Test sets have the
# exact same proportion of digits (0-9). This prevents bias.
X_train, X_test, y_train, y_test = train_test_split(
    X_total, y_total,
    test_size=0.2,
    random_state=42,
    stratify=y_total  # <--- STRATIFIED SPLIT
)

# One-hot encode labels
y_train_cat = keras.utils.to_categorical(y_train, 10)
y_test_cat = keras.utils.to_categorical(y_test, 10)

print(f"Data loaded and split. Train shape: {X_train.shape}, Test
shape: {X_test.shape}")

# ==========================================
# JAR 2: TASK
# Purpose: Define the Input and Output goal
# ==========================================
print("\n--- JAR 2: TASK ---")
# Input: 28x28 grayscale images
# Output: Probability distribution over 10 classes (Digits 0-9)
input_shape = (28, 28, 1)
num_classes = 10
print(f"Task: Image Classification. Input: {input_shape} -> Output:
{num_classes} Classes")

# ==========================================
# JAR 3: MODEL
# Purpose: The Mathematical Architecture
# ==========================================
print("\n--- JAR 3: MODEL ---")
model = keras.Sequential([
    keras.Input(shape=input_shape),
    layers.Flatten(),              # Flatten 2D image to 1D vector
    layers.Dense(128, activation="relu"), # Hidden layer
    layers.Dense(num_classes, activation="softmax") # Output layer
])
model.summary()

# ==========================================
# JAR 4: LOSS
# Purpose: Error Function
# ==========================================
print("\n--- JAR 4: LOSS ---")
# We use Categorical Crossentropy because our targets are one-hot
encoded.
loss_fn = "categorical_crossentropy"
print(f"Loss Function selected: {loss_fn}")

# ==========================================
# JAR 5: LEARNING
# Purpose: Optimization Algorithm to minimize Loss
# ==========================================
print("\n--- JAR 5: LEARNING ---")
# Optimizer: Adam (Adaptive Moment Estimation)
optimizer_algo = "adam"
model.compile(loss=loss_fn, optimizer=optimizer_algo,
metrics=["accuracy"])

# Training the model
batch_size = 128
epochs = 5
history = model.fit(X_train, y_train_cat, batch_size=batch_size,
epochs=epochs, validation_split=0.1)

# ==========================================
# JAR 6: ACCURACY (Evaluation)
# Purpose: Test on unseen data
# ==========================================
print("\n--- JAR 6: ACCURACY ---")
test_loss, test_acc = model.evaluate(X_test, y_test_cat, verbose=0)
print(f"Test Accuracy: {test_acc * 100:.2f}%")

# ==========================================
# PICKLE & SIZE CHECK
# ==========================================
print("\n--- PICKLE SAVE & SIZE CHECK ---")

# Save the training history variable using Pickle
filename = "training_history_variable.pkl"
with open(filename, "wb") as f:
    pickle.dump(history.history, f)

# Check size
file_size = os.path.getsize(filename)
file_size_mb = file_size / (1024 * 1024)

print(f"Variable saved to: {filename}")
print(f"File Size: {file_size_mb:.5f} MB")
```

Output Screenshot:



25MML0041_LAB TASK 2_EDGE INTELLIGENCE
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all

```
file_size = os.path.getsize(filename)
file_size_mb = file_size / (1024 * 1024)

print(f"Variable saved to: {filename}")
print(f"File Size: {file_size_mb:.5f} MB")
```

```
--- JAR 1: DATA ---
Data loaded and split. Train shape: (56000, 28, 28, 1), Test shape: (14000, 28, 28, 1)

--- JAR 2: TASK ---
Task: Image Classification. Input: (28, 28, 1) -> Output: 10 Classes

--- JAR 3: MODEL ---
Model: "sequential_3"
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━┓
┃ Layer (type)                       ┃ Output Shape              ┃        Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━┩
│ flatten_3 (Flatten)                │ (None, 784)               │              0 │
├────────────────────────────────────┼───────────────────────────┼────────────────┤
│ dense_6 (Dense)                    │ (None, 128)               │        100,480 │
├────────────────────────────────────┼───────────────────────────┼────────────────┤
│ dense_7 (Dense)                    │ (None, 10)                │          1,290 │
└────────────────────────────────────┴───────────────────────────┴────────────────┘
 Total params: 101,770 (397.54 KB)
 Trainable params: 101,770 (397.54 KB)
 Non-trainable params: 0 (0.00 B)

--- JAR 4: LOSS ---
Loss Function selected: categorical_crossentropy

--- JAR 5: LEARNING ---
Epoch 1/5
394/394 ━━━━━━━━━━━━━━━━━━━━ 4s 8ms/step - accuracy: 0.8237 - loss: 0.6419 - val_accuracy: 0.9379 - val_loss: 0.2180
Epoch 2/5
394/394 ━━━━━━━━━━━━━━━━━━━━ 4s 5ms/step - accuracy: 0.9481 - loss: 0.1868 - val_accuracy: 0.9550 - val_loss: 0.1525
Epoch 3/5
394/394 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.9636 - loss: 0.1302 - val_accuracy: 0.9629 - val_loss: 0.1254
Epoch 4/5
394/394 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.9728 - loss: 0.0957 - val_accuracy: 0.9666 - val_loss: 0.1090
Epoch 5/5
394/394 ━━━━━━━━━━━━━━━━━━━━ 2s 5ms/step - accuracy: 0.9785 - loss: 0.0790 - val_accuracy: 0.9693 - val_loss: 0.0994

--- JAR 6: ACCURACY ---
Test Accuracy: 97.03%

--- PICKLE SAVE & SIZE CHECK ---
Variable saved to: training_history_variable.pkl
File Size: 0.00024 MB
```

Hyperparameters & Training Explanation:

- **Epochs (5):** The number of times the model sees the entire dataset. We chose 5 because MNIST is simple and converges quickly; too many epochs might lead to overfitting.
- **Batch Size (128):** The model doesn't update weights after every single image (which is slow/noisy) or after the whole dataset (which requires huge memory). It updates after every 128 images. This is a balance between speed and stability.
- **Learning Rate (Adam default):** This controls how big of a "step" the optimizer takes to correct errors. We used the Adam optimizer because it automatically adjusts this rate during training, making it efficient for beginners.
- ***6 JARS:***
    - Data: The fuel (images).
    - Task: The goal (classify digits).
    - Model: The math (Neural Network layers).
    - Loss: The error metric (difference between prediction and actual).
    - Learning: The correction mechanism (Optimizer/Backpropagation).
    - Accuracy: The final report card (Evaluation on test set).
- **Pickle Size**
  The pickle file saves the history variable (accuracy/loss logs).

Q2. Data Acquisition

Objective

To understand and implement data acquisition and dataset preparation using the Edge Impulse platform for object classification, including data collection, labeling, and train–test split.

Platform Used

- Edge Impulse Studio (Web Platform)
- Target device: Raspberry Pi 5 (configured in Edge Impulse)[As we can see in the edge impulse target device in upper right side]
- Data type: Image data

In this lab, the objective was to collect real-world image data using Edge Impulse, label the data correctly, and prepare it for machine learning by splitting it into training and testing datasets.

Objects Collected

The following 6 object classes were collected:

1. Desktop
2. Laptop

3. Chair

4. Keyboard

5. Mouse

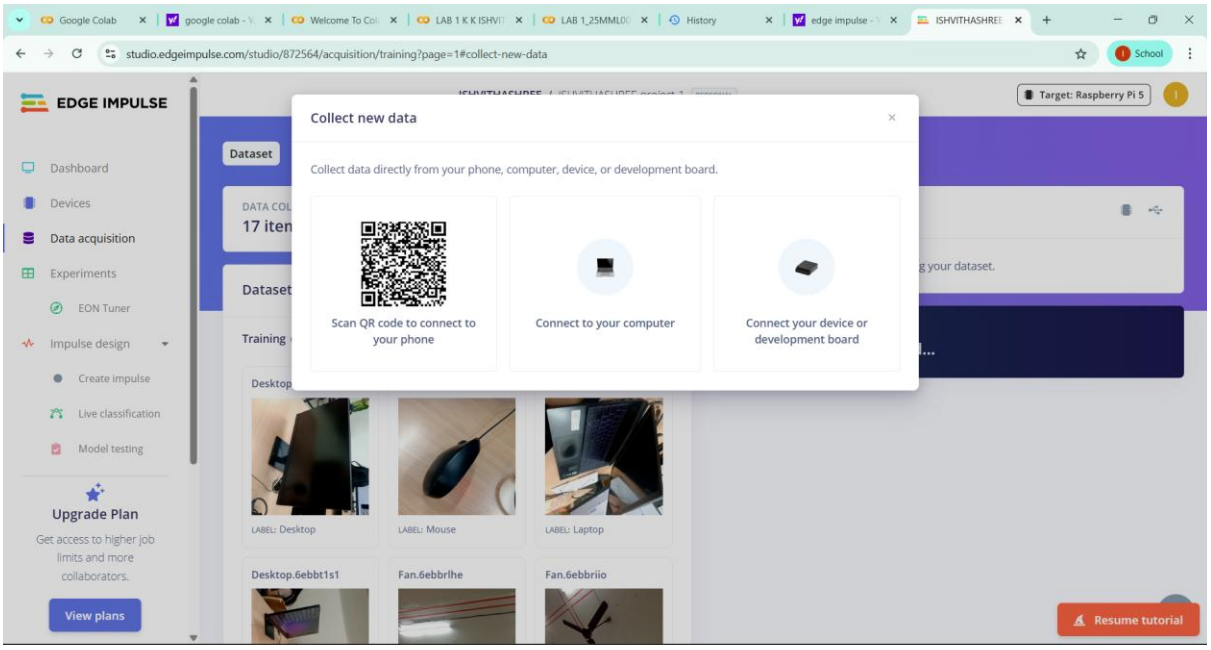
6. Fan

Each object was captured from multiple angles and orientations to improve dataset diversity.

Steps :

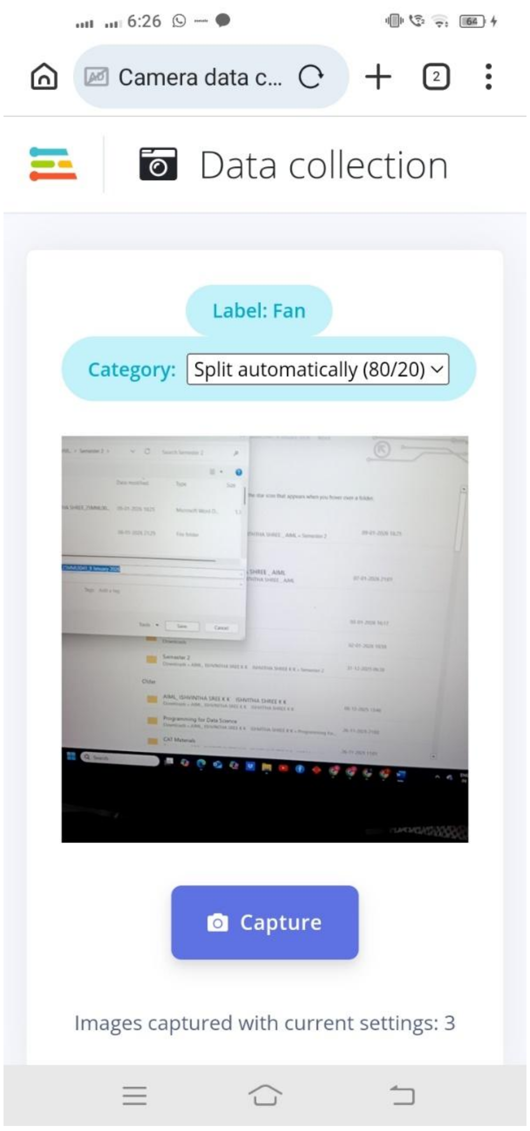
Step 1: Project Setup

- Created a new project in Edge Impulse Studio.
- Selected the target device and navigated to the Data Acquisition section and connect to phone.


studio.edgeimpulse.com/studio/872564/acquisition/training?page=1#collect-new-data

Collect new data

Collect data directly from your phone, computer, device, or development board.


Data collection

Label: Fan

Category: Split automatically (80/20)

Images captured with current settings: 3

Step 2: Data Collection

- Uploaded image samples for each object class.
- Multiple images were captured for each object under different viewpoints.
- Each image was manually labeled with the correct class name.

Step 3: Dataset Organization

- A total of 30 image samples were collected.
- The dataset was automatically divided into: Training set: 80% , Testing set: 20%
- This split ensures proper model training and unbiased evaluation.

Step 4: Verification Using Data Explorer

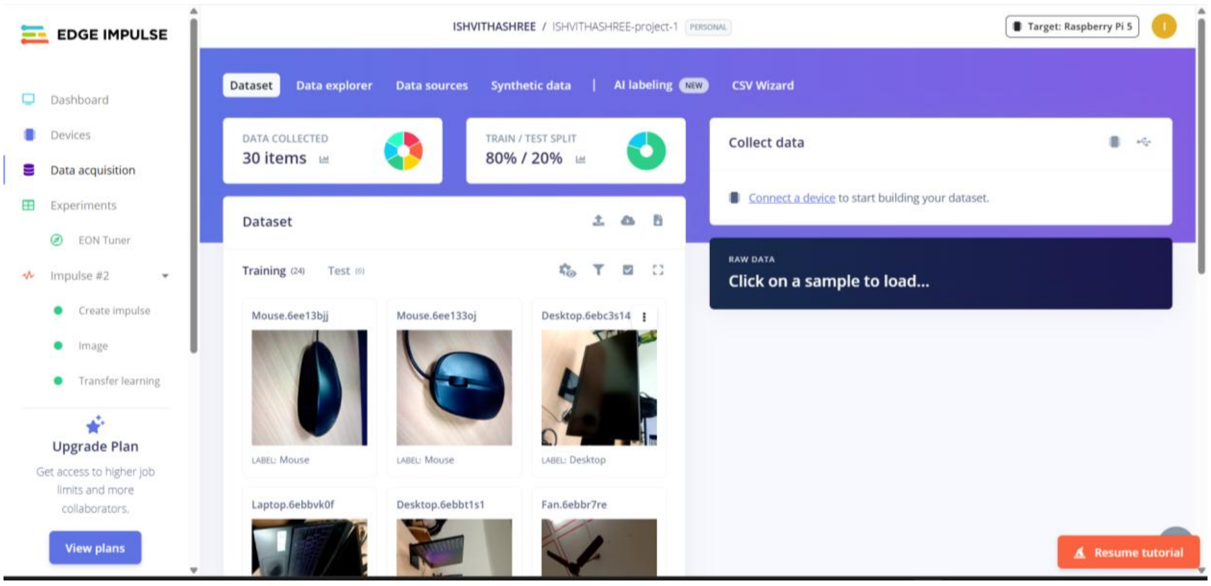
- Verified uploaded images and labels using the Dataset view.
- Confirmed correct separation of training and testing samples.
- Ensured no label mismatch or data duplication.

Observation

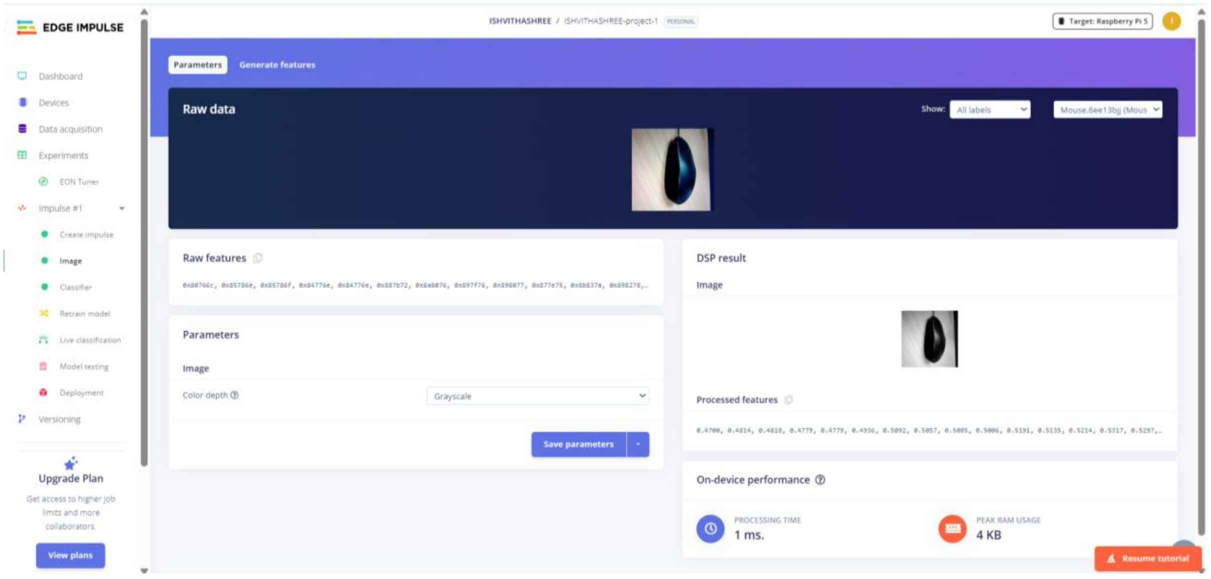
- Edge Impulse provides an intuitive interface for data acquisition and labeling.
- Collecting images from different angles improves model generalization.
- Proper train–test split is essential for reliable performance evaluation.

Result

A labeled image dataset containing 6 object classes was successfully created and organized in Edge Impulse, ready for feature extraction and model training.



Edge Impulse #1 → Classification:



ISHVITHASHREE / ISHVITHASHREE-project-1 PERSONAL

Target: Raspberry Pi 5

Parameters | Generate features

Raw data

Show: All labels | Mouse.6ee13bjj (Mous

Raw features

```
0x80766c, 0x85786e, 0x85786f, 0x84776e, 0x84776e, 0x887b72, 0x8a8076, 0x897f76, 0x898077, 0x877e75, 0x8b837a, 0x898278,…
```

Parameters

Image

Color depth: Grayscale

Save parameters

DSP result

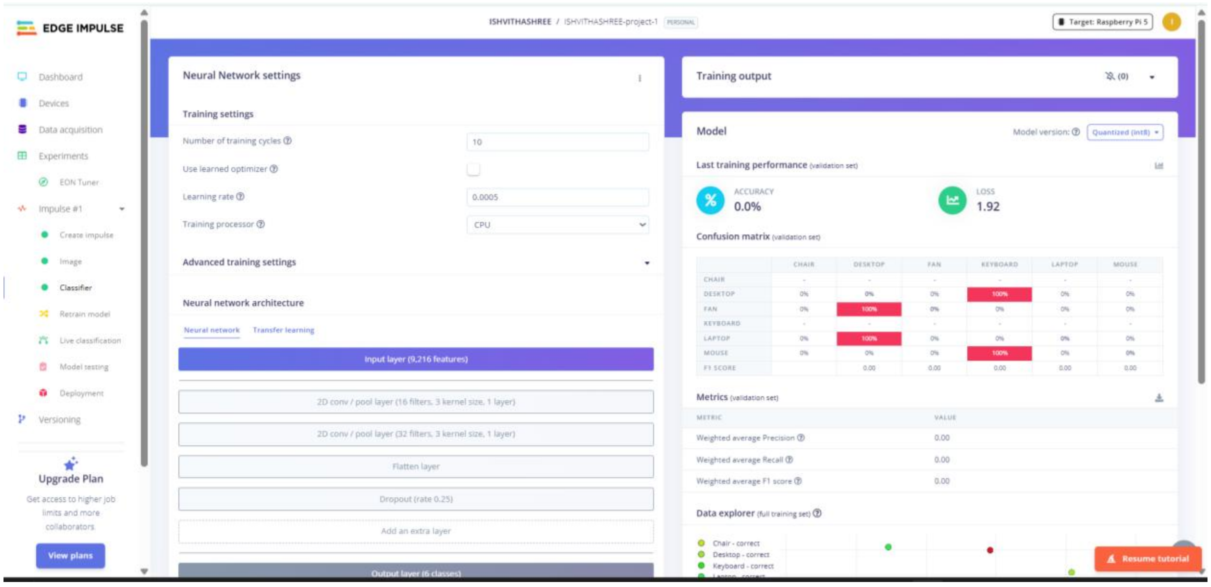
Image

Processed features

```
0.4700, 0.4814, 0.4818, 0.4779, 0.4779, 0.4936, 0.5092, 0.5057, 0.5085, 0.5006, 0.5191, 0.5135, 0.5214, 0.5317, 0.5297,…
```

On-device performance

PROCESSING TIME	1 ms.
PEAK RAM USAGE	4 KB



ISHVITHASHREE / ISHVITHASHREE-project-1 PERSONAL

Target: Raspberry Pi 5

Neural Network settings

Training settings

Number of training cycles	10
Use learned optimizer	
Learning rate	0.0005
Training processor	CPU

Advanced training settings

Neural network architecture

Neural network | Transfer learning

Input layer (9,216 features)

2D conv / pool layer (16 filters, 3 kernel size, 1 layer)

2D conv / pool layer (32 filters, 3 kernel size, 1 layer)

Flatten layer

Dropout (rate 0.25)

Add an extra layer

Output layer (6 classes)

Training output

Model Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY	0.0%
LOSS	1.92

Confusion matrix (validation set)

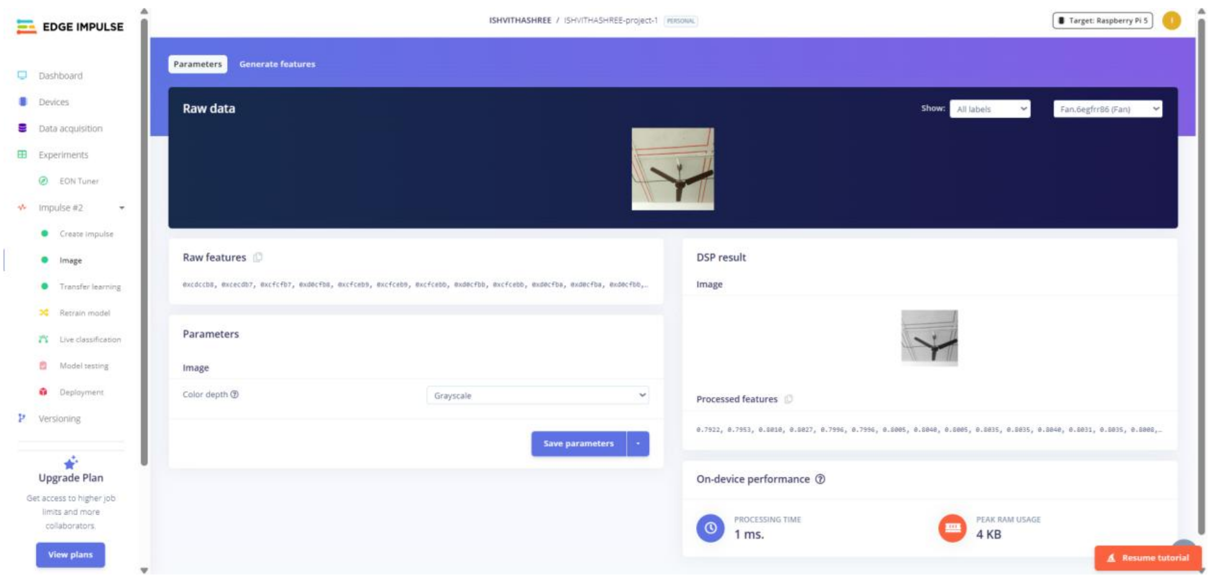
	CHAIR	DESKTOP	FAN	KEYBOARD	LAPTOP	MOUSE
CHAIR	-	-	-	-	-	-
DESKTOP	0%	0%	0%	100%	0%	0%
FAN	0%	100%	0%	0%	0%	0%
KEYBOARD	-	-	-	-	-	-
LAPTOP	0%	100%	0%	0%	0%	0%
MOUSE	0%	0%	0%	100%	0%	0%
F1 SCORE		0.00	0.00	0.00	0.00	0.00

Metrics (validation set)

METRIC	VALUE
Weighted average Precision	0.00
Weighted average Recall	0.00
Weighted average F1 score	0.00

Data explorer (full training set)

Edge Impulse #2 → Transfer Learning:



Raw data

Show: All labels | Fan.6egfrr86 (Fan)

Raw features

```
0xcdccb8, 0xcecdb7, 0xcfcfb7, 0xd0cfb8, 0xcfceb9, 0xcfceb9, 0xcfcebb, 0xd0cfbb, 0xcfcebb, 0xd0cfba, 0xd0cfba, 0xd0cfbb,…
```

Parameters

Image

Color depth: Grayscale

DSP result

Image

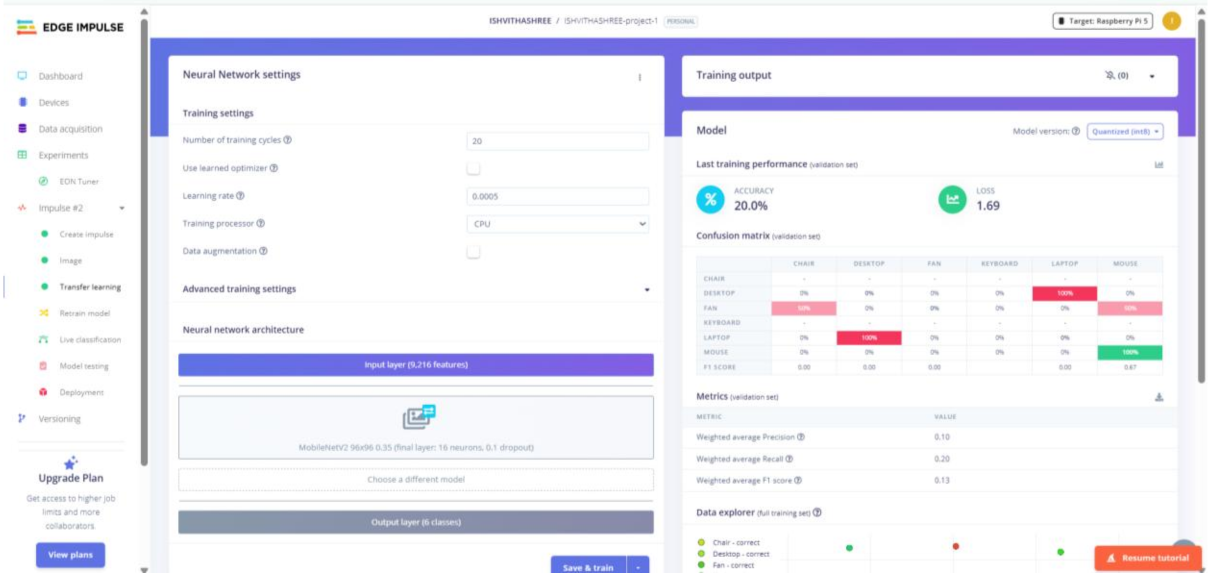
Processed features

```
0.7922, 0.7953, 0.8010, 0.8027, 0.7996, 0.7996, 0.8005, 0.8040, 0.8005, 0.8035, 0.8035, 0.8040, 0.8031, 0.8035, 0.8008,…
```

On-device performance

PROCESSING TIME 1 ms.

PEAK RAM USAGE 4 KB



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 (9,216 features)

MobileNetV2 96x96 0.35 (final layer: 16 neurons, 0.1 dropout)

Output layer (6 classes)

Training output (0)

Model

Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY 20.0%

LOSS 1.69

Confusion matrix (validation set)

	CHAIR	DESKTOP	FAN	KEYBOARD	LAPTOP	MOUSE
CHAIR	-	-	-	-	-	-
DESKTOP	0%	0%	0%	0%	100%	0%
FAN	50%	0%	0%	0%	0%	50%
KEYBOARD	-	-	-	-	-	-
LAPTOP	0%	100%	0%	0%	0%	0%
MOUSE	0%	0%	0%	0%	0%	100%
F1 SCORE	0.00	0.00	0.00		0.00	0.67

Metrics (validation set)

METRIC	VALUE
Weighted average Precision	0.10
Weighted average Recall	0.20
Weighted average F1 score	0.13

Data explorer (full training set)

● Chair - correct
● Desktop - correct
● Fan - correct

Conclusion

This lab helped in understanding the end-to-end data acquisition workflow using Edge Impulse, including dataset creation, labeling, and preparation for edge-based machine learning applications.

Using Edge Impulse, I collected and labeled image data for six objects, organized them into training and testing sets.